BAB I PENDAHULUAN

1.1 Latar Belakang

Pesatnya dunia jaringan komputer membawa dampak yang tidak sederhana. Perkembangan dunia jaringan komputer ini juga membawa dampak negatif dalam beberapa hal. Salah satunya dampak negatif tersebut adalah semakin kompleksnya pengelolaan sistem-sistem yang terdapat dalam jaringan, khususnya pada permasalahan pengelolaan *user*. Penambahan *user* baru merupakan pekerjaan yang masih tergolong relatif sederhana ketika jaringan hanya terdiri dari lima server atau bahkan kurang dari jumlah itu. Masing-masing server dapat dengan mudah saling terhubung, melakukan penambahan akun *user* baru dengan tetap menjaga konsistensinya, mengatur password, dan notifikasi *user*. Akan tetapi proses-proses ini menjadi tidak mudah ketika lingkungan jaringan berkembang menjadi sepuluh, lima puluh, seratus server atau bahkan lebih. Pengelolaan jaringan akan semakin sulit karena banyaknya aplikasi-aplikasi yang diimplementasikan dalam jaringan. Semakin banyak sistem dan aplikasinya maka semakin banyak mekanisme login yang diberlakukan. *User* akan disibukkan dengan mengingat banyak kombinasi *userid* dan *password*, atau disulitkan dengan mekanisme login yang berulang-ulang (Garman,2002).

Sistem direktori terpusat telah banyak dimanfaatkan sebagai solusi dari persoalan ini. Sistem direktori terpusat ini menawarkan SingleID dan dapat dimanfaatkan untuk keperluan *single sign on* sehingga pengelolaan jaringan besar menjadi lebih mudah (Christian, 2004).

Single sign on telah banyak dibicarakan sebagai solusi dalam pengelolaan login dalam jumlah besar. Hampir sebagaian besar *user* pada sistem IT yang modern

seperti sekarang ini, akan menghadapi metode *multiple login*. Metode *multiple login* banyak memberikan dampak negatif bagi pihak yang mengimplementasikan. Resiko *multiple login* dari sisi waktu yang dihabiskan yaitu, ketika harus melakukan login berulang-ulang dibandingkan dengan metode single login. Aspek lain yang tidak kalah penting adalah dari segi biaya. Biaya yang dimaksud disini adalah biaya yang dihabiskan ketika *user* menghubungi *help desk* saat *user* lupa akan kombinasi userid dan password miliknya. Selain kedua aspek tersebut, aspek berikutnya adalah dari segi keamanan penyimpanan password. Beberapa *user* memilih untuk menuliskan password pada kertas atau media lainnya yang justru tidak aman karena dapat diketahui pihak lain selain *user* yang bersangkutan (Christian, 2004).

Konsep dasar dari adalah bagaimana agar *user* hanya perlu melakukan login sekali dan untuk kemudian computer akan menggantikan perannya pada login-login berikutnya secara otomatis atas nama *user* tersebut. Senada dengan konsep *Single sign on*, Single ID juga merupakan suatu konsep yang bertujuan memudahkan pengelolaan administrasi jaringan. Single ID ini merupakan konsep dari identitas tunggal untuk banyak aplikasi. Beban *user* untuk mengingat bermacam-macam kombinasi userid dan password dapat dikurangi atau bahkan dieliminasi.

Selain itu, dengan adanya direktori terpusat ini dapat dikembangkan suatu system terdistribusi, baik aplikasi direktori terdistribusi, maupun system terdistribusi lainnya. Salah satu teknologi yang ada untuk keperluan direktori terpusat yaitu *Lightweight Directory Access Protocol* (LDAP). LDAP merupakan protocol yang ditujukan untuk pengaksesan direktori (Wahl,1997).

Lightweight Directory Access Protocol (LDAP) telah banyak dimanfaatkan untuk penggunaan aplikasi-aplikasi direktori yang semakin meningkat seperti *mail server* dan *remote login*. Direktori LDAP digunakan untuk penyimpanan informasi

personal dan aturan-aturan otentikasi. Data yang disimpan bersifat statis sehingga cache bisa digunakan untuk peningkatan kinerja (Uli, 2000).

Pengaksesan direktori menggunakan LDAP memiliki kelemahan dalam sistem keamanannya. Banyak implementasi LDAP tanpa memperhatikan keamanan saat pengaksesan data. Data seperti misalnya userid dan password masih ditranmisikan melalui jaringan pada saat pengaksesan direktori. Hal ini menjadi lubang keamanan yang tidak bisa diremehkan mengingat banyaknya metode kejahatan cyber di era sekarang.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, dapat diambil rumusan yang menjadi permasalah utama dalam tugas akhir ini yaitu :

1. Bagaimana membuat sistem direktori terpusat dengan menggunakan LDAP sehingga dapat dimanfaatkan untuk keperluan *single sign on* sehingga pengelolaan jaringan besar menjadi lebih mudah.
2. Bagaimana membuat sebuah sistem otentikasi yang memberikan keamanan saat pengaksesan direktori LDAP.
3. Menutupi kelemahan dari system otentikasi direktori LDAP dengan menggunakan protokol Kerberos

1.3 Batasan Masalah

Berdasarkan rumusan masalah yang telah disebutkan sebelumnya, maka batasan yang diberikan untuk rumusan masalah diatas yaitu :

1. Proses otentikasi, penyimpanan dokumen dan proses backup dilakukan secara terpusat dalam server LDAP.

2. Pengujian sistem otentikasi Kerberos dengan GSSAPI, sebagai mekanisme yang digunakan dalam protokol Kerberos.
3. Dalam penelitian ini hanya akan diadakan sebuah master KDC (*Key distribution center*) tanpa mengadakan replikasinya.

1.4 Tujuan Penelitian

Tujuan dalam melakukan penelitian ini adalah mengembangkan sistem otentikasi yang aman bagi pengaksesan direktori LDAP. Tujuan berikutnya dari penelitian ini yaitu menguji sistem otentikasi LDAP yang menggunakan protokol Kerberos dengan sistem otentikasi LDAP tanpa adanya Kerberos.

1.5 Manfaat Penelitian

Penetian ini diharapkan akan mampu memberikan manfaat, diantaranya :

1. Memberikan kemudahan otentikasi bagi *user* yang mengakses direktori LDAP.
2. Memberikan kemudahan kepada admin jaringan dalam pengelolaan sistem yang menggunakan LDAP sebagai backend jaringannya.
3. Menawarkan keamanan selama pengaksesan sumber informasi yang tersimpan dalam direktori LDAP.

1.6 Metode Penelitian

1.6.1 Studi Pustaka

Pada penelitian ini, langkah-langkah penyelesaian diawali dengan studi pustaka. Studi pustaka, yaitu pengumpulan literatur melalui buku- buku dan sumber lain yang berkaitan dengan penelitian yang dilakukan dalam rangka mencari konsep dasar dari penelitian ini.

Selain itu, langkah ini juga didapat dari pengumpulan literatur melalui internet, yang dilakukan dengan tujuan pemilihan perangkat lunak yang akan diimplementasikan sebagai Sistem *Authentikasi* Kerberos, serta pencarian referensi mengenai tahap dalam implementasi, termasuk kebutuhan *hardware* serta *software* pendukung lainnya agar sistem berjalan dengan baik. Kemudian membahas tentang pengembangan sistem otentikasi menggunakan sistem otentikasi LDAP yang menggunakan protokol Kerberos dengan sistem otentikasi LDAP tanpa adanya Kerberos.

1.6.2 Implementasi Perangkat Lunak

Implementasi dilakukan berdasarkan hasil dari langkah pengumpulan literatur yang meliputi :

1. Desain Arsitektur Jaringan dan Alokasi *Account*

   Tahap ini merupakan tahap perancangan arsitektur jaringan komputer yang akan digunakan untuk membangun Sistem *Authentikasi* Kerberos serta pengalokasian *account* pada directory LDAP.

2. Pengadaan *Hardware*

   Tahapan ini merupakan tahap pengadaan perangkat keras untuk Sistem *Authentikasi* server, serta komputer yang menjadi *client* dari Sistem *Authentikasi* Kerberos tersebut. Selain itu diperlukan pula perangkat keras yang digunakan sebagai pendukung keberlangsungan transmisi data *client*.

3. Instalasi dan Konfigurasi *Software*

   Tahapan ini merupakan tahap instalasi *software* pada komputer yang akan dijadikan Sistem *Authentikasi* Kerberos, serta komputer yang akan dijadikan *client*. Kemudian dilanjutkan dengan melakukan konfigurasi *software* pada Sistem *Authentikasi* Kerberos serta *client-client*.

4. Pengujian

   Setelah konfigurasi selesai dilakukan, tahap selanjutnya adalah pengujian kinerja Sistem *Authentikasi* Kerberos tersebut dalam menghadapi adanya penggunaan *account* oleh pihak yang tidak memiliki hak akses. Protokol Kerberos melakukan otentikasi dengan menggunakan *ticket* yang dikeluarkan oleh pihak ketiga dan *authenticator* yang dibuat sendiri oleh *client*. Pada tahap ini, dilakukan uji coba Sistem *Authentikasi* yang menggunkan Kerberos dan Sistem *Authentikasi* yang tidak menggunakan Kerberos

   Dengan dibangunnya protokol Kerberos dapat mengetahui perbedaan keamanan yang menggunakan Kerberos ataupun tidak.

1.7 Sistematika Penulisan

Penulisan laporan tugas akhir akan disusun dalam tujuh bab. Berguna untuk memberikan gambaran tentang permasalahan yang akan dibahas, dibawah ini penulis uraikan sistematika penulisan tugas akhir ini nantinya :

BAB I	PENDAHULUAN Bab ini berisi latar belakang masalah, perumusan masalah, batasan masalah, tujuan dan penelitian, metodelogi penelitian, dan sistematika penulisan.
BAB II	TINJAUAN PUSTAKA Bab ini berisi penjelasan yang memuat uraian sistematis tentang informasi hasil penelitian *Single sign on* yang memanfaatkan LDAP, kemudian penelitian mengenai protokol LDAP yang digunakan sebagai *backend* manajemen suatu jaringan Wi-Fi. Pada bab ini diuraikan pula perbedaan-

	perbedaan antara penelitian-penelitian sejenis sebelumnya dengan penelitian yang dilakukan kali ini.
BAB III	LANDASAN TEORI Bab ini berisi uraian mengenai landasan teori yang harus dipahami sebelum membahas bagian inti dari tugas akhir, yaitu mengenai perbedaan konsep otentikasi dan otorisasi, hal-hal yang berkaitan dengan protokol Kerberos, dan teori-teori tentang LDAP. Sub bab otentikasi dan otorisasi menguraikan definisi otentikasi dan otorisasi, serta menguraikan perbedaan-perbedaan yang mendasar diantara dua konsep tersebut. Sub bab Kerberos memberikan penjelasan mengenai dasar, terminology dan konsep kerja Kerberos. Sub bab LDAP memberikan penjelasan tentang dasar, skema dan konsep kerja LDAP.
BAB IV	ANALISIS DAN PERANCANGAN SISTEM Bab ini menjelaskan tentang perancangan arsitektur dalam sistem otentikasi LDAP menggunakan Kerberos yang akan diimplementasikan, kemudian menerangkan peran Kerberos bagi LDAP serta modul-modul SASL GSSAPI yang akan digunakan untuk keperluan pengimplementasian Kerberos pada direktori LDAP.
BAB V	IMPLEMENTASI Bab ini berisi penjelasan mengenai instalasi dan konfigurasi pada Kerberos dan LDAP berserta modul-modul SASL GSSAPI yang diperlukan untuk mengintegrasikan keduannya.

BAB 2 LANDASAN TEORI

2.1 Otentikasi dan Otorisasi

Otentikasi *(authentication)* menurut Arkills (2003) adalah jaminan kepada suatu entitas bahwa entitas yang lain merupakan dia seperti yang diklaimnya, sedangkan otorisasi *(authorizotion)* lebih menekankan apa saja yang boleh dilihat dan dilakukan oleh suatu identitas.

Banyak sistem otentikasi yang juga menawarkankan fungsi otorisasi, schingga kedua konsep ini kadang membingungkan. Sebagai contoh pada sistem Unix tradisional file /etc/passwd berisi data otentikasi dan otorisasi. Lebih buruk lagi dengan adanya fungsi direktori. Oleh karena itulah tidak mudah ketika harus membedakan antara konsep-konsep tersebut. Akan tetapi pada intinya otentikasi adalah proses membuktikan siapa yang mengakses, sedangka otorisasi memperhatikan apa saja yang boleh dilakukan oleh *user* yang telah diotentikasi tadi (Arkills, 2003)

Kusdrianto dan Raharjo (2005) menyebutkan bahwa secara umum akses terhadap sumber daya membutuhkan dua hal, otentikasi dan otorisasi. Otentikasi adalah mekanisme untuk mengenali pengguna atau sumber daya. Otentikasi dilakukan dengan tiga hal : (1) sesuatu yang dimiliki, (2) sesuatu yang diketahui, (3) sesutau yang melekat (pada *user*). Contoh dari ketiga hal tersebut antara lain :

1. Sesuatu yang dimiliki : kunci, kartu tanda pengenal, kartu ATM, smartcard, session key, token, digital certificate.
2. Sesuatu yang diketahui : userid dan password, PIN, pass phrase.

3. Sesuatu yang melekat pada seseorang : sidik jari, wajah, retina mata, DNA.

Otorisasi adalah mekanisme untuk memberikan ijin untuk mengakses sumber daya. Biasanya ini terkait dengan klasifikasi sumber daya dan *access control*. Jadi otorisasi mengatakan siapa boleh melakukan apa.

2.2 Kerberos

Kata Kerberos bermula dari mitologi Yunani yang berisi legenda Cerberus. Cerberus adalah penjaga alam neraka yang dikuasai oleh Hades dan istrinya Persephone. Bentuk dari Cerberus memiliki lima kepala, sedangkan Apollodorus menggambarkannya sebagai suatu makhluk yang aneh dengan tiga kepala berbentuk anjing dengan ekor naga. Cerberus sering digambarkan dengan suatu makhluk kejam berkepala tiga. Orang-orang Yunani percaya ketika seseorang meninggal dunia, rohnya akan dikirim ke Hades dan kekal selamnya disana. Bagi orang-orang yang tidak baik selama hidupnya akan menjalani hukuman berat disana. Cerberus sebagai penjaga gerbang Hades, menjamin hanya roh orang-orang yang meninggal yang dapat memasuki wilayah Hades, juga menjamin bahwa roh-roh yang telah masuk kedalamnya tidak akan pernah dapat keluar lagi (Garman, 2003)

Sebagai penjaga gerbang Hades, Cerberus mengotentikasikan pihak ketiga yang dipercaya bersama-sama (*trusted third-party*). Protokol ini menawarkan otentikasi pada jaringan yang tidak aman. Kerberus Versi 1 sampai 3 digunakan secara internal dalam Project Athena, sedangkan Versi 4 diperuntukkan untuk digunakan secara umum. Oleh karena lingkungan yang berbeda dari Project Athena, maka dibuatlah Versi 5 yang merupakan perbaikan dari Versi 4 tetapi tidak sepenuhnya saling mendukung. Kini yang disebut standar protokol Kerberos adalah Kerberos Versi 5. Selain versi gratisnya oleh MIT, Kerberos tersedia juga dalam bentuk versi komesial misalnya oleh Microsoft dan Sun (Christian, 2004).

Kerberos merupakan salah satu sistem yang menggunakan kriptografi kunci simetri 3DES dan *Advanced Encryption Standart* untuk menjaga informasi yang penting pada jaringan terbuka (Freitag, 2004).

Menurut Garman (2003), tujuan dari sistem Kerberos adalah meningkatkan keamanan dan kenyamanan sekaligus. Kerberos memberikan layanan-layanan yang didefinisikan seperti berikut :

1. Aman
   Kerberos tidak pernah melewatkan *password* dalam bentuk aslinya dalam jaringan.
2. *Single sign on*
   User hanya perlu login sekali untuk mengakses semua sumber daya yang mendukung Kerberos.
3. *Trusted third-party*
   Kerberos menggunakan *centralized authentication server* sebagai pihak ketiga yang dipercaya oleh semua sistem dalam jaringan.
4. *Mutual authentication*
   Dengan adanya *mutual authentication*, bukan hanya *user* yang meyakinkan server bahwa dia adalah *user* yang sebenarnya, tetapi *user* juga dapat percaya bahwa server yang dia hadapi adalah server yang diklaim sebagai server sesungguhnya.

2.2.1 *Principal*, *Instance*, dan *Realm* dari Kerberos

Garman (2003) juga menjelaskan mengenai *realm, principal, dan instance* Kerberos. *Principal* merepresentasikan *user* atau layanan jaringan dari suatu host. Masing-masing *principal* memiliki nama yang unik. Setiap *principal* dimulai dengan *username atau service name. Username dan service name* ini kemudian diikuti oleh *instance* yang sifatnya opsional. *Instance* ini digunakan dalam dua keadaan : untuk

service *principal* dan untuk membuat *principal* khusus yang dimasukkan dalam administrasi. Sebagai contoh, administrator dapat memiliki dua *principal* : satu, *principal* yang digunakan sehari-hari, dan satu lagi *principal* admin digunakan untuk meningkatkan hak akses.

Username dan *instance* disatukan dalam *realm* sebagai bentuk identitas yang bersifat unik. Kerberos mendefinisikannya sebagai *realm*. Melalui sebuah konvensi, nama *realm* Kerberos diberikan dari nama DNS yang diubah kedalam bentuk huruf kapital. Misalnya Wedgie International yang memiliki domain wedgie.org akan diberikan nama *realm* WEDGIE.ORG.

Walaupun konvensi menetapkan nama *realm* sama dengan nama domain, akan tetapi tetap diperbolehkan apabila hendak menggunakan nama yang lain. Harus diperhatikan pula bahwa nama *realm* bersifat case-sensitif, sehingga wedgie.org tidak sama dengan WEDGIE.ORG

Kasus lainnya menentukan *principal* yang diberikan kepada John Doe, yang bekerja di IT Department di Widgie International. *Principal* yang bisa diberikan jdoe/it.WEDGIE.ORG.

Ini adalah bentuk sederhana dari *principal* yang bisa digunakan, dan merupakan *principal* yang valid baik di Kerberos 4 maupun Kerberos 5. *Principal* tersebut merepresentasikan username dari jdoe, tanpa *instance*, dan sebuah *realm* IT.WEDGIE.ORG.

Mutual authentication di Kerberos mengijinkan otentikasi di kedua belah pihak. Oleh sebab itu bukan hanya *user* yang diberikan *principal*, tetapi host dan server juga. Service *principal* sedikit berbeda dengan *user principal*. Komponen username dalam service *principal* merepresentasikan nama service. Pada host *principal*, komponen username menunjukkan host.

Principal dalam Kerberos 4 tersusun oleh tiga komponen yaitu username, *instance*, dan *realm*. Seperti contoh *user principal* dari John Doe menjadi jdoe.admin@IT.WEDGIE.ORG

Contoh service *principal* dari host unixsvr.it.doesystem.com di *realm* IT.WEDGIE.ORG menjadi host.unixsvr@IT.WEDGIE.ORG.

Service *principal* di Kerberos 4 hanya disusun dati hostname dari server dan tidak ada komponen domain.

Format dari *principal* pada Kerberos 4 adalah sebagai berikut :

user[.*instance*]@*REALM*

service.hostname@REALM

Principal dalam Kerberos 5 memiliki komponen dasar yang sama seperti Kerberos 4. Sedikit perbedaan dalam komponen *instance* dimana pada Kerberos 5 dapat berisi beberapa komponen sub-*instance*. Selain itu *principal* dalam Kerberos 5 tidak menggunakan dot(.) untuk memisahkan antara komponen username dan *instance*, tetapi menggunakan forward slash (/) (Garman, 2003).

Contoh *user principal* jdoe.admin@IT.WEDGIE.ORG ini sama dengan contoh pertama pada Kerberos 4 pada sub bab sebelumnya yang menunjukkan format *principal* John Doe dengan *instance* admin.

Pada Kerberos 5, baik host maupun service *principal* memasukkan Fully *Qualified Domain Name* (FQDN) dari host dimana service terinstal. Dengan menggabungkan FQDN pada *principal*, administrator dapat memiliki lebih dari satu host dengan hostname yang sama tetapi domain yang berbeda pada *realm* yang sama. Host/unixsvr.it.wedgie.org@it .wedgie.org merupakan host unixsvr.it.wedgie.org pada *realm* it.wedgie.org.

Format *principal* Kerberos 5 secara umum yaitu component[/component][/component]...*@REALM*.

Ada dua tipe *principal* Kerberos 5 yang bisa digunakan. Tipe pertama yang biasa digunakan yaitu username [*/instance*]*@REALM*, sedangkan tipe berikutnya adalah service/fully-qualified-domain-name*@REALM*.

2.2.2 *Key distribution center* (KDC)

Masih oleh Garman (2003), *Key distribution center* atau disingkat KDC merupakan bagian penting dari sistem Kerberos. KDC terdiri dari tiga komponen logika yaitu database *principal* beserta kunci enkripsinya, *Authentication Server, dan Ticket Granting Server.*

Suatu *realm* harus memiliki paling tidak sebuah KDC. Sebaiknya mesin yang bertindak sebagai KDC terpisah dari mesin tempat diletakkannya aplikasi atau *service* yang ditawarkan untuk alasan keamanan. Lebih baik lagi menempatkan KDC pada tempat yang aman secara fisik.

Masing-masing KDC berisi database dari seluruh *principal* pada suatu *realm*. Sebagian besar *software* KDC juga menyimpan informasi tambahan, misalnya *password* lifetime, last password change, dan yang lainnya.

Hal lain yang tak kalah pentingnya yaitu konsep tiket dalam Kerberos. Secara konseptual, tiket Kerberos adalah struktur data terenkripsi yang dikeluarkan oleh KDC. Tujuan dari tiket ini yang pertama yaitu untuk memastikan kebenaran dari *end participant*. Kedua adalah untuk membangun kunci enkripsi sementara, diantara dua pihak sehingga tercipta komunikasi yang lebih aman.

2.3 LDAP

2.3.1 Pengenalan Direktori

Buku karangan Arkill(2003) yang berjudul *LDAP Directories Explained : An Introduction and Analysis* menjelaskan bahwa sebuah direktori merupakan cara yang lebih efisien untuk mencari dan mengatur informasi. Jika terdapat banyak sumber informasi untuk dicari, dimungkinkan adanya komunikasi antara satu sumber dengan sumber yang lain atau adanya informasi yang belum ter-*update*. Lebih dari itu, berpindah dari satu direktori ke direktori lain adalah hal yang sangat bisa membuat frustasi. Direktori sebaiknya diatur secara terpusat, hal ini penting sebagai sebuah otorisasi sebagai sumber informasi. Dengan cara ini tidak perlu lagi adanya pencarian dalam beberapa sumber untuk suatu informasi dan selanjutnya dapat dipastikan kebenaran informasi yang diperoleh tersebut.

Salah satu kegunaan direktori adalah menjadi sarana interaksi langsung ketika mencari informasi secara manual. Aplikasi perangkat lunak dapat mengambil informasi dalam sebuah direktori untuk memberikan suatu informasi yang disajikan dengan lebih baik dalam bentuk layanan-layanan. Layanan tersebut memberikan landasan bagaimana pihak lain dapat berinteraksi dalam dunia digital, mengenalkan *user* antara satu dengan yang lain, mengatur otorisasi, mengijinkan *user* berkomunikasi dengan *user* lain, disamping untuk menjaga informasi itu sendiri. Masing-masing dari layanan dasar yang seringkali disebut infrastruktur ini harus mempunyai informasi tentang identitas dari sumber informasi itu sendiri. Secara tegas dapat dikatakan bahwa ada sebuah keuntungan ketika memilikisebuah aset informasi yang diatur dan didefinisikan dengan jelas apa dan bagaimana metode akses data tersebut.

Sebuah direktori biasanya berisi entri-entri yang statis dengan frekuensi perubahan yang sedikit karena direktori di desain untuk memberikan respon sangat cepat dalam pencarian informasi. Sebuah database sering berisi entri-entri yang frekuensi perubahannya tinggi. Database didesain untuk penyajian data yang mudah, mendukung manipulasi data, serta mendukung proses baca dan tulis data yang terus menerus (Arkills, 2003).

Masing-masing teknologi mempunyai kelebihan dan kekurangan. Database baik pada penyimpanan objek yang memiliki pengurutan dengan banyak cara. Database biasanya mengimplementasikan sebuah mekanisme penguncian untuk mencegah dua sistem dari penulisan informasi yang sama, sedangkan direktori tidak. *Query* yang kompleks biasanya lebih banyak dalam sebuah basisdata daripada direktori. Database mengatur data yang besar dengan sangat baik, dimana direktori tidak didesain untuk tujuan ini. Database mengijinkan seseorang untuk menyimpan prosedur proses yang efisien dari permintaan yang komplek.

Di sisi lain, banyak ahli teknologi informasi mengetahui bahwa pengimplementasian direktori penting. Dalam buku yang sama, diuraikan keuntungan dari direktori yaitu :

1. Membuat administrasi jaringan lebih mudah
   - pengaturan secara terpusat untuk informasi *user*.
   - pengaturan secara terpusat pada konfigurasi mesin oleh komputer.
   - pengaturan terpusat dari akun pengguna.
   - mengurangi biaya dari dukunganpengaturan terpusat yang telah diterapkan.
2. Keseragaman akses untuk sumber daya jaringan :
   - penggunaan penamaan yang seragam.
   - potensial untuk Single SignOn pada sumber-sumber dalam jaringan.

3. Pencarian informasi pada satu tujuan
   - informasi kontak
   - lokasi sumber daya jaringan
   - potensial sebagai katalog sejumlah data
4. Improvisasi pengaturan data
   - meningkatkan konsistensi data yang biasa digunakan secara luas
   - menyediakan pengaturan keamanan terpusat untuk data bisnis yang penting.
   - mengatur data dalam struktur logika.
5. Membantu kelancaran proses bisnis
6. Memberikan tempat penampungan dan pencarian untuk aplikasi dan layanan data.

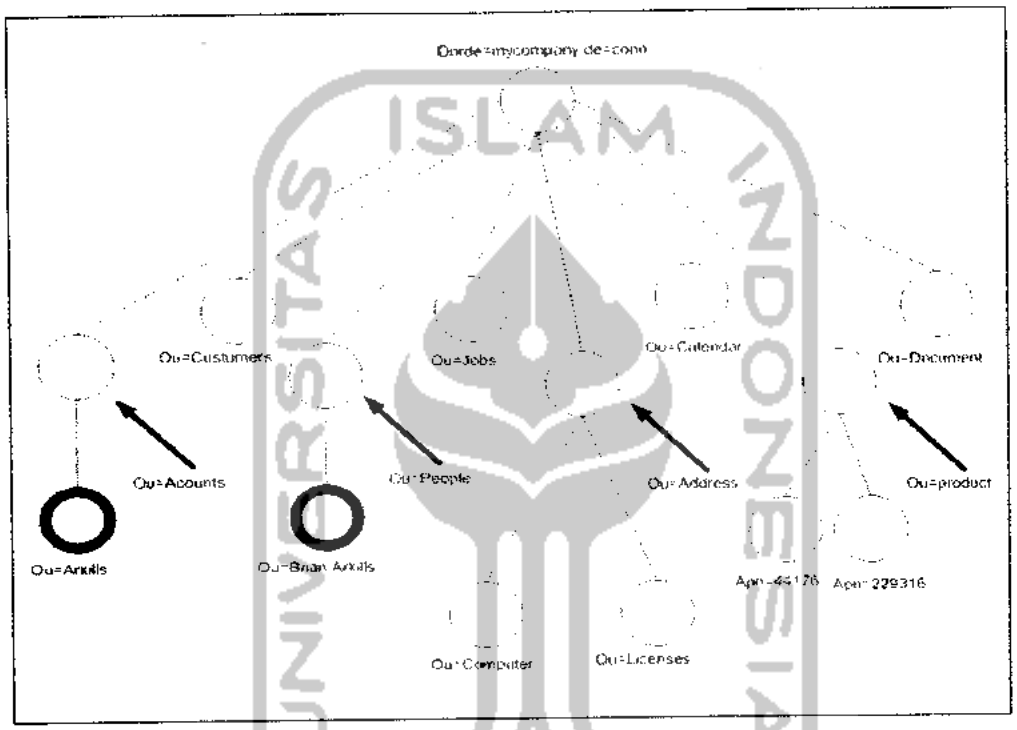
2.3.2 Pengenalan LDAP

2.3.2.1 Namespace

Setiap direktori membutuhkan *namespace* untuk memberikan nama suatu entri. Menurut Arkill (2003) secara umum *namespace* pada LDAP adalah sistem yang digunakan untuk mereferensikan suatu objek dalam direktori LDAP atau dengan kata lain namespace merupakansuatu sistem penamaan.

Namespace diorganiasasikan dalam sebuah hirarki sehingga pengaturannya dapat didelegasikan pada beberapa titik dalam struktur hirarki. Hirarki adalah turunan dari namespace yang memberiakn arti efektif pada delegasi koorperatif dari manajemen (Arkill, 2003). Hal ini merupakan keutungan yang signifikan dari LDAP dibanding database dan hal tersebut dijadikan salah satu faktor utama dalam memutuskan bagaimana data diorganisasikan dalam suatu direktori. Banyak dari direktori yang digunakan untuk membagi namespace, sebagai contoh yang terjadi misalnya sebuah standar internet yaitu DNS *(Domain Name System)*. DNS secara alami didefinisikan secara hirarki. *Namespace* dari LDAP juga berupa hirarki

sehingga hampir sama dengan DNS. Banyak direktori LDAP menggunakan *namespace* dari DNS, sehingga *namespace* dari LDAP bekerja seperti DNS. Hal ini membantu membuat LDAP lebih atraktif dan memberikan kontribusi pengembangan masa depan dari direktori LDAP yang terintegrasi secara global.



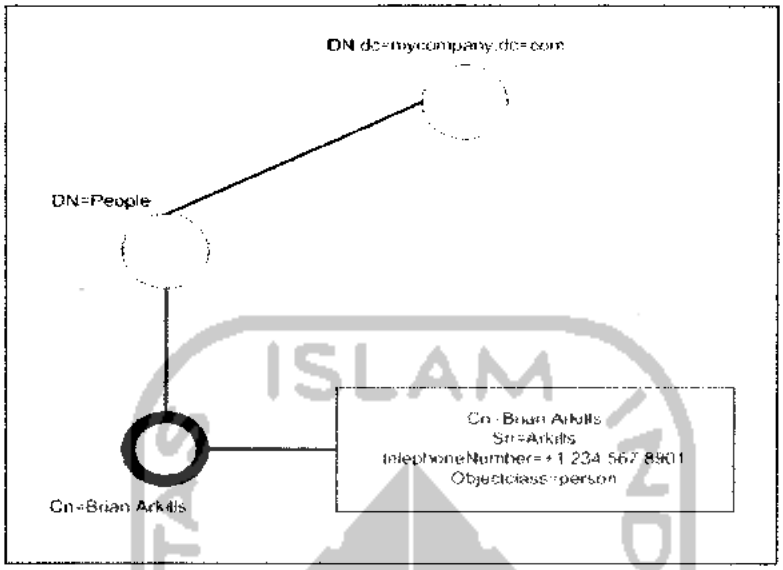
Gambar 2.1 Contoh Hirarki pada *Namespace* LDAP (Arkills, 2003)

Gambar 2.1 menunjukkan sebuah versi sederhana dari direktori mycompany.com. Nama dari direktori utama dikenal sebagai basis DN (*Distinguish Name*). Direktori utama tidak membutuhkan entri. Server berbasis DN biasanya mencocokan nama DNS dari server direktori dan menggunakan atribut *Domain Component* (DC) untuk merepresentasikan zona DNS. Tetapi bagaimanapun juga, server yang berbasis DN tidak perlu disamakan dengan nama server DNS. Server direktori yang berbasis DN mungkin berbeda untuk kepentingan fleksibilitas yang lebih besar dalam membuat desain dari sebuah arsitektur direktori yang terdistribusi di atas beberapa server direktori. Fleksibilitas untuk membuat sebuah direktori yang

terdistribusi menggunakan *namespace* adalah kunci keuntungan LDAP dibanding database.

Masing-masing entri dalam direktori mempunyai sebuah nama unik yang diketahui sebagai *Distinguish Name(*DN*)*. Masing-masing entri juga mempunyai sebuah nama lokal yang disebut dengan *Relative Distinguish Name(*DN*)*. Masing-masing entri juga mempunyai sebuah nama lokal yang disebut dengan *Relative Distinguish Name* / RDN (Arkill, 2003). RDN unik untuk semua entri dalam *container* tersebut. Sebuah *container* dapat diumpamakan seperti sebuah direktori atau folder dalam sebuah system file. Menurut Arkills (2003), RDN adalah tipe atribut dan bagian nilainya.

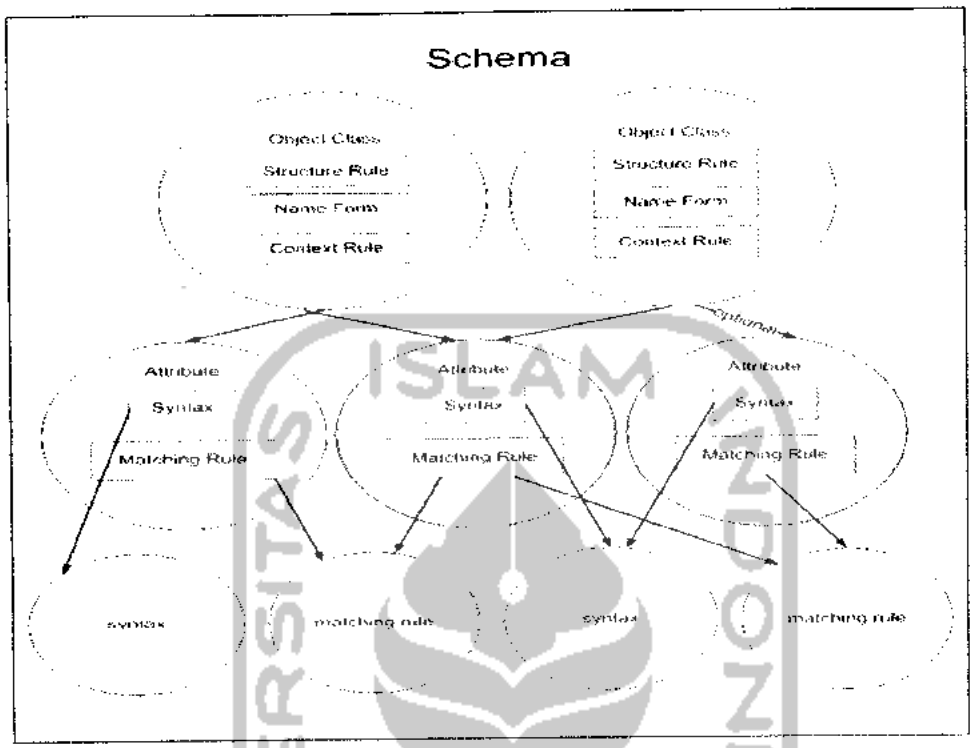
Entri DN dari person seperti yang ditunjukkan oleh gambar 2.2 dapat menjadi cn=Brian Arkills, ou=peole, de=mycompany, dc=com, Perlu diperhatikan bahwa masing-masing komponen RDN di sini termasuk keduanya yaitu atribut dan nilainya. Sebagai contoh, komponen tunggal cn=Briian Arkills mempunyai atribut cn dan nilai atribut Brian Arkills. Nilai atribut tanpa tipe atribut tidak sesuai untuk entri distinguish, karena nilainya mungkin mengacu ke tipe atribut yang berbeda pada banyak entri. Catatan bahwa cn=Brian Arkills harus unik dari semua entri dalam *container* ou=People agar termasuk kualifikasi RDN.



Gambar 2.2 Contoh RDN (Arkills, 2003)

2.3.2.2 Schema

Arkills (2003), *schema* mendefinisikan aturan-aturan yang mengatur hal-hal apa saja yang bisa dilakukan oleh direktori LDAP. *Schema* memegang peranan penting, tetapi *user* tidak perlu tahu apa saja dilakukan oleh *schema.* Di samping itu, *schema* juga menentukan tipe data entri yang bisa dibuat. Memodifikasi schema dapat meningkatkan fungsionalitas direktori. Memodifikasi tersebut misalnya menambahkan tipe baru pada objek atau membuat tipe baru pada atribut.



Gambar 2.3 Diagram Konsep Schema Arkilis (2003)

Schema terdiri dari beberapa. Gambar 2.3 menggambarkan bagaimana masing-masing elemen schema saling berelasi. Dari gambar tersebut bisa diketahui bahwa terdapat interdependensi antar elemen, dalam faktanya masing-masing elemen bisa saja bergantung pada beberapa elemen yang lain.

Kelas objek mendefinisikan tipe-tipe entri yang diperbolehkan dalam sebuah direktori. Kelas objek berisi :

1. *content rule* : mendefinisikan atribut-atribut pada kelas objek.

2. *structure rule* : menjelaskan bagaimana masing-masing kelas objek dapat berpartisipasi dalam *name space.*

3. *name form* : menjelaskan atribut apa saja yang busa digunakan untuk memberikan nama entri pada kelas objek.

4. *attribute* : mendefinisikan jenis informasi yang diasosialisasikan dengan masing-masing kelas objek, dan kemudian di dalam entri.

5. *attribute type* : didefinisikan melalui sintaks, matching rules, dan informasi operasional yang lain.

6. *syntax* : menentukan bagaimana nilai data direpresentasikan.

7. *matching rule* : menentukan bagaimana membandingkan nilai-nilai data tersebut pada operasi LDAP.

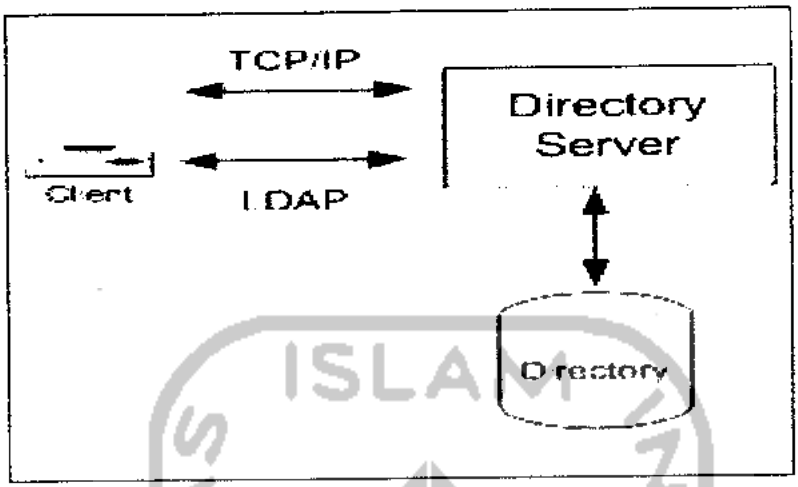
2.3.2.3 Interaksi *client*

1. Model *client*-server

LDAP menggunakan TCP/IP untuk berkomunikasi. Sebuah *client* agar dapat tersambung ke sebuah direktori LDAP. LDAP mengurangi kelebihan beban untuk membuat sebuah sesi yang mengijinkan banyak operasi dari sesi *client* yang sama. Hal tersebut juga memberikan trafik yang efisien dari penempatan karena hampir semua data yang disimpan dalam direktori adalah berupa informasi teks. LDAP menggunakan BER *encording* untuk mengacak nilai data atribut yang melewati antara server dan *client*. Ini berupa metode pengacakan komplek yang diadopsi dari X.500 (Priantoro, 2005).

Arkills (2003) menyatakan bahwa set dari operasi LDAP saling berhubugan dengan set dari standar *application programming interface* (API) dalam bahasa yang berbeda. API adalah sebuah set dari fungi-fungsi ini memberikan sebuah level yang lebih tinggi dalam penyampaiannya dengan menyembunyikan kode-kode yang susah didalamnya. Beberapa API berupa sumber yang tertutup *(closed source)*, artinya kodc-kode dari fungsi tersebut disembunyikan dari semua orang. Lainnya adalah kode yang terbuka (*open source*), dalam arti bahwa setiap orang dapat melihat detail dari code tersebut. API dan LDAP semuanya adalah *open source* (Arkills,2003). Sebagai contoh fungsi dari LDAP API, yang membuat operasi penambahan entri. Ada standar fungsi LDAP yaitu ldap_add() dalam standar API bahasa C dimana sebuah aplikasi akan menggunakannya untuk meminta server melakukan operasi penambahan. Semua aplikasi yang menggunakan API dari LDAP untuk berinteraksi dengan server LDAP disebut juga LDAP-*enabled*.

Gambar 2.4 menggambarkan server LDAP pada lingkungan TCP/IP. Gambar tersebut menunjukkan interaksi antara klien dengan directory server melalui TCP/IP yang kemudian directory server tersebut berinteraksi terhadap direktori LDAP sesuai perintah yang diberikan oleh klien.



Gambar 2.4 Server LDAP pada TCP/IP Environment (Volgmaier, 2004)

2. Client

Client LDAP dapat berupa perangkat lunak tunggal (standalone) dimana seseorang berinteraksi dengan mengetikkan perintah yang dibutuhkan, ataupun dapat berupa bagian mengetikkan perintah yang dibutuhkan, ataupun dapat berupa bagian yang terintegrasi dengan perangkat lunak yang mempunyai operasi otomatis dan perintah yang digunakan tersembunyi dari pengguna. Sebagai contoh sistem operasi Windowa 2000 telah mengintegrasikan fungsi-fungsi klien LDAP ke dalam beberapa bagain dari aplikasi ini. Dalam windows 2000, dapat dilakukan pemilihan opsi pencarian dari menu start, dan mencari data people (Outlook,Situs, dll) atau printer dalam Microsoft Active Directory (atau dengan kata lain server LDAP). Ada banyak situs LDAP-enabled yang memberikan sebuah interface tunggal (seringkali disebut portal) untuk seseorang yang menggunakan pencarian dan pemutahiran entri dalam server LDAP dari perusahaannya. Sebagai tambahan ke situs tersebut, hampir semua browser yang modern mendukung protokol LDAP dan mampu sebagai klien untuk mengambil informasi dari LDAP. Fleksibilitas dari integrasi adalah salah satu alasan utama kenapa banyak perusahaan perangkat lunak yang mengadopsi protokol LDAP. Keindahan dari LDAP standar terbuka menjadi dikenal direalisasikan bahwa setiap

klien LDAP atau aplikasi LDAP-enabled dapat dengan sukses berkomunikasi dengan setiap server LDAP, tidak berdasar pada bagian sistem operasi klien ataupun server. Standar terbuka, model multipleplatform membuat integrasi mudah, tetapi sebuah pasar dapat membuat integrasi menjadi sulit. Ini berarti bahwa implementasi dari LDAP adalah komplek, lingkungan sistem operasi yang tidak homogen secara signifikan lebih mudah daripada mengimplementasikan teknologi lain.

3. Operasi

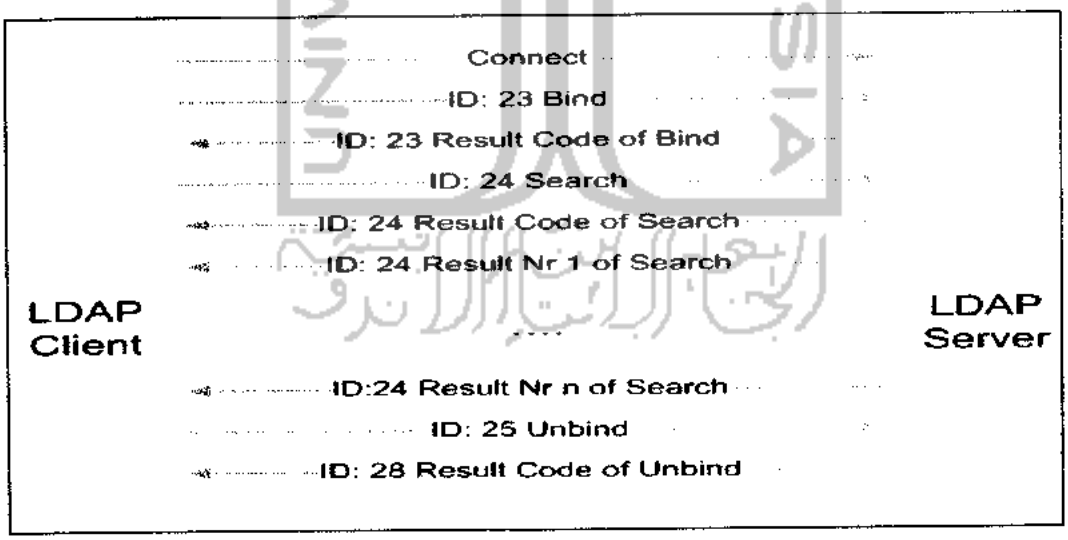
Ada sepuluh operasi dalam LDAP. Terbatasnya jumlah dari operasi sangat penting, pada program client interaksi dengan direktori sangat lebih mudah daripada program client yang berinteraksi dengan teknologi lain yang hampir sama. Operasi-operasi tersebut dapat digolongkan dalam tiga kategori dasar, seperti ditunjukkan dalam tabel 2.1.

Tabel 2.1 Operasi pada LDAP (Arkills, 2003)

Kategori	OperasiLDAP
Operasi sesi client	*Bind, unbind, dan abandon*
Query dan operasi pengambilan	*Search dan compare*
Operasi modifikasi	*Add, modify, modifyRDN dan delete*
Tambahan	Tambahan

Operasi tambahan adalah unik untuk seluruh operasi. Operasi tambahan sebagai tempat untuk emplementasi direktori yang spesifik untuk menambah fungsi dari protokol yang masih mempunyai sebuah sintaks yang didefinisikan terlebih dahulu. Perancang LDAP menunjukkan banyaknya perhatian dengan menambahkan operasi tambahan. Dengan melakukan standarisasi untuk melakukan penambahan fungsi operasional, mereka mengurangi persepsi yang salah dalam jumlah yang terbatas dari operasi. Operasi sesi client membantu untuk mengontrol konteks sesi client server untuk semua *request subsequent* operasi LDAP dari client tersebut. Operasi bind dan unbind mengijinkan client untuk menyambung identitas dengan LDAP. Identitas ini dapat digunakan oleh direktori dalam menentukan hak akses untuk melakukan operasi lain, dan dapat digunakan untuk mengatrol akses ke informasi direktori. Operasi abandon mengijinkan client membatalkan sebuah operasi *request*. Operasi *query* mengijinkan client melihat informasi dalam direktori. Hampir semua pembaca LDAP yang baru membutuhkan pengetahuan bagaimana untuk menggunakan pencarian cerdas pada sebuah direktori. Operasi pencarian adalah operasi yang paling sering dilakukan, keahlian dalam menggunakannya akan lebih bernilai. Operasi pencarian mempunyai banyak parameter, faktanya, parameter-parameter lebih banyak dari operasi lain. Operasi pembandingan mengijinkan sebuah client untuk me-*request* sebuah verifikasi dari informasi yang diasosialisasikan dengan sebuah entri, client mengirimkan nilai dari entri dan server merespon dengan sukses jika cocok atau gagal jika tidak cocok. Operasi modifikasi mengijinkan client untuk mengubah informasi dalam direktori. Operasi ini mungkin terbatas dalam beberapa *instant,* sebagai contoh dalam suatu kasus dari sebuah direktori yang *read-only*. Operasi modifyRDN hanya satu dalam kebutuhan dari kumpulan penjelasan. Operasi modifyRDN mengijinkan client untuk mengubah nama dari sebuah entri dan mungkin memindahkan entri ke kontainer yang berbeda.

Gambar 2.5 mengilustrasikan proses-proses ini. Gambar 2.5 memberikan contoh proses pencarian serta memberikan gambaran apa yang terjadi. Berawal dari client membuka koneksi ke komputer dimana server LDAP terinstal, kemudian client melakukan bind terhadap server LDAP. Proses bind ini mengkualifikasi client menggunakan ID dan password *user* yang valid. Apabila ID dan password *user* tidak server terima, maka server mengasumsikan bahwa client meminta koneksi sebagai *anonymous user*, *user* dengan level akses terendah jika ada. Server menjawab dengan kode hasil untuk menunjukkan proses bind sukses. Selanjutnya client mengirimkan query ke server, Sekali lagi server mengeksekusi request tersebut dan mengirimkan kembali data yang diminta apabila ada, juga mengirimkan kode hasil dari operasi pada pesan yang terpisah. Apabila server menemukan lebih dari satu hasil, maka akan dikirimkan dengan sejumlah pesan LDAP. Di akhir percakapan, client mengirimkan *request unbind* sehingga server menutup koneksi (Voglmaier, 2004).



Gambar 2.5 Komunikasi Antara Client LDAP dan Server

2.3.2.4 Keamanan

Voglmaier (2004) dalam bukunya yang berjudul *The ABCs of LDAP: How to Install, Run, and Administer LDAP Service,* menyebutkan bahwa sebelum client dapat mengakses data yang ada pada server LDAP, client harus menyelesaikan dua proses terlebih dahulu, yaitu otentikasi dan otorisasi. Dua proses ini sedikit berbeda antara satu sama lain. Fokus kali ini lebih kepada otentikasi oleh karena otorisasi belum terbahas secara tuntas dalam standar.

Otentikasi terjadi ketika client mengidentifikasikan dirinya ke server agar bisa terhubung. Proses ini tergantung dari mekasnisme otentikasi yang digunakan. Cara paling mudah agar terhubung ke server adalah tanpa perlu menunjukkan identitas. Cara ini disebut *anonymous connection* dengan hak akses terendah. Ada beberapa skema otentikasim dari *simple authentication* dengan *user dan password,* hingga otentikasi menggunakan sertifikat. Sertifikat yang dimaksud disini memeberikan jaminan kepada server bahwa client benar-benar merupakan client yang mereka kliam. Sertifikat-sertifikat ini juga mampu menjamiun identitas server bagi client (Voglaier, 2004).

Begitu client dikenali oleh server, client akan mendapatkan hak akses terhadao data. Otorisasi merupakan proses dimana server mengijinkan hak akses yang sesuai kepada client yang sebelumnya telah diotentikasi. Ini berarti *user* dapat membaca dan menulis data dengan batasan-batasan yang tergantung dari level akses yang dimilikinya. Untuk mendefinisikan apa saja yang bisa dilakukan client, server harus memelihara *access control information (ACI).*

Menurut Voglmaier (2004), ada beberapa level otentikasi yang berbeda sehingga muncul beberapa metode yang bermacam-macam pula untuk mengotentikasi client, yaitu :

1. Anonymous Access

   Jenis pertama dari otentikasi adalah tidak ada otentikasi sama sekali, atau bisa juga disebut *anonymous bind* karena server tidak tahu menahu siapa yang sedang meminta koneksi. *Anonymous bind* digunakan oleh data publik semisal buku telepon publik. Konfigurasi server menentukan apakah akses *anonymous* diijinkan atau tidak.

2. Basic Authentication

   Selain akses *anonymous,* otentikasi yang juga sederhana adalah *basic authentication* yang biasanya digunakan oleh protokol HTTP. *Client* secara sederhanan mengirimkan *user credential* melalui jaringan. Dalam LDAP ini berarti *client* mengirimkan *distinguished nama dari user dan passwordnya.* Kedunya dikirimkan melalui jaringan dalam bentuk *plaintext* tanpa enkripsi. Metode ini tidak masalah untuk lingkungan yang terjamin kemanannya, tetapi dalam intranet sekalipun hal ini bukanlah ide yang bagus.

   Server mencari atribut *user*Password dalam entri yang sesuai dengan *distinguish name.* Nilai dari atribut ini dicocokkan dengan input yang diberikan *user*. Apabila password tersebut sesuai, koneksi dapat dibangun ke server LDAP. Sekali lagi, ini bukanlah metode paling aman diantara metode-metode yang lain, tetapi dapat juga diterima apabila diimplementasikan di dilingkungan intranet.

3. LDAP over SSL/TLS

   Protokol SSL (Secure Socket *Layer*) mengimplementasikan mekanisme-mekanisme keamanan pada lapisan protokol TCP/IP yaitu antara *transport layer dan aplication layer,* dalam contohnya adalah *layer* dibawah LDAP. SSL didasarkan pada kriptografi public key dan dikembangkan oleh Netscape. Protokol TLS (Transport *Layer* Security) diciptakan dari SSL versi tiga.

   Tujuan dari protokol TLS itu sendiri adalah untuk menyediakan integritas privasi data. Ini berarti protokol TLS menjamin data yang dikirim antara dua

pihak sampai tanpa ada modifikasi dan komunikasi di antara mereka dijamin terenkripsi.

4. Kerberos

LDAP (v2) mendukung mekanisme *bind* berdasar pada Kerberos. Akan tetapi ini tidak dapat didukung secara langsung pada LDAP(v3). Tidak didukung secara langsung di sini artinya bahwa mekanisme bind ini dapat digunakan sebagai mekanisme keamanan ketika ada persetujuan yang terbangun oleh protokol SASL. Kerberos itu sendiri merupakan protokol yang bergantung pada pihak ketiga, *Authentication Server*. Kerberos adalah mekanisme keamanan yang direncanakan bagi lingkungan yang benar-benar tidak aman misalnya saja internet. Kerberos tidak berasumsi mengenai integritas pesan terkirim antara dua pihak yang sedang berkomunikasi. Komunikasi dienkripsikan dan kedua pihak yang berkomunikasi dapat yakin akan identitas pihak lainnya.

Protokol Kerberos sangat stabil dan handal untuk memenuhi kebutuhan keamanan khususnya dalam komunikasi di lingkungan yang tidak aman. Dapat juga digunakan bersama dengan banyak protokol dan merupaka *platform independent.*

5. SASL

Simple Authentication and Security Layer (SASL) merupakan suatu metode penyediaan layanan otentikasi bagi protokol-protokol yang *connection oriented* seperti misalnya LDAP. Standar SASL didefinisikan pada RCF 2222, SASL. Standar ini memungkinkan klien dan server menyetujui suatu *layer* keamanan untuk enkripsi. Setelah server dank lien terhubung, mereka menyetujui suatu mekanisme keamanan untuk percakapan selanjutnya. Salah satu mekanisme ini adalah Kerberos.

Pada saat buku tersebut ditulis, sejumlah mekanismeyang didukung SASL, meliputi :

- *Anonymous*
- CRAM-MD5
- Digest-MD5
- *External*
- Kerberos (v4)
- Kerberos (v5)
- SecureID
- Secure Remote Password
- S/Key
- X.509

Otentikasi member *credential* kepada server yang dibutuhkan agar *user* diijinkan mengakses server. Selain itu juga untuk memverifikasi identitas dari pihak lain, baik dari sisi server maupun sisi klien. Begitu koneksi terjalin, klien dan server dapat bertukar pesan dengan utuh tanap ada modifikasi ataupun penangkapan pesan dari pihak yang tidak berhak.

Otorisasi atau bisa juga disebut *access control* adalah proses dimana server mengijinkan hak akses yang sesuain bagi *user* yang terkoneksi. Ketika buku tersebut dibuat, standar mengenai otorisasi belum dibahas secara luas. Ini bukan berarti server direktori tidak mendukung otorisasi. *Access Control Information* (ACI) terdapat dalam *access control list* (ACL).

Access Control Information dapat berisi spesifikasi-spesifikasi berikut ini:

a. Data Protection : server dapat mengijinkan ataupun manolak akses ke direktori maupun DN.

b. Data Access : server dapat menentukan klien-klien yang memiliki akses ke sistem

- *Anonymous* : semua *user*, tanpa otentikasi

- *Authenticated user* : user-user yang telah diotentikasi oleh sistem.
- *Self* : user yang berhubungan dengan entri tujuan.
- *Distinguished name* : user yang sesuai dengan ekspresi pada distinguished name

c. *Level of access*

- No.access
- Hak untuk *bind*
- Hak untuk eksekusi atau mambandingkan
- Hak untuk eksekusi pencarian
- Hak untuk membaca hasil pencarian
- Hak untuk memodifikasi entri

- *Further Requirement* : server dapat membatasi nomor IP ataupun nama *host* yang dilayani, waktu layanan dapat diijinkan atau ditolak, dan lainnya.

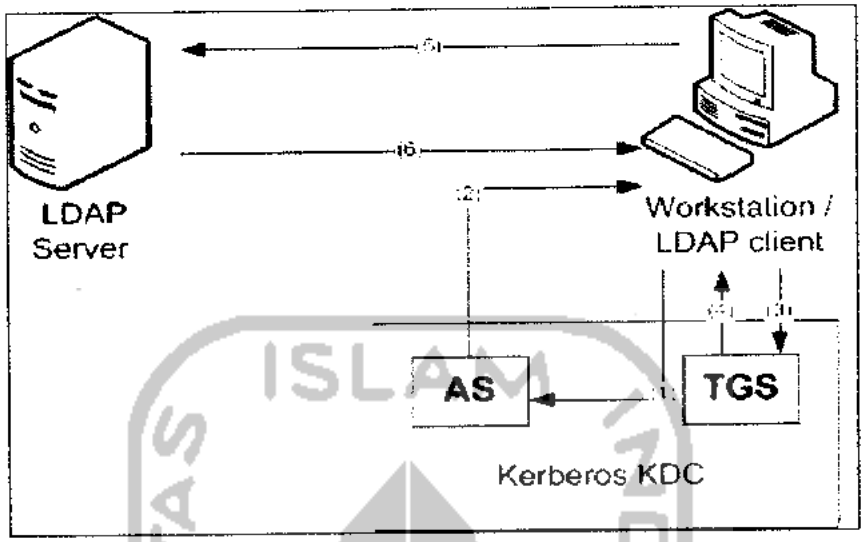
BAB 3 ANALISIS DAN PERANCANGAN SISTEM

3.1 Desain Penelitian

Kerberos dengan LDAP akan diintegrasikan dalam penelitian ini sebagai dasar solusi keamanan otentikasi bagi LDAP. Ada beberapa strategi penggabungan Kerberos dan LDAP. Penelitian ini akan mengimplementasikan salah satu strategi pengintegrasian yaitu melakukan otentikasi *bind* terhadap server LDAP menggunakan tiket Kerberos. Dengan opsi ini *user* akan diotentikasi oleh Kerberos untuk kemudian menggunakan tiket yang diperoleh untuk mengakses direktori LDAP melalui SASL GSSAPI. Secara garis besar, sistem ini dibagi menjadi dua bagian yang berbeda, yaitu otentikasi dan database *user*. Database *user* ini berisi informasi mengenai atribut-atribut yang dimiliki *user* yang selanjutnya akan ditangani oleh server LDAP. LDAP akan menyimpan informasi ini dan membuatnya tersedia bagi seluruh *host* pada *realm*. Fungsi Kerberos hanya untuk mengelola otentikasi yang aman yang tentunya Kerberos tidak tahu menahu mengenai atribut-atribut yang dimiliki oleh *user*.

3.1.1 Desain Arsitektur

Desain arsitektur sistem kerberos pada gambar 3.1 menggambarkan proses-proses yang dilakukan sistem. Model *client server* yang digunakan dalam penelitian ini yaitu server LDAP dengan otentikasi yang ditawarkan Kerberos. *Client* LDAP berupa perangkat lunak tunggal (*standalone*). Interaksi klien dengan direktori LDAP dilakukan dengan operasi-operasi LDAP *client* yang telah dijelaskan di bab sebelumnya.



Gambar 3.1 Arsitektur Sistem Kerberos

Dari gambar 3.1 dapat dijabarkan sebagai berikut :

1. *User* melakukan proses otentikasi ke Kerberos Authentication Server dan mendapatkan *initial ticket* (TGT).
2. *Kerberos Authentication Server* mengembalikan TGT.
3. *User* memulai aplikasi *client* LDAP dengan parameter :

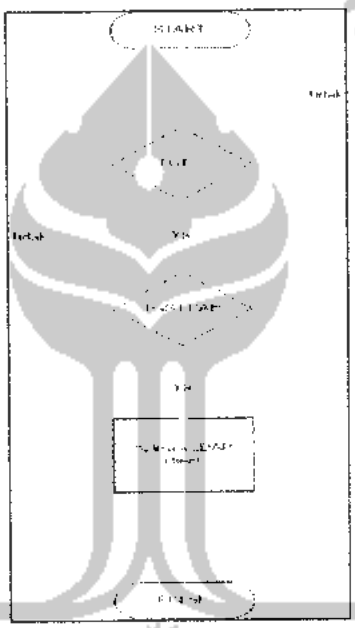
   Mechanism -GSSAPI
   Server -hostname

   Aplikasi *client* LDAP secara internal memanggil ldap_sasl_bind() dengan GSSAPI sebagai mekanisme otentikasi yang digunakan. *Client* memverifikasi apakah mekanisme GSSAPI memang didukung. Setelah diverifikasi, klien LDAP mengirim tiket Kerberos yang telah terbungkus *request* LDAP ke server LDAP, Aplikasi *client* (menggunakan ldap_sasl_bind()) meminta *Service Ticket* kepada server LDAP menggunakan TGT yang dimiliki *user.*
4. TGS *(Ticket Granting Server)* memberikan LDAP *service request.*

5. *Aplikasi* klien (ldap_sasl_bind()) mengirim *service ticket* (ST) kepada server LDAP. Server LDAP memverifikasi tiket tersebut menggunakan mekanisme SASL GSSAPI.

6. Berdasarkan hasil dari validasi, ldap_sasl-bind() mengembalikan nilai sukses atau gagal ke aplikasi.

3.1.2 Flowchart Otentikasi Kerberos



Gambar 3.2 Flowchart Otentikasi Kerberos dengan *Client* LDAP

Alur proses yang akan dijalankan sistem dapat diilustrasikan dengan flowchart pada gambar 3.2. Gambar 3.2 menggambarkan rancangan sistem melalui flowchart. Awal menjalankan sistem, *user* harus memiliki TGT. Apabila TGT telah diperoleh, maka proses dilanjutkan. Setelah itu *user* memperoleh TGS, maka proses selanjutnya adalah pengaksesan direktori LDAP melalui aplikasi LDAP *client*.

3.2 Analisis Kebutuhan

3.2.1 Kebutuhan Server

Berikut ini adalah komponen yang dibutuhkan oleh komputer server.

Tabel 3.1 Komponen Komputer Server

Nama	Komponen
Komputer server	a. Fedora 9 b. Kerberos V5 c. *Key distribution center* / KDC d. SASL e. OpenLDAP-2.4.15

3.2.2 Kebutuhan *Client*

Kebutuhan *client* terdiri dari kebutuhan atas perangkat keras serta perangkat lunak. Perangkat lunak yang dibutuhkan adalah sebagai berikut

Tabel 3.2 Kebutuhan Komputer *Client*

Nama	Kebutuhan komponen
Komputer *client*	a. Windows dan Linux b. IP address eth0 = 192.168.1.10 c. IP gateway = 192.168.1.2

3.2.3 Perangkat Lunak Pendukung

Berikut ini adalah perangkat lunak yang digunakan :

1. Fedora 9

   Fedora 9 merupakan salah satu distro linux yang digunakan untuk membangun sistem otentikasi dengan menggunakan *kerberos*. Sistem operasi ini digunakan sebagai server. Pertimbangan menggunakan Fedora 9 dikarenakan adanya dukungan dari *software-software* yang akan digunakan. Implementasi sistem *kerberos* dalam Fedora ini didukung dengan perintah-perintah dasar dari linux sendiri.

2. Kerberos V5

   Kerberos diperlukan pada satu platform dengan struktur *single directory tree* yang berisi file sumber maupun file objek sehingga menjadi lebih sederhana dibandingkan apabila pengimplementasian *kerberos* pada bermacam-macam platform.

3. *Key distribution center*

   KDC menerbitkan tiket *kerberos*. Masing-masing KDC berisi salinan database yang disebarkan ke slave-slave KDC pada waktu-waktu tertentu. Semua perubahan database termasuk perubahan *password* dilakukan pada master KDC. *Slave* KDC menyediakan *ticket granting service*, tetapi tidak menyediakan administrasi database. Oleh sebab itulah *client* masih dapat memperoleh tiket meskipun master KDC tidak tersedia. Namun dalam penelitian ini hanya akan diadakan sebuah master KDC tanpa mengadakan replikasinya.

4. SASL

   Simple Authentication and Security Layer (SASL) merupakan suatu metode penyediaan layanan otentikasi bagi protokol-protokol yang *connection oriented* seperti misalnya LDAP. Standar SASL didefinisikan pada RFC 2222, *Simple*

Authentication and Security Layer. Standar ini memungkinkan *client* dan server menyetujui suatu *layer* keamanan untuk enkripsi. Setelah server dan *client* terhubung, mereka menyetujui suatu mekanisme keamanan untuk percakapan selanjutnya. Salah satu mekanisme ini adalah Kerberos.

5. OpenLDAP-2.4.15

   Dalam penelitian ini digunakan distribusi OpenLDAP-2.4.15 sebagai *software* penyedia protokol LDAP.

BAB 4 IMPLEMENTASI

4.1 Instalasi

4.1.1 Kerberos

4.1.1.1 Instalasi Kerberos V5

Kerberos diperlukan pada satu platform dengan struktur *single directory tree* yang berisi file-file sumber maupun file-file objek sehingga menjadi lebih sederhana dibandingkan dengan pengimplementasian Kerberos pada bermacam-macam platform. Mekanisme yang kedua membutuhkan proses *build* tersendiri untuk masing-masing platform dan tidak akan dijelaskan dalam penelitian ini.

Prosedur *build* yang harus dilakukan yaitu :

1. *cd /krb5-1.6.3/src*
2. *./configure*
3. *make*

Sesudah proses *build* Kerberos, selanjutnya adalah instalasi binary. Dapat dilakukan dengan perintah *make install.*

Pengujian proses *build* tersebut apakah berhasil atau tidak dengan perintah *make check* serta harus dilakukan dari level paling atas dari direktori *build.*

4.1.1.2 Instalasi KDC

KDC (*Key distribution center*) menerbitkan tiket Kerberos. Masing-masing KDC berisi salinan database Kerberos. Master KDC berisi salinan utama dari

database yang disebarkan ke *slave-slave* KDC pada waktu-waktu tertentu. Semua perubahan database termasuk perubahan *password* dilakukan pada master KDC. *Slave* KDC menyediakan *ticket granting service,* tetapi tidak menyediakan administrasi database. Oleh sebab itulah *client* masih bisa memperoleh tiket meskipun master KDC tidak tersedia. Namun dalam penelitian ini hanya akan diadakan sebuah master KDC tanpa mengadakan replikasinya.

Modifikasi file-file konfigurasi yaitu *krb5.conf* dank dc.*conf* ditujukan untuk menggambarkan informasi-informasi yang sesuai (seperti *hostname* dan nama *realm*).

File *krb.conf* berisikan informasi konfigurasi Kerberos, termasuk lokasi KDC dan server admin pada *realm* yang dibangun, *realm* default dan aplikasi Kerberos, dan pemetaan *hostname* ke dalam *realm* Kerberos. Normalnya, *krb5.conf* terinstal didirektori */etc.*

Table 4.1 berisi nama bagian-bagian dalam file *krb5.conf* beserta deskripsinya. File *krb5.conf* dapat menyertakan seluruh bagian ataupun hanya beberapa bagian saja.

Tabel 4.1 Komponen File *krb5.conf*

Nama Bagian	Deskripsi
Libdefaults	Berisi nilai-nilai default yang digunakan oleh library Kerberos V5
Login	Berisi nilai-nilai default yang digunakan oleh program login Kerberos V5
Appdefaults	Berisi nilai-nilai default yang bisa digunakan oleh aplikasi Kerberos
Realms	Berisi subbagian-subbagian yang berisi nama-nam *realm.* Masing-masing subbagian menggambarkan informasi mengenai

	realm tertentu, termasuk dimana ditemukan server Kerberos untuk suatu *realm*
Domain_realm	Berisi relasi-relasi yang memetakan nama domain dan subdomain ke nama *realm* Kerberos. Ini digunakan oleh program-program untuk menentukan *realm* yang mana yang menjadi domain *realm* dari *host* tersebut
Logging	Berisi relasi-relasi yang menentukan proses log program Kerberos
Capaths	Berisi jalur-jalur otentikasi yang digunakan dengan otentikasi *cross-realm* langsung. Entri dalam bagian ini digunakan oleh *client* untuk menentukan *realm* lanjutan pada *cross-realm authentication.*

Gambar 4.1 memperlihatkan contoh file *krb5.conf.* Pada gambar tersebut tidak seluruh bagian *krb5.conf* yang disebutkan dalam table 4.1 diikutsertakan. Hanya beberapa bagian yang diperlukan untuk pengimplementasian serta pengujian saja.

```
[logging]
 default = FILE:/var/log/krb5libs.log
 kdc = FILE:/var/log/krb5kdc.log
 admin_server = FILE:/var/log/kadmind.log

[libdefaults]
 default_realm = AUTH.LANGHUA
 dns_lookup_realm = false
 dns_lookup_kdc = false
 ticket_lifetime = 24h
 forwardable = yes

[realms]
 AUTH.LANGHUA = {
  kdc = auth.langhua:88
 # kdc = dc-02.auth.langhua.00
  admin_server = auth.langhua:749
  default_domain = auth.langhua
 }

[domain_realm]
  auth.langhua = AUTH.LANGHUA
  .auth.langhua = AUTH.LANGHUA

[kdc]
  profile = /var/kerberos/krb5kdc/kdc.conf

[appdefaults]
  pam = {
    debug = false
    ticket_lifetime = 36000
    renew_lifetime = 36000
    forwardable = yes
    krb4_convert = false
```

Gambar 4.1 File krb.conf

Dari gambar tersebut dapat diketahui bahwa secara default *realm* yang ada adalah AUTH.LANGHUA, sedangkan mesin KDC terdapat pada *host* kerberos.auth.langhua. Server admin juga terdapat pada *host* yang sama yaitu kerberos.auth.langhua.

File *kdc.conf* berisi konfigurasi KDC, termasuk default-default yang digunakan untuk menerbitkan tiket Kerberos. Biasanya *kdc.conf* terinstal di direktori */usr/local/var/krb5kdc*. *Kdc.conf* memiliki format yang sama dengan *krb5.conf*.

Bagian-bagian yang menyusun file *kdc.conf* diperlihatkan pada table 4.2 berikut ini.

Tabel 4.2 Komponen File *kdc.conf*

Nama Bagian	Deskripsi
Kdcdefaults	Berisi nilai-nilai default untuk proses KDC secara keseluruhan
Realms	Barisi subbagian-subbagian yang berisi nama-nama *realm*. Masing-masing subbagian menggambarkan informasi mengenai *realm* tertentu, termasuk dimana ditemukan server Kerberos untuk suatu *realm*
Logging	Berisi relasi-relasi yang menentukan cara proses log program-program Kerberos

4.1.1.3 Pembuatan Database Kerberos

Selanjutnya perintah *kdb5_util* akan digunakan pada master KDC untuk membuat database Kerberos dan *stash* file apabila diinginkan. *Stash* file merupakan salinan local *master key*. *Stash* file ini digunakan untuk mengotentikasi KDC itu sendiri sebelum menjalankan daemon *kadmind dan krb5kdc* (seperti bagian dari rangkaian booting mesin). Jika ditentukan untuk menginstal *stash* file ini, KDC akan

meminta master key setiap kali dijalankan. Ini berarti KDC tidak akan dijalankan secara otomatis, misalnya setelah sistem boot.

Hal penting lainnya yang perlu diperhatikan adalah bahwa *kdb5_util* akan meminta *master key* untuk database Kerberos. Kunci ini dapat berupa string. Kunci yang baik adalah yang dapat diingat tetapi tak seorangpun mampu menebaknya.

Gambar 4.2 berikut ini memperlihatkan mekanisme pembuatan database Kerberos dan *stash* file pada master KDC menggunakan perintah *kdb_util.*

```
[root@komas laptop ]# /usr/local/sbin/kdb5_util create -r AUTH.LANGHUA -s
Loading random data
Initializing database '/usr/local/var/krb5kdc/principal' for realm 'AUTH.LANGHUA',
master key name 'K/M@AUTH.LANGHUA'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:
Re-enter KDC database master key to verify:
```

Gambar 4.2 Mekanisme Pembuatan Database Kerberos

Langkah-langkah diatas akan membentuk lima file didirektori yang dispesifikasikan di *kdc.conf.* Kelima file tersebut yaitu file database Kerberos, *principal.db* dan *principal.ok;* kemudian file administrasi database Kerberos, *principal.kadm5;* file penguncian administrasi database, *principal.kadm5.lock;* dan stash file, k5stash. Direktori default adalah */usr/local/var/krb5kdc.* Jika tidak menginginkan *stash file,* maka perintah diatas dijalankan tanpa –s.

4.1.1.4 Access Control List User Kerberos

Berikut adalah pembuatan file *Access Control List,* dan meletakkan sedikitnya satu *principal* yang bertindak sebagai administrator. File ini digunakan oleh daemon *kadmind* untuk mengatur *principal* yang dapat melihat dan memodifikasi hak akses pada file database Kerberos. Nama file tersebut seharusnya

sesuai dengan nilai yang telah diatur pada *kdc.conf* untuk *acl_file*. Nama default yang digunakan adalah */usr/local/var/krb5kdc/kadm5.acl*.

Format file tersebut sebagai berikut :

Kerberos_principal *permission* *[target_principal][restriction]*

Principal Kerberos dapat mengikutsertakan "*" (wildcard) sehingga jika menginginkan semua *principal* dangan *instance* "admin" memiliki hak akses keseluruhan pada database, dapat dituliskan "**/admin@REALM*" dimana "*REALM*" disini adalah nama *realm* Kerberos yang telah didefinisikan.

Berikut ini isi dari file *kadm5.acl* :

```
*/admin@AUTH.LANGHUA.AC.ID      *
kerberos/admin@AUTH.LANGHUA.AC.ID  *
ldap/admin@AUTH.LANGHUA.AC.ID   *
ldap/auth.langhua.ac.id@AUTH.LANGHUA.AC.ID  *
```

Pada file diatas, seluruh *principal* pada *realm* AUTH.LANGHUA dengan *instance* admin memiliki keseluruhan hak akses administrative. ldap/auth.langhua.ac.id@AUTH.LANGHUA.AC.ID merupakan *service principal* untuk layanan LDAP.

4.1.1.5 Penambahan Administrator pada Database Kerberos

Langkah selanjutnya adalah menambahkan *principal* administrative pada database Kerberos. Untuk melakukan ini digunakan *kadmin.local* pada master KDC. *Principal* administrative yang dibuat harus ditambahkan pada file acl.

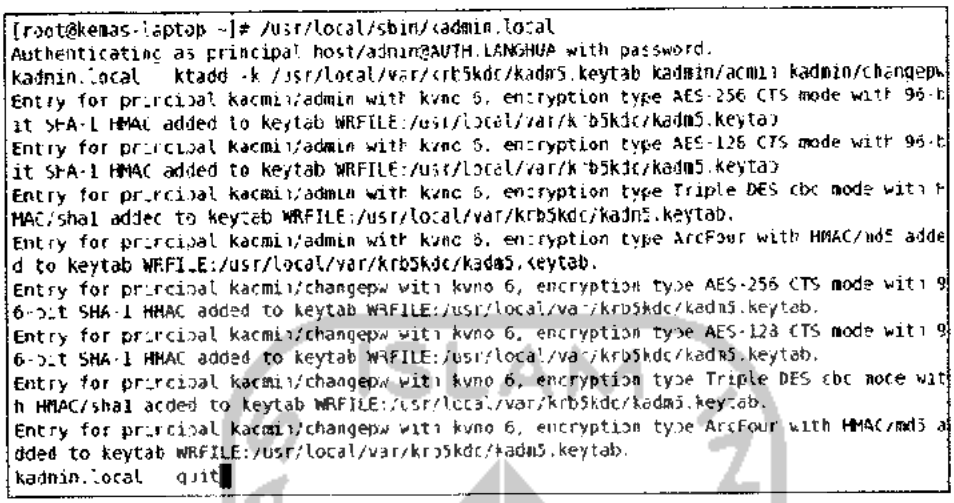
```
[root@kemas-laptop ~]# /usr/local/sbin/kdb5_util create -r AUTH.LANGHUA -s
Loading random data
Initializing database '/usr/local/var/krb5kdc/principal' for realm 'AUTH.LANGHUA',
master key name 'K/M@AUTH.LANGHUA'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:
Re-enter KDC database master key to verify:
kdb5_util: File exists while creating database '/usr/local/var/krb5kdc/principal'
[root@kemas-laptop ~]# /usr/local/sbin/kadmin.local
Authenticating as principal host/admin@AUTH.LANGHUA with password.
kadmin.local:  addprinc admin/admin@AUTH.LANGHUA
WARNING: no policy specified for admin/admin@AUTH.LANGHUA; defaulting to no policy
Enter password for principal "admin/admin@AUTH.LANGHUA":
Re-enter password for principal "admin/admin@AUTH.LANGHUA":
```

Gambar 4.3 Mekanisme Penambahan Administrator

Mekanisme penambahan *principal* admin ditunjukkan melalui gambar 4.3 diatas. Operasi *addprinc* akan menambahkan *principal* pada database Kerberos. Perintah yang digunakan menurut gambar 4.3 tersebut adalah *addprinc* [illegible] yang akan menambahkan *user principal* admin dengan *instance* admin pada *realm AUTH.LANGHUA*.

4.1.1.6 Pembuatan Keytab Kadmind

Keytab kadmin adalah kunci sukses yang akan mengaktifkan daemon *kadmind4* dan *v5passwdd* yang digunakan untuk mendekripsikan tiket administrator maupun *client* Kerberos. Sebelumnya perlu membuat *keytab kadmin* dengan entri-entri *principal kadmin/admin* dan *kadmin/changepw*. *Principal-principal* ini diletakkan pada database Kerberos secara otomatis ketika dibuat. Gambar 4.4 memperlihatkan tahapan-tahapan proses tersebut.



Gambar 4.4 Mekanisme Penambahan *keytab*

Sesuai dengan argument –k, *ktadd* akan menyimpan *keytab* terekstraksi pada */usr/local/sbin/kadmind.*

4.1.2 SASL

Computing Services Department di Carnigie Mellon University membuat library-librari SASL yang bisa diperoleh dari . Library-librari SASL versi 2 mendukung beberapa mekanisme SASL, diantaranya:

1. ANONYMOUS
2. CRAM-MD5
3. DIGEST-MD5
4. GSSAPI (MIT Kerberos 5 atau Heimdal Kerberos 5)
5. KERBEROS_V4
6. PLAIN

Alasan utama integrasi SASL dengan LDAP adalah otentikasi Kerberos akan dijalankan saat mengakses server LDAP. Demi kepentingan fleksibilitas dalam

otentikasi, akan dikembangkan server menggunakan dukungan SASL. Instalasi library SASL diperlukan dalam penelitian ini ditujukan untuk otentikasi pihak ketiga, dalam hal ini adalah Kerberos. Lebih khusus lagi menjembatani integrasi LDAP dengan Kerberos untuk keperluan inilah diperlukan adanya library SASL.

Proses pembangunan SASL tidak jauh berbeda dengan paket-paket yang lain. Dalam kesempatan kali ini digunakan distribusi Cyrus SASL versi 2. Gambar 4.5 menggambarkan proses pembangunan yang dilakukan pada distribusi Cyrus SASL.

```
Cd cyrus-sasl-2.1.22
./configure
Make
./bin/su -c "make install ; ln -s /usr/local/lib/sasl2 /usr/lib/sasl2"
```

Gambar 4.5 Proses Pembangunan CyrusSASL

4.1.3 LDAP

4.1.3.1 Instalasi OpenLDAPv3

Dalam penelitian ini digunakan distribusi OpenLDAP-2.4.15 sebagai *software* penyedia protokol LDAP. Setelah file sumber diekstrak, perintah berikut ini akan menjalankan konfigurasi yang telah ditentukan :

1. *./configure*
2. *Make depend*
3. *Make*
4. *Make test*
5. *Make install*

Tabel 4.3 Paket Instalasi OpenLDAP

Nama	Deskripsi
Libexec/slapd	Server LDAP
Libexec/slurpd	Replikasi LDAP
Bin/ldapadd *Bin/ldapmodify* *Bin/ldapdelete* *Bin/ldapmodrdn*	Perintah untuk menambah, memodifikasi, dan menghapus entri dari server LDAP.
Bin/ldapsearch *Bin/ldapcompare*	Perintah untuk mencari sebuah direktori LDAP dan membandingkan atribut tertentu pada sebuah entri
Sbin/slappasswd	Perintah yang akan membuat hash *password* yang cocok untuk digunakan dalam *slapd.conf*
*Lib/libladp** *Lib/liblber** *Include/ldap*.h* *Include/lber*.h*	SDK dari *client* OpenLDAP
Bin/ldappasswd	Digunakan untuk mengubah atribut *password* pada entri LDAP. Tool ini seperti */bin/passwd*
Sbin/salpdadd *Sbin/slapcat* *Sbin/slapindex*	Digunakan untuk memanipulasi data local yang tersimpan di *backend* yang digunakan oleh daemon *slapd*

Paket OpenLDAP berisi library-librari *client, server,* dan *development.* Tabel 4.3 menggambarkan komponen-komponen yang ada dalam paket instalasi. Seluruh *pathname* adalah relative tergantung proses instalasi yang dilakukan, tetapi secara default ada pada */usr/local.*

4.1.3.2 Konfigurasi OpenLDAP

File *slapd.conf* adalah file sumber yang utama dari server OpenLDAP *standalone (slapd), replication helper daemon (slurpd),* dan perintah-perintah yang berhubungan seperti *slpadd* dan *slapcat.* Sebagai aturan utama, tools *client*

OpenLDAP seperti *ldapmodify* dan *ldapsearch* menggunakan *ldap.conf* untuk pengaturan default.

Dalam file konfigurasi unix tradisional, *slapd.conf* adalah sebuah file ASCII dengan aturan :

1. Baris kosong dan baris yang diawali dengan tanda # akan diabaikan.
2. Parameter dan nilai yang diasosiasikan dipisahkan oleh karakter spasi atau tabulasi.
3. Sebuah baris dengan spasi kosong dalam kolom pertama dianggap sebagai kelanjutan dari sebelumnya.

Untuk sebuah kebutuhan umum, file *slapd.conf* yang digunakan oleh OpenLDAPv3 dapat dibagi menjadi dua bagian. Bagian pertama berisi parameter-parameter yang akan berpengaruh pada tingkah laku keseluruhan dari server OpenLDAP. Bagian kedua berupa parameter-parameter yang berhubungan dengan *backend* database yang digunakan oleh daemon *slapd*. Konfigurasi *slapd.conf* yang menunjukkan bagian tersebut ditunjukkan oleh gambar 4.6.

```
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#

include         /etc/openldap/schema/corba.schema
include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/duaconf.schema
include         /etc/openldap/schema/dyngroup.schema
include         /etc/openldap/schema/inetorgperson.schema
include         /etc/openldap/schema/java.schema
include         /etc/openldap/schema/misc.schema
include         /etc/openldap/schema/nis.schema
include         /etc/openldap/schema/openldap.schema
include         /etc/openldap/schema/ppolicy.schema
include         /etc/openldap/schema/collective.schema
include         /etc/openldap/schema/krb5-kdc.schema

# Allow LDAPv2 client connections.  This is NOT the default
allow bind_v2

# Do not enable referrals until AFTER you have a working directory
# service AND an understanding of referrals.
#referral       ldap://root.openldap.org

pidfile         /var/run/openldap/slapd.pid
argsfile        /var/run/openldap/slapd.args

# Load dynamic backend modules:
# modulepath    /usr/lib/openldap  or /usr/lib64/openldap
```

Gambar 4.6 Konfigurasi slapd.conf

Bagian umum mulai dari awal file sampai dengan direktif database yang pertama. Mulai dari bagian database ditandai dengan parameter database, sampai dengan akhir file. Dimungkinkan juga untuk mendefinisikan banyak database yang disediakan oleh instalasi tunggal *slapd*. Masing-masing independent secara logika, dan file database yang diasosiasikan akan disimpan perbagian.

4.1.3.3 Schema

OpenLDAPv3 secara default menyertakan beberapa skema yang popular untuk digunakan oleh administrator. Kebutuhan aplikasi yang akan digunakan dalam direktori akan menentukan skema yang dipakai. Secara default, file ini berlokasi didalam direktori */usr/local/etc/openldap/schema* setelah proses instalasi. Dalam file konfigurasi, parameter *include* digunakan untuk menentukan skema yang akan diikutkan kedalam server. Pendefinisian skema dalam file konfigurasi ditunjukkan oleh gambar 4.7.

```
include         /etc/openldap/schema/corba.schema
include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/duaconf.schema
include         /etc/openldap/schema/dyngroup.schema
include         /etc/openldap/schema/inetorgperson.schema
include         /etc/openldap/schema/java.schema
include         /etc/openldap/schema/misc.schema
include         /etc/openldap/schema/nis.schema
include         /etc/openldap/schema/openldap.schema
include         /etc/openldap/schema/ppolicy.schema
include         /etc/openldap/schema/collective.schema
include         /etc/openldap/schema/krb5-kdc.schema
```

Gambar 4.7 Pendefinisian schema

Beberapa file schema yang biasanya disertakan dalam setiap instalasi OpenLDAP adalah :

1. *Corba.schema*

   Skema untuk menyimpan objek dari corba.

2. *Core.schema*

   Skema utama dari OpenLDAP. Skema ini mendefinisikan atribut dan objek dasar dari LDAPv3.

3. *Cosine.schema*

   Skema untuk mendukung direktori cosine dan X.500.

4. *Inetorgperson.schema*

   Skema yang mendefinisikan objectclass InetOrgPerson dan atribut yang diasosialisasikan. Biasanya untuk menyimpan informasi kontak dari orang lain.

5. *Java.schema*

   Skema untuk menyimpan objek serial dari java, objek java marshaled, objek remote java, atau referensi JDNI.

6. *Misc.schema*

   Skema yang mendefinisikan sebuah grup kecil dari objek dan atribut lainnya. Sekarang ini, file ini berisi skema yang dibutuhkan untuk mengimplementasikan routing email dalam sendmail yng berbasis LDAP.

7. *Nis.schema*

   Skema yang mendefinisikan atribut dan objek yang dibutuhkan untuk menggunakan LDAP dangen Network Information Service (NIS).

8. *Openldap.schema*

   Objek-objek lain yang digunakan oleh proyek OpenLDAP. Digunakan untuk informasi.

4.1.3.4 Penyedia Data

Setelah bagian utama dari *slapd.conf,* dilanjutkan satu atau beberapa bagian dari database, masing-masing mendefinisikan sebuah partisi direktori. Sebuah bagian

database dimulai dengan direktif database dan seterusnya sampai ditemukan database selanjutnya. Parameter untuk database yang mungkin adalah :

1. Bdb

   Menggunakan database dari Berkeley DB 4. *Backend* ini sering digunakan untuk *indexing* dan *caching* yang akan meningkatkan performa, *backend* ini paling direkomendasikan untuk digunakan pada server OpenLDAP.
2. Ldbm

   Sebuah database yang diimplementasikan oleh GNU database manager atau paket *software* Sleepycat Berkeley DB. *Backend* ini implementasi yang lebih tua dari *backend bdb.*
3. Passwd

   Backend ini menggunakan sistem file passwd untuk memberikan sebuah antarmuka direktori.
4. Sql

   Backend ini menggunakan databse yang berbasis SQL. Databse yang didukung antara lain MySQL, Oracle, MSSQL, IBMDB2, PosgreSQL, dan timesten.
5. Shell

   Backend shell memperbolehkan untuk menggunakan database alternative (eksternal). Direktif ini mengijinkan kita untuk menentukan program eksternal yang dipanggil untuk masing-masing operasi dasar LDAPv3.

Gambar 4.8 menunjukkan contoh pendefinisian *backend.* Dari gambar tersebut dapat diketahui bahwa *backend* yang digunakan adalah bdb yaitu Berkeley DB 4 sesuai keterangan sebelumnya. Pada gambar 4.8 juga terdapat keterangan

mengnai *userid* yang bertindak sebagai root pada server LDAP. Dari gambar 4.8 tersebut didefinisikan bahwa yang bertindak sebagai rootdn adalah ldapadm pada *realm auth.langhua* menggunakan mekanisme otentikasi GSSAPI.

```
#######################################################################
# ldbm and/or bdb database definitions
#######################################################################

database        bdb
suffix          "dc=auth,dc=langhua"
rootdn          "uid=root,cn=auth,langhua,cn=gssapi,cn=auth"
# suffix                "dc=my-domain,dc=com"
# checkpoint    1024 15
# rootdn                "cn=Manager,dc=my-domain,dc=com"
# Cleartext passwords, especially for the rootdn, should
# be avoided.  See slappasswd(8) and slapd.conf(5) for details.
# Use of strong authentication encouraged.
# rootpw                secret
# rootpw                {SSHA}o/zDZrYFi9Zoz0gNO9OlWpNwz+WlHYuf
```

Gambar 4.8 Pendefinisian backend

Parameter index menyatakan atribut apa pada *slapd* yang seharusnya dipelihara indexnya. Index digunakan untuk mengoptimalkan pencarian, hampir sama seperti index yang digunakan oleh database relasional.

4.1.3.5 Access Control List (ACL)

ACL direktori yang diberikan oleh OpenLDAP mempunyai sintaks yang sederhana, tetapi sangat fleksibel dan kuat dalam implementasinya.

1. *Wildcard (*)*

   Apabila cocok dengan setiap pengguna yang tersambung, termasuk koneksi anonym.

2. *Self*

   Cocok dengan DN dari pengguna yang sekarang tersambung, diasumsikan jika pengguna tersebut telah sukses diotentikasi pada permintaan bind sebelumnya.

3. *Anonymous*

Cocok dengan bukan pengguna yang terotentikasi.

4. *User*

   Koneksi pengguna yang telah terotentikasi.

5. *Regular Expression (regex)*

   Cocok dengan sebuah DN atau sebuah identitas SASL. Hal yang perlu diingat bahwa nama login yang digunakan untuk menentukan sebuah pengguna diambil dari DN (misalnya: *dn="cnuser1,ou=people,dc=kemas,dc=com"*) atau sebuah identitas SASL (misalnya: *dn="uid=user1,cn=gssapi,cn=auth"*). Table 4.4 merangkum bermacam-macam hak akses. Level yang lebih tinggi memiliki semua kemampuan dari level dibawahnya.

Tabel 4.4 Level Akses

Level Access	Hak Akses
Write	Dapat mengubah nilai atribut.
Read	Dapat membaca hasil pencarian.
Search	Dapat melakukan pemfilteran suatu pencarian.
Compare	Dapat membandingkan atribut.
Auth	Dapat melakukan bind (otentikasi).
None	Tidak mendapatkan akses.

Gambar 4.9 menunjukkan konfigurasi ACL pada server LDAP. Pada gambar dapat dilihat bahwa *uid=ldapadm* yang merupakan *user principal* Kerberos diberi hak *write* keseluruh entri direktori LDAP.

```
sasl-realm      AUTH.LANGHUA
sasl-host       auth.langhua

access to attrs=userPassword
        by self write
        by dn="uid=test,cn=auth.langhua,cn=gssapi,cn=auth" write
        by anonymous auth
        by * none

access to *
        by dn="uid=test,cn=auth.langhua,cn=gssapi,cn=auth" write
        by self write
        by * read
```

Gambar 4.9 Konfigurasi ACL

4.1.3.6 Konfigurasi SASL pada *slapd.conf*

File *slapd.conf* memiliki tiga bagian opsi berkenaan dengan SASL.

Sasl-host	*hostname*
Sasl-realm	*string*
Sasl-secprops	*properties*

Sasl-host merupakan *fully qualified domain name* dari host yang digunakan untuk otentikasi SASL yatiu *auth.langhua. Sasl-realm* merupakan domain SASL yang digunakan untuk otentikasi yaitu AUTH.LANGHUA. Parameter ketiga yaitu, *sasl-secprops* mengijinkan penentuan kondisi berbeda terhadap property SASL. Nilai yang mungkin untuk parameter ini digambarkan pada tabel 4.5

Tabel 4.5 Deskripsi Parameter sasl-secprops

Flag	Deskripsi
None	Pengaturan properti keamanan secara default (*noplain, noanonymous).*
Noplain	Mematikan mekanisme-mekanisme serangan pasif.
Noactive	Mematikan mekanisme-mekanisme penyerangan aktif.
Noanonymous	Mematikan mekanisme-mekanisme yang mendukung login secara

	anonym.
Frowardsec	Mekanisme-mekanisme yang melewatkan credential dari *client.*
Passcred	Mekanisme-mekanisme yang melewatkan *credential* dari *client.*
Minssf=factor	Menentukan kekuatan keamanan minimum. Nilai yang mungkin yaitu; 0 (tidak ada penjagaan), 1 (penjagaan integritas saja), 56 (membolehkan enkripsi DES), 112 (mengijinkan 3DES atau metode enkripsi string lainnya), dan 128 (mengijinkan RC4, Bowlfish, atau algoritma enkripsi dari kelas ini).
Maxssf=factor	Menentukan pengaturan keamanan maximum. Nilai yang mungkin identik seperti pada minssf.
Maxbufsize=size	Menentukan ukuran maksimum dari *layer* keamanan untuk buffer. Nilai 0 mematikan *layer* keamanan. Nilai default yaitu nilai maksimum dari INT_MAX (sebagai contoh 65536).

Pada parameter *sasl-secprop* ini, penentuan nilainya boleh menggunakan banyak kombinasi nilai. Properti default adalah *noanonymous* dan *noplain.*

Pemahaman menyeluruh mengenai parameter *sasl-secprop* juga diperlukan untuk hasil optimal untuk *plugin-plugin cyrus-sasl.* Tabel 4.6 berisi ringkasan mekanisme-mekanisme dan *flag* properti yang tersedia.

Tabel 4.6 Properti Mekanisme Otentikasi SASL

SASL Mechanism	Security Property Flags	Maxssf
ANONYMOUS	NOPLAIN	0
CRAM-MD5	NOPLAIN NOANONYMOUS	0

DIGEST-MD5	NOPLAIN NOANONYMOUS	128 jika dikompilasi dengan RC; 112 jika dikompilasi bersama DES; 0 jika dikompilasi dengan RC4 nor DES lainnya
GSSAPI	NOPLAIN NOACTIVE NOANONYMOUS	56
KERBEROS-V4	NOPLAIN NOACTIVE NOANONYMOUS	56
LOGIN	NOANONYMOUS	0
PLAIN	NOANONYMOUS	0
SCRAM-MD5	NONE	0
SRP	NOPLAIN	0

Penambahan baris-baris dibawah ini kebagian global dari file *slapd.conf* :

Sasl-secprops noplain, noanonymous, minssf=56

Apabila dibandingkan dengan nilai *sasl-secprops* dengan mekanisme pada tabel 4.6 menunjukkan bahwa server akan mengijinkan mekanisme-mekanisme dibawah ini untuk otentikasinya :

1. DIGEST-MD5
2. GSSAPI
3. KERBEROS_4

4.2 Hasil Penelitian dan Pembahasan

4.2.1 Pengujian Kerberos

Konfigurasi pada Kerberos perlu dilakukan apakah berhasil atau tidak. Pengujian dilakukan dengan perintah *kinit* yang akan memperoleh dan menyimpan sementara suatu inisial *Ticket Granting Ticket*. Untuk selanjutnya memasukkan *password* yang sesuai dengan *username*.

Gambar 4.10 kinit AUTH.LANGHUA

Dari gambar tersebut tampak bahwa *kinit* meminta TGT bagi ldapadm. Sebelumnya *user principal* ldapadm telah ditambahkan dalam database Kerberos. Apabila *password* yang dimasukkan sesuai maka *Ticket Granting Ticket* dapat diperoleh.

Perintah *kinit* akan menunjukkan daftar tiket-tiket Kerberos dan *principal* dalam *credential cache*, maupun key yang terapat dalam *keytab*.

Gambar 4.11 klist

Dari proses pengujian tersebut dapat diketahui bahwa TGT telah diperoleh melaui *kinit*. Dengan demikian berarti Kerberos telah dapat dijalankan.

4.2.2 Pengujian SASL

4.2.2.1 Penambahan LDAP Service

Penambahan *ldap service* diperluan agar server LDAP dapat menerima mekanisme GSSAPI yang ditawarkan. LDAP akan menjadi klien dari server SASL dimana mekanisme yang akan diberikan adalah GSSAPI.

Addprinc –randkey ldap/auth.langhua@AUTH.LANGHUA

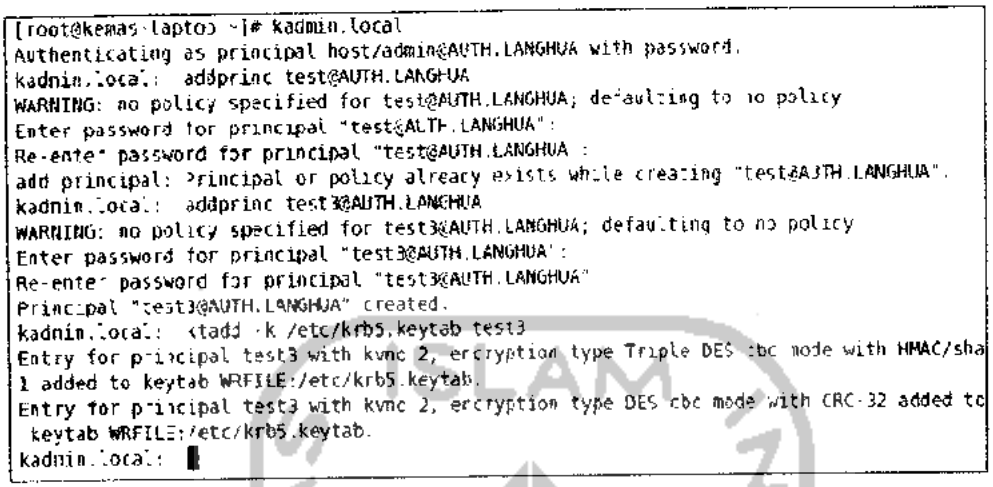
Perintah di atas akan menambahka *service principal* yaitu LDAP. *Service principal* LDAP yang telah terbentuk harus ditambahkan dalam *keytab* milik SASL yaitu krb5.keytab yang terbentuk pada /etc/krb5.keytab.

Ktadd –k /etc/krb5.keytab ldap/auth.langhua@AUTH.LANGHUA

4.2.2.2 Penambahan *User Principal test*

User principal test akan digunakan sebagai user dalam pengujian SASL GSSAPL. Penambahan *user principal* test akan dilakukan dengan perintah addprinc test@AUTH.LANGHUA. Langkah selanjutnya adalah menambahkannya ke dalam *keytab* dengan perintah ktadd –k /etc/krb5.keytab test.

Gambar berikut ini memperlihatkan proses penambahan *user principal test.*



Gambar 4.12 Penambahan *user principal* test

Langkah berikutnya adalah memperoleh tiket untuk *principal test* dengan *kinit-k test.*

4.2.2.3 Pengujian GSSAPI

Pengujian GSSAPI diperlukan untuk mengetahui apakah mekanisme SASL GSSAPI telah berhasil dikonfigurasikan. Pengujian menggunakan program *sample – server* dan *sample-client.*

Dari salah satu terminal perintah *sasl2-sample-server –s ldap –m GSSAPI.* Secara garis besar –s pada perintah tersebut menunjukkan *service* atau layanan yaitu LDAP. Kemudian mengidentifikasi mekanisme otentikasi yang akan digunakan yaitu GSSAPI, sehingga maksud dari perintah di atas adalah pengujian server SASL terhadap klien dalam hal ini adalah layanan LDAP apakah mekanisme otentikasi SASL GSSAPI yang ditawarkan berhasil dikonfigurasikan dengan baik atau tidak.

Gambar berikut ini menggambarkan tentang pengujian *sample-server* dari SASL GSSAPI yang dilakukan.

```
[root@kemas-laptop ~]# sasl2-sample-server -s ldap -m GSSAPI
trying 2, 1, 6
trying 10, 1, 6
bind: Address already in use
```

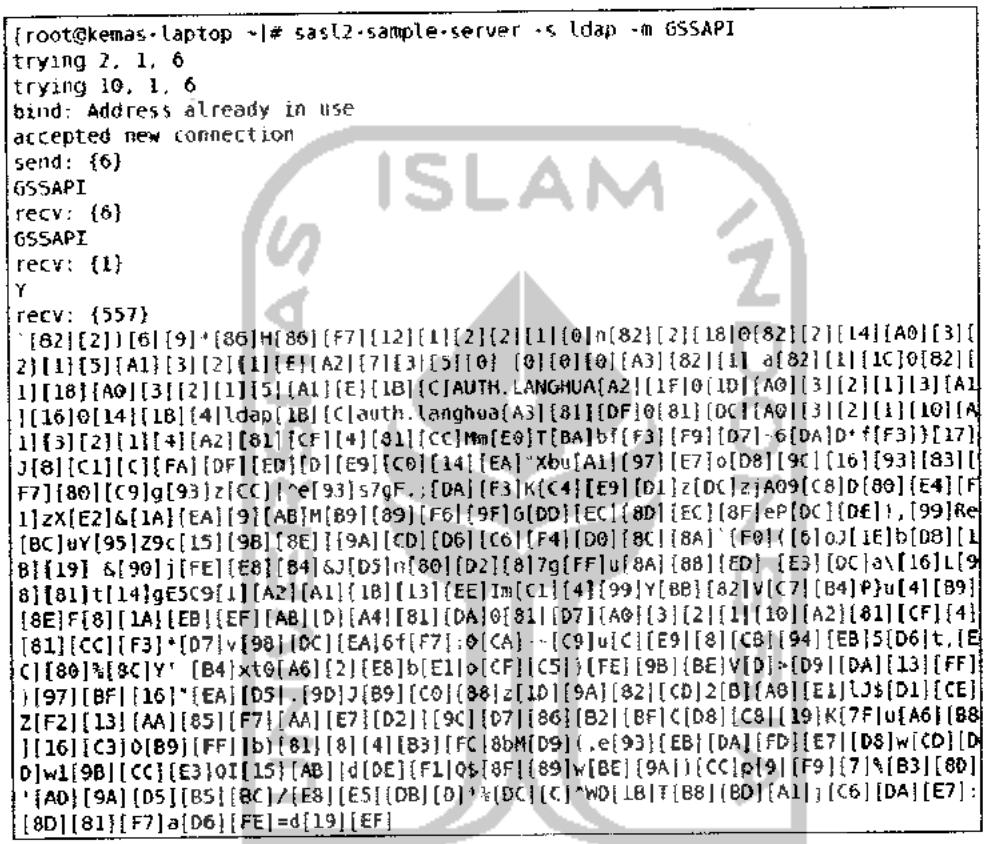
Gambar 4.13 Pengujian *sample-server* SASL GSSAPI

Dari terminal yang lain digunakan perintah *sasl2-sample-client –s ldap –m GSSAPI AUTH.LANGHUA* yang akan berkomunikasi dengan *sample-server* yang telah dijalankan sebelumnya. AUTH.LANGHUA dimasukkan dalam perintah diatas sebagai petunjuk *host* yang akan menjalankan *service*. *Userid* test dimasukkan ketika permintaan *userid* untuk otorisasi. Tentunya setelah user test ketika telah memperoleh TGT saat *kinit* berhasil dijalankan.

```
[root@kemas-laptop ~]# sasl2-sample-client -s ldap -m GSSAPI auth.langhua
receiving capability list... recv: {6}
GSSAPI
GSSAPI
please enter an authorization id: test
send: {6}
GSSAPI
send: {1}
Y
send: {557}
`[82][2]){6][9]*[86]H[86][F7][12][1][2][2][1][0]n[82][2][18]0[82][2][14][A0][3]{[
2][1][5][A1]{3][2][1]{E][A2][7][3][5][0] [0][0][0][A3][82][1] a[82][1][1C]0[82][[
1][18][A0][3][2][1][5][A1][E][1B][C]AUTH.LANGHUA[A2][1F]0[1D][A0][3][2][1][3][A1
][16]0[14][1B][4]ldap[1B][C]auth.langhua[A3][81][DF]0[81][DC][A0][3][2][1][10][A
1][3][2][1][4][A2][81][CF][4][81][CC]Mm[E0]T[BA]bf[F3][F9][D7]~6[DA]D*f[F3]}[17]
J[8][C1][C][FA][DF][ED][D][E9][C0][14][EA]"Xbu[A1][97][E7]o[D8][9C][16][93][83][
F7][80][C9]g[93]z[CC]{^e[93]s7gF.;[DA][F3]K[C4][E9][D1]z[DC]zjA09[C8]D[80][E4][F
1]zX[E2]&[1A][EA][9][AB]M[B9][89][F6][9F]G[DD][EC][8D][EC][8F]eP[DC][DE]!,[99]Re
[BC]uY[95]Z9c[15][9B][8E][9A][CD][D6][C6][F4][D0][8C][8A] [F0]([6]oJ[1E]b[D8][1
B][19] &[90]][FE][E8][B4]&J[D5]n[80][D2][8]7g[FF]u[8A][88][ED] [E3][DC]a\[16]L[9
8][81]t[14]gE5C9[1][A2][A1][1B][13][EE]Im[C1][4][99]Y[BB][82]V[C7][B4]P}u[4][B9]
[8E]F[8][1A][EB][EF][AB][D][A4][81][DA]0[81][D7][A0][3][2][1][10][A2][81][CF][4]
[81][CC][F3]*[D7]v[98][DC][EA]6f[F7]:0[CA]~[C9]u[C][E9][8][C8][94][EB]5[D6]t,[E
C][80]%[8C]Y' [B4]Xt0[A6][2][E8]b[E1]o[CF][C5])[FE][9B][BE]V[D]>[D9][DA][13][FF]
)[97][BF][16]"[EA][D5].[9D]J[B9][C0][88]z[1D][9A][82][CD]2[B][A8][E1]lJ$[D1][CE]
Z[F2][13][AA][85][F7][AA][E7][D2]][9C][D7][86][B2][BF]C[D8][C8][19]K[7F]u[A6][B0
][16][C3]O[B9][FF]]b)[81][8][4][B3][FC]8bM[D9](.e[93][EB][DA][FD][E7][D8]w[CD][D
D]w1[9B][CC][E3]OI[15][AB]|d[DE][F1]Q$[8F][89]w[BE][9A])[CC]p[9][F9][7]%[B3][8D]
'[AD][9A][D5][B5][BC]/[E8][E5][DB][D]*%[DC][C]^WD[1B]T[B8][BD][A1])[C6][DA][E7]:
[8D][81][F7]a[D6][FE]=d[19][EF]
```

Gambar 4.14 Pengujian SASL GSSAPI *sample-client*

Gambar 4.14 menggambarkan komunikasi antara *sample-server* dan *sample-client* yang ada disisi *client*. Negosiasi antara server dan *client* terjalin sehingga otentikasi berhasil ataukah sebaliknya.



Gambar 4.15 Pengujian SASL GSSAPI *sample-server*

Gambar diatas menunjukkan komunikasi pengujian dari sisi server. Jika telah diperoleh keterangan *succesfull authentication* di akhir komunikasi antara server dan *client*, maka konfigurasi SASL GSSAPI telah berhasil dan didukung oleh LDAP.

4.2.3 Pengujian LDAP

Salah satu operasi *client* LDAP adalah *ldapsearch* yang digunakan dengan tujuan mencari entri-entri dalam direktori LDAP. Gambar dibawah ini menunjukkan operasi LDAP.

Dari gambar tersebut dapat diketahui bahwa mekanisme otentikasi yang digunakan adalah SASL GSSAPI dengan *username* . *Username* merupakan *user principal* Kerberos yang telah memperoleh TGTsebelumnya. Hasil operasi *ldapsearch* diatas dapat dilihat lebih lengka dalam lampiran. Apabila server LDAP tidak mendukung mekanisme SASL GSSAPI, maka otentikasi akan gagal.

```
[kemas@kemas-laptop ~]$ ldapsearch -x -b 'dc=auth,dc=langnua'
# extended LDIF
#
# LDAPv3
# base <dc=auth,dc=langnua> with scope subtree
# filter: (objectClass=*)
# requesting: ALL
#

# auth.langnua
dn: dc=auth,dc=langnua
objectClass: dcObject
objectClass: organization
o: Home LDAP Server
dc: auth

# Manager, auth.langnua
dn: cn=Manager,dc=auth,dc=langnua
objectClass: organizationalRole
cn: Manager

# users, auth.langnua
dn: ou=users,dc=auth,dc=langnua
ou: users
objectClass: top
objectClass: organizationalUnit

# addressbook, auth.langnua
dn: ou=addressbook,dc=auth,dc=langnua
ou: addressbook
objectClass: top
objectClass: organizationalUnit

# search result
search: 2
result: 0 Success
```

Gambar 4.16 Operasi *ldapsearch*

4.2.4 Keamanan Pengaksesan Direktori

Status akses ke direktori dapat diketahui melalui perintah *ldapwboami*. Otentikasi yang dilakukan dengan mekanisme simple bind menggunakan entri-entri yang ada dalam server LDAP sebagai usernya.

Operasi *ldapwboami* akan dijalankan oleh *user1* yang merupakan salah satu *entry* dalam direktori LDAP. Sebelumnya *user1* harus mengotentikasi dirinya menggunakan mekanisme *simple bind –W* akan meminta *user1* memasukkan *password.*

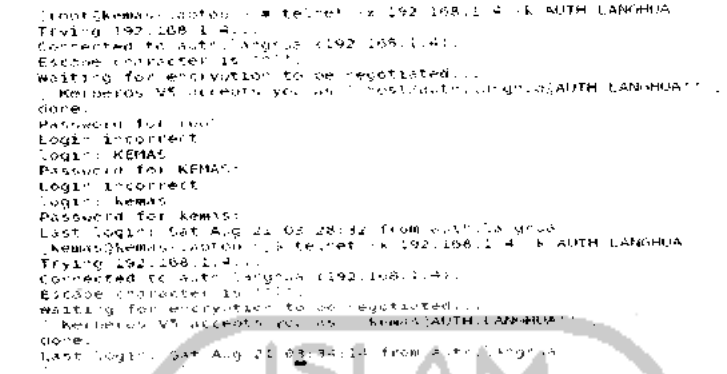
4.2.5 Perbandingan Sistem Otentikasi LDAP Menggunakan Kerberos dan Tanpa Kerberos

Pengujian dan perbandingan dilakukan pada sistem otentikasi server LDAP. Sistem yang pertama kali diuji adalah sistem otentikasi untuk mengakses server LDAP tanpa menggunakan Kerberos yaitu menggunakan *simple bind*, sedangkan sistem yang selanjutnya diuji adalah sistem otentikasi LDAP menggunakan otentikasi yang ditawarkan Kerberos dengan modul SASL GSSAPI. Kedua sistem diuji pada lingkungan pengujian yang sama. Tabel 4.7 memperlihatkan hasil pengujian dan perbandingan diantara kedua sistem.

Tabel 4.7 Perbandingan Sistem Otentikasi LDAP dengan Kerberos dan tanpa Kerberos

Keterangan	Otentikasi LDAP Tanpa Kerberos	Otentikasi LDAP Menggunakan Kerbros
Mekanisme otentikasi ke direktori LDAP	*Simple bind*	SASL, GSSAPI sebagai modul otentikasi tambahan.
Keamanan akses ke direktori LDAP	Kurang aman karena *password* masih dilewatkan ke dalam jaringan.	Lebih aman karena *password* tidak ditransmisikan dalam jaringan

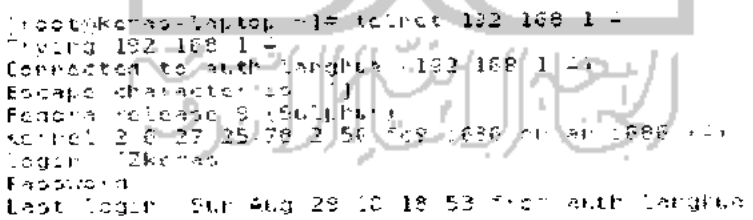
Berikut merupakan hasil pengujian keamanan akses ke direktori LDAP, yaitu dengan menggunakan perintah telnet. Bagian pertama dilakukan operasi telnet pada direktori LDAP yang menggunakan Kerberos.



Gambar 4.17 Operasi telnet menggunakan Kerberos

Gambar 4.18 menerangkan pengujian server LDAP yang menggunakan Kerberos. Operasi yang digunakan untuk pengujian adalah *telnet –x 192.168.1.4 –k AUTH.LANGHUA*, yaitu operasi untuk masuk ke computer lain dalam suatu jaringan. Lebih rinci operasi diatas adalah telnet meminta untuk masuk ke dalam jaringan IP 192.168.1.4 sebagai server untuk mengakses direktori dari auth.langhua. Hasil pengujian diatas memperlihatkan bahwa telnet tidak mendapatkan informasi apapun dari direktori auth.langhua yang memakai *protocol Kerberos* sebagai pengamannya.

Berikut merupakan hasil operasi telnet pada direktori LDAP yang tidak menggunakan Kerberos.



Gambar 4.18 Operasi telnet tanpa menggunakan Kerberos

Gambar 4.19 menerangkan pengujian server LDAP yang tidak menggunakan Kerberos. Operasi yang digunakan untuk pengujian adalah *telnet 192.168.1.4*. Telnet meminta untuk masuk ke jaringan IP 192.168.1.4 sebagai server LDAP. Hasil

pengujian diatas memperlihatkan bahwa telnet mendapatkan informasi system yang digunakan server LDAP.

Dari hasil pengujian diatas dapat disimpulkan bahwa operasi telnet pada direktori LDAP yang menggunakan Kerberos lebih aman karena enkripsi data berjalan dalam sistem server dibandingkan dengan LDAP yang tidak menggunkan Kerberos.

Tabel 4.7 juga menunjukkan bahwa otentikasi yang ditawarkan Kerberos meningkatkan keamanan pengaksesan direktori LDAP walaupun disisi lain kecepatan akses ke direktori LDAP yang menggunakan Kerberos relative lebih lama dibandingkan dengan sistem yang tanpa Kerberos.

4.2.6 Perbandingan Kerberos dengan Protokol Lain

Tabel dibawah ini merupakan perbandingan antara protokol Kerberos dengan protokol Radius, beberapa data dibawah ini didapat dari referensi yang dirujuk dari beberapa sumber diantaranya www.ilkom.unsri.ac.id.

Tabel 4.8 Perbandingan Kerberos dengan Radius

Radius	Kerberos
Sistem bekerja lambat, karena Radius bekerja dengan 3 konsepnya. Pertama otentikasi, memastikan apakah *client* terdaftar dalam jaringan. Kedua otorisasi, mengetahuai hak akses pada *client*. Ketiga akunting, merupakan pendaftaran *account* apakah *client* sah atau tidak di Radius.	Kerberos menawarkan satu kenyamanan pada user, hanya dengan memasukkan password sekali dan meminta ticket (TGT) pada TGS, maka kita bisa meminta layanan pada banyak akun yang kita inginkan. Selain itu, ticket tersebut berlaku dalam periode waktu yang pendek sampai masa expirednya
Skema proteksi yang dipakai adalah *stream-chiper*. Hal ini memungkinkan penyusup mendapatkan informasi *shared secret* apabila mereka melakukan sniffing kejaringan *wireless* dan mencoba masuk ke *Radius server*.	Tingkat keamanannya tinggi. *Password* tidak dikirimkan melintasi jaringan.
Tidak adanya autentikasi dan verifikasi terhadap *access request*.	Kerberos bersifat *transparent*. *User* tidak perlu mengetahui tahap-tahap otentikasi yang dilakukan di dalam jaringan. *User* hanya tinggal*login* ke jaringan melalui program inisialisasi kinit, memasukkan*username* dan *password*, lalu *user* memperoleh otentikasi ke *server* yang dituju.
Radius tidak menyediakan layanan SSH.	Kerberos menyediakan layanan SSH yang dapat mengelola server jarak jauh.

BAB 5 KESIMPULAN DAN SARAN

5.1 Kesimpulan

1 Server LDAP menggunakan Kerberos berhasil dibangun dengan *Krb5, OpenLDAP-2.4.15, Cyrus-SASL.*

2 DNS harus disetting terlebih dahulu sebelum membangun sistem LDAP menggunakan Kerberos.

3 Berdasarkan pengujian pada waktu implementasi, maka dapat diambil kesimpulan bahwa Kerberos dengan SASL, GSSAPI meningkatkan keamanan otentikasi saat mengakses direktori LDAP, akan tetapi memerlukan waktu yang lebih lama dibandingkan dengan sistem otentikasi yang tidak menggunakan Kerberos.

5.2 Saran

Pengembangan sistem berikutnya diharapkan mampu menggabungkan Kerberos dan LDAP untuk dasar solusi *Single sign on* dengan banyak aplikasi dalam suatu jaringan yang lebih luas. Dengan LDAP sebagai *backend user* aplikasi dan Kerberos sebagai pihak ketiga untuk server otentikasi.