BAB I

PENDAHULUAN

1.1 Latar Belakang

Teknologi informasi pada awalnya dimulai dari ditemukannya bahasa sebagai alat penyampaian suatu informasi tertentu. Bahasa memungkinkan seseorang untuk dapat memahami informasi yang akan disampaikan oleh orang lain. Seiring dengan pertumbuhan tingkat kecerdasan manusia dan kebutuhan manusia akan informasi yang tepat, cermat, akurat dan efisien dan secepat mungkin, perkembangan teknologi informasi pun mengalami peningkatan yang signifikan. Kebutuhan akan informasi yang semakin besar tersebut kini dihadirkan dalam suatu media yang dikenal dengan sebutan Internet.

Media informasi konvensional seperti koran, buku dan majalah sudah mulai dirasakan tidak efektif dan efisien lagi. Penyampaian informasi melalui media konvensional tersebut sudah mulai beralih ketingkat modern yang bersifat digital dan disebut sebagai Internet.

Internet saat ini mulai menjadi sebuah kebutuhan bagi sebagian masyarakat Indonesia. Biaya yang relatif murah dan kebutuhan akan akses Internet serta jaringan telekomunikasi yang luas, serta berbagai macam koneksi yang ditawarkan provider Internet merupakan penyebab berkembangnya pengguna Internet di Indonesia, sehingga pertumbuhan pengguna akses Internet di Indonesia tumbuh secara signifikan,

Semakin besarnya jumlah pengguna Internet yang berbanding lurus dengan kebutuhan bandwidth yang juga semakin besar dan semakin cepat, oleh karena itu diperlukan suatu sistem yang dapat menjadikan penggunaan bandwidth lebih efektif dan efisien. Hal ini dapat disiasati dengan penggunaan proxy server pada suatu jaringan komputer yang terhubung dengan Internet.

1.2 Rumusan Masalah

Permasalahan yang dibahas dalam tugas akhir ini adalah bagaimana cara membangun dan memaksimalkan suatu Proxy Server yang dapat diaplikasikan pada topologi yang dapat memaksimalkan kinerja dari Proxy Server dan relatif lebih baik dari topologi yang biasa digunakan pada saat ini yaitu Proxy Server berada setelah Bandwidth Limiter dari sisi Network Klien.

1.3 Batasan Masalah

Batasan masalah dalam penelitian tugas akhir ini adalah sebagai berikut:

1. Sistem Operasi yang digunakan adalah Linux Slackware.
2. Kernel yang digunakan adalah kernel Linux 2.6.
3. Tool yang digunakan adalah :
   a. Cttproxy 2.6.
   b. Iptables 1.3.
   c. Squid 2.7 STABLE.

1.4 Tujuan Penelitian

Penelitian dalam tugas akhir "TProxy Transparant Server Berbasis Linux" ini dibuat dengan tujuan untuk membangun suatu Proxy Server yang dapat memberikan keluaran TPROXY sehingga dapat diaplikasikan dalam topologi jaringan dengan Proxy Server berada diantara Network Klien dan Bandwidth Limiter pada jaringan, sehingga dapat meningkatkan kinerja dari jaringan Internet yang menggunakan Bandwidth Limiter dan Proxy Server.

1.5 Manfaat Penelitian

Dengan adanya penelitian yang dilakukan ini diharapkan dapat memberi manfaat antara lain :

1. Mempopulerkan TProxy Server beserta topologinya sehingga dapat diterapkan pada jaringan Internet.

2. Dengan adanya TProxy Server, diharapkan dapat meningkatkan kinerja dari jaringan Internet.

3. Mempermudah network administrator pada saat maintenance, troubleshooting dan konfigurasi.

1.6 Metodologi Penelitian

Metode-metode yang digunakan dalam penelitian ini adalah :

1.6.1 Metode Pengumpulan Literatur

Metode pengumpulan literatur adalah metode yang digunakan untuk mengumpulkan data yang diperlukan dalam penelitian, metode yang digunakan adalah :

a. Studi pustaka

Metode studi pustaka, yaitu metode pengumpulan literatur yang dilakukan melalui buku-buku yang berkaitan dengan penelitian yang dilakukan dalam rangka mencari konsep dasar dari penelitian ini.

b. Browsing

Metode browsing, merupakan metode pengumpulan literatur melalui Internet, yang dilakukan dengan tujuan pemilihan perangkat lunak yang akan diimplementasikan sebagai TProxy, seta pencarian referensi mengenai tahapan dalam implementasi TProxy, termasuk kebutuhan hardware serta software pendukung lainnya agar system berjalan dengan baik . dan yang terpenting adalah metode browsing digunakan sebagai jalan keluar dalam menyelesaikan masalah yang ditemui selama melakukan penelitian.

1.6.2 Metode Implementasi Sistem

Metodologi disusun berdasarkan hasil perolehan dari metodologi pengumpulan literatur yang meliputi :

a. Perencanaan arsitektur jaringan komputer

Tahap ini merupakan tahap perancangan arsitektur jaringan komputer yang akan digunakan untuk membangun jaringan dengan sistem TProxy.

b. Pengadaan hardware

Tahapan ini merupakan tahap pengadaan perangkat keras yang meliputi komputer untuk TProxy server, router box yang akan digunakan sebagai bandwidth shaping, komputer klien, kabel UTP dan konektor RJ45.

c. **Instalasi dan konfigurasi software**

Tahapan ini merupakan tahapan instalasi software pada komputer yang akan dijadikan TProxy server, konfigurasi standar pada bandwidth limiter, serta komputer yang akan di jadikan client, kemudian dilnjutkan mengkonfigurasi software pada TProxy server.

d. **Pengujian**

Setelah konfigurasi selesai dilakukan, tahap selanjutnya adalah pengujian kinerja TProxy trsebut dalam keadaan nyata dimana sistem tersebut akan diimplementasikan.

1.7 Sistematika Penulisan

Pada penyusunan tugas akhir ini terdiri dari bagian pokok laporan, yaitu bagian pendahuluan yang memuat halaman judul, lembar pengesahan, halaman persembahan, halaman motto, kata pengantar, abstraksi, daftar isi, daftar tabel, dan daftar gambar.

Bagian tubuh laporan berisi laporan tugas akhir yang dipisahkan menurut bab-bab sebagai berikut:

BAB I PENDAHULUAN

Membahas teori tentang penyusunan sistem TProxy Server, yang meliputi latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian serta sistematika penelitian.

BAB II LANDASAN TEORI

Membahas dasar-dasar teori yang akan digunakan dalam merancang dan membangun aplikasi TProxy Server berbasis linux beserta tinjauan pustaka yang digunakan.

BAB III METODOLOGI

Memuat uraian tentang metode-metode TProxy Server yang berupa metode analisis, analisis masalah dan hasil analisis yang mencakup kebutuhan perencanaan arsitektur jaringan komputer, pengadaan perangkat keras, instalasi dan konfigurasi software, serta pengujian sistem.

BAB IV HASIL DAN PEMBAHASAN

Memuat dokumentasi hasil pengujian terhadap aplikasi TProxy Server, yang dibandingkan kesesuaiannya dengan analisis dan perancangan yang telah dilakukan sebelumnya.

BAB V SIMPULAN DAN SARAN

Bab ini berisi kesimpulan yang diambil dari hasil pembahasan TProxy Server pada bab-bab sebelumnya serta saran untuk perbaikan laporan yang telah dibuat.

BAB II

LANDASAN TEORI

2.1 Squid

Squid adalah suatu perangkat yang digunakan sebagai *proxy cache* untuk web yang mendukung HTTP, HTTPS, FTP, dan banyak lagi. Hal ini dapat menghemat *bandwidth* dan meningkatkan waktu respon dengan caching, yaitu menggunakan kembali halaman web yang sering diminta. Squid memiliki kontrol akses yang luas dan dapat membangun server akselerator besar. Sistem ini dapat berjalan pada system operasi yang biasa digunakan, termasuk Windows dan berlisensi dibawah GNU GPL.[squid-cache,Squid]

Squid memiliki banyak jenis penggunaan, mulai dari mempercepat server web dengan melakukan *caching* permintaan yang berulang-ulang, *caching* DNS, caching situs web, dan caching pencarian komputer di dalam jaringan untuk sekelompok komputer yang menggunakan sumber daya jaringan yang sama, hingga pada membantu keamanan dengan cara melakukan penyaringan (*filter*) lalu lintas. Meskipun seringnya digunakan untuk protokol HTTP dan FTP, Squid juga menawarkan dukungan terbatas untuk beberapa protocol lainnya termasuk Transport Layer Security (TLS), Secure Socket Layer (SSL), Internet Gopher, dan HTTPS. Versi Squid 3.1 mencakup dukungan protokol IPV6 dan Internet Content Adaptation Protocol (ICAP).

Squid pada awalnya dikembangkan oleh Duane Wessels sebagai Harvest object cache, yang merupakan bagian dari proyek Harvest yang dikembangkan di University of Colorado at Boulder. Pekerjaan selanjutnya dilakukan hingga selesai di University of California, San Diego dan didanai melalui National Science Foundation. Squid kini hampir secara eksklusif dikembangkan dengan cara usaha sukarela.

Squid umumnya didesain untuk berjalan di atas sistem operasi mirip UNIX, meski Squid juga bisa berjalan di atas sistem operasi Windows. Karena dirilis di bawah lisensiGNU General Public Licnese, maka Squid merupakan perangkat lunak bebas.

3.1 Proxy

Kata Proxy berasal dari bahasa latin proximus, yang berarti dekat. Proxy server adalah sebuah server yang membantu kita untuk mempertahankan privasi kita dalam mengakses Internet. Saat kita mengakses Internet menggunakan proxy, web page yang kita kunjungi tidak akan dapat melihat kita. Web page tersebut hanya akan melihat proxy yang kita gunakan saja.[Iwan Nurwijayanto,2010]

Proxy Server bertindak sebagai *gateway* terhadap dunia Internet untuk setiap komputer klien. *Proxy server* tidak terlihat oleh komputer klien, seorang pengguna yang berinteraksi dengan Internet melalui sebuah proxy server tidak akan mengetahui bahwa sebuah *proxy server* sedang menangani request yang dilakukannya kecuali pengguna tersebut menggunakan tool-tool tertentu yang dapat mendeteksi port proxy yang digunakan. *Web Server* yang menerima *request* dari *proxy server* akan menginterpretasikan *request-request* tersebut seolah-olah *request* itu datang secara langsung dari komputer klien, bukan dari *proxy server*.

Proxy server juga dapat digunakan untuk mengamankan jaringan pribadi yang dihubungkan ke sebuah jaringan publik seperti halnya Internet. *Proxy server* memiliki lebih banyak fungsi daripada *router* yang memiliki fitur *packet filtering* karena memang *proxy server* beroperasi pada level yang lebih tinggi dan memiliki kontrol yang lebih menyeluruh terhadap akses jaringan. *Proxy server* yang berfungsi sebagai sebuah agen keamanan untuk sebuah jaringan pribadi, umumnya dikenal sebagai *firewall.*

Caching merupakan sebuah cara untuk menyimpan objek-objek Internet yang diminta seperti halnya data halaman web yang bisa diakses melalui HTTP, FTP dan Gopher di dalam sebuah sistem yang lebih dekat dengan situs yang

memintanya. Beberapa penjelajah web dapat menggunakan *cache* Squid lokal untuk sebagai *server proxy HTTP*, sehingga dapat mengurangi waktu akses dan juga tentu saja konsumsi *bandwidth*. Hal ini sering berguna bagi para penyedia layanan Internet untuk meningkatkan kecepatan kepada para pelanggannya, dan LAN yang membagi saluran Internet. Karena memang bentuknya sebagai *proxy* ia berlaku sebagaimana layaknya klien, sesuai dengan permintaan klien, *web cache* bisa menyediakan anonimitas dan keamanan. Tapi, *web cache* juga bisa menjadi masalah yang signifikan bila melihat maslah privasi, karena memang ia dapat mencatat banyak data, termasuk URL yang diminta oleh klien, kapan hal itu terjadi, nama dan versi penjelajah web yang digunakan klien serta sistem operasinya, dan dari mana ia mengakses situs itu.

Selanjutnya, sebuah program klien sebagai contoh adalah penjelajah web bisa menentukan secara ekplisit *proxy server* yang digunakan bila memang hendak menggunakan proxy umumnya bagi para pelanggan ISP atau bisa juga menggunakan proxy tanpa konfigurasi ekstra, yang sering disebut sebagai *Transparent Caching*, di mana semua permintaan HTTP ke jaringan luar akan diolah oleh *proxy server* dan semua respons disimpan di dalam *cache*. Kasus kedua umumnya dilakukan di dalam perusahaan dan korporasi semua klien berada di dalam LAN yang sama dan sering memiliki masalah privasi yang disebutkan di atas.

Squid memiliki banyak fitur yang bisa membantu melakukan koneksi secara anonim, seperti memodifikasi atau mematikan beberapa *field header* tertentu dalam sebuah permintaan HTTP yang diajukan oleh klien. Saat itu terpenuhi, apa yang akan dilakukan oleh Squid adalah tergantung orang yang menangani komputer yang menjalankan Squid. Orang yang meminta halaman web melalui sebuah jaringan yang secara transparan yang menggunakan biasanya tidak mengetahui bahwa informasi semua permintaan HTTP yang mereka ajukan dicatat oleh Squid.

4.1 Kernel

Kernel adalah inti dari sistem operasi. Kernel memiliki control penuh atas segala sesuatu yang terjadi. Kernel dirancang untuk mengkoordinasikan antara komponen-komponen yang berbeda dari system operasi, seperti disk drive, jaringan, keyboard, dan menjalankan program.[linuxdig,Kernel]

Karena akses terhadap perangkat keras terbatas, sedangkan ada lebih dari satu program yang harus dilayani dalam waktu yang bersamaan, maka kernel juga bertugas untuk mengatur kapan dan berapa lama suatu program dapat menggunakan satu bagian perangkat keras tersebut. Hal tersebut dinamakan sebagai multiplexing.

Akses kepada perangkat keras secara langsung merupakan masalah yang kompleks, oleh karena itu kernel biasanya mengimplementasikan sekumpulan abstraksi hardware. Abstraksi-abstraksi tersebut merupakan sebuah cara untuk menyembunyikan kompleksitas, dan memungkinkan akses kepada perangkat keras menjadi mudah dan seragam. Sehingga abstraksi pada akhirnya memudahkan pekerjaan programer.

Untuk menjalankan sebuah komputer , tidak harus menggunakan kernel sistem operasi. Sebuah program dapat saja langsung di*load* dan dijalankan diatas mesin 'telanjang' komputer, yaitu bilamana pembuat program ingin melakukan pekerjaannya tanpa bantuan abstraksi perangkat keras atau bantuan sistem operasi. Teknik ini digunakan oleh komputer generasi awal, sehingga bila kita ingin berpindah dari satu program ke program lain, kita harus mereset dan me*load* kembali program-program tersebut.

Sebuah kernel sistem operasi tidak harus ada dan dibutuhkan untuk menjalankan sebuah komputer. Program dapat langsung dijalankan secara langsung di dalam sebuah mesin contohnya adalah CMOS Setup sehingga para pembuat program tersebut membuat program tanpa adanya dukungan dari sistem operasi atau *hardware abstraction*. Cara kerja seperti ini, adalah cara kerja yang

digunakan pada zaman awal-awal dikembangkannya komputer pada sekitar tahun 1950. Kerugian dari diterapkannya metode ini adalah pengguna harus melakukan reset ulang komputer tersebut dan memuatkan program lainnya untuk berpindah program, dari satu program ke program lainnya. Selanjutnya, para pembuat program tersebut membuat beberapa komponen program yang sengaja ditinggalkan di dalam komputer, seperti halnya loader atau debugger, atau dimuat dari dalam ROM Read-Only Memory. Seiring dengan perkembangan zaman komputer yang mengalami akselerasi yang signifikan, metode ini selanjutnya membentuk apa yang disebut dengan kernel sistem operasi.

Selanjutnya, para arsitek sistem operasi mengembangkan kernel sistem operasi yang pada akhirnya terbagi menjadi empat bagian yang secara desain berbeda, sebagai berikut:

- **Kernel Monolitik**. Kernel monolitik mengintegrasikan banyak fungsi di dalam kernel dan menyediakan lapisan abstraksi perangakat keras secara penuh terhadap perangkat keras yang berada di bawah sistem operasi.
- **Mikrokernel**. Mikrokernel menyediakan sedikit saja dari abstraksi perangkat keras dan menggunakan aplikasi yang berjalan di atasnya yang disebut dengan server untuk melakukan beberapa fungsionalitas lainnya.
- **Kernel Hibrida**. Kernel hibrida adalah pendekatan desain mikrokernel yang dimodifikasi. Pada *hybrid kernel*, terdapat beberapa tambahan kode di dalam ruangan kernel untuk meningkatkan performanya.
- **Exokernel**. Exokernel menyediakan hardware abstraction secara minimal, sehingga program dapat mengakses hardware secara langsung. Dalam pendekatan desain exokernel, library yang dimiliki oleh sistem operasi dapat melakukan abstraksi yang mirip dengan abstraksi yang dilakukan dalam desain *monolithic kernel*.

2.3.1 Kernel Monolitik

Pendekatan kernel monolitik didefinisikan sebagai sebuah antarmuka virtual yang berada pada tingkat tinggi di atas perangkat keras, dengan sekumpulan primitif atau system call untuk mengimplementasikan layanan-layanan sistem operasi, seperti halnya manajemen proses, konkurensi (*concurrency*), dan manajemen memori pada modul-modul kernel yang berjalan di dalam mode supervisor.

Meskipun jika setiap modul memiliki layanan operasi-operasi tersebut terpisah dari modul utama, integrasi kode yang terjadi di dalam monolithic kernel sangatlah kuat, dan karena semua modul berjalan di dalam *address space* yang sama, sebuah *bug* dalam salah satu modul dapat merusak keseluruhan sistem. Akan tetapi, ketika implementasi dilakukan dengan benar, integrasi komponen internal yang sangat kuat tersebut justru akan mengizinkan fitur-fitur yang dimiliki oleh sistem yang berada di bawahnya dieksploitasi secara efektif, sehingga membuat sistem operasi dengan *monolithic kernel* sangatlah efisien meskipun sangat sulit dalam pembuatannya.

Pada sistem operasi modern yang menggunakan *monolithic kernel*, seperti halnya Linux, FreeBSD, Solaris, dan Microsoft Windows, dapat memuat modul-modul yang dapat dieksekusi pada saat *kernel* tersebut dijalankan sehingga mengizinkan ekstensi terhadap kemampuan kernel sesuai kebutuhan, dan tentu saja dapat membantu menjaga agar kode yang berjalan di dalam ruangan kernel (*kernel-space*) seminim mungkin.

Di bawah ini ada beberapa sistem operasi yang menggunakan Monolithic kernel:

- Kernel sistem operasi UNIX tradisional, seperti halnya *kernel* dari sistem operasi UNIX keluarga BSD (NetBSD, BSD/I, FreeBSD, dan lainnya).
- Kernel sistem operasi GNU/Linux, Linux.
- Kernel sistem operasi Windows (versi 1.x hingga 4.x; kecuali Windows NT.

2.3.2 Mikrokernel

Pendekatan mikrokernel berisi sebuah abstraksi yang sederhana terhadap hardware, dengan sekumpulan primitif atau system call yang dapat digunakan untuk membuat sebuah sistem operasi agar dapat berjalan, dengan layanan-layanan seperti manajemen thread, komunikasi antar *address space*, dan komunikasi antar proses. Layanan-layanan lainnya, yang biasanya disediakan oleh kernel, seperti halnya dukungan jaringan, pada pendekatan *microkernel* justru diimplementasikan di dalam ruangan pengguna (*user-space*), dan disebut dengan server.

Server atau disebut sebagai peladen adalah sebuah program, seperti halnya program lainnya. Server dapat mengizinkan sistem operasi agar dapat dimodifikasi hanya dengan menjalankan program atau menghentikannya. Sebagai contoh, untuk sebuah mesin yang kecil tanpa dukungan jaringan, server jaringan (istilah *server* di sini tidak dimaksudkan sebagai komputer pusat pengatur jaringan) tidak perlu dijalankan. Pada sistem operasi tradisional yang menggunakan *monolithic kernel*, hal ini dapat mengakibatkan pengguna harus melakukan rekompilasi terhadap kernel, yang tentu saja sulit untuk dilakukan oleh pengguna biasa yang awam.

Dalam teorinya, sistem operasi yang menggunakan *microkernel* disebut jauh lebih stabil dibandingkan dengan *monolithic kernel*, karena sebuah *server* yang gagal bekerja, tidak akan menyebabkan *kernel* menjadi tidak dapat berjalan, dan *server* tersebut akan dihentikan oleh kernel utama. Akan tetapi, dalam prakteknya, bagian dari *system state* dapat hilang oleh server yang gagal bekerja tersebut, dan biasanya untuk melakukan proses eksekusi aplikasi pun menjadi sulit, atau bahkan untuk menjalankan server-server lainnya.

Sistem operasi yang menggunakan *microkernel* umumnya secara dramatis memiliki kinerja di bawah kinerja sistem operasi yang menggunakan *monolithic kernel*. Hal ini disebabkan oleh adanya *overhead* yang terjadi akibat proses

input/output dalam *kernel* yang ditujukan untuk mengganti konteks *(context switch*) untuk memindahkan data antara aplikasi dan server.

Beberapa sistem operasi yang menggunakan microkernel:

- IBM AIX, sebuah versi UNIX dari IBM
- Amoeba, sebuah kernel yang dikembangkan untuk tujuan edukasi
- Kernel Mach, yang digunakan di dalam sistem operasi GNU/Hurd, NextSTEP, OPENSTEP, dan Mac OS/X
- Minix, kernel yang dikembangkan oleh Andrew Tanenbaum untuk tujuan edukasi
- Symbian OS, sebuah sistem operasi yang populer digunakan pada hand phone, hanheld device, embedded device, dan PDA Phone.

2.3.3 Kernel Hibrida

Kernel Hibrida aslinya adalah mikrokernel yang memiliki kode yang tidak menunjukkan bahwa kernel tersebut adalah mikrokernel di dalam ruangan *kernel*-nya. Kode-kode tersebut ditaruh di dalam ruangan *kernel* agar dapat dieksekusi lebih cepat dibandingkan jika ditaruh di dalam ruangan *user*. Hal ini dilakukan oleh para arsitek sistem operasi sebagai solusi awal terhadap masalah yang terjadi di dalam mikrokernel: kinerja.

Beberapa orang banyak yang bingung dalam membedakan antara kernel hibrida dan kernel monolitik yang dapat memuat modul kernel setelah proses booting, dan cenderung menyamakannya. Antara kernel hibrida dan kernel monolitik jelas berbeda. Kernel hibrida berarti bahwa konsep yang digunakannya diturunkan dari konsep desain kernel monolitik dan mikrokernel. Kernel hibrida juga memiliki secara spesifik memiliki teknologi pertukaran pesan (*message passing*) yang digunakan dalam mikrokernel, dan juga dapat memindahkan beberapa kode yang seharusnya bukan kode kernel ke dalam ruangan kode kernel karena alasan kinerja.

Di bawah ini adalah beberapa sistem operasi yang menggunakan kernel hibrida:

- BeOS, sebuah sistem operasi yang memiliki kinerja tinggi untuk aplikasi multimedia.
- Novell NetWare, sebuah sistem operasi yang pernah populer sebagai sistem operasi jaringan berbasis IBM PC dan kompatibelnya.
- Microdoft Windows NT dan semua keturunannya.

2.3.4 Exokernel

Sebenarnya, Exokernel bukanlah pendekatan kernel sistem operasi yang umum seperti halnya microkernel atau monolithic kernel yang populer, melainkan sebuah struktur sistem operasi yang disusun secara vertikal.

Ide di balik exokernel adalah untuk memaksa abstraksi yang dilakukan oleh developer sesedikit mungkin, sehingga membuat mereka dapat memiliki banyak keputusan tentang abstraksi hardware. Exokernel biasanya berbentuk sangat kecil, karena fungsionalitas yang dimilikinya hanya terbatas pada proteksi dan penggandaan sumber daya.

Kernel-kernel klasik yang populer seperti halnya monolithic dan microkernel melakukan abstraksi terhadap hardware dengan menyembunyikan semua sumber daya yang berada di bawah hardware abstraction layer atau di balik driver untuk hardware. Sebagai contoh, jika sistem operasi klasik yang berbasis kedua kernel telah mengalokasikan sebuah lokasi memori untuk sebuah hardware tertentu, maka hardware lainnya tidak akan dapat menggunakan lokasi memori tersebut kembali.

Exokernel mengizinkan akses terhadap hardware secara langsung pada tingkat yang rendah: aplikasi dan abstraksi dapat melakukan request sebuah alamat memori spesifik baik itu berupa lokasi alamat physical memory dan blok

di dalam hard disk. Tugas kernel hanya memastikan bahwa sumber daya yang diminta itu sedang berada dalam keadaan kosong belum digunakan oleh yang lainnya dan tentu saja mengizinkan aplikasi untuk mengakses sumber daya tersebut. Akses hardware pada tingkat rendah ini mengizinkan para programmer untuk mengimplementasikan sebuah abstraksi yang dikhususkan untuk sebuah aplikasi tertentu, dan tentu saja mengeluarkan sesuatu yang tidak perlu dari kernel agar membuat kernel lebih kecil, dan tentu saja meningkatkan performa.

Exokernel biasanya menggunakan library yang disebut dengan libOS untuk melakukan abstraksi. libOS memungkinkan para pembuat aplikasi untuk menulis abstraksi yang berada pada level yang lebih tinggi, seperti halnya abstraksi yang dilakukan pada sistem operasi tradisional, dengan menggunakan cara-cara yang lebih fleksibel, karena aplikasi mungkin memiliki abstraksinya masing-masing. Secara teori, sebuah sistem operasi berbasis Exokernel dapat membuat sistem operasi yang berbeda seperti halnya Linux, UNIX, dan Windows dapat berjalan di atas sistem operasi tersebut.

5.1 Gateway

Sebuah gateway jaringan adalah suatu system internetworking yang mampu menggabungkan dua jaringan yang menggunakan protocol jaringan yang berbeda. Sebuah gateway jaringan dapat diimplementasikan sepenuhnya dalam perangkat lunak, dalam perangkat keras, atau kombinasi dari keduanya. Tergantung pada jenis protocol yang digunakan, gateway jaringan dapat beroperasi pada setiap tingkat OSI Layer.[Bradley Mitchel,1999]

Seiring dengan merebaknya internet, definisi gateway seringkali bergeser. Tidak jarang pula pemula menyamakan gateway dengan router yang sebetulnya tidak benar.

Kadangkala, kata gateway digunakan untuk mendeskripkan perangkat yang menghubungkan jaringan komputer besar dengan jaringan komputer besar lainnya. Hal ini muncul karena seringkali perbedaan protokol komunikasi dalam jaringan komputer hanya terjadi di tingkat jaringan komputer yang besar.

6.1 Iptables

Iptables adalah suatu program yang berisi *command line* yang digunakan pada Linux 2.4.x dan 2.6.x untuk mengkonfigurasi packet filtering IPv4 yang ditujukan untuk dioperasikan oleh system administrator.[netfilter,Iptables]

Sesuai kegunaannya untuk melakukan packet filtering, yaitu salah satu jenis teknologi keamanan yang digunakan untuk mengatur paket-paket apa saja yang diizinkan masuk ke dalam sistem atau jaringan dan paket-paket apa saja yang diblokir. Packet filtering umumnya digunakan untuk memblokir lalu lintas yang mencurigakan yang datang dari alamat IP yang mencurigakan, nomor port TCP/UDP yang mencurigakan, jenis protokol aplikasi yang mencurigakan, dan kriteria lainnya. Akhir-akhir ini, fitur packet filtering telah dimasukkan ke dalam banyak sistem operasi (IPTables dalam GNU/Linux, dan IP Filter dalam Windows) sebagai sebuah fitur standar, selain tentunya firewall dan router.

Packet filtering terbagi menjadi dua jenis, yakni:

- Static packet filtering (Penapisan paket statis)
- Dynamic packet filtering (Penapisan paket dinamis), atau sering juga disebut sebagai Stateful Packet Filter.

2.5.1 Static Packet Filtering (Penapisan Paket Statis)

Static packet filtering akan menentukan apakah hendak menerima atau memblokir setiap paket berdasarkan informasi yang disimpan di dalam header sebuah paket seperti halnya alamat sumber dan tujuan, port sumber dan tujuan,

jenis protokol, serta informasi lainnya. Jenis ini umumnya ditemukan di dalam sistem-sistem operasi dan router dan menggunakan sebuah tabel daftar pengaturan akses (access control list) yang berisi peraturan yang menentukan takdir setiap paket diterima atau ditolak.

Administrator jaringan dapat membuat peraturan tersebut sebagai daftar yang berurutan. Setiap paket yang datang kepada filter, akan dibandingkan dengan setiap peraturan yang diterapkan di dalam filter tersebut, hingga sebuah kecocokan ditemukan. Jika tidak ada yang cocok, maka paket yang datang tersebut ditolak, dan berlaku sebaliknya.

Peraturan tersebut dapat digunakan untuk menerima paket atau menolaknya dengan menggunakan basis informasi yang diperoleh dari header protokol yang digunakan, dan jenis dari paket tersebut. Kebanyakan perangkat yang memiliki fitur packet filtering, menawarkan kepada administrator jaringan untuk membuat dua jenis peraturan, yakni inbound rule dan outbound rule. Inbound rule merujuk kepada inspeksi paket akan dilakukan terhadap paket yang datang dari luar, sementara outbound rule merujuk inspeksi paket akan dilakukan terhadap paket yang hendak keluar.

2.5.2 Dynamic Packet Filtering (Penapisan Paket Dinamis)

Dynamic packet filtering beroperasi seperti halnya static packet filtering, tapi jenis ini juga tetap memelihara informasi sesi yang mengizinkan mereka untuk mengontrol aliran paket antara dua host secara dinamis, dengan cara membuka dan menutup port komunikasi sesuai kebutuhan. Jenis ini seringnya diimplementasikan di dalam produk firewall, di mana produk-produk tersebut dapat digunakan untuk mengontrol aliran data masuk ke jaringan dan aliran data keluar dari jaringan.

Sebagai contoh, sebuah dynamic packet filter dapat dikonfigurasikan sedemikian rupa sehingga hanya lalu lintas inbound protokol Hypertext Transfer

Protocol (HTTP) saja yang diizinkan masuk jaringan, sebagai respons dari request dari klien HTTP yang berada di dalam jaringan. Untuk melakukan hal ini, lalu lintas oubound yang melalui port 80/TCP akan diizinkan, sehingga request HTTP dari klien yang berada di dalam jaringan dapat diteruskan dan disampaikan ke luar jaringan. Ketika sebuah request HTTP outbound datang melalui filter, filter kemudian akan melakukan inspeksi terhadap paket untuk memperoleh informasi sesi koneksi TCP dari request yang bersangkutan, dan kemudian akan membuka port 80 untuk lalu lintas inbound sebagai respons terhadap request tersebut. Ketika respons HTTP datang, respons tersebut akan melalui port 80 ke dalam jaringan, dan kemudian filter pun menutup port 80 untuk lalu lintas inbound.

Pendekatan seperti ini tidak mungkin dilakukan di dalam static packet filtering, yang hanya dapat dikonfigurasikan untuk memblokir lalu lintas inbound ke port 80 atau membukanya, bukan sebagian dari lalu lintas tersebut. Meskipun demikian, dynamic packet filtering juga dapat dikelabui oleh penyerang, karena para penyerang dapat membajak sebuah sesi koneksi TCP dan membuat lalu lintas yang datang ke jaringan merupakan lalu lintas yang diizinkan. Selain itu, dynamic packet filtering juga hanya dapat digunakan pada paket-paket TCP saja, dan tidak dapat digunakan untuk paket User Datagram Protocol (UDP) atau paket Internet Control Message Protocol (ICMP), mengingat UDP dan ICMP bersifat connectionless yang tidak perlu membangun sebuah sesi koneksi (seperti halnya TCP) untuk mulai berkomunikasi dan bertukar informasi.

7.1 Bandwidth

Bandwidth adalah ukuran kapasitas pengiriman informasi yang digunakan dalam dunia telepon, jaringan komputer, sinyal frekuensi radio, dan monitor. Bandwidth biasanya diukur dalam satuan hertz (Hz) dan bits atau bytes per second (bps). Hz diukur berdasarkan rentang perbedaan frekuensi terendah dan frekuensi tertinggi yang dipancarkan. Bps diukur berdasarkan jumlah bit atau byte data terkirim per detik.[smitdev,2008]

Bandwidth / Lebar pita dalam teknologi komunikasi adalah perbedaan antara frekuensi terendah dan frekuensi tertinggi dalam rentang tertentu. Sebagai contoh, line telepon memiliki bandwidth 3000Hz (Hertz), yang merupakan rentang antara frekuensi tertinggi (3300Hz) dan frekuensi terendah (300Hz) yang dapat dilewati oleh line telepon ini.

Bandwidth adalah luas atau lebar cakupan frekuensi yang digunakan oleh sinyal dalam medium transmisi. Dalam kerangka ini, Bandwidth dapat diartikan sebagai perbedaan antara komponen sinyal frekuensi tinggi dan sinyal frekuensi rendah. frekuensi sinyal diukur dalam satuan Hertz. sinyal suara tipikal mempunyai Bandwidth sekitar 3 kHz, analog TV broadcast (TV) mempunyai Bandwidth sekitar 6 MHz. Bandwidth diartikan juga sebagai takaran jarak frekuensi. Dalam bahasa mudahnya, adalah sebuah takaran lalu lintas data yang masuk dan yang keluar. Dalam dunia hosting, kita di berikan jatah Bandwidth setiap bulan tergantung seberapa dalam kita merogoh kocek. Habisnya Bandwidth ditentukan seberapa banyak kita mengupload atau mendownload. Makin banyak anda melakukan aktivitas upload, ditambah makin banyaknya pengunjung yang mengakses, maka makin berkurang jatah Bandwidth yang diberikan. Misalkan, suatu situs web diberi jatah Bandwidth sebesar 1,5 Giga dalam sebulan. Dan sudah sejak bulan Desember rasanya jatah Bandwidth yang diberikan kurang. Pada bulan Desember, jatah Bandwidth habis sehari sebelum tahun baru. Dan berturut-turut bulan Januari, Februari, Maret dan April habis dalam 3 minggu. Otomatis, dalam seminggu terakhir situs web tersebut tidak bisa diakses.

2.6.1 Digital Bandwidth

Digital Bandwidth adalah jumlah atau volume data yang dapat dikirimkan melalui sebuah saluran komunikasi dalam satuan bits per second tanpa distorsi.

2.6.2 Analog Bandwidth

Analog Bandwidth adalah perbedaan antara frekuensi terendah dengan frekuensi tertinggi dalam sebuah rentang frekuensi yang diukur dalam satuan Hertz (Hz) atau siklus per detik, yang menentukan berapa banyak informasi yang bisa ditransimisikan dalam satu saat.

2.6.3 Bandwidth Komputer

Bandwidth Komputer di dalam jaringan Komputer, Bandwidth sering digunakan sebagai suatu sinonim untuk data transfer rate yaitu jumlah data yang dapat dibawa dari sebuah titik ke titik lain dalam jangka waktu tertentu (pada umumnya dalam detik). Jenis Bandwidth ini biasanya diukur dalam bps (bits per second). Adakalanya juga dinyatakan dalam Bps (bytes per second). Suatu modem yang bekerja pada 57,600 bps mempunyai Bandwidth dua kali lebih besar dari modem yang bekerja pada 28,800 bps. Secara umum, koneksi dengan Bandwidth yang besar/tinggi memungkinkan pengiriman informasi yang besar seperti pengiriman gambar/images dalam video presentation.

2.6.4 Alokasi Bandwidth

Alokasi atau reservasi Bandwidth adalah sebuah proses menentukan jatah Bandwidth kepada pemakai dan aplikasi dalam sebuah jaringan. Termasuk didalamnya menentukan prioritas terhadap berbagai jenis aliran data berdasarkan seberapa penting atau krusial dan delay-sensitive aliran data tersebut. Hal ini memungkinkan penggunaan Bandwidth yang tersedia secara efisien, dan apabila sewaktu-waktu jaringan menjadi lambat, aliran data yang memiliki prioritas yang lebih rendah dapat dihentikan, sehingga aplikasi yang penting dapat tetap berjalan dengan lancar. Besarnya saluran atau Bandwidth akan berdampak pada kecepatan transmisi. Data dalam jumlah besar akan menempuh saluran yang memiliki Bandwidth kecil lebih lama dibandingkan melewati saluran yang memiliki Bandwidth yang besar. Kecepatan transmisi tersebut sangat dibutuhkan untuk

aplikasi Komputer yang memerlukan jaringan terutama aplikasi real-time, seperti videoconferencing. Penggunaan Bandwidth untuk LAN bergantung pada tipe alat atau medium yang digunakan, umumnya semakin tinggi Bandwidth yang ditawarkan oleh sebuah alat atau medium, semakin tinggi pula nilai jualnya. Sedangkan penggunaan Bandwidth untuk WAN bergantung dari kapasitas yang ditawarkan dari pihak ISP, perusahaan harus membeli Bandwidth dari ISP, dan semakin tinggi Bandwidth yang diinginkan, semakin tinggi pula harganya. sebuah teknologi jaringan baru dikembangkan dan infrastruktur jaringan yang ada diperbaharui, aplikasi yang akan digunakan umumnya juga akan mengalami peningkatan dalam hal konsumsi Bandwidth. Video streaming dan Voice over IP (VoIP) adalah beberapa contoh penggunaan teknologi baru yang turut mengkonsumsi Bandwidth dalam jumlah besar

8.1 Bandwidth Limiter

Bandwidth Limiter atau Pembatas Bandwidth adalah sebuah cara atau aplikasi atau software untuk membatasi koneksi internet dari user dalam suatu Local Area Network atau Wide Area Network sehingga koneksi yang didapat oleh semua user dalam satu struktur jaringan, diharapkan untuk bisa hampir setara. Untuk mendapatkan software pembatas bandwidth atau lebih dikenal dengan bandwidth limiter, bukan merupakan sebuah jalan yang mudah apalagi untuk pengguna sistem operasi windows. Kebanyakan tips yang diberikan oleh para master jaringan untuk melakukan pembatasan bandwidth adalah menggunakan Linux**.**

Seiring dengan perkembangan teknologi informasi, khususnya di bidang jaringan, maka telah diciptakan suatu sistem yang berfungsi untuk memanage bandwidth yang dikenal dengan proses shaping, queueing dan mangleing pada sistem operasi mikrotik yang selain itu juga dapat berfungsi sebagai router dan firewall pada jaringan.

BAB III

METODOLOGI

Dalam perancangan sebuah sistem, kondisi, kejadian serta permasalahan yang terjadi merupakan suatu acuan dalam merancang sebuah sistem yang efektif, sehingga dalam pelaksanaannya nanti dapat digunakan dengan baik dan bahkan dapat dikembangkan menjadi sistem yang lebih sempurna. Metodologi sangat diperlukan dalam membuat analisis terhadap pembuatan dan pengembangan suatu sistem. Metode analisis berfungsi untuk menganalisis kebutuhan pembuatan sistem TProxy ini.

3.1 Metode Analisis

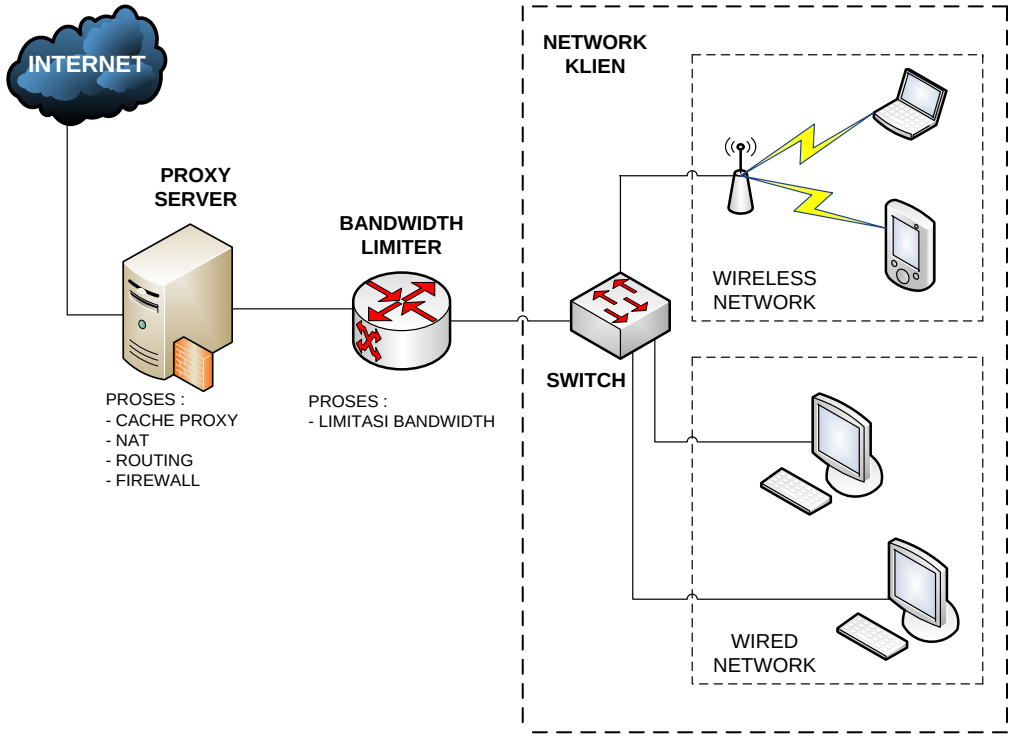
Pada sub ini akan menjelaskan metode analisis TProxy dan eksplorasi TProxy yang diimplementasikan pada suatu jaringan komputer sederhana yang terhubung dengan Internet.

Analisis kebutuhan perangkat dilakukan untuk mengetahui semua permasalahn serta kebutuhan yang diperlukan untuk mengimplementasikan aplikasi TProxy. Analisis dimulai dengan mempelajari literatur yang didapat, kemudian merealisasikannya secara bertahap sesuai dengan konsep penelitian. Dimulai dengan pemilihan perangkat keras yang sesuai dan dilanjutkan dengan perangkat lunak, kemudian perancangan arsitektur jaringan dan dilanjutkan dengan instalasi dan konfigurasi pada TProxy.

3.2 Analisis Masalah

Analisis masalah sistem adalah prosedur yang dilakukan untuk menilai aplikasi TProxy yang akan dibuat. Analisis sistem bertujuan untuk mengidentifikasi objek-objek yang dibutuhkan oleh sistem. Setelah objek-objek tersebut teridentifikasi, maka dapat di implementasikan secara terperinci sehingga didapatkan hasil yang sesuai dengan tujuan akhir penelitian.

Permasalahan yang dianalisa dalam tugas akhir ini adalah bagaimana cara membangun dan memaksimalkan suatu Proxy Server yang dapat diaplikasikan pada topologi yang dapat memaksimalkan kinerja dari Proxy Server dan relatif lebih baik dari topologi yang biasa digunakan pada saat ini yaitu Proxy Server berada setelah Bandwidth Limiter dari sisi Network Klien (Gambar 3.1).



Gambar 3.1 Proxy Server Berada Setelah Bandwidth Limiter.

Pada topologi tersebut terdapat beberapa permasalahan Proxy Server tidak dapat bekerja dengan maksimal , yang diantaranya :

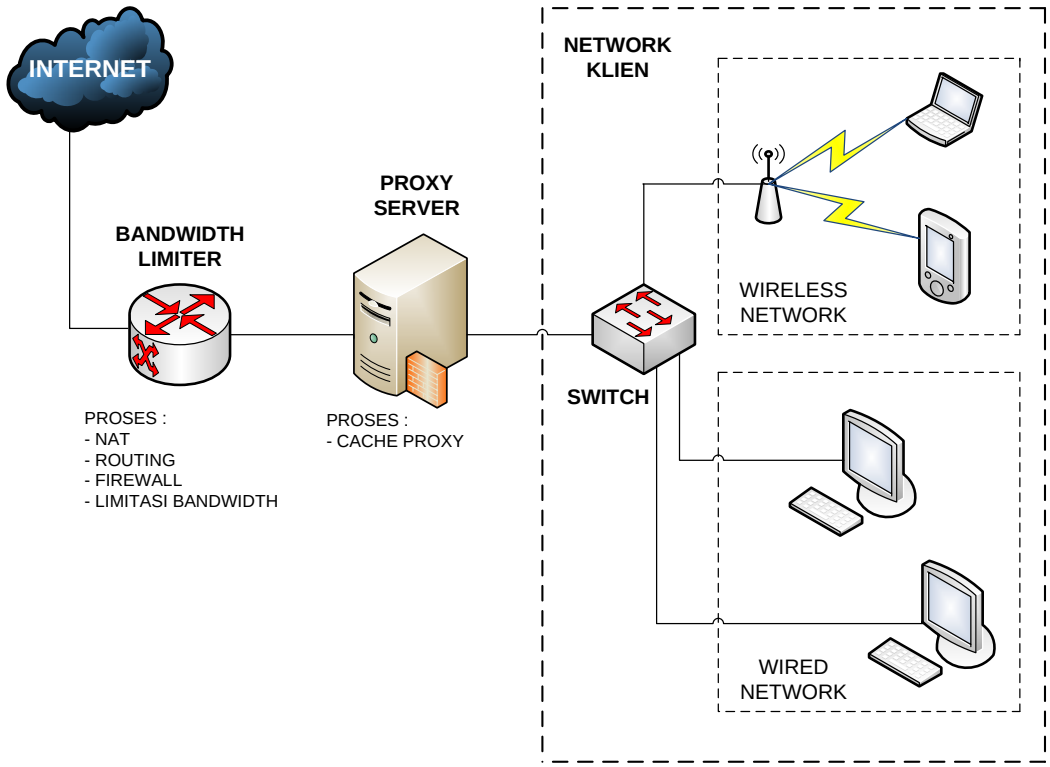
1. Dikarenakan terdapat beberapa proses yang terjadi di dalam satu server sementara sistem Proxy membutuhkan *resource* yang maksimal dari sistem.

2. Jawaban dari permintaan Network Klien kepada Proxy Server telah terbatasi oleh proses shaping pada Bandwidth Limiter.

3. Tidak adanya limiter setelah Proxy Server menuju Internet, maka permintaan menuju Internet dari Proxy Server tidak dapat dibatasi dan akan memakan sisa bandwidth yang ada.

4. Rumitnya konfigurasi pada saat maintenance atau pada saat troubleshooting dari proses yang terdapat pada Proxy Server.

Untuk mengatasi beberapa permasalahan tersebut maka dilakukan perubahan topologi, yaitu Proxy Server ditempatkan diantara Network Klien dan Bandwidth Limiter, dan Proxy Server berfungsi sebagai gatway transisi bagi Network Klien (Gambar 3.2).



Gambar 3.2 Proxy Server Berada Diantara Bandwidth Limiter dan Klien.

Topologi alternatif tersebut diharapkan dapat diaplikasikan pada jaringan Internet yang menggunakan Proxy Server dan Bandwidth Limiter, dengan beberapa kelebihan sebagai berikut :

1. Proses caching pada Proxy Server dapat bekerja lebih maksimal, karena resource pada sistem dapat lebih terfokus , karena hanya digunakan untuk proses caching pada Proxy.

2. Adanya Bandwidth Limiter setelah Proxy server menuju Internet maka permintaan menuju Internet dari Proxy Server dapat dibatasi.

3. Jawaban dari permintaan Network Klien kepada Proxy Server akan langsung diterima oleh Network Klien dengan arus tanpa terbatasi proses shaping pada Bandwidth Limiter sehingga kecepatan browsing

Internet pada Network Klien akan lebih cepat, dan Bandwidth Limiter hanya membatasi arus permintaan dari Network Klien dan Proxy Server Menuju Internet.

4. Konfigurasi pada saat maintenance atau troubleshooting akan lebih mudah karena Proxy Server sudah tidak tercampur lagi dengan proses-proses lainnya.

Proses maksimalisasi dari Proxy Server tersebut tidak hanya dapat dilakukan dengan merubah posisi dari Proxy Server dan Bandwidth Limiter pada schematicnya saja, melainkan ada beberapa permasalahan yang harus diselesaikan agar Proxy Server tersebut dapat ditempatkan diantara Network Klien dan Bandwidth Limiter, sehingga harus merubah keseluruhan dari proses instalasi dan konfigurasi dari Proxy Server sehingga dapat memberikan keluaran TPROXY.

3.3 Hasil Analisis

Dari hasil analisis didapat suatu yang akan dibuat, yaitu sistem Proxy yang dapat memberikan keluaran TPROXY didalamnya sehingga dapat dijalankan pada suatu topologi jaringan Internet dimana proxy server berada diantara bandwidth limiter dan network klien.

3.3.1 Perangkat Keras dan Perangkat Lunak Yang Digunakan

3.3.1.1 Perangkat Keras Yang Digunakan

Untuk dapat menjalankan Sistem TProxy ini yang paling penting adalah kebutuhan *hardware*. Adapun spesifikasi minimalnya sebagai berikut :

1. Komputer dengan spesifikasi *minimum* prosessor 1 Ghz**,** RAM 1024 MB.
2. *Harddisk* 40 GB.

3. *Ethernet card* minimal 2.
4. Router box sebagai bandwidth limiter
5. Komputer klien
6. Kabel UTP dengan konektor RJ45

3.3.1.2 Perangkat Lunak Yang Digunakan

Untuk mendukung sistem tersebut diperlukan kebutuhan lainnya yaitu kebutuhan perangkat lunak, diantaranya :

1. Sistem Operasi Linux Slackware.
2. Kernel Linux 2.6.
3. Cttproxy 2.6.
4. Iptables 1.3.
5. Squid 2.7 STABLE.
6. Sistem Operasi Mokrotik pada router box
7. Sistem Operasi Windows pada komputer klien

3.3.2 Gambaran Umum Sistem

3.3.2.1 Cara Kerja

Proxy bekerja dengan mendengarkan request dari klien internal dan mengirim request tersebut ke jaringan eksternal seolah-olah proxy server itu sendiri yang menjadi client. Pada waktu proxy server menerima respon dari server publik, ia akan memberikan respon tersebut ke client yang asli seolah-olah ia adalah public server.

3.3.2.2 Kemanan

Dari segi keamanan, Proxy server dapat menyembunyikan semua user di belakang satu mesin, dapat memfilter URL, dan dapat membuang content yang mencurigakan atau ilegal. Jadi meskipun mula-mula dibuat sebagai cache nonsekuriti, proxy server juga dapat difungsikan sebagai alat untuk melakukan firewalling.

Proxy server memperbarui request layanan pada jaringan eksternal atas nama client mereka pada jaringan private. Ini secara otomatis menyembunyikan identitas dan jumlah client pada jaringan internal dari jaringan eksternal. Karena posisi mereka di antara client internal dan server publik, proxy juga dapat menyimpan content yang sering diakses dari jaringan publik untuk mengurangi akses ke jaringan publik tersebut. Kebanyakan implementasi nyata proxy sekuriti meliputi pemfilteran paket dan Network Address Translation untuk membangun firewall yang utuh. Teknologi tersebut dapat digabungkan dengan proxy untuk menghilangkan serangan langsung dari jaringan eksternal menuju jaringan lokal.

3.3.2.3 Pemblokiran URL

Pemblokiran URL memungkinkan administrator untuk menolak situs tertentu berdasarkan URL mereka. Secara teori, ini akan menjauhkan pengguna internet dari situs web yang tidak boleh mereka akses. Fungsi ini mudah di implementasikan, dikarenakan Proxy mengecek setiap request dengan daftar halaman yang ditolak sebelum ia memperbarui request tersebut. Jika URL diblokir, proxy tidak akan meminta atau membereikan halaman tersebut.

Namun, pemblokiran URL mudah diatasi, karena situs web biasa ditulis dengan menggunakan alamat IP atau bahkan keseluruhan nomor alamat. User dapat mengetik apa saja dalam web browser mereka untuk mengakses halaman yang sama, oleh karena itu URL bloker pada proxy dibuat dengan alamat lengkap web tersebut dan URL bloker akan mengecek alamat lengkap URL.

Masalah lain dengan pemblokiran URL adalah memperbarui situs yang di blokir. Situs seperti hacking, pornografi, dan situs game mempunyai masa hidup yang singkat, situs-situs tersebut dapat muncul dan hilang dengan cepat.

3.3.2.4 Pemblokiran Routing

Paket Transport layer perlu diarahkan karena request semuanya diperbaharui. Hal ini menghilangkan eksploitasi Transport layer seperti routing, fragmentasi, dan beragam serangan denial of service. Dengan menghilangkan routing, kita dapat memastikan bahwa semua protocol yang belum ditentukan tidak akan dilewatkan menuju jaringan public. Pemblokiran routing mungkin merupakan salah satu keuntungan proxy server yang penting, karena paket TCP/IP sebenarnya lewat antara jaringan internel dan eksternal. Banyak serangan denial of service dan eksploitasi yang dapat dicegah.

Sayangnya, pemblokiran routing tidak begitu sering digunakan karena banyaknya protokol yang ada. Sedapat mungkin jangan perbolehkan paket low-level melewati proxy server. Kebanyakan proxy server memperbolehkan untuk membuat proxy TCP generik untuk semua port yang menggunakan proxy SOCKS generik atau utiiliti redirect Unix. Proxy generik ini, meskipun mereka tidak dapat melakukan pemfilteran content, tetapi memungkinkan untuk mencegah paket TCP/IP berlalu-lalang antar jaringan.

3.3.2.5 Menyembunyikan Client

Fitur keamanan utama proxy server adalah menyembunyikan client, Seperti Network Address Translation, proxy server dapat membuat seluruh jaringan internal muncul sebagai satu mesin dari Internet karena hanya satu mesin yang melewatkan request ke Internet.

Seperti Network Address Translatation, proxy server mencegah host eksternal untuk mengakses layanan pada mesin internal. Pada proxy server, tidak ada routing ke client karena domain alamat jaringan internal dan eksternal bisa saja tidak kompatibel dan karena transport layer routing tidak ada di antara kedua jaringan.

Proxy melakukan fitur ini dengan memperbarui request, bukan mengganti dan menghitung ulang header alamat. Sebagai contoh, pada waktu client membuat request melalui proxy server, proxy server menerima request tersebut seolah-olah web server tujuan pada jaringan internal. Ia kemudian memperbarui request ke jaringan eksternal seolah web browser biasa. Pada waktu proxy menerima respon dari web server yang sebenarnya, ia memberikan respon tersebut kepada client internalnya. Hanya HTTP yang dilewatkan melalui proxy, bukan TCP atau IP. TCP/IP (dan protokol low-level lainnya) diperbarui oleh proxy; mereka tidak akan dilewatkan melalui proxy.

3.3.2.6 Logging

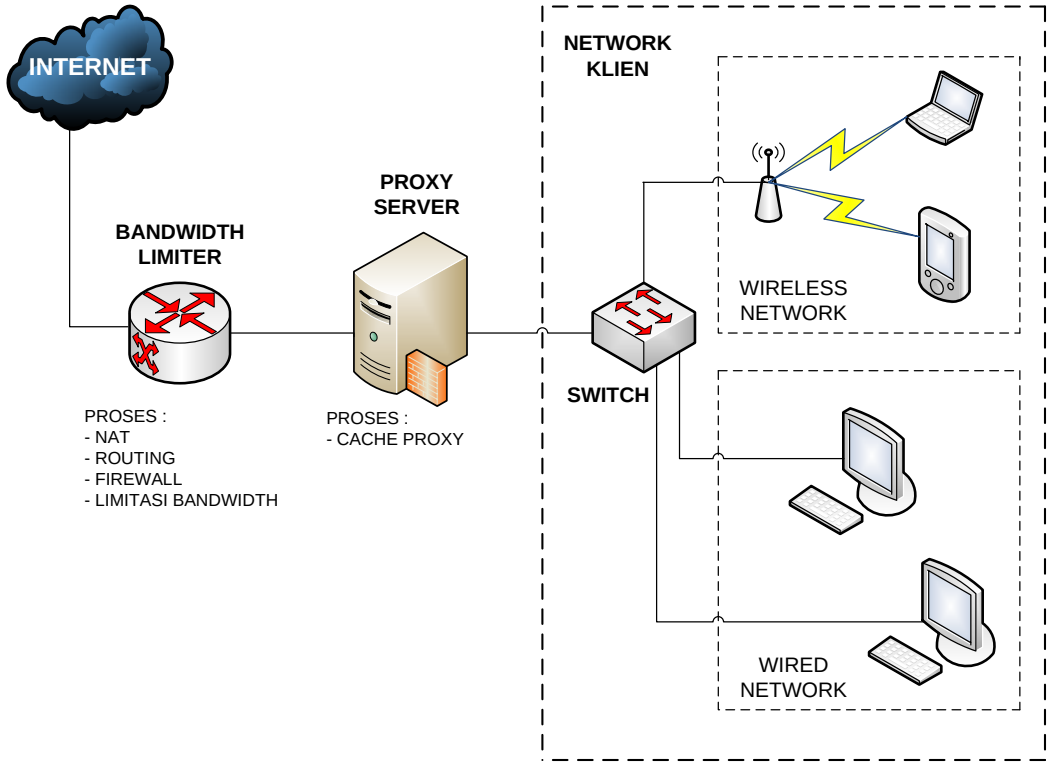
Manfaat keamanan yang lain dari proxy adalah fasilitas logging yang disediakan. Proxy memastikan bahwa semua content mengalir melalui satu poin, yang bisa menjadi tempat pemeriksaan data jaringan. Kebanyakan proxy server akan mencatat penggunaan proxy oleh user dan dapat dikonfigurasi untuk mencatat situs yang mereka kunjungi. Ini akan memungkinkan kita untuk mengatur ulang cara browsing user jika administrator curiga akan adanya aktivitas yang ilegal atau tidak semestinya.

Fasilitas alert disediakan oleh beberapa proxy untuk memperingatkan administrator atas serangan yang terjadi, meskipun proxy umumnya bukan sasaran penyerangan. Namun, fasilitas ini bisa digunakan untuk meningkatkan kewaspadaan pada interface eksternal, yang sering kali dicoba untuk dieksplotiasi oleh hacker.

3.3.3 Analisis Kebutuhan Sistem

3.3.3.1 Arsitektur Jaringan

Arsitektur jaringan yang akan diaplikasikan dan menjadi kebutuhan pada jaringan dengan TProxy server ini, adalah jaringan dimana proxy server berada diantara bandwidth limiter dan network klien, sesuai dengan tujuan dari penelitian.



Gambar 3.3 Arsitektur Jaringan Dengan TProxy.

3.3.3.2 Kebutuhan Proses

Di dalam sistem TProxy dibutuhkan proses-proses antara lain :

1. Menerima masukan : data hasil browsing Internet.
2. Melakukan proses perekaman data hasil browsing *(caching)*.
3. Memberikan data yang dibutuhkan : data hasil browsing yang telah direkam pada proses *caching* yang dipanggil kembali pada saat komputer klien melakukan browsing Internet.

3.3.3.3 Kebutuhan Server

Kebutuhan dari TProxy terdiri dari kebutuhan perangkat keras dan perangkat lunak, kebutuhan perangkat keras telah dianalisis pada bagian sebelumnya. Maka analisis selanjutnya adalah proses implementasi TProxy sebagai aplikasi.

Pemilihan software yang dibutuhkan untuk TProxy didasarkan dari pengumpulan literatur dan pemilihan software dengan metode browsing yang telah dijelaskan pada bab sebelumnya. Pemilihan software tersebut dilakukan dengan alasan mendapatkan software dari distribusi yang stabil dan tepat, sehingga dapat berjalan selaras antara beberapa software yang mendukung terbentuknya TProxy tersebut.

Beberapa hal yang harus dilakukan dalam tahapan implementasi TProxy:

1. Instalasi Sistem Operasi linux
2. Kompilasi kernel yang mendukung sistem TProxy
3. Kompilasi squid proxy.
4. Instalasi squid proxy.
5. Konfigurasi squid proxy.
6. Melakukan direktif arus internet manuju Tproxy dengan iptables.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Implementasi Secara Umum

Tahapan implementasi sistem merupakan tahap dimana sistem mampu diaplikasikan dalam keadaan yang sesungguhnya. Dari implementasi ini akan diketahui apakah sistem yang dibuat dapat berjalan dengan baik atau tidak. Serta apakah sistem menghasilkan output yang sesuai dengan perancangan yang ada.

4.2 Tahapan Konfigurasi Perangkat Lunak

Dalam pengaplikasian perangkat lunak, terdapat beberapa tahap yang terdiri atas :

4.2.1. Tahapan Instalasi Sistem Operasi

Dalam penelitian ini sistem operasi yang digunakan adalah salah satu sistem operasi dari distribusi Linux, yaitu Slackware. Dalam tahapan ini yang paling terpenting adalah pembagian partisi dari harddisk yang digunakan, dikarenakan partisi tersebut akan menjadi partisi penyimpanan cache hasil browsing Internet yang disesuaikan dengan konfigurasi yang dibuat.

4.2.2. Tahapan Kompilasi dan Patching Kernel

Kompilasi dan patching kernel merupakan salah satu tahapan terpenting dalam penelitian ini, dikarenakan harus tepatnya pemilihan kernel dan perangkat lunak lainnya yang dibutuhkan untuk sistem TProxy ini.

Perangkat lunak yang digunakan dalam tahapan ini adalah :

1. Kernel linux 2.6

   Perangkat lunak ini dipilih berdasarkan metode browsing dan perbandingan yang dilakukan pada beberapa kernel lainnya sampai

didapatkan kernel linux 2.6 yang dinyatakan stabil dan dapat berjalan dengan baik pada sistem TProxy.

2. Kernel cttproxy 2.6

   Perangkat lunak ini merupakan kernel tambahan yang didalamnya berisi initi-inti dari kebutuhan sistem TProxy yang akan digabungkan dengan kernel linux 2.6 yang telah dikompilasi. Kernel cttproxy 2.6 ini merupakan kernel yang sesuai dan stabil untuk dijalankan bersama kernel linux 2.6.

Berikut ini adalah tahapan instalasi dan kompilasi kernel :

a. Direktori /usr/src

   Direktori /usr/src adalah direktori dimana kernel linux berada dan kernel linux 2.6 yang telah disimpan pada direktori tersebut, dengan mengetikkan perintah #cd /usr/src seperti pada gambar sebagai berikut :

Gambar 4.1 Direktori /usr/src

b. Ekstraksi kernel

   Setelah berada pada direktori /usr/src dengan mengetikkan perintah di atas, tahap selanjutnya adalah mengekstrak paket linux 2.6 yang masih berupa file kompresi tar.gz dengan mengetikkan perintah #tar xzvf linux-2.6.20.21.tar.gz seperti pada gambar :

Gambar 4.2 Ekstraksi kernel

c. Penghapusan file @linux

   Setelah berhasil mengekstrak file tersebut menjadi sebuah folder berisi paket-paket yang siap digunakan, selanjutnya adalah menghapus file *linux* yang merupakan link dari folder kernel linux sebelumnya dengan menggunakan perintah #rm linux seperti tertera pada gambar :

```
root@slx-64:/usr/src# rm linux
```

Gambar 4.3 Menghapus file @linux

d. Link direktori kernel dengan file @linux

Berikutnya adalah membuat kembali file linux yang telah dihapus, untuk dikoneksikan dengan folder linux yang baru diekstrak, dikarenakan sistem akan membaca file linux tersebut sebagai folder kernel yang baru dan akan dijalankan dalam penelitian TProxy ini, seperti pada gambar :

```
root@slx-64:/usr/src# ln -s linux-2.6.20.21 linux
```

Gambar 4.4 Link direktori linux

e. Ekstraksi cttproxy

Ekstraksi paket cttproxy adalah tahapan pengekstrakan paket cttproxy yang akan digunakan untuk dipatch dengan source kernel yang baru, file ini adalah file yang dibutuhkan sistem sehingga memberikan keluaran TProxy.

```
root@slx-64:/usr/src# tar xzvf cttproxy-2.6.20-2.0.6.tar.gz
```

Gambar 4.5 Ekstraksi cttproxy

f. Direktori linux

Direktori source kernel linux adalah direktori dimana paket-paket kernel berada, yang dibutuhkan untuk keluaran sistem TProxy dan dieksekusi dari dalam folder tersebut. Adapun memasuki direktori linux adalah dengan mengetikkan perintah cd /usr/src/linux, seperti pada gambar.

```
root@slx-64:/usr/src# cd /usr/src/linux
```

Gambar 4.6 Direktori /usr/src/linux

g. Patching kernel

Patching kernel adalah tahapan dimana kernel linux digabungkan dengan kernel cttproxy, untuk selanjutnya dikompilasi.

Gambar 4.7 Patching kernel

h. Menu konfigurasi kernel

Dalam tahapan ini, dilakukan pengaktifan untuk dapat memasuki ke menu konfigurasi kernel yang selanjutnya akan dikompilasi dan dieksekusi pada beberapa opsi yang dibutuhkan dalam penelitian ini, dengan mengetikkan perintah #zcat /proc/config.gz >.config seperti pada gambar :

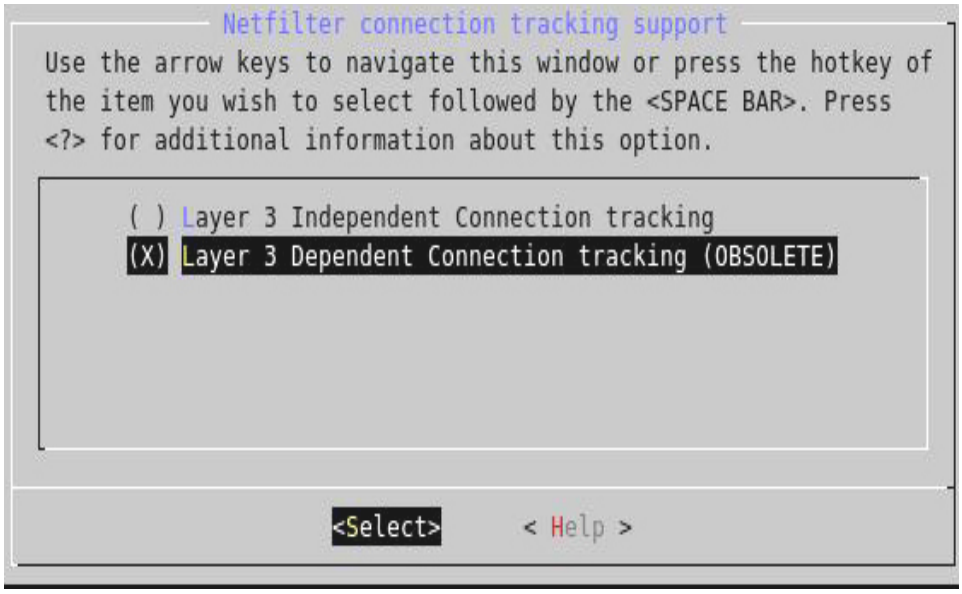
Gambar 4.8 Mengaktifkan menu konfigurasi kernel

Pada tahapan ini sistem akan melakukan loading menu dimana paket-paket kernel yang terdapat didalamnya dapat dikompilasi dan dieksekusi sesuai dengan kebutuhan sistem dengan menggunakan perintah #make menuconfig seperti pada gambar :

Gambar 4.9 Memasuki menu konfigurasi kernel

i. Layer 3 Dependent Connection Tracking

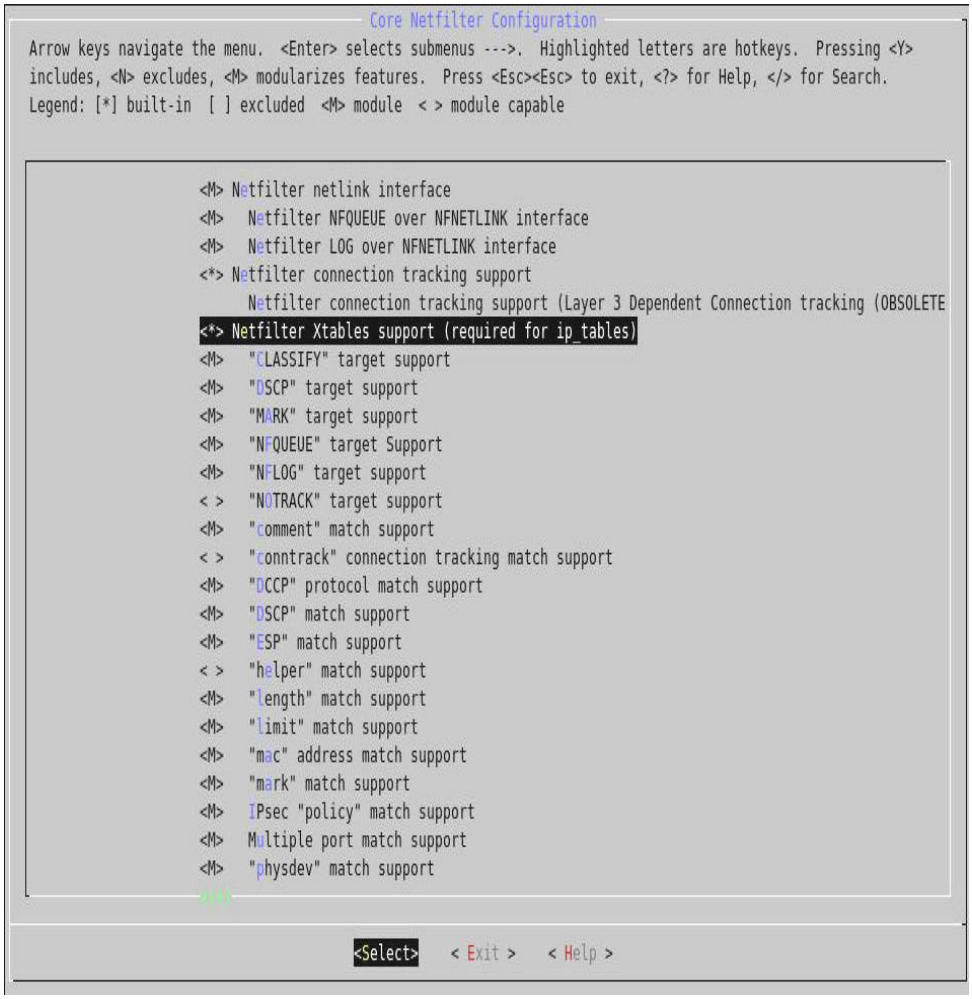
Menu ini merupakan sub menu dari Netfilter Connection Tracking Support, dengan opsi defaultnya yaitu Layer 3 independent Connection Tracking. Pada sub menu ini akan difungsikan sebagai built-in kernel seperti pada gambar :



Gambar 4.10 Layer 3 Dependent Connection tracking (OBSOLETE)

j. Netfilter Xtables support

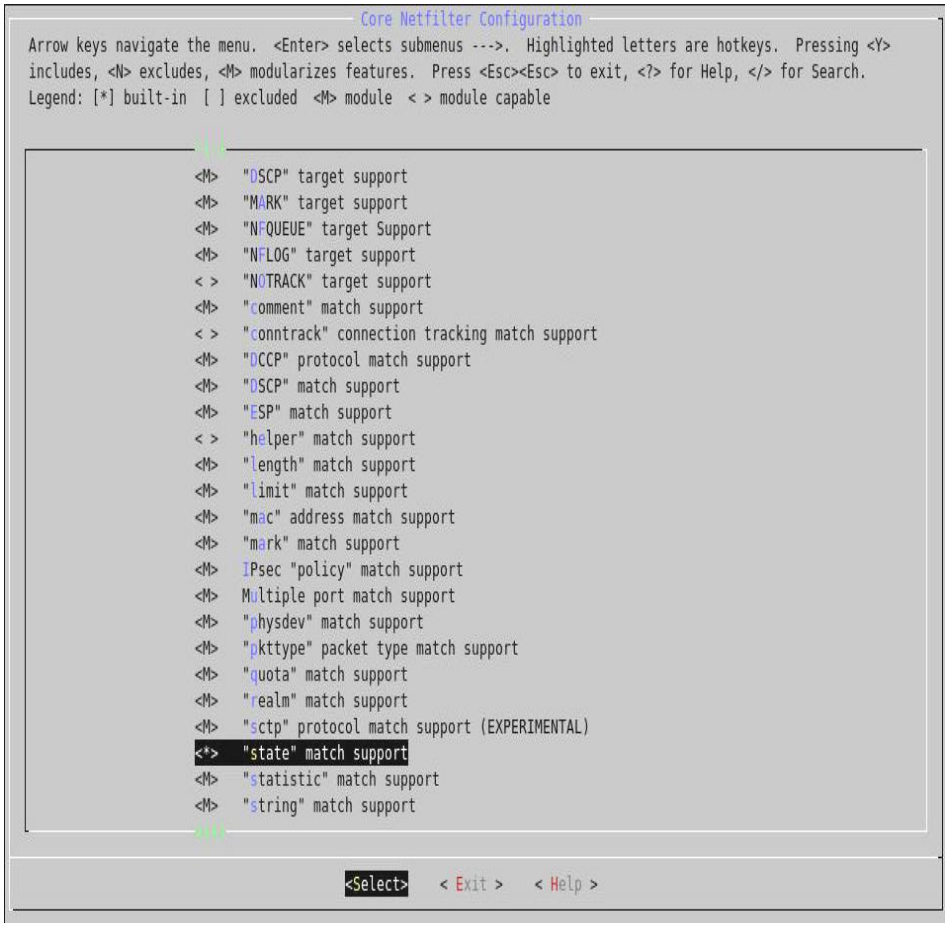
Pada menu ini berhubungan dengan beberapa menu lain yang akan berfungsi sebagai penapisan filter-filter yang dilewatkan, terutama yang berhubungan dengan iptables yang akan dibahas pada sub bab selanjutnya, menu ini juga diaktifkan sebagai built-in kernel, tertera pada gambar berikut :



Gambar 4.11 Netfilter Xtables Support

k. State match support

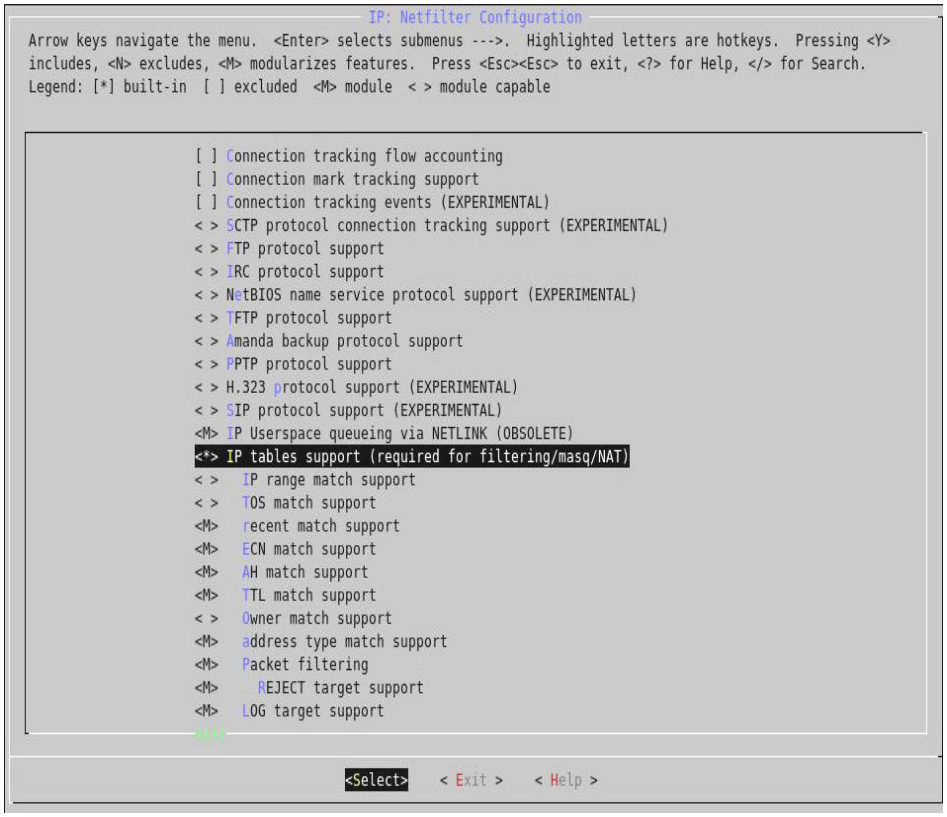
Menu ini akan berfungsi sebagai proses penting , dikarenakan akan menjadi penghubung antara menu Netfilter dengan menu IPtables Suport. Menu ini harus dieksekusi sebagai built-in kernel seperti tertera pada gambar.



Gambar 4.12 State match support

l. IP tables support

Menu ini jelas sangat dibutuhkan, dikarenakan menu ini yang akan mendukung menu-menu selanjutnya yang perlu diaktifkan sebagai built-in kernel, dan apabila menu ini tidak diaktifkan sebagai built-in kernel maka menu-menu selanjutnya tidak akan muncul, terlihat pada gambar bahwa menu ini diaktifkan sebagai built-in kernel.



Gambar 4.13 IP tables support

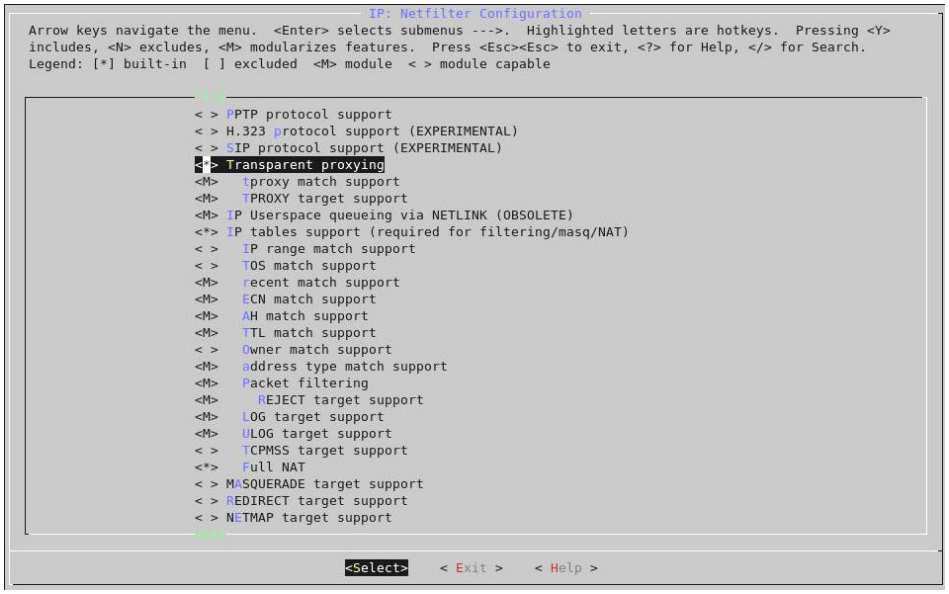
m. Full NAT

Pada menu Full NAT ini, disamping dibutuhkan untuk proses NAT atau network address translation juga dibutuhkan untuk mensupport dimana menu Transparent Proxying ditampilkan. Dikarenakan menu Full NAT ini sebagai salah satu syarat dalam mengaktifkan TProxy, sehingga apabila tidak dieksekusi sebagai built-in kernel, proses TProxy tidak akan dapat berjalan dikarenakan proses Transparent Proxying tidak dapat dimunculkan, pengeksekusian menu Full NAT sebagai built-in kernel dapat dilihat pada gambar :

Gambar 4.14 Full NAT

n. Transparent Proxying

Menu ini adalah menu dimana tahapan kompilasi sistem TProxy dilakukan, menu ini hanya akan muncul apabila proses pemilihan menu-menu sebelumnya sesuai dan saling mendukung sehingga sistem TProxy ini dapat berjalan. Pada menu ini terdapat dua sub menu yaitu tproxy match suppot dan TPROXY target support yang dimana keduanya akan dieksekusi sebagai module dan tidak sebagain built-in kernel seerti tertera pada gambar :



Gambar 4.15 Transparent Proxying

o. Proses penyimpanan konfigurasi kernel

Setelah proses kompilasi selesai, selanjutnya sebelum pengeksekusian dilakukan penyimpanan konfigurasi dari beberapa menu dan sub menu yang telah dikompilasi pada menu konfigurasi kernel. Keterangan tertera pada gambar:

Gambar 4.16 Save konfigurasi kernel

p. Make & Make Install

Tahapan ini merupakan tahapan eksekusi dari hasil kompilasi beberapa menu dan sub menu yang telah dijelaskan sebelumnya. Proses make dan make install ini memakan waktu cukup lama tergantung dari spesifikasi hardware yang digunakan. Proses ini dapat dilakukan dengan mengetikkan perintah make, lalu make install seperti pada gambar di bawah ini :

Gambar 4.17 Make

Gambar 4.18 Make Install

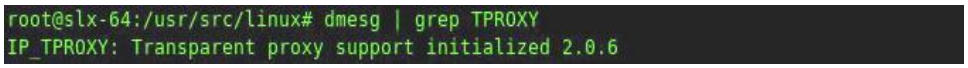
q. Reboot

Restart sistem dibutuhkan dalam proses ini, dikarenakan kernel yang baru dikompilasi akan dieksekusi sepenuhnya pada saat sistem kembali dinyalakan dalam proses restart. Seperti pada gambar :

Gambar 4.19 Reboot

r. Pengecekan keluaran sistem TProxy

Sistem TProxy yang dihasilkan dari hasil kompilasi dan eksekusi kernel yang telah dilakukan sebelumnya dapat dilihat dengan menggunakan perintah #dmesg | grep TPROXY seperti pada gambar. Apabila proses kompilasi sudah benar maka akan menghasilkan keluaran seperti pada gambar, dan apabila proses kompilasi tidak berhasil, maka keluaran tersebut tidak akan muncul.



Gambar 4.20 Pengecekan status TPROXY

4.2.3. Tahapan Instalasi IPtables

Dalam penelitian ini package iptables yang digunakan adalah Iptables 1.3, yaitu suatu perangkat lunak yang telah diuji kesesuaiannya dengan kernel linux 2.6 dan cttproxy 2.6 , package ini akan berfungsi untuk kebutuhan packet filtering dan penapisan arus klien menuju TProxy server dengan konfigurasi pada syntax iptables yang telah mendukung sistem TProxy. Berikut ini adalah tahapan instalasi iptables untuk kebutuhan sistem :

a. Penghapusan paket IP tables

Dikarenakan IP tables yang sudah terinstal pada sistem pada saat proses instalasi sistem operasi pertama tidak dapat mensupport sistem

TProxy, maka IP tables tersebut harus diganti menggunakan IP tables yang dapat mensupport TProxy . Oleh karena itu IP tables yang sudah terinstal pada sistem akan dihapus terlebih dahulu dengan menggunakan perintah #removepkg iptables seperti tertera pada gambar :

Gambar 4.21 Penghapusan paket IP Tables

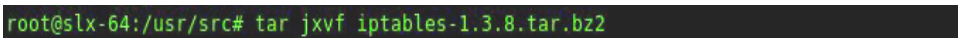
b. Memasuki direktori /usr/src

Paket IP tables yang mensupport TProxy akan diinstall pada direktori /usr/src bersama paket-paket lainnya yang dibutuhkan sistem dan telah dikompilasi dan terinstal sebelumnya, maka dibutuhkan untuk masuk ke direktori tersebut dengan menggunakan perintah #cd /usr/src seperti pada gambar :

Gambar 4.22 Direktori /usr/src

c. Ekstraksi IP tables

Paket IP tables yang terdapat pada direktori /usr/src masih merupakan paket dalam bentuk kompresi dan perlu diekstraksi terlebih dahulu untuk dapat melakukan proses selanjutnya dengan mengetikkan perintah pada console #tar jxvf iptables-1.3.8.tar.bz2 seperti terterra pada gambar.



Gambar 4.23 Ekstraksi IP Tables

d. Direktori IP tables

Agar dapat melakukan proses instalasi pada IP tables 1.3 yang telah diekstrak dan menjadi direktori, diperlukan untuk masuk ke dalam

direktori IP tables tersebut dengan menggunakan perintah #cd /iptables-1.3.8 seperti pada gambar.

Gambar 4.24 Direktori IP tables

e. Patch IP tables TProxy

Paket IP tables yang telah diekstraksi dalam folder /usr/src , perlu mengenali sistem TProxy yang terdapat dalam paket cttproxy, untuk dapat mengenali paket-paket tersebut maka diperlukan proses patching yang berfungsi sebagai pengenalan dan penggabungan paket-paket yang terdapat dalam IP tables dengan paket-paket yang terdapat dalam direktori cttproxy dengan menggunakan perintah #cat /usr/src/cttproxy-2.6.20-2.0.6/iptables/iptables-1.3-cttproxy.diff | patch –p1 dari dalam direktori IP tables yaitu #cd /usr/src/iptables-1.3.8 seperti pada gambar berikut ini.

Gambar 4.25 Patch IP tables TProxy

f. Mode TProxy-test

Di dalam direktori IP tables terdapat packet-paket yang digunakan untuk kebutuhan sistem TProxy, akan tetapi belum diaktifkan. Salah satunya adalah .tproxy-test yang masih berupa paket non-executeable, file tersebut diperlukan sistem TProxy pada saat start-up dalam bentuk executeable, oleh karena itu file .tproxy-test perlu dirubah sebagai file executeable dengan menggunakan perintah #chmod +x extensions/.tproxy-test seperti pada gambar.

Gambar 4.26 Mode TProxy-test

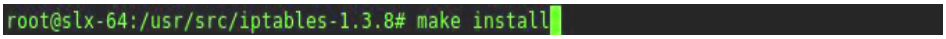
g. Kompilasi IP tables

IP tables yang akan diinstall perlu dikenalkan pada direktori /usr/src/linux dimana paket-paket kernel yang sebelumnya telah terinstall berada, kompilasi IP tables ini dapat dilakukan dengan menggunakan perintah #make KERNELDIR=/usr/src/linux seperti pada gambar:

Gambar 4.27 Kompilasi IP tables

h. Instalasi IP tables

Proses ini adalah proses dimana paket IP tables yang telah dikenalkan pada kernel cttproxy dan kernel linux akan diinstalasi sepenuhnya ke dalam sistem dengan menggunakan perintah #make install seperti pada gambar:



Gambar 4.28 Instalasi IP Tables

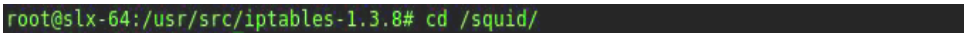
4.2.4. Tahapan Instalasi dan konfigurasi Squid Proxy

Squid merupakan daemon yang digunakan sebagai proxy server dan web cache, dalam penelitian ini package squid yang digunakan adalah squid 2.7 STABLE yang akan dikompilasi dan diinstalasi sesuai kebutuhan sistem, yaitu dapat mensupport sistem TProxy.

Berikut ini adalah tahapan dari proses instalasi squid :

a. Direktori /squid

Diakrenakan daemon squid yang dibutuhkan sistem akan diinstall pada direktori /squid, maka diperlukan memasuki direktori tersebut dengan menggunakan perintah #cd /squid seperti pada gambar.



Gambar 4.29 Direktori /squid

b. Ekstaksi paket squid

Di dalam direktori /squid diasumsikan telah terdapat paket squid-2.7.STABLE4 yang masih dalam bentuk kompresi dan perlu mengestrak terlebih dahulu untuk menggunakan paket-paket yang terdapat di dalamnya, dengan menggunakan perintah #tar zxvf squid-2.7.STABLE4 seperti pada gambar.

Gambar 4.30 Ekstraksi paket squid

c. Direktori /squid/squid-2.7.STABLE4

Untuk dapat mengeksekusi paket-paket squid yang telah diekstrak, maka perlu untuk berganti direktori dan masuk ke dalam direktori squid yang telah diekstrak pada proses sebelumnya, yaitu dengan mengetikkan perintah apda console #cd /squid/squid-2.7.STABLE4 seperti tertera pada gambar di bawah ini.

Gambar 4.31 Direktori /squid/squid-2.7.STABLE4

d. Kompilasi paket squid

Pada proses ini, akan dilakukan kompilasi pad paket-paket squid yang akan digunakan dan beberapa fitur yang diperlukan dan akan di gunakan dalam sistem, proses kompilasi tersebut tertera pada gambar di bawah ini.

Gambar 4.32 Kompilasi paket squid

e. Install squid

Proses instalasi paket-paket squid yang telah dikompilasi akan dilakukan pada sistem dan mengeluarkan output dari hasil instalasi berupa direktori-direktori yang diperlukan untuk kebutuhan sistem, adapun instalasi tersebut dapat dilakukan dengan menggunakan perintah #make install seperti pada gambar berikut ini.

Gambar 4.33 Install squid

f. Konfigurasi squid

Proses konfigurasi squid ini dilakukan dalam file squid.conf yang terdapat pada direktori /squid/sys/etc , file ini berupa file konfigurasi yang akan berpengaruh pada fitur-fitur yang akan dijalankan oleh sistem, adapun konfigurasi yang dibutuhkan untuk proses TProxy adalah sebagai berikut.

```
http_port 3182 tproxy transparent

acl CLIENT src 192.168.100.0/24

acl localhost src 127.0.0.1/255.255.255.255
```

```
tcp_outgoing_address 10.10.10.1
udp_incoming_address 0.0.0.0

maximum_icp_query_timeout 2000
hierarchy_stoplist cgi-bin ?

cache_mem 64 MB

redirect_rewrites_host_header off
emulate_httpd_log off

forwarded_for off

cache_swap_low 98%
cache_swap_high 99%

maximum_object_size 256 MB
minimum_object_size 0 KB
maximum_object_size_in_memory 256 KB

dns_nameservers 10.10.10.2

dns_children 50
ipcache_size 3840
ipcache_low 98
ipcache_high 99
fqdncache_size 3840

log_fqdn off
log_icp_queries off

cache_dir aufs /squid/cache/c1 4000  16 256
cache_dir aufs /squid/cache/c2 4000  16 256
cache_dir aufs /squid/cache/c3 4000  16 256


refresh_pattern /.gif 4320 50% 43200
refresh_pattern /.jpg 4320 50% 43200
refresh_pattern /.jpeg 4320 50% 43200
```

```
refresh_pattern /.png 4320 50% 43200
refresh_pattern ^http://mail.yahoo.com/.* 1440 100% 10080
refresh_pattern ^http://*.yahoo.*/.* 1440 100% 7200
refresh_pattern ^http://*.yimg.*/.* 720 100% 7200
refresh_pattern ^http://*.google.com/.* 720 100% 10080
refresh_pattern ^http://*.blogsome.com/.* 720 80% 10080
refresh_pattern ^http://*.wordpress.com/.* 720 80% 10080
refresh_pattern ^http://*.detik*.*/.* 1440 90% 2880
refresh_pattern ^ftp: 10080 95% 241920 reload-into-ims override-lastmod
refresh_pattern . 360 95% 120960 reload-into-ims override-lastmod

request_header_max_size 10 KB
request_body_max_size 10 MB

reload_into_ims on
pipeline_prefetch on
vary_ignore_expire on

quick_abort_min 0
quick_abort_max 0
quick_abort_pct 98

half_closed_clients off
shutdown_lifetime 30 seconds

acl all src 0.0.0.0/0.0.0.0

############ ACL ACCESS/BLOK ##############
acl UPDT dstdomain update.microsoft.com windowsupdate.microsoft.com
v5stats.windowsupdate.microsoft.com download.microsoft.com rs.update.microsoft.com
stats.upd
acl YMM dstdomain radio.music.yahoo.com
acl BLK dstdomain .windowsupdate.com
acl YOUTUBE dstdomain .youtube.com
acl TRADE dstdomain .forexfactory.com
acl manager proto cache_object

no_cache deny TRADE
```

```
acl SSL_ports port 443 563
acl Safe_ports port 80          # http
acl Safe_ports port 81          # 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 110         # pop
acl Safe_ports port 25          # smtp
acl Safe_ports port 210         # wais

acl CONNECT method CONNECT

http_access allow manager localhost
http_access deny manager !localhost


#http_access deny blacklist
# Deny requests to unknown ports
http_access deny !Safe_ports
# Deny CONNECT to other than SSL ports
#http_access deny CONNECT !SSL_ports
#acl pornsite url_regex "/squid/sys/etc/porn"


##########acces list##############
#http_access deny BLKDOMAIN
http_access allow CLIENT
http_access deny all

#################################

http_reply_access allow all
icp_access allow all
reply_body_max_size 0 allow all
cache_effective_user squid
cache_effective_group root
visible_hostname guswan.kristiawanto@google.com
via off
```

```
logfile_rotate 1
log_icp_queries yes

cache_store_log none
access_log /squid/sys/var/logs/access.log squid
store_avg_object_size 6 MB
store_objects_per_bucket 20

maximum_single_addr_tries 3


cache_mgr guswan.kristiawanto@google.com
cachemgr_passwd adm

coredump_dir /tmp
pipeline_prefetch on
ie_refresh on

pid_filename /squid/sys/var/logs/squid.pid
```

g. Group squid

Dikarenakan paket squid yang telah terinstall hanya dapat dijalankan oleh group tertentu dan kepemilikan tertentu, maka dibuatlah group squid pada sistem ini dengan menggunakan perintah #groupadd squid seperti pada gambar berikut ini.

```
root@slx-64:/squid/squid-2.7.STABLE4# groupadd squid
```

Gambar 4.34 Group squid

h. User squid

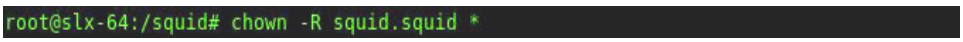
User ini digunakan untuk merubah kepemilikan paket-paket squid yang akan berjalan pada sistem dengan group squid yang telah dibuat sebelumnya, oleh karena keseluruhan dari paket squid terdapat pada direktori /squid , maka user squid akan diperuntukkan hanya pada

direktori /squid tersebut dengan group squid . Proses ini dilakukan dengan menggunakan perintah #useradd squid –g squid –d /squid seperti tertera pada gambar berikut ini.

Gambar 4.35 User squid

i. Perubahan kepemilikan

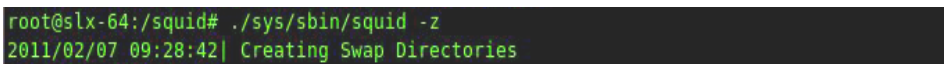
Agar proses squid berjalan dengan maksimal, maka diperlukan perubahan kepemilikan dari root menjadi kepemilikan dari user dan group terterntu yang khusus dibuat untuk direktori /squid tersebut. User dan group yang telah dibuat sebelumnya yaitu user squid dengan group squid akan menjadi pemilik dari direktori /squid secara keseluruhan dengan menggunakan perintah #chown –R squid.squid * , perintah tersebut dilakukan di dalam direktori /squid seperti tertera pada gambar.



Gambar 4.36 Perubahan kepemilikan

j. Membuat swap cache

Pada saat squid dieksekusi untuk berjalan pertama kali, swap directories yang akan menjadi direktori tempat berada cache dari data hasil browsing yang diperlukan pada sistem proxy tidak terbentuk sendiri, oleh karena itu untuk membentuk swap directories pada saat squid pertama kali dijalankan, dilakukan dengan mengetikkan perintah #./sys/sbin/squid –z , dan jika file konfigurasi squid.conf yang telah dikonfigurasi tidak ada kesalahan , maka akan terlihat keluaran tanpa pesan error seperti apda gambar berikut ini.



Gambar 4.37 Swap Cache

k. Menjalankan squid

Untuk menjalankan squid sebagai daemon proxy, setelah swap directories berhasil dibentuk dapat menggunakan perintah #./sys/sbin/squid pada direktori /squid seperti pada gambar berikut ini.

Gambar 4.38 Menjalankan squid

l. Logging squid

Pada daemon squid ini, terdapat fasilitas logging yang salah satu fungsinya adalah untuk merekam dan menampilkan alamat-alamat web yang sudah diakses maupun yang sedang diakses. Pada instalasi squid pertama kali log tersebut masuk ke dalam direktori /squid/sys/var/logs/ , agar log tersebut bergabung bersama log lainnya dalam sistem yang terdapat pada direktori /var/log/ maka digunakan perintah #ln –s /squid/sys/var/logs /var/log/squid , yang berarti dibuatkan link akses dari direktori /squid/sys/var/logs menuju direktori /var/log/squid seperti pada gambar berikut ini.

Gambar 4.39 Logging squid

4.2.5. Tahapan konfigurasi lanjutan

Tahapan konfigurasi lanjutan yang dilakukan pada sistem bertujuan untuk menjadikan sistem lebih baik dengan tambahan konfigurasi yang dilakukan pada Sistem Operasi Linux Slackware. Beberapa konfigurasi tambahan pada Sistem Operasi adalah sebagai berikut :

a. Penjadwalan rotasi partisi swap dari cache squid

Dikarenakan cache dari data hasil browsing penyimpanannya menyebar pada beberapa partisi swap yang telah dibuat, maka perlu penjadwalan yang mengeksekusi perotasian dari cache-cache tersebut

agar tersebar merata pada setiap partisi swap yang telah ditentukan, berikut adalah tahapan-tahapan pembuatan penjadwalan otomatis untuk merotasi cache tersebut pada partisi :

1. Export visual

   Maksud dari export visual disini adalah, membuat file cron agar dapat dibuka oleh tool mcedit yang terdapat pada sistem operasi, file cron tersebut menyimpan jadwal-jadwal otomatis yang telah dibuat oleh sistem secara default pada saat instalasi sistem operasi, proses pengeksportan tersebut dilakukan dengan menggunakan perintah #export VISUAL=mcedit seperti pada gambar berikut.

Gambar 4.40 Export Visual

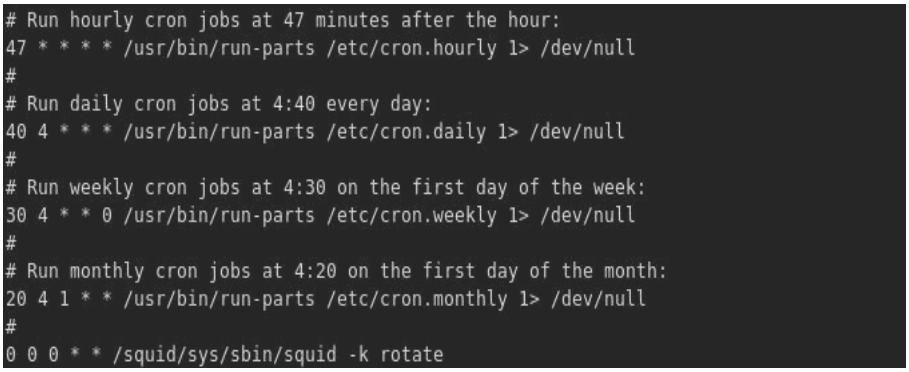
2. File cron

   Pembuatan jadwal rotasi otomatis dilakukan pada file cron yang di tambahkan syntax untuk merotasi cache squid pada partisi. Untuk dapat mengedit file cron tersebut, dilakukan dengan menggunakan perintah #crontab –e seperti pada gambar.

Gambar 4.41 Cron

3. Menambahkan jadwal rotasi squid

   Penambahan jadwal untuk merotasi secara otomatis cache squid yang terdapat pada partisi swap dapat dilakukan dengan menggunakan perintah pada console #/squid/sys/sbin/squid –k rotate. Maka diperlukan penambahan perintah itu di dalam file cron yang telah diekport melalui mcedit dengan ketentuan syntax eksekusi itu dijalankan setiap hari pada jam 00 menit 00 sesuai dengan setting waktu yang berjalan pada sistem operasi sepetri tertera pada gambar berikut ini.



Gambar 4.42 Penambahan jadwal pada cron

1. Direktif arus client menuju proxy

   Untuk mengarahkan arus client menuju TProxy , maka diperlukan filtering atau penampisan paket menuju TProxy dengan IP tables. Paket IP tables yang telah dikompilasi dan diinstall sebelmunya, tidak dengan sendirinya melakukan direktif arus menuju TProxy, IP tables hanya menyediakan fasilitas untuk penampisan paket-paket data, salah satunya adalah penampisan arus port 80 menuju TProxy. Hal ini memerlukan syntax yang dimiliki oleh IP tables untuk penampisan tersebut yaitu dengan menggunakan perintah sebagai berukut #iptables –t tproxy –A PREROUTING –i eth1 –p tcp --dport 80 –j TPROXY --on-port 3182 yang berarti seluruh arus yang melewati interface eth1 dengan jenis paket tcp yang mengakses port 80 akan dilewatkan menuju TPROXY yang memiliki port 3182 seperti pada gambar berikut ini.

Gambar 4.43 Direktif arus client menuju proxy

2. Konfigurasi rc.local

   File rc.local yang terdapat pada direktori /etc/rc.d/ merupakan suatu file yang dimana seluruh konfigurasi yang terdapat di dalamnya akan dieksekusi pada saat sistem operasi pertama dijalankan. Untuk menjaga pada saat toroubleshooting seperti pada saat server terpaksa harus melakukan restart sistem, maka perintah-perintah yang diperlukan dapat dieksekusi pada saat start up dan disimpan pada konfigurasi rc.local. Dalam sistem TProxy ini bebrapa hal yang perlu dieksekusi pada saat start

up adalah syntax untuk menjalankan proses squid dan syntax iptables yang melakukan direktif arus internet menuju TProxy seperti pada gambar berikut ini.

```
/squid/sys/sbin/squid

iptables -t tproxy -A PREROUTING -i eth1 -p tcp --dport 80 -j TPROXY --on-port 3182
```

Gambar 4.44 Konfigurasi rc.local

4.3 Pengujian Hasil Implementasi

Sebuah sistem yang berfungsi sebagai server dan digunakan untuk pengaplikasian TProxy, perlu memenuhi beberapa syarat utama yaitu :

a. Mendukung dan dapat menghasilkan output TProxy.
b. Dapat menjalankan daemon squid sebagai proxy.
c. Melakukan direktif arus klien menuju Internet yang dilewatkan menuju sistem proxy dengan menggunakan IP Tables.
d. Melakukan Logging data dari hasil browsing Internet yang disimpan dalam cache proxy.
e. Traffic klien yang dilewatkan melalui TProxy dapat di*shaping* dengan baik oleh bandwidth limiter.
f. Melakukan trigger untuk mengaktifkan TProxy secara otomatis pada saat server mengalami troubleshoot hard restart atau restart.

Oleh karena beberapa hal tersebut, maka perlu dilakukan pengujian untuk memenuhi kebutuhan sistem agar dipastikan dapat berjalan dengan baik sebelum diaplikasikan secara nyata dalam kondisi yang sesungguhnya. Yaitu dengan melaukan pengujian dengan melakukan beberapa hal berikut :

4.3.1 Pengujian keluaran TProxy

Pengujian keluaran TProxy sebagai komponen utama dalam sistem dan tolak ukur bahwa kompilasi kernel telah berhasil dilakukan dengan menggunakan

perintah # dmesg | grep TPROXY , dan akan menghasilkan keluaran seperti tertera pada gambar sebagai berikut.

```
root@slx-64:/usr/src/linux# dmesg | grep TPROXY
IP_TPROXY: Transparent proxy support initialized 2.0.6
```

Gambar 4.45 Pengujian keluaran TProxy

4.3.2 Pengujian Squid berjalan sebagai sistem proxy

Pada penelitian ini, dikarenakan menggunakan daemon squid sebagai sistem proxy yang diintegrasikan dengan kernel yang telah dikompilasi ulang dengan sistem TProxy, maka proses squid pada sistem harus dipastikan berjalan terlebih dahulu sebelum sistem diaplikasikan, pengujian bahwa proses squid yang telah diinstalasi sudah berjalan pada sistem yaitu dengan menggunakan perintah # pstree dan akan menghasilkan keluaran yang menunjukkan bahwa proses squid muncul pada proses tree sistem linux seperti tertera pada gambar berikut ini.

```
     |         |-khubd
     |         |-kmirrord
     |         |-kmmcd
     |         |-kpsmoused
     |         |-kseriod
     |         |-ksnapd
     |         |-ksuspend_usbd
     |         |-kswapd0
     |         |-2*[pdflush]
     |         |-reiserfs/0
     |         |-reiserfs/1
     |         |-scsi_eh_0
     |         |-scsi_eh_1
     |         |-scsi_eh_2
     |         |-scsi_eh_3
     |         |-scsi_eh_4
     |         |-scsi_tgtd/0
     |         |-scsi_tgtd/1
     |         |-usb-storage
     |         |-xfsdatad
     |         `-xfslogd
     |-migration/0
     |-migration/1
     |-3*[mount.ntfs-3g]
     |-ntpd
     |-rpc.portmap
     |-rpc.statd
     |-saslauthd---4*[saslauthd]
     |-2*[sendmail]
     |-snmpd
     |-squid
     |-sshd
     |-start_kdeinit
     |-syslogd
     `-udevd
root@slx-64:~#
```

Gambar 4.46 Prosess Tree

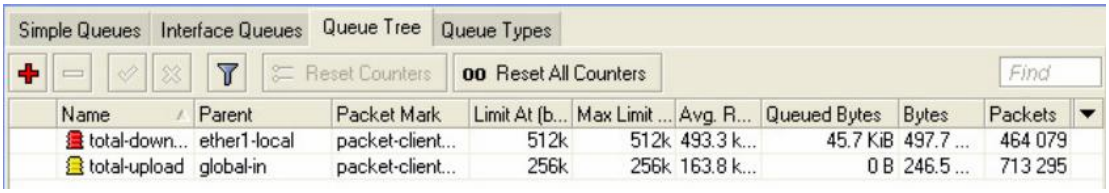
4.3.3 Logging data hasil browsing

Data hasil browsing yang akan direkam pada cache proxy akan terlihat pada hasil logging data browsing yang akan terlihat dengan menggunakan perintah berikut, # tail –f /var/log/squid/access.log seperti tertera pada gambar berikut.

Gambar 4.47 Prosses Logging

4.3.4 Prosess shaping pada bandwidth limiter

Proses shaping pada bandwidth limiter yaitu proses dimana pembatasan arus request klien menuju internet dapat dibatasi oleh sistem limiter, dimana proses limiter dengan topologi pada penelitian ini tidak dapat berjalan apabila sistem tidak menggunakan proses TProxy. Pembatasan bandwidth pada bandwidth limiter seperti tertera pada gambar berikut ini.



Gambar 4.48 Prosess Shaping

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Aplikasi TProxy telah selesai dibuat dan telah diupayakan sesuai dengan perencanaan serta perencanaan tahap sebelum implementasi yakni tahap analisis kebutuhan arsitektur jaringan, serta perangkat lunak dan perangkat keras, dengan tambahan beberapa keterbatasan yang ditemukan selama tahap pengujian, sehingga dapat diambil kesimpulan :

1. Kelebihan jaringan dengan arsitektur jaringan yang menggunakan sistem TProxy adalah tidak adanya limitasi bandwidth dari klien menuju proxy sehingga request dan proses pendistribusian cache relative lebih cepat.
2. Arsitektur jaringan yang menggunakan sistem TProxy akan lebih maksimal apabila distribusi menggunakan kabel dibandingkan dengan menggunakan perangkat wireless, dikarenakan tidak adanya limitasi bandwidth dari klien menuju proxy akan sedikit beresiko apabila menggunakan perangkat wireless yang memiliki kapasitas kecil. Limitasi bandwidth hanya berjalan pada saat request klien menuju internet.
3. Kekurangan dari sistem pada penelitian ini, adalah arsitektur jaringan yang merupakan satu rangkaian dan tidak dapat terpisah, sehingga apabila salah satu bagian dari rangkaian bandwidth limiter dan TProxy server mengalami gangguan, koneksi Internet akan terputus.

5.2 Saran

Saran yang dapat diberikan untuk penelitian selanjutnya adalah apabila akan menggunakan arsitektur jaringan dengan menggunakan sistem TProxy yaitu, dikarenakan kekurangan dari arsitektur jaringan yang mendukung sistem TProxy ini merupakan satu rangkaian yang dimana apabila salah satu dari perangkat utama mengalami gangguan maka jaringan akan terputus total, maka disarankan menggunakan topologi Full mesh pada sistem TProxy ini dengan menambah satu pasang perangkat lagi yaitu Bandwidth Limiter dan Satu Unit TProxy server sebagai redundansi dari sistem.

DAFTAR PUSTAKA

Squid, *Squid Optimising Web Delivery*

Avalaible at http://www.squid-cache.org/

Nurwijayanto, Iwan.2010. *Mengenal Proxy Server dan Fungsinya.*

Avalaible at http://www.klik-kanan.com/mengenal-proxy-server-dan-fungsinya.htm

Kernel, *Linuxdig Linux Dictionary*

Avalaible at http://www.linuxdig.com/documents/dictionary/terms/Kernel.php

Mitchell, Bradley. 1999. *Wireless/Networking*

Avalaible at http://compnetworking.about.com/od/networkdesign/g/network-gateway.htm

Iptables, *The netfilter.org "iptables" project*

Avalaible at http://netfilter.org/projects/iptables/index.html

Smitdev. 2008. *Daftar Istilah Komputer, IT Glosary, Definisi*

Avalaible at http://www.smitdev.com/posts/bandwidth87.php?g=75