

Wireless LAN access using MAC-based authentication with Mikrotik and FreeRADIUS

Tugas Akhir

Disampaikan Sebagai Bagian Dari Persyaratan Kelulusan Diploma 3
Program Studi Teknik Komputer

Oleh :

Hepriyanti Fransiska Siahaan	11106008
Sunardo Panjaitan	11106016
Sanhenra Sinaga	11106071



Politeknik Informatika Del

2009

Lembar Pengesahan Tugas Akhir

Politeknik Informatika Del

Wireless LAN access using MAC-based authentication with Mikrotik and FreeRADIUS

Oleh:

Hepriyanti Fransiska Siahaan	11106008
Sunardo Panjaitan	11106016
Sanhenra Sinaga	11106071

**Dinyatakan memenuhi syarat dan karenanya disetujui dan disahkan sebagai
Laporan Tugas Akhir Diploma 3
Program Studi Teknik Komputer**

Pembimbing

Ramot Lubis, S.T., M.Sc.

NIDN. 0130047801

Prakata

Puji dan syukur kepada Tuhan Yang Maha Esa atas rahmat-Nya yang menyertai penulis selama penulisan naskah ini, sehingga setiap tahapan yang direncanakan untuk penyusunan naskah ini dapat dilalui dengan baik.

Naskah ini ditulis sebagai bagian dari syarat kelulusan Diploma 3 Politeknik Informatika Del. Maksud penulisan naskah ini adalah untuk membantu pembaca mengetahui bagaimana cara yang dapat dilakukan untuk mengatur penggunaan *wireless* LAN yang sifatnya terbuka sehingga hanya ditujukan pada pengguna tertentu yang sudah terdaftar.

Penulis mengucapkan terima kasih kepada Bapak Ramot Lubis sebagai pembimbing, yang telah memberikan arahan dan bimbingan dalam menyelesaikan naskah ini. Terimakasih juga penulis sampaikan kepada Ibu Rosni Lumbantoruan sebagai Koordinator Tugas Akhir atas saran dan waktu yang diberikan selama pengerjaan naskah ini. Ucapan terimakasih juga penulis sampaikan kepada semua pihak atas bantuan yang diberikan kepada penulis selama penyelesaian penulisan naskah ini.

Penulis mengharapkan naskah ini dapat memberikan informasi kepada pembaca mengenai penggunaan Mikrotik dan FreeRADIUS pada *wireless* LAN dengan autentikasi *wireless client* berdasarkan MAC address.

Penulis menyadari bahwa naskah ini tidak lepas dari kesalahan, oleh karena itu penulis mengharapkan kritik dan saran yang bersifat membangun untuk perbaikan dokumen ini.

Sitoluama, 28 Agustus 2009

Hepriyanti Fransiska Siahaan
Sunardo Panjaitan
Sanhenra Sinaga

Abstrak

Perkembangan teknologi *wireless* saat ini sudah semakin pesat karena penggunaan layanan *wireless* yang cukup mudah. Salah satu contoh penggunaan teknologi *wireless* adalah *wireless* LAN. Penggunaan *wireless* LAN yang mudah, menjadi daya tarik tersendiri bagi para pengguna komputer untuk menggunakan teknologi ini untuk mengakses suatu jaringan komputer atau internet.

Masalah yang sering dihadapi apabila menerapkan *wireless* LAN adalah tentang keamanannya. Hal ini disebabkan oleh *wireless* LAN yang sifatnya terbuka (*open access*) sehingga setiap perangkat yang memiliki kartu *wireless* (*wireless card*) dapat mendeteksi suatu jaringan *wireless* secara langsung. Sebagian besar penyedia *wireless* yang tidak menerapkan sistem keamanan yang memadai, memungkinkan pengguna yang tidak berhak (ilegal) dapat masuk ke jaringan komputer tersebut. Untuk mengatasi masalah tersebut, maka perlu dilakukan autentikasi sehingga pengguna *wireless* LAN hanya ditujukan pada pengguna tertentu yang sudah terdaftar. Hal ini akan membantu penyedia layanan tersebut untuk mengamankan sistem jaringan *wireless* yang dibuatnya. Jenis autentikasi ada beberapa macam, sebagai contoh yaitu autentikasi dengan menggunakan *username* dan *password* seperti pada layanan *email*. Dalam kajian ini, autentikasi dilakukan berdasarkan *MAC address* (ID perangkat), dimana pengguna yang dapat mengakses *wireless* LAN adalah pengguna yang *MAC address*-nya sudah terdaftar pada penyedia *wireless* LAN tersebut.

Pada kajian ini, studi tentang *Wireless LAN access using MAC-based authentication with Mikrotik and FreeRADIUS* diharapkan dapat membantu untuk implementasi jaringan *wireless* (WLAN) dengan menggunakan autentikasi berdasarkan *MAC address*.

Kata kunci : *Wireless* LAN, Mikrotik, FreeRADIUS, *MAC Address*

Daftar Isi

Prakata.....	2
Abstrak.....	3
Daftar Isi	4
Daftar Tabel	6
Daftar Gambar	7
Bab I Pendahuluan.....	8
1.1 Latar Belakang.....	8
1.2 Tujuan	9
1.3 Lingkup	9
1.4 Pendekatan	9
1.5 Singkatan dan Istilah.....	10
1.6 Sistematika Penyajian	11
Bab II Tinjauan Pustaka.....	13
2.1 Wireless LAN	13
2.1.1 Keuntungan menggunakan WLAN.....	14
2.1.2 Kekurangan menggunakan WLAN.....	15
2.2 MAC Address	15
2.3 Protokol AAA	18
2.3.1 Authentication.....	18
2.3.2 Authorization	19
2.3.3 Accounting.....	20
2.4 Access Point.....	21
2.4.1 WEP (Wired Equivalent Privacy).....	21
2.4.2 WPA (Wi-Fi Protected Access).....	22
2.4.3 WPA-Personal.....	22
2.4.4 WPA2-Personal.....	23
2.4.5 WPA2-Personal Mixed	23
2.4.6 WPA-Enterprise.....	23
2.4.7 WPA2-Enterprise.....	24
2.4.8 WPA2-Enterprise Mixed	24

2.5	Remote Authentication Dial-In User Service (RADIUS).....	25
2.6	FreeRADIUS.....	27
2.7	Mikrotik Router OS	28
Bab III	Analisis dan Perancangan	31
3.1	Analisis	31
3.1.1	Mekanisme Autentikasi Server RADIUS	31
3.2	Perancangan	34
3.2.1	Perancangan perangkat yang akan digunakan	34
3.2.2	Perancangan topologi jaringan.....	35
Bab IV	Pelaksanaan.....	38
4.1	Instalasi	38
4.1.1	Instalasi Mikrotik Router OS	38
4.1.2	Instalasi FreeRADIUS	39
4.2	Konfigurasi	40
4.2.1	Konfigurasi Mikrotik Router OS	41
Bab V	Pengujian.....	44
5.1	Pengujian Koneksi Langsung	44
5.2	Pengujian Koneksi dengan Menggunakan Autentikasi MAC Address	45
Bab VI	Kesimpulan dan Saran	47
6.1	Kesimpulan	47
6.2	Saran	47
Daftar Pustaka dan Rujukan.....		48
Daftar Pustaka		48
Rujukan		49
Lampiran		50
Lampiran A		50
Lampiran B		95
Lampiran C		97
Lampiran D		98
Lampiran E.....		99

Daftar Tabel

Tabel 1 . Daftar Singkatan	10
Tabel 2 . Daftar Istilah	10
Tabel 3 . Kode Paket RADIUS dalam desimal.....	26
Tabel 4 . Fitur-fitur Mikrotik	29

Daftar Gambar

Gambar 1 . Ethernet pada OSI Layer.....	16
Gambar 2 . MAC Address (<i>Physical Address</i>)	17
Gambar 3 . Struktur MAC Address.....	18
Gambar 4 . Proses Autentikasi.....	19
Gambar 5 . Arsitektur jaringan AAA.....	20
Gambar 6 . Struktur format data RADIUS	26
Gambar 7 . Mekanisme Autentikasi RADIUS <i>Server</i>	32
Gambar 8 . Autentikasi dan Autorisasi RADIUS	32
Gambar 9 . Operasi <i>Filtering Mac Address</i>	33
Gambar 10 . Topologi dasar jaringan 1	36
Gambar 11 . Topologi dasar jaringan 2	37
Gambar 12 . Instalasi Mikrotik	39
Gambar 13 . Instalasi FreeRADIUS	40
Gambar 14 . <i>List jaringan wireless</i>	44
Gambar 15 . Alamat IP pengguna.....	45

Bab I

Pendahuluan

Bab Pendahuluan berisi uraian mengenai latar belakang pemilihan topik, tujuan, dan lingkup pelaksanaan kajian, serta pendekatan yang dilakukan dalam menyelesaikan persoalan yang menjadi kajian Tugas Akhir.

1.1 Latar Belakang

Wireless LAN bersifat terbuka (*open access*) dimana semua pengguna dapat menggunakan layanan *wireless* tersebut selama berada dalam jangkauan. Karena sifatnya terbuka dimana setiap perangkat yang memiliki kartu *wireless* (*Wireless Card*) dapat mendeteksi suatu jaringan *wireless* secara langsung, maka perlu dilakukan autentikasi sehingga pengguna *wireless* LAN hanya ditujukan pada pengguna tertentu yang sudah terdaftar. Autentikasi dilakukan berdasarkan *MAC address*, dimana pengguna yang dapat melakukan akses terhadap jaringan adalah pengguna yang *MAC address*-nya sudah terdaftar pada penyedia *wireless* LAN tersebut.

Alasan menggunakan autentikasi berdasarkan *MAC address* adalah ada kondisi tertentu dimana penggunaan *MAC address* lebih efisien dibanding dengan autentikasi lain, seperti penggunaan *username* dan *password*.

Mikrotik dan FreeRADIUS digunakan sebagai aplikasi yang berperan untuk melakukan autentikasi berdasarkan *MAC address*. Mikrotik digunakan disebabkan memiliki beberapa fitur yang mendukung untuk studi implementasi pada kajian ini seperti proses *routing*, DHCP, dan sebagainya, kemudian administrasinya yang mudah dan tidak membutuhkan *resources* yang besar, seperti pada spesifikasi perangkat yang digunakan. Administrasi Mikrotik bisa dilakukan melalui terminal, SSH, *Windows application* (WinBox) atau *web based*. FreeRADIUS merupakan aplikasi *Open Source Software* yang mendukung protokol RADIUS akan digunakan sebagai sistem yang mengatur proses autentikasi. Penggunaan Mikrotik dan FreeRADIUS diharapkan mampu menyediakan akses terhadap suatu *wireless client* dengan menggunakan autentikasi berdasarkan *MAC address*.

1.2 Tujuan

Tujuan dari pelaksanaan Tugas Akhir ini adalah untuk melakukan studi implementasi *wireless* LAN menggunakan autentikasi berdasarkan *MAC address* dengan mengintegrasikan Mikrotik dan FreeRADIUS. Autentikasi yang dimaksud adalah autentikasi secara terpusat (*centralized authentication*) sehingga mempermudah proses administrasi, *monitoring*, dan *maintenance* terhadap *wireless client*.

Pelaksanaan Tugas Akhir ini juga diharapkan dapat menyediakan panduan bagi pembaca agar pembaca dapat mengimplementasikan jaringan *wireless* dengan menggunakan autentikasi berdasarkan *MAC address* dengan mengintegrasikan Mikrotik dan FreeRADIUS.

1.3 Lingkup

Lingkup yang dibahas pada kajian ini yaitu studi implementasi autentikasi berdasarkan *MAC address* pada *wireless* LAN dengan mengintegrasikan Mikrotik dan FreeRADIUS. Dalam pelaksanaan studi implementasi ini, Mikrotik digunakan sebagai sistem operasi independen berbasis Linux khusus untuk PC Router dan FreeRADIUS sebagai suatu aplikasi yang mendukung protokol RADIUS yang digunakan sebagai sistem yang mengatur proses autentikasi pengguna terhadap suatu jaringan *wireless*. Pada pelaksanaan kajian ini juga akan menyediakan panduan bagi pembaca, sehingga pembaca dapat mengimplementasikan jaringan *wireless* dengan menggunakan autentikasi berdasarkan *MAC address* dengan mengintegrasikan Mikrotik dan FreeRADIUS. Panduan yang dimaksud yaitu hasil akhir dari naskah pelaksanaan kajian ini.

1.4 Pendekatan

Pendekatan yang dilakukan dalam menyelesaikan kajian Tugas Akhir ini, antara lain:

1. Melakukan studi literatur dengan mempelajari bahan mengenai *wireless* LAN, *MAC address*, Mikrotik Router OS, Protokol RADIUS, dan FreeRADIUS.
2. Melakukan analisis mengenai cara kerja Mikrotik Router OS dan FreeRADIUS, yang akan digunakan untuk implementasi *wireless* LAN.
3. Melakukan implementasi yang berupa instalasi Mikrotik Router OS dan FreeRADIUS serta melakukan konfigurasi. Tahapan berikutnya adalah mengintegrasikan Mikrotik Router OS dan FreeRADIUS sehingga dapat digunakan untuk implementasi *wireless* LAN dengan autentikasi berdasarkan *MAC address*.

4. Melakukan dokumentasi mengenai kegiatan yang dilakukan selama implementasi.
5. Melakukan pengujian terhadap *wireless* LAN yang telah diimplementasikan.

1.5 Singkatan dan Istilah

Daftar singkatan yang digunakan dalam dokumen diuraikan pada Tabel 1.

Tabel 1 . Daftar Singkatan

No	Singkatan	Defenisi
1.	BIA	<i>Burned-in Address</i>
2.	CHAP	<i>Challenge handshake Authentication Protocol</i>
3.	CSMA/CA	<i>Carrier Sense Multiple access with Collision Avoidance</i>
4.	DEC	<i>Digital Equipment Corporation</i>
5.	DOM	<i>Disk On Module</i>
6.	EAP	<i>Extensible Authentication Protocol</i>
7.	FCC	<i>Federal Communication Commission</i>
8.	IEEE	<i>Institute of Electrical and Electronics Engineers</i>
9.	ISM	<i>Industrial, Scientific, and Medical</i>
10.	IV	<i>Initialization Vector</i>
11.	LAN	<i>Local Area Network</i>
12.	MAC	<i>Media Access Control</i>
13.	NAS	<i>Network Access Server</i>
14.	OFDM	<i>Orthogonal Frequency Division Multiplexing</i>
15.	OUI	<i>Organizational Unique Identifier</i>
16.	PAM	<i>Password Authentication Module</i>
17.	PAP	<i>Password Authentication Protocol</i>
18.	PSTN	<i>Public Switched Telephone Network</i>
19.	RADIUS	<i>Remote Authentication Dial-In User Service</i>
20.	RF	<i>Radio Frequency</i>
21.	TKIP	<i>Temporal Key Integrity Protocol</i>
22.	WEP	<i>Wired Equivalent Privacy</i>
23.	WPA	<i>Wi-Fi Protected Access</i>
24.	CA	<i>Certificate Authority</i>
25.	DHCP	<i>Dynamic Host Control Protocol</i>

Daftar istilah yang digunakan dalam dokumen diuraikan pada Tabel 2.

Tabel 2 . Daftar Istilah

No	Istilah	Defenisi
1.	Mikrotik	Sistem operasi independen (tidak terbatas pada <i>vendor hardware</i> spesifik) berbasis Linux khusus untuk komputer yang difungsikan sebagai Router.
2.	FreeRADIUS	Aplikasi unix yang memungkinkan seseorang untuk mengatur protokol RADIUS, yang dapat digunakan untuk autentikasi, otorisasi dan akuntansi berbagai jenis akses jaringan.
3.	MAC <i>address</i>	Sebuah alamat jaringan yang diimplementasikan pada lapisan

No	Istilah	Defenisi
		<i>data-link</i> dalam tujuh lapisan model OSI, yang merepresentasikan sebuah <i>node</i> tertentu dalam jaringan.
4.	RADIUS	Suatu protokol yang dikembangkan untuk menyediakan akses terpusat untuk melakukan <i>authentication</i> , <i>authorization</i> , dan <i>accounting</i> .
5.	PAP	Sebuah protokol yang dalam proses autentikasinya <i>password</i> tidak dienkripsi.
6.	CHAP	Sebuah protokol yang dalam proses autentikasinya <i>password</i> terlebih dahulu dienkripsi.
7.	WPA	Protokol yang dikembangkan oleh <i>Wi-Fi Alliance</i> sebagai respon dari banyaknya kelemahan yang ditemukan pada sistem sebelumnya, <i>Wired Equivalent Privacy</i> (WEP).
8.	WEP	Suatu metoda pengamanan jaringan nirkabel, disebut juga dengan <i>Shared Key Authentication</i> .
9.	TKIP	Sebuah protokol hasil pengembangan dari WPA.
10.	AES	Sebuah algoritma simetrik blok cipher yang ditujukan untuk menggantikan algoritma DES yang menggunakan 128 bit data dan menggunakan <i>key</i> 128/192/256 bit
11.	NAS	Sebuah <i>server</i> yang berguna untuk meneruskan informasi pengguna ke RADIUS <i>server</i> , kemudian mengembalikan respon ke pengguna.
12.	LAN	Jaringan komputer yang jaringannya hanya mencakup wilayah kecil; seperti jaringan komputer kampus, gedung, kantor, dalam rumah, sekolah atau yang lebih kecil.
13.	DHCP	Pemberian IP <i>address</i> secara dinamis.

1.6 Sistematika Penyajian

Setelah uraian pada bab Pendahuluan yang membahas latar belakang, tujuan, lingkup, pendekatan dan sistematika penyajian, materi pembahasan selanjutnya mengikuti sistematika sebagai berikut.

Bab Tinjauan Pustaka menjelaskan landasan teori mengenai *wireless* LAN, MAC *address*, Protokol AAA, RADIUS, FreeRADIUS, dan Mikrotik Router OS.

Pada bab Analisis dan Desain menjelaskan analisis yang dilakukan selama pelaksanaan kajian ini yang mencakup analisis tentang cara kerja FreeRADIUS serta rancangan topologi jaringan yang akan diimplementasikan pada *wireless* LAN.

Bab Pelaksanaan mendeskripsikan proses yang dilakukan dalam melaksanakan kajian yaitu berupa instalasi dan konfigurasi Mikrotik Router OS dan FreeRADIUS, serta integrasi antara Mikrotik dan FreeRADIUS.

Bab Pengujian berisi penjelasan tentang hasil yang telah dicapai selama pelaksanaan kajian serta pengujian terhadap hasil kajian. Pada bab ini juga akan dijelaskan masalah-masalah yang muncul selama pelaksanaan kajian.

Bab Kesimpulan dan Saran memberikan uraian kesimpulan yang diperoleh dari kajian, dan saran-saran yang dianjurkan untuk pengembangan implementasi selanjutnya.

Bab II

Tinjauan Pustaka

Bab Tinjauan Pustaka mencakup teori mengenai *wireless* LAN, *MAC address*, Protokol AAA, RADIUS, FreeRADIUS dan Mikrotik Router OS.

2.1 Wireless LAN

Local Area Network biasa disingkat LAN adalah jaringan komputer yang jaringannya hanya mencakup wilayah kecil, seperti jaringan komputer kampus, gedung, kantor, dalam rumah, sekolah atau yang lebih kecil [LAN]. *Local Area Network* menghubungkan dua atau lebih *workstation* atau perangkat lain pada sebuah bangunan atau daerah yang secara geografis terbatas [CC1]. Dalam menghubungkan *workstation* atau perangkat tersebut, dibutuhkan media transmisi. Berdasarkan bentuknya, media transmisi memiliki 2 bentuk yaitu :

1. *Guided media*

Menyediakan jalur transmisi sinyal yang terbatas secara fisik, mencakup *twisted-pair Cable*, *coaxial cable*, dan kabel *fiber optic*. *Twisted-pair* dan *coaxial cable* menggunakan konduktor logam yang menerima dan mentransmisikan sinyal dalam bentuk aliran listrik. *Optical fiber*/serat optik menerima dan mentransmisikan sinyal data dalam bentuk cahaya.

2. *Unguided media*

Unguided media atau komunikasi tanpa kabel (nirkabel), mentransmisikan gelombang elektromagnetik tanpa menggunakan konduktor secara fisik. Sinyal dikirimkan secara *broadcast* melalui udara (air, dalam beberapa kasus). Media transmisi ini dapat menggunakan *wireless* (*microwave*, *bluetooth*, *infrared*, radio, dll) atau menggunakan satelit.

Media transmisi *wireless* ada beberapa jenis, yaitu:

1. *Microwave* merupakan *high-end* dari RF (*Radio Frequency*), sekitar 1 – 30 GHz.

Transmisi dengan *microwave* memberikan 4 hal yang perlu diperhatikan :

- Alokasi frekuensi
- *Interference*
- *Line-of sight*
- Jarak tanpa *repeater* antara 10 – 100 km

2. Radio yaitu media transmisi yang menggunakan gelombang elektromagnetik untuk transmisi data. Gelombang radio ini berada pada jangkauan frekuensi 10 hertz (Hz) sampai beberapa gigahertz (GHz).
3. *Infrared* yaitu media transmisi yang menggunakan *transmitter/receiver* yang memodulasikan *no coherent infrared light*. *Infrared* memiliki data rate yang tinggi dan konsumsi dayanya lebih kecil [IRD].
4. *Bluetooth* yaitu teknologi *wireless* yang mampu menyediakan layanan komunikasi data dan suara dengan jarak jangkauan yang terbatas. *Bluetooth* beroperasi pada pita frekuensi 2,4 GHz dengan jarak jangkauan layanan yang terbatas (sekitar 10 meter) [TRI].

Penggunaan *wireless* saat ini sangat berperan penting dalam memenuhi kebutuhan komunikasi. Jaringan komputer dengan menggunakan *wireless* (nirkabel) lebih mudah penggunaannya dibandingkan dengan menggunakan kabel, karena dapat diakses dari mana saja selama berada dalam jangkauan *wireless* tersebut. Teknologi *wireless* banyak diterapkan dalam jaringan komputer atau lebih dikenal dengan *wireless LAN*.

Wireless LAN menggunakan gelombang radio sebagai media transmisinya. *Wireless LAN* didefinisikan oleh IEEE (*Institute of Electrical and Electronics Engineers*) pada IEEE 802.11. Teknologi WLAN mampu memberikan kecepatan akses yang tinggi hingga 11 Mbps (IEEE 802.11 b) dan 54 Mbps (IEEE 802.11 g) dalam jarak hingga 100 meter.

2.1.1 Keuntungan menggunakan WLAN

Perkembangan WLAN yang sangat pesat tidak terlepas dari kelebihanannya dibanding LAN. WLAN memiliki beberapa keuntungan, diantaranya[WL1]:

1. Mobility

Mampu menghemat waktu ketika memerlukan fleksibilitas perpindahan tanpa harus memutuskan koneksi yang sudah ada sebelumnya.

2. Productivity

Jika terjadi perpindahan (*mobile*), dengan WLAN pengguna bisa memanfaatkan waktu yang dibutuhkan selama perpindahan tersebut sehingga pengguna lebih produktif.

3. Expandability

Jika terjadi pertumbuhan jaringan, WLAN tidak membutuhkan banyak perubahan. Jika dibandingkan dengan LAN, maka dibutuhkan penambahan kabel dan *port* bahkan mungkin harus mengganti perangkat seperti *switch* dan *hub*.

4. *Cost*

WLAN mengurangi *cost* (biaya) karena tidak membutuhkan peralatan lain selain *wireless card*, selain itu akan mengurangi biaya pemasangan/penambahan jaringan.

5. *Deployment*

Deployment pada WLAN lebih mudah dibanding dengan LAN. Dengan menggunakan kabel pada suatu jaringan, akan menambahkan biaya dan kerumitan untuk memasang kabel ke berbagai lokasi (yang mungkin sangat sulit untuk mencapai lokasi dalam gedung).

2.1.2 Kekurangan menggunakan WLAN

Selain memiliki kelebihan, WLAN juga mempunyai kekurangan, yaitu [WL1]:

1. *Security*

Karena WLAN dapat diakses dari mana saja selama berada dalam jangkauan WLAN tersebut. Paket yang dikirimkan oleh seseorang bisa saja diterima oleh orang lain. Hal ini memungkinkan seseorang bisa melakukan "*snoping*" dengan tujuan yang tidak baik.

2. *Reliability* (Keandalan)

Kualitas transmisi dan kecepatan WLAN sangat rentan dengan gangguan seperti atenuasi (redaman) dan *noise*. Semakin jauh jaraknya, maka data yang akan sampai semakin sedikit.

3. *Speed*

Kecepatan WLAN dalam hal transmisi data jauh dibawah LAN. Kecepatan WLAN (1-100 Mbps) cukup lambat dibanding kecepatan LAN (100 Mbps hingga beberapa Gbps). Selain itu semakin banyak *host* yang menggunakan layanan WLAN, maka kecepatan transmisinya akan semakin lambat.

2.2 MAC Address

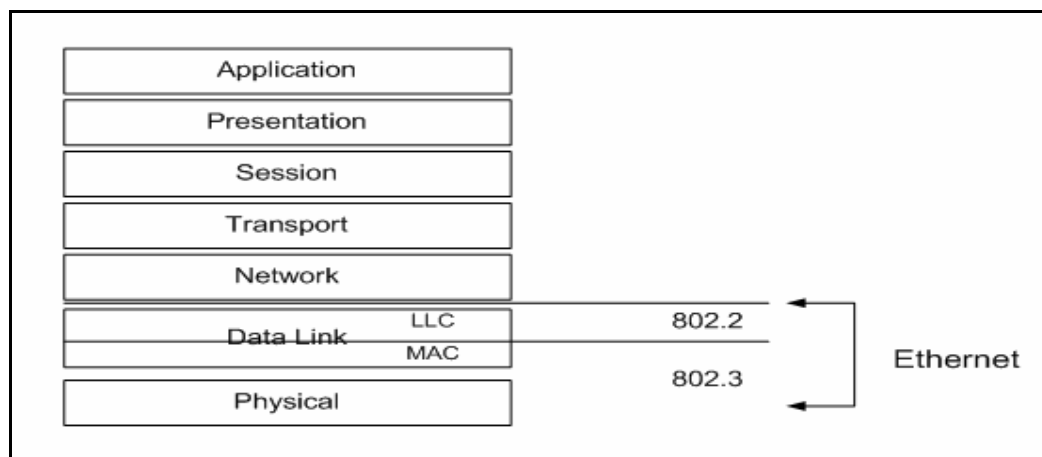
MAC *address* (*Media Access Control Address*) adalah sebuah alamat jaringan yang diimplementasikan pada lapisan *data-link* dalam tujuh lapisan model OSI, yang merepresentasikan sebuah *node* tertentu dalam jaringan. Dalam sebuah jaringan berbasis *Ethernet*, MAC *address* merupakan alamat yang unik yang memiliki panjang 48-bit (6

byte) yang mengidentifikasi sebuah komputer, *interface* dalam sebuah *router*, atau *node* lainnya dalam jaringan. MAC *address* juga sering disebut sebagai *Ethernet address*, *physical address*, atau *hardware address* [MAC].

Ethernet merupakan jenis skenario perkabelan dan pemrosesan sinyal untuk data jaringan komputer yang dikembangkan oleh Robert Metcalfe dan David Boggs di *Xerox Palo Alto Research Center* (PARC) pada tahun 70-an. Standar pertama *Ethernet* dipublikasikan pada tahun 1980 oleh *Digital Equipment Corporation* (DEC), Xerox, dan Intel. Dan pada tahun 1985 distandarisasi oleh *Institute of Electrical and Electronics Engineers* (IEEE) dengan nomor 802 [CC2]. Ada beberapa tipe *ethernet*, yaitu :

- 10 Mbps - *10Base-T Ethernet*
- 100 Mbps - *Fast Ethernet*
- 1000 Mbps - *Gigabit Ethernet*
- 10 Gbps - *10 Gigabit Ethernet*

Perbedaan tersebut yaitu *data rate* tipe *Ethernet* tersebut terjadi di *layer Physical* sehingga sering disebut *Ethernet PHY*). *Ethernet* bekerja pada *Data Link* dan *Physical OSI layer*. *Data Link layer* dibagi menjadi dua sublayer yaitu *Logical Link Layer* (LLC) yang didefinisikan pada IEEE 802.2 dan *Media Access Control* (MAC) yang didefinisikan pada IEEE 802.3.

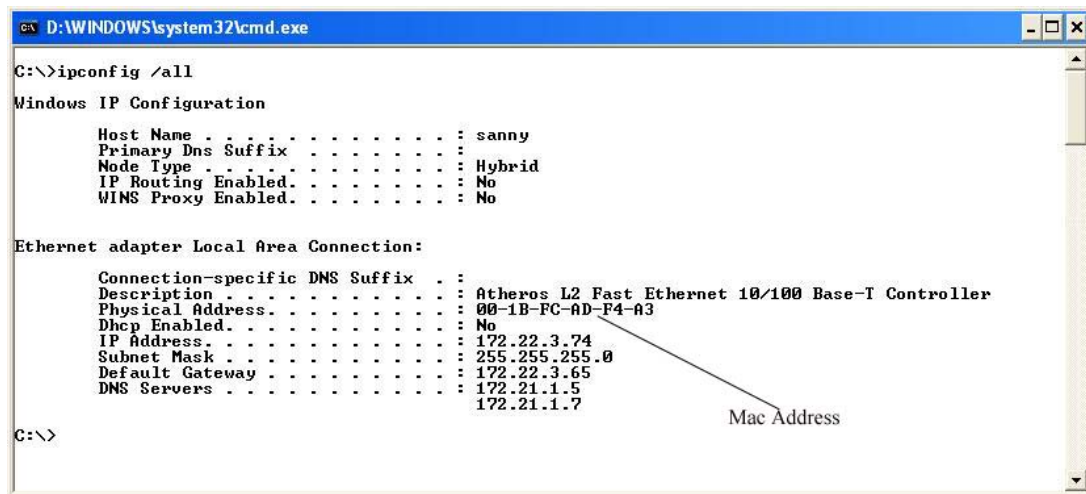


Gambar 1 . Ethernet pada OSI Layer

LLC diimplementasikan pada *software (driver)* dan independen terhadap *physical equipment*. MAC diimplementasikan *hardware*, yaitu *Network Interface Card* (NIC) atau lebih dikenal dengan kartu jaringan [CC3].

Masalah utama yang muncul dalam transmisi adalah bagaimana caranya mengidentifikasi setiap *device*. Untuk mengatasi masalah tersebut, sebuah *idenfier* yang unik yang disebut alamat MAC (*MAC address*) dibuat untuk membantu dalam menentukan alamat *source* (pengirim) dan *destination* (penerima) dalam sebuah jaringan. *MAC address* ditambahkan sebagai bagian dari PDU (*Protocol Data Unit*) *Data link layer*.

Untuk melihat nilai *MAC address* bisa dilakukan dengan cara mengetikkan `ipconfig /all` pada sistem operasi Windows atau `ifconfig` pada sistem operasi Linux.



```
C:\>ipconfig /all

Windows IP Configuration

    Host Name . . . . . : sanny
    Primary Dns Suffix . . . . . : 
    Node Type . . . . . : Hybrid
    IP Routing Enabled. . . . . : No
    WINS Proxy Enabled. . . . . : No

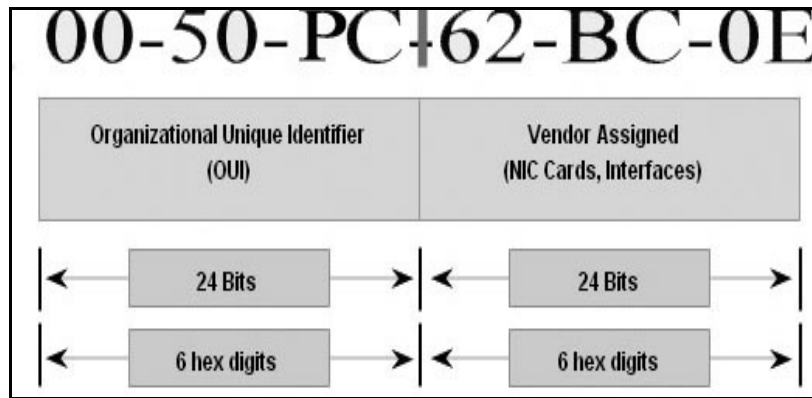
Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : 
    Description . . . . . : Atheros L2 Fast Ethernet 10/100 Base-T Controller
    Physical Address. . . . . : 00-1B-FC-AD-F4-A3
    Dhcp Enabled. . . . . : No
    IP Address. . . . . : 172.22.3.74
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 172.22.3.65
    DNS Servers . . . . . : 172.21.1.5
                           172.21.1.7

C:\>
```

Gambar 2 . MAC Address (*Physical Address*)

Nilai *MAC address* berdasarkan aturan IEEE untuk memastikan alamat global yang unik untuk setiap perangkat *Ethernet*. *MAC address* terdiri dari 48 bit dan dinyatakan dalam bentuk 12 digit hexadesimal, enam digit pertama menyatakan *vendor* pembuat perangkat jaringan tersebut atau menyatakan OUI (*Organizational Unique Identifier*) dan enam digit lainnya menyatakan *serial number interface*. Alamat MAC yang sering disebut sebagai *burned-in address* (BIA) karena di-*burned* ke dalam ROM (*Read-Only Memory*) pada NIC. Ini berarti alamat di-*encode* ke dalam *chip* ROM secara permanen [CC4]. Untuk lebih jelasnya, perhatikan gambar 3.



Gambar 3 . Struktur MAC Address

Ketika komputer hidup (*starts up*), maka NIC menyalin alamat ke dalam RAM. Jika *ethernet* sudah ditentukan menggunakan layanan DHCP sebelumnya (bukan *static IP address*), maka MAC sangat berpengaruh dalam proses permintaan dan pemberian IP *address*. *Host* yang bersangkutan mengirimkan *message* dengan menggunakan MAC *address* dan pengirimannya secara *broadcast*. Ketika memeriksa *frame*, adalah alamat di RAM yang digunakan membandingkan alamat *source* dengan alamat *destination*. Alamat MAC yang digunakan oleh NIC untuk menentukan apakah pesan harus disampaikan ke lapisan atas untuk diproses.

2.3 Protokol AAA

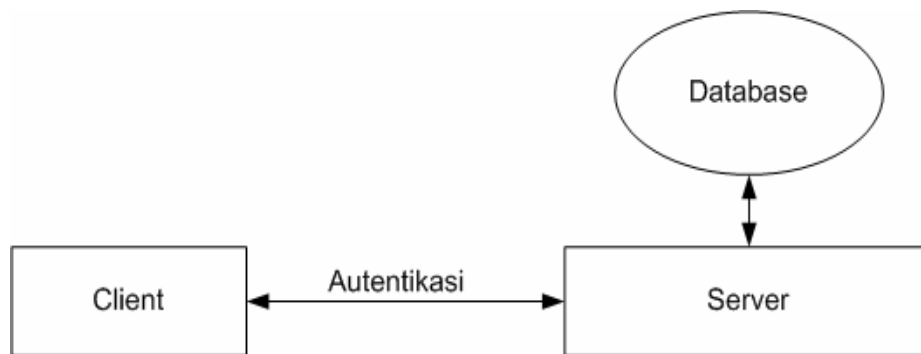
Dalam keamanan sistem (*computer security*), AAA adalah singkatan dari “*authentication, authorization and accounting*”. Protokol AAA kadang-kadang digabungkan dengan *auditing* (memeriksa) sehingga sering disebut sebagai AAAA. Protokol AAA mengatur mekanisme bagaimana tata cara berkomunikasi, baik antara *client* ke domain-domain jaringan maupun antar *client* dengan domain yang berbeda dengan tetap menjaga keamanan pertukaran data.

2.3.1 Authentication

Autentikasi (*Authentication*) yaitu proses pengesahan identitas pengguna (*end user*) untuk mengakses suatu jaringan. Autentikasi menyediakan informasi bagi pengirim dan penerima untuk memvalidasi yang satu dengan yang lainnya sesuai identitasnya. Proses ini diawali dengan pengiriman kode unik (misalnya, *username, password, pin*) oleh pengguna kepada *server*. Di sisi *server*, sistem akan menerima kode unik tersebut, selanjutnya membandingkan dengan kode unik yang disimpan dalam *database server*.

Jika hasilnya sama, maka *server* akan mengirimkan hak akses kepada pengguna. Namun jika hasilnya tidak sama, maka *server* akan mengirimkan pesan kegagalan dan menolak hak akses pengguna. Proses ini merupakan salah satu aspek yang sangat penting dalam sistem keamanan jaringan.

Pengertian sederhana dari autentikasi adalah pengiriman *password* antara entitas yang ingin melakukan autentikasi tersebut.



Gambar 4 . Proses Autentikasi

Pada gambar 4 menunjukkan salah satu proses autentikasi. Pengguna melakukan autentikasi dengan mengirimkan *username* dan *password* ke *server*. Kemudian *server* akan membandingkan *username* dan *password* tersebut ke *database*. Jika hasilnya sama, maka *server* akan mengembalikan informasi autentikasi kepada pengguna, sehingga pengguna dapat mengakses *server*.

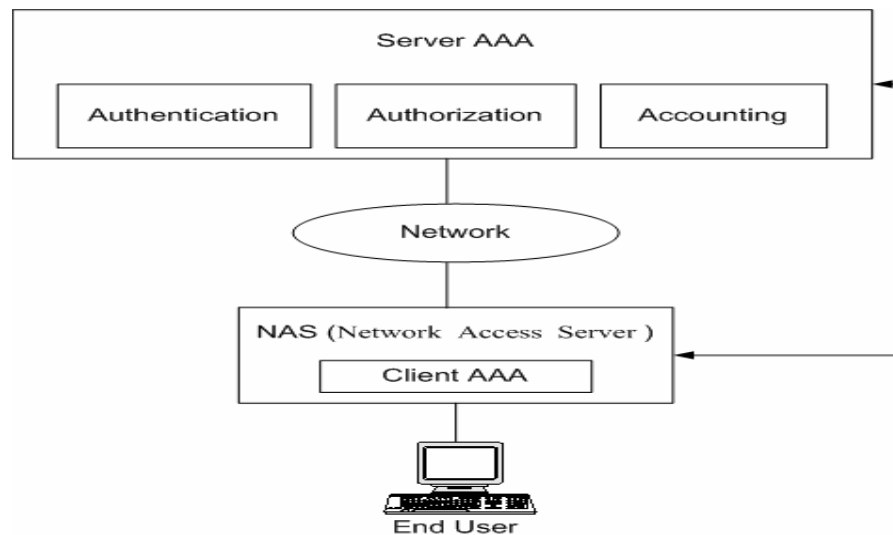
Alasan menggunakan autentikasi yaitu untuk meningkatkan keamanan, memungkinkan *administrator* untuk melacak masalah khusus untuk pengguna, dan memungkinkan pengaturan hak akses pengguna terhadap *server*.

2.3.2 Authorization

Autorisasi (*Authorization*) adalah hak dan izin yang diberikan kepada pengguna atau aplikasi yang memungkinkan untuk mengakses suatu jaringan atau suatu komputer (*computer resource*). Autorisasi merupakan proses pengecekan hak akses pengguna, mana saja hak-hak akses yang diperbolehkan dan mana saja yang tidak. Autorisasi merujuk kepada pemberian hak akses tertentu kepada suatu entitas atau pengguna, berdasarkan proses autentikasi, hak akses apa saja yang dapat mereka minta (*request*) pada sistem yang sedang berjalan. Autorisasi biasanya memiliki batasan-batasan, misalnya batasan waktu, lokasi, atau larangan melakukan proses *login* oleh pengguna..

2.3.3 Accounting

Pencatatan (*Accounting*) merupakan proses pengumpulan data atau informasi mengenai berapa lama pengguna melakukan koneksi terhadap suatu sistem. *Accounting* merujuk kepada pelacakan dari konsumsi sumber daya jaringan oleh pengguna. Informasi ini dapat digunakan untuk pengelolaan (*management*), perencanaan (*planning*), penagihan (*billing*), atau tujuan lainnya. Informasi yang dikumpulkan dalam *accounting* adalah identitas pengguna, sifat dari layanan yang diberikan, saat memulai layanan, dan ketika berakhir.



Gambar 5 . Arsitektur jaringan AAA

Gambar 5 menunjukkan komponen-komponen dalam jaringan AAA. Sistem terdiri dari *Server AAA*, jaringan perantara, *Network Access Server* (NAS) dan pelanggan (*end user*). *Server AAA* dapat dibangun dari beberapa *server*, yang berisi data pelanggan. Jaringan perantara dapat berbentuk jaringan internet, jaringan PSTN (*Public Switched Telephone Network*), dan sebagainya. Dalam jaringan ini juga dapat berisikan *server AAA* lain, yang tersebar sehingga membentuk *server* perantara. *Server* perantara ini berperan membantu menyebarkan informasi AAA ke pelanggan yang lokasinya jauh dari *server*. NAS berperan sebagai pintu gerbang (*gateway*) pelanggan untuk berhubungan dengan jaringan. Perangkat ini dapat berbentuk router, terminal *server* atau sebuah *host*, yang dilengkapi dengan aplikasi *AAA client*. Pelanggan yaitu pengguna sistem itu sendiri, misalnya, perangkat komputer.

Mekanisme kerja jaringan AAA adalah sebagai berikut :

- Pelanggan melakukan koneksi ke peralatan NAS dan sebagai langkah awal koneksi ke jaringan.
- NAS sebagai *client* AAA kemudian melakukan pengumpulan informasi pelanggan dan melanjutkan data pelanggan ke *server* AAA.
- *Server* AAA menerima dan memproses data pelanggan, kemudian memberikan balasan ke NAS berupa pesan penerimaan atau penolakan pendaftaran dari pelanggan. Pesan ini mungkin juga disertai dengan data lain jika diperlukan.
- NAS sebagai *client* AAA kemudian menyampaikan pesan *server* AAA itu kepada pelanggan, bahwa pendaftaran ditolak atau diterima beserta layanan-layanan yang diperkenankan untuk diakses.

2.4 Access Point

Access Point merupakan perangkat keras yang digunakan dalam jaringan tanpa kabel (*wireless*) atau jaringan Wi-Fi (*Wireless-Fidelity*). *Access Point* berfungsi untuk menghubungkan antara jaringan *wireless* dengan jaringan lainnya atau internet. *Access point* juga dapat berfungsi sebagai *filtering*, *firewall*, *router*, yaitu melakukan identifikasi terhadap jaringan dengan memberikan WEP (*Wireless Equivalent Privacy*) atau WPA *wireless*. Perangkat *access point* yang lama bekerja pada standar 802.11b dan kapasitas data yang dapat dikirimkan kecil. Untuk mengirimkan data dalam kapasitas yang besar, hendaknya menggunakan perangkat dengan standar 802.11g dan standar ini kompatibel dengan standar di bawahnya, artinya masih dapat menggunakan perangkat dengan standar 802.11b. Sebuah perangkat *access point* biasanya mempunyai *port ethernet* yang biasanya dihubungkan dengan kabel UTP.

Access Point memiliki beberapa *security mode* yang dapat digunakan. Jenis-jenis *security wireless* tersebut yaitu WEP, WPA, WPA-*Personal*, WPA2-*Personal*, WPA2-*Personal Mixed*, WPA-*Enterprise*, WPA2-*Enterprise*, dan WPA2-*Enterprise Mixed*.

2.4.1 WEP (Wired Equivalent Privacy)

(WEP) *Wired Equivalent Privacy* adalah suatu metoda pengamanan jaringan nirkabel, disebut juga dengan *Shared Key Authentication*. *Shared Key Authentication* adalah metoda autentikasi yang membutuhkan penggunaan WEP. Enkripsi WEP menggunakan kunci yang dimasukkan (oleh *administrator*) ke *client* maupun *access point*. Kunci ini

harus sesuai dari yang diberikan akses *point* ke pengguna, dengan yang dimasukkan pengguna untuk autentikasi menuju *access point*.

Proses *Shared Key Authentication* yaitu sebagai berikut:

1. *Client* meminta asosiasi ke *access point*, langkah ini sama seperti *Open System Authentication*.
2. *Access point* mengirimkan *text challenge* ke *client* secara transparan.
3. *Client* akan memberikan respon dengan mengenkripsi *text challenge* dengan menggunakan kunci WEP dan mengirimkan kembali ke *access point*.
4. *Access point* memberi respon atas tanggapan *client*, akses *point* akan melakukan *decrypt* terhadap respon enkripsi dari *client* untuk melakukan verifikasi bahwa *text challenge* dienkripsi dengan menggunakan WEP *key* yang sesuai. Pada proses ini, *access point* akan menentukan apakah *client* sudah memberikan kunci WEP yang sesuai. Apabila kunci WEP yang diberikan oleh *client* sudah benar, maka *access point* akan merespon positif dan langsung melakukan autentikasi terhadap *client*. Namun bila kunci WEP yang dimasukkan *client* salah, maka *access point* akan merespon negatif dan *client* tidak akan diberi autentikasi.

Selain itu, algoritma *Wired Equivalent Privacy* (WEP) digunakan untuk menjaga privasi komunikasi nirkabel (*wireless*) dari *eavesdropping* (pihak penyadap). Kegunaan kedua dari algoritma WEP adalah mengamankan akses tidak sah (*unauthorized access*) ke sebuah jaringan nirkabel (*wireless network*).

2.4.2 WPA (Wi-Fi Protected Access)

Wi-Fi Protected Access (WPA) adalah protocol yang dikembangkan oleh *Wi-Fi Alliance* sebagai respon dari banyaknya kelemahan yang ditemukan pada sistem sebelumnya, *Wired Equivalent Privacy* (WEP).

Protokol baru ini mengimplementasikan banyak hal baru dari standar IEEE 802.11i dan telah disiapkan untuk menjadi pengganti WEP ketika 802.11i masih dalam tahap persiapan. Hal paling signifikan dari pengembangan WPA adalah penambahan *Temporal Key Integrity Protocol* (TKIP).

2.4.3 WPA-Personal

WPA-Personal adalah salah satu jenis *wireless security* yang menggunakan algoritma TKIP dan AES pada enkripsi datanya. *User* dapat memilih salah satu algoritma yang

diinginkan dari kedua jenis algoritma tersebut. Secara *default* WPA-Personal ini menggunakan algoritma TKIP. WPA-Personal ini menggunakan 8 sampai 63 karakter untuk *shared key*-nya. *Mode security wireless* ini juga membutuhkan waktu *timeout* untuk setiap pergantian *key* pada proses enkripsi yang dilakukan. Secara *default* waktu yang dibutuhkan untuk pergantian *key* enkripsi adalah 3600 detik.

2.4.4 WPA2-Personal

WPA2-Personal adalah salah satu jenis *wireless security* yang menggunakan algoritma AES untuk enkripsi datanya. Selain itu, WPA2-Personal menggunakan *Pre-shared Key* sepanjang 8 sampai 63 karakter. *Mode security wireless* ini juga membutuhkan waktu *timeout* untuk setiap pergantian *key* pada proses enkripsi yang dilakukan. Secara *default* waktu yang dibutuhkan untuk pergantian *key* enkripsi adalah 3600 detik.

2.4.5 WPA2-Personal Mixed

WPA-Personal merupakan gabungan dari WPA-Personal dan WPA2-Personal. Untuk mode gabungan secara otomatis memilih TKIP dan AES sebagai algoritma enkripsi data. Selain itu, WPA2-Personal menggunakan *Pre-shared Key* sepanjang 8 sampai 63 karakter. Mode keamanan *wireless* ini juga membutuhkan waktu *timeout* untuk setiap pergantian *key* pada proses enkripsi yang dilakukan. Secara *default* waktu yang dibutuhkan untuk pergantian *key* enkripsi adalah 3600 detik.

2.4.6 WPA-Enterprise

Fitur *wireless security* yang ini digunakan pada saat berkordinasi dengan *server* RADIUS untuk autentikasi *client*. WPA ini hanya dapat digunakan ketika *server* RADIUS terhubung dengan *access point*. WPA ini menggunakan dua metode enkripsi yaitu TKIP dan AES. Secara *default* algoritma yang digunakan adalah TKIP. *Mode security wireless* ini juga membutuhkan waktu *timeout* untuk setiap pergantian *key* pada proses enkripsi yang dilakukan. Secara *default* waktu yang dibutuhkan untuk pergantian *key* enkripsi adalah 3600 detik. Selain itu, WPA-Enterprise memiliki tiga jenis utama proses *backup* data yaitu:

1. *Primary/Backup RADIUS Server*

Backup data ini membutuhkan *IP address* dari *server* RADIUS. *Backup server* RADIUS hanya dapat digunakan jika *server* utama RADIUS tidak tersedia.

2. *Primary/Backup RADIUS Server Port*

Backup data ini membutuhkan nomor *port* yang digunakan oleh *server* RADIUS. Secara *default port* yang digunakan adalah *port* 1812 *Backup server* RADIUS hanya dapat digunakan jika *server* utama RADIUS tidak tersedia.

3. *Primary/Backup Shared Secret*

Untuk *backup* ini membutuhkan *shared secret key* yang digunakan oleh *access point* dan *server* RADIUS. Hal ini dilakukan untuk *backup* data pada *server* RADIUS jika *server* RADIUS yang utama tidak tersedia.

2.4.7 WPA2-Enterprise

WPA2-Enterprise ini digunakan pada saat berkoordinasi dengan sebuah *server* RADIUS untuk autentikasi pengguna. WPA2-Enterprise ini hanya dapat digunakan ketika *server* RADIUS terhubung dengan *access point*. WPA2-Enterprise ini selalu menggunakan algoritma AES untuk enkripsi datanya. WPA2-Enterprise memiliki waktu *timeout* untuk melakukan pergantian terhadap *key*. Secara *default* waktu *timeout*nya adalah 3600 detik. Ada tiga jenis utama proses *backup* data pada WPA2-Enterprise yaitu:

1. *Primary/Backup RADIUS Server*

Backup data ini membutuhkan IP *address* dari *server* RADIUS. *Backup server* RADIUS hanya dapat digunakan jika *server* utama RADIUS tidak tersedia.

2. *Primary/Backup RADIUS Server Port*

Backup data ini membutuhkan nomor *port* yang digunakan oleh *server* RADIUS. Secara *default port* yang digunakan adalah *port* 1812 *Backup server* RADIUS hanya dapat digunakan jika *server* utama RADIUS tidak tersedia.

3. *Primary/Backup Shared Secret*

Untuk *backup* ini membutuhkan *shared secret key* yang digunakan oleh *access point* dan *server* RADIUS. Hal ini dilakukan untuk *backup* data pada *server* RADIUS jika *server* RADIUS yang utama tidak tersedia.

2.4.8 WPA2-Enterprise Mixed

Security mode ini merupakan hasil transisi dari WPA-Enterprise dan WPA2-Enterprise. Secara otomatis *access point* akan memilih algoritma enkripsi yang digunakan oleh setiap pengguna. Ada tiga jenis utama proses *backup* data pada WPA2-Enterprise Mixed yaitu:

1. *Primary/Backup RADIUS Server*

Backup data ini membutuhkan IP *address* dari *server* RADIUS. *Backup server* RADIUS hanya dapat digunakan jika *server* utama RADIUS tidak tersedia.

2. *Primary/Backup RADIUS Server Port*

Backup data ini membutuhkan nomor *port* yang digunakan oleh *server* RADIUS. Secara *default port* yang digunakan adalah *port* 1812 *Backup server* RADIUS hanya dapat digunakan jika *server* utama RADIUS tidak tersedia.

3. *Primary/Backup Shared Secret*

Untuk *backup* ini membutuhkan *shared secret key* yang digunakan oleh *access point* dan *server* RADIUS. Hal ini dilakukan untuk *backup* data pada *server* RADIUS jika *server* RADIUS yang utama tidak tersedia.

2.5 Remote Authentication Dial-In User Service (RADIUS)

Remote Authentication Dial-In User Service (RADIUS) merupakan suatu protokol yang dikembangkan untuk menyediakan akses terpusat untuk melakukan *authentication*, *authorization*, dan *accounting* [AG1]. RADIUS menggunakan AAA konsep untuk mengelola akses jaringan, dikenal dengan istilah "AAA Transaction". RADIUS pada awalnya dikembangkan oleh Livingston untuk jaringan terminal *server*. RADIUS didefinisikan di dalam RFC 2865, yang pada awalnya digunakan untuk melakukan autentikasi terhadap akses jaringan secara jarak jauh dengan menggunakan koneksi *dial-up*. Karena dukungan yang luas dan sifatnya terbuka, protokol RADIUS sering digunakan oleh ISP, jaringan nirkabel (*wireless*), layanan email, *access point* atau *web server* untuk melakukan autentikasi terhadap akses jaringan.

RADIUS pada umumnya banyak digunakan oleh penyedia layanan internet (*Internet Service Provider*) untuk mengelola akses ke internet atau jaringan internal. RADIUS juga bisa dipakai untuk mengamankan jaringan yang ada.

Beberapa fitur yang disediakan oleh RADIUS adalah sebagai berikut :

1. *Client/Server Model*

Network Access Server (NAS) beroperasi sebagai pengguna RADIUS. NAS bertanggung jawab untuk meneruskan informasi pengguna ke RADIUS *server*, kemudian mengembalikan respon ke pengguna. RADIUS *server* bertanggung jawab untuk menerima permintaan koneksi pengguna, autentikasi, dan mengembalikan semua informasi yang diperlukan kepada pengguna. RADIUS *server* dapat bertindak sebagai *proxy client* bagi RADIUS *server* lainnya atau autentikasi lainnya.

2. Network Security

Transaksi antara pengguna dan server RADIUS adalah dengan melakukan proses autentikasi. Selain itu, setiap *username* dan *password* pengguna dienkripsi diantara pengguna dan server RADIUS, untuk menghapuskan kemungkinan bahwa seseorang melakukan *snooping* pada jaringan sehingga dapat menemukan *username* dan *password*.

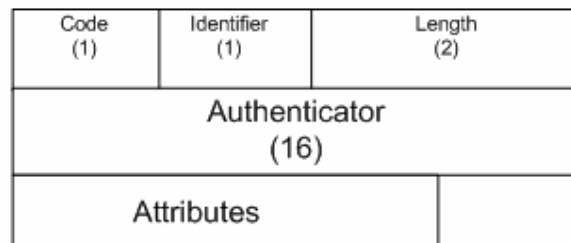
3. Flexible Authentication Mechanisms

RADIUS *server* dapat mendukung berbagai metode untuk melakukan proses autentikasi pengguna.

4. Extensible Protocol

Mendukung autentikasi *Password Authentication Protocol* (PAP) dan *Challenge handshake Authentication Protocol* (CHAP). Atribut nilai-nilai baru dapat ditambahkan tanpa mengganggu yang telah diimplementasi dalam suatu protokol.

Struktur format data pada RADIUS dapat dilihat pada gambar 6. Setiap kolom paket ditransmisikan dari kiri ke kanan.



Gambar 6 . Struktur format data RADIUS

Struktur format data RADIUS pada Gambar 6 terdiri dari lima bagian, yaitu:

1. Code

Code memiliki panjang satu oktet, digunakan untuk membedakan tipe pesan RADIUS yang dikirimkan pada paket. Kode-kode tersebut (dalam desimal) adalah:

Tabel 3 . Kode Paket RADIUS dalam desimal

Kode dalam desimal	Keterangan Kode
1	Access-Request
2	Access-Accept
3	Access-Reject
4	Accounting-Request

Kode dalam desimal	Keterangan Kode
5	Accounting-Response
11	Access-Challenge
12	Status-Server (experimental)
13	Status-Client (experimental)
255	Reserved

2. Identifier

Memiliki panjang satu oktet, bertujuan untuk mencocokkan permintaan (*request*) dan balasan (*replies*).

3. Length

Memiliki panjang dua oktet, memberikan informasi mengenai panjang paket mencakup *Code*, *Identifier*, *Length*, *Authenticator* dan *Attribute fields*.

4. Authenticator

Memiliki panjang 16 oktet, digunakan untuk membuktikan balasan dari RADIUS *server*, selain itu digunakan juga untuk algoritma *password*.

5. Attributes

Field atribut panjangnya bervariasi, dan berisi *list* atribut yang dibutuhkan.

Protokol RADIUS yang digunakan sebagai salah satu sistem keamanan *wireless* LAN melalui autentikasi pengguna *wireless* LAN, ternyata memiliki beberapa celah keamanan. Mekanisme proteksi menggunakan *username* dan *password* ternyata tidak cukup aman untuk diterapkan. Hal ini disebabkan oleh penerapan teknik enkripsi dan kriptografi yang tidak benar.

2.6 FreeRADIUS

Server FreeRADIUS adalah aplikasi unix yang memungkinkan seseorang untuk mengatur protokol RADIUS, yang dapat digunakan untuk autentikasi, otorisasi dan akuntansi berbagai jenis akses jaringan. FreeRADIUS dikembangkan oleh **Alan DeKok** dan **Miquel van Smoorenburg** pada bulan Agustus 1999. Beberapa aplikasi RADIUS *server* ada yang berbentuk komersial dan *free/open source*. Untuk versi komersial contohnya Interlink's Secure.XS, VOP Radius *Small Business*, Funk Odyssey *Server*, dan sebagainya. Sedangkan untuk versi *free/open source* contohnya adalah FreeRADIUS,

Cistron Radius Server, ICRADIUS, XtRADIUS, Open RADIUS, dan sebagainya. Biasanya harga versi komersialnya sangat mahal. Harga RADIUS server komersial tersebut kebanyakan tidak terjangkau para pemilik *hotspot*, terutama bagi kalangan akademis, sehingga sebagian besar memilih menggunakan RADIUS server yang bersifat *free/open source*.

Dari sekian banyak *software* RADIUS server, FreeRADIUS paling banyak digunakan. Beberapa alasan menggunakan FreeRADIUS server adalah sebagai berikut :

- *Free open source* RADIUS server
- Terintegrasi dengan LDAP (*LightWeight Directory Access Protocol*)
- Mendukung berbagai jenis *database* (MySQL, SQL Server, Oracle, dll) bahkan bisa dengan menggunakan teks biasa.
- Dapat berjalan pada berbagai sistem operasi. Pada umumnya FreeRADIUS berjalan pada UNIX dan turunannya (Linux, FreeBSD, NetBSD, Solaris, dll).
- Mendukung berbagai tipe *Access Point* (AP)/ *Network Access Server* (NAS).
- Meng-*cache* semua *file* konfigurasi di memori.
- Menyimpan log pada UNIX *wtm* *file format* dan RADIUS *detail logfiles*
- *Support Simultaneous-Use* = X parameter. Fitur ini mencegah *double login*.
- *Support atribut Vendor-Specific*.
- *Support proxying*
- Mendukung PAM (*Pluggable Authentication Modules*)

2.7 Mikrotik Router OS

Pada prinsipnya router adalah sebuah komputer. Router bekerja pada *layer Network* OSI. Komponen router adalah CPU, RAM, ROM, Flash dan sistem operasi. Router berfungsi :

- Menentukan jalur terbaik (*best path*) untuk mengirimkan paket.
- Mem-*forward* paket ke tujuan (*destination*).

Router bekerja berdasarkan *routing table*-nya. Router adalah *network center* yang menghubungkan *multiple network*. Hal ini berarti router memiliki minimal dua buah *interface* (kartu jaringan), yaitu satu sebagai *input* dan yang lainnya sebagai *output*.

Router memiliki sistem operasi yang berfungsi *Extended Machine - User/Computer Interface* dan *Resources Manager*. Contoh sistem operasi router yang terkenal adalah Cisco IOS, Vyatta, dan Mikrotik. Contoh router yang terkenal adalah Cisco 2621XM.

Namun harga sebuah router mahal. Alternatif lain yang lebih murah adalah menjadikan sebuah PC (*Personal Computer*) menjadi router.

Mikrotik Router OS adalah sistem operasi independen (tidak terbatas pada *vendor hardware* spesifik) berbasis Linux khusus untuk komputer yang difungsikan sebagai Router. Didesain untuk memberikan kemudahan bagi penggunaanya. Administrasinya bisa dilakukan melalui *Windows application* (WinBox) atau melalui *web based*. MikroTik Router OS yang berbentuk *software* dan dapat diinstal pada komputer biasa (PC).

Mikrotik Router OS memiliki beberapa jenis, antara lain yaitu :

- Mikrotik RouterOS yang berbentuk *software* yang apat diinstal pada komputer biasa (*Personal Computer*).
- *Built-in Hardware* MikroTik dalam bentuk perangkat keras yang khusus dikemas dalam *board* router yang didalamnya sudah terinstal MikroTik RouterOS yang disebut DOM (*Disk On Module*).

Tabel 4 . Fitur-fitur Mikrotik

No	Fitur	Keterangan
1.	<i>Address List</i>	Pengelompokan IP <i>Address</i> berdasarkan nama
2.	<i>Asynchronous</i>	Mendukung serial PPP <i>dial-in / dial-out</i> , dengan otentikasi CHAP, PAP, MSCHAPv1 dan MSCHAPv2, RADIUS, <i>dial on demand</i> , modem <i>pool</i> hingga 128 <i>ports</i> .
3.	<i>Bonding</i>	Mendukung dalam pengkombinasian beberapa antarmuka ethernet ke dalam 1 pipa pada koneksi cepat.
4.	<i>Bridge</i>	Mendukung fungsi <i>bridge spanning tree</i> , <i>multiple bridge interface</i> , <i>bridging firewalling</i> .
5.	<i>Data Rate Management</i>	QoS berbasis HTB dengan penggunaan <i>burst</i> , PCQ, RED, SFQ, FIFO <i>queue</i> , CIR, MIR, limit antar <i>peer to peer</i>
6.	DHCP	Mendukung DHCP tiap antarmuka; DHCP <i>Relay</i> ; DHCP <i>Client</i> , <i>multiple network</i> DHCP; <i>static</i> dan <i>dynamic</i> DHCP <i>leases</i> .
7.	<i>Firewall</i> dan NAT	Mendukung pemfilteran koneksi <i>peer to peer</i> , source NAT dan <i>destination</i> NAT.
8.	<i>Hotspot</i>	<i>Hotspot gateway</i> dengan otentikasi RADIUS. Mendukung <i>limit data rate</i> , SSL ,HTTPS.
9.	IPSec	Protokol AH dan ESP untuk IPSec; MODP Diffie-Hellmann groups 1, 2, 5; MD5 dan algoritma SHA1 <i>hashing</i> ; algoritma enkripsi menggunakan DES, 3DES, AES-128, AES-192, AES-256; <i>Perfect Forwarding Secresy</i> (PFS) MODP <i>groups</i> 1, 2,5

No	Fitur	Keterangan
10.	ISDN	Mendukung ISDN <i>dial-in/dial-out</i> . Dengan otentikasi PAP, CHAP, MSCHAPv1 dan MSCHAPv2, RADIUS. Mendukung 128K <i>bundle</i> , Cisco HDLC, x751, x75ui, x75bui <i>line</i> protokol.
11.	RADIUS	Mendukung protokol RADIUS.
12.	MNDP	MikroTik <i>Discovery Neighbour</i> Protokol, juga mendukung Cisco <i>Discovery</i> Protokol (CDP)
13.	<i>Monitoring/Accounting</i>	Laporan <i>Traffic</i> IP, log, statistik grafik yang dapat diakses melalui HTTP
14.	NTP	<i>Network Time Protocol</i> untuk <i>server</i> dan <i>clients</i> ; sinkronisasi menggunakan sistem GPS
15.	<i>Poin to Point Tunneling Protocol</i>	Support PPTP, PPPoE dan L2TP <i>Access Concentrator</i> ; protokol otentikasi menggunakan PAP, CHAP, MSCHAPv1, MSCHAPv2; otentikasi dan laporan RADIUS; enkripsi MPPE; kompresi untuk PPPoE; <i>limit data rate</i> .
16.	<i>Proxy</i>	<i>Cache</i> untuk FTP dan HTTP <i>proxy server</i> , HTTPS <i>proxy</i> ; <i>transparent proxy</i> untuk DNS dan HTTP; mendukung protokol SOCKS; mendukung <i>parent proxy</i> ; <i>static</i> DNS
17.	<i>Routing protocol</i>	<i>Support routing</i> statik dan dinamik; RIP v1/v2, OSPF v2, BGP v4
18.	SDSL	Mendukung Single Line DSL; mode pemutusan jalur koneksi dan jaringan
19.	<i>Simple Tunnel</i>	Tunnel IPIP dan EoIP (Ethernet over IP)
20.	SNMP	Simple Network Monitoring Protocol mode akses read-only
21.	<i>Synchronous</i>	Support sinkronisasi V.35, V.24, E1/T1, X21, DS3 (T3) media ttypes; sync-PPP, Cisco HDLC; Frame Relay line protokol; ANSI-617d (ANDI atau annex D) dan Q933a (CCITT atau annex A); Frame Relay jenis LMI
22.	<i>Utilities/tool</i>	Ping, Traceroute; bandwidth test; ping flood; telnet; SSH; packet sniffer; Dinamik DNS update
23.	VLAN	Mendukung <i>Virtual</i> LAN IEEE 802.1q untuk jaringan <i>ethernet</i> dan <i>wireless</i> ; <i>multiple</i> VLAN; VLAN <i>bridging</i>
24.	VoIP	Mendukung aplikasi <i>Voice over</i> IP
25.	VRRP	Mendukung <i>Virtual Router Redundant Protocol</i>
26.	WinBox	Aplikasi <i>mode</i> GUI untuk meremote dan mengkonfigurasi MikroTik RouterOS.

Bab III

Analisis dan Perancangan

Pada bab Analisis dan Desain akan dijelaskan analisis yang dilakukan selama pelaksanaan kajian ini yang mencakup analisis mekanisme autentikasi *server* RADIUS serta perancangan topologi jaringan yang akan diimplementasikan pada studi implementasi *wireless* LAN.

3.1 Analisis

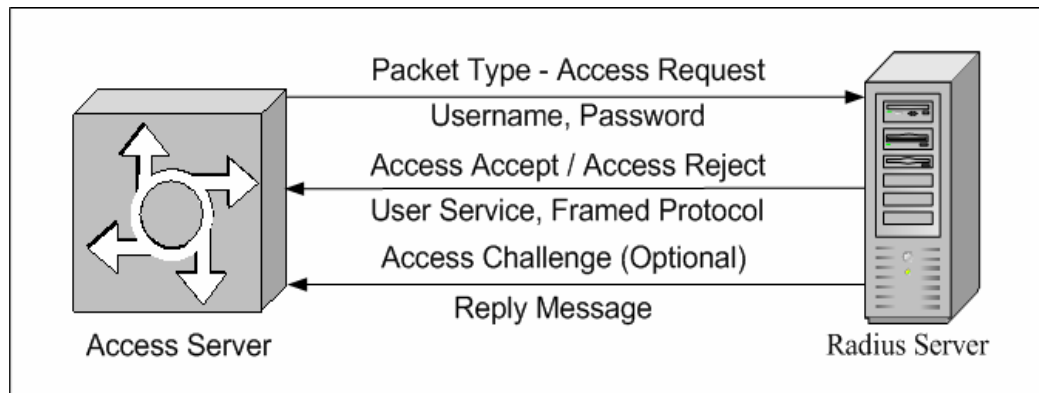
Pada sub bab ini, akan dijelaskan mengenai mekanisme autentikasi pada FreeRADIUS yang mencakup autentikasi MAC address pada FreeRADIUS dengan menggunakan Mikrotik sebagai *access server* serta autentikasi MAC address pada FreeRADIUS dengan menggunakan *access point* sebagai *access server*.

3.1.1 Mekanisme Autentikasi Server RADIUS

Pada *server* RADIUS, *login* pengguna terdiri dari permintaan (*Access-Request*) dari NAS ke *server* RADIUS dan memberikan respon yang sesuai (*Access-Accept* atau *Access-Reject*) dari *server*. Paket "*Access-Request*" berisi *username*, *password* terenkripsi, alamat IP NAS, dan *port*.

Ketika *server* RADIUS menerima *Access-Request* dari NAS, *server* RADIUS mencari *database* untuk pengguna yang di-list (terdaftar). Jika *account* tidak ada dalam *database*, *default* profil di-load atau *server* RADIUS segera mengirimkan sebuah pesan "*Access-Reject*". Pesan "*Access-Reject*" tersebut dapat disertai pesan teks yang menunjukkan alasan penolakan.

Pada RADIUS, autentikasi dan otorisasi digabungkan secara bersamaan. Jika *account* ditemukan dan *password* benar, maka *server* RADIUS mengembalikan respon "*Access-Accept*", termasuk daftar atribut yang menjelaskan parameter yang akan digunakan untuk sesi ini. Parameter termasuk jenis layanan (*frame*), alamat IP pengguna yang diberikan (statis atau dinamis), ACL (*Access Control List*) yang berlaku, atau *static route* untuk menginstal *tabel routing* NAS. Informasi konfigurasi dalam RADIUS *server* mendefinisikan apa yang akan diinstal pada NAS. Gambar 7 menjelaskan urutan autentikasi dan otorisasi.

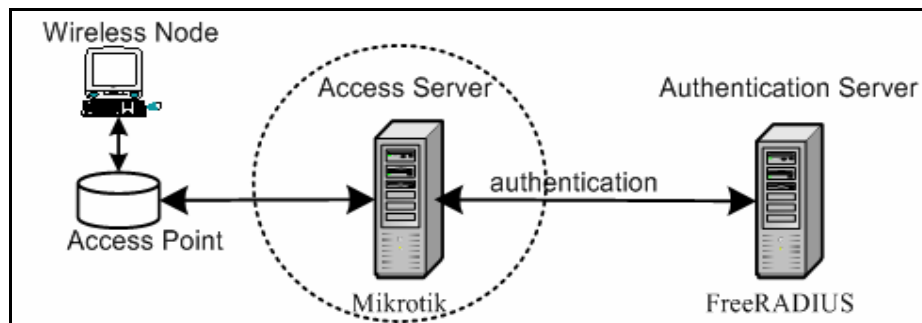


Gambar 7 . Mekanisme Autentikasi RADIUS Server

Pada kajian ini, autentikasi MAC *address* dapat dilakukan dengan beberapa cara, yaitu dengan menggunakan Mikrotik sebagai *access server* atau *access point* sebagai *access server*. Penjelasan untuk tiap-tiap cara yang digunakan yaitu pada sub bab berikut ini.

3.1.1.1 Autentikasi MAC address pada FreeRADIUS dengan Mikrotik

Autentikasi MAC *address* pada FreeRADIUS dengan menggunakan Mikrotik sebagai *access Server* adalah seperti pada gambar 8.

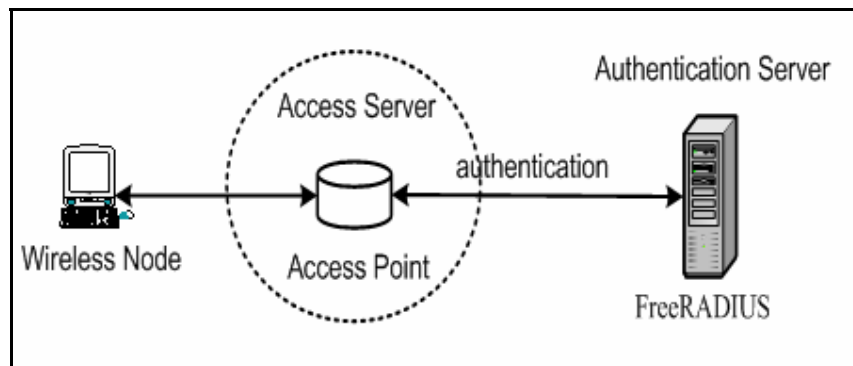


Gambar 8 . Autentikasi dan Autorisasi RADIUS

Pada gambar 8, Mikrotik sebagai *access server* akan terhubung langsung dengan FreeRADIUS. Informasi atau atribut-atribut yang diterima dari *access point* akan dilanjutkan (*forward*) oleh Mikrotik kepada *server* FreeRADIUS. Pada bagian ini, Mikrotik akan diatur sebagai *access server* sehingga akan secara langsung yang melakukan *request* terhadap *server* FreeRADIUS.

3.1.1.2 Autentikasi MAC address pada FreeRADIUS dengan Access Point

Autentikasi MAC *address* dapat juga dilakukan dengan menggunakan *Access point* akan digunakan sebagai *access server*, seperti pada gambar 9.



Gambar 9 . Operasi *Filtering Mac Address*

Pada gambar 9, *Access point* sebagai *access server* akan terhubung langsung dengan FreeRADIUS, sehingga *access point* yang akan mengirimkan informasi atau atribut-atribut kepada server FreeRADIUS secara langsung. Dalam hal ini, *access point* diasumsikan memiliki fitur yang mendukung protokol RADIUS, sehingga *access point* dapat melakukan autentikasi secara langsung dengan server FreeRADIUS.

3.1.1.3 Filtering MAC Address

Pada gambar 7, hal pertama yang akan dilakukan *access server* adalah melakukan *Access Request*. Ketika melakukan *access request*, *access server* akan mengirimkan beberapa atribut untuk melakukan komunikasi dengan FreeRADIUS. Atribut-atribut yang dikirimkan oleh *access point* untuk melakukan komunikasi ke FreeRADIUS adalah sebagai berikut:

```
Packet-Type = Access-Request
Thu Apr 16 14:12:52 2009
  User-Name = "sanhenra"
  NAS-IP-Address = 10.1.0.6
  NAS-Port = 0
  Called-Station-Id = "00-1E-E5-9D-64-B1:TK_03"
  Calling-Station-Id = "00-1B-77-48-0A-2B"
  Framed-MTU = 1400
  NAS-Port-Type = Wireless-802.11
  Connect-Info = "CONNECT 11Mbps 802.11b"
  Client-IP-Address = 10.1.0.6
```

Dari informasi atribut-atribut tersebut, terdapat beberapa informasi yang akan dikirimkan oleh *access point* kepada FreeRADIUS untuk melakukan komunikasi. Pada kajian ini, informasi yang penting adalah *Calling-Station-ID*. Atribut tersebut merupakan alamat MAC dari *wireless node* (pengguna).

Calling-Station-Id

Calling-Station-Id merupakan salah satu atribut yang akan dikirimkan oleh *access server* yaitu yang berisi informasi MAC *address wireless node* (pengguna) yang mengakses suatu *access point*. Atribut ini akan digunakan oleh FreeRADIUS sebagai informasi yang unik dari *client*.

Dalam kasus ini, atribut ini (*Calling-Station-Id*) akan di-*filter* atau diambil nilainya, dan yang akan digunakan sebagai *identity* untuk melakukan autentikasi.

3.2 Perancangan

Tahap perancangan bertujuan untuk mengetahui beberapa kebutuhan yang diperlukan ketika akan melakukan studi implementasi. Perancangan yang dilakukan dalam pelaksanaan kajian ini adalah perancangan perangkat yang akan digunakan sehingga masalah mengenai perangkat yang digunakan saat melakukan studi implementasi dapat diatasi, kemudian perancangan topologi jaringan *wireless LAN* dengan menggunakan Mikrotik dan FreeRADIUS.

3.2.1 Perancangan perangkat yang akan digunakan

Dalam perancangan *wireless LAN* dengan menggunakan Mikrotik dan FreeRADIUS sebagai *authenticator*, memerlukan beberapa perangkat yang dibutuhkan untuk melakukan implementasi. Perancangan perangkat-perangkat yang dibutuhkan adalah sebagai berikut:

- Komputer (*Personal Computer*) untuk instalasi Mikrotik yang akan dijadikan sebagai router OS. Dalam hal ini, PC harus memiliki minimal satu buah NIC *card* yang akan digunakan sebagai *interface* pada Mikrotik.
- Komputer (*Personal Computer*) yang akan diinstal sistem operasi Linux dalam hal ini adalah Fedora Core 6 sebagai sistem operasi untuk melakukan instalasi FreeRADIUS yang digunakan untuk autentikasi.
- Switch yang digunakan untuk menghubungkan semua perangkat.

- Kabel *straight thought* yang akan menghubungkan *server* Mikrotik dengan switch, *server* FreeRADIUS dengan switch, dan *server* Mikrotik dengan perangkat *wireless* (*access point*) .
- Perangkat *wireless* yang akan digunakan sebagai *access point*.

Tahap perancangan ini dilakukan untuk mempermudah dalam pelaksanaan tahapan studi implementasi Tugas Akhir ini.

Perencanaan perangkat yang akan digunakan ada beberapa hal yang perlu diketahui mengenai spesifikasi hardware yang akan digunakan untuk instalasi Mikrotik Router OS, yaitu:

1. Prosesor Intel Pentium 3, dan akan lebih baik jika menggunakan spesifikasi yang lebih, agar performansinya lebih baik.
2. RAM - minimal 128 MB, dan dianjurkan untuk menggunakan 64 MB atau lebih.
3. *Harddisk* - Menggunakan standar ATA/IDE *drive* dengan kapasitas minimal 64 MB atau lebih.

Perencanaan perangkat yang akan digunakan untuk instalasi FreeRADIUS yang dalam hal ini dilakukan pada sistem operasi linux Fedora Core 6 adalah sebagai berikut:

1. Prosesor Intel Pentium 4, dan akan lebih baik jika menggunakan spesifikasi yang lebih tinggi, agar performansinya lebih baik.
2. RAM - minimal 256 MB.
3. *Harddisk* - Menggunakan standar ATA/IDE atau Serial ATA *drive* dengan kapasitas minimal 6GB.

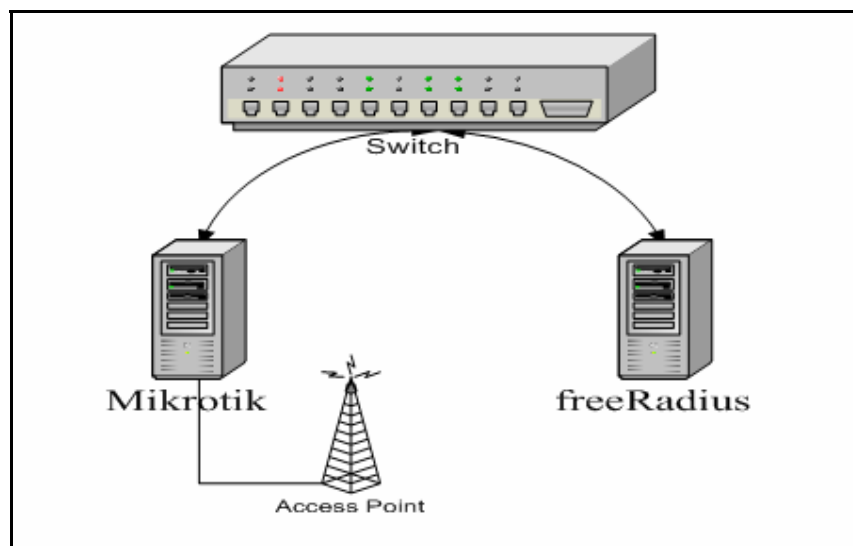
Pada studi ini, perancangan terhadap perangkat pendukung seperti switch yaitu dengan menggunakan switch standar yang biasa digunakan pada suatu jaringan lokal. *Access point* yang akan digunakan adalah Linksys WAP4400N. Perangkat tersebut sudah mendukung *wireless security* yaitu *security* WPA2-*enterprise* yang dapat diintegrasikan langsung dengan *server* RADIUS.

3.2.2 Perancangan topologi jaringan

Tahapan ini bertujuan untuk merencanakan topologi jaringan ketika akan melakukan studi implementasi pada *wireless* LAN yang memenuhi kebutuhan pengguna saat ini dan dapat

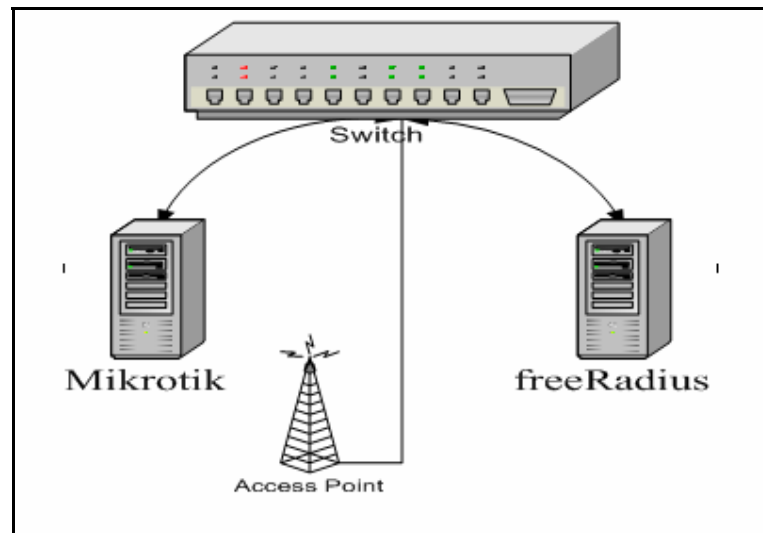
dikembangkan di masa yang akan datang sejalan dengan peningkatan kebutuhan jaringan yang lebih besar. Perancangan mencakup perencanaan secara fisik mencakup media yang digunakan dan infrastruktur *wireless* LAN yakni hubungan antara suatu komponen dengan komponen lain.

Bentuk topologi dasar jaringan untuk implementasi *wireless* LAN pada studi kasus ini terdapat beberapa jenis. Topologi jaringan yang mungkin ketika akan melakukan studi implementasi pada Tugas Akhir ini antara lain adalah seperti gambar 10.



Gambar 10 . Topologi dasar jaringan 1

Pada gambar 10, *server* mikrotik harus memiliki 2 buah *interface* dimana salah satu *interface*-nya akan terhubung dengan *switch*, dan satu *interface* akan terhubung dengan *access point*. Pada topologi ini, *access point* akan memiliki *network address* yang berbeda dengan *network address* FreeRADIUS dan *interface* Mikrotik yang terhubung dengan *switch*. Perancangan topologi pada gambar 10, akan digunakan pada pelaksanaan studi ini. Hal ini dilakukan karena dukungan dari perangkat yang digunakan ketika akan melakukan implementasi.



Gambar 11 . Topologi dasar jaringan 2

Pada gambar 11, topologi ini digunakan jika *server* Mikrotik hanya memiliki satu *interface* saja. *Access point* akan dihubungkan langsung dengan *switch*, sehingga semua komponen yang terhubung akan memiliki *network address* yang sama.

Bab IV Pelaksanaan

Pada bab Pelaksanaan akan dijelaskan cara instalasi *tools* yang akan digunakan seperti Mikrotik Router OS dan FreeRADIUS dan konfigurasi *tools* tersebut, konfigurasi perangkat *wireless* yang digunakan sebagai *access point*, serta integrasi antara Mikrotik Router OS dan FreeRADIUS.

Pada kajian ini, implementasi yang dilakukan adalah autentikasi berdasarkan MAC *address* pada FreeRADIUS dengan menggunakan Mikrotik sebagai *access server*.

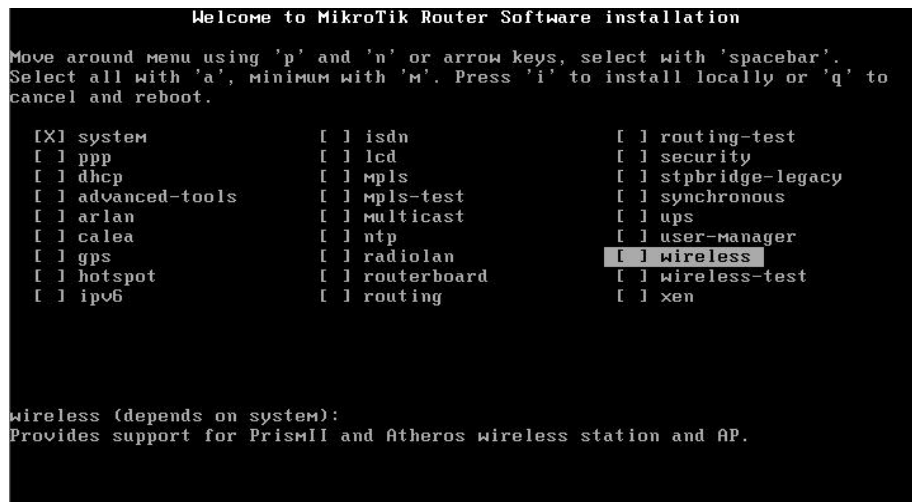
4.1 Instalasi

Pada sub bab ini, akan dijelaskan tentang cara-cara instalasi *software* yang digunakan dalam pelaksanaan kajian ini. Adapun *software* yang akan diinstalasi adalah Mikrotik Router OS, dan FreeRADIUS.

4.1.1 Instalasi Mikrotik Router OS

Pada instalasi Mikrotik Router OS, ada beberapa cara yang dapat dilakukan dalam instalasi Mikrotik Router OS, misalnya langsung menggunakan CD installer Mikrotik atau melalui jaringan atau lebih dikenal dengan istilah *NetInstall*. Dalam kajian ini, instalasi dilakukan dengan menggunakan CD *installer* Mikrotik. Tahapan yang harus dilakukan dalam instalasi adalah sebagai berikut:

- *Booting* dengan menggunakan CD *installer* Mikrotik Router OS.
- Pilih *packages* yang akan di instalasi sesuai kebutuhan, secara *default package* "system" akan ditandai seperti pada gambar 12. *Packages* yang dimaksud adalah program pendukung yang akan digunakan untuk implementasi *wireless* LAN, seperti *system*, *dhcp*, *routing*, *wireless*, dan sebagainya.
- Untuk melakukan proses berikutnya, tekan tombol "i" dan ikuti tahapan selanjutnya.
- Saat proses instalasi system dimulai, secara otomatis proses instalasi akan membuat partisi sendiri, termasuk proses *formatting disk*.
- Semua *packages* akan diinstal, dan setelah proses tersebut selesai Mikrotik perlu di-*restart*.
- Setelah proses instalasi selesai, Mikrotik Router OS sudah dapat digunakan atau dikonfigurasi sesuai kebutuhan.



Gambar 12 . Instalasi Mikrotik

4.1.2 Instalasi FreeRADIUS

Instalasi FreeRADIUS yang dilakukan pada kajian ini dilakukan pada sistem operasi Fedora Core 6. FreeRADIUS yang digunakan adalah versi 1.1.3. Untuk menginstall FreeRADIUS ada beberapa cara, yaitu dengan menggunakan RPM (*Redhat Packages Manager*) yang diinstal secara langsung pada sistem operasi atau menggunakan instalasi melalui jaringan atau *yum*. Pada kajian ini, instalasi dilakukan langsung melalui jaringan. Langkah-langkah untuk instalasi adalah sebagai berikut:

- Konfigurasi file “/etc/yum.repos.d/yum.conf” dengan merujuk pada *web repository* dari *software* yang akan diinstal. Contoh konfigurasinya adalah sebagai berikut:

```
[core]
name=Fedora Core $releasever - $basearch
baseurl=http://reponm.nm.del.ac.id/
enabled=1
gpgcheck=0
```

Pada contoh diatas, *software* yang akan diinstal berasal dari *repository local* PI Del.

- Untuk melakukan instalasi, dengan melakukan perintah berikut pada terminal sistem operasi Fedora Core 6.

```
[root@fedora ~]$ yum install freeradius
```

Hasil dari perintah tersebut adalah seperti pada gambar 13.

- Ikuti proses instalasi berikutnya sampai hasilnya *complete*.

Pada gambar 14 menunjukkan proses instalasi FreeRADIUS dan beberapa *packages* yang ikut diinstal bersamaan dengan *software* FreeRADIUS.

```

root@localhost:/etc/yum.repos.d
File Edit View Terminal Tabs Help
Dependencies Resolved

=====
Package                Arch      Version      Repository    Size
=====
Installing:
freeradius              i386      1.1.3-1      core          1.2 M
Installing for dependencies:
lm_sensors              i386      2.10.0-3.1   core          494 k
net-snmp                i386      1:5.3.1-11.fc6 core          699 k
net-snmp-utils          i386      1:5.3.1-11.fc6 core          178 k

Transaction Summary
=====
Install      4 Package(s)
Update       0 Package(s)
Remove       0 Package(s)

Total download size: 2.5 M
Is this ok [y/N]: y
Downloading Packages:
(1/4): net-snmp-utils-5.3 100% |=====| 178 kB  00:00
(2/4): lm_sensors-2.10.0- 100% |=====| 494 kB  00:00
(3/4): freeradius-1.1.3-1 100% |=====| 1.2 MB  00:00
(4/4): net-snmp-5.3.1-11. 100% |=====| 699 kB  00:00
Running Transaction Test
warning: net-snmp-utils-5.3.1-11.fc6: Header V3 DSA signature: NOKEY, key ID 4f2
a6fd2
Finished Transaction Test

```

Gambar 13 . Instalasi FreeRADIUS

Untuk menjalankan FreeRADIUS dan untuk melihat proses yang sedang berjalan adalah dengan menggunakan perintah berikut ini:

```

[root@freeradius ~]# service radiusd start
[root@freeradius ~]# radiusd -X

```

Untuk menghentikan proses FreeRADIUS yaitu :

```

[root@freeradius ~]# service radiusd stop

```

4.2 Konfigurasi

Pada sub bab ini, akan dijelaskan tentang cara-cara konfigurasi *software* yang digunakan dalam pelaksanaan kajian ini. Adapun *software* yang akan diinstal adalah Mikrotik Router OS sebagai sistem operasinya dan FreeRADIUS sebagai *server* RADIUS .

4.2.1 Konfigurasi Mikrotik Router OS

Konfigurasi Mikrotik yang digunakan sebagai router dapat dilakukan dengan beberapa cara, yaitu dengan menggunakan terminal (*text based*) atau dengan menggunakan *windows application* atau *winbox*. Pada pelaksanaan kajian ini, konfigurasi Mikrotik Router OS dilakukan dengan menggunakan terminal (*text based*) .

4.2.1.1 Konfigurasi IP address

Setiap perangkat jaringan akan memiliki suatu alamat yang disebut dengan *IP address*.

Untuk mengatur *ip address* pada Mikrotik adalah dengan cara berikut ini:

```
[admin@mikrotik] > ip address add address=[address/netmask]  
interface=[interface]
```

Perintah “*interface*” diatas untuk mengatur *IP address* sesuai yang dibutuhkan *interface* yang akan diatur *IP address*-nya. Pada pelaksanaan kajian ini, server Mikrotik memiliki dua buah *interface*, sehingga cara untuk mengatur *ip address* untuk setiap *interface* adalah sbagai berikut:

```
[admin@mikrotik] > ip address add address=172.22.3.10/24  
interface=ether1  
[admin@mikrotik] > ip address add address=10.1.0.1/24  
interface=ether2
```

Untuk melihat hasil konfigurasi *ip address* yaitu dengan menggunakan perintah:

```
[admin@mikrotik] > ip address print
```

4.2.1.2 Konfigurasi DHCP Server

Konfigurasi DHCP dilakukan untuk memberikan alamat IP kepada pengguna yang akan menggunakan *wireless LAN* pada saat melakukan pengujian.

Membuat DHCP *server* :

1. Membuat sebuah *IP address pool*.

```
[admin@mikrotik] > ip pool add name=client ranges=10.1.0.10-  
10.1.0.200
```

2. Konfigurasi DHCP *network* 10.1.0.0/16

```
[admin@mikrotik] > ip dhcp-server network add  
address=10.1.0.0 gateway=10.1.0.1 netmask=16 dns-  
server=172.22.3.10 domain=mikrotik.tk03.net
```

3. Tambahkan DHCP *server*

```
[admin@mikrotik] > ip dhcp-server add interface=ether2  
address-pool=client
```

4.2.1.3 Konfigurasi RADIUS

Pada bagian konfigurasi ini adalah untuk menambahkan layanan, alamat dan *secret* RADIUS pada MikroTik.

```
[admin@mikrotik] > radius add service=hotspot,dhcp  
address172.22.3.10 secret=tk03
```

4.2.1.4 Manambahkan Hotspot

Pada bagian ini akan dijelaskan cara untuk menambahkan layanan *hotspot* baru.

1. Untuk menambahkan layanan *hotspot* adalah sebagai berikut.

```
[admin@mikrotik] > ip hotspot name=wireless interface=ether2  
address-pool=client
```

2. Membuat *profiles* baru yang berisi *hotspot address* dan menggunakan metode autentikasi *login* berdasarkan MAC dengan menggunakan RADIUS.

```
[admin@mikrotik] > ip hotspot profile add name=wireless  
login-by=mac use-radius=yes
```

4.2.1.5 Konfigurasi FreeRADIUS

Pada sub bab ini diuraikan mengenai cara konfigurasi FreeRADIUS. Dalam melakukan konfigurasi FreeRADIUS dengan cara mengubah *file* konfigurasi yang ada pada *server* FreeRADIUS. Daftar *file* yang diubah untuk pelaksanaan kajian ini adalah sebagai berikut:

- /etc/raddb/users
- /etc/raddb/clients.conf
- /etc/raddb/radiusd.conf

Konfigurasi untuk tiap-tiap *file* konfigurasi adalah sebagai berikut:

/etc/raddb/users

File ini digunakan oleh FreeRADIUS untuk menyimpan informasi tentang pengguna. Dalam hal ini, informasi tersebut adalah berupa MAC *address* dari pengguna yang

melakukan koneksi ke FreeRADIUS *server*. Secara *default*, isi *file* ini adalah kosong. *Format* dari isi *file* konfigurasi ini adalah sebagai berikut.

```
"MAC address"    User-Password == ""
```

/etc/raddb/clients.conf

Pada *file* ini, konfigurasi yang dilakukan yaitu dengan menambahkan daftar alamat IP pengguna yang melakukan koneksi ke *server* FreeRADIUS. Isi konfigurasi yang di tambah pada *file* ini adalah sebagai berikut.

```
client 172.22.3.10 {  
    secret          = tk03  
    shortname       = local  
}
```

/etc/raddb/radiusd.conf

File ini merupakan *file* konfigurasi utama pada FreeRADIUS. Adapun isi dari *file* konfigurasi ini adalah pada Lampiran A.

Bab V Pengujian

Pada bab Pengujian akan dijelaskan pengujian yang dilakukan selama pelaksanaan kajian ini yang mencakup pengujian koneksi pengguna terhadap *wireless access point* dengan Mikrotik menggunakan autentikasi MAC *address* pada FreeRADIUS.

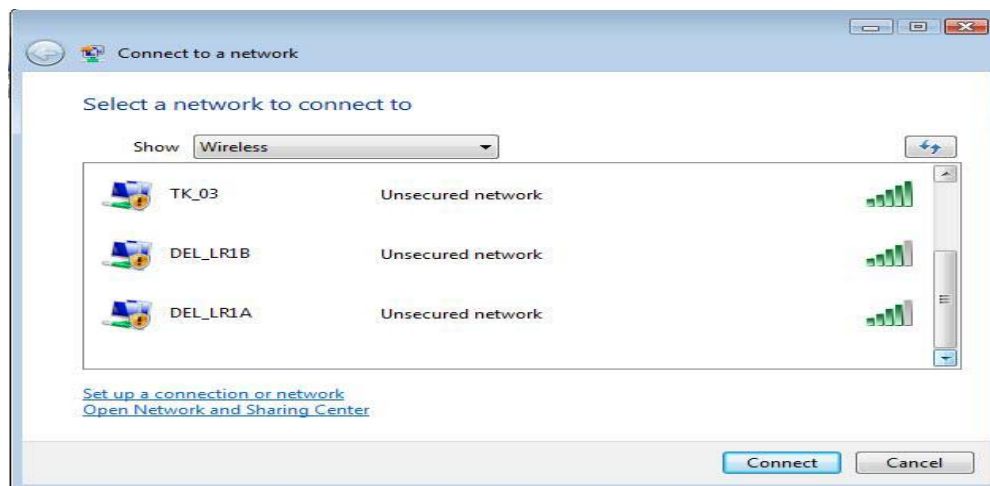
5.1 Pengujian Koneksi Langsung

Pengujian koneksi antara pengguna dengan *wireless access point* dan Mikrotik dengan menggunakan perangkat berikut ini:

1. Laptop yang sudah dilengkapi dengan *wireless network card*.
2. *Wireless Access Point* Linksys WAP4400N.
3. Mikrotik PC *router*.

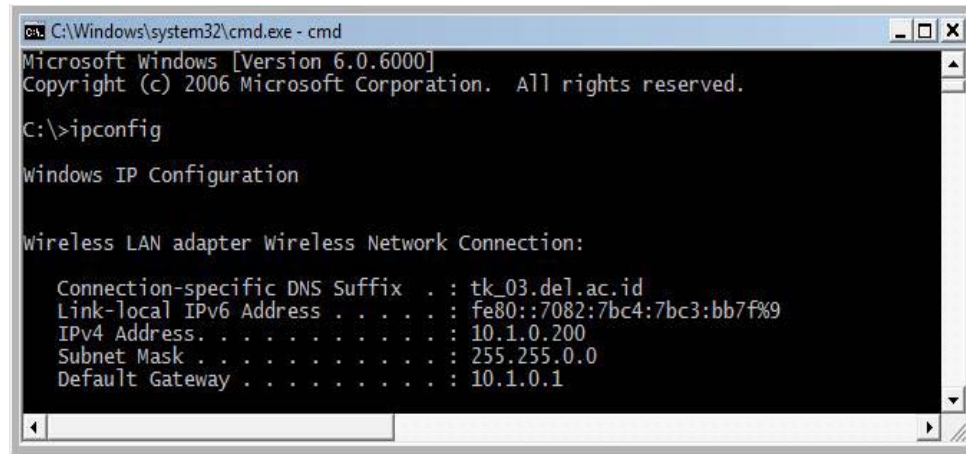
Pada bagian ini, koneksi dilakukan secara langsung tanpa menggunakan autentikasi. Pengujian dilakukan untuk mengetahui apakah pengguna sudah dapat terhubung dengan *wireless access point* dan dapat terhubung dengan pengguna lain yang sudah terhubung pada jaringan yang sama. Pada kajian ini, pemberian alamat IP untuk setiap pengguna yang terhubung dengan menggunakan DHCP yang diatur pada Mikrotik dengan menggunakan alamat IP seperti pada penjelasan bab Implementasi.

Penjelasan mengenai pengujian koneksi secara langsung seperti pada penjelasan berikut ini :



Gambar 14. List jaringan *wireless*

Pada gambar 15, terdapat beberapa daftar jaringan *wireless*. Pada kasus ini, pengujian dilakukan terhadap jaringan *wireless* yang SSID-nya TK_03. Jika pengguna sudah terhubung dengan *wireless access point*, Mikrotik akan memberikan alamat IP secara dinamis (DHCP) seperti pada gambar 16 berikut ini.



Gambar 15 . Alamat IP pengguna

Pengujian koneksi secara langsung pengguna terhadap *wireless access point* dan Mikrotik telah berhasil dilakukan, sehingga pengguna dapat menggunakan akses *wireless*.

5.2 Pengujian Koneksi dengan Menggunakan Autentikasi MAC Address

Pengujian koneksi dengan menggunakan autentikasi MAC address adalah sebagai berikut.

Pada pengujian ini, ada beberapa langkah yang dilakukan, yaitu sebagai berikut:

- Menambahkan MAC *address* pengguna yang akan menggunakan *wireless* pada file `/etc/raddb/users`.
"00-1B-9E-32-E4-DE" User-Password == ""
- Mengakses *wireless* dengan menggunakan MAC *address* yang belum terdaftar dan mencoba melakukan akses terhadap jaringan

Pada pengujian ini, terlihat bahwa atribut yang dikirimkan oleh Mikrotik adalah seperti berikut ini.

```
NAS-Port-Type = Wireless-802.11
Calling-Station-Id = "00:13:E8:0C:90:0D"
Called-Station-Id = "wireless"
NAS-Port-Id = "ether2"
User-Name = "00:13:E8:0C:90:0D"
```

```
NAS-Port = 2154823815
Acct-Session-Id = "80700087"
Framed-IP-Address = 10.1.0.169
Mikrotik-Attr-10 = 0x0a010138
CHAP-Challenge = 0x5484b26465b8dfa04e6b8838444f2fc9
CHAP-Password = 0xbcdfae3611a602c1727441b27f18a78e64
Service-Type = Login-User
WISPr-Logoff-URL = "http://10.1.0.1/logout"
NAS-Identifier = "MikroTik"
NAS-IP-Address = 172.22.3.10
```

Pada pengujian ini, atribut yang dikirim mengalami perubahan dari yang telah dikirim oleh access point. Pada kasus ini, Mikrotik mengubah atribut yang diterima dari *access point* sebelum diteruskan ke FreeRADIUS. Hal yang terpenting pada atribut yang diubah oleh Mikrotik adalah atribut *MAC address* pengguna dijadikan sebagai *username*.

Autentikasi dan otorisasi pada study kasus ini berjalan secara paralel. Jika seorang *client* yang memiliki *MAC address* yang sudah terdaftar, maka autentikasi akan berhasil (FreeRADIUS memberikan balasan bahwa *login ok*). Pengguna tersebut akan mendapatkan koneksi *wireless* dan otorisasi berhasil sehingga pengguna memiliki hak akses terhadap jaringan. *Output* pada FreeRADIUS terdapat pada Lampiran B.

Namun jika *MAC address* yang tidak terdaftar mencoba mengakses WLAN, maka autentikasi gagal (FreeRADIUS memberikan balasan bahwa *login incorrect*). Pengguna tersebut tetap mendapatkan koneksi *wireless*, tetapi tidak bisa melakukan akses apapun terhadap jaringan (otorisasi gagal). *Output* pada FreeRADIUS terdapat pada Lampiran C.

Bab VI

Kesimpulan dan Saran

Pada bab ini diuraikan mengenai kesimpulan dan saran setelah memerhatikan hasil-hasil yang diperoleh selama pelaksanaan Tugas Akhir.

6.1 Kesimpulan

Kesimpulan yang diperoleh selama pengerjaan Tugas Akhir ini adalah:

1. Mengetahui bahwa autentikasi MAC *address* dapat membatasi pengguna untuk menggunakan layanan *wireless* LAN.
2. Dapat mengetahui bagaimana integrasi antara Mikrotik dan FreeRADIUS dalam pengimplementasian akses layanan *wireless* LAN dengan menggunakan autentikasi MAC *address*.
3. Autentikasi MAC *address* dapat dilakukan dengan dua cara, yaitu dengan menggunakan *access point* sebagai *access Server* dan Mikrotik sebagai *access server*. Pada kajian ini, autentikasi yaitu dengan menggunakan Mikrotik sebagai *access server*.

6.2 Saran

Dalam pengimplementasian *wireless* LAN menggunakan autentikasi berdasarkan MAC *address* dengan mengintegrasikan Mikrotik dan FreeRADIUS sehingga dapat menyediakan akses kepada pengguna terhadap suatu penyedia *wireless* LAN dibutuhkan pemahaman yang cukup untuk dapat mempermudah dalam pengembangannya ke masa yang akan datang.

Selain menggunakan autentikasi berdasarkan MAC *address*, juga perlu ditambahkan autentikasi lainnya seperti autentikasi menggunakan *username* dan *password*, atau sebagai contoh yaitu autentikasi dengan menggunakan layanan LDAP agar tingkat keamanannya menjadi lebih baik.

Daftar Pustaka dan Rujukan

Daftar Pustaka

- [R1] <http://www.freeradius.org>, diakses tanggal 30 maret 2009.
- [R2] <http://www.mikrotik.com>, diakses tanggal 30 maret 2009.
- [R3] <http://www.mikrotik.co.id>, diakses tanggal 30 maret 2009.
- [R4] <http://en.wikipedia.org/wiki/Authentication>, diakses 2 april 2009.
- [R5] <http://en.wikipedia.org/wiki/Authorization>, diakses tanggal 2 april 2009.
- [R6] http://en.wikipedia.org/wiki/AAA_protocol, diakses tanggal 2 april 2009.
- [R7] <http://en.wikipedia.org/wiki/RADIUS>, diakses tanggal 2 april 2009.
- [R8] <http://www.mikrotik.com/testdocs/ros/2.9/refman2.9.pdf>, diakses tanggal 30 maret 2009.
- [R9] CISCO, "Wireless-N Access Point with Power Over Ethernet User Guide", CISCO, 2008, halaman 25-30.
- [R10] Miranti, 2008. Metode Keamanan Wireless LAN pada Perangkat Komputer dan Handphone. FAKULTAS ILMU KOMPUTER UNIVERSITAS SRIWIJAYA, 2008.

Rujukan

- [MAC] http://id.wikipedia.org/wiki/MAC_address, diakses tanggal 20 agustus 2009.
- [LAN] http://id.wikipedia.org/wiki/Local_Area_Network, diakses tanggal 20 agustus 2009.
- [IRD] <http://teknik-informatika.com/wireless-lan>, diakses tanggal 20 agustus 2009.
- [TRI] <http://www.elektroindonesia.com/elektro/khu36.html>, diakses tanggal 21 agustus 2009.
- [CC1] CCNA Exploration 4.0 LAN Switching and Wireless, Glossary.
- [WL1] http://en.wikipedia.org/wiki/Wireless_LAN, diakses tanggal 30 maret 2009.
- [CC2] CCNA Exploration 4.0 Network Fundamental, Chapter 9 Ethernet, “9.1.1 Ethernet-Standards and Implementation”.
- [CC3] CCNA Exploration 4.0 Network Fundamental, Chapter 9 Ethernet, “9.1.3 Logical Link Control – Connectiong to the Upper Layers”.
- [CC4] CCNA Exploration 4.0 Network Fundamental, Chapter 9 Ethernet, “9.3.2.2 MAC Address Structure”.
- [RFC] <http://tools.ietf.org/html/rfc2865>, diakses tanggal 21 agustus 2009.
- [AG1] Agung W. Setiawan , Remote Authentication Dial In User Service (RADIUS) untuk Autentikasi Pengguna Wireless LAN, DEPARTEMEN TEKNIK ELEKTRO FAKULTAS TEKNOLOGI INDUSTRI INSTITUT TEKNOLOGI BANDUNG 2005, BAB III.
- [MIK] MikroTik, MikroTik Router OS v2.9 Reference Manual, 2007.

Lampiran

Lampiran A

File konfigurasi FreeRADIUS

/etc/raddb/radiusd.conf

```
##
## radiusd.conf -- FreeRADIUS server configuration file.
##
## http://www.freeradius.org/
## $Id: radiusd.conf.in,v 1.188.2.4.2.12 2006/07/29 19:43:30
nbk Exp $
##

# The location of other config files and
# logfiles are declared in this file
#
# Also general configuration for modules can be done
# in this file, it is exported through the API to
# modules that ask for it.
#
# The configuration variables defined here are of the form
${foo}
# They are local to this file, and do not change from request
to
# request.
#
# The per-request variables are of the form %{Attribute-Name},
and
# are taken from the values of the attribute in the incoming
# request. See 'doc/variables.txt' for more information.

prefix = /usr
exec_prefix = /usr
sysconfdir = /etc
localstatedir = /var
sbindir = /usr/sbin
logdir = ${localstatedir}/log/radius
raddbdir = ${sysconfdir}/raddb
radacctdir = ${logdir}/radacct

# Location of config and logfiles.
confdir = ${raddbdir}
run_dir = ${localstatedir}/run/radiusd

#
# The logging messages for the server are appended to the
# tail of this file.
#
log_file = ${logdir}/radius.log
```

```

#
# libdir: Where to find the rlm_* modules.
#
#   This should be automatically set at configuration time.
#
#   If the server builds and installs, but fails at execution
time
#   with an 'undefined symbol' error, then you can use the libdir
#   directive to work around the problem.
#
#   The cause is usually that a library has been installed on
your
#   system in a place where the dynamic linker CANNOT find it.
When
#   executing as root (or another user), your personal
environment MAY
#   be set up to allow the dynamic linker to find the library.
When
#   executing as a daemon, FreeRADIUS MAY NOT have the same
#   personalized configuration.
#
#   To work around the problem, find out which library contains
that symbol,
#   and add the directory containing that library to the end of
'libdir',
#   with a colon separating the directory names. NO spaces are
allowed.
#
#   e.g. libdir = /usr/local/lib:/opt/package/lib
#
#   You can also try setting the LD_LIBRARY_PATH environment
variable
#   in a script which starts the server.
#
#   If that does not work, then you can re-configure and re-build
the
#   server to NOT use shared libraries, via:
#
#       ./configure --disable-shared
#       make
#       make install
#
libdir = /usr/lib

#
# pidfile: Where to place the PID of the RADIUS server.
#
#   The server may be signalled while it's running by using this
#   file.
#
#   This file is written when ONLY running in daemon mode.
#
#   e.g.: kill -HUP `cat /var/run/radiusd/radiusd.pid`
#
pidfile = ${run_dir}/radiusd.pid

```

```

# user/group: The name (or #number) of the user/group to run
radiusd as.
#
#   If these are commented out, the server will run as the
user/group
#   that started it. In order to change to a different
user/group, you
#   MUST be root ( or have root privileges ) to start the server.
#
#   We STRONGLY recommend that you run the server with as few
permissions
#   as possible. That is, if you're not using shadow passwords,
the
#   user and group items below should be set to 'nobody'.
#
#   On SCO (ODT 3) use "user = nouser" and "group = nogroup".
#
# NOTE that some kernels refuse to setgid(group) when the value
of
# (unsigned)group is above 60000; don't use group nobody on
these systems!
#
# On systems with shadow passwords, you might have to set 'group
= shadow'
# for the server to be able to read the shadow password file.
If you can
# authenticate users while in debug mode, but not in daemon
mode, it may be
# that the debugging mode server is running as a user that can
read the
# shadow info, and the user listed below can not.
#
user = radiusd
group = radiusd

# max_request_time: The maximum time (in seconds) to handle a
request.
#
# Requests which take more time than this to process may be
killed, and
# a REJECT message is returned.
#
# WARNING: If you notice that requests take a long time to be
handled,
# then this MAY INDICATE a bug in the server, in one of the
modules
# used to handle a request, OR in your local configuration.
#
# This problem is most often seen when using an SQL database.
If it takes
# more than a second or two to receive an answer from the SQL
database,

```

```

# then it probably means that you haven't indexed the database.
See your
# SQL server documentation for more information.
#
# Useful range of values: 5 to 120
#
max_request_time = 30

# delete_blocked_requests: If the request takes MORE THAN
'max_request_time'
# to be handled, then maybe the server should delete it.
#
# If you're running in threaded, or thread pool mode, this
setting
# should probably be 'no'. Setting it to 'yes' when using a
threaded
# server MAY cause the server to crash!
#
delete_blocked_requests = no

# cleanup_delay: The time to wait (in seconds) before cleaning
up
# a reply which was sent to the NAS.
#
# The RADIUS request is normally cached internally for a short
period
# of time, after the reply is sent to the NAS. The reply packet
may be
# lost in the network, and the NAS will not see it. The NAS
will then
# re-send the request, and the server will respond quickly with
the
# cached reply.
#
# If this value is set too low, then duplicate requests from the
NAS
# MAY NOT be detected, and will instead be handled as separate
requests.
#
# If this value is set too high, then the server will cache too
many
# requests, and some new requests may get blocked. (See
'max_requests'.)
#
# Useful range of values: 2 to 10
#
cleanup_delay = 5

# max_requests: The maximum number of requests which the server
keeps
# track of. This should be 256 multiplied by the number of
clients.
# e.g. With 4 clients, this number should be 1024.
#

```

```

# If this number is too low, then when the server becomes busy,
# it will not respond to any new requests, until the
'cleanup_delay'
# time has passed, and it has removed the old requests.
#
# If this number is set too high, then the server will use a bit
more
# memory for no real benefit.
#
# If you aren't sure what it should be set to, it's better to
set it
# too high than too low. Setting it to 1000 per client is
probably
# the highest it should be.
#
# Useful range of values: 256 to infinity
#
max_requests = 1024

# bind_address: Make the server listen on a particular IP
address, and
# send replies out from that address. This directive is most
useful
# for machines with multiple IP addresses on one interface.
#
# It can either contain "*", or an IP address, or a fully
qualified
# Internet domain name. The default is "*"
#
# As of 1.0, you can also use the "listen" directive. See below
for
# more information.
#
bind_address = *

# port: Allows you to bind FreeRADIUS to a specific port.
#
# The default port that most NAS boxes use is 1645, which is
historical.
# RFC 2138 defines 1812 to be the new port. Many new servers
and
# NAS boxes use 1812, which can create interoperability
problems.
#
# The port is defined here to be 0 so that the server will pick
up
# the machine's local configuration for the radius port, as
defined
# in /etc/services.
#
# If you want to use the default RADIUS port as defined on your
server,
# (usually through 'grep radius /etc/services') set this to 0
(zero).

```

```

#
# A port given on the command-line via '-p' over-rides this one.
#
# As of 1.0, you can also use the "listen" directive. See below
for
# more information.
#
port = 0

#
# By default, the server uses "bind_address" to listen to all
IP's
# on a machine, or just one IP. The "port" configuration is
used
# to select the authentication port used when listening on those
# addresses.
#
# If you want the server to listen on additional addresses, you
can
# use the "listen" section. A sample section (commented out) is
included
# below. This "listen" section duplicates the functionality of
the
# "bind_address" and "port" configuration entries, but it only
listens
# for authentication packets.
#
# If you comment out the "bind_address" and "port" configuration
entries,
# then it becomes possible to make the server accept only
accounting,
# or authentication packets. Previously, it always listened for
both
# types of packets, and it was impossible to make it listen for
only
# one type of packet.
#
#listen {
    # IP address on which to listen.
    # Allowed values are:
    #     dotted quad (1.2.3.4)
    #     hostname    (radius.example.com)
    #     wildcard    (*)
# ipaddr = *

    # Port on which to listen.
    # Allowed values are:
    #     integer port number (1812)
    #     0 means "use /etc/services for the proper port"
# port = 0

    # Type of packets to listen for.
    # Allowed values are:
    #     auth listen for authentication packets

```



```

#       acct listen for accounting packets
#
#       type = auth
#}

# hostname_lookups: Log the names of clients or just their IP
addresses
# e.g., www.freeradius.org (on) or 206.47.27.232 (off).
#
# The default is 'off' because it would be overall better for
the net
# if people had to knowingly turn this feature on, since
enabling it
# means that each client request will result in AT LEAST one
lookup
# request to the nameserver. Enabling hostname_lookups will
also
# mean that your server may stop randomly for 30 seconds from
time
# to time, if the DNS requests take too long.
#
# Turning hostname lookups off also means that the server won't
block
# for 30 seconds, if it sees an IP address which has no name
associated
# with it.
#
# allowed values: {no, yes}
#
hostname_lookups = no

# Core dumps are a bad thing. This should only be set to 'yes'
# if you're debugging a problem with the server.
#
# allowed values: {no, yes}
#
allow_core_dumps = no

# Regular expressions
#
# These items are set at configure time. If they're set to
"yes",
# then setting them to "no" turns off regular expression
support.
#
# If they're set to "no" at configure time, then setting them to
"yes"
# WILL NOT WORK. It will give you an error.
#
regular_expressions    = yes
extended_expressions   = yes

```

```

# Log the full User-Name attribute, as it was found in the
request.
#
# allowed values: {no, yes}
#
log_stripped_names = no

# Log authentication requests to the log file.
#
# allowed values: {no, yes}
#
log_auth = yes

# Log passwords with the authentication requests.
# log_auth_badpass - logs password if it's rejected
# log_auth_goodpass - logs password if it's correct
#
# allowed values: {no, yes}
#
log_auth_badpass = no
log_auth_goodpass = no

# usercollide: Turn "username collision" code on and off. See
the
# "doc/duplicate-users" file
#
# WARNING
# !!!!!!! Setting this to "yes" may result in the server
behaving
# !!!!!!! strangely. The "username collision" code will ONLY
work
# !!!!!!! with clear-text passwords. Even then, it may not do
what
# !!!!!!! you want, or what you expect.
# !!!!!!!
# !!!!!!! We STRONGLY RECOMMEND that you do not use this
feature,
# !!!!!!! and that you find another way of acheiving the same
goal.
# !!!!!!!
# !!!!!!! e,g. module fail-over. See
'doc/configurable_failover'
# WARNING
#
usercollide = no

# lower_user / lower_pass:
# Lower case the username/password "before" or "after"
# attempting to authenticate.
#
# If "before", the server will first modify the request and then
try
# to auth the user. If "after", the server will first auth
using the

```

```

# values provided by the user. If that fails it will reprocess
the
# request after modifying it as you specify below.
#
# This is as close as we can get to case insensitivity. It is
the
# admin's job to ensure that the username on the auth db side is
# *also* lowercase to make this work
#
# Default is 'no' (don't lowercase values)
# Valid values = "before" / "after" / "no"
#
lower_user = no
lower_pass = no

# nospace_user / nospace_pass:
#
# Some users like to enter spaces in their username or password
# incorrectly. To save yourself the tech support call, you can
# eliminate those spaces here:
#
# Default is 'no' (don't remove spaces)
# Valid values = "before" / "after" / "no" (explanation above)
#
nospace_user = no
nospace_pass = no

# The program to execute to do concurrency checks.
checkrad = ${sbindir}/checkrad

# SECURITY CONFIGURATION
#
# There may be multiple methods of attacking on the server.
This
# section holds the configuration items which minimize the
impact
# of those attacks
#
security {
    #
    # max_attributes: The maximum number of attributes
    # permitted in a RADIUS packet. Packets which have MORE
    # than this number of attributes in them will be dropped.
    #
    # If this number is set too low, then no RADIUS packets
    # will be accepted.
    #
    # If this number is set too high, then an attacker may be
    # able to send a small number of packets which will cause
    # the server to use all available memory on the machine.
    #
    # Setting this number to 0 means "allow any number of
attributes"
    max_attributes = 200

```

```

#
# reject_delay: When sending an Access-Reject, it can be
# delayed for a few seconds. This may help slow down a DoS
# attack. It also helps to slow down people trying to
brute-force
# crack a users password.
#
# Setting this number to 0 means "send rejects immediately"
#
# If this number is set higher than 'cleanup_delay', then
the
# rejects will be sent at 'cleanup_delay' time, when the
request
# is deleted from the internal cache of requests.
#
# Useful ranges: 1 to 5
reject_delay = 1

#
# status_server: Whether or not the server will respond
# to Status-Server requests.
#
# Normally this should be set to "no", because they're
useless.
# See: http://www.freeradius.org/rfc/rfc2865.html#Keep-Alives
#
# However, certain NAS boxes may require them.
#
# When sent a Status-Server message, the server responds
with
# an Access-Accept packet, containing a Reply-Message
attribute,
# which is a string describing how long the server has been
# running.
#
status_server = no
}

# PROXY CONFIGURATION
#
# proxy_requests: Turns proxying of RADIUS requests on or off.
#
# The server has proxying turned on by default. If your system
is NOT
# set up to proxy requests to another server, then you can turn
proxying
# off here. This will save a small amount of resources on the
server.
#
# If you have proxying turned off, and your configuration files
say
# to proxy a request, then an error message will be logged.

```

```

#
# To disable proxying, change the "yes" to "no", and comment the
# $INCLUDE line.
#
# allowed values: {no, yes}
#
proxy_requests = yes
$INCLUDE ${confdir}/proxy.conf

# CLIENTS CONFIGURATION
#
# Client configuration is defined in "clients.conf".
#

# The 'clients.conf' file contains all of the information from
the old
# 'clients' and 'naslist' configuration files. We recommend
that you
# do NOT use 'client's or 'naslist', although they are still
# supported.
#
# Anything listed in 'clients.conf' will take precedence over
the
# information from the old-style configuration files.
#
$INCLUDE ${confdir}/clients.conf

# SNMP CONFIGURATION
#
# Snmp configuration is only valid if SNMP support was enabled
# at compile time.
#
# To enable SNMP querying of the server, set the value of the
# 'snmp' attribute to 'yes'
#
snmp = no
$INCLUDE ${confdir}/snmp.conf

# THREAD POOL CONFIGURATION
#
# The thread pool is a long-lived group of threads which
# take turns (round-robin) handling any incoming requests.
#
# You probably want to have a few spare threads around,
# so that high-load situations can be handled immediately. If
you
# don't have any spare threads, then the request handling will
# be delayed while a new thread is created, and added to the
pool.
#
# You probably don't want too many spare threads around,

```

```

# otherwise they'll be sitting there taking up resources, and
# not doing anything productive.
#
# The numbers given below should be adequate for most
situations.
#
thread pool {
    # Number of servers to start initially --- should be a
reasonable
    # ballpark figure.
    start_servers = 5

    # Limit on the total number of servers running.
    #
    # If this limit is ever reached, clients will be LOCKED
OUT, so it
    # should NOT BE SET TOO LOW. It is intended mainly as a
brake to
    # keep a runaway server from taking the system with it as
it spirals
    # down...
    #
    # You may find that the server is regularly reaching the
    # 'max_servers' number of threads, and that increasing
    # 'max_servers' doesn't seem to make much difference.
    #
    # If this is the case, then the problem is MOST LIKELY that
    # your back-end databases are taking too long to respond,
and
    # are preventing the server from responding in a timely
manner.
    #
    # The solution is NOT do keep increasing the 'max_servers'
    # value, but instead to fix the underlying cause of the
    # problem: slow database, or 'hostname_lookups=yes'.
    #
    # For more information, see 'max_request_time', above.
    #
    max_servers = 32

    # Server-pool size regulation. Rather than making you
guess
    # how many servers you need, FreeRADIUS dynamically adapts
to
    # the load it sees, that is, it tries to maintain enough
    # servers to handle the current load, plus a few spare
    # servers to handle transient load spikes.
    #
    # It does this by periodically checking how many servers
are
    # waiting for a request. If there are fewer than
    # min_spare_servers, it creates a new spare. If there are
    # more than max_spare_servers, some of the spares die off.
    # The default values are probably OK for most sites.

```

```

#
min_spare_servers = 3
max_spare_servers = 10

# There may be memory leaks or resource allocation problems
with # the server. If so, set this value to 300 or so, so that
the # resources will be cleaned up periodically.
#
# This should only be necessary if there are serious bugs
in the # server which have not yet been fixed.
#
# '0' is a special value meaning 'infinity', or 'the
servers never
# exit'
max_requests_per_server = 0
}

# MODULE CONFIGURATION
#
# The names and configuration of each module is located in this
section.
#
# After the modules are defined here, they may be referred to by
name,
# in other sections of this configuration file.
#
modules {
#
# Each module has a configuration as follows:
#
#     name [ instance ] {
#         config_item = value
#         ...
#     }
#
# The 'name' is used to load the 'rlm_name' library
# which implements the functionality of the module.
#
# The 'instance' is optional. To have two different
instances
# of a module, it first must be referred to by 'name'.
# The different copies of the module are then created by
# inventing two 'instance' names, e.g. 'instance1' and
'instance2'
#
# The instance names can then be used in later
configuration
# INSTEAD of the original 'name'. See the 'radutmp'
configuration
# below for an example.
#

```

```

        # PAP module to authenticate users based on their stored
password
        #
        # Supports multiple encryption schemes
        # clear: Clear text
        # crypt: Unix crypt
        # md5: MD5 encryption
        # sha1: SHA1 encryption.
        # DEFAULT: crypt
        pap {
            encryption_scheme = crypt
        }

        # CHAP module
        #
        # To authenticate requests containing a CHAP-Password
attribute.
        #
        chap {
            authtype = CHAP
        }

        # Pluggable Authentication Modules
        #
        # For Linux, see:
        # http://www.kernel.org/pub/linux/libs/pam/index.html
        #
        # WARNING: On many systems, the system PAM libraries have
        # memory leaks! We STRONGLY SUGGEST that you do
not
        # use PAM for authentication, due to those memory
leaks.
        #
        pam {
            #
            # The name to use for PAM authentication.
            # PAM looks in /etc/pam.d/${pam_auth_name}
            # for it's configuration. See 'redhat/radiusd-pam'
            # for a sample PAM configuration file.
            #
            # Note that any Pam-Auth attribute set in the
'authorize'
            # section will over-ride this one.
            #
            pam_auth = radiusd
        }

        # Unix /etc/passwd style authentication
        #
        unix {
            #
            # Cache /etc/passwd, /etc/shadow, and /etc/group
            #

```



```

# The default is to NOT cache them.
#
# For FreeBSD and NetBSD, you do NOT want to enable
# the cache, as it's password lookups are done via a
# database, so set this value to 'no'.
#
# Some systems (e.g. RedHat Linux with pam_pwbd) can
# take *seconds* to check a password, when the passwd
# file containing 1000's of entries. For those
systems,
# you should set the cache value to 'yes', and set
# the locations of the 'passwd', 'shadow', and
'group'
# files, below.
#
# allowed values: {no, yes}
cache = no

# Reload the cache every 600 seconds (10mins). 0 to
disable.
cache_reload = 600

#
# Define the locations of the normal passwd, shadow,
and
# group files.
#
# 'shadow' is commented out by default, because not
all
# systems have shadow passwords.
#
# To force the module to use the system password
functions,
# instead of reading the files, leave the following
entries
# commented out.
#
# This is required for some systems, like FreeBSD,
# and Mac OSX.
#
# passwd = /etc/passwd
shadow = /etc/shadow
# group = /etc/group

#
# The location of the "wtmp" file.
# This should be moved to it's own module soon.
#
# The only use for 'radlast'. If you don't use
# 'radlast', then you can comment out this item.
#
radwtmp = ${logdir}/radwtmp
}

```

```

# Extensible Authentication Protocol
#
# For all EAP related authentications.
# Now in another file, because it is very large.
#
$INCLUDE ${confdir}/eap.conf

# Microsoft CHAP authentication
#
# This module supports MS-CHAP and MS-CHAPv2
authentication.
# It also enforces the SMB-Account-Ctrl attribute.
#
mschap {
#
# As of 0.9, the mschap module does NOT support
# reading from /etc/smbpasswd.
#
# If you are using /etc/smbpasswd, see the 'passwd'
# module for an example of how to use /etc/smbpasswd

# if use_mppe is not set to no mschap will
# add MS-CHAP-MPPE-Keys for MS-CHAPv1 and
# MS-MPPE-Recv-Key/MS-MPPE-Send-Key for MS-CHAPv2
#
#use_mppe = no

# if mppe is enabled require_encryption makes
# encryption moderate
#
#require_encryption = yes

# require_strong always requires 128 bit key
# encryption
#
#require_strong = yes

# Windows sends us a username in the form of
# DOMAIN\user, but sends the challenge response
# based on only the user portion. This hack
# corrects for that incorrect behavior.
#
#with_ntdomain_hack = no

# The module can perform authentication itself, OR
# use a Windows Domain Controller. This configuration
# directive tells the module to call the ntlm_auth
# program, which will do the authentication, and
return
# the NT-Key. Note that you MUST have "winbindd" and
# "nmbd" running on the local machine for ntlm_auth
# to work. See the ntlm_auth program documentation
# for details.
#

```

```

        # Be VERY careful when editing the following line!
        #
        #ntlm_auth = "/path/to/ntlm_auth --request-nt-key --
username=%{Stripped-User-Name:-%{User-Name:-None}} --
challenge=%{mschap:Challenge:-00} --nt-response=%{mschap:NT-
Response:-00}"
    }

    # Lightweight Directory Access Protocol (LDAP)
    #
    # This module definition allows you to use LDAP for
    # authorization and authentication.
    #
    # See doc/rlm_ldap for description of configuration options
    # and sample authorize{} and authenticate{} blocks
    #
    # However, LDAP can be used for authentication ONLY when
the
    # Access-Request packet contains a clear-text User-Password
    # attribute. LDAP authentication will NOT work for any
other
    # authentication method.
    #
    # This means that LDAP servers don't understand EAP. If
you
    # force "Auth-Type = LDAP", and then send the server a
    # request containing EAP authentication, then
authentication
    # WILL NOT WORK.
    #
    # The solution is to use the default configuration, which
does
    # work.
    #
    # Setting "Auth-Type = LDAP" is ALMOST ALWAYS WRONG. We
    # really can't emphasize this enough.
    #
    ldap {
        server = "ldap.your.domain"
        # identity = "cn=admin,o=My Org,c=UA"
        # password = mypass
        basedn = "o=My Org,c=UA"
        filter = "(uid=%{Stripped-User-Name:-%{User-Name}})"
        # base_filter = "(objectclass=radiusprofile)"

        # set this to 'yes' to use TLS encrypted connections
        # to the LDAP database by using the StartTLS extended
        # operation.
        # The StartTLS operation is supposed to be used with
normal
        # ldap connections instead of using ldaps (port 689)
connections
        start_tls = no

```

```

# tls_cacertfile= /path/to/cacert.pem
# tls_cacertdir      = /path/to/ca/dir/
# tls_certfile       = /path/to/radius.crt
# tls_keyfile        = /path/to/radius.key
# tls_randfile       = /path/to/rnd
# tls_require_cert   = "demand"

# default_profile = "cn=radprofile,ou=dialup,o=My
Org,c=UA"
# profile_attribute = "radiusProfileDn"
access_attr = "dialupAccess"

# Mapping of RADIUS dictionary attributes to LDAP
# directory attributes.
dictionary_mapping = ${raddbdir}/ldap.attrmap

ldap_connections_number = 5

#
# NOTICE: The password_header directive is NOT case
insensitive
#
# password_header = "{clear}"
#
# Set:
#     password_attribute = nspmPassword
#
# to get the user's password from a Novell eDirectory
# backend. This will work *only if* freeRADIUS is
# configured to build with --with-edir option.
#
# The server can usually figure this out on its own,
and pull
# the correct User-Password or NT-Password from the
database.
#
# Note that NT-Passwords MUST be stored as a 32-digit
hex
# string, and MUST start off with "0x", such as:
#
#     0x000102030405060708090a0b0c0d0e0f
#
# Without the leading "0x", NT-Passwords will not
work.
#
# This goes for NT-Passwords stored in SQL, too.
#
# password_attribute = userPassword
#
# Un-comment the following to disable Novell
eDirectory account
# policy check and intruder detection. This will work
*only if*

```

```

# FreeRADIUS is configured to build with --with-edir
option.
#
# edir_account_policy_check=no
#
# groupname_attribute = cn
# groupmembership_filter =
"(|(&(objectClass=GroupOfNames)(member=%{Ldap-
UserDn}))(&(objectClass=GroupOfUniqueNames)(uniquemember=%{Ldap-
UserDn})))"
# groupmembership_attribute = radiusGroupName
# timeout = 4
# timelimit = 3
# net_timeout = 1
# compare_check_items = yes
# do_xlat = yes
# access_attr_used_for_allow = yes

#
# By default, if the packet contains a User-Password,
# and no other module is configured to handle the
# authentication, the LDAP module sets itself to do
# LDAP bind for authentication.
#
# You can disable this behavior by setting the
following
# configuration entry to "no".
#
# allowed values: {no, yes}
# set_auth_type = yes
}

# passwd module allows to do authorization via any passwd-
like
# file and to extract any attributes from these modules
#
# parameters are:
# filename - path to filename
# format - format for filename record. This parameters
# correlates record in the passwd file and RADIUS
# attributes.
#
# Field marked as '*' is key field. That is, the
parameter
# with this name from the request is used to
search for
# the record from passwd file
# Attribute marked as '=' is added to reply_items
instead
# of default configure_items
# Attribute marked as '~' is added to request_items
#
# Field marked as ',' may contain a comma
separated list

```

```

#           of attributes.
#   authtype - if record found this Auth-Type is used to
authenticate
#           user
#   hashsize - hashtable size. If 0 or not specified records
are not
#           stored in memory and file is red on every
request.
#   allowmultiplekeys - if few records for every key are
allowed
#   ignorenislike - ignore NIS-related records
#   delimiter - symbol to use as a field separator in passwd
file,
#           for format ':' symbol is always used. '\0',
'\n' are
#           not allowed
#
#   An example configuration for using /etc/smbpasswd.
#
#passwd etc_smbpasswd {
#   filename = /etc/smbpasswd
#   format = "*User-Name::LM-Password:NT-Password:SMB-
Account-CTRL-TEXT::"
#   authtype = MS-CHAP
#   hashsize = 100
#   ignorenislike = no
#   allowmultiplekeys = no
#}

#   Similar configuration, for the /etc/group file. Adds a
Group-Name
#   attribute for every group that the user is member of.
#
#passwd etc_group {
#   filename = /etc/group
#   format = "=Group-Name::*,User-Name"
#   hashsize = 50
#   ignorenislike = yes
#   allowmultiplekeys = yes
#   delimiter = ":"
#}

# Realm module, for proxying.
#
#   You can have multiple instances of the realm module to
#   support multiple realm syntaxs at the same time. The
#   search order is defined by the order in the authorize and
#   preacct sections.
#
#   Four config options:
#   format           - must be 'prefix' or 'suffix'
#   delimiter        - must be a single character
#   ignore_default   - set to 'yes' or 'no'

```

```

#         ignore_null      - set to 'yes' or 'no'
#
# ignore_default and ignore_null can be set to 'yes' to
prevent # the module from matching against DEFAULT or NULL realms.
This
# may be useful if you have have multiple instances of the
# realm module.
#
# They both default to 'no'.
#

# 'realm/username'
#
# Using this entry, IPASS users have their realm set to
"IPASS".
realm IPASS {
    format = prefix
    delimiter = "/"
    ignore_default = no
    ignore_null = no
}

# 'username@realm'
#
realm suffix {
    format = suffix
    delimiter = "@"
    ignore_default = no
    ignore_null = no
}

# 'username%realm'
#
realm realmpercent {
    format = suffix
    delimiter = "%"
    ignore_default = no
    ignore_null = no
}

#
# 'domain\user'
#
realm ntdomain {
    format = prefix
    delimiter = "\\\"
    ignore_default = no
    ignore_null = no
}

# A simple value checking module
#

```

```

# It can be used to check if an attribute value in the
request
# matches a (possibly multi valued) attribute in the check
# items This can be used for example for caller-id
# authentication. For the module to run, both the request
# attribute and the check items attribute must exist
#
# i.e.
# A user has an ldap entry with 2 radiusCallingStationId
# attributes with values "12345678" and "12345679". If we
# enable rlm_checkval, then any request which contains a
# Calling-Station-Id with one of those two values will be
# accepted. Requests with other values for
# Calling-Station-Id will be rejected.
#
# Regular expressions in the check attribute value are
allowed
# as long as the operator is '=~'
#
checkval {
    # The attribute to look for in the request
    item-name = Calling-Station-Id

    # The attribute to look for in check items. Can be
multi valued
    check-name = Calling-Station-Id

    # The data type. Can be
    # string,integer,ipaddr,date,abinary,octets
    data-type = string

    # If set to yes and we dont find the item-name
attribute in the
    # request then we send back a reject
    # DEFAULT is no
    #notfound-reject = no
}

# rewrite arbitrary packets. Useful in accounting and
authorization.
#
#
# The module can also use the Rewrite-Rule attribute. If it
# is set and matches the name of the module instance, then
# that module instance will be the only one which runs.
#
# Also if new_attribute is set to yes then a new attribute
# will be created containing the value replacewith and it
# will be added to searchin (packet, reply, proxy,
proxy_reply or config).
# searchfor,ignore_case and max_matches will be ignored in
that case.
#

```



```

        # Backreferences are supported: %{0} will contain the string
the whole match
        # and %{1} to %{8} will contain the contents of the 1st to
the 8th parentheses
        #
        # If max_matches is greater than one the backreferences will
correspond to the
        # first match

        #
#attr_rewrite sanecallerid {
        # attribute = Called-Station-Id
        # may be "packet", "reply", "proxy", "proxy_reply" or
"config"
        # searchin = packet
        # searchfor = "[+ ]"
        # replacewith = ""
        # ignore_case = no
        # new_attribute = no
        # max_matches = 10
        # ## If set to yes then the replace string will be
appended to the original string
        # append = no
        #}

        # Preprocess the incoming RADIUS request, before handing it
off
        # to other modules.
        #
        # This module processes the 'huntgroups' and 'hints' files.
        # In addition, it re-writes some weird attributes created
        # by some NASes, and converts the attributes into a form
which
        # is a little more standard.
        #
        preprocess {
            huntgroups = ${confdir}/huntgroups
            hints = ${confdir}/hints

            # This hack changes Ascend's wierd port numberings
            # to standard 0-??? port numbers so that the "+" works
            # for IP address assignments.
            with_ascend_hack = no
            ascend_channels_per_line = 23

            # Windows NT machines often authenticate themselves as
            # NT_DOMAIN\username
            #
            # If this is set to 'yes', then the NT_DOMAIN portion
            # of the user-name is silently discarded.
            #
            # This configuration entry SHOULD NOT be used.
            # See the "realms" module for a better way to handle
            # NT domains.

```

```

with_ntdomain_hack = no

# Specialix Jetstream 8500 24 port access server.
#
# If the user name is 10 characters or longer, a "/"
# and the excess characters after the 10th are
# appended to the user name.
#
# If you're not running that NAS, you don't need
# this hack.
with_specialix_jetstream_hack = no

# Cisco (and Quintum in Cisco mode) sends it's VSA
attributes
# with the attribute name *again* in the string, like:
#
#   H323-Attribute = "h323-attribute=value".
#
# If this configuration item is set to 'yes', then
# the redundant data in the the attribute text is
stripped
# out.  The result is:
#
#   H323-Attribute = "value"
#
# If you're not running a Cisco or Quintum NAS, you
don't
# need this hack.
with_cisco_vsa_hack = no
}

# Livingston-style 'users' file
#
files {
    usersfile = ${confdir}/users
    acctusersfile = ${confdir}/acct_users
    preproxy_usersfile = ${confdir}/preproxy_users

    # If you want to use the old Cistron 'users' file
    # with FreeRADIUS, you should change the next line
    # to 'compat = cistron'.  You can the copy your
'users'
    # file from Cistron.
    compat = no
}

# Write a detailed log of all accounting records received.
#
detail {
    # Note that we do NOT use NAS-IP-Address here, as
    # that attribute MAY BE from the originating NAS, and
    # NOT from the proxy which actually sent us the
    # request.  The Client-IP-Address attribute is ALWAYS
    # the address of the client which sent us the

```

```

# request.
#
# The following line creates a new detail file for
# every radius client (by IP address or hostname).
# In addition, a new detail file is created every
# day, so that the detail file doesn't have to go
# through a 'log rotation'
#
# If your detail files are large, you may also want
# to add a ':%H' (see doc/variables.txt) to the end
# of it, to create a new detail file every hour,
e.g.:
#
# ....detail-%Y%m%d:%H
#
# This will create a new detail file for every hour.
#
detailfile = ${radacctdir}/${Client-IP-
Address}/detail-%Y%m%d

#
# The Unix-style permissions on the 'detail' file.
#
# The detail file often contains secret or private
# information about users. So by keeping the file
# permissions restrictive, we can prevent unwanted
# people from seeing that information.
detailperm = 0600

#
# Certain attributes such as User-Password may be
# "sensitive", so they should not be printed in the
# detail file. This section lists the attributes
# that should be suppressed.
#
# The attributes should be listed one to a line.
#
#suppress {
#    # User-Password
#}

#
# Many people want to log authentication requests.
# Rather than modifying the server core to print out more
# messages, we can use a different instance of the 'detail'
# module, to log the authentication requests to a file.
#
# You will also need to un-comment the 'auth_log' line
# in the 'authorize' section, below.
#
detail auth_log {
    detailfile = ${radacctdir}/${Client-IP-Address}/auth-
detail-%Y%m%d

```

```

#
# This MUST be 0600, otherwise anyone can read
# the users passwords!
# detailperm = 0600
}

#
# This module logs authentication reply packets sent
# to a NAS. Both Access-Accept and Access-Reject packets
# are logged.
#
# You will also need to un-comment the 'reply_log' line
# in the 'post-auth' section, below.
#
detail reply_log {
    detailfile = ${radacctdir}/%{Client-IP-
Address}/reply-detail-%Y%m%d

#
# This MUST be 0600, otherwise anyone can read
# the users passwords!
# detailperm = 0600
}

#
# This module logs packets proxied to a home server.
#
# You will also need to un-comment the 'pre_proxy_log' line
# in the 'pre-proxy' section, below.
#
# detail pre_proxy_log {
# detailfile = ${radacctdir}/%{Client-IP-Address}/pre-
proxy-detail-%Y%m%d

#
# This MUST be 0600, otherwise anyone can read
# the users passwords!
# detailperm = 0600
# }

#
# This module logs response packets from a home server.
#
# You will also need to un-comment the 'post_proxy_log'
line
# in the 'post-proxy' section, below.
#
# detail post_proxy_log {
# detailfile = ${radacctdir}/%{Client-IP-
Address}/post-proxy-detail-%Y%m%d

#
# This MUST be 0600, otherwise anyone can read

```

```

        # the users passwords!
        # detailperm = 0600
    # }

    #
    # The rlm_sql_log module appends the SQL queries in a log
    # file which is read later by the radsqlrelay program.
    #
    # This module only performs the dynamic expansion of the
    # variables found in the SQL statements. No operation is
    # executed on the database server. (this could be done
    # later by an external program) That means the module is
    # useful only with non-"SELECT" statements.
    #
    # See rlm_sql_log(5) manpage.
    #
    sql_log {
    #     path = ${radacctdir}/sql-relay
    #     acct_table = "radacct"
    #     postauth_table = "radpostauth"
    #
    #     Start = "INSERT INTO ${acct_table} (AcctSessionId,
    #     UserName, \
    #         NASIPAddress, FramedIPAddress, AcctStartTime,
    #     AcctStopTime, \
    #         AcctSessionTime, AcctTerminateCause) VALUES
    #     \
    #         ('${Acct-Session-Id}', '${User-Name}', '${NAS-IP-
    #     Address}', \
    #         '${Framed-IP-Address}', '%S', '0', '0', '');"
    #     Stop = "INSERT INTO ${acct_table} (AcctSessionId,
    #     UserName, \
    #         NASIPAddress, FramedIPAddress, AcctStartTime,
    #     AcctStopTime, \
    #         AcctSessionTime, AcctTerminateCause) VALUES
    #     \
    #         ('${Acct-Session-Id}', '${User-Name}', '${NAS-IP-
    #     Address}', \
    #         '${Framed-IP-Address}', '0', '%S', '${Acct-Session-
    #     Time}', \
    #         '${Acct-Terminate-Cause}');"
    #     Alive = "INSERT INTO ${acct_table} (AcctSessionId,
    #     UserName, \
    #         NASIPAddress, FramedIPAddress, AcctStartTime,
    #     AcctStopTime, \
    #         AcctSessionTime, AcctTerminateCause) VALUES
    #     \
    #         ('${Acct-Session-Id}', '${User-Name}', '${NAS-IP-
    #     Address}', \
    #         '${Framed-IP-Address}', '0', '0', '${Acct-Session-
    #     Time}', '');"
    #
    #     Post-Auth = "INSERT INTO ${postauth_table}
    \

```

```

#         (user, pass, reply, date) VALUES
# \
#         ('${User-Name}', '${User-Password:-Chap-Password}',
# \
#         '${reply:Packet-Type}', '%S');"
#     }

#
# Create a unique accounting session Id. Many NASes re-use
# or repeat values for Acct-Session-Id, causing no end of
# confusion.
#
# This module will add a (probably) unique session id
# to an accounting packet based on the attributes listed
# below found in the packet. See doc/rlm_acct_unique for
# more information.
#
acct_unique {
    key = "User-Name, Acct-Session-Id, NAS-IP-Address,
Client-IP-Address, NAS-Port"
}

# Include another file that has the SQL-related
configuration.
# This is another file only because it tends to be big.
#
# The following configuration file is for use with MySQL.
#
# For Postgresql, use:          ${confdir}/postgresql.conf
# For MS-SQL, use:              ${confdir}/mssql.conf
# For Oracle, use:              ${confdir}/oraclesql.conf
#
# $INCLUDE ${confdir}/sql.conf

# For Cisco VoIP specific accounting with Postgresql,
# use:          ${confdir}/pgsql-voip.conf
#
# You will also need the sql schema from:
#     src/billing/cisco_h323_db_schema-postgres.sql
# Note: This config can be use AS WELL AS the standard sql
# config if you need SQL based Auth

# Write a 'utmp' style file, of which users are currently
# logged in, and where they've logged in from.
#
# This file is used mainly for Simultaneous-Use checking,
# and also 'radwho', to see who's currently logged in.
#
radutmp {
    # Where the file is stored. It's not a log file,
    # so it doesn't need rotating.

```

```

#
filename = ${logdir}/radutmp

# The field in the packet to key on for the
# 'user' name, If you have other fields which you
want
# to use to key on to control Simultaneous-Use,
# then you can use them here.
#
# Note, however, that the size of the field in the
# 'utmp' data structure is small, around 32
# characters, so that will limit the possible choices
# of keys.
#
# You may want instead: %{Stripped-User-Name:-%{User-
Name}}
username = %{User-Name}

# Whether or not we want to treat "user" the same
# as "USER", or "User". Some systems have problems
# with case sensitivity, so this should be set to
# 'no' to enable the comparisons of the key attribute
# to be case insensitive.
#
case_sensitive = yes

# Accounting information may be lost, so the user MAY
# have logged off of the NAS, but we haven't noticed.
# If so, we can verify this information with the NAS,
#
# If we want to believe the 'utmp' file, then this
# configuration entry can be set to 'no'.
#
check_with_nas = yes

# Set the file permissions, as the contents of this
file
# are usually private.
perm = 0600

callerid = "yes"
}

# "Safe" radutmp - does not contain caller ID, so it can be
# world-readable, and radwho can work for normal users,
without
# exposing any information that isn't already exposed by
who(1).
#
# This is another 'instance' of the radutmp module, but it
is given
# then name "sradutmp" to identify it later in the
"accounting"

```

```

# section.
radutmp sradutmp {
    filename = ${logdir}/sradutmp
    perm = 0644
    callerid = "no"
}

# attr_filter - filters the attributes received in replies
from
# proxied servers, to make sure we send back to our RADIUS
client
# only allowed attributes.
attr_filter {
    attrsfile = ${confdir}/attrs
}

# counter module:
# This module takes an attribute (count-attribute).
# It also takes a key, and creates a counter for each
unique
# key. The count is incremented when accounting packets
are
# received by the server. The value of the increment
depends
# on the attribute type.
# If the attribute is Acct-Session-Time or of an integer
type we add the
# value of the attribute. If it is anything else we
increase the
# counter by one.
#
# The 'reset' parameter defines when the counters are all
reset to
# zero. It can be hourly, daily, weekly, monthly or never.
#
# hourly: Reset on 00:00 of every hour
# daily: Reset on 00:00:00 every day
# weekly: Reset on 00:00:00 on sunday
# monthly: Reset on 00:00:00 of the first day of each month
#
# It can also be user defined. It should be of the form:
# num[hwdm] where:
# h: hours, d: days, w: weeks, m: months
# If the letter is omitted days will be assumed. In
example:
# reset = 10h (reset every 10 hours)
# reset = 12 (reset every 12 days)
#
# The check-name attribute defines an attribute which will
be
# registered by the counter module and can be used to set
the

```



```

# maximum allowed value for the counter after which the
user
# is rejected.
# Something like:
#
# DEFAULT Max-Daily-Session := 36000
#         Fall-Through = 1
#
# You should add the counter module in the instantiate
# section so that it registers check-name before the files
# module reads the users file.
#
# If check-name is set and the user is to be rejected then
we
# send back a Reply-Message and we log a Failure-Message in
# the radius.log
# If the count attribute is Acct-Session-Time then on each
login
# we send back the remaining online time as a Session-
Timeout attribute
#
# The counter-name can also be used instead of using the
check-name
# like below:
#
# DEFAULT Daily-Session-Time > 3600, Auth-Type = Reject
#         Reply-Message = "You've used up more than one hour
today"
#
# The allowed-servicetype attribute can be used to only
take
# into account specific sessions. For example if a user
first
# logs in through a login menu and then selects ppp there
will
# be two sessions. One for Login-User and one for Framed-
User
# service type. We only need to take into account the
second one.
#
# The module should be added in the instantiate, authorize
and
# accounting sections. Make sure that in the authorize
# section it comes after any module which sets the
# 'check-name' attribute.
#
counter daily {
    filename = ${raddbdir}/db.daily
    key = User-Name
    count-attribute = Acct-Session-Time
    reset = daily
    counter-name = Daily-Session-Time
    check-name = Max-Daily-Session
    allowed-servicetype = Framed-User

```

```

        cache-size = 5000
    }

    #
    # This module is an SQL enabled version of the counter
module.
    #
    # Rather than maintaining seperate (GDBM) databases of
    # accounting info for each counter, this module uses the
data
    # stored in the raddacct table by the sql modules. This
    # module NEVER does any database INSERTs or UPDATES. It is
    # totally dependent on the SQL module to process Accounting
    # packets.
    #
    # The 'sqlmod_inst' parameter holds the instance of the sql
    # module to use when querying the SQL database. Normally it
    # is just "sql". If you define more and one SQL module
    # instance (usually for failover situations), you can
    # specify which module has access to the Accounting Data
    # (radacct table).
    #
    # The 'reset' parameter defines when the counters are all
    # reset to zero. It can be hourly, daily, weekly, monthly
or
    # never. It can also be user defined. It should be of the
    # form:
    #   num[hdw] where:
    #   h: hours, d: days, w: weeks, m: months
    #   If the letter is ommited days will be assumed. In
example:
    #   reset = 10h (reset every 10 hours)
    #   reset = 12 (reset every 12 days)
    #
    # The 'key' parameter specifies the unique identifier for
the
    # counter records (usually 'User-Name').
    #
    # The 'query' parameter specifies the SQL query used to get
    # the current Counter value from the database. There are 3
    # parameters that can be used in the query:
    #       %k    'key' parameter
    #       %b    unix time value of beginning of reset period
    #       %e    unix time value of end of reset period
    #
    # The 'check-name' parameter is the name of the 'check'
    # attribute to use to access the counter in the 'users'
file
    # or SQL radcheck or radcheckgroup tables.
    #
    # DEFAULT Max-Daily-Session > 3600, Auth-Type = Reject
    # Reply-Message = "You've used up more than one hour
today"
    #

```

```

sqlcounter dailycounter {
    counter-name = Daily-Session-Time
    check-name = Max-Daily-Session
    sqlmod-inst = sql
    key = User-Name
    reset = daily

    # This query properly handles calls that span from the
    # previous reset period into the current period but
    # involves more work for the SQL server than those
    # below
    # For mysql:
    query = "SELECT SUM(AcctSessionTime - \
        GREATEST((%b - UNIX_TIMESTAMP(AcctStartTime)),
0)) \
        FROM radacct WHERE UserName='%{k}' AND \
        UNIX_TIMESTAMP(AcctStartTime) + AcctSessionTime
> '%b'"

    # For postgresql:
    query = "SELECT SUM(AcctSessionTime - \
        GREATER((%b - AcctStartTime::ABSTIME::INT4), 0))
\
        FROM radacct WHERE UserName='%{k}' AND \
        AcctStartTime::ABSTIME::INT4 + AcctSessionTime >
'%b'"

    # This query ignores calls that started in a previous
    # reset period and continue into into this one. But it
    # is a little easier on the SQL server
    # For mysql:
    query = "SELECT SUM(AcctSessionTime) FROM radacct
WHERE \
        UserName='%{k}' AND AcctStartTime >
FROM_UNIXTIME('%b') "

    # For postgresql:
    query = "SELECT SUM(AcctSessionTime) FROM radacct
WHERE \
        UserName='%{k}' AND AND
AcctStartTime::ABSTIME::INT4 > '%b'"

    # This query is the same as above, but demonstrates an
    # additional counter parameter '%e' which is the
    # timestamp for the end of the period
    # For mysql:
    query = "SELECT SUM(AcctSessionTime) FROM radacct \
        WHERE UserName='%{k}' AND AcctStartTime BETWEEN
\
        FROM_UNIXTIME('%b') AND FROM_UNIXTIME('%e') "

    # For postgresql:
    query = "SELECT SUM(AcctSessionTime) FROM radacct \

```

```

#                               WHERE UserName='%{k}' AND
AcctStartTime::ABSTIME::INT4 \
#                               BETWEEN '%b' AND '%e'"
    }

    sqlcounter monthlycounter {
        counter-name = Monthly-Session-Time
        check-name = Max-Monthly-Session
        sqlmod-inst = sql
        key = User-Name
        reset = monthly

        # This query properly handles calls that span from the
        # previous reset period into the current period but
        # involves more work for the SQL server than those
        # below
        # The same notes above about the differences between
mysql
        # versus postgres queries apply here.
        query = "SELECT SUM(AcctSessionTime - \
                GREATEST((%b - UNIX_TIMESTAMP(AcctStartTime)),
0)) \
                FROM radacct WHERE UserName='%{k}' AND \
                UNIX_TIMESTAMP(AcctStartTime) + AcctSessionTime
> '%b'"

        # This query ignores calls that started in a previous
        # reset period and continue into this one. But it
        # is a little easier on the SQL server
        #
        query = "SELECT SUM(AcctSessionTime) FROM radacct
WHERE \
#
        #                               UserName='%{k}' AND AcctStartTime >
FROM_UNIXTIME('%b') "

        # This query is the same as above, but demonstrates an
        # additional counter parameter '%e' which is the
        # timestamp for the end of the period
        #
        query = "SELECT SUM(AcctSessionTime) FROM radacct \
        #
        #                               WHERE UserName='%{k}' AND AcctStartTime BETWEEN
\
        #                               FROM_UNIXTIME('%b') AND FROM_UNIXTIME('%e') "
    }

    #
    # The "always" module is here for debugging purposes. Each
    # instance simply returns the same result, always, without
    # doing anything.
    always fail {
        rcode = fail
    }
    always reject {
        rcode = reject
    }
    always ok {

```

```

        rcode = ok
        simulcount = 0
        mpp = no
    }

#
# The 'expression' module currently has no configuration.
#
# This module is useful only for 'xlat'. To use it,
# put 'exec' into the 'instantiate' section. You can then
# do dynamic translation of attributes like:
#
# Attribute-Name = `%{expr:2 + 3 + %{exec: uid -u}}`
#
# The value of the attribute will be replaced with the
output
# of the program which is executed. Due to RADIUS protocol
# limitations, any output over 253 bytes will be ignored.
expr {
}

#
# The 'digest' module currently has no configuration.
#
# "Digest" authentication against a Cisco SIP server.
# See 'doc/rfc/draft-sterman-aaa-sip-00.txt' for details
# on performing digest authentication for Cisco SIP
servers.
#
digest {
}

#
# Execute external programs
#
# This module is useful only for 'xlat'. To use it,
# put 'exec' into the 'instantiate' section. You can then
# do dynamic translation of attributes like:
#
# Attribute-Name = `%{exec:/path/to/program args}`
#
# The value of the attribute will be replaced with the
output
# of the program which is executed. Due to RADIUS protocol
# limitations, any output over 253 bytes will be ignored.
#
# The RADIUS attributes from the user request will be
placed
# into environment variables of the executed program, as
# described in 'doc/variables.txt'
#
exec {
    wait = yes
    input_pairs = request

```

```

}

#
# This is a more general example of the execute module.
#
# This one is called "echo".
#
# Attribute-Name = `${echo:/path/to/program args}`
#
# If you wish to execute an external program in more than
# one section (e.g. 'authorize', 'pre_proxy', etc), then it
# is probably best to define a different instance of the
# 'exec' module for every section.
#
exec echo {
    #
    # Wait for the program to finish.
    #
    # If we do NOT wait, then the program is "fire and
    # forget", and any output attributes from it are
ignored.
    #
    # If we are looking for the program to output
    # attributes, and want to add those attributes to the
    # request, then we MUST wait for the program to
    # finish, and therefore set 'wait=yes'
    #
    # allowed values: {no, yes}
    wait = yes

    #
    # The name of the program to execute, and it's
    # arguments. Dynamic translation is done on this
    # field, so things like the following example will
    # work.
    #
    program = "/bin/echo ${User-Name}"

    #
    # The attributes which are placed into the
    # environment variables for the program.
    #
    # Allowed values are:
    #
    # request          attributes from the request
    # config           attributes from the configuration
items list
    # reply            attributes from the reply
    # proxy-request    attributes from the proxy request
    # proxy-reply      attributes from the proxy reply
    #
    # Note that some attributes may not exist at some
    # stages. e.g. There may be no proxy-reply
    # attributes if this module is used in the

```

```

# 'authorize' section.
#
input_pairs = request

#
# Where to place the output attributes (if any) from
# the executed program. The values allowed, and the
# restrictions as to availability, are the same as
# for the input_pairs.
#
output_pairs = reply

#
# When to execute the program. If the packet
# type does NOT match what's listed here, then
# the module does NOT execute the program.
#
# For a list of allowed packet types, see
# the 'dictionary' file, and look for VALUES
# of the Packet-Type attribute.
#
# By default, the module executes on ANY packet.
# Un-comment out the following line to tell the
# module to execute only if an Access-Accept is
# being sent to the NAS.
#
#packet_type = Access-Accept
}

# Do server side ip pool management. Should be added in
post-auth and
# accounting sections.
#
# The module also requires the existence of the Pool-Name
# attribute. That way the administrator can add the Pool-
Name
# attribute in the user profiles and use different pools
# for different users. The Pool-Name attribute is a *check*
item not
# a reply item.
#
# Example:
# radiusd.conf: ippool students { [...] }
# users file : DEFAULT Group == students, Pool-Name :=
"students"
#
# ***** IF YOU CHANGE THE RANGE PARAMETERS YOU MUST
*****
# ***** THEN ERASE THE DB FILES
*****
#
# ippool main_pool {

# range-start,range-stop: The start and end ip

```

```

# addresses for the ip pool
range-start = 192.168.1.1
range-stop = 192.168.3.254

# netmask: The network mask used for the ip's
netmask = 255.255.255.0

# cache-size: The gdbm cache size for the db
# files. Should be equal to the number of ip's
# available in the ip pool
cache-size = 800

# session-db: The main db file used to allocate ip's
to clients
session-db = ${raddbdir}/db.ippool

# ip-index: Helper db index file used in multilink
ip-index = ${raddbdir}/db.ipindex

# override: Will this ippool override a Framed-IP-
Address already set
override = no

# maximum-timeout: If not zero specifies the maximum
time in seconds an
# entry may be active. Default: 0
maximum-timeout = 0
}

# $INCLUDE ${confdir}/sqlippool.conf

# OTP token support. Not included by default.
# $INCLUDE ${confdir}/otp.conf

}

# Instantiation
#
# This section orders the loading of the modules. Modules
# listed here will get loaded BEFORE the later sections like
# authorize, authenticate, etc. get examined.
#
# This section is not strictly needed. When a section like
# authorize refers to a module, it's automatically loaded and
# initialized. However, some modules may not be listed in any
# of the following sections, so they can be listed here.
#
# Also, listing modules here ensures that you have control over
# the order in which they are initialized. If one module needs
# something defined by another module, you can list them in
order
# here, and ensure that the configuration will be OK.
#
instantiate {

```



```

#
# Allows the execution of external scripts.
# The entire command line (and output) must fit into 253
bytes.
#
# e.g. Framed-Pool = `%{exec:/bin/echo foo}`
exec

#
# The expression module doesn't do authorization,
# authentication, or accounting. It only does dynamic
# translation, of the form:
#
# Session-Timeout = `%{expr:2 + 3}`
#
# So the module needs to be instantiated, but CANNOT be
# listed in any other section. See 'doc/rlm_expr' for
# more information.
#
expr

#
# We add the counter module here so that it registers
# the check-name attribute before any module which sets
# it
#
daily
}

# Authorization. First preprocess (hints and huntgroups files),
# then realms, and finally look in the "users" file.
#
# The order of the realm modules will determine the order that
# we try to find a matching realm.
#
# Make *sure* that 'preprocess' comes before any realm if you
# need to setup hints for the remote radius server
authorize {
#
# The preprocess module takes care of sanitizing some
bizarre
# attributes in the request, and turning them into
attributes
# which are more standard.
#
# It takes care of processing the 'raddb/hints' and the
# 'raddb/huntgroups' files.
#
# It also adds the %{Client-IP-Address} attribute to the
request.
preprocess

#
# If you want to have a log of authentication requests,
# un-comment the following line, and the 'detail auth_log'

```

```

# section, above.
auth_log

# attr_filter

#
# The chap module will set 'Auth-Type := CHAP' if we are
# handling a CHAP request and Auth-Type has not already
been set
chap

#
# If the users are logging in with an MS-CHAP-Challenge
# attribute for authentication, the mschap module will find
# the MS-CHAP-Challenge attribute, and add 'Auth-Type :=
MS-CHAP'
# to the request, which will cause the server to then use
# the mschap module for authentication.
mschap

#
# If you have a Cisco SIP server authenticating against
# FreeRADIUS, uncomment the following line, and the
'digest'
# line in the 'authenticate' section.
# digest

#
# Look for IPASS style 'realm/', and if not found, look for
# '@realm', and decide whether or not to proxy, based on
# that.
# IPASS

#
# If you are using multiple kinds of realms, you probably
# want to set "ignore_null = yes" for all of them.
# Otherwise, when the first style of realm doesn't match,
# the other styles won't be checked.
#
suffix
# ntdomain

#
# This module takes care of EAP-MD5, EAP-TLS, and EAP-LEAP
# authentication.
#
# It also sets the EAP-Type attribute in the request
# attribute list to the EAP type from the packet.
eap

#
# Read the 'users' file
files

```

```

#
# Look in an SQL database. The schema of the database
# is meant to mirror the "users" file.
#
# See "Authorization Queries" in sql.conf
# sql

#
# If you are using /etc/smbpasswd, and are also doing
# mschap authentication, the un-comment this line, and
# configure the 'etc_smbpasswd' module, above.
# etc_smbpasswd

#
# The ldap module will set Auth-Type to LDAP if it has not
# already been set
# ldap

#
# Enforce daily limits on time spent logged in.
# daily

#
# Use the checkval module
# checkval
}

# Authentication.
#
#
# This section lists which modules are available for
authentication.
# Note that it does NOT mean 'try each module in order'. It
means
# that a module from the 'authorize' section adds a
configuration
# attribute 'Auth-Type := FOO'. That authentication type is
then
# used to pick the appropriate module from the list below.
#

# In general, you SHOULD NOT set the Auth-Type attribute. The
server
# will figure it out on its own, and will do the right thing.
The
# most common side effect of erroneously setting the Auth-Type
# attribute is that one authentication method will work, but the
# others will not.
#
# The common reasons to set the Auth-Type attribute by hand
# is to either forcibly reject the user, or forcibly accept him.
#
authenticate {

```

```

#
# PAP authentication, when a back-end database listed
# in the 'authorize' section supplies a password. The
# password can be clear-text, or encrypted.
Auth-Type PAP {
    pap
}

#
# Most people want CHAP authentication
# A back-end database listed in the 'authorize' section
# MUST supply a CLEAR TEXT password. Encrypted passwords
# won't work.
Auth-Type CHAP {
    chap
}

#
# MSCHAP authentication.
Auth-Type MS-CHAP {
    mschap
}

#
# If you have a Cisco SIP server authenticating against
# FreeRADIUS, uncomment the following line, and the
'digest'
# line in the 'authorize' section.
# digest

#
# Pluggable Authentication Modules.
# pam

#
# See 'man getpwent' for information on how the 'unix'
# module checks the users password. Note that packets
# containing CHAP-Password attributes CANNOT be
authenticated
# against /etc/passwd! See the FAQ for details.
#
unix

# Uncomment it if you want to use ldap for authentication
#
# Note that this means "check plain-text password against
# the ldap database", which means that EAP won't work,
# as it does not supply a plain-text password.
#
Auth-Type LDAP {
#
#     ldap
#
}

#
# Allow EAP authentication.

```

```

        eap
    }

#
# Pre-accounting.  Decide which accounting type to use.
#
preacct {
    preprocess

    #
    # Ensure that we have a semi-unique identifier for every
    # request, and many NAS boxes are broken.
    acct_unique

    #
    # Look for IPASS-style 'realm/', and if not found, look for
    # '@realm', and decide whether or not to proxy, based on
    # that.
    #
    # Accounting requests are generally proxied to the same
    # home server as authentication requests.
# IPASS
    suffix
# ntdomain

    #
    # Read the 'acct_users' file
    files
}

#
# Accounting.  Log the accounting data.
#
accounting {
    #
    # Create a 'detail'ed log of the packets.
    # Note that accounting requests which are proxied
    # are also logged in the detail file.
    detail
# daily

    # Update the wtmp file
    #
    # If you don't use "radlast", you can delete this line.
    unix

    #
    # For Simultaneous-Use tracking.
    #
    # Due to packet losses in the network, the data here
    # may be incorrect.  There is little we can do about it.
    radutmp
# sradutmp

```

```

        # Return an address to the IP Pool when we see a stop
record.
#    main_pool

#
# Log traffic to an SQL database.
#
# See "Accounting queries" in sql.conf
#    sql

#
# Instead of sending the query to the SQL server,
# write it into a log file.
#
#    sql_log

# Cisco VoIP specific bulk accounting
#    pgsql-voip
}

# Session database, used for checking Simultaneous-Use. Either
the radutmp
# or rlm_sql module can handle this.
# The rlm_sql module is *much* faster
session {
    radutmp

#
# See "Simultaneous Use Checking Querie" in sql.conf
#    sql
}

# Post-Authentication
# Once we KNOW that the user has been authenticated, there are
# additional steps we can take.
post-auth {
    # Get an address from the IP Pool.
#    main_pool

#
# If you want to have a log of authentication replies,
# un-comment the following line, and the 'detail reply_log'
# section, above.
    reply_log

#
# After authenticating the user, do another SQL query.
#
# See "Authentication Logging Queries" in sql.conf
#    sql

```

```

#
# Instead of sending the query to the SQL server,
# write it into a log file.
#
# sql_log

#
# Un-comment the following if you have set
# 'edir_account_policy_check = yes' in the ldap module sub-
section of
# the 'modules' section.
#
# ldap
#
# Access-Reject packets are sent through the REJECT sub-
section of the
# post-auth section.
# Uncomment the following and set the module name to the
ldap instance
# name if you have set 'edir_account_policy_check = yes' in
the ldap
# module sub-section of the 'modules' section.
#
# Post-Auth-Type REJECT {
#     insert-module-name-here
# }

}

#
# When the server decides to proxy a request to a home server,
# the proxied request is first passed through the pre-proxy
# stage. This stage can re-write the request, or decide to
# cancel the proxy.
#
# Only a few modules currently have this method.
#
pre-proxy {
#     attr_rewrite

#     Uncomment the following line if you want to change
attributes
#     as defined in the preproxy_users file.
#     files

#     If you want to have a log of packets proxied to a home
#     server, un-comment the following line, and the
#     'detail pre_proxy_log' section, above.
#     pre_proxy_log
}

#
# When the server receives a reply to a request it proxied

```

```

# to a home server, the request may be massaged here, in the
# post-proxy stage.
#
post-proxy {

    # If you want to have a log of replies from a home server,
    # un-comment the following line, and the 'detail
post_proxy_log'
    # section, above.
# post_proxy_log

# attr_rewrite

    # Uncomment the following line if you want to filter
replies from
    # remote proxies based on the rules defined in the 'attrs'
file.

# attr_filter

#
# If you are proxying LEAP, you MUST configure the EAP
# module, and you MUST list it here, in the post-proxy
# stage.
#
# You MUST also use the 'nostrip' option in the 'realm'
# configuration. Otherwise, the User-Name attribute
# in the proxied request will not match the user name
# hidden inside of the EAP packet, and the end server will
# reject the EAP request.
#
eap
}

```

Lampiran B

Autentikasi FreeRADIUS berhasil

```

rad_recv: Access-Request packet from host 172.22.3.10:33420,
id=129, length=205
  NAS-Port-Type = Wireless-802.11
  Calling-Station-Id = "00:13:E8:0C:90:0D"
  Called-Station-Id = "wireless"
  NAS-Port-Id = "ether2"
  User-Name = "00:13:E8:0C:90:0D"
  NAS-Port = 2154823815
  Acct-Session-Id = "80700087"
  Framed-IP-Address = 10.1.0.169
  Mikrotik-Attr-10 = 0x0a010138
  CHAP-Challenge = 0x5484b26465b8dfa04e6b8838444f2fc9
  CHAP-Password = 0xbcdfae3611a602c1727441b27f18a78e64
  Service-Type = Login-User
  WISPr-Logoff-URL = "http://10.1.0.1/logout"
  NAS-Identifier = "MikroTik"

```



```

        NAS-IP-Address = 172.22.3.10
    Processing the authorize section of radiusd.conf
modcall: entering group authorize for request 0
    modcall[authorize]: module "preprocess" returns ok for request
    0
radius_xlat:  '/var/log/radius/radacct/172.22.3.10/auth-detail-
20090819'
rlm_detail: /var/log/radius/radacct/%{Client-IP-Address}/auth-
detail-%Y%m%d expands to
/var/log/radius/radacct/172.22.3.10/auth-detail-20090819
    modcall[authorize]: module "auth_log" returns ok for request 0
    rlm_chap: Setting 'Auth-Type := CHAP'
    modcall[authorize]: module "chap" returns ok for request 0
    modcall[authorize]: module "mschap" returns noop for request 0
    rlm_realm: No '@' in User-Name = "00:13:E8:0C:90:0D", looking
up realm NULL
    rlm_realm: No such realm "NULL"
    modcall[authorize]: module "suffix" returns noop for request 0
    rlm_eap: No EAP-Message, not doing EAP
    modcall[authorize]: module "eap" returns noop for request 0
    users: Matched entry 00:13:E8:0C:90:0D at line 2
    modcall[authorize]: module "files" returns ok for request 0
modcall: leaving group authorize (returns ok) for request 0
    rad_check_password:  Found Auth-Type CHAP
auth: type "CHAP"
    Processing the authenticate section of radiusd.conf
modcall: entering group CHAP for request 0
    rlm_chap: login attempt by "00:13:E8:0C:90:0D" with CHAP
password
    rlm_chap: Using clear text password  for user 00:13:E8:0C:90:0D
authentication.
    rlm_chap: chap user 00:13:E8:0C:90:0D authenticated succesfully
    modcall[authenticate]: module "chap" returns ok for request 0
modcall: leaving group CHAP (returns ok) for request 0
Login OK: [00:13:E8:0C:90:0D] (from client local port 2154823815
cli 00:13:E8:0C:90:0D)
    Processing the post-auth section of radiusd.conf
modcall: entering group post-auth for request 0
radius_xlat:  '/var/log/radius/radacct/172.22.3.10/reply-detail-
20090819'
rlm_detail: /var/log/radius/radacct/%{Client-IP-Address}/reply-
detail-%Y%m%d expands to
/var/log/radius/radacct/172.22.3.10/reply-detail-20090819
    modcall[post-auth]: module "reply_log" returns ok for request 0
modcall: leaving group post-auth (returns ok) for request 0
Sending Access-Accept of id 129 to 172.22.3.10 port 33420
Finished request 0
Going to the next request
--- Walking the entire request list ---
Waking up in 6 seconds...
--- Walking the entire request list ---
Cleaning up request 0 ID 129 with timestamp 4a8bb2c5
Nothing to do.  Sleeping until we see a request.

```

Lampiran C

Autentikasi FreeRADIUS gagal

```
rad_recv: Access-Request packet from host 172.22.3.10:33420,
id=130, length=205
    NAS-Port-Type = Wireless-802.11
    Calling-Station-Id = "00:1E:E5:9D:64:B2"
    Called-Station-Id = "wireless"
    NAS-Port-Id = "ether2"
    User-Name = "00:1E:E5:9D:64:B2"
    NAS-Port = 2154823816
    Acct-Session-Id = "80700088"
    Framed-IP-Address = 10.1.0.189
    Mikrotik-Attr-10 = 0x0a010006
    CHAP-Challenge = 0x7a424321e7cdade16f36861873bb56c2
    CHAP-Password = 0x907c04346acd2280b8a8f0b2df4458caa7
    Service-Type = Login-User
    WISPr-Logoff-URL = "http://10.1.0.1/logout"
    NAS-Identifier = "MikroTik"
    NAS-IP-Address = 172.22.3.10
    Processing the authorize section of radiusd.conf
modcall: entering group authorize for request 0
    modcall[authorize]: module "preprocess" returns ok for request
0
radius_xlat: '/var/log/radius/radacct/172.22.3.10/auth-detail-
20090819'
rlm_detail: /var/log/radius/radacct/%{Client-IP-Address}/auth-
detail-%Y%m%d expands to
/var/log/radius/radacct/172.22.3.10/auth-detail-20090819
    modcall[authorize]: module "auth_log" returns ok for request 0
    rlm_chap: Setting 'Auth-Type := CHAP'
    modcall[authorize]: module "chap" returns ok for request 0
    modcall[authorize]: module "mschap" returns noop for request 0
    rlm_realm: No '@' in User-Name = "00:1E:E5:9D:64:B2", looking
up realm NULL
    rlm_realm: No such realm "NULL"
    modcall[authorize]: module "suffix" returns noop for request 0
    rlm_eap: No EAP-Message, not doing EAP
    modcall[authorize]: module "eap" returns noop for request 0
    users: Matched entry DEFAULT at line 158
    modcall[authorize]: module "files" returns ok for request 0
modcall: leaving group authorize (returns ok) for request 0
    rad_check_password: Found Auth-Type CHAP
auth: type "CHAP"
    Processing the authenticate section of radiusd.conf
modcall: entering group CHAP for request 0
    rlm_chap: login attempt by "00:1E:E5:9D:64:B2" with CHAP
password
    rlm_chap: Could not find clear text password for user
00:1E:E5:9D:64:B2
    modcall[authenticate]: module "chap" returns invalid for
request 0
```

```

modcall: leaving group CHAP (returns invalid) for request 0
auth: Failed to validate the user.
Login incorrect (rlm_chap: Clear text password not available):
[00:1E:E5:9D:64:B2] (from client local port 2154823816 cli
00:1E:E5:9D:64:B2)
Delaying request 0 for 1 seconds
Finished request 0
Going to the next request
--- Walking the entire request list ---
Waking up in 1 seconds...
--- Walking the entire request list ---
Waking up in 1 seconds...
rad_recv: Access-Request packet from host 172.22.3.10:33420,
id=130, length=205
Sending Access-Reject of id 130 to 172.22.3.10 port 33420
Waking up in 1 seconds...
--- Walking the entire request list ---
Waking up in 4 seconds...
--- Walking the entire request list ---
Cleaning up request 0 ID 130 with timestamp 4a8bb492
Nothing to do. Sleeping until we see a request.

```

Lampiran D

Daftar File pada FreeRADIUS

```

-rw-r--r-- 1 root root 422 Aug 19 05:46 acct_users
-rw-r--r-- 1 root root 4074 Aug 19 05:46 attrs
drwxr-xr-x 3 root root 4096 Aug 19 05:46 certs
-rw-r----- 1 root root 189 Aug 19 05:46 clients
-rw-r----- 1 root root 2921 Aug 19 05:46 clients.conf
-rw-r--r-- 1 root root 929 Aug 19 05:46 dictionary
-rw-r--r-- 1 root root 9985 Aug 19 05:46 eap.conf
-rwxr-xr-x 1 root root 4620 Aug 19 05:46 example.pl
-rw-r--r-- 1 root root 2396 Aug 19 05:46 hints
-rw-r--r-- 1 root root 1604 Aug 19 05:46 huntgroups
-rw-r--r-- 1 root root 2439 Aug 19 05:46 ldap.attrmap
-rw-r--r-- 1 root root 1020 Aug 19 06:47 naslist
-rw-r----- 1 root root 856 Aug 19 05:46 naspasswd
-rw-r--r-- 1 root root 3358 Aug 19 05:46 otp.conf
-rw-r--r-- 1 root root 1734 Aug 19 05:46 otppasswd.sample
-rw-r--r-- 1 root root 1039 Aug 19 05:46 preproxy_users
-rw-r--r-- 1 root root 8834 Aug 19 05:46 proxy.conf
-rw-r--r-- 1 root root 66080 Aug 19 05:46 radiusd.conf
-rw-r--r-- 1 root root 187 Aug 19 05:46 realms
-rw-r--r-- 1 root root 1405 Aug 19 05:46 snmp.conf
-rw-r--r-- 1 root root 3329 Aug 19 05:46 sqlippool.conf
-rw-r--r-- 1 root root 7210 Aug 19 15:09 users

```

Lampiran E

Daftar MAC *address* yang terdaftar pada pengujian /etc/raddb/users

"00:1B:77:48:0A:2B"	User-Password == ""
"00:13:E8:0C:90:0D"	User-Password == ""
"00:16:76:39:05:1A"	User-Password == ""
"00:1E:E5:9D:64:B2"	User-Password == ""
"00:1B:77:55:ED:14"	User-Password == ""
"00:1A:73:10:69:E7"	User-Password == ""
"00:21:00:68:82:13"	User-Password == ""
"00:1E:4C:AC:75:8B"	User-Password == ""