UNIVERSITAS INDONESIA

RANCANG BANGUN DAN IMPLEMENTASI *POLICY SERVER* DAN RESPON *MANAGEMENT SYSTEM* PADA NAC *(NETWORK ADMISSION CONTROL)* UNTUK MENINGKATKAN KEAMANAN JARINGAN

SKRIPSI

NURSANTUSO
0706199741

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
UNIVERSITAS INDONESIA
DEPOK
JUNI 2009

UNIVERSITAS INDONESIA

RANCANG BANGUN DAN IMPLEMENTASI *POLICY SERVER* DAN RESPON *MANAGEMENT SYSTEM* PADA NAC *(NETWORK ADMISSION CONTROL)* UNTUK MENINGKATKAN KEAMANAN JARINGAN

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

NURSANTUSO
0706199741

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
UNIVERSITAS INDONESIA
DEPOK
JUNI 2009

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama : Nursantuso

NPM : 07 06 19 97 41

Tanda Tangan :

Tanggal : 17 Juni 2009

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :
Nama : Nursantuso
NPM : 0706199741
Program Studi : Teknik Elektro
Judul Skripsi : Rancang bangun dan Implementasi *Policy Server* dan Respon *Management System* pada NAC *(Network Admission Control)* Untuk Meningkatkan Keamanan Jaringan

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro, Fakultas Teknik, Universitas Indonesia

DEWAN PENGUJI

Pembimbing : Muhammad Salman ST, MIT (.....)

Penguji : Dr. Ir. Anak Agung Putri Ratna M.Eng (.....)

Penguji : Ir. Endang Sriningsih MT, Si (.....)

Ditetapkan di : Depok

Tanggal : 29 Juni 2009

KATA PENGANTAR

Puji syukur saya panjatkan kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Jurusan Teknik Elektro pada Fakultas Teknik Universitas Indonesia. Saya menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu, saya mengucapkan terima kasih kepada:

1. Muhammad Salman ST, MIT selaku dosen pembimbing yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan saya dalam penyusunan skripsi ini;
2. Orang tua dan keluarga saya yang telah memberikan bantuan dukungan secara moral,
3. Sahabat yang telah banyak membantu saya dalam menyelesaikan skripsi ini.

Akhir kata, saya berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Depok, 11 Juni 2009

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai civitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Nursantuso
NPM : 0706199741
Program Studi : Teknik Elektro
Departemen : Teknik Elektro
Fakultas : Teknik
Jenis karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul :

RANCANG BANGUN DAN IMPLEMENTASI *POLICY SERVER* DAN RESPON *MANAGEMENT SYSTEM* PADA NAC *(NETWORK ADMISSION CONTROL)* UNTUK MENINGKATKAN KEAMANAN JARINGAN

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/ formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan skripsi saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 17 Juni 1009
Yang menyatakan

(…….)

ABSTRAK

Nama : Nursantuso
Program Studi : Teknik Elektro
Judul : Rancang bangun dan Implementasi *Policy Server* dan Respon *Management System* pada NAC *(Network Admission Control)* Untuk Meningkatkan Keamanan Jaringan

Peningkatan kebutuhan akses internet menyebabkan peningkatan permintaan akses ke jaringan yang aman. Keadaan ini menuntut *administrator* jaringan agar lebih selektif dalam memperbolehkan *user* melakukan akses ke jaringan. Selain melakukan seleksi terhadap user, seorang *administrator* jaringan juga bertanggung jawab untuk melindungi jaringan dari gangguan yang dilakukan oleh *user* didalam jaringan atau dari luar jaringan. Ada beberapa teknologi yang bisa digunakan untuk memecahkan permasalahan diatas diantaranya adalah teknologi NAC.

NAC adalah teknologi keamanan jaringan komputer dimana *client* komputer harus melaporkan ”id” user sebelum diperbolehkan masuk kedalam jaringan. Teknologi NAC mempunyai 3 variasi yang berbeda yaitu: Cisco NAC (CNAC), NAP dan TNC. Pada skripsi ini akan dibahas tentang rancang bangun NAC *server* dengan fokus pembahasan di *policy server*. Design NAC *server* yang digunakan pada skripsi ini dibagi menjadi 2 bagian yaitu *policy server* dan IDS *server*. *Policy server* bertugas untuk melakukan authentifikasi terhadap *user* yang akan mengakses ke *network devices* jaringan. Selain melakukan authentifikasi, *server* ini juga bertugas untuk melakukan *reporting* kepada *administrator* jaringan bila terjadi gangguan pada jaringan melalui SMS.

Sistem *policy server* dibagi menjadi beberapa modul yaitu *User* Interface modul, Authentifikasi Modul, *Monitoring* Modul dan SMS Modul. Komunikasi antar modul dalam sistem menggunakan port TCP/IP. Pada bagian SMS modul, sistem ini terhubung langsung dengan sebuah modem Itegno yang bertugas mengirim pesan kepada *administrator* jaringan dan menerima perintah dari *administrator* jaringan sebagai reaksi terhadap adanya gangguan pada jaringan. Sementara *network devices* yang digunakan pada arsitektur jaringan ini adalah sebuah *switch* dan *router*.

Pengujian sistem dilakukan pada setiap modul dengan skenario yang berbeda. Jaringan yang digunakan untuk pengujian adalah jaringan lokal berskala kecil. Dari hasil pengujian, sistem bekerja dengan baik pada interval pengiriman command lebih dari 1 detik. Tingkat keberhasilan sistem dalam mengirimkan pesan adalah 88.24% dengan time proses 5 detik, sementara tingkat keberhasilan sistem dalam menerima dan menjalankan *command* yang diberikan melalui sms adalah 75% dengan *time* proses 2 detik.

Kata Kunci : NAC, *Policy Server*, SMS

ABSTRACT

Name : Nursantuso
Study Program: Electrical Engineering
Title : Design and Implementation of Policy Server and Response Management System on Network Admission Control (NAC) for Improving the Network Security

Improvements of requirement access internet cause improvement of request access to secure network. This situation claim administrator network to be more selective to enable user access to network. Besides selecting user, an administrator network also has responsibility to protecting network from another trouble user in network or from another user from the outside of network. There are some technologies can be used to solve this problem, for example technology NAC.

NAC is computer network security technology where client computer have to report "id" user before enabled enter into network. NAC have 3 different variations that are: Cisco NAC (CNAC), NAP and TNC. This project will be study about design and implementation of NAC server, especially in policy server and response management system. NAC server design that used at this project divided becomes 2 shares that are policy server and IDS server. Policy server undertakes to do user authentication before user access into network devices. Besides doing user authentication, policy server also undertake to do reporting to administrator network when happened the trouble in network through SMS.

Policy server system divided becomes some modules that are User Interface module, Authentifikasi Module, Monitoring Module and SMS Module. Communications between modules in system use the port TCP/IP. SMS module connected with a modem Itegno that have functions to deliver the message to administrator network and accept command from administrator network as reaction of intrusion in network. Network devices that are used in this network architecture are a switch and router.

System test performed in each module with different scenario. System test used a small local area network. From test result, system works in maximum performance when interval of sending command is more than 1 second. Level of achievement system when delivery message are 88.24% with 5 seconds of time processing and 75% for receive and execute command with 2 seconds of time processing.

Key Words: NAC, *Policy Server*, SMS

DAFTAR ISI

DAFTAR LAMPIRAN

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SINGKATAN

DHCP	Dynamic Host Configuration Protocol
IPv4	Internet Protocol version 4
NAC	Network Admission Control
PDU	Protocol Data Unit
TCP/IP	Transmission Control Protocol / Internet Protocol

BAB 1
PENDAHULUAN

1.1. LATAR BELAKANG

Kebutuhan akan akses internet dewasa ini sangat tinggi, hal ini mengakibatkan peningkatan permintaan akses masuk ke jaringan. Namun keterbatasan akses yang bisa disediakan oleh *server* menimbulkan masalah lain, karena tidak semua orang bisa masuk ke jaringan yang sama. Keadaan ini juga mengakibatkan *server* harus bersikap selektif terhadap *client* yang diperbolehkan untuk masuk ke suatu jaringan. *Client* disini merupakan piranti yang berhubungan langsung dengan *user*, misalkan *Personal Computer* (PC), *notebook* dll. Seleksi terhadap *client* yang akan masuk ke jaringan dapat dilakukan dengan proses authentifikasi *user*, melalui *user* dan *password*, atau dapat juga dilakukan dengan melakukan seleksi ”id” dari *client* yang akan mengakses suatu jaringan.

Setelah sebuah *client* masuk ke suatu jaringan, ada masalah lain yang muncul yaitu masalah security terhadap serangan *virus*, *worms*, pencurian data dll. Kembali *server* memiliki peranan yang besar untuk mengatasi hal ini yaitu dengan melakukan *monitoring* jaringan. Dengan demikian *server* kita dapat memberikan perlindungan terhadap *client* yang ada pada jaringan kita.

Untuk mengatasi keadaan diatas, maka dibuatlah arsitektur jaringan berbasis NAC *(Network access Control)*. NAC adalah teknologi keamanan jaringan komputer dimana *client* komputer harus melaporkan ”id” user sebelum boleh masuk kedalam jaringan [3]. Teknologi NAC mempunyai 3 variasi yang berbeda yaitu: Cisco NAC (CNAC), NAP dari *Microsoft* dan *Trusted Network Connect* dari konsorsium TCG. Pada perkembangannyan anggota-anggota NAP dan TNC sedang bekerja sama dalam organisasi IETF untuk menggabung NAP dan TNC.

Tujuan utama dari NAC adalah mengatur *client* pada jaringan kita, namun proteksi pada jaringan tidak cukup dengan hanya mengatur *client* jaringan kita namun juga dibutuhkan sebuah sistem yang bisa melakukan *monitoring* jaringan bila ada gangguan pada jaringan. Oleh karena itu pada skripsi ini akan dibangun jaringan NAC sederhana menggunakan perangkat CISCO. Pada skripsi ini

jaringan NAC yang dibuat dibagi menjadi dua bagian yaitu *policy server*, IDS *server*.

Policy server bertugas untuk melakukan authentifikasi terhadap *client* pada jaringan yang terhubung dengan *network access* jaringan. Selain melakukan authentifikasi, *server* ini juga bertugas untuk melakukan *reporting* kepada *administrator* jaringan bila terjadi gangguan pada jaringan.

1.2. Tujuan

Tujuan dari penulisan skripsi ini adalah merancang dan mengimplementasikan sistem *policy server* dan respon *management system* pada jaringan NAC sederhana menggunakan perangkat CISCO. Dengan penambahan fitur yang dimasukkan pada *policy server* diharapkan konsep jaringan NAC yang dibuat dapat diterapkan pada jaringan yang lain.

1.3. Pembatasan Masalah

Ada beberapa modul yang digunakan pada design *policy server* yang dibuat, yaitu modul authentifikasi, modul *reporting* dan modul *monitoring* jaringan. Pada modul authentifikasi akan dilakukan pengecekan terhadap *username* dan *password user* yang akan mengakses jaringan. *Monitoring* jaringan yang dimaksudkan disini adalah *monitoring* perangkat jaringan dari *policy server*, dengan modul ini *administrator* diharapkan lebih mudah melihat *client* yang melakukan akses pada jaringan.

Bila terjadi gangguan pada jaringan modul *reporting* akan mengirimkan pesan kepada *administrator* jaringan melalui SMS. Indikasi adanya gangguan pada jaringan berdasarkan pada informasi yang dikirimkan oleh IDS *server* melalui socket TCP/IP. Selain melakukan *reporting* kepada *user*, modul ini juga bertugas untuk melakukan eksekusi *command* tertentu pada *server* berdasar instruksi yang diberikan oleh *administrator* melalui SMS.

Modul – modul *policy server* dibuat menggunakan platfrom Linux, menggunakan beberapa bahasa pemrograman, yaitu C, perl dan PHP. Sementara perangkat jaringan yang digunakan adalah *switch* CISCO 2950 dan router CISCO

2611. Untuk melakukan *reporting* kepada *administrator* jaringan, *policy server* menggunakan modem Itegno 3000 untuk mengirim dan menerima SMS.

1.4. Metodologi Penelitian

Metode yang dilakukan pada skripsi ini adalah membuat *policy server* pada jaringan sederhana dimana terdiri dari *server*, *network device* dan *client*. Pengujian dilakukan pada setiap modul yang digunakan oleh *policy server* secara intensif sehingga bisa diperoleh data – data yang kemudian digunakan untuk menganalisa sistem yang sudah dibuat.

1.5. Sistematika Penulisan

Sistematika penulisan pada skripsi ini dibuat menjadi 5 bab. Bab 1 berisi tentang latar belakang, tujuan dan batasan masalah yang dibahas dalam skripsi kali ini. Bab 2 berisi study literature tentang teori – teori yang berhubungan dengan jaringan komputer dan komponen – komponen yang digunakan pada pembuatan *policy server*. Bab 3 merupakan pembahasan tentang design dari modul *policy server*. Bab 4 berisi tentang analisa dan pengujian modul *policy server* yang sudah dirancang pada BAB III. Dan pada bab 5 berisi kesimpulan dari hasil analisa pada BAB IV.

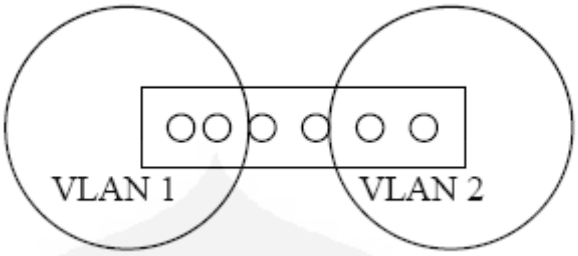
BAB 2
KOMPONEN SISTEM *POLICY SERVER*

2.1. *NETWORK ADMISSION CONTROL*

NAC *(Network Admission Control)* adalah teknologi keamanan jaringan komputer dimana *client* komputer harus melaporkan ”id” user sebelum boleh masuk kedalam jaringan. Teknologi NAC mempunyai 3 variasi yang berbeda yaitu: Cisco NAC (CNAC), NAP dari Microsoft dan *Trusted Network Connect* dari konsorsium TCG. Pada perkembangannyan anggota-anggota NAP dan TNC sedang bekerja sama dalam organisasi IETF untuk menggabung NAP dan TNC [3].

Sistem keamanan CNAC berkerja pada layer data link dan network pada OSI Layer dengan cara melakukan *filtering* terhadap MAC address *client* untuk melakukan management *user* yang mengakses ke jaringan kita. Keuntungan dari penerapan konsep ini adalah dapat digunakan dalam lingkungan jaringan yang besar dan dapat mengurangi jumlah *server* yang melakukan management pada jaringan. Proses management jaringan dilakukan pada *network devices* (misal: *switch, router*) pada saat *user* yang melakukan *login* ke jaringan dengan cara melakukan *switching* pada segment jaringan.

Network devices yang sering digunakan pada jaringan CNAC adalah *Switch* CISCO. Perangkat ini bekerja pada layer 2 berfungsi untuk membagi-bagi segment pada network. Pada teknologi *switch* proses *segmentasi* dilakukan menggunakan VLAN. VLAN adalah suatu metode yang memisahkan segmen-segmen pada *switch*, dimana antara 1 segmen dengan segmen lain tidak dapat terkoneksi, koneksi dapat dilakukan dengan menggunakan *router*. Dalam sebuah *switch* yang sudah de-segmentasi mempunyai *network-id* dan *broadcast domain* yang berbeda. Keuntungan lain dari proses *segmentasi* ini adalah pada saat proses pengiriman packet, pengiriman packet tidak lagi di-*broadcast* ke semua *client* jaringan yang terhubung di *switch* tetapi hanya di *broadcast* ke satu segment saja.



Gambar 2.1 Konsep VLAN

Pada gambar diatas adalah ilustrasi penggunaan VLAN pada sebuah *switch*. Dengan konsep tersebut kita seolah – olah mempunyai 2 buah jaringan yang berbeda pada sebuah *switch*. Ada beberapa metode *switching* dalam jaringan komputer yaitu:

- *Cut Through* , pada metode ini *frame* diperiksa sampai *field destination*. *cut through* terdiri dari 2 jenis yaitu *fast forward* yaitu metode *switch*ing dimana *latency* paling kecil karena paket data akan langsung dikirim begitu *destination address* diketahui. Tidak ada pengecekan *error* pada paket sehingga paket yang *error* pun akan tetap dikirim. *Fragment - free* merupakan metode *switching* dimana paket data baru akan dikirim bila hasil pemeriksaan awal dari 64 *byte* pertama tidak menunjukkan adanya *error*.
- *Store & Forward* adalah metode *switching* dimana paket data baru akan dikirimkan bila hasil perhitungan CRC menunjukan paket tersebut adalah paket data yang valid dengan kata lain,harus diperiksa sampai *field* FCS dulu baru bisa dikirim.
- *Adaptive Cut Through* merupakan pengembangan dari metode *cut trough* dimana paket data akan dikirimkan secara *cut trough*. Namun bila terjadi kesalahan pada paket yang dikirim sampai batas jumlah tertentu *(threshold),* maka metode *switching* akan diubah menjadi *store and forward*.

Pada perkembangannya *switch* CISCO dapat diberikan IP. IP ini diberikan bukan kepada setiap interface *Fast Ethernet* yang dimilikinya, namun kepada interface VLAN default, yaitu VLAN (Virtual LAN) merupakan fitur *switch* yang

memungkinkan *client* untuk tergantung ke dalam sebuah LAN, tanpa terbatas pada ruangan, atau gedung. *Switch* mempunyai default VLAN, yaitu VLAN1 milik *switch* sudah mempunyai anggota yaitu semua port - port yang ada pada *switch*. VLAN yang diberikan IP disebut sebagai management VLAN. Fungsi pemberian IP pada *switch* bertujuan agar *switch* tersebut bisa berkomunikasi dengan network lain.

Disamping melakukan proses *switching* sebuah *server* juga bisa melakukuan proses *assigning* IP kepada *client* secara otomatis. Proses pengalokasian IP *user* oleh *server* dilakukan oleh DHCP *server*. Selain melakukan tugas untuk authentifikasi *server* juga bisa melakukan *monitoring* jaringan dan *reporting* kepada admn jaringan menggunakan alert – alert tertentu, misalkan menggunakan SMS bila terjadi suatu gangguan terhadap jaringan.

2.2. DHCP *SERVER*

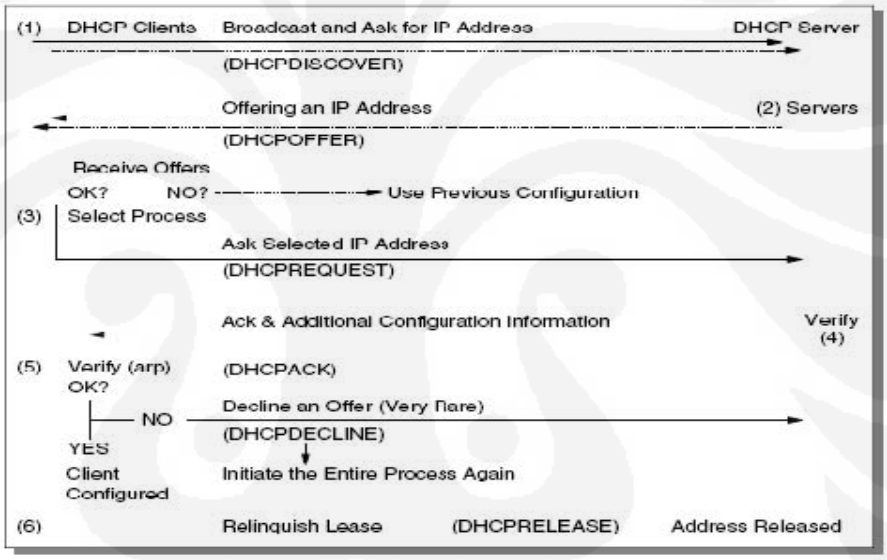
DHCP memberikan *framework* untuk disampaikan kepada *client* yang berisikan informasi tentang konfigurasi jaringan. DHCP bekerja berdasarkan protokol BOOTP, dimana ditambahkan fungsi untuk mengalokasikan penggunaan IP address dan konfigurasi jaringan lainnya.

Spesifikasi DHCP dapat dilihat pada RFC 2131 tentang *Dynamic Host Configuration Protocol*, dan RFC 2132 tentang DHCP options and BOOTP vendor *extension*[6]. DHCP melakukan transaksi dengan melihat pada jenis pesan yang dikirimkan. Pesan-pesan tersebut antara lain:

- DHCPDISCOVER : *broadcast* oleh *client* untuk menemukan *server*
- DHCPOFFER : respon dari *server* karena menerima DHCPDISCOVER dan menawarkan IP address kepada *client*
- DHCPREQUEST : pesan dari *client* untuk mendapatkan informasi jaringan
- DHCPACK : *acknowledge* dari *server*
- DHCPNACK : negatif *acknowledge* dari *server* yang menyatakan waktu sewa dari *client* sudah kadaluwarsa
- DHCPDECLINE : pesan dari *client* yang menyatakan bahwa dia sedang menggunakan informasi dari *server*

- DHCPRELEASE : pesan dari *client* bahwa *client* sudah tidak menggunakan lagi informasi dari *server*
- DHCPINFORM : pesan dari *client* bahwa dia sudah menggunakan informasi jaringan secara manual.

Proses pengalokasian IP pada *client* dapat dilihat pada Gambar 2.2. Pada gambar tersebut *client* diasumsikan belum memiliki IP address dan DHCP *server* memiliki 1 blok alamat jaringan dimana dapat digunakan pada jaringan tersebut. Setiap *server* memiliki sebuah *database* yang berisikan info IP address dan sewa *(leases)* penggunaan jaringan pada suatu tempat penyimpanan yang permanent.



Gambar 2.2 Proses Pemberian IP oleh DHCP *Server* [6]

Keterangan proses yang terjadi pada gambar diatas adalah sebagai berikut:

(1) *Client* melakukan *broadcast* DHCPDISCOVER pada jaringan lokal.

(2) *Server* merespon dengan pesan DHCPOFFER, dimana informasi ini juga memberikan informasi tentang alamat IP.

(3) DHCP *client* menerima 1 atau lebih pesan DHCPOFFER dari 1 atau lebih DHCP *server*. *Client* memilih salah satu informasi itu dan mengirimkan pesan DHCPREQUEST dan informasi jaringan mana yang dipilih.

(4) *Server* menerima pesan DHCPREQUEST tersebut dan membalas dengan mengirim pesan DHCPACK dengan mengirimkan informasi lengkap.

(5) *Client* menerima DHCPACK dan melakukan konfigurasi terhadap *interface* jaringan.

(6) Apabila *client* sudah tidak menginginkan lagi alamat IP tersebut, *client* akan mengirimkan pesan DHCPRELEASE.

2.3. LAMP

LAMP, adalah kependekan dari Linux + Apache + MySQL + PHP, merupakan sebuah paket perangkat lunak untuk menjalankan website dinamik dan sebagai sebuah web *server*[10]. LAMP ini terdiri dari beberapa komponen yang kesemuanya termasuk ke dalam keluarga *open source*. Yaitu Linux sebagai sistem operasinya, Apache sebagai *web server*-nya, MySQL sebagai *database*-nya, dan PHP sebagai bahasa pemrogramannya.

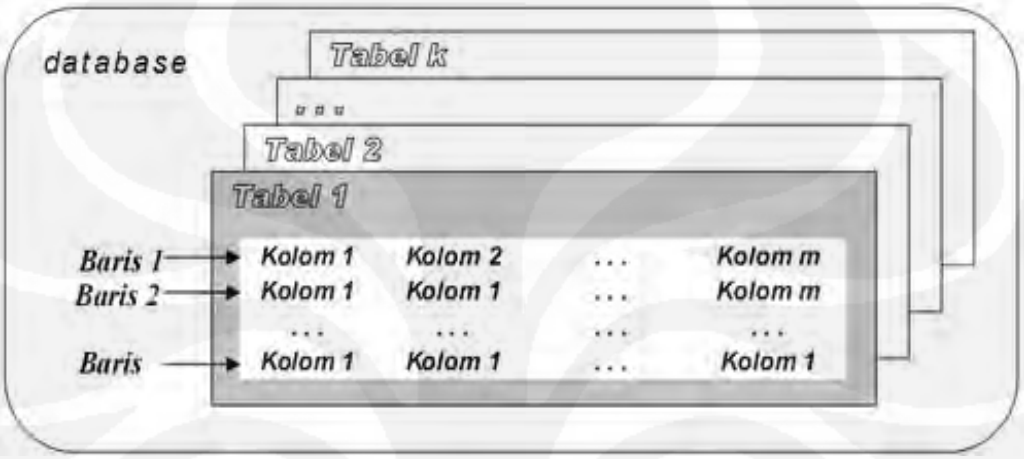
2.3.1. MySQL

MySQL adalah salah satu jenis *database server* yang sangat terkenal dan banyak digunakan untuk membangun aplikasi web yang menggunakan *database* sebagai sumber dan pengelolaan datanya [10]. Kepopuleran MySQL antara lain disebabkan karena MySQL menggunakan SQL sebagai bahasa dasar untuk mengakses *database* sehingga mudah untuk digunakan, cepat secara kinerja query, dan mencukupi untuk kebutuhan *database* perusahaan-perusahaan skala menengah-kecil. Selain itu MySQL juga bersifat *open source* dan *free* pada berbagai *platform* (kecuali pada Windows, yang bersifat *shareware*). MySQL didistribusikan dengan lisensi *open source* GPL *(General Public License)* mulai versi 3.23, pada bulan Juni 2000. *Software* MySQL bisa di-download di *"http://www.mysql.org"* atau *"http://www.mysql.com"*.

MySQL merupakan *database* yang pertama kali didukung oleh bahasa pemrograman *script* untuk internet (PHP dan Perl). MySQL dan PHP dianggap sebagai pasangan *software* pengembangan aplikasi web yang ideal. MySQL lebih sering digunakan untuk membangun aplikasi berbasis web, umumnya pengembangan aplikasinya menggunakan bahasa pemrograman *script* PHP.

MySQL termasuk RDBMS *(Relational Database Management* Sistem*)*. Itulah sebabnya istilah tabel, baris, dan kolom digunakan pada MySQL. Pada

MySQL, sebuah *database* mengandung satu atau sejumlah tabel. Tabel terdiri atas sejumlah kolom dan baris, dimana setiap kolom berisi sekumpulan data yang memiliki tipe yang sejenis, dan baris merupakan sekumpulan data yang saling berkaitan dan membentuk informasi. Kolom biasanya juga disebut sebagai *field* dan informasi yang tersimpan dalam setiap baris disebut dengan *record*.



Gambar 2.3. Struktur Data Dalam *Database* MySQL [10]

Software MySQL secara default akan diletakkan pada direktori "*/var*" dan "*/usr*" jika di-install pada sistem operasi Linux. Direktori yang penting dalam struktur direktori MySQL adalah direktori "*/var/lib/mysql*" dan "*/usr/bin*". Sub-direktori "*/usr/bin*" merupakan direktori yang menyimpan semua program *database* MySQL, sedangkan sub-direktori "*/var/lib/mysql*" digunakan untuk menyimpan data dan file-file yang dibutuhkan oleh MySQL untuk menyimpan *database*. Setiap *database* MySQL dibuatkan sebagai sebuah direktori didalam sub-direktori "*/var/lib/mysql*" ini. Tipe data yang digunakan pada MySQL dapat dilihat pada tabel berikut:

Tabel 2.1. Tipe Data Pada MySQL [10]

TIPE DATA	UKURAN	KETERANGAN
TINYINT	1 byte	Nilai integer yg sangat kecil
SMALLINT	2 bytes	Nilai integer yang kecil
MEDIUMINT	3 bytes	Integer dengan nilai medium
INT	4 bytes	Integer dengan nilai standar
BIGINT	8 bytes	Integer dengan nilai besar
FLOAT	4 bytes	Bilangan desimal dengan single-precission
DOUBLE	8 bytes	Bilangan desimal dengan double-precission
DECIMAL(M,D)	M bytes (D+2, if M < D)	Bilangan float (desimal) yang dinyatakan sebagai string
CHAR(M)	M bytes, 1 <= M <= 255	String karakter dengan panjang yang tetap
VARCHAR(M)	L+1 bytes, L <= M and 1 <= M <= 255	String karakter dengan panjang yang tidak tetap
TINYBLOB	L+1 bytes, L < 2^8	BLOB (Binary Large Object) yang sangat kecil
BLOB	L+1 bytes, L < 2^16	BLOB berukuran kecil
MEDIUMBLOB	L+1 bytes, L < 2^24	BLOB berukuran sedang
LONGBLOB	L+1 bytes, L < 2^32	BLOB berukuran besar
TINYTEXT	L+1 bytes, L < 2^8	String teks yang sangat kecil
TEXT	L+1 bytes, L < 2^16	String teks berukuran kecil
MEDIUMTEXT	L+1 bytes, L < 2^24	String teks berukuran medium
LONGTEXT	L+1 bytes, L < 2^32	String teks berukuran besar
ENUM('v1','v2',...)	1 or 2 bytes, (65535 values max)	Enumerasi, kolom dapat diisi dengan 1 member enumerasi
SET('val1','val2',...)	1, 2, 3, 4 or 8 bytes, (64 max)	Himpunan, kolom dapat diisi dengan beberapa nilai anggota himpunan
DATE	3 bytes	"1000-01-01" sampai "9999-12-31"
TIME	3 bytes	"-832:59:59" sampai "838-59:59"
DATETIME	8 bytes	"1000-01-01 00:00:00" sampai "9999-12-31 23:59:59"
TIMESTAMP	4 bytes	Range: 19700101000000 (suatu nilai tanggal pada tahun 2037
YEAR	1 byte	1901 sampai 2155
NULL		Nilai kosong (hampa)

2.3.2. Web *Server*

Web *server* merupakan *server* internet yang mampu melayani koneksi transfer data dalam protokol HTTP [11]. Web *server* merupakan hal yang terpenting dari *server* di internet dibandingkan *server* lainnya seperti *mail server*, ftp *server* ataupun *news server*. Hal ini disebabkan web *server* telah dirancang

untuk dapat melayani beragam jenis data, dari teks sampai grafis 3 dimensi. Kemampuan ini telah menyebabkan berbagai institusi seperti universitas maupun perusahaan dapat menerima kehadirannya dan juga sekaligus menggunakannya sebagai sarana di internet.

Web *server* juga dapat menggabungkan dengan dunia mobile *wireless* internet atau yang sering disebut sebagai WAP *(Wireless Access Protocol)*, yang banyak digunakan sebagai sarana *handphone* yang memiliki fitur WAP. Dalam kondisi ini, web*server* tidak lagi melayani data file HTML tetapi telah melayani WML *(Wireless Markup Language)*.

Salah satu *software* yang biasa digunakan oleh banyak web master di dunia adalah apache. *Software* tersebut dapat kita download secara gratis dari web resmi apache, yaitu *"http://www.apache.org"*. Dalam Penggunaannya Apache merupakan *software open source* yang sekarang ini sudah merebut pasar dunia lebih dari 50%. Web *server* ini *fleksibel* terhadap berbagai sistem operasi seperti windows9x/NT ataupun unix/linux.

Apache merupakan turunan dari web*server* yang dikeluarkan oleh NCSA yaitu NCSA HTTPd pada sekitar tahun 1995. Kelebihan web *server* Apache :

- *Freeware*
- Mudah di-*install*.
- Mampu beroperasi pada berbagai *platform* sistem operasi.
- Mudah mengkonfigurasinya.
- Apache Web *server* mudah dalam menambahkan periferal lainnya ke dalam *platform* web *server*nya, misalnya: untuk menambahkan modul, cukup hanya menset file konfigurasinya agar mengikutsertakan modul itu ke dalam kumpulan modul lain yang sudah dioperasikan.
- Dapat dijadikan pengganti bagi NCSA web *server*.
- Perbaikan terhadap kerusakan dan *error* pada NCSA 1.3 dan 1.4
- Merespon *client* lebih cepat daripada *server* NCSA.
- Mampu di kompilasi sesuai dengan spesifikasi HTTP yang sekarang.
- Kita dapat men-set respon *error* yang akan dikirim web *server* dengan menggunakan file atau *skript*.

- Secara otomatis menjalankan file index.html, halaman utamanya, untuk ditampilkan secara otomatis pada *client*nya.
- Lebih aman karena memiliki level-level pengamanan
- Apache mempunyai komponen dasar terbanyak di antara web *server*-web *server* lain, yang berarti bahwa web *server* Apache termasuk salah satu dari web*server* yang lengkap.
- Performansi dan konsumsi sumberdaya *(resource)* dari web*server* apache tidak terlalu banyak, hanya sebesar 20 MB untuk file-file dasarnya dan setiap daemonnya hanya memerlukan sebesar 950 KB memory per-child.
- Mendukung transaksi yang aman *(secure transaction)* menggunakan SSL *(Secure Socket Layer)*.
- Mempunyai dukungan teknis melalui web.
- Mempunyai kompatibilitas platform yang tinggi.
- Mendukung *third party* berupa modul-modul tambahan.

2.3.3. PHP

PHP adalah singkatan dari *"PHP: Hypertext Preprocessor"*, yang merupakan sebuah bahasa *scripting* yang terpasang pada HTML [12]. Sebagian besar sintaks mirip dengan bahasa C, Java dan Perl, ditambah beberapa fungsi PHP yang spesifik. Tujuan utama penggunaan bahasa ini adalah untuk memungkinkan perancang web menulis halaman web dinamik dengan cepat.

Kelebihan PHP adalah dapat berjalan di berbagai sistem operasi seperti windows 98/NT, UNIX/LINUX, solaris maupun macintosh. PHP merupakan *software* yang *open source* yang dapat anda *download* secara gratis dari situs resminya yaitu "*http://www.php.net"*, ataupun dari situs-situs yang menyediakan *software* tersebut seperti di *"ftp://gerbang.che.itb.ac.id"*.

Software ini juga dapat berjalan pada web *server* seperti PWS *(Personal Web Server)*, Apache, IIS, AOL *Server*, fhttpd, phttpd dan sebagainya. PHP juga merupakan bahasa pemograman yang dapat kita kembangkan sendiri seperti untuk menambah fungsi-fungsi baru. Keunggulan lainnya dari PHP adalah bahwa PHP juga mendukung komunikasi dengan layanan seperti protokol IMAP, SNMP, NNTP, POP3 dan bahkan HTTP.

PHP dapat diinstal sebagai bagian atau modul dari apache web *server* atau sebagai CGI *script* yang mandiri. Banyak keuntungan yang dapat diperoleh jika menggunakan PHP sebagai modul dari apache, di antaranya adalah :

- Tingkat keamanan yang cukup tinggi.
- Waktu eksekusi yang lebih cepat dibandingkan dengan bahasa pemograman web lainnya yang berorientasi pada *server-side* scripting.
- Akses ke sistem *database* yang lebih fleksibel. seperti MySQL.

2.3.4. Instalasi LAMP

Dalam penerapannya, LAMP tidak dalam satu bendel, artinya tiap komponen itu terpisah. Jadi berdiri sendiri-sendiri. Baik linux, apache, mysql dan php-nya berdiri sendiri. Jadi keempatnya diinstall secara terpisah, setelah terinstall barulah dikonfigurasi supaya dapat berjalan beriringan. Walaupun ada yang sudah dalam satu bendel, jadi apache, mysql dan php (minus linux) sudah dalam satu paket, tinggal menginstall satu paket sudah terkonfigurasi semuanya.

Banyak distribusi linux sekarang sudah menyertakan paket LAMP, jadi dengan menginstall distro linux tersebut, semua paket sudah terinstall, tanpa perlu mengkonfigurasi lagi. Namun ada juga yang belum terinstall. Jadi terpaksa apache, mysql dan php-nya diinstall secara manual[9].

Di dalam pendistribusiannya, apache, mysql dan php terdiri dalam beberapa macam paket. Ada yang dalam versi sourcenya (tar.gz/tar.bz2), ada yang dalam versi rpm (untuk distribusi linux keluarga red hat), tgz (untuk distro keluarga slackware), deb (untuk distro keluarga debian), dll. Kali ini akan dibahas cara menginstall baik apache, mysql, dan php ke dalam distro Debian, dan menggunakan versi deb-nya. Adapun langkah-langkahnya sebagai berikut :

- untuk menginstall mysql

```
# aptitude install mysql-server mysql-client
# /usr/bin/mysqladmin -u root password ‘masukkan password anda’
```

- untuk menginstall Apache dan php

```
# aptitude install apache2 apache2-doc
# aptitude install php5 php5-mysql libapache2-mod-php5
```

- Install kan mysql berbasis web base yakni phpmyadmin :

```
# aptitude install phpmyadmin
```

kemudian edit pada apache2.conf dan tambahkan *script* :

```
Include /etc/phpmyadmin/apache.conf
```

Kemudian restart services apache

```
# /etc/init.d/apache2 restart
```

Untuk melakukan pengecekan servis mysql apakah sudah terintegrasi dengan php atau belum. Dapat dilakukan dengan cara membuat sebuah file php di */var/www/* dengan nama tes.php. Listing programnya adalah :

```
<?php
        $connect=mysql_connect(”localhost”,”root”,””);
        if($connect){
                echo “sukses”;
        }else{
                echo “gagal”;
        }
?>
```

Setelah disimpan, kemudian buka di browser dengan alamat: “*http://localhost/tes.php”*. Jika tampil tulisan sukses maka php dan mysql sudah terintegrasi dengan benar. Jika muncul tulisan gagal, maka php dan mysql belum terintegrasi dengan benar.

2.4. KONSEP PENGIRIMAN SMS

Pengiriman sms menggunakan menggunakan sebuah modem dengan cara melakukan mengirimkan AT *command* kepada modem. Pengiriman AT *command* diikuti dengan pengiriman data PDU dari sms. Pada prinsipnya terdapat dua mode untuk mengirim dan menerima SMS, yaitu mode teks dan mode PDU (*Protocol Data Unit*). Sistem mode teks tidak didukung oleh semua operator GSM maupun terminal yang ada. Pada mode teks, pesan yang dikirim tidak dirubah. Teks yang dikirim tetap dalam bentuk aslinya dengan panjang mencapai 160 (7 bit *default*

alphabet) atau 140 (8 bit) karakter. Sesungguhnya mode teks adalah hasil *encoding* yang direpresentasikan dalam format PDU.

PDU mode adalah format *message* dalam heksadesimal *octet* dan *semi-decimal octet* dengan panjang mencapai 160 (7 bit *default alphabet*) atau 140 (8 bit) karakter. Data yang mengalir ke/dari SMS-Center harus berbentuk PDU (*Protocol Data Unit*). PDU berisi bilangan-bilangan heksadesimal yang mencerminkan bahasa I/O. PDU terdiri atas beberapa *header*. *Header* yang dikirim SMS ke SMS-Center berbeda dengan header SMS yang diterima dari SMS-Center.

2.4.1. Kode PDU

PDU untuk mengirim SMS ke SMS-Center, kode PDU untuk mengirim SMS terdiri atas tujuh *header*, yaitu :

a) Nomor SMS-Center

Header pertama ini terbagi atas tiga *subheader*, yaitu :

- Jumlah pasangan heksadesimal SMS-Center dalam bilangan heksa.
- Kode nasional dan internasional
- No SMS-Center dalam pasangan yang dibalik. Jika tertinggal satu angka heksa yang tidak memiliki pasangan maka angka tersebut dipasangkan dengan huruf F di depannya.

Misalkan No SMS-Center untuk *Pro X*L adalah 0818445009 atau 62818445009 bisa diubah menjadi kode PDU 06818018445009 atau 07912618485400F9. Langkah-langakahnya adalah sebagai berikut :

- Cara I: SMS-Center : 0818445009
  06 →ada 6 pasang
  81 →1 pasang
  80-18-44-05-90
  Digabung menjadi kode PDU : 06818018440509

- Cara II: SMS-Center : 62818445009
  07 → ada 7 pasang
  91 →1 pasang
  26-18-48-54-00-F9
  Digabung menjadi kode PDU : 07912618485400F9.

b) Tipe SMS

Untuk mengirim SMS (SEND) maka tipe SMS-nya adalah 1. Jadi bilangan heksanya adalah 01.

c) Nomor Referensi SMS

Nomor referensi ini diberikan nilai *default* 0 (heksadesimal = 00).

d) Nomor Ponsel Penerima

Aturan penulisan *header* PDU untuk nomor ponsel penerima sama halnya dengan aturan penulisan *header* PDU SMS-Center. Header ini juga terbagi atas tiga bagian yaitu jmlah bilangan desimal nomor ponsel yang dituju (heksa), Kode Nasional / Internasional, dan Nomor ponsel yang dituju. Misalkan bahwa nomor ponsel yang dituju adalah 081338720083 maka kode PDU-nya dapat ditulis dengan 2 cara yaitu :

- Cara I No Ponsel yang dituju : 081 74778283
  0B → ada 11 angka
  81
  80-71-74-87-82-F3
  Digabung menjadi kode PDU : 0B818071748782F3

- Cara II No Ponsel yang dituju : 6281 74778283
  0C → ada 12 angka
  91
  26-18-47-77-28-38
  Digabung menjadi kode PDU : 0C91261847772838

e) Bentuk SMS

Bentuk-bentuk SMS biasanya dibedakan menjadi tiga tipe yaitu :

0 → 00 → dikirim sebagai SMS

1 → 01 → dikirim sebagai *telex*

2 → 02 → dikirim sebagai *faximile*

Jadi untuk mengirimkan data dalam bentuk SMS harus digunakan kode PDU 00.

f) Skema Encoding Data I/O

Skema encoding SMS yang ada sekarang ini menggunakan 2 bentuk skema encoding yaitu :

- Skema 7 bit → ditandai dengan angka 0 → 00 (bilangan heksadesimal)
- Skema 8 bit → ditandai dengan angka yang lebih besar dari 0 kemudian diubah menjadi angka heksadesimal yang sesuai. Kebanyakan ponsel / SMS *Gateway* yang ada menggunakan skema 7 bit sehingga harus digunakan kode 00.

g) Isi SMS

Header ini terdiri dari 2 *subheader*, yaitu panjang (jumlah huruf) dan isi yang berupa pasangan bilangan Heksadesimal. Jika menggunakan ponsel/SMS *gateway* berskema *encoding* 7 bit maka ketika mengetikkan suatu huruf dari keypadnya berarti telah dibuat 7 angka 1/0 secara berurutan. Skema 7 bit tersebut diperlihatkan oleh tabel 1. Ada dua langkah yang harus dilakukan untuk mengubah isi SMS ke kode PDU, yaitu :

- Langkah Pertama adalah mengubahnya menjadi kode 7 bit.
- Langkah Kedua adalah mengubah kode 7 bit menjadi 8 bit yang diwakili oleh pasangan heksadesimal.

Pesan “Ada Kejadian!!!” dikodekan menjadi 7 bit *default alphabet (septet)* sehingga harus di-*encode* menjadi 8 bit (*octet*) untuk mendapatkan deretan kode PDU-nya. Tabel 2.3 menunjukkan cara pengkodean 7 bit menjadi 8 bit. Dengan

demikian hasil konversi kata “Ada Kejadian!!!” menjadi bilangan heksadesimal (kode PDU) adalah :

”0F417218B42CABC3E474D81D0A8500”

Setelah mendapatkan masing-masing header maupun *subheader* untuk mengirimkan SMS maka langkah selanjutnya adalah menggabungkan menjadi sebuah PDU yang lengkap. Misalkan untuk mengirimkan kata “Ada Kejadian!!!” ke ponsel dengan nomor 08174778283 melalui SMS-Center Telkomsel tanpa membatasi jangka waktu validitas SMS maka PDU lengkapnya adalah :

”07912618485400F901000B818071748782F300000F417218B42CABC3E474D81D0A85000 “.

Tabel 2.2. Skema *Encoding* 7 Bit [8]

				b 7	0	0	0	0	1	1	1	1
				b 6	0	0	1	1	0	0	1	1
				b 5	0	1	0	1	0	1	0	1
b 4	b 3	b 2	b 1		0	1	2	3	4	5	6	7
0	0	0	0	0	@	Δ	SP	0	-	P	¨	p
0	0	0	1	1			!	1	A	Q	A	q
0	0	1	0	2	$	Φ	“	2	B	R	B	r
0	0	1	1	3		Γ	#	3	C	S	C	s
0	1	0	0	4		Λ		4	D	T	D	t
0	1	0	1	5		Ω	%	5	E	U	E	u
0	1	1	0	6		Π	&	6	F	V	F	v
0	1	1	1	7		Ψ	‘	7	G	W	G	w
1	0	0	0	8		Σ	(	8	H	X	H	x
1	0	0	1	9		Θ	)	9	I	Y	I	y
1	0	1	0	10	LF	Ξ	*	:	J	Z	J	z
1	0	1	1	11			+	;	K	Ä	K	ä
1	1	0	0	12			,	<	L	Ö	L	ö
1	1	0	1	13	CR		-	=	M		M	
1	1	1	0	14		ß	.	>	N	Ü	N	ü
1	1	1	1	15			/	?	O		O	

Tabel 2.3. Pengkodean 7 Bit *(Septet)* Menjadi 8 Bit *(Octet)* [8]

Karakter	Desimal (ASCII)	Septet (7Bit)	Octet (8 Bit)	Hexa
A	65	1000001	11110000	0F
D	100	1100100	10000010	41
A	97	1100001	11100100	72
Spasi	32	0100000	11000000	18
K	107	1001011	10110100	B4
E	101	1100101	10110000	2C
J	106	1101010	10101011	AB
a	97	1100001	11000011	C3
d	100	1100100	11100100	E4
i	105	0101001	11101000	74
a	97	1100001	11011000	D8
n	110	1101110	11110100	1D
!	33	0100001	10100000	0A
!	33	0100001	10000101	85
!	33	0100001	00000000	00

2.4.2. PDU untuk terima SMS dari SMS-Center

Header-header yang digunakan untuk menerima SMS dari SMS-Center hampir sama dengan *header* yang dipakai untuk mengirim SMS ke SMS-Center. Delapan *header* yang dipakai untuk terima SMS adalah :

a) No SMS-Center
b) Tipe SMS. Untuk menerima SMS digunakan tipe SMS = 4 (heksadesimal = 04)
c) No ponsel pengirim
d) Bentuk SMS
e) Skema *encoding*
f) Tanggal dan Waktu SMS di stamp di SMS-Center. Tanggal dan waktu SMS di stamp di SMS-Center diwakili oleh 12 bilangan heksa (6-pasang) dengan format penulisan, ‘yy/mm/dd hh:mm:ss’ yang dibolak-balik dalam pasangannya. Misalkan terdapat cuplikan kode PDU yang menunjukkan tanggal dan waktu SMS di stamp di SMS-Center sebagai berikut:

301192018454 → 03/11/29 10:48:45 → 29 November 2003 10:48:45 WIB

g) Batas waktu validitas (“00”)

h) Isi SMS

Misalkan kode PDU yang diterima dan akan diterjemahkan adalah sebagai berikut :

“07912618485400F9040C912618477728380000406050113110000 2B120 “

mempunyai arti :

- SMS tersebut dikirim melalui SMS-Center +62818445009
- SMS tersebut merupakan SMS terima
- SMS tersebut dikirim dengan nomer ponsel 08174778283
- SMS tersebut diterima dalam bentuk SMS
- SMS tersebut memiliki skema *encoding* 7 bit
- SMS tersebut tidak memiliki batas waktu valid
- SMS tersebut isinya adalah “1A”

Pesan “1A” yang dikodekan menjadi 7 bit *default alphabet* (*septet*) harus terlebih dahulu di-*encode* menjadi 8 bit (*octet*). Pada contoh di atas, isi pesan dalam deretan bilangan heksadesimal adalah “B120”. Pada Tabel 2.4 menunjukkan cara pendekodean isi SMS dari deretan bilangan heksadesimal menjadi deretan karakter-karakter [8].

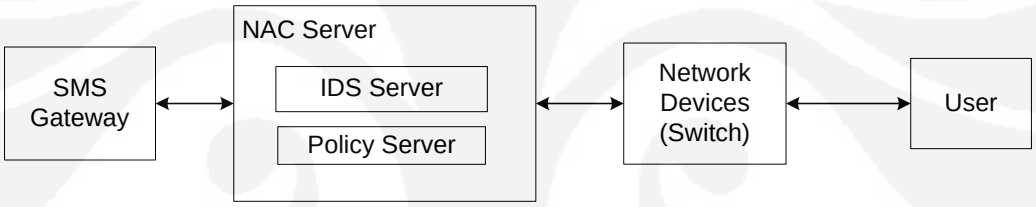
Tabel 2.4. Pendekodean 8 bit (octet) menjadi 7 bit [8]

Hexa	Desimal	Octet(8 bit)	Septet (7 bit)	Karakter
B1	49	**10110001**	0110001	1
20	65	**00100000**	**1000001**	A

BAB 3
PERANCANGAN *POLICY SERVER*

3.1. PERANCANGAN SISTEM

Pada skripsi ini, konsep jaringan NAC dibagi menjadi 2 bagian yaitu *policy server* dan IDS *server*. *Policy server* pada NAC digunakan untuk mengatur *user* yang akan mengakses jaringan kita berdasarkan identitas yang dimiliki oleh *user*, namun pada pada skripsi ini ada penambahan fungsi yang dilakukan oleh *policy server* yaitu melakukan *monitoring* terhadap *network device* dan melakukan proses *reporting* kepada *administrator* jaringan bila jaringan mengalami suatu gangguan melalui SMS. Secara umum *design* sistem pada NAC *server* yang akan dibuat adalah sebagai berikut:



Gambar 3.1. Design Jaringan NAC

Pada jaringan ini *network device* yang digunakan adalah *switch* CISCO 2950. *Server* jaringan menggunakan linux *server*, media penyimpanan data pada *server* menggunakan *database* MySQL dan SMS *gateway* menggunakan modem itegno 3000. Prinsip kerja jaringan diatas adalah sebagai berikut:

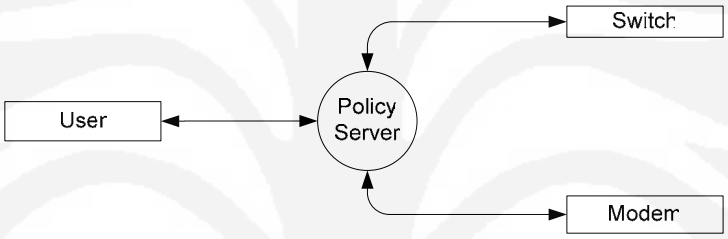
a) *User* baru melakukan akses ke jaringan, secara default *user* mendapatkan IP secara otomatis.
b) NAC *server* mengirimkan autentifikasi menggunakan web *browser*.
c) *User* mengirimkan *username* dan *password*.
d) NAC *Server* melakukan cek ke *database* dan mengirimkan respon ke *client.*
e) NAC *Server* melakukan setting VLAN pada *switch* sesuai dengan *username*.

f) *User* dapat masuk ke jaringan dan memperoleh *previledge* sesuai dengan *username* yang digunakan.

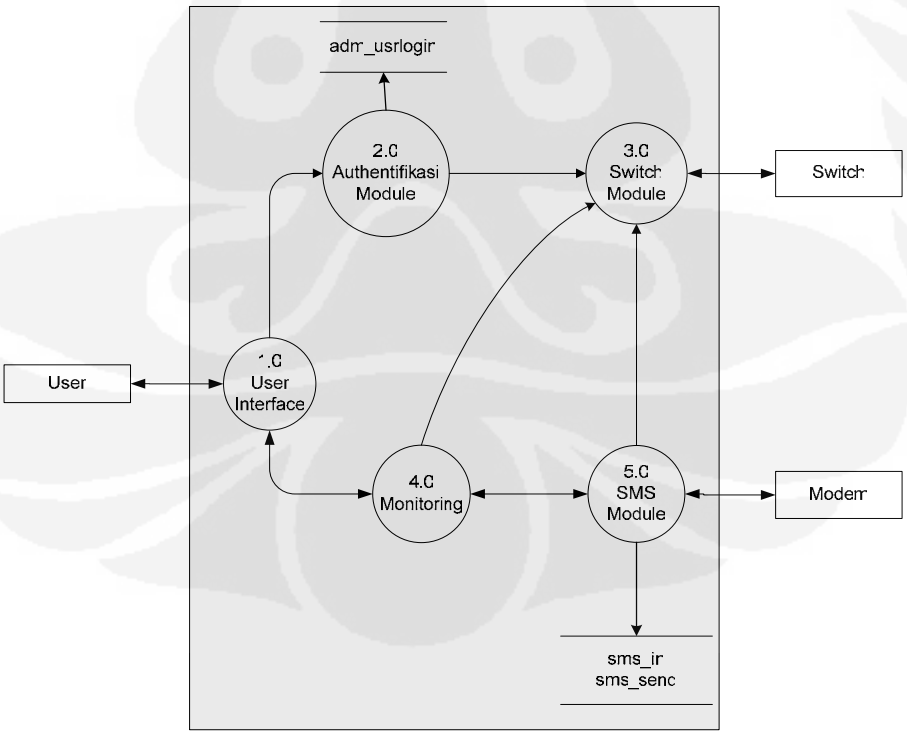
g) Bila terdeteksi suatu serangan terhadap *server*, NAC akan menutup port VLAN yang terdeteksi melakukan serangan dan mengirimkan pesan kepada *administrator* jaringan menggunakan sms.

h) *Administrator* jaringan dapat mengubah setting *switch* dengan cara mengirimkan *command* tertentu melalui SMS ke nomor yang digunakan oleh sms *gateway*.

Design sistem dari *policy server* yang digunakan pada skripsi ini adalah sebagai berikut:



Gambar 3.2. Data Flow Diagram Level 0



Gambar 3.3. Data Flow Diagram Level 1

Proses spesisfikasi:

No. Proses	1.0
Nama Proses	*User* Interface
Deskripsi	Digunakan untuk mendapatkan interface *user* dengan sistem

No. Proses	2.0
Nama Proses	Authentifikasi Modul
Deskripsi	Modul yang digunakan untuk melakukan proses authentifikasi *user*

No. Proses	3.0
Nama Proses	*Switch* Modul
Deskripsi	Modul yang digunakan untuk komunikasi dengan *switch*

No. Proses	4.0
Nama Proses	*Monitoring*
Deskripsi	Digunakan untuk mendapatkan informasi tentang kondisi *network device*

No. Proses	5.0
Nama Proses	SMS Modul
Deskripsi	Modul yan digunakan untuk mengirim report dan menerima instruksi *command* dari *administrator* jaringan melalui SMS

Sementara itu list tabel yang digunakan dalam sistem ini disimpan pada *database* 'tugasakhir' adalah sebagai berikut:

Tabel 3.1. List Nama Tabel di *Database* Tugasakhir

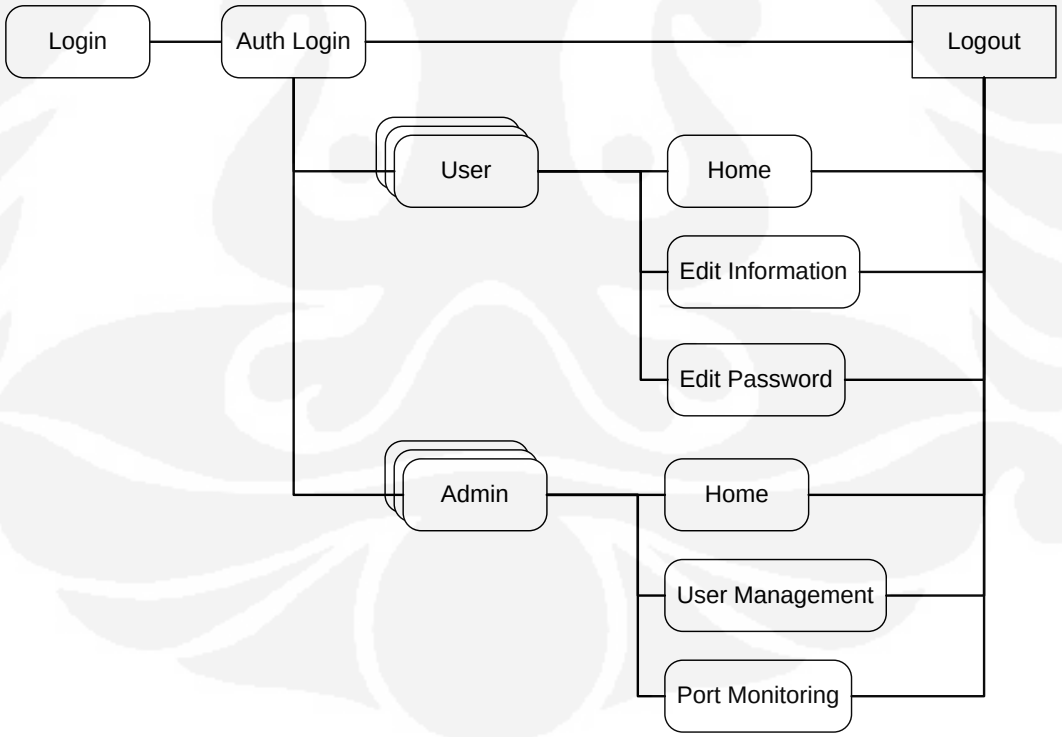
Nama Tabel	Jumlah Kolom	Keterangan
adm_menu	4	Digunakan untuk menyimpan link menu *User* Interface
adm_swport	4	Digunakan untuk menyimpan data port *switch*
adm_*user*info	6	Berisi informasi tentang *user*
adm_usr*login*	4	Berisi data *login user*
ctrl_panel	3	Berisi data *setting* yang digunakan dalam sistem
sms_in	5	Berisi data sms yang masuk ke sms *gateway*
sms_send	5	Berisi data sms yang dikirimkan oleh sms *gateway*

3.2. DESIGN MODUL *POLICY SERVER*

Modul *policy server* dibagi menjadi beberapa jenis yaitu modul *user interface*, modul authentifikasi, modul *monitoring* dan modul sms seperti yang terlihat pada Gambar 3.3. Berikut ini adalah penjelasan untuk masing – masing modul.

3.2.1. Modul *User Interface*

Modul ini dibuat dengan tujuan untuk mempermudah interaksi antara *user* dengan sistem. Modul ini juga digunakan untuk mendapatkan informasi *username* dan *password user* yang akan digunakan oleh authentifikasi modul. Selain itu modul ini juga berfungsi untuk menampilkan *report* dari proses *monitoring network device* dari jaringan. Setiap *user* mempunyai menu yang berbeda sesuai dengan tipe *user*. Secara global konsep dari modul *user interface* dapat dilihat pada Gambar 3.4. Sementara *Design* tampilan dari awal modul *user interface* yang digunakan dapat dilihat pada Gambar 3.5.



Gambar 3.4. Konsep Menu Modul *User Interface*

Gambar 3.5. Tampilan Awal *User Interface*

3.2.2. Authentifikasi Modul

Modul ini bertugas untuk melakukan proses authentifikasi *user* yang akan mengakses jaringan. Bila *login* berhasil dilakukan maka modul ini akan mengirimkan perintah *switch* modul untuk melakukan setting VLAN *switch* sesuai dengan identitas *user*. *User* pada jaringan dibagi menjadi tiga jenis yaitu *administrator*, karyawan dan staff. Dimana masing – masing tipe *user* ini memiliki *previledge* yang berbeda seperti yang terdapat pada Tabel 3.2. Proses merubah konfigurasi vlan dilakukan dengan cara mengirimkan *command* ke *switch* modul melalui port TCP/IP.

Tabel 3.2. Tabel List *Previledge User*

Jenis *User* / Jenis *Policy*	Administrator	Staff	Karyawan
Informasi tentang *user*	Ya	Ya	Ya
Akses jaringan lokal	Ya	Ya	Ya
Akses Internet	Ya	Ya	Tidak
Notifikasi SMS	Ya	Tidak	Tidak
User Management	Ya	Tidak	Tidak

3.2.3. *Switch* Modul

Tugas utama dari modul ini adalah berkomunikasi dengan *switch* sesuai dengan instruksi yang diberikan baik oleh authentifikasi modul, *monitoring* modul

atau SMS modul. Modul ini dibuat menggunakan bahasa pemrograman C seperti yang terlihat pada Lampiran 3. Modul ini menggunakan port TCP/IP untuk komunikasi dengan modul yang lain. Flowchart yang digunakan pada modul ini dapat dilihat pada Gambar 3.6.

Format data yang harus dikirimkan ke modul ini dapat dilihat pada Tabel 3.3. Selain berhubungan dengan modul yang ada di-*policy server*, modul ini juga berhubungan dengan IDS *server*. Untuk komunikasi dengan IDS *server*, format data yang digunakan berbeda dengan format pada Tabel 3.3. Format data yang digunakan adalah:

IDSS, priority X|source y.y.y.y

Keterangan:

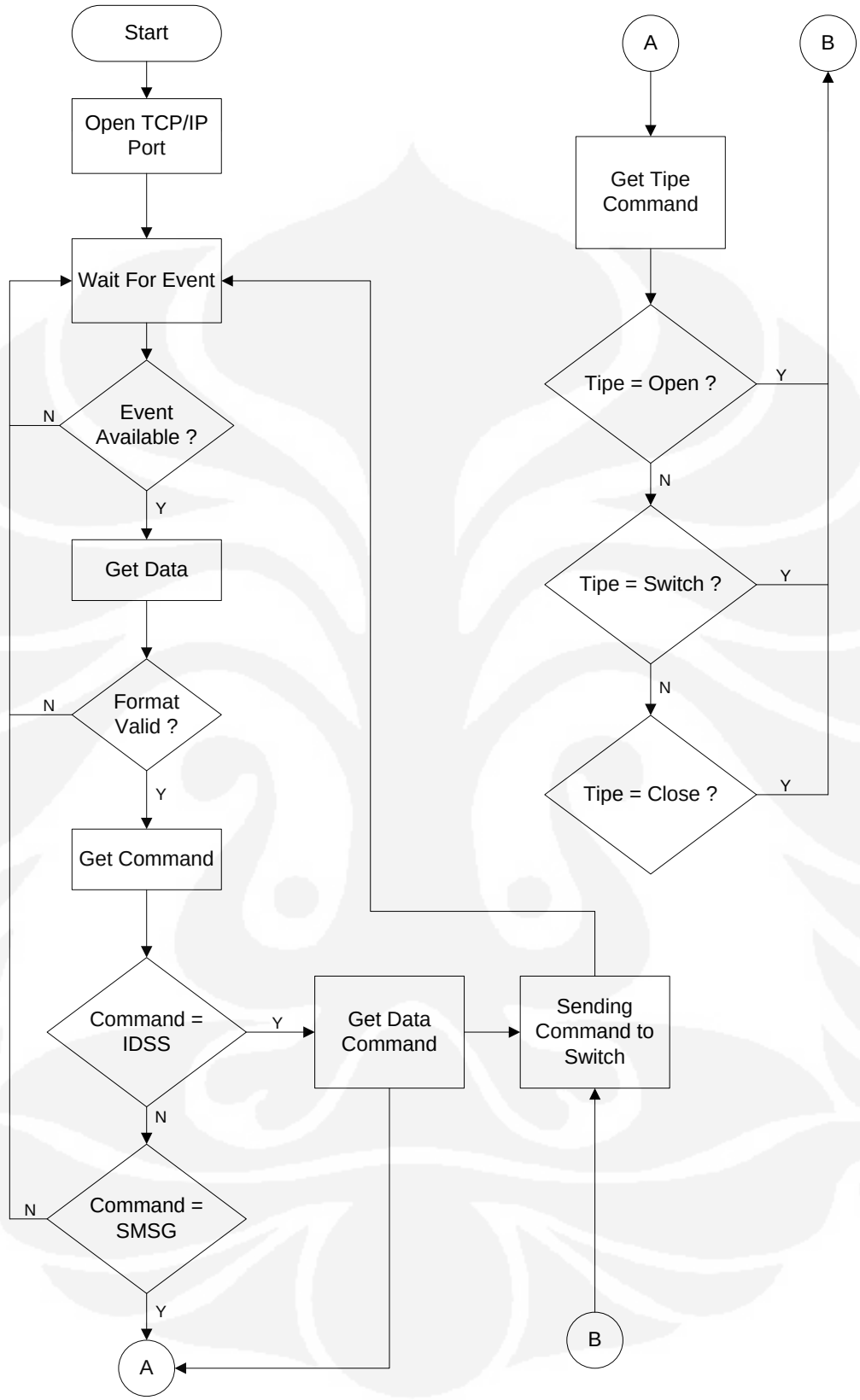
- X adalah nilai priority dari gangguan pada jaringan
- y.y.y.y adalah alamat IP yang melakukan gangguan pada jaringan.

Perbedaan format data ini bertujuan untuk mengidentifikasi sumber dari data, bila data berasal dari *policy server*, modul akan melakukan eksekusi program sesuai dengan *command* yang diperoleh, tetapi bila data berasal IDS *server*, maka modul akan mengirimkan data ke SMS modul melalui port TCP/IP dan akan diteruskan ke *administrator* jaringan melalui SMS.

Tabel 3.3. Tabel List Format Data *Switch* Modul

Command	Keterangan
SMSG,open-P-V	*Command* untuk melakukan open port P pada *switch* dengan vlan id 'V'
SMSG,close-P	*Command* untuk menutup akses port P pada *switch*
SMSG,*switch*-P-V	*Command* untuk mengubah vlan id pada port P ke vlan id 'V'

Sementara untuk melakukan komunikasi denga *switch*, program ini menggunakan perl sebagai penghubungnya. Program perl inilah yang akan diakses oleh *switch* modul sesuai dengan *command* yang ada di Tabel 3.3.



Gambar 3.6. Flowchart Program Switch Modul

3.2.4. *Monitoring* Modul

Modul ini digunakan untuk melakukan *monitoring* port *switch* yang digunakan, *monitoring* yang dimaksud adalah melihat kondisi port (aktif/tidak) dan *user* mana yang melakukan akses. Untuk dapat melihat kondisi port, maka program ini mengakses serial port menggunakan *perl, script* yang dapat dilihat pada Gambar 3.7 untuk *script* perl lainnya dapat dilihat pada Lampiran 2.

```
#!/usr/bin/perl
use Net::Telnet::Cisco ;
open(MYOUTFILE, "> /var/www/tugas/log/swmac.txt");

my $session = Net::Telnet::Cisco->new(Host => '192.168.1.2');
$session->login('', 'cisco');
# Enable mode
if ($session->enable("cisco") ) {
      @output = $session->cmd('show mac-address-table');
      print "My privileges: @output\n";
      print MYOUTFILE @output;
} else {
     warn "Can't enable: " . $session->errmsg;
}
$session->close;
close(MYOUTFILE);
```

Gambar 3.7. Listing Program Untuk Mendapatkan Informasi Port *Switch*

Hasil dari eksekusi program diatas adalah file dengan nama “swmac.txt”, file ini yang dibaca oleh modul *monitoring* dan ditampilan pada halaman web. Sementara algoritma yang digunakan pada proses ini adalah sebagai berikut:

1. Get Data From Switch
2. Insert Port and Mac address to table
3. Get ARP
4. Get Mac address and IP address
5. Insert IP address if Mac address similar with mac adrress in table

Gambar 3.8. Algoritma Port *Monitoring*

3.2.5. SMS Modul

Modul ini bertujuan untuk melakukan *reporting* terhadap *administrator* jaringan bila terjadi gangguan pada jaringan. Selain melakukan *reporting*, modul ini juga akan melakukan eksekusi *command* untuk melakukan sesuai dengan

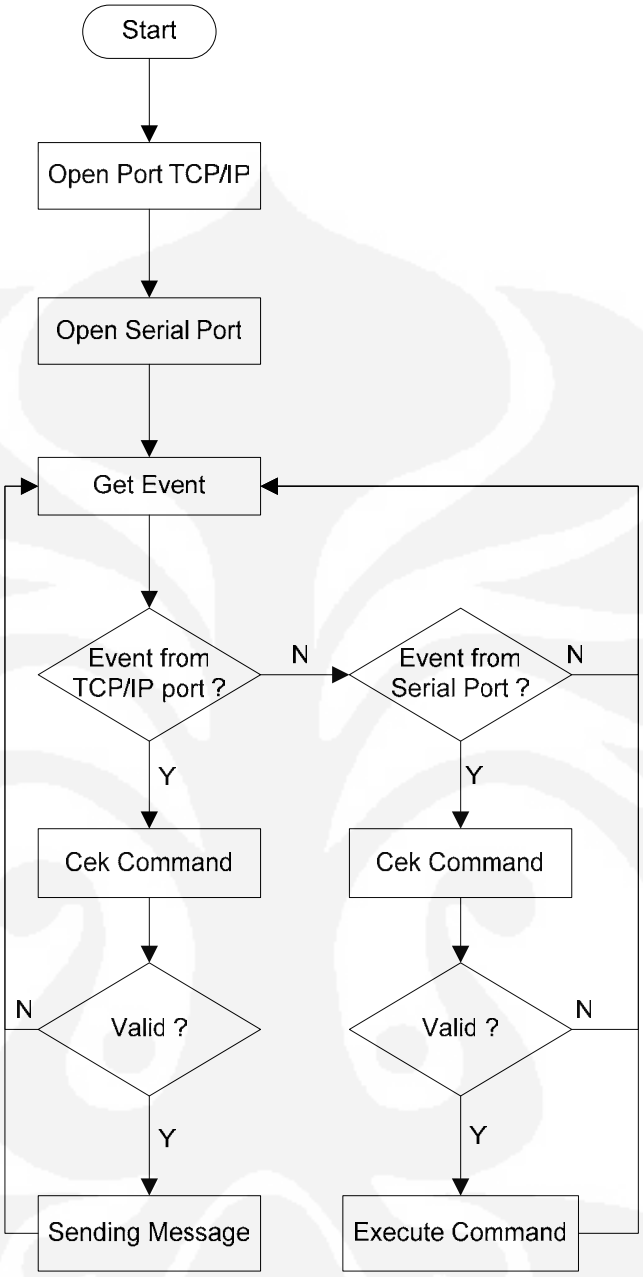
command yangn diberikan oleh *administrator*. Format data yang harus dikirimkan melalui port TCP/IP untuk mengirimkan SMS adalah sebagai berikut:

Admin, <Isi Pesan>

Sementara untuk format isi SMS untuk merubah konfigurasi *switch* dapat dilihat pada Tabel 3.4. Flowchart program yang digunakan pada modul ini ada pada Gambar 3.9.

Tabel 3.4. Tabel List Command SMS

Jenis *command*	Keterangan
Admin,openport-P-V	*Command* untuk melakukan open port P pada *switch* dengan vlan id 'V'
Admin,closeport-P	*Command* untuk menutup akses port P pada *switch*
Admin,*switch*port-P-V	*Command* untuk mengubah vlan id pada port P ke vlan id 'V'



Gambar 3.9. Flowchart Program SMS Modul

Pada proses *"Sending Message"* prosedur yang digunakan adalah sebagai berikut:

```
1. Get Data
2. Create PDU from Data
3. Sending AT+Command to Modem
4. Sending PDU Data
5. Wait Response from Modem
```

Gambar 3.10. Algoritma Prosedur Sending Message

Sementara pada prosedur *"execute command"*, algoritma yang digunakan adalah sebagai berikut:

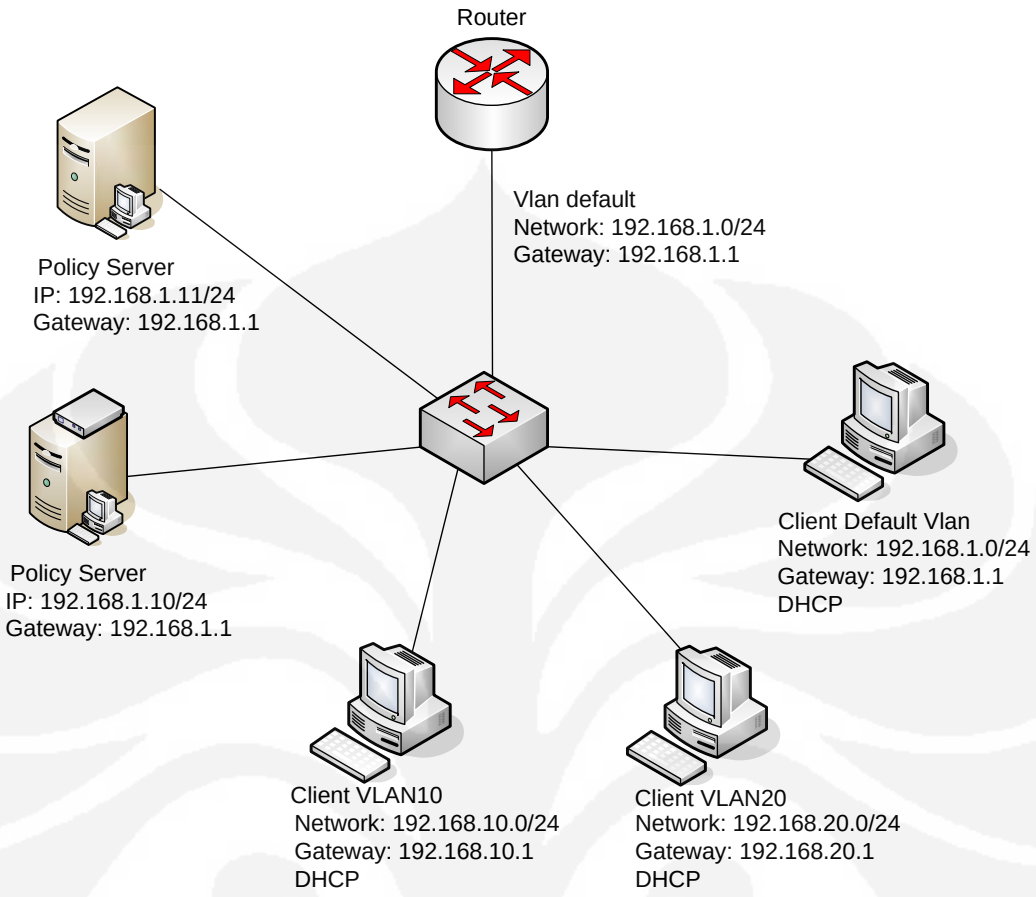
```
1. Get Data From Command
2. Cek Tipe Command
3. If command Valid then
   Sending data to switch module
```

Gambar 3.11. Algoritma Prosedur *Execute Command*

3.3. DESIGN JARINGAN

Pada skripsi ini, jaringan yang akan digunakan adalah jaringan sederhana yang terhubung menggunakan sebuah *switch* CISCO 2950-24. Untuk mempermudah pengelopokan *user* jaringan, *user* jaringan dikelompokan dalam sebuah vlan sesuai dengan tipe dari *user*. Ada 3 macam vlan yang digunakan yaitu vlan *default* untuk *user* yang baru, vlan10 yang digunakan untuk *user* dengan tipe karyawan dan vlan20 untuk *user* dengan tipe staff. Untuk menghubungkan *policy server* dengan masing – masing vlan digunakan sebuah *router*. *Router* disini bertugas untuk menghubungkan vlan dengan *server* dan memfilter *packet* akan menuju ke jaringan vlan menggunakan *access list*. Selain melakukan pemfilteran *packet*, *router* juga digunakan sebagai DHCP *server* jaringan vlan. Design jaringan yang digunakan dapat dilihat pada Gambar 3.12. Spesifikasi teknis perangkat keras yang digunakan pada skripsi adalah sebagai berikut:

a) *Policy Server* dengan *platform* Linux
   Spesifikasi: Notebook dengan Inter Core Duo Processor 1,66 GHz, Memory 1 G RAM, Ethernet 100Mbps.
b) IDS *Server* dengan *platform* Linux
   Spesifikasi: Notebook dengan Inter Core Duo Processor 1,66 GHz, Memory 1 G RAM, Ethernet 100Mbps.
c) *Switch* CISCO 2950-24.
d) *Router* CISCO 2621.
e) Modem Itegno 3000



Gambar 3.12. Design Jaringan NAC *Server*

Konfigurasi *switch* pada jaringan diatas adalah sebagai berikut:

```
hostname Switch
enable password cisco
vlan 10
  name karyawan
vlan 20
  name staff
interface FastEthernet0/1
  switchport mode trunk
  no ip address
interface Vlan1
  ip address 192.168.1.2 255.255.255.0
  no ip route-cache
line con 0
line vty 0 4
  password cisco
  login
line vty 5
  password cisco
  login
line vty 6 15
  login
```

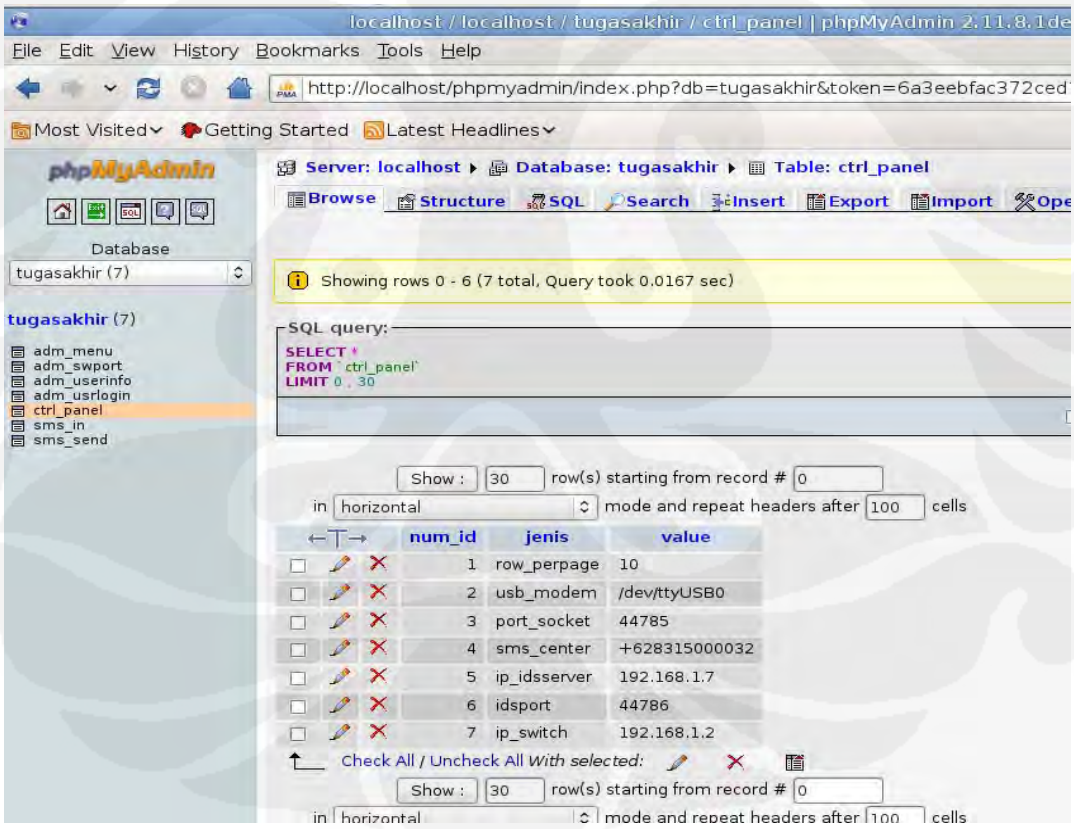
Semetara untuk konfigurasi *router* yang digunakan adalah sebagai berikut:

```
interface FastEthernet0/0
 no ip address
 duplex auto
 speed auto
interface FastEthernet0/0.1
 encapsulation dot1Q 1 native
 ip address 192.168.1.1 255.255.255.0
 ip access-group 101 in
interface FastEthernet0/0.10
 encapsulation dot1Q 10
 ip address 192.168.10.1 255.255.255.0
 ip access-group 110 in
interface FastEthernet0/0.20
 encapsulation dot1Q 20
 ip address 192.168.20.1 255.255.255.0
 ip access-group 120 in
ip dhcp pool vlan10
   network 192.168.10.0 255.255.255.0
   default-router 192.168.10.1
ip dhcp pool vlan20
   network 192.168.20.0 255.255.255.0
   default-router 192.168.20.1
ip dhcp pool vlan1
   network 192.168.1.0 255.255.255.0
   default-router 192.168.1.1
ip dhcp excluded-address 192.168.1.10
ip dhcp excluded-address 192.168.10.1
ip dhcp excluded-address 192.168.20.1
access-list 101 permit ip 192.168.1.10 0.0.0.1 192.168.10.0 0.0.0.255
access-list 101 permit ip 192.168.1.10 0.0.0.1 192.168.20.0 0.0.0.255
access-list 101 deny   ip 192.168.1.0 0.0.0.254 192.168.10.0 0.0.0.255
access-list 101 deny   ip 192.168.1.0 0.0.0.254 192.168.20.0 0.0.0.255
access-list 101 permit ip any any
access-list 110 permit ip 192.168.10.0 0.0.0.255 192.168.1.10 0.0.0.1
access-list 110 deny   ip 192.168.10.0 0.0.0.255 192.168.1.0 0.0.0.254
access-list 110 deny   ip 192.168.10.0 0.0.0.255 192.168.20.0
0.0.0.255
access-list 110 permit ip any any
access-list 120 permit ip 192.168.20.0 0.0.0.255 192.168.1.10 0.0.0.1
access-list 120 deny   ip 192.168.20.0 0.0.0.255 192.168.1.0 0.0.0.254
access-list 120 deny   ip 192.168.20.0 0.0.0.255 192.168.10.0
0.0.0.255
access-list 120 permit ip any any
line vty 0 5
 password cisco
 login
```

BAB 4
PENGUJIAN DAN ANALISA SISTEM

4.1. PENGATURAN KONFIGURASI AWAL SISTEM

Pada bagian ini akan dilakukan pengujian sistem yang sudah dibuat berdasarkan perancangan pada bab sebelumnya. Pengujian sistem dilakukan pada setiap modul dengan beberapa kali pengulangan pengujian. Modul – modul yang diuji adalah *user interface* dan authentifikasi modul, *switch* modul, port *monitoring* dan SMS Modul. Sebelum melakukan pengujian terhadap sistem, ada beberapa parameter yang harus dimasukkan ke dalam tabel 'ctrl_panel' di *database*. Parameter – parameter ini dimasukkan menggunakan phpmyadmin seperti yang terlihat pada Gambar 4.1. Parameter – parameter yang perlu diubah adalah port usb modem, sms center, alamat IP IDS *server* dan alamat IP *switch*.



Gambar 4.1. Tampilan PHPMyAdmin Untuk Edit Parameter

Sementara itu modul yang digunakan ada 2 modul yang harus dijalankan pada saat pengujian berlangsung yaitu *switch* modul dan sms modul di *console* linux. Untuk menjalankan *switch* modul langkah – langkah yang harus dilakukan adalah sebagai berikut:

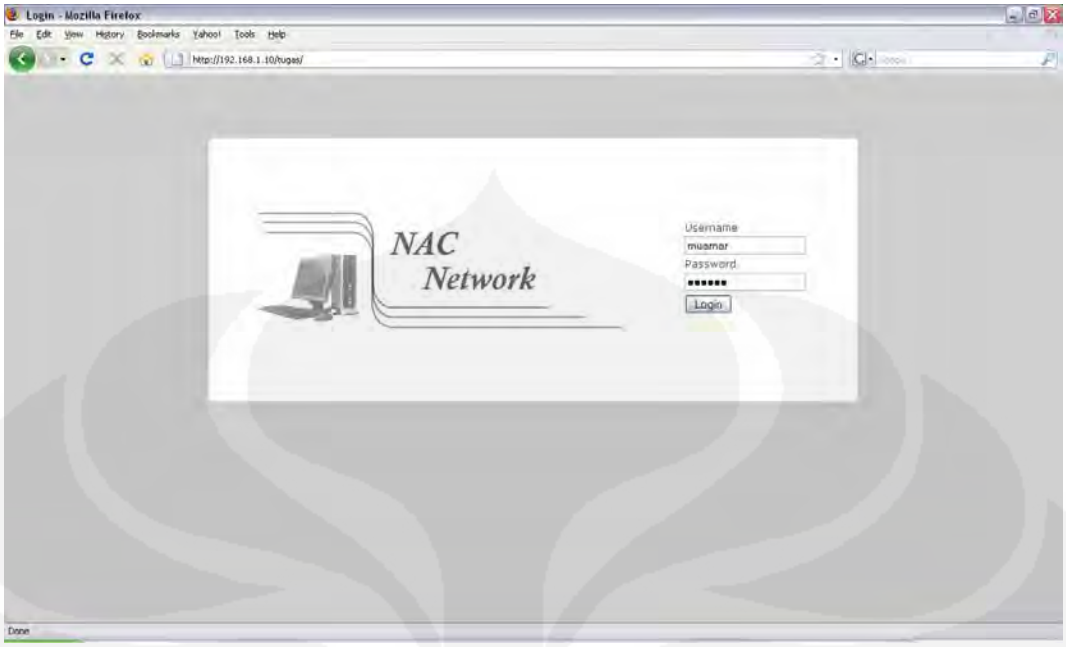
```
# cd /home/swmod
# make clean
# make
# ./swmod
```

Sementara untuk sms modul prosedur yang harus dilakukan adalah sebagai berikut:

```
# cd /home/smsg
# make clean
# make
# ./smsg
```

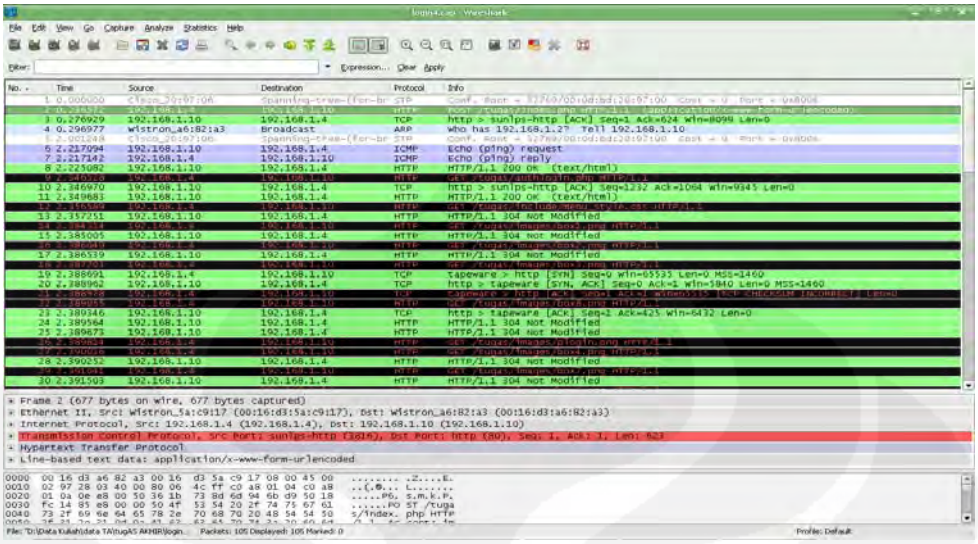
4.2. PENGUJIAN *USER* INTERFACE DAN AUTHENTIFIKASI MODUL

Pengujian modul ini dilakukan untuk melihat keberhasilan *user* dalam melakukan *login* pada jaringan melalui web. *Username* dan *password* disimpan pada tabel ‘adm_usrlogin’. Setiap *username* mewakili satu *client* jaringan. Proses *login* dilakukan dengan cara mengakases web *policy server* (192.168.1.10) seperti yang terlihat pada Gambar 4.2. Bila *login* yang dilakukan berhasil maka tampilan web akan menuju ke link *“authentifikasi login”* dan sistem akan mengirimkan perintah ke *switch* modul untuk melakukan perubahan vlan yang sesuai dengan tipe dari *user* tersebut. Sementara *print screen* hasil pengujian *user interface* modul dapat dilihat pada Lampiran 1.



Gambar 4.2. Tampilan Awal Proses *Login*

Pada tahap ini *user* diminta menunggu proses perubahan vlan yang terjadi pada *switch* jaringan. Keberhasilan sistem merubah vlan sesuai dengan tipe *user* yang *login* dapat dilihat dari perubahan IP pada *user*. Untuk *user* dengan tipe 'karyawan', IP yang diperoleh adalah IP network 192.168.10.0/24 sementara untuk *user* dengan tipe 'staff', IP yang diperoleh adalah IP network 192.168.20.0/24. Untuk melihat *time* respon sistem dalam melakukan proses diatas dapat diukur dengan menggunakan program wireshark seperti pada Gambar 4.3. Respon sistem diukur mulai dari *user* melakukan *login* sampai *user* mendapatkan IP sesuai dengan tipe *user*. Proses pengukuran dilakukan pada saat *user* melakukan *login* dan *logout* dari sistem sementara jumlah percobaan yang dilakukan adalah 10 kali percobaan. Hasil dari proses ini dapat dilihat pada Tabel 4.1.

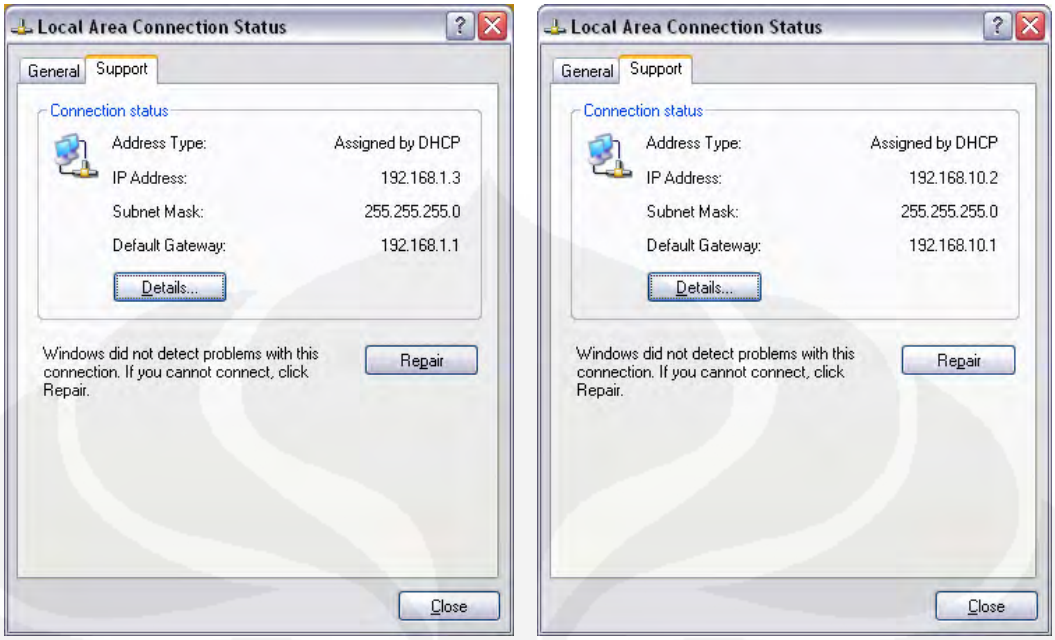


Gambar 4.3. *Print Screen* wireshark

Tabel 4.1. Tabel Respon Sistem Pada Saat *User Login* dan *Logout*

Percobaan	*Login* (s)	*Logout* (s)
1	42.901	41.808
2	40.340	37.859
3	39.665	41.094
4	44.189	40.567
5	41.390	41.650
6	42.049	40.544
7	43.380	40.425
8	43.876	41.270
9	42.648	41.649
10	40.929	38.837

Dari diatas dapat diperoleh nilai rata – rata dari respon sistem pada saat *user* melakukan *login* dan *logout* yaitu 42.137 detik untuk *login* dan 40.570 detik untuk *logout.* Proses ini menjadi lama karena adanya proses *switching* vlan di-*switch* dan pengalokasian IP DHCP dari *router*. *Time* proses ini bisa berubah bila perangkat jaringan yang digunakan berbeda dengan perangkat jaringan yang digunakan pada skripsi ini. Contoh perubahan IP yang terjadi pada *user* sebelum dan setelah melakukan *login* dapat dilihat pada Gambar 4.4.



(a) (b)

Gambar 4.4. (a). Konfigurasi IP Sebelum Melakukan *Login*, (b). Konfigurasi IP Setelah Melakukan *Login*

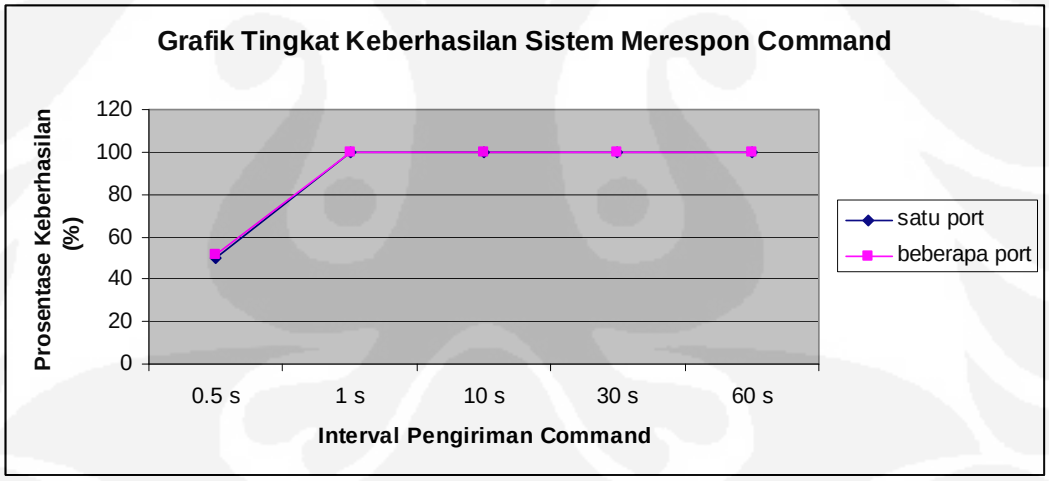
4.3. PENGUJIAN *SWITCH* MODUL

Pada pengujian modul ini akan dilihat tingkat keberhasilan sistem dalam merubah konfigurasi *switch*. Proses pengujian dilakukan cara mengirimkan *command – command* yang dikirimkan authentifikasi modul ke *switch* modul dengan variasi waktu tertentu. Proses ini dilakukan sebagai simulasi sistem pada saat menangani *user* yang banyak. Dengan test ini dapat dilihat kemampuan *switch* modul untuk merespon setiap *command* yang masuk. Keberhasilan modul ini merubah konfigurasi yang ada di *switch* dapat dilihat dari konfigurasi *switch* setelah dilakukan pengiriman *command*. Misalkan kita mengirimkan *command* untuk mengubah port 8 ke vlan 10. *Command* yang digunakan adalah "WEB,192.168.1.2|8|10|0002.3FEC.D93B". Dan konfigurasi *switch* berubah seperti berikut:

```
interface FastEthernet0/7
 no ip address
!
interface FastEthernet0/8
 switchport access vlan 10
 mac-address 0002.3fec.d93b
 no ip address
!
interface FastEthernet0/9
 no ip address
```

Gambar 4.5. Konfigurasi *Switch* Setelah Menerima *Command Open* Port

Pada proses pengujian, pengiriman *command* dilakukan oleh program lain yang bisa diatur waktu pengiriman *command*-nya. Sementara *command* yang dikirimkan ditujukan untuk satu port *switch* dan untuk beberapa port *switch*. Sementara waktu yang digunakan untuk mengirimkan *command* adalah tiap 0.5 detik, 1 detik, 10 detik, 30 detik dan 60 detik. Hasil dari pengujian untuk pengiriman *command* terhadap satu port dapat dilihat pada Gambar 4.6



Gambar 4.6. Grafik Tingkat Keberhasilan Sistem Merespon *Command*

Dari Gambar 4.6 dapat dilihat bahwa *switch* modul dapat merespon *command* yang masuk dengan baik bila *command* yang masuk berjarak 1 detik dari *command* yang sebelumnya. Kegagalan modul menerima menjalankan respon yang diberikan pada waktu yang bersamaan karena modul ini tidak berhasil memilah *command* yang masuk secara berderet. Keadaan ini dapat dilihat pada log modul Gambar 4.7.

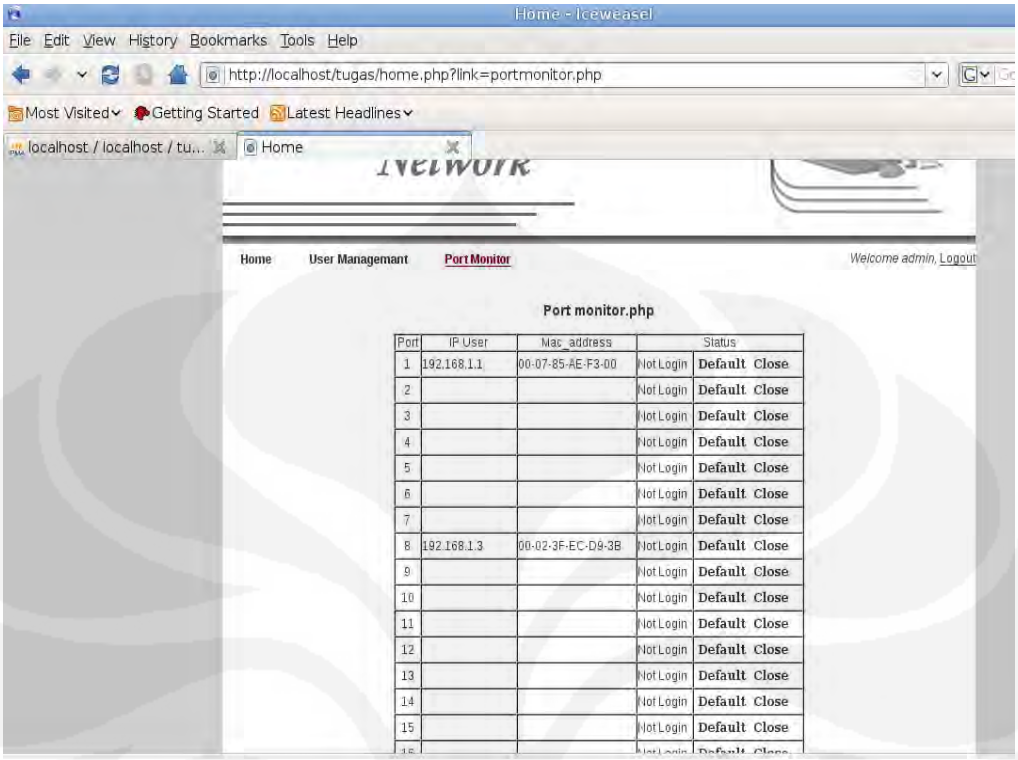
```
06/13/09 14:12:49 - From WEB -> read 18 bytes [WCLO,192.168.1.2|8]
06/13/09 14:12:49 - /usr/bin/perl /var/www/tugas/program/sswitchdef.pl 192.168.1.2 8 1
06/13/09 14:12:50 - From Policy Server -> read 55 bytes
[WEB,192.168.1.2|8|20|0016.D3E7.9274WDEF,192.168.1.2|8|1]
06/13/09 14:12:50 - From WEB -> read 55 bytes
[WEB,192.168.1.2|8|20|0016.D3E7.9274WDEF,192.168.1.2|8|1]
06/13/09 14:12:50 - /usr/bin/perl /var/www/tugas/program/sopenrev1.pl 192.168.1.2 8 20
0016.D3E7.9274WDEF
```

Gambar 4.7. Log Kegagalan *Switch* Modul

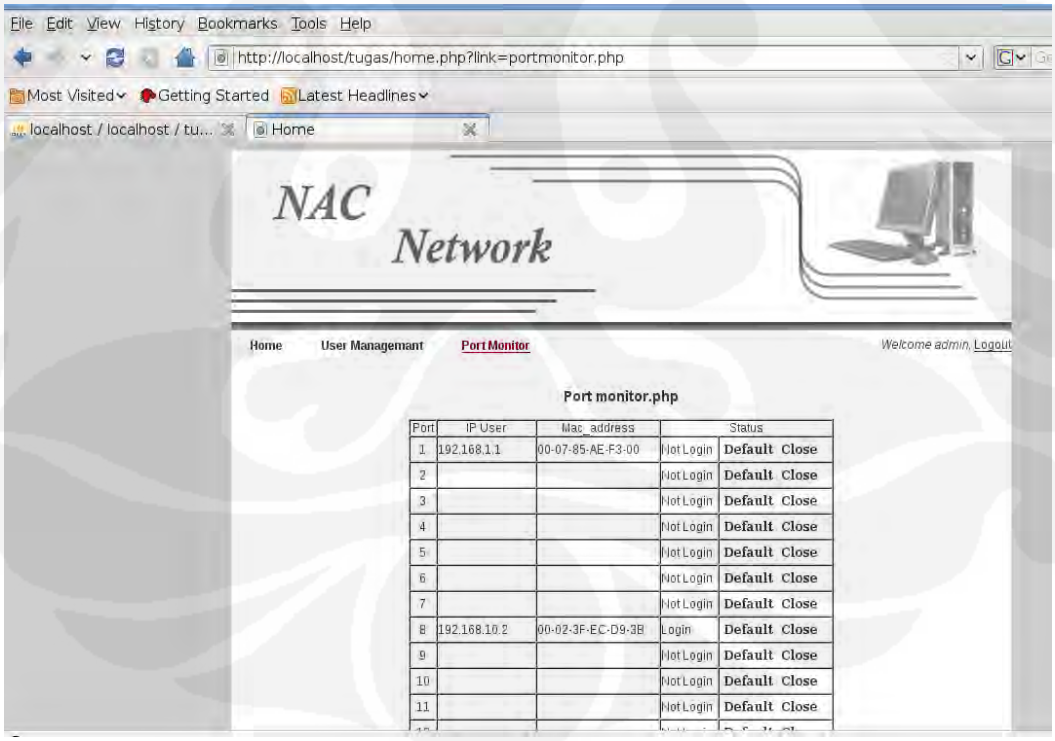
Seperti pada Gambar 4.7 modul tidak berhasil memilah *command* WEB dan WDEF yang masuk secara bersamaan. Modul hanya membaca *command* yang masuk terlebih dahulu dan tidak mengetahui akhir dari *command*, hal ini disebabkan karena tidak adanya protokol penanda akhir dari *command* yang masuk. Kegagalan mengetahui akhir karakter dari *command* berdampak pada kesalahan untuk pemberian parameter untuk menjalankan *script* merubah konfigurasi *switch*. Dengan kondisi seperti ini, modul belum bisa merespon dengan baik bila ada *command* yang masuk secara bersamaan dan berakibat pada ketidak mampuan sistem untuk merespon suatu kejadian yang terjadi secara bersamaan, misalkan pada saat ada beberapa beberapa *user* yang melakukan *login* dan *logout* dalam waktu yang bersamaan.

4.4. PENGUJIAN PORT *MONITORING* MODUL

Proses pengujian pada modul ini adalah memonitor tampilan web pada menu port monitor untuk *user administrator*. Proses pengujian dilakukan dengan cara melihat perubahan data pada tampilan web pada saat ada *user* melakukan koneksi ke *switch* dan pada saat *user* melakukan *login* dan *logout*. Pada Gambar 4.8. terlihat tampilan web pada saat ada *user* baru melakukan askes ke *switch* sementara pada Gambar 4.9 merupakan tampilan pada saat *user* sudah melakukan *login* ke jaringan NAC.



Gambar 4.8. Tampilan Port Monitor Pada Saat Ada *User* Baru Terhubung Dengan *Switch*



Gambar 4.9. Tampilan Port Monitor Pada Saat *User* Sudah Melakukan *Login*

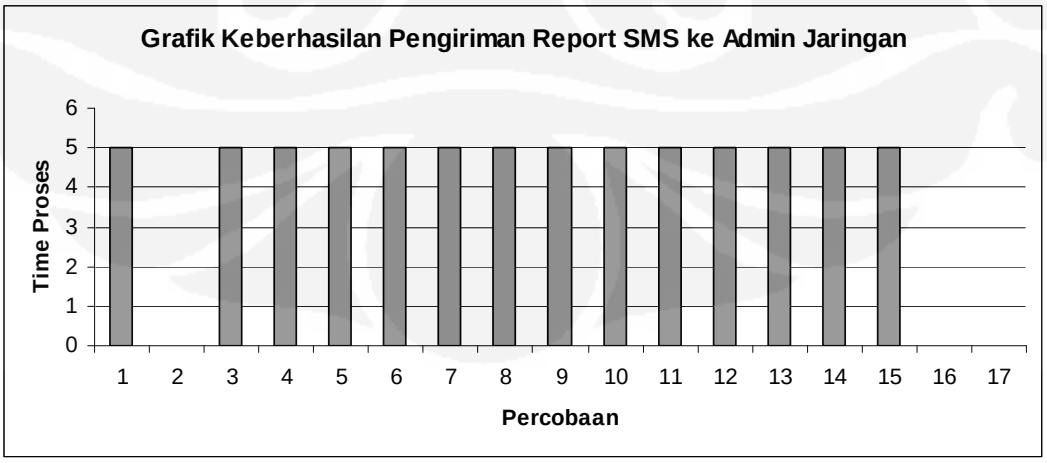
4.5. PENGUJIAN SMS MODUL

Pengujian pada modul dibedakan menjadi 2 macam yaitu mengirim pesan ke *administrator* dan menerima *command* dari *administrator* melalui SMS. Proses pengiriman SMS merupakan bentuk respon sistem dari adanya laporan bahwa terdapat gangguan pada jaringan. Informasi dari IDS *server* diterima oleh *switch* modul, proses pengiriman sms dilakukan dengan cara pengiriman *command* ke port 44785. Semua sms yang dikirimkan oleh modul disimpan dalam tabel 'sms_send' di *database*.

Proses pengujian dilakuan dengan cara melakukan gangguan pada *server*, sehingga *server* melakukan pengiriman data ke *switch* modul. Data ini yang akan diterjemahkan oleh *switch* modul, kemudian mengirimkan pengiriman data ke port sms *gateway* dan melakukan perubahan konfigurasi sesuai dengan tingkat priority gangguan sebagai reaksi terhadap gangguan tersebut. Reaksi terhadap gangguan tergantung pada nilai priority gangguan tersebut, seperti yang terlihat pada Tabel 4.2. Pengiriman sms notifikasi ke *administrator* jaringan tidak meilhat tingkat priority dari gangguan tersebut. Setiap informasi gangguan yang di dapatkan dari IDS *server* dilakaukan pengiriman notifikasi sms.

Tabel 4.2. List Tindakan Pertama Gangguan Pada *Switch*

Priority	Tindakan Pertama
0	Menutup port pada *switch*
1	Menutup port pada *switch*
2	Mengubah setting port *switch* ke default vlan
3	Tidak ada tindakan



Gambar 4.10. Grafik Keberhasilan Pengiriman *Report* SMS ke *Administrator* Jaringan

Dari hasil percobaan pengiriman pesan kepada *administrator* jaringan, 88.24% pesan yang dikirimkan sampai kepada *administrator* jaringan. Nilai ini diperoleh dari hasil pengujian sebanyak 17 kali dengan kegagalan sebanyak 3 kali. Dari tiga kegagalan ini pengiriman *command* ini terdapat 3 pesan *error* dari modem yaitu :

- +CMS ERROR: 38 → *Error* ini disebabkan karena pulsa kartu yang digunakan tidak mencukupi untuk melakukan sms.
- +CMS ERROR: 512 → *Error* karena perangkat masih belum siap mengirimkan sms.

Pengiriman sms dikatakan berhasil bila respon modem yang diperoleh menunjukkan *index* sms yang sudah dikirimkan. Seperti yang terlihat pada log di Gambar 4.11.

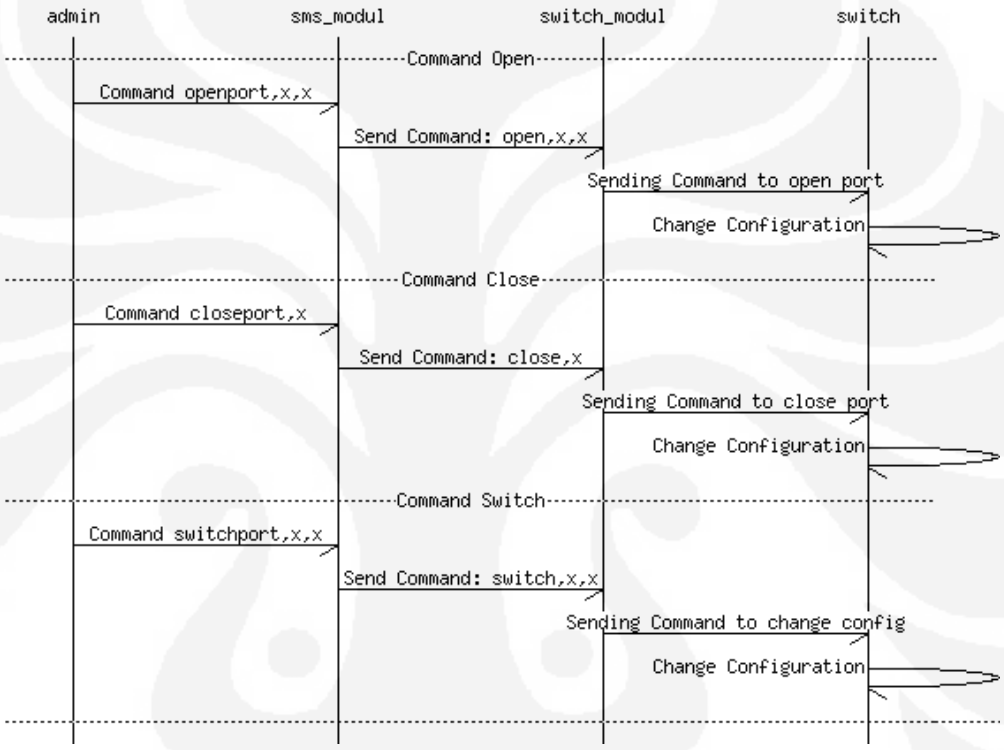
```
06/14/09 18:35:27 - read 56 bytes [Admin, Intrusion detection from 192.168.20.2, priority 2]
06/14/09 18:35:27 -  Intrusion detection from 192.168.20.2, priority 2
06/14/09 18:35:27 - ------------------------------------------------------------
06/14/09 18:35:27 - psan =  Intrusion detection from 192.168.20.2, priority 2, serport = 5
06/14/09 18:35:27 - jumlah admin = 1
06/14/09 18:35:27 - To modem: at+cmgs=59
06/14/09 18:35:27 - <- buf[0]: >
06/14/09 18:35:27 - To modem:
0791265805000000310000D91265896359052F30000AA32A0A49B2EAFCFD36F37885CA
697C7F4F4DB0D32CBDF6D502C2773C56C38970CE692B1407079FA2D4FD3F32019   Z
06/14/09 18:35:32 - <- buf[0]: +CMGS: 25
```

Gambar 4.11. SMS Modul Pada Saat *Sending*

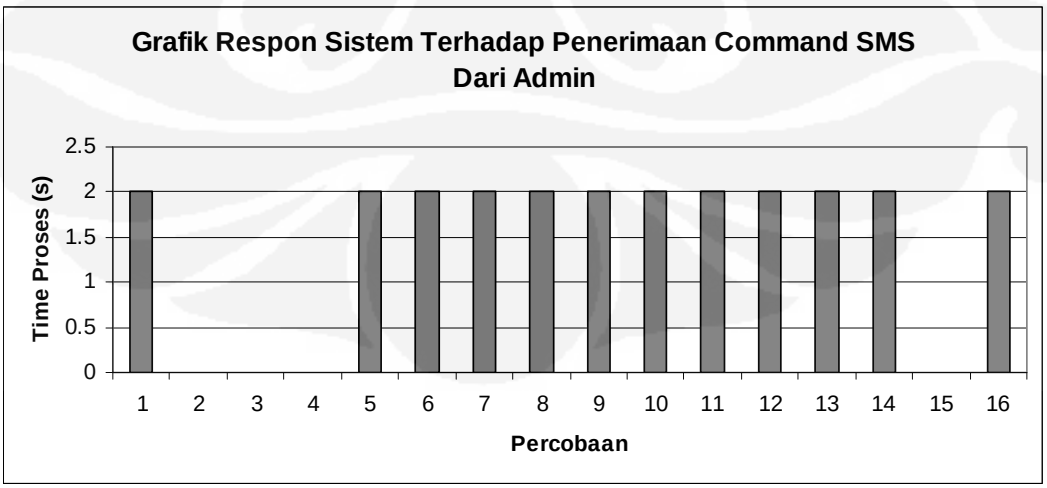
Sementara *time* proses dari pengiriman sms mulai dari menerima *command* dari *switch* modul sampai menerima respon dari modem adalah 5 detik. Bila pada saat pengiriman sms berlangsung dan ada *command* lain masuk melalui port 44785 maka *command* tersebut akan diabaikan sehingga pesan tidak dikirimkan ke *administrator* jaringan.

Selain melakukan pengujian dengan pengiriman sms, pada modul ini juga dilakukan pengujian terhadap penerimaan sms. Pengujian dilakukan dengan cara melakukan pengiriman *command* melalui sms ke nomor sms *gateway*. Proses pengujian dikatakan berhasil bila sms yang diterima berhasil dibaca dan modul mengirimkan *command* ke *switch* modul untuk melakukan perubahan konfigurasi

pada *switch* sesuai dengan *command* yang diberikan. Semua sms masuk yang berhasil dibaca oleh modul ini disimpan pada modul ini disimpan pada tabel 'sms_in' di *database*. Untuk mempermudah melakukan analisa tentang proses apa saja yang terjadi pada modul sms pada saat menerima *command* dari *administrator* melaui sms, berikut ini adalah *"Message Sequence Chart"* dari proses menerima sms dari *administrator* seperti yang terlihat pada Gambar 4.12.



Gambar 4.12. *Message Sequence Chart* Proses Penerimaan *Command* dari *Administrator*



Gambar 4.13 Grafik Keberhasilan Pengiriman SMS ke *Administrator* Jaringan

Pada percobaan penerimaan sms tingkat keberhasilan penerimaan sms adalah 75%, angka ini diperoleh dari 16 kali percobaan penerimaan sms oleh modul ada 12 kali yang berhasil dibaca dan berhasil mengirimkan *command* ke *switch* seperti yang terlihat pada Gambar 4.13. Kegagalan penerimaan *command* di sebabkan oleh kegagalan modul untuk mendecode data PDU dan mendapatkan index dari sms yang masuk. keadaan ini dapat dilihat pada log aktifify pada Gambar 4.14.

```
06/14/09 21:41:34 - To modem: at+cmgr=21
06/14/09 21:41:34 - <- buf[0]: "SM",4
06/14/09 21:41:34 - <- buf[0]: +CMS ERROR: 321
06/14/09 21:41:56 - <- buf[0]: +CMTI: "SM",5
06/14/09 21:41:56 - To modem: at+cmgr=5
06/14/09 21:41:56 - <- buf[0]: +CMG
06/14/09 21:41:56 - <- buf[0]: +CMGR: 0,,37
06/14/09 21:41:56 - <- buf[1]:
07912658050000F0240D91265896359052F3000090604112146582134 17
06/14/09 21:41:56 - <- buf[0]: 23BED66BDE16537FC2DA7B76C2D190C
06/14/09 18:35:32 - <- buf[1]: OK
```

Gambar 4.14. Log Kegagalan SMS Modul Pada Saat Menerima sms

Error diatas disebabkan karena kesalahan membaca index dari sms yang masuk. Sehingga modem memberikan respon *error* '+CMS ERROR: 321', sementara *error* yang lain adalah kegagalan sistem dalam menterjemahkan kode PDU yang didapat pada *server*.

Respon *time* modul terhadap *notifikasi* sms yang masuk adalah 2 detik, hal ini dihitung mulai dari notifikasi oesan dikirimkan dan pesan tersebut dikirimkan ke *switch* modul. Untuk melihat keberhasilan modul sistem pada saat pengujian penerimaan sms dapat dilihat pada konfigurasi *switch* setelah *command* dikirimkan. Misalkan *command* yang dikirimkan adalah "Admin,openport-6-10", maka konfigurasi *switch* yang diperoleh terlihat pada Gambar 4.15.

```
interface FastEthernet0/5
 switchport mode access
 no ip address
!
interface FastEthernet0/6
 switchport access vlan 10
 switchport mode access
 no ip address
!
interface FastEthernet0/7
 switchport mode access
 no ip address
```

Gambar 4.15. Konfigurasi *Switch* Setelah Proses Penerimaan *Command* Dari *Administrator*

BAB 5
KESIMPULAN

1. *Switch* Modul bekerja dengan maksimal bila input *command* yang diberikan berjarak 1 detik dari *command* sebelumnya.
2. Kegagalan *switch* modul menjalankan *command* yang bersamaan disebabkan oleh tidak adanya penanda akhir dari *command* yang masuk.
3. Tingkat keberhasilan sistem mengirimkan pesan ke *administrator* jaringan adalah 88.24% dengan *time* proses pengiriman pesan 5 detik.
4. Tingkat keberhasilan sistem untuk menjalankan *command* yang dikirimkan oleh *administrator* melalui sms adalah 75% dengan *time* proses selama 2 detik.
5. Dengan menggunakan spesifikasi teknis seperti yang digunakan pada skripsi ini, respon sistem pada saat *user* melakukan *login* dan *logout* adalah 42.137 detik dan 40.570 detik.

DAFTAR REFERENSI

[1]. CISCO Networking Academy, "Network Fundamentals", CCNA Exploration 4.0 (CISCO SISTEMS, Inc., 2007).

[2]. www.cisco.com, "Cisco NAC Appliance (Clean Access)", diakses pada 18 February 2009. <http://www.cisco.com/en/US/products/ps6128/products_configuration_example09186a0080a30ad7.shtml#overview>

[3]. "satu lagi startup NAC hangus", diakses pada 2 februari 2009, <http://hardjono.net/2008/03/23/satu-lagi-startup-nac-hangus/>

[4]. Ubuntu Documentation, "ApacheMySQLPHP", diakses pada 14 Maret 2009. <https://help.ubuntu.com/community/ApacheMySQLPHP>

[5]. "VLAN DHCP Cisco 2600 series",diakses pada 3 juni 2009. <http://dewa.cisco.or.id/forum/index.php?PHPSESSID=619d01db988015e0760da2a575baf360&topic=4.msg5#msg5>

[6]. Sukaridhoto, Sritrusta. "Buku Jaringan Komputer", Surabaya.

[7]. "AT *Command*s Interface Guide for x50a", Wavecom, Desember 2004

[8]. Wiharta, D. M, Supranartha, Agus, "Perancanngan dan pembuatan sistem kontrol dengan memanfaatkan layanan sms telepon selular berbasis mikrokontroller AT89C51", Universitas Udayana, Desember, 2005.

[9]. "Installasi Web *Server* di Linux Ubuntu", diakses pada 14 Maret 2009. <http://klublinux.com/installasi-web-*server*-di-linux-ubuntu.html>

[10]. "Pengantar MySQL (Databas MySQL)", diakses pada 14 Maret 2009. <http://the-exploration.net/index.php/pengantar-mysql-database-mysql/>

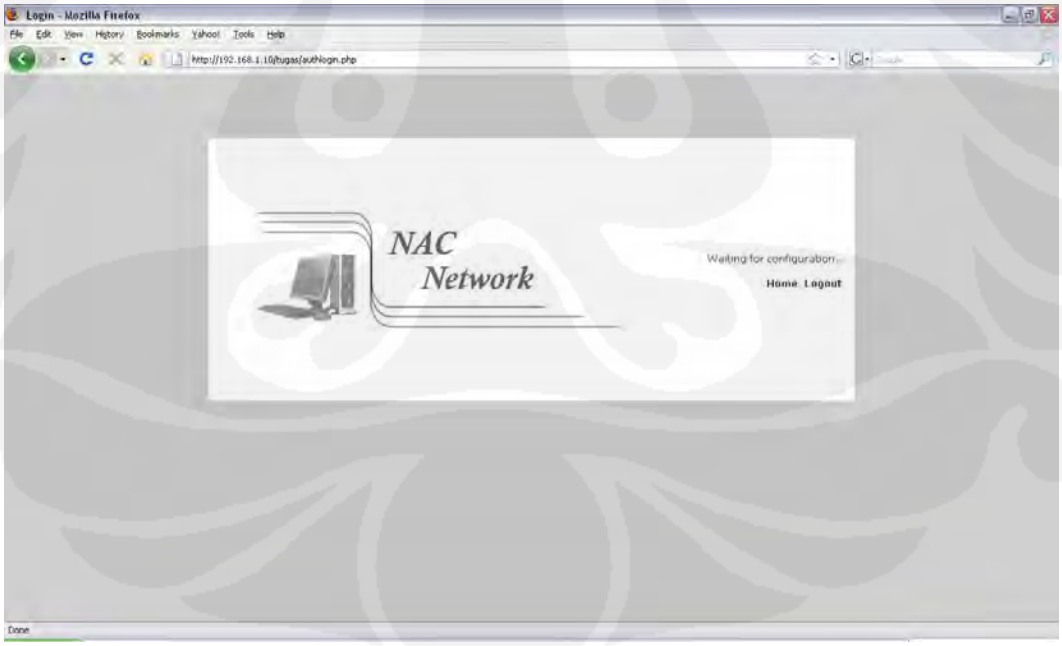
[11]. Ria, Anita Sesar, "Pengenalan PHP", STMIK CIC.

[12]. Galuh, Putra, "Web Portal dengan PHPNuke", Modul SMK-TI & Learning By Doing.

LAMPIRAN 1: *Screen Capture User* Interface Modul

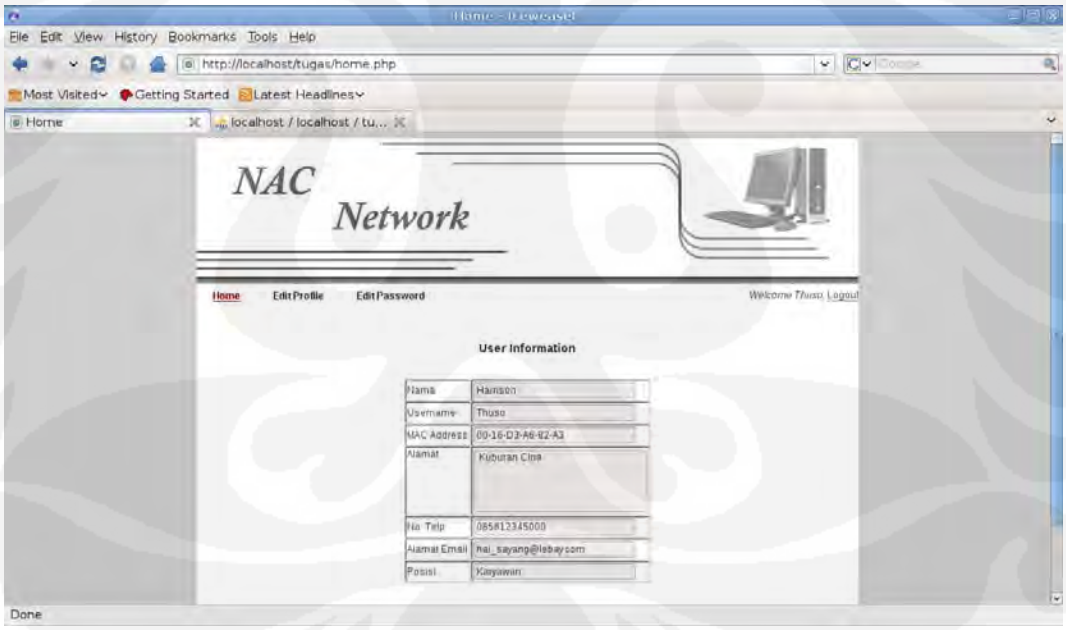
- Tampilan pada saat *login*

- Tampilan pada saat *Login* Berhasil

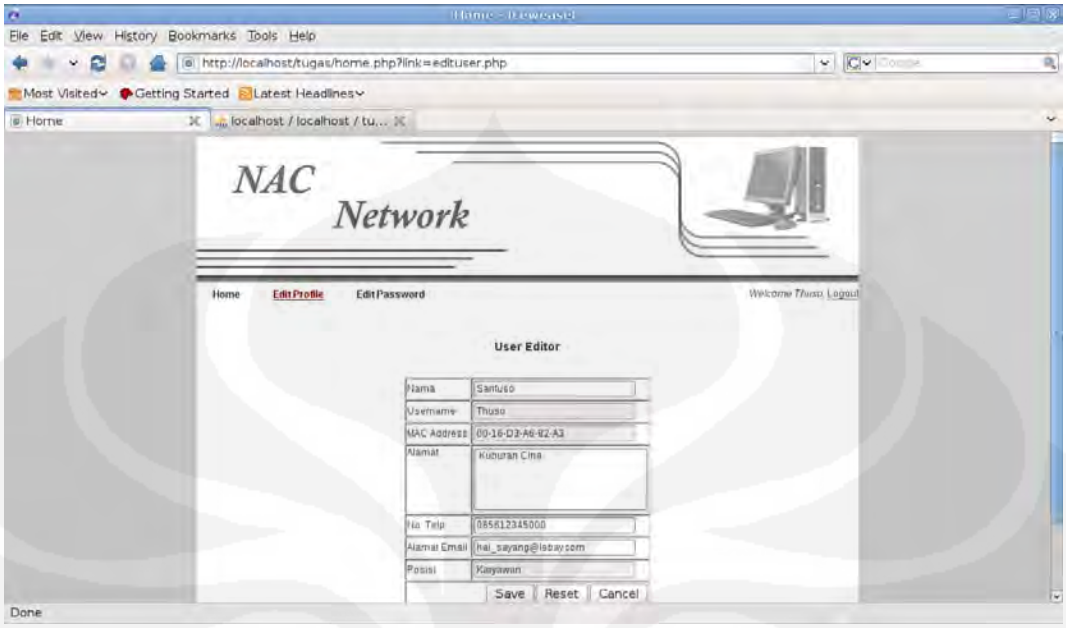


- Tampilan pada saat *user Login* gagal

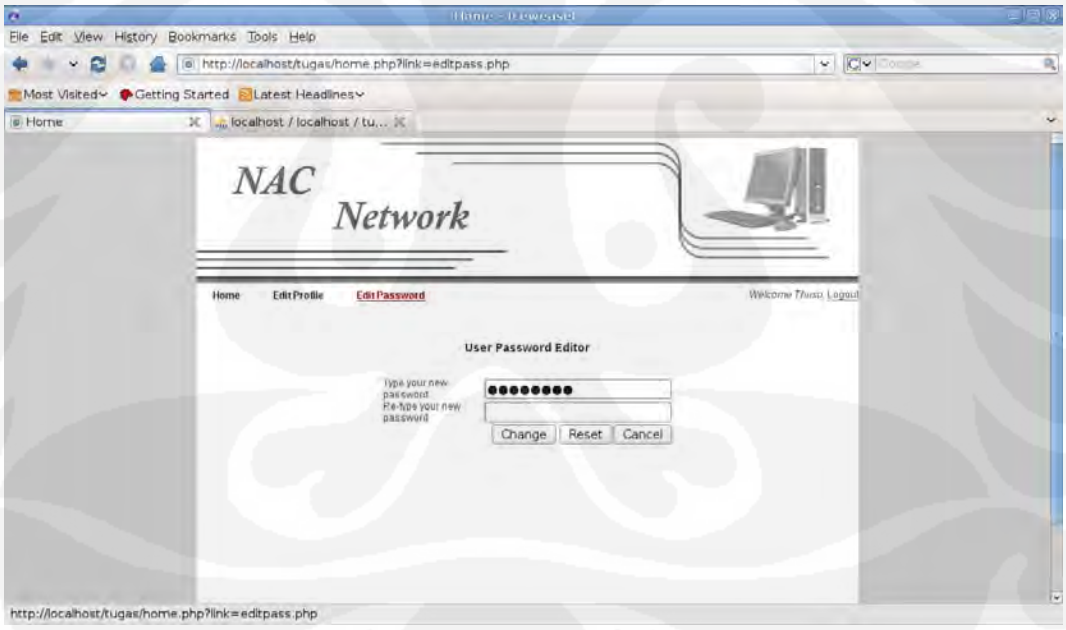
- Tampilan web menu Home untuk *user*



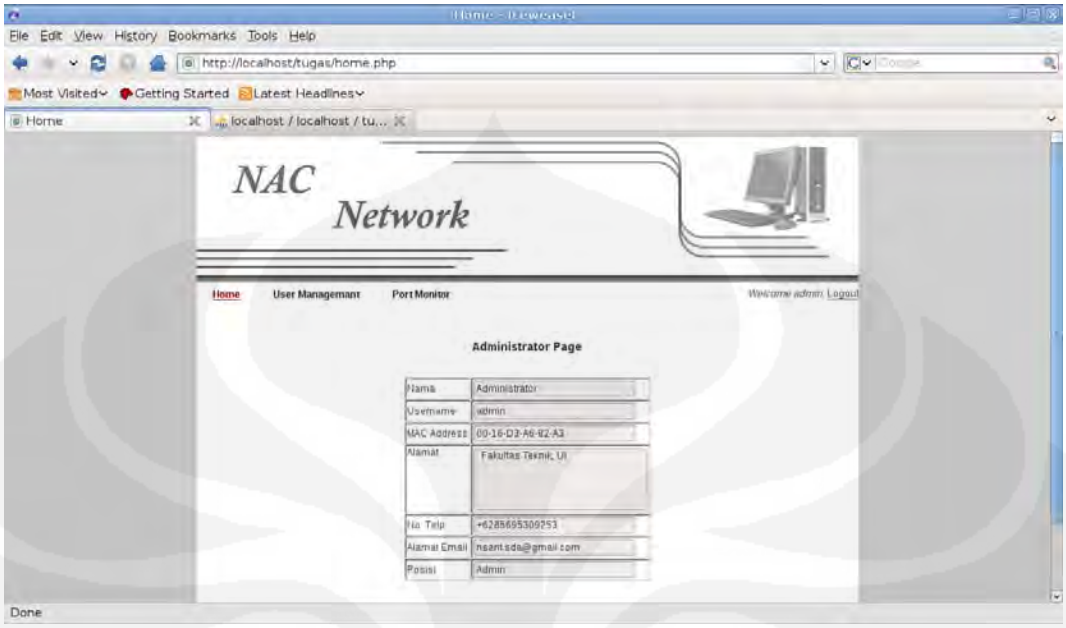
- Tampilan web menu edit *user* profile untuk *user*



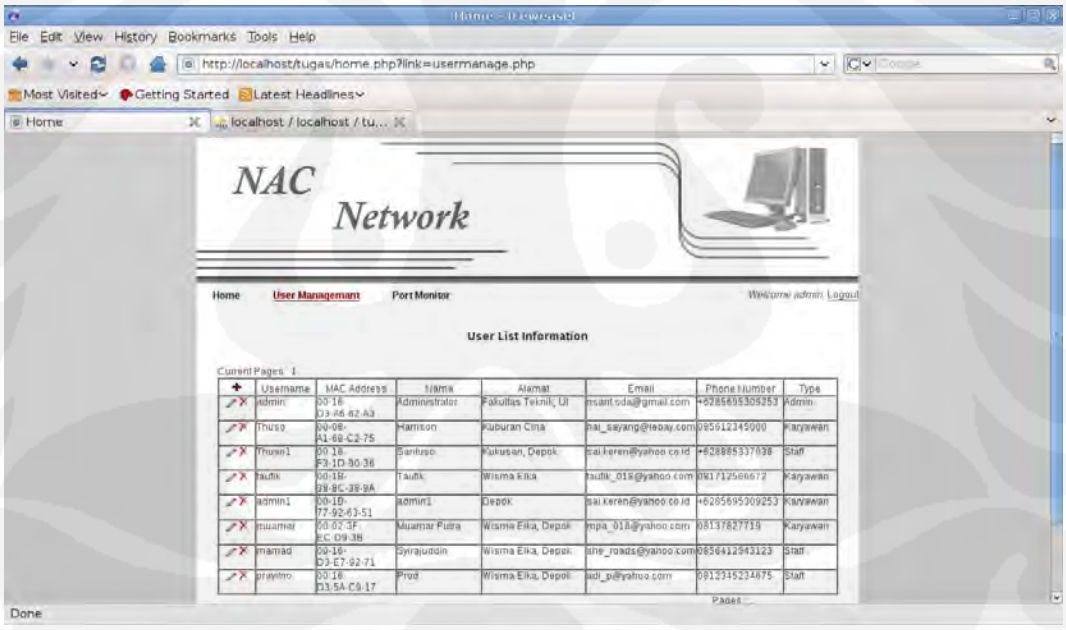
- Tampilan web menu edit *password* untuk *user*



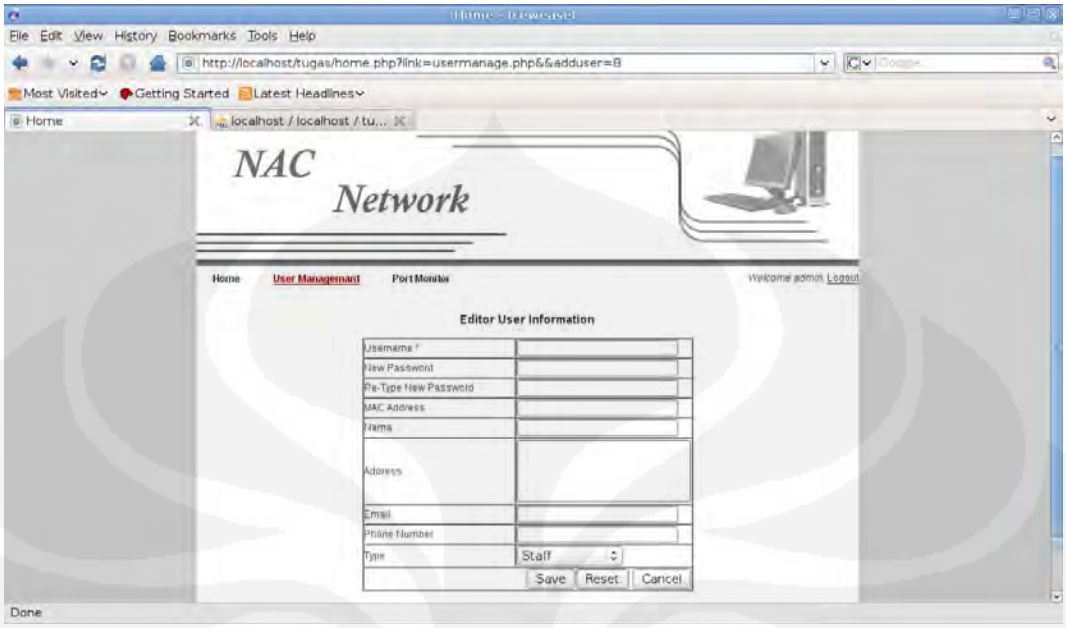
- Tampilan Home untuk *adminstrator* jaringan



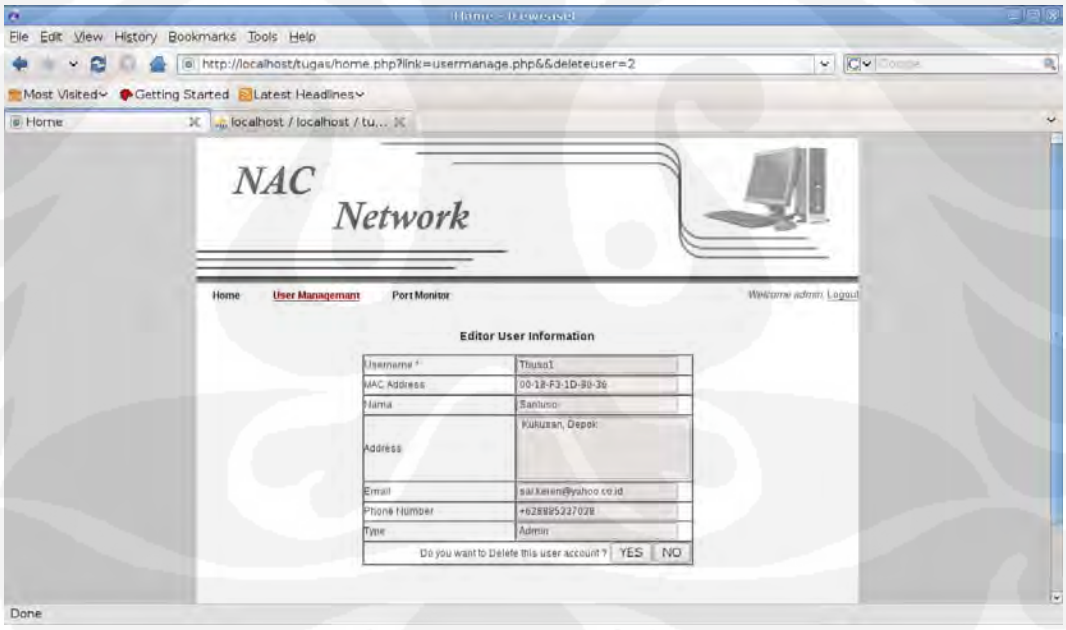
- Tampilan menu *user* management untuk *administrator*



- Tampilan menu add dan edit *user* untuk *administrator*



- Tampilan menu delete *user* untuk *administrator*



LAMPIRAN 2: *Script* Perl *Programming* Pada *Switch* Modul

- getmac.pl

```
#!/usr/bin/perl
my $iphost = $ARGV[0] ;
use Net::Telnet::Cisco ;
open(MYOUTFILE, "> /var/www/tugas/log/swmac.txt");
  my $session = Net::Telnet::Cisco->new(Host => $iphost);
  $session->login('', 'cisco');
  if ($session->enable("cisco") ) {
      @output = $session->cmd('show mac-address-table');
      print MYOUTFILE @output;
  } else {
     warn "Can't enable: " . $session->errmsg;
  }
$session->close;
close(MYOUTFILE);
```

- getarp.pl

```
#!/usr/bin/perl
my $iphost = $ARGV[0] ;
use Net::Telnet::Cisco ;
open(MYOUTFILE, "> /var/www/tugas/log/arpv.txt");
  my $session = Net::Telnet::Cisco->new(Host => $iphost);
  $session->login('', 'cisco');
  if ($session->enable("cisco") ) {
      @output = $session->cmd('show arp');
      print MYOUTFILE @output;
  } else {
     warn "Can't enable: " . $session->errmsg;
  }
$session->close;
close(MYOUTFILE);
```

- getvlan.pl

```
#!/usr/bin/perl
my $iphost = $ARGV[0] ;
use Net::Telnet::Cisco ;
open(MYOUTFILE, "> /var/www/tugas/log/swvlan.txt");
  my $session = Net::Telnet::Cisco->new(Host => $iphost);
  $session->login('', 'cisco');
  if ($session->enable("cisco") ) {
      @output = $session->cmd('show vlan bri');
      print MYOUTFILE @output;
  } else {
     warn "Can't enable: " . $session->errmsg;
  }
$session->close;
close(MYOUTFILE);
```

- scloserev1.pl

```
#/usr/bin/perl
my $iphost = $ARGV[0] ;
my $port = $ARGV[1] ;
my $port1 = "0/". $port;
my $cmd1 = 'interface FastEthernet '. $port1;
print $cmd1."\n";
use Net::Telnet::Cisco;
my $session = Net::Telnet::Cisco->new(Host => $iphost );
$session->login('', 'cisco');
  if ($session->enable("cisco") ) {
            $session->cmd('configure terminal');
            $session->cmd( $cmd1);
            $session->cmd('no mac-address');
            $session->cmd( 'shutdown');
  } else {
     warn "Can't enable: " . $session->errmsg;
  }
  $session->close;
```

- sopenrev1.pl

```
my $vlanid = $ARGV[2] ;
my $macadrs = $ARGV[3] ;
my $port1 = "0/". $port;
my $cmd1 = 'interface FastEthernet '. $port1;
print $cmd1."\n";
my $cmd2 = 'switchport access vlan '. $vlanid;
print $cmd2."\n";
my $cmd3 = 'mac-address '. $macadrs;
print $cmd3."\n";
use Net::Telnet::Cisco;
my $session = Net::Telnet::Cisco->new(Host => $iphost );
$session->login('', 'cisco');
  if ($session->enable("cisco") ) {
            $session->cmd('configure terminal');
            $session->cmd( $cmd1);
            $session->cmd( 'shutdown');
            $session->cmd( $cmd2);
            $session->cmd( $cmd3);
            $session->cmd( 'no shutdown');
  } else {
     warn "Can't enable: " . $session->errmsg;
  }
  $session->close;
```

- s*witch*def.pl

```
#/usr/bin/perl
my $iphost = $ARGV[0] ;
my $port = $ARGV[1] ;
my $vlanid = $ARGV[2] ;
my $port1 = "0/". $port;
my $cmd1 = 'interface FastEthernet '. $port1;
print $cmd1."\n";
```

```
my $cmd2 = 'switchport access vlan '. $vlanid;
print $cmd2."\n";
use Net::Telnet::Cisco;
my $session = Net::Telnet::Cisco->new(Host => $iphost );
$session->login('', 'cisco');
  if ($session->enable("cisco") ) {
            $session->cmd('configure terminal');
            $session->cmd( $cmd1);
            $session->cmd( $cmd2);
            $session->cmd('no mac-address');
            $session->cmd( 'shutdown');
            $session->cmd( 'no shutdown');
  } else {
     warn "Can't enable: " . $session->errmsg;
  }
  $session->close;
```

LAMPIRAN 3: *Script* C *Programming* Pada *Switch* Modul

- Makefile

```
## Makefile smsg
SOURCES.c= server.c
INCLUDES= /usr/include/mysql
CFLAGS=
SLIBS= -lmysqlclient
PROGRAM= swmod
OBJECTS= $(SOURCES.c:.c=.o)
.KEEP_STATE:
debug := CFLAGS= -g
all debug: $(PROGRAM)
$(PROGRAM): $(INCLUDES) $(OBJECTS)
		$(LINK.c) -o $@ $(OBJECTS) $(SLIBS)
clean:
		rm -f $(PROGRAM) $(OBJECTS)
```

- *server*.c

```
#include <termios.h>
#include <sys/ioctl.h>
#include <mysql/mysql.h>
#include "swmod.h"
#include <time.h>
#define BACKLOG 10
FILE * pFile;
char d2log[512];

void init_log(){
	time_t rawtime;
	struct tm * timeinfo;
	char buffert [80], name[256];
	memset(name,0,sizeof(name));
	time ( &rawtime );
	timeinfo = localtime ( &rawtime );
	strftime (buffert, 80, "%Y%m%d",timeinfo);
	sprintf(name,"log/swmod_%s.txt", buffert);
	pFile = fopen (name,"a+");
}

void Writelog(char pesan[1024]){
	time_t rawtime;
	struct tm * timeinfo;
	char buffert [80], ins_text[1014], buffert1 [80];
	time ( &rawtime );
	timeinfo = localtime ( &rawtime );
	init_log();
	strftime (buffert,80,"%x %X",timeinfo);
	sprintf(ins_text,"%s - %s", buffert, pesan);
	fprintf(pFile, "%s\n", ins_text);
	fprintf(stdout, "-> %s\n", pesan);
	fclose(pFile);
}
```

```
opendb(MYSQL *mysql)
{
        mysql_init(mysql);
        if (!mysql_real_connect(mysql, "", "admin", "admin123", "tugasakhir", 0, NULL, 0)) {
                Writelog("Connection to database TPUDB failed");
                return (0);
        }
        return (1);
}

closedb(MYSQL *mysql)
{
        mysql_close(mysql);
        return (0);
}

open_svrport(int *listensock, int port)
{
        int yes=1;
        if ((*listensock = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        Writelog("ERROR : opening socket");
        exit(1);
    }
    if (setsockopt(*listensock,SOL_SOCKET,SO_REUSEADDR,&yes,sizeof(int)) == -1) {
        Writelog("ERROR : setsockopt");
        exit(1);
    }
    my_addr.sin_family = AF_INET; // host byte order
    my_addr.sin_port = htons(port); // short, network byte order
    my_addr.sin_addr.s_addr = INADDR_ANY; // automatically fill with my IP
    memset(&(my_addr.sin_zero), '\0', 8); // zero the rest of the struct
    if (bind(*listensock, (struct sockaddr *)&my_addr, sizeof(struct sockaddr))== -1)
        {
        Writelog("ERROR : bind");
        exit(1);
    }
     if (listen(*listensock, BACKLOG) == -1) {
        Writelog("ERROR : listen");
        exit(1);
    }
        return (*listensock);
}

open_cliport(int *listensock, int port)
{
        struct sockaddr_in servaddr;
        if ((*listensock = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        Writelog("ERROR : opening socket");
        exit(1);
    }

        memset(&servaddr,0,sizeof(servaddr));
        servaddr.sin_family=AF_INET;
        servaddr.sin_port=htons(port);
        inet_pton(AF_INET,"127.0.0.1",&servaddr.sin_addr);
        connect(*listensock,(struct sockaddr*)&servaddr,sizeof(servaddr));
}
```

```
int getIPids server(MYSQL *mysql, char *ip_addr){
        int i, n;
        char msreq[500];
        int portids;

        MYSQL_RES *res;
        MYSQL_ROW row;

        strcpy(msreq, "select value from ctrl_panel where jenis ='ip_ids server' or
jenis='idsport'");
        mysql_query(mysql, msreq);

        res = mysql_store_result(mysql);
        n = mysql_num_rows(res);
        if (n) {
                for (i=0; i<n; i++) {
                        row = mysql_fetch_row(res);
                        if (i==0) strcpy(ip_addr, row[0]);
                        if (i==1) sscanf(row[0], "%d", &portids);
                }
        }
        mysql_free_result(res);
        return (portids);
}

sendto switch(MYSQL *mysql, char *parama){

        char *str1;
        char scomand[10], execp[150], bufer[500];
        int ports, vlans, fl, n;
        char msreq[500], sip_addr[25];
        int portids;

        MYSQL_RES *res;
        MYSQL_ROW row;

        strcpy(msreq, "select value from ctrl_panel where jenis ='ip_ switch ' ");
        mysql_query(mysql, msreq);

        res = mysql_store_result(mysql);
        n = mysql_num_rows(res);
        if (n) {
                bzero(sip_addr, 25);
                row = mysql_fetch_row(res);
                strcpy(sip_addr, row[0]);
        }
        mysql_free_result(res);


        strcpy(bufer, parama);
        str1 = strtok(bufer, "-\n");
        strcpy(scomand, str1);

        fl=0;
        while (1)
    {
         str1 = strtok(NULL, "-\n");
```

```
            if (str1 == NULL) break;

                        if(fl==0) sscanf(str1, "%d", &ports);
                        else if(fl==1) sscanf(str1, "%d", &vlans);
                        fl++;
      }

            if(!strcmp(scomand,"open")){
                        sprintf(execp,"/usr/bin/perl      /var/www/tugas/program/sopen.pl      %s      %d
%d",sip_addr,  ports,  vlans);
                        sprintf(d2log, "exec: %s", execp);
                        Writelog(d2log);
                        sistem(execp);
            }else if(!strcmp(scomand,"close")){
                        sprintf(execp,"/usr/bin/perl      /var/www/tugas/program/sclose.pl      %s      %d",
sip_addr, ports);
                        sprintf(d2log, "exec: %s", execp);
                        Writelog(d2log);
                        sistem(execp);
            }else if(!strcmp(scomand,"switch")){
                        sprintf(execp,"/usr/bin/perl   /var/www/tugas/program/sswitch.pl   %s   %d   %d",
sip_addr, ports,  vlans);
                        sprintf(d2log, "exec: %s", execp);
                        Writelog(d2log);
                        sistem(execp);
            }else {
                        sprintf(d2log, " %s", scomand);
                        Writelog(d2log);
            }
}

sendtoadmins(MYSQL *mysql, char *parama){
            char  msreq[500],  msg[500],  bufer[500],  bufer1[100],  bufer2[100],  bufer3[100],
sourceip[25], pcomand[50];
            int portidg, cfdcmd, n, priority, sport;
            char *str1, *str2;

            MYSQL_RES *res;
            MYSQL_ROW row;

            strcpy(bufer, parama);
            str1 = strtok(bufer,"|\n");
            strcpy(bufer1, str1);
            str1 = strtok(NULL,"|\n");
            strcpy(bufer2, str1);
            str1 = strtok(bufer1," ");
            while (1)
      {
             str1 = strtok(NULL, " ");
             if (str1 == NULL) break;
                        sscanf(str1, "%d", &priority);
      }

            str1 = strtok(bufer2," ");
            while (1)
 {
  str1 = strtok(NULL, " ");
```

```
  if (str1 == NULL) break;
                strcpy(sourceip, str1);
}
        strcpy(msreq, "/usr/bin/curl 'http://localhost/tugas/getmac_switch.php'");
sistem(msreq);

        strcpy(msreq, "/usr/bin/curl 'http://localhost/tugas/cekarp.php'");
        sistem(msreq);

        sprintf(msreq, "select port_n from adm_swport  where ip_user ='%s' ", sourceip);
        mysql_query(mysql, msreq);
        res = mysql_store_result(mysql);
        n = mysql_num_rows(res);
        if (n) {
                row = mysql_fetch_row(res);
                sscanf(row[0], "%d", &sport);
        }
        mysql_free_result(res);
                n=0;
                strcpy(msreq, "select value from ctrl_panel where jenis ='port_socket' ");
                mysql_query(mysql, msreq);
                res = mysql_store_result(mysql);
                n = mysql_num_rows(res);
                if (n) {
                        row = mysql_fetch_row(res);
                        sscanf(row[0], "%d", &portidg);
                }
                mysql_free_result(res);

                switch (priority){
                        case 0:
                                open_cliport(&cfdcmd, portidg);
                                sprintf(msg,"Admin, Intrusion detection from %s, priority
%d", sourceip, priority);
                                write(cfdcmd,msg,strlen(msg));
                                close(cfdcmd);
                                cfdcmd = 0;
                                sprintf(pcomand,"close-%d", sport);
                                sendtoswitch(mysql, pcomand);
                                break;
                        case 1:
                                open_cliport(&cfdcmd, portidg);
                                sprintf(msg,"Admin, Intrusion detection from %s, priority
%d", sourceip, priority);
                                write(cfdcmd,msg,strlen(msg));
                                close(cfdcmd);
                                cfdcmd = 0;
                                sprintf(pcomand,"close-%d", sport);
                                sendtoswitch(mysql, pcomand);
                                break;
                        case 2:
                                open_cliport(&cfdcmd, portidg);
                                sprintf(msg,"Admin, Intrusion detection from %s, priority
%d", sourceip, priority);
                                write(cfdcmd,msg,strlen(msg));
                                close(cfdcmd);
                                cfdcmd = 0;
                                sprintf(pcomand,"switch-%d-%d", sport, 1);
```

```
                                        sendto*switch*(mysql, pcomand);
                                        break;
                                case 3:
                                        open_cliport(&cfdcmd, portidg);
                                        sprintf(msg,"Admin, Intrusion detection from %s, priority
%d", sourceip, priority);
                                        write(cfdcmd,msg,strlen(msg));
                                        close(cfdcmd);
                                        cfdcmd = 0;
                                        break;
                                default:
                                        break;
                        }
}

fromweb(MYSQL *mysql, char *parama){
        char *str1, ip[32], mac[32], execp[256], buffer1[500];
        int k, port, vlan;

        strcpy(buffer1, parama);
        str1 = strtok(buffer1,"|");
        strcpy(ip, str1);
        k=1;
        while (1)
  {
   str1 = strtok(NULL, "|");
   if (str1 == NULL) break;
                        if(k==1)
                                sscanf(str1, "%d", &port);
                        else if(k==2)
                                sscanf(str1, "%d", &vlan);
                        else if(k==3)
                                strcpy(mac, str1);
                        k++;
  }
        sprintf(execp,"/usr/bin/perl  /var/www/tugas/program/sopenrev1.pl  %s  %d  %d  %s",ip,
port,  vlan, mac);
        Writelog(execp);
        sistem(execp);
}

fromwebdef(MYSQL *mysql, char *parama){
        char *str1, ip[32], execp[256], buffer1[500];
        int k, port, vlan;

        strcpy(buffer1, parama);
        str1 = strtok(buffer1,"|");
        strcpy(ip, str1);
        k=1;
        while (1)
      {
         str1 = strtok(NULL, "|");
         if (str1 == NULL) break;
                        if(k==1)
                                sscanf(str1, "%d", &port);
                        else if(k==2)
                                sscanf(str1, "%d", &vlan);
                        k++;
```

```
	}
		sprintf(execp,"/usr/bin/perl /var/www/tugas/program/sswitchdef.pl %s %d %d",ip, port, vlan);
		Writelog(execp);
		sistem(execp);
}

fromwebclose(MYSQL *mysql, char *parama){
		char *str1, ip[32], execp[256], buffer1[500];
		int k, port, vlan;

		strcpy(buffer1, parama);
		str1 = strtok(buffer1,"|");
		strcpy(ip, str1);
		k=1;
		while (1)
	{
		 str1 = strtok(NULL, "|");
		 if (str1 == NULL) break;
			if(k==1)
				sscanf(str1, "%d", &port);
			k++;
	}
		sprintf(execp,"/usr/bin/perl /var/www/tugas/program/scloserev1.pl %s %d",ip, port);
		Writelog(execp);
		sistem(execp);
}

int main(){
		MYSQL mysql;
		int val, listencmd,fdcmd=0, maxd, portids;
		int n, len, lenm, i, nread, nwrite, cont=0, j, ipaddrs[25], ipclient[25] ;
		fd_set rset, allset;
		struct timeval waittime;
		struct sockaddr_in their_addr; // connector's address information
		int sin_size;
		char req[500], rep[500], cdata[200], xreq[500];
		char *str1;

		Writelog("Start");

		opendb(&mysql);
		portids = getIPidsserver(&mysql, &ipaddrs);
		open_svrport(&listencmd, portids);

		FD_ZERO(&rset);
		FD_ZERO(&allset);
		FD_SET(listencmd, &allset);
		maxd = max(0, listencmd);


		while (1) {
			rset = allset;
			waittime.tv_sec = 1;
			waittime.tv_usec = 0;

			n = select(maxd+1, &rset, NULL, NULL, &waittime);
			if (n < 0) { //Interrupt
```

```
                } else if (!n) {  //Timeout
                } else { //Data

                        while (n > 0) {
                                if (FD_ISSET(listencmd, &rset) && listencmd) {
                                        sin_size = sizeof(struct sockaddr_in);
                                        if ((fdcmd = accept(listencmd, (struct sockaddr *)&their_addr,&sin_size)) == -1) {
                                                Writelog("ERROR : accept");
                                                continue;
                                        }
                                        strcpy(ipclient, inet_ntoa(their_addr.sin_addr));
                                        if (fdcmd > 0) {
                                                val = fcntl(fdcmd, F_GETFL, 0);
                                                fcntl(fdcmd, F_SETFL, val | O_NONBLOCK);
                                                maxd = max(maxd, fdcmd);
                                                FD_SET(fdcmd, &allset);
                                        } else {
                                                sprintf(d2log, "ERROR : %s", strerror(errno));
                                                Writelog(d2log);
                                        }
                                        n--;
                                        if (n <= 0) break;
                                }
                                if (FD_ISSET(fdcmd, &rset) && fdcmd) {
                                        len = 0;
                                        while ((nread = read(fdcmd, &req[len], 500-len)) > 0) len += nread;
                                        if (!len) {  //Disconnected
                                                FD_CLR(fdcmd, &allset);
                                                if (maxd == fdcmd) {
                                                        maxd = max(listencmd, 0);
                                                }
                                                close(fdcmd);
                                                fdcmd = 0;
                                        } else {
                                                req[len] = 0;
                                                if(req[len-1]=='\n') req[len-1] = 0;
                                                if(!strcmp(ipaddrs, ipclient)){
                                                        sprintf(d2log, "From IDS Server -> read %d bytes [%s] ", strlen(req), req);
                                                        Writelog(d2log);
                                                        if (!strncmp(req, "IDSS", 4)) {
                                                                str1 = strtok(req, ",\n");
                                                                str1 = strtok(NULL, ",\n");
                                                                sendtoadmins(&mysql, str1);
                                                        }
                                                }
                                                if(!strcmp("127.0.0.1", ipclient)){
                                                        sprintf(d2log, "From Policy Server -> read %d bytes [%s] ", strlen(req), req);
                                                        Writelog(d2log);
                                                        if (!strncmp(req, "SMSG", 4)) {
                                                                str1 = strtok(req, ",\n");
                                                                str1 = strtok(NULL, ",\n");
```

```
                                                    sendto*switch*(&mysql,
str1);
                                                }
                                        }
                                        if(!strncmp(req, "WEB", 3)){
                                                sprintf(d2log, "From WEB -> read
%d bytes [%s] ", strlen(req), req);
                                                Writelog(d2log);
                                                str1 = strtok(req, ",\n");
                                                str1 = strtok(NULL, ",\n");
                                                fromweb(&mysql, str1);
                                        }
                                        if(!strncmp(req, "WDEF", 4)){
                                                sprintf(d2log, "From WEB -> read
%d bytes [%s] ", strlen(req), req);
                                                Writelog(d2log);
                                                str1 = strtok(req, ",\n");
                                                str1 = strtok(NULL, ",\n");
                                                fromwebdef(&mysql, str1);
                                        }
                                        if(!strncmp(req, "WCLO", 4)){
                                                sprintf(d2log, "From WEB -> read
%d bytes [%s] ", strlen(req), req);
                                                Writelog(d2log);
                                                str1 = strtok(req, ",\n");
                                                str1 = strtok(NULL, ",\n");
                                                fromwebclose(&mysql, str1);
                                        }
                                        printf("read %d bytes [%s] \n", strlen(req),
req);
                                }
                                n--;
                                if (n <= 0) break;
                        }
                }
        }
        closedb(&mysql);
        return 0;
}
```