PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK WINDOWS *DOMAIN* LOGIN DALAM SISTEM OPERASI LINUX DEBIAN MENGGUNAKAN ACTIVE DIRECTORY DAN WINBIND

TUGAS AKHIR

Disusun Oleh :

FANNI ICHWAN FAIZAL
NRP. 5101 100 034

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2006

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK WINDOWS *DOMAIN* LOGIN DALAM SISTEM OPERASI LINUX DEBIAN MENGGUNAKAN ACTIVE DIRECTORY DAN WINBIND

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer
Pada
Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui / Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
Febriliyan Samopa, M.Kom.	Royyana Muslim Ijtihadie, S.Kom.
NIP. 132 206 858	NIP. 051 100 001

SURABAYA
FEBRUARI 2006

ABSTRAK

Linux adalah Sistem Operasi (OS) yang saat ini sudah mulai populer dan dikenal luas. Salah satu hal yang membuat sistem operasi Linux ini terkenal adalah karena sistem operasi Linux ini adalah gratis. Dan juga bagi para pengembang sistem operasi ini disukai karena dengan menggunakan sistem operasi Linux. Pengembang diberi kebebasan bereksperimen yang tidak didapatkan karen Windows. Mulai dari konfigurasinya yang sangat bermacam-macam dan luas. Hingga source aplikasi yang dimilikinya juga termasuk dalam open source community. Sehingga para pengembang dapat saling bekerja sama untuk belajar dan mengembangkan aplikasi.

Namun penggunaan Linux pada umumnya menggunakan login lokal untuk memakai komputer. Sehingga hanya para user yang terdaftar yang dapat menggunakan komputer itu. Hal ini pada awalnya tidak bermasalah. Namun dalam perkembangannya jumlah pemakai Linux bertambahdan jumlah komputer yang menggunakan Linux juga bertmbah. Sehingga sudah banyak ditemui dimana dalam suatu domain banyak ditemui komputer dengan sistem operasi Linux namun tidak terhubung dengan domain.

Domain merupakan solusi atas metode untuk pengaturan dan administrasi komputer dengan skala besar. Dimana banyak komputer yang dijadikan client dapat dikendalikan dan dikontrol lebih mudah lebih efisien.

Dalam aplikasi ini. Client dengan OS Linux dapat melakukan login dengan menggunakan login domain Windows Server 2003 dan Login yang ada dalam client itu sendiri.

Dan setelah aplikasi ini dijalankan, client dapat melakukan login dan dapat menjalankan aplikasi dalam client. Dan sistem otentikasi ini juga aman dari usaha pembobolan karena memiliki tingkat keamanan yang sama dengan client Windows.

Kata kunci : OS, login, domain, login

KATA PENGANTAR

Segala puji syukur penulis panjatkan ke hadirat Allah SWT, karena hanya dengan kehendak dan kuasa-Nya, penulis dapat menyelesaikan Tugas Akhir yang berjudul:

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK WINDOWS *DOMAIN* LOGIN DALAM SISTEM OPERASI LINUX DEBIAN MENGGUNAKAN ACTIVE DIRECTORY DAN WINBIND

Tugas Akhir ini dibuat guna memenuhi persyaratan akademik dalam rangka ujian akhir bagi mahasiswa Strata 1 (S1) Jurusan Teknik Informatika , Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Dalam penyusunan Tugas Akhir ini penulis telah berusaha sebaik-baiknya, akan tetapi tetap masih memiliki banyak kekurangan. Karena itu penulis mengharapkan adanya kritik dan saran yang membangun guna menambah manfaat serta mengurangi kesalahan dan kekurangan yang ada.

Pada akhirnya penulis berharap semoga laporan ini dapat memberikan manfaat bagi kita semua.

Surabaya, Januari 2006

Penulis

UCAPAN TERIMA KASIH

Pada Kesempatan kali ini,penulis ingin mengucapkan terima kasih yang besar-besarnya atas bantuan dan dukungan yang tak ternilai kepada :

1. Orang tuaku tercinta. Terima kasih atas doa, pengorbanan, kesabaran, tanggung jawab, perhatian, kasih sayang dan dorongan semangat selama ini. Hanya berkat doa dan bimbinganmulah, penulis dapat menyelesaikan Tugas Akhir ini.
2. Bapak Febriliyan Samopa selaku dosen pembimbing I. Penulis mengucapkan terima kasih dan penghormatan yang setinggi-tingginya atas bantuan, bimbingan, saran dan masukan selama pengerjaan Tugas Akhir ini.
3. Bapak Royyana M. Ijtihadie selaku dosen pembimbing II, yang senantiasa memberikan bimbingan, saran dan masukan dalam pengerjaan Tugas Akhir ini.
4. Bapak Muhammad Husni selaku dosen wali. Terima kasih atas bimbingan dan kesabaran yang telah diberikan dan pertolongannya selama mengikuti perkuliahan.
5. Bapak Yudhi Purwananto selaku Kajur Teknik Informatika. Terima kasih atas pengabdiannya pada jurusan.
6. Dosen-dosen Penguji TA Bapak Arunanto, Bapak Wahyu Suadi, Bapak Shidiq. Terima kasih atas saran dan bimbingannya yang telah diberikan untuk penulis.

7. Seluruh Staf Dosen Pengajar Teknik Informatika, Staf Dosen TPB. Terima Kasih atas Ilmu yang telah diberikan semoga bermanfaat dan semoga Allah SWT membalas jasa-jasa beliau.
8. Teman-teman C11 Kris, Sempal, dan Yeni yang telah lulus sebelum saya namun masih setia membantu dan mendukung saya selama pengerjaan TA ini.
9. Teman-Teman TA-ers di Lab AJK Rolly, Phe, Laudy, Rudi, Pras, dan Wafa. Terima kasih atas bantuan dan dukungannya untuk sesegera mungkin menyelesaikan kuliah dan lulus bersama-sama.
10. Teman-Teman TA-ers di Lab Komputing Baried dan Bolon. Terima Kasih atas bantuan dan fasilitas yang dipinjamkan semasa mengerjakan Tugas Akhir.
11. Teman-Teman Lab AJK Galih, Rico, Sokam, Cakso, Sinchan, Feri, Jian, Gandi, Acong, tiktok, Victor dan administrator AJK lainnya. Terima kasih atas bantuan dan kerja kerasnya pada Lab.
12. Seluruh kawan-kawan C11 baik yang sudah lulus ataupun yang belum. Terima kasih atas kebersamaan dan bantuannya semasa penulis kuliah.
13. Seluruh kawan-kawan C0E,C0F,C10,C12,C13,C14 baik yang sudah lulus maupun yang masih kuliah. Terima kasih atas kebersamaan dan bantuannya buat penulis semasa kuliah.
14. Mas Yudhi di tata usaha yang telah membantu penulis dalam masalah administrasi akademis selama kuliah.
15. Seluruh karyawan tata usaha dan ruang baca yang telah banyak membantu penulis selama kuliah. Pak Narno, Mbak Fatin, Mbak Eva, Mas Sugeng, Mas Gayuh, Mas Hermono serta karyawan yang lain.

16. Satpam kampus TC, Pak Moko, Pak Karmono, Pak Bagio, Pak Jarwo. Terima kasih telah memberikan keamanan selama kuliah.

Pihak-pihak lain yang tidak dapat disebutkan disini. Terima kasih atas bantuannya baik pada saat penulis aktif dalam kegiatan perkuliahan maupun diluar kegiatan perkuliahan.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

BAB 1

PENDAHULUAN

Bab ini akan menjelaskan latar belakang, permasalahan yang diangkat sebagai bahasan utama, serta tujuan yang ingin dicapai dari pembuatan Tugas Akhir ini. Selain itu bab ini juga mendefinisikan batasan masalah, metodologi pembuatan Tugas Akhir dan sistematika pembahasan keseluruhan Tugas Akhir.

1.1. LATAR BELAKANG

Dibandingkan dengan Windows, sistem operasi Linux memiliki beberapa kelebihan antara lain, sistem konfigurasi yang membebaskan *user* (root) untuk merubah secara teks sehingga menghasilkan perubahan yang lebih dalam dan detail sehingga dapat lebih berkreasi, pendokumentasian manual yang lengkap dari tiap-tiap konfigurasi yang dapat dilakukan, dan secure shell (ssh) yang membuat *remoting* membutuhkan *bandwith* yang tidak terlalu besar karena hanya melalui teks. Karenanya dengan fasilitas ini, sistem operasi Linux mendapat perhatian khusus bagi para pengembang, terutama untuk bidang jaringan komputer. Dan menjadi bagian tersendiri dalam perkembangan dunia IT saat ini.

Dengan fitur yang ditawarkan oleh Linux seperti di atas, *user* dari Linux mayoritas berada suatu jaringan baik sebagai *server* ataupun *client*. Dan tidak menutup kemungkinan *user* dari sistem operasi Linux dalam suatu jaringan yang dikelola dan diorganisir menggunakan fasilitas jaringan Microsoft (Windows *Server*).

Dalam suatu jaringan dimana jumlah pemakai fasilitas komputer (*user*) besar, *administrator* tidak dapat mengontrol tiap-tiap *user* pemakai komputer. *Administrator* dapat mengunakan fasilitas Windows *Server* yaitu *domain login* untuk mengatur *login* masing-masing. Sehingga setiap ada *user* yang ingin menggunakan komputer dalam *domain* tersebut, dapat menggunakan *login domain*-nya sendiri yang telah terdaftar dalam *domain server*.

Inilah yang menjadi salah satu kelemahan Linux, sebab hal tersebut hanya berlaku apabila sistem operasi *client* yang digunakan oleh *user* adalah Microsoft Windows. Karena konfigurasi *login* bawaan dari linux adalah *login workstation* yaitu *user login* yang digunakan merupakan *login* masing-masing *client*/workstation, sehingga tiap *user* yang akan menggunakan *client* tersebut harus memiliki login di *client* tersebut untuk login di *client* itu.

Untuk mengantisipasi hal di atas, maka diperlukan suatu cara agar *user* dapat menggunakan *domain account*-nya yang berada di Windows *Server* untuk *login* di *client* yang menggunakan sistem operasi linux. Agar *user* tidak perlu harus memiliki *login* di *client* tersebut, hal ini dibutuhkan untuk mengurangi beban *administrator* dalam mengatur login *user*. Namun hal ini tidak dapat dilaksanakan secara langsung dari instalasi Linux, mengingat Linux dan Windows *Server* memiliki *platform* yang berbeda.

Namun dengan adanya Active Directory (AD) dari Microsoft dan Winbind dari Linux, hal di atas memungkinkan. Dengan adanya *service* AD, hubungan, kerjasama, dan keamanan Windows dengan sistem operasi lain meningkat. Sehingga informasi *user account* yang disimpan dalam *server* dengan sistem

operasi Windows *Server* 2003 dapat diakses oleh *client* dengan sistem operasi selain Windows. Dan dengan adanya Winbind, dengan konfigurasi yang tepat ditambah dengan *user* dari smb. *Client* dengan sistem operasi Linux dapat membuka koneksi dengan *server* yang menggunakan sistem operasi Windows *Server* 2003 dan membaca informasi *user domain account* yang disimpan dalam *server* tersebut.

1.2. PERMASALAHAN

Permasalahan yang diangkat dalam tugas akhir ini adalah bagaimana mengkonfigurasi Linux Debian untuk memiliki kemampuan:

1. Dalam menerima *input login*, Linux Debian dapat meminta *input user login workstation* atau *user domain login.*
2. Linux mampu membuka hubungan dengan Windows Server 2003 menggunakan fasilitas Samba.
3. Linux Debian mampu membaca AD yang dimiliki oleh *domain server* yang menggunakan Windows Server 2003.
4. Linux Debian dapat menerima *user domain login* dan membuat *folder* sebagai *home* direktori *user* yang login menggunakan login domain, dan mengambil data-data *user* tersebut dari *server* domain.

1.3. TUJUAN

Tujuan dari pembuatan tugas akhir ini ialah dengan memanfaatkan Microsoft AD, informasi *user domain account* yang disimpan dalam Windows *Server* 2003 dapat diberikan kepada *client* dengan Sistem Operasi Linux Debian. Dan dengan

menggunakan Winbind dan smb, *client* tersebut dapat membuka koneksi dan meminta informasi *user domain account* yang tersimpan dalam AD *account database* sebagai otentikasi. Sehingga hasil akhirnya untuk menghasilkan sebuah *client* dengan sistem operasi Linux Debian dengan konfigurasi *login* yang mampu meminta informasi *user account* dari *login workstation* yang tersimpan dalam *client* dan juga *login domain* yang disimpan dalam *server* yang menggunakan sistem operasi Windows *Server* 2003.

1.4. BATASAN MASALAH

Dari permasalahan-permasalahan diatas, maka batasan masalah dalam tugas akhir ini ialah:

1. Sistem Operasi yang digunakan oleh *client* dalam tugas terakhir ini adalah Linux Debian Sarge dan yang digunakan oleh *server database domain* adalah Windows *Server* 2003. Sehingga *user* dari Sistem operasi selain yang tertulis kemungkinan besar akan membutuhkan konfigurasi yang berbeda.
2. Linux Debian sebagai *client* memiliki koneksi dengan AD milik Windows *Server* 2003 yang memiliki *database domain user account* hanya sebatas untuk memverifikasi *domain account* yang digunakan untuk *login* di Linux Debian.
3. Dalam *client* telah dipersiapkan suatu *account guest* yang memiliki akses hanya sebatas *read-only* dan memiliki akses terhadap *mount drive*

yang terbatas. *Account* ini digunakan untuk percobaan *user* yang memiliki *account domain* tapi tidak memiliki *account workstation.*

Untuk penggabungan dengan *domain* jaringan setelah terbentuk koneksi dengan AD dan Kerberos telah dikonfigurasi menggunakan perintah Linux net.

1.5. METODOLOGI PEMBUATAN TUGAS AKHIR

Pembuatan tugas akhir ini terbagi menjadi beberapa tahapan pengerjaan, yaitu:

1. **Studi Literatur dan Perangkat Lunak**

   Studi literatur dilakukan dilakukan untuk mempelajari konsep dari teknologi yang digunakan. Informasi diperoleh baik melalui buku maupun referensi dari *internet*. Informasi yang dicari pada tahap ini antara lain mencari referensi di internet tentang petunjuk *user* dari PAM, SAMBA, winbindd, Kerberos, dan shadow.

2. **Perancangan Aplikasi**

   Pada tahap ini dilakukan perencanaan dan perancangan sistem login seperti apa yang kira-kira akan dihasilkan terutama masalah input login yang terbagi dalam dua bagian. Dan dari studi literatur yang telah dilaksanakan dapat direncakan seperti apa koneksi yang akan dibuat antara Linux Debian dengan Windows *Server* 2003. kemudian dapat diperkirakan informasi penting apa saja yang akan diambil dan digunakan.

3. **Implementasi**

   Pada tahap ini dilakukan implementasi dari yang telah dipelajari dalam studi literatur dan yang telah dirancang dalam tahap sebelumnya. Dengan

mengkombinasikan antara sistem login yang akan dibuat dengan sistem koneksi dengan Windows *Server* 2003 dan pertukaran informasi/data apa saja yang dapat dilewatkan dalam koneksi tersebut.

4. Uji Coba dan Evaluasi

Pada tahap ini, dengan asumsi implementasi sudah selesai, dilakukan uji coba terhadap workstation dengan sistem operasi Linux yang telah dikonfigurasi ulang sistem loginnya dalam suatu domain jaringan. Dan disesuaikan apakah telah berhasil untuk menyelesaikan permasalahan yang telah tertulis diatas dan sesuai ruang lingkupnya dengan batasan permasalah telah tertulis diatas. Dan apabila ditemui ketidaksesuaian maka akan dilakukan penyesuaian hingga sesuai.

5. Penyusunan Buku Tugas Akhir

Pada tahap terakhir ini akan disusun buku sebagai dokumentasi dari pelaksanaan Tugas Akhir. Dokumen ini juga diharapkan dapat berguna bagi orang lain yang memiliki keinginan untuk mempelajari dan mengembangkan sistem tersebut lebih lanjut. Terutama bagi para *administrator* yang ingin membuat *workstation* dalam jaringannya yang menggunakan sistem operasi Linux dapat menggunakan *login domain.*

1.6. SISTEMATIKA PEMBAHASAN

Sistematika penulisan tugas akhir ini dapat dijelaskan sebagai berikut:

BAB I Pendahuluan

Bab ini berisi latar belakang masalah, permasalahan, tujuan dan manfaat, batasan permasalahan, metodologi yang digunakan, dan sistematika penyusunan buku tugas akhir.

BAB II Dasar Teori

Bab ini berisi konsep paket UDP dan juga format paket DHCP beserta hal-hal yang berhubungan dalam pembuatan Tugas Akhir ini.

BAB III Perancangan Perangkat Lunak

Bab ini menjelaskan perancangan yang dibutuhkan untuk membuat aplikasi yang diinginkan beserta fasilitas-fasilitas yang disediakan dalam Tugas Akhir

BAB IV Implementasi Perangkat Lunak

Bab ini membahas implementasi terhadap desain yang dilakukan pada tahap sebelumnya, termasuk di dalamnya disertakan *script-script* yang merupakan inti dari aplikasi.

BAB V Uji Coba dan Evaluasi

Bab ini akan membahas tentang uji coba dan evaluasi yang dilakukan terhadap aplikasi yang telah saya buat dan juga optimasi beserta penanganan bug-bug yang mungkin timbul tanpa sepengetahuan saya.

BAB VI Kesimpulan dan Saran

Bab ini menjelaskan mengenai kesimpulan dan saran dari keseluruhan Tugas Akhir ini.

BAB 2

DASAR TEORI

BAB 2

DASAR TEORI

Pada bab ini akan dibahas beberapa pengertian umum dan konsep yang berkaitan dengan perancangan sistem otentikasi dan informasi mengenai komponen-komponen otentikasi dan pendukungnya baik di sistem operasi Linux dan Windows *Server* 2003. Pembahasan dimulai dari penjelasan sistem operasi, kelebihan dan kekurangan Linux dan Windows *Server* 2003, konsep dan fungsi Active Directory, penjelasan tentang SMB, Samba, winbind, PAM, kerberos, superuser do, Kerberos, shadow, dan openssh.

2.1. SISTEM OPERASI

Sistem operasi komputer dapat didefinisikan sebagai penghubung antara komputer dengan pemakai komputer. Fungsi dasar sistem operasi pada umumnya adalah *set up file sistem*, yaitu menulis *file* baru, memperbaiki *file* yang telah ada, menggandakan *file*, mengganti nama *file*, atau memindahkan *file*. Fungsi lain adalah memanggil dan menjalankan program dari pemakai, menyediakan sarana komunikasi antara komputer dengan perlengkapannya seperti *terminal*, *printer*, dan sarana penyimpan informasi seperti *tape* dan *disk*.

2.1.1 Linux

Linux adalah sistem operasi seperti UNIX, yang merupakan implementasi independen dari POSIX (Portable Operating Sistem Standard for Computer Environments), meliputi true-multitasking, virtual memory, shared libraries,

demand-loading, proper memory management, dan multiuser. Linux seperti layaknya UNIX, mendukung banyak software mulai dari TEX, X Window, GNU C/C++ sampai ke TCP/IP. Linux adalah sistem operasi yang disebarkan secara luas dengan gratis di bawah lisensi GNU General Public License (GPL), yang berarti juga *cource code* Linux tersedia.

Linux pertama kali dibuat oleh Linus Torvalds di Universitas Helsinki, terinspirasi dari Minix. Minix adalah sistem UNIX kecil yang dikembangkan oleh Andy Tanenbaum. Linux versi 0.02 hanya dapat menjalankan bash (GNU Bourne Again Shell) dan gcc (GNU C Compiler). Sekarang Linux adalah sistem UNIX yang lengkap, dapat digunakan untuk jaringan, pengembangan *software*, dan bahkan untuk sehari-hari. Linux sekarang merupakan alternatif sistem operasi yang jauh lebih murah jika dibandingkan dengan sistem operasi komersial, dengan kemampuan yang setara bahkan lebih.

Linux telah menjadi suatu tren terutama di kalangan pendidikan dan bisnis. Di kalangan pendidikan, Linux dianggap sebagai obyek untuk dikembangkan, diteliti, dan digunakan untuk pengajaran. Di bidang bisnis, Linux dianggap sebagai suatu peluang untuk menjalankan bisnis sehingga dapat mendatangkan keuntungan. Alasan yang membuat Linux terkenal adalah disebabkan Linux gratis, relatif lebih stabil, tersedia dalam berbagai wajah dengan gaya yang berbeda-beda. Linux memiliki *Graphical User Interface* (GUI) tersendiri dengan berbagai macam pilihan (GDM, KDM, iceWM, dan lain-lain), dan dapat dijadikan *client* dan *server* yang handal. Beberapa wajah-

wajah Linux, di antaranya adaah Slackware, SuSE, RedHat, Mandrake, Turbo Linux, Linux PPC, Debian, dan Trustix.

Sistem operasi LINUX terdiri atas dua bagian kernel dan utility (program aplikasi). Kernel di memori, utility di disk. Kernel adalah suatu *file eksekutable* yang merupakan isi dari sistem operasi LINUX. Kernel berada di dalam memori komputer dari saat komputer baru dihidupkan sampai dengan komputer dimatikan. Kernel merupakan penghubung antara perangkat keras dan pemakai. Kernel inilah yang mengontrol akses ke komputer, mengorganisasi memori, mengatur *file* sistem, dan mengatur pemakaian peralatan antar pemakai. Parameter kernel biasanya tidak perlu diubah-ubah dan untuk mengubahnya diperlukan pengalaman

Bagian dari kernel yang berhubungan dengan alat input output adalah driver, yang harus disesuaikan untuk setiap sistem yang baru. Proses penyesuaian ini disebut *porting*.

Utility berada di disk dan hanya dibawa ke memori pada saat diperlukan. Shell dan semua perintah dalam LINUX dapat disebut utility. Shell adalah program yang bertindak sebagai penerjemah. Mekanisme yang dipakai untuk meminta pelayanan kernel disebut *sistem call*.

Di tingkat *networking*, Linux bisa bekerja sama dengan baik sekali dengan sistem operasi lainnya. Linux mempunyai dukungan TCP/IP yang sangat bagus, dan juga mempunyai dukungan SMB untuk Microsoft *File* Sharing and Printing melalui paket Samba, Apple *File* and Printer Sharing lewat Netatalk, dan IPX/SPX untuk Novell.

Dalam lingkungan campuran Windows/Linux, menggunakan Samba *server* dan sistem Smb*client*, komputer Linux akan tampil di Network Neighborhood dari sistem Windows, hampir tidak bisa dibedakan dengan NT. Komputer Linux juga mempunyai akses penuh ke *file* dan printer yang di-*share* baik dari Windows 95, maupun Windows NT.

Linux tidak menyembunyikan informasi dari *user*. Hal ini berarti informasi penuh dari keadaan sistem dan pesan kesalahan selalu tersedia. Hal ini memungkinkan diagnostik masalah dengan cepat dan bisa diperbaiki dengan cepat pula.

Linux menyediakan alat-alat untuk menampilkan *user* dari memori dan CPU untuk masing-masing program, untuk menentukan program mana yang menggunakan suatu *file* pada suatu saat, untuk melacak program pada saat berjalan, dan meneruskan pesan-pesan kesalahan dari keseluruhan komputer di jaringan ke satu komputer untuk memudahkan *monitoring*.

Kekurangan Linux dalam administrasi jaringan, seperti mengelola *user*, grup, berbagi *folder*, dan printer adalah tidak *user friendly*, memerlukan keahlian dalam menguasai *sintaks* bahasa tertentu sehingga diperlukan administrator yang sudah terbiasa bekerja di lingkungan Linux.

Linux pun juga memiliki berbagai macam jenis distribusi yang memiliki kelebihan masing-masing. Yang digunakan dalam pengerjaan TA ini adalah Linux debian. Berikut akan dijelaskan lebih lanjut mengenai Linux Debian.

2.1.1.1 Linux Debian

Salah satu distribusi Linux terstabil adalah Debian. Debian adalah salah satu distro Linux yang didistribusikan secara gratis oleh para pengembangnya. Debian dikembangkan untuk memenuhi seluruh kebutuhan *user*, mulai dari game, pengolah dokumen, menjalankan bisnis, membuat program, dan lain-lain.

2.1.1.2 Fitur-Fitur Linux Debian

Berikut fitur-fitur yang dimiliki oleh Debian.

1. Sudah banyak digunakan oleh organisasi-organisasi dan individual.
2. Memiliki sistem paket yang terbaik, yakni menggunakan dpkg, Debian Endured Packaging Sistem.
3. Instalasi yang mudah. Instalasi debian cukup mudah karena instalasi dapat dilakukan melalui berbagai jenis media. Mulai dari CD, disket, dan jaringan.
4. Jumlah *software* yang sangat banyak. Debian memiliki total 15490 *software* yang dapat di-*install*
5. Paket-paket yang telah terintegrasi dengan baik. Dari sekian banyak paket tersebut, paket-paket tersebut telah terintegrasi dengan baik. Karena dikembangkan oleh banyak pihak yang telah memeriksa dan memperbaikinya.
6. *Cource code* tersedia. Ini merupakan hal yang penting bagi para pengembang *software*. Karena *tools* dan bahasa pemrograman

disediakan dan *cource code* tersedia. Sehingga dapat melakukan pengembangan lebih jauh.

7. Mudah untuk di-*upgrade.* Melihat dari sistem paket yang sangat baik. Untuk meng-*upgrade* Debian sangatlah mudah. Cukup jalankan "apt-get update" dan "apt-get upgrade" untuk update *software* dan "apt-get dist-update" dan "apt-get dist-upgrade" untuk *update* kernel.
8. Memilki sistem pendeteksi dan pencarian *bug*, Semua *bug* dapat dilaporkan ke internet dan diperlihatkan semuanya kepada *user* dan bagaimana pengangannya.

2.1.2 Windows *Server* 2003

Windows *Server* 2003 merupakan pengembangan dari sistem operasi Windows *Server* 2000. Windows *Server* 2000 merupakan sebuah sistem operasi yang menggabungkan fitur keamanan dari Windows NT dengan kompatibilitas yang luas dari Windows versi sebelumnya, yaitu Windows 95.

Kelebihan Windows *Server* 2003, lebih cepat, lebih stabil, tingkat keamanan tinggi, mendukung driver dan memiliki kemampuan baru, seperti tanpa kabel (wireless), dan lain-lain. *Hosting* Windows *Server* 2003 murah, cepat, dan mudah, serta mendukung ASPupload, CGI, ASP, PHP, CDONTS dengan harga terjangkau.

Keluarga Windows *Server* 2003 terdiri dari berbagai perkakas manajemen automatis, termasuk Microsoft *Software* Update Service (SUS) dan *wizard* konfigurasi *server*. Pengelolaan *Group Policy* lebih mudah dengan Group

Policy Management Console (GPMC), yang memudahkan banyak organisasi untuk menggunakan lebih baik layanan Active Directory.

Fitur administrasi baru pada keluarga Windows *Server* 2003 di antaranya adalah penamaan *domain*, mengelola *cross-domain* dan *cross-forest,* serta pilihan Resultant Set of Policy (RSoP).

Windows *Server* 2003 juga memudahkan *backup* dengan adanya The Volume Copy Shadow Service. Berbagi *file* juga mudah dengan adanya The Web-based Distributed Authoring and Versioning (WebDAV).

Kekurangannya, memerlukan hardware dengan RAM yang cukup tinggi, agar dapat berjalan dengan baik.

Windows *Server* 2003 terdiri dari berbagai versi, yaitu Windows *Server* 2003, Web Edition, Windows Small Business *Server* (SBS), Windows *Server* 2003, Standard Edition., Windows *Server* 2003, Enterprise Edition, dan Windows *Server* 2003, Datacenter Edition.

2.1.2.1 Windows *Server* 2003, Small Business *Server* (SBS)

Salah satu produk Windows *Server* 2003 yang belum *familiar* di kalangan *user* komputer adalah Windows Small Business *Server* (SBS). Microsoft Windows SBS merupakan salah satu produk berbasiskan *server* yang didesain untuk membantu perusahaan menengah ke bawah dalam meningkatkan efisiensi dan produktivitas mereka.

Microsoft Windows SBS di bagi menjadi 2 jenis yaitu Windows SBS 2003 Standard Edition dan Windows SBS 2003 Premium Edition.

Secara umum beberapa manfaat yang dapat di peroleh oleh *user* antara lain :

1. **Membantu untuk melindungi informasi bisnis.**

   Dengan menggunakan SBS 2003 sebagai basis jaringan komputer maka secara otomatis akan didapatkan keamanan informasi dalam menjalankan bisnis karena dalam hal ini SBS 2003 termasuk dalam keluarga besar Microsoft Windows *Server* 2003 yang merupakan salah satu produk terbaru untuk sistem operasi *server* yang sangat dapat diandalkan.

2. **Meningkatkan performansi dan efisiensi kerja**

   Dengan SBS 2003 kita dapat menyelesaikan pekerjaan dalam waktu yang lebih efisien karena akan memberikan kemudahan untuk mencari data, pertukaran data informasi dan pengaksesan data jarak jauh.

   Beberapa point yang berkaitan untuk hal ini adalah :

   a. Central, pemusatan tempat penyimpanan data sehingga membuat kontrol terpusat untuk penyimpanan data dan informasi guna memberikan kenyamanan kepada *user*
   b. Windows Share Point Services, *software* yang berfungsi untuk membantu *user* dalam kemudahan pertukaran informasi, dokumen, dan lainnya
   c. Outlook Web Access, memungkinkan *user* untuk mengakses *email* mereka kapan dan di mana pun mereka berada melalui internet

3. Kemudahan pengaksesan Informasi melalui berbagai media.

Beberapa fitur yang berkaitan untuk hal ini adalah:

a. *Remote* Web Workplace, *remote* portal yang memungkinkan *user* yang memiliki hak akses untuk mengakses fungsi *remote access* melalui internet

b. Portable Device Access, panduan akses dimana beberapa peralatan bergerak seperti PDA ataupun SmartPhone akan dapat berkomunikasi dan bertukar data saat mereka tidak di kantor melalui Active Sync

c. Interactive Web Presence, fitur ini didapatkan melalui SBS Premium Edition dimana termasuk di dalamnya SQL *Server* 2000, sehingga memungkinkan untuk membangun website interaktif dengan kolaborasi *database*.

4. Peningkatan penjualan dan pengurangan biaya melalui aplikasi bisnis

Dengan menggunakan SBS Premium Edition sudah dapat dibangun satu *platform* jaringan terpadu dengan produk-produk yang handal baik dari sisi sistem operasi *server*, *database server*, *email server* dan juga *firewall*.

Dengan semua fitur seperti berbagi *file*, mengeprint, meng-*email*, mengirim fax, mengakses internet, dan kemampuan dasar untuk menjalankan aplikasi bisnis maka Windows Small Business *Server* 2003 memberikan solusi lengkap untuk *server* jaringan.

2.1.2.2 Windows *Server* 2003, Web Edition

Windows *Server* 2003, Web Edition didesain dengan tujuan layanan Web dan *hosting,* Fungsinya tunggal, yaitu sebagai penyedia layanan internet (Internet Service Providers)

Kelebihan Windows *Server* 2003, Web Edition antara lain:

- Platform yang efektif untuk ASP.NET berbasiskan intranet dan Internet
- Memasukkan arsitektur baru dari IIS 6.0, ASP.NET, dan the Microsoft .NET Framework.
- Mendukung 2 jalur simetri (two-way symmetric), yaitu multiprocessing (SMP), 2 gigabytes (GB) of RAM, dan 10 in-bound koneksi *Server* Message Block (SMB).
- Web *server* yang ekonomis dan harga bersaing untuk organisasi yang self-hosting untuk membangun halaman web, situs web.aplikasi, dan layanan web dengan sangat cepat.
- *User* dapat menggunakan Windows *Server* 2003, Web Edition, untuk meng-*install* sebagai berikut: *software* Web *server* misal IIS, *software* manajemen web, misal Microsoft Application Center.

Kekurangan dari Windows *Server* 2003, Web Edition, adalah didesain hanya untuk keperluan web. Windows *Server* 2003, Web Edition meskipun termasuk keluarga Windows *Server* 2003 dan anggota dari Microsoft Active Directory tetapi tidak dapat menjadi *domain controller.* Konsekuensinya, organisasi tidak dapat menggunakan Windows *Server*

2003, Web Edition sendirian untuk menerapkan administrasi seperti Group Policy, *Software* Restriction Policies, *Remote Install*ation Services, Microsoft Metadirectory Services, Internet Authentication Service (IAS), dan lain-lain.

2.1.2.3 Windows *Server* 2003, Standard Edition dan_Windows *Server* 2003, Enterprise Edition

Windows *Server* 2003, Standard Edition, dan Windows *Server* 2003, Enterprise Edition menyediakan beberapa fitur ekstra dan kemampuan:

1. Layanan *cluster.* Cluster *server* menyediakan toleransi kesalahan untuk penanganan database, berbagi *file*, berbagi data intranet, penyampaian pesan, dan aplikasi bisnis umum.Pada Windows *Server* 2003, Enterprise Edition, ukuran layanan *cluster* bertambah dari empat *node* ke delapan *node*, sehingga memudahkan untuk menambahkan/mencopot hardware.
2. Terminal Services *Session* Directory adalah fitur yang mengijinkan *user* untuk berhubungan/memutuskan hubungan *session* pada *server*.
3. Windows Sistem Resource Manager (WSRM) memudahkan administrator untuk mengalokasikan CPU dan memori tiap aplikasi. Ini berguna untuk terminalidasi *server*.
4. Dynamic Host Configuration Protocol (DHCP) mengurangi kompleksitas ketika mengkonfigurasi *host* pada TCP/IP.
5. Layanan jaringan dan komunikasi lebih reliabel dan keamanan melindungi jaringan dengan/tanpa kabel.

6. Layanan penyimpanan memudahkan untuk mengelola, lebih reliabel, backup lebih efisien, sehingga mengurangi biaya dan menambah produktivitas.
7. Layanan aplikasi dan web pada Windows *Server* 2003 memudahkan koneksi dan solusi *server* web terintegrasi.

Windows *Server* 2003, Enterprise Edition, berbeda dengan Windows *Server* 2003, Standard Edition, dalam hal dukungannya terhadap performasi *server* dan kemampuan layanan *server* cluster. Windows *Server* 2003, Enterprise Edition, menyediakan dukungan untuk Eight-way symmetric multiprocessing (SMP) untuk menambah performasi *server* dan kapasitas, sehingga prosesor dapat ditambahkan. Windows *Server* 2003, Enterprise Edition, mendukung hingga delapan prosesor pada sistem tunggal.

Dengan Windows *Server* 2003, Enterprise Edition, organisasi dapat memasang aplikasi yang besar. Sebagai contoh, Windows *Server* 2003, Enterprise Edition, mendukung jaringan, penyampaian pesan, inventarisasi dan sistem pelayanan pelanggan, *database*, *E-commerce*, dan *server file* dan printer.

Berapapun besarnya suatu organisasi, Windows *Server* 2003, Enterprise Edition, pilihan baik untuk menjalankan aplikasi yang selalu tersedia dalam setiap waktu.

2.1.2.4 Windows *Server* 2003, Datacenter Edition

Windows *Server* 2003, Datacenter Edition, adalah solusi yang tepat untuk *database*, proses transaksi *real-time,* dan terminalidasi *server*.

Windows *Server* 2003, Datacenter Edition, memiliki fitur-fitur baru yang membuatnya lebih baik daripada Windows *Server* 2000 dalam hal *server cluster*, dan pelayanan Active Directory. Sebagai tambahan, Windows *Server* 2003, Datacenter Edition, memperkenalkan teknologi baru, seperti *common language* dan The Windows Sistem Resource Manager (WSRM).

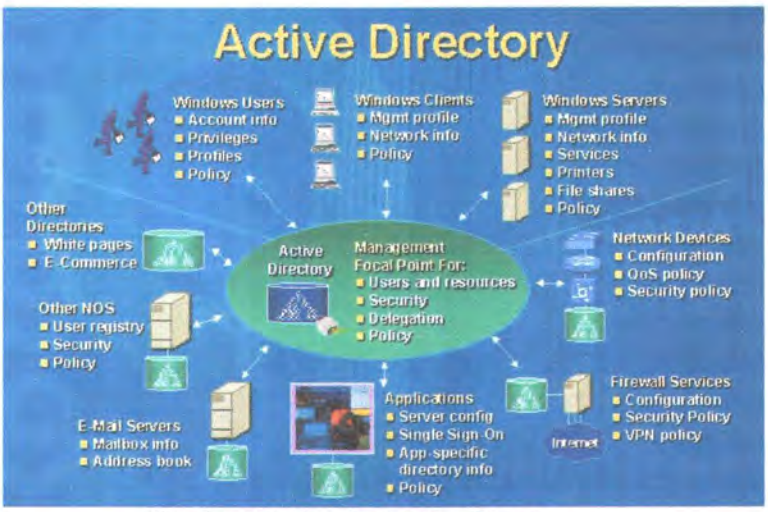
Windows *Server* 2003, Datacenter Edition, tersedia dalam dua versi, 32-bit dan 64-bit. Pada Platform Intel 32-bit, Windows *Server* 2003, Datacenter Edition, mendukung Physical Address Extension (PAE), yang menambah kemampuan memori sistem hingga 64 GB dari RAM fisik.

Non-Uniform Memory Access (NUMA) mendukung. *Sistem firmware* dapat membuat tabel yang disebut Static Resource Affinity Table yang menjelaskan topologi sistem. Topologi NUMA dari sistem Windows *Server* 2003, Datacenter Edition, menggunakan tabel tersebut untuk meningkatkan efisiensi dari sistem operasi untuk proses-proses aplikasi, penjadwalan *thread*, dan manajemen memori.

2.2. Konsep dan Fungsi Active Directory

Active Directory (AD) merupakan fitur terpenting Windows 2000 yang membedakan dari arsitektur Windows NT. Active Directory adalah pengganti

teknologi NT SAM *database* yang berisi konfigurasi keamanan, *user*, grup, dan komputer, dengan banyak kelebihan lain yang dimiliki AD..



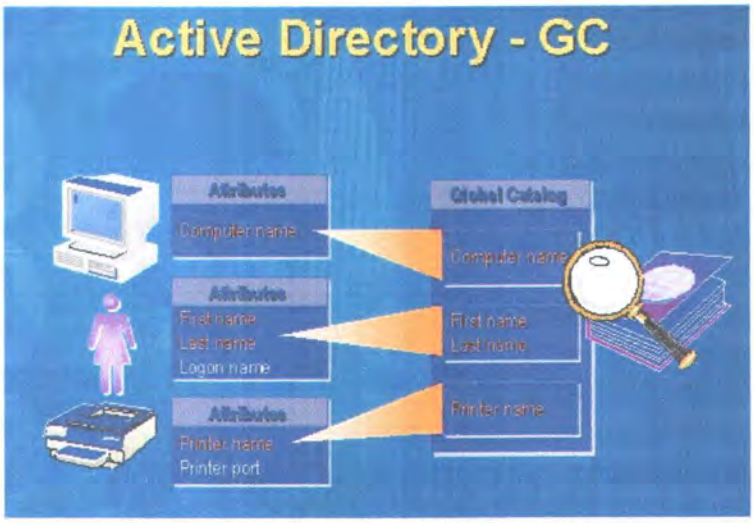
Gambar 2.1. Topologi Active Directory

Active Directory adalah layanan direktori yang menyimpan konfigurasi jaringan, baik *user*, grup, komputer, hardware, serta berbagai *policy* keamanan dalam satu *database* terpusat seperti yang ditunjukkan oleh gambar 2.1. Peranan AD dalam jaringan dapat diibaratkan sebagai buku telepon, yang menyimpan daftar alamat dan informasi penting untuk mengenali berbagai obyek dalam jaringan

Peran utama AD adalah menyediakan sarana untuk melakukan administrasi jaringan secara terpusat, baik di level *domain* maupun lintas *domain*, selama antar *domain* tersebut masih berada dalam satu forest.

Kehadiran AD semakin memudahkan administrator dalam mengelola jaringan sehingga tidak diperlukan kehadiran administrator secara fisik, karena konfigurasi *user*, grup, dan komputer dapat dilakukan secara *remote*. Hal tersebut dimungkinkan dengan adanya Global Catalog (GC) yang menyimpan konfigurasi

komputer dan obyek jaringan, dimana GC tersebut dapat diakses dari manapun di dalam jaringan dengan memanfaatkan infrastruktur AD.



Gambar 2.2. Infrastruktur Active Directory

Gambar 2.2 menunjukkan peranan Active Directory dalam menangani Global Catalog sehingga dapat diakses dari mana saja di dalam jaringan

2.2.1 Kemudahan Dan Fitur Active Directory

Kemudahan dan fitur Active Directory antara lain:

1. Kemudahan Administrasi

AD menyediakan *single sign on* dalam hal administrasi semua sumber daya jaringan. Seorang administrator dapat melakukan login dari komputer manapun di dalam jaringan dan melakukan konfigurasi terhadap obyek dan setiap komputer dalam jaringan.

2. Skalabilitas

AD mampu mengelola sampai dengan jutaan obyek, dibandingkan arsitektur Windows NT yang hanya mampu menangani maksimal 40000 obyek dalam satu *domain*.

3. **Standar Terbuka *(Open Standard)***

AD sesuai dan mendukung berbagai protokol dan teknologi standar yang ada, misal Lightweight Directory Access Protocol (LDAP), sehingga AD dapat berkomunikasi dengan Novell Directory Service dan teknologi lain yang menggunakan LDAP. Support terhadap Hypertext *Transfer* Protocol (HTTP) memungkinkan AD diakses dari web browser dan berbagai bahasa pemrograman pengakses data. Windows 2000 juga mengadopsi Kerberos 5 sebagai protokol autentifikasinya, sehingga sesuai dengan berbagai produk yang menggunakan protokol sejenis. Sistem penamaan *domain* dalam AD menggunakan standar *Domain Name* Sistem (DNS), sehingga memudahkan untuk melakukan koneksi dengan internet.

4. Keamanan

Keamanan penyimpanan informasi. Tiap obyek dalam AD memiliki Access Control List (ACL) dengan daftar *resource* yang dapat mengakses obyek.

5. Replikasi Direktori

Replikasi direktori untuk seluruh *Domain* Controllers (DCs) pada *domain* memudahkan untuk mengakses dan toleransi kesalahan.

6. Kemudahan Komunikasi

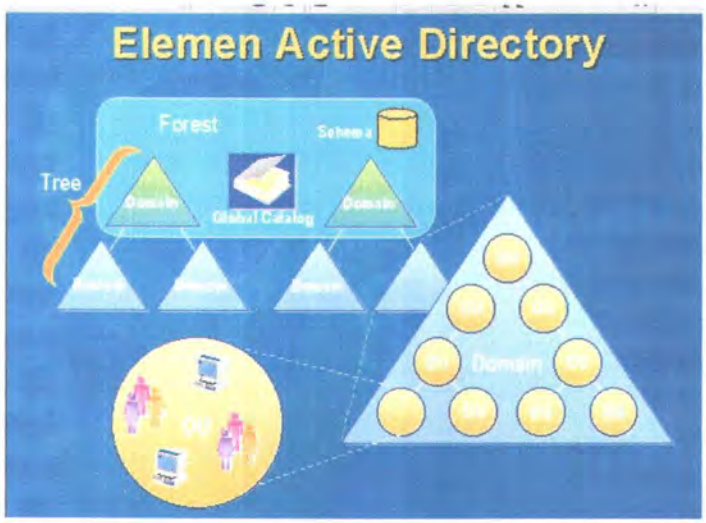
Komunikasi *multiple-protocol.* AD's X. 500 Foundation membuat *user* dapat berkomunikasi melalui berbagai protokol, misal: LDAPv2, LDAPv3, dan HTTP.

7. Fleksibilitas dalam Desain

Desain yang dapat dikembangkan sehingga *user* dapat menambahkan tipe obyek baru atau *build* pada obyek yang telah ada. Sebagai contoh, *user*r dapat menambahkan atribut *salary* pada obyek *user.*

2.2.2 Hierarki Struktur Active Directory

AD terdiri dari berbagai obyek, yang merupakan representasi obyek-obyek yang terdapat di dalam jaringan, baik hardware, *user*, maupun *domain.*



Gambar 2.3. Elemen Active Directory

Hierarki struktur adalah sebagai berikut seperti ditunjukkan pada gambar 2.3.

1. Obyek

Istilah yang digunakan untuk menyebut suatu unit tertentu yang terdapat di dalam jaringan, misalnya *user*, grup, printer, ataupun *folder* yang di-*share*.

2. Container

Container adalah tempat yang di dalamnya berisi berbagai macam obyek.

3. Organizational Unit

OU adalah representasi *container*, yang di dalamnya berisi berbagai macam obyek. OU merupakan kesatuan terkecil dimana pengaturan Group Policy dalam Windows 2000 dapat diterapkan. Biasanya OU tersebut mencerminkan kesatuan organisasi tertentu dalam jaringan, misalnya dapat didefinisikan OU untuk sales, marketing, direksi, dan sebagainya.

Dengan menerapkan *policy* di level OU maka secara otomatis akan diterapkan ke semua *user* dan grup yang terdapat di dalam OU tersebut. Hal ini tentunya sangat memudahkan pekerjaan seorang administrator.

4. Domain

Merupakan kesatuan jaringan terkecil, yang di dalamnya berisi berbagai macam obyek. *Domain* merupakan *security boundary*, sehingga seluruh obyek dalam satu *domain* berada dalam autoritas keamanan yang sama. Sebuah organisasi dapat memiliki lebih dari satu *domain* dalam jaringannya, tergantung pada kebutuhan bisnis maupun kebijakan keamanannya

5. Tree

Merupakan gabungan dari berbagai *domain* yang masing-masing berada dalam satu induk *name space*.

6. Forest

Beberapa tree dapat bergabung menjadi sebuah forest dan masing-masing *domain* tersebut menggunakan *name space* yang berbeda. *Domain* yang berada dalam satu forest menggunakan GC yang sama, sehingga informasi konfigurasi dan obyek jaringan antar *domain* dalam satu forest dapat saling ditukar dan diakses secara terpusat.

Active Directory pada Windows *Server* 2003 mengembangkan kemampuan AD pada Windows 2000. Kemudahan itu antara lain: kemudahan penamaan *domain*, penyimpanan *query* pada *Users* & Computer MMC, sehingga dapat membangun XML dari hasil *query* dan meng-*update* hasilnya, kemudahan pencarian, dan dapat secara langsung membuat *user* dan komputer baru pada lokasi sesuai pilihan.

2.3. MANAJEMEN IDENTITAS

Dengan perkembangan kegiatan yang dimiliki, sebuah organisasi selalu berusaha untuk mengelola keamanan akses dari informasi dan aplikasi dalam keseluruhan elemen sistem informasi yang dimiliki baik di dalam maupun di luar organisasi. Selain itu, organisasi harus dapat menyediakan kemampuan untuk

mengendalikan pertumbuhan *user* sistem, dari dalam maupun luar organisasi tanpa membahayakan informasi yang sensitif dan beresiko.

2.3.1 Komponen-Komponen Penyusun Manajemen Identitas

Manajemen identitas terdiri atas enam komponen penyusun yang menghubungkannya pada sistem yang berlaku pada organisasi. Komponen-komponen tersebut yaitu:

1. **Arsitektur Informasi Organisasi**

   Komponen ini merupakan langkah pertama dan yang utama untuk memahami kebutuhan bisnis dari arsitektur organisasi. Bagi organisasi yang mengembangkan aktifitasnya pada *e-business*, langkah awal harus dilakukan yaitu mengerti kunci proses bisnis, aplikasi yang dibutuhkan, pengelolaan informasi, dan transaksi yang sesuai dengan kebutuhan bisnisnya. Sebagai bagian dari langkah ini, organisasi harus mengetahui hak pemakai tertentu terhadap sumber daya tertentu dengan tingkat keamanan tertentu pula.

2. **Manajemen Hak dan Ijin**

   Strategi utama dari manajemen identitas adalah menetapkan susunan dari ijin dan hak keamanan yang berbasis peran (role). Organisasi harus menentukan:

   a. Siapa yang mengakses sistem dan aplikasi
   b. Apa atau kemana pihak tersebut dapat masuk
   c. Apa yang berhak dilakukan
   d. Bagaimana ketentuan untuk mengijinkan dan menolak akses

e. Siapa yang memiliki identitas dan mengakses informasi

f. Siapa yang harus mengelola informasi ini

g. Bagaimanakah batasan atau aturan untuk informasi pribadi

3. Layanan Direktori

Direktori Organisasi berperan sebagai media terpusat untuk menyimpan dan mengelola identitas *user*, dan hak akses yang diberikan, terhadap aplikasi, informasi, dan sumber daya jaringan. Hal ini dilengkapi dengan metode pencarian umum yang dapat digunakan oleh beberapa aplikasi untuk menerima data.

Pembuatan direktori organisasi memerlukan peran penting dari proses analisis, perencanaan, dan integrasi. Organisasi yang lebih besar dapat memiliki ratusan direktori-direktori kecil dan terpisah, pada fasilitas email, sistem operasi jaringan, dan aplikasi yang menggunakan informasi *user*. Sebuah direktori organisasi besar, mengirim dan menerima informasi dari direktori-direktori organisasi di bawahnya.

4. Otentikasi User

Otentikasi adalah komponen utama dari setiap strategi manajemen identitas. Otentikasi adalah proses untuk memastikan identitas dari *user*, sehingga akses *user* terhadap sumber daya yang dilindungi dapat secara benar untuk diterima atau ditolak. Otentikasi dapat dilakukan secara terpusat.

Otorisasi harus dilakukan pada sumber daya yang akan dilindungi. Setiap sistem harus memutuskan siapa yang dapat mengakses direktori tertentu,

halaman web *folder* email, dan lainnya. Dua hal yang bisa membantu memudahkan administrasi otorisasi dalam hal ini adalah *user* dari grup dan atribut.

Grup adalah kumpulan dari identitas. Dengan membuat sebuah grup, memungkinkan untuk melakukan otorisasi akses untuk lebih dari satu identitas dengan satu aturan. Sebuah arsitektur otentikasi bisa mengidentifikasi sebuah identitas sebagai member dari sebuah grup tapi tidak dapat memutuskan akses apa yang dapat diberikan pada grup tersebut. Informasi anggota grup dan informasi otorisasi dapat disimpan dalam simpanan terpusat (seperti direktori LDAP) dan digunakan oleh aplikasi yang setuju menggunakan grup dan format otorisasi tersebut.

Atribut adalah properti dari identitas (atau grup). Atribut dapat digunakan dalam proses otorisasi dengan membedakan apakah identitas yang akan diotentikasi memiliki atribut yang dibutuhkan sebelum mengakses sebuah *resource*. Otorisasi atribut juga sama untuk grup. Dengan menggunakan satu (kenggotaan grup atau atribut identitas), dapat ditentukan otentikasi untuk beberapa identitas.

Otentikasi bukan merupakan perlindungan terhadap privasi atau integritas data. Standar SSL adalah salah satu protokol yang menerapkan otentikasi, privasi, dan integritas data. Protokol SSL berjalan di atas TCP/IP dan di bawah HTTP, LDAP, dan protokol layer jaringan tingkat tinggi lainnya. SSL membuat sebuah *SSL-enabled server* untuk mengotentikasi dirinya

ke sebuah *SSL-enabled client*, juga *client* mengotentikasi dirinya ke *server*, dan membuat kedua pihak untuk membuat koneksi yang terenkripsi.

Sebuah koneksi SSL yang *server-authenticated*, yaitu yang memungkinkan dua arah otentikasi, memang sangat sulit untuk disadap, mengubah data tanpa terdeteksi, atau mengganti identitas ke *server*. Namun, sekali *client* sudah terotentikasi ke *server*, setiap *user* dapat membuat koneksi dan mendapatkan akses yang dimiliki oleh *server* tersebut.

Otentikasi *client* adalah elemen utama dalam kemanan jaringan dalam sebagian besar intranet dan ekstranet. Bagian berikut menjelaskan perbedaan dua jenis otentikasi *client*

a. Otentikasi Dasar (*Basic Authentication*)

Semua *server* mengijinkan otentikasi *client* dengan menggunakan *user name* dan *password*. Bentuk ini bisa berjalan dalam koneksi tanpa enkripsi, atau otentikasi *server* dan SSL terenkripsi.

b. Otentikasi Kuat (*Strong authentication*)

Otentikasi *client* berbasis *certificate*, seperti yang diimplementasikan oleh Netscape, menggunakan SSL. Bentuk otentikasi ini mendukung *single sign-on*

5. Penyediaan User (*User Provisioning*)

User provisioning adalah proses untuk menyebarkan atau mengimplementasikan hak akses berdasarkan kebijakan bisnis untuk setiap anggota dalam ataupun luar organisasi. Dalam strategi manajemen

identitas yang efektif, terdapat administrasi terpusat untuk pemberian dan penerapan hak akses ini.

6. ***Workflow***

   Workflow adalah proses yang dibuat automatis, yang mendukung *user provisioning*. Seperti yang dilakukan secara manual oleh administrator, *workflow provisioning* memicu penambahan dan perubahan pada aplikasi dan informasi lain.

2.3.2 Kemampuan-Kemampuan Umum Sistem Manajemen Identitas

Berikut ini adalah kemampuan-kemampuan yang umum terdapat dalam sebuah sistem manajemen identitas.

1. **Reset *Password***

   Kemampuan dari elemen ini yaitu meng-*enable user* untuk melakukan reset terhadap *password* dan *unlock account* yang dimiliki tanpa melibatkan banyak peran dari help desk. Pada aplikasi umum, *user* mengakses aplikasi reset *password* melalui sebuah *browser*, Windows *Client*, atau telepon. *User* akan diotentikasi dengan beberapa pertanyaan untuk membuktikan jawaban yang benar dan pernah diajukan oleh *user* pada saat pembuatan *account*.

2. **Sinkronisasi *Password***

   User hanya membutuhkan satu *password* untuk melewati beberapa sistem yang berbeda, sehingga dapat mengurangi kemungkinan terjadinya lupa *password*. Hal ini berbeda dengan solusi *single sign on*, *user* tetap harus memasukkan nama atau ID dan *password* untuk masing-masing sistem

atau aplikasi. Solusi ini pada umumnya berupa aplikasi atau *software*. *Software* ini biasanya berada dalam *server*, dan koneksi API dari *software* menuju basis data, sistem help desk, dan sistem keamanan.

2.4. *SERVER* MESSAGE BLOCK (SMB)

Server Message Block adalah nama dari I/O Redirection dari NetBIOS Extended *User Interface* (NETBEUI). Protokol ini juga kerap disebut Common Internet *File* Sistem (CIFS), LanManager atau protokol NetBIOS.

SMB/CIFS merupakan protokol yang digunakan mesin Windows 95/98 dan NT untuk berkomunikasi dengan *server* Samba dan begitu sebaliknya. SMB protokol digunakan untuk operasi *file-file* printer, seperti: membuka dan menutup *file*, membuat dan menghapus direktori, membaca dan menghapus *file*, mencari *file*, dan proses antrian *file* dalam *print spool*. Tiap antrian tersebut dapat di-*encode* ke dalam pesan SMB dan ditransmisikan ke dan dari *server*.

2.4.1 Format SMB

Richard Shape, anggota tim Samba mendefinisikan SMB sebagai protokol *request-response* [4]. *Client* mengirim SMB request ke *server*, dan *server* mengirim SMB response ke *client*. Pada dasarnya, *form*at SMB terdiri atas *string header* dan *command*.

2.4.1.1 Format Header SMB

Tabel 2.1 menunjukkan *form*at header SMB. Jika *client* pertama kali mencoba koneksi ke *server*, *client* belum diberi nilai TID (Tree Identifier).

Jika hubungan telah berhasil dilakukan, null TID (xFFFF) ditempatkan di header field. Field header SMB terlihat pada tabel 2.1

Tabel 2.1. Format Header SMB

Field	Size (bytes)	Description
0xFF'SMB'	1	Protocol identifier
COM	1	Kode perintah, dari 0x00 to 0xFF
RCLS	1	Error class
REH	1	Reserved
ERR	2	Kode kesalahan
REB	1	Reserved
RES	14	Reserved
TID	2	Tree identifier; ID unik untuk *resource* yang digunakan oleh *client*
PID	2	Caller process ID
UID	2	*User* identifier
MID	2	Multiplex identifier; digunakan untuk rute *request* dalam sebuah proses

2.4.1.2 Format Perintah SMB

Header berisi byte-byte yang merupakan perintah SMB atau balasan masing-masing perintah, seperti untuk Open *File* (SMBopen) atau Get Print Queue (SMBsplretq), mempunyai parameter dan data sendiri. Seperti SMB header fields, tidak semua perintah perlu digunakan, bergantung pada perintah yang diperlukan. Contohnya, perintah Get *Server* Attributes (SMBdskattr) akan menetapkan WTC dan BCC menjadi nol. Segmen perintah tersebut dapat dilihat pada tabel 2.2.

Tabel 2.2.Segmen Perintah SMB

Field	Size (bytes)	Description
WCT	1	Word Count
VWV	Variable	Parameter words (satuan WCT)
BCC	2	Parameter byte count
DATA	Variable	Data (satuan BCC)

2.4.2 Variasi SMB

Protokol SMB selalu bertambah dengan perintah-perintah baru. Setiap versi baru yang muncul selalu kompatibel dengan versi sebelumnya. Ini memungkinkan suatu LAN untuk memiliki berbagai *client* dan *server* yang menjalankan berbagai versi SMB pada suatu waktu. Tabel 2.3 memberikan gambaran beberapa versi protokol SMB. ID string digunakan sebagai pendefinisian level protokol yang digunakan untuk berkomunikasi oleh *client server*.

2.4.3 SMB Client-*Server*

SMB adalah protokol *client/server*. Secara sederhana, hal ini berarti *client* mengirim permintaan ke *server*, kemudian *server* merespon dengan memberikan balasan ke *client*. Adapun komputer yang berfungsi sebagai *server* dapat saja sewaktu-waktu membuat dirinya sebagai *client*.

Pada contoh berikut, ada dua komputer Windows 95/98 bernama A dan B Komputer A berbagi printer dengan jaringan, sedangkan B berbagi disk direktori. Dengan demikian, dikatakan bahwa A merupakan *client* pada saat mengakses disk pada B dan sebagai *server* pada saat B menggunakan printer.

Tidak ada implikasi jumlah r*esource* yang ada untuk membuat suatu *server* atau besarnya kapasitas disk maupun kecepatan processor. Sebuah *server* saja dapat berjalan di atas mesin 486 yang tersambung ke sebuah printer, misalnya.

Tabel 2.3. Contoh Protokol SMB

Protocol *Name*	ID String	Used By
Core	PC NETWORK PROGRAM 1.0	
Core Plus	MICROSOFT NETWORKS 1.03	

LAN Manager 1.0	LANMAN1.0	
LAN Manager 2.0	LM1.2X002	
LAN Manager 2.1	LANMAN2.1	
NT LAN Manager 1.0	NT LM 0.12	Windows NT 4.0
Samba's NT LM 0.12	Samba	Samba
Common Internet *File* Sistem	CIFS 1.0	Windows 2000

2.5. SAMBA

2.5.1 Pendahuluan

Samba merupakan implementasi dari protokol SMB (*Server* Message Block) pada sistem UNIX. Protokol ini digunakan oleh MSWindows NT untuk *File* and Printing Sharing Service. Dengan mengaktifkan Samba pada mesin Linux maka *user* dapat berbagi *file* dan printer dengan Windows 95/98/2000 atau Windows NT. Dengan kata lain, dengan menjalankan Samba, maka suatu *server* Linux dapat tampak seperti suatu Windows NT *Server* bagi mesin Windows lainnya.

Pada Linux *user* dapat me-*mounting* direktori yang di-*share* pada Windows juga dapat mengakses secara langsung pada direktori tersebut. Sedangkan pada Windows, *user* dapat melihat direktori yang di-*share* berupa ikon yang terdapat dalam Network Neighborhood.

Samba dibuat untuk menjalankan dua proses tersebut dengan transport protokol TCP/IP. Mekanisme protokol tersebut didokumentasikan dalam dua RFC, yang merupakan dokumen standar internet, yakni RFC 1001 dan 1002. Protokol NetBIOS digunakan untuk menyampaikan pesan, berjalan di atas protokol transport TCP/IP, sehingga disebut NBT (Net BIOS over TCP/IP)

2.5.2 Kemampuan Samba

Samba dapat bekerja dengan baik pada lingkungan jaringan berbasis Windows. Hampir semua fasilitas dari Microsoft dapat dimanfaatkan secara optimal oleh Samba. Samba mendukung WINS *Server*, dapat berfungsi pula sebagai *master browser* maupun *domain master browser*. Samba mendukung otentikasi *password* teks biasa maupun *password* terenkripsi. Kemampuan Samba secara umum dapat dilihat pada tabel 2.4

Tabel.2.4. Tabel Kemampuan Samba

Kemampuan sebagai	Dukungan
File Server	Yes
Printer *Server*	Yes
Primary *Domain* Controller	Yes
Backup *Domain* Controller	No
Windows 95/98 Authentication	Yes
Local Master Browser	Yes
Local Backup Browser	No
Domain Master Browser	Yes
Primary WINS *Server*	Yes
Secondary WINS *Server*	No

2.5.3 Daemon-daemon Samba

Samba di Linux terdiri atas dua daemon:

1. **Smbd**

   Smbd, daemon (atau program TSR = Terminate and Stay Resident) yang berjalan sebagai *background process*, yang bertanggung jawab untuk menata sumber daya yang dapat digunakan bersama antara *server* Samba dan *client*nya, memberi layanan pemakaian *file* dan printer bersama pada jaringan yang menggunakan protokol SMB, serta menyediakan otentikasi dan otorisasi untuk *client* SMB, yaitu Windows

95/98, Windows NT, Windows for Workgroups atau LAN Manager. Daemon ini bertanggung jawab terhadap seluruh aktifitas antara Samba *server* dan *client* dalam jaringan. Konfigurasi dari smbd ada dalam smbd.conf.

2. **Nmbd**

   Nmbd, merupakan *name server* yang meniru fungsi WINS dan NetBIOS. Daemon yang memanfaatkan Windows Internet *Name* Service (WINS) dan membantu *client* untuk browsing di Network Neighborhood. Daemon ini melayani permintaan *name server* dan memberikan respon yang sesuai, dan menyediakan informasi serta daftar *browse* untuk Network Neighborhood. Konfigurasi dari nmbd juga ada dalam smb.conf.

2.5.4 Komponen-komponen Pendukung Samba

1. **Smbclient**

   Smb*client* – merupakan aplikasi FTP seperti yang disediakan oleh Linux yang dapat digunakan untuk berhubungan dengan Samba Share. Berfungsi sebagai *client* yang memungkinkan *user* untuk mengakses *share* SMB di komputer lain, seperti mengakses *resource* yang ada di komputer lain lewat Map Network Drive di jaringan Windows NT.

2. **Smbtar**

   Smbtar – sebuah program untuk menyimpan data yang di-*share* ke media penyimpan seperti tape, disk, dan lainnya.

3. **Nmblookup**

   Nmblookup – program yang menyediakan NetBIOS melalui TCP/IP *name* lookup. Utilitas untuk melakukan *name query* (meminta nama NetBIOS dari komputer-komputer yang sedang *on-line*) dari mesin Linux.

4. **Smbpasswd**

   Smbpasswd – utilitas untuk mengubah *password* terenkripsi SMB, baik di *server* Samba maupun Windows NT. Sebuah program yang memperbolehkan admin untuk mengganti *password* yang digunakan Samba.

5. **Smbstatus**

   Smbstatus – program yang memperlihatkan *client* pada jaringan yang sedang terhubung ke Samba *server*. Utilitas untuk mengecek koneksi yang sedang berlangsung ke *server*.

6. **Testparm**

   Testparm – program sederhana yang mengesahkan *file* konfigurasi Samba, untuk mengecek konfigurasi smb.conf

7. **Testprns**

   Testprns – program yang bertugas untuk menguji berbagai jenis printer untuk mengetahui apakah printer tersebut telah dikenal oleh daemon pada Samba.

8. SWAT

SWAT – Samba Web Administration Tool, program bantu yang memberikan antarmuka model web untuk mengadministrasi Samba, mempermudah mengedit *file* smb.conf, mengatur *resource* yang dibagi pakai, dan melihat status Samba terakhir.

2.5.5 *File-file* Instalasi Samba

File-file yang ter-*install* yang sering digunakan untuk mengkonfigurasi dan menjalankan Samba antara lain :

1. /usr/bin/smbd

   `/usr/bin/smbd` merupakan daemon yang menyediakan *File* and Printing Sharing Service di sistem UNIX untuk SMB *Client* seperti Windows 95/98 atau Windows NT. Perintah untuk menjalankan daemon ini adalah `/usr/bin/smbd -D`

2. /usr/bin/nmbd

   `/usr/bin/nmbd` merupakan daemon yang menyediakan penamaan NetBIOS dan kemampuan browsing bagi SMB *Client*. Perintah untuk menjalankan daemon ini adalah `/usr/bin/nmbd -D`

3. /usr/bin/smbclient

   `/usr/bin/smbclient` untuk mengakses direktori yang di-*share* di Windows dengan model FTP. Perintah untuk menjalankannya adalah `/usr/bin/smbclient`. Contoh *user* darinya dapat dilihat pada gambar 2.4.

4. /usr/bin/smbmount

`/usr/bin/smbmount` untuk me-*mount* direktori yang di-*share* di Windows sehingga dapat dibaca layaknya CDROM yang di- *mount* pada /mnt/cdrom. Perintah untuk menggunakannya adalah `/usr/bin/smbmount`. Contoh terdapat pada gambar 2.5.

```
[root@namec samba-2.0.6]# /usr/bin/smbclient
//Planet-3/pic added interface
ip=192.168.0.1 bcast=192.168.0.255 nmask=255.255.255.0
Got a positive name query response from 192.168.0.13
( 192.168.0.13 )
Password:
smb: \>
```

Gambar 2.4. Contoh pemakaian /usr/bin/smbclient

```
[root@namec samba-2.0.6]# /usr/bin/smbmount
//Planet-3/oky  /mnt/share
Password: [root@namec samba-2.0.6]# cd /mnt/share
[root@namec share]# ls
```

Gambar 2.5. Contoh pemakaian /usr/bin/smbmount

5. /usr/bin/smbumount

`/usr/bin/smbumount` untuk *unmounting* setelah selesai bekerja dengan direktori yang di-*mount*. Perintah untuk menggunakannya adalah `/usr/bin/smbumount`. Contoh *user* dari perintah ini terdapat pada gambar 2.6.

```
[root@namec /]# /usr/bin/smbumount /mnt/share
[root@namec /]# cd /mnt/share
      [root@namec share]# ls
```

Gambar 2.6. Contoh pemakaian /usr/bin/smbumount

6. /usr/bin/smbstatus

`/usr/bin/smbstatus` melaporkan status koneksi Samba. Contoh *user* dari perintah ini terdapat pada gambar 2.7.

```
[root@namec /]# /usr/bin/smbstatus
Samba version 2.0.6
Service uid gid pid machine
Anton Anton 1004 15514 planet-3 (192.168.0.13) Fri Sep 6
10:150
No locked files
Share mode memory usage (bytes):
   1048464(99%) free + 56(0%) used + 56(0%) overhead =
                  1048576(100%) total
```

Gambar 2.7 Contoh pemakaian /usr/bin/smbstatus

7. /usr/bin/mksmbpasswd

`/usr/bin/mksmbpasswd.sh` adalah skrip shell untuk menambahkan *user* pada /etc/passwd milik sistem Linux ke /etc/smbpasswd milik Samba. Cara menggunakannya adalah `cat /etc/passwd | mksmbpasswd.sh > /etc/smbpasswd`

8. /usr/bin/smbadduser

`/usr/bin/smbadduser` menambahkan *user* ke *file user* samba (/etc/smb*user*s) dan *file password* Samba (/etc/smbpasswd).

9. /usr/bin/smbpasswd

`/usr/bin/smbpasswd` untuk mengubah *password user*. Contoh *user* dari perintah ini terdapat pada gambar 2.8.

```
[root@namec /]# /usr/bin/smbadduser pelatihan:training
Adding: pelatihan to /etc/smbpasswd
Adding: {pelatihan = training} to /etc/smbusers
ENTER password for pelatihan
New SMB password:
Retype new SMB password:
Password changed for user pelatihan.
```

Gambar 2.8. Contoh pemakaian /usr/bin/smbadduser

10. /usr/doc/samba-2.0.6/

`/usr/doc/samba-2.0.6/` berisi seluruh dokumentasi Samba dan contoh-contoh konfigurasi Samba.

11. /etc/smb.conf

/etc/smb.conf merupakan *file* konfigurasi Samba.

12. /etc/smbpasswd

/etc/smbpasswd merupakan *password file* yang akan digunakan Samba untuk proses otentikasi.

13. /etc/smbusers

/etc/smbusers berisi pemetaan *user* Linux dengan *user* Windows yang akan digunakan Samba untuk proses otentikasi.

2.5.6 Konfigurasi Samba

Saat *daemon-daemon* Samba dihidupkan, *daemon-daemon* tersebut akan membaca *file* /etc/smb.conf untuk mendapatkan berbagai informasi yang diperlukan untuk menghubungkan jaringan Windows dengan LINUX. Informasi tersebut antara lain, nama *workgroup*, *password file*, direktori yang di-*share*, hak akses. Gambar 2.10 contoh konfigurasi Samba standar pada /etc/smb.conf :

```
[global]
# workgroup = NT-Domain-Name atau Workgroup-Name
workgroup = PLANET
# server string = NT Description atau deskripsi server
Samba
server string = Samba Server
# hanya mengizinkan network 192.168.0 dan network 127
untuk # # #mengakses server Samba
hosts allow = 192.168.0. 127.
# Samba menggunakan file log berbeda untuk tiap mesin
yang connect
log file = /var/log/samba/log.%m
# security level, user level atau share level
# User level mengakibatkan proses otentikasi dilakukan 1
kali
# direktori yang di share diakses berdasarkan priviledge
user.
# Share level mengakibatkan proses otentikasi berulang-
ulang
# direktori yang di-share menentukan sendiri permission-
nya
```

```
security = user
# enkripsikan password bila terkoneksi dengan WIN9x/NT
encrypt passwords = yes
# file password yang digunakan untuk proses otentikasi
smb passwd file = /etc/smbpasswd
# sinkronisasikan perubahan UNIX password dengan SAMBA
password
unix password sync = Yes
# bagian ini dibiarkan default
socket    options    =    TCP_NODELAY    SO_RCVBUF=8192
SO_SNDBUF=8192
# ==== Share Definitions ====
[homes]
comment = Home Directories
browseable = no
writable = yes
[doc]
comment = Linux Documentation
path = /usr/doc
public = yes
Writable = yes
Printable = no
[source]
comment = Linux Source
path = /home/ftp/pub
public = yes
[upload]
comment = Upload file
path = /home/ftp/upload
public = no
writable = yes
browseable = yes
          readonly = no
```

gambar 2.9Contoh file smb.conf

Keterangan singkat :

1. ***Comment***

   Comment merupakan deskripsi lebih lengkap dari sebuah *share*

2. ***Path***

   Path menentukan direktori lokal yang di-*share*

3. ***Public***

   Public bila 'yes' berlaku seperti *anonymous* pada FTP

2.6. NSSWITCH

2.6.1 Penjelasan

Berbagai fungsi dalam *library* C butuh untuk dikonfigurasi untuk bekerja dengan benar di lingkungan lokal. Secara tradisional, ini biasa dilakukan dengan menggunakan *file* (contoh: /etc/passwd) tapi *name* service lain (seperti Network Information Service (NIS) dan *Domain Name* Service (DNS)) menjadi populer dan digunakan dalam *library* C yang biasanya saling membutuhkan.

Library GNU C memiliki solusi yang lebih jelas untuk permasalahan ini. Hal ini didesain mengikuti metode yang digunakan Sun Mycrosistems di *library* C dari solaris 2. *Library* GNU C kemudian mengikutinya dan menamakan skema ini *Name* Service Switch (NSS).

Ide dasarnya adalah untuk meletakkan implemetasi dari service yang berbeda yang ditawarkan untuk mengakses database dalam modul yang terpisah. Hal ini memiliki kelebihan:

- Pengembang dapat menambah service baru tanpa perlu menambahkan ke dalam *library* GNU C.
- Masing-masing modul dapat di-*update* secara terpisah
- *Library* C menjadi semakin kecil

Untuk memenuhi tujuan yang pertama tadi, berikut akan dijelaskan modulnya. Untuk mendapatkan implementasi dari *service* baru penting untuk diingat bagaimana fungsi dalam modul tersebut dipanggil. Yang tidak

didesain untuk langsung digunakan dalam pemrograman. Bahkan programmer hanya perlu menggunakan fungsi yang sudah terdokumentasi dan standar untuk mengakses database.

Database yang tersedia di NSS antara lain:

- Alias → alias email
- Ethers → nomor Ethernet
- Group → grup dari *user*
- Hosts → nama *host* dan jumlah
- Netgroup → daftar jaringan *user* dan *host* yang luas
- Networks → nama jaringan dan jumlah
- Protocols → protokol jaringan
- Passwd → *password user*
- Rpc → nama dan jumlah dari *Remote* Procedure Call
- Service → service jaringan
- Shadow → *password* pengguna shadow

2.6.2 *File* Konfigurasi

Berikut adalah contoh *file* konfigurasi NSS

```
# /etc/nsswitch.conf
#
# Name Service Switch configuration file.
#

passwd:     db files nis
shadow:     files
group:      db files nis

hosts:      files nisplus nis dns
networks:   nisplus [NOTFOUND=return] files

ethers:     nisplus [NOTFOUND=return] db files
protocols:  nisplus [NOTFOUND=return] db files
rpc:        nisplus [NOTFOUND=return] db files
services:   nisplus [NOTFOUND=return] db files
```

gambar 2.10. Contoh file /etc/nsswitch.conf

Kolom pertama adalah *database*, dimana sesuai seperti yang disebutkan dalam *database* yang tersedia. Kemudian di sebelahnya adalah menspesifikasikan bagaimana proses *look up* terjadi. Dan harus dispesifikasikan terpisah.

Spesifikasi konfigurasi dari masing-masing database dapat mengandung 2 hal berbeda:

- Spesifikasi *service*
- Reaksi dari hasil *look-up*

2.6.2.1 Services dalam Konfigurasi NSS

Pada *file* contoh diatas disebutkan 4 service berbeda: *file*s, db, nis, nisplus. Ini tidak berarti bahwa *service* ini tersedia untuk semua tempat dan tidak berarti semua *service* ini selalu tersedia.

Pada kenyataannya, nama-nama ini adalah *string* sederhana yang digunakan kode NSS untuk mencari alamat fungsi secara implicit.

Diasumsikan nama *service* akan digunakan untuk *look-up*. Kode dari service ini diimplementasikan di dalam modul bernama libnss_*name*. Dalam sebuah sistem yang mendukung *shared libraries* kenyataannya dengan nama (contoh) libnss_*name*.so.2. Nomor yang paling akhir menunjukkan versi yang digunakan. Umumnya *user* tidak perlu menspesifikasikan nama directory, cukup dengan nama *service*-nya saja.

2.6.2.2 Action dalam Konfigurasi NSS

Item kedua dalam spesifikasi memberikan *user* kontrol yang lebih baik dalam proses *look-up*. *Item action* berada diantara 2 nama *service* dan ditulis diantara tanda kurang. Bentuk umumnya antara lain:

```
[ ( !? status = action )+ ]
```

Gambar 2.11. Action Konfigurasi NSS

Dimana:

```
status => success | notfound | unavail | tryagain
action => return | continue
```

Gambar 2.12. Hasil Konfigurasi NSS

Dalam kasus dimana *keyword* itu tidak signifikan. Nilai status adalah hasil dari pemanggilan fungsi dari *service* tertentu. Yang memiliki arti:

Success → Tidak ada kesalahan yang muncul dan data yang dicari ada. *Action* standar untuk ini adalah *return*.

Notfound → Proses *look-up* bekerja baik, tapi nilai yang dibutuhkan tidak ketemu. *Action* standarnya adalah *continue*.

Unavail → *Service* ini tidak ada. Ini dapat berarti *file* yang dibutuhkan tidak tersedia atau untuk DNS, *server* tidak tersedia atau tidak dapat *query*. *Action* standarnya adalah *continue*.

Tryagain → *Service* ini untuk sementara tidak ada. Ini dapat berarti *file* dikunci atau *server* saat ini tidak dapat menerima koneksi lebih. *Action* standarnya adalah *continue*.

2.7. WINBINDD

2.7.1 Penjelasan

Winbindd adalah daemon yang menyediakan *service* untuk *Name Service Switch* (NSS) yang ada di mayoritas *library* C modern. NSS dapat membuat data *user* dan informasi sistem diambil dari *service database* lain seperti NIS dan DNS. Dan cara kerjanya dapat diatur melalui *file* konfigurasi /etc/nsswitch.conf. *User* dan grup dapat disesuaikan Id-nya ditentukan oleh administrator Samba.

Service yang disediakan oleh winbindd dinamakan winbind dan dapat digunakan untuk mengambil informasi data *user* dan grupnya dari sebuah *server* Windows NT. *Service* ini juga menyediakan service otentikasi melalui modul PAM yang bersesuaian.

2.7.2 Nama dan ID

User dan grup di *server* Windows NT diberikan *relative id* (rid) yang unik untuk *domain* tersebut saat *user* atau group itu dibuat. Untuk mengubah *user* atau grup di Windows NT menjadi *user* dan grup unix, diperlukan pemetaan

dari id *user* dan grup id diperlukan. Ini adalah tugas yang harus dijalankan winbind.

Saat *user* dan grup winbind diambil dari *server*, *id user* dan grup dialokasikan dalam sebuah nilai. Ini dilakukan terhadap saat yang datang pertama, pertama dilayani, walaupun semua *user* dan grup yang telah ada akan segera dipetakan saat *client* melakukan perintah enumerasi *user* dan grup. Ids unix yang telah dialokasikan disimpan dalam *database* dibawah Samba.

2.7.3 Konfigurasi

Konfigurasi *daemon* winbindd dapat dilakukan melalui parameter konfigurasi di smb.conf. Semua *parameter* harus diletakkan spesifik didalam bagian global.

1. *winbind separator*
2. *idmap uid*
3. *idmap gid*
4. *winbind cache time*
5. *winbind enum users*
6. *winbind enum groups*
7. *template homedir*
8. *template shell*
9. *winbind use default domain*

2.8. PAM

2.8.1 Pendahuluan

PAM (*Plugable Authentication Module*) adalah koleksi modul yang berfungsi sebagai pembatas antara *service* yang ada dengan *user*. Modul ini memiliki *user* dari yang luas, mulai dari melarang *user* dari grup UNIX biasa (atau netgroup, subnet, dan lain-lain) untuk login, hinga melakukan pembatasan *resources* sehingga *user* tidak dapat menggunakan *resources* berlebihan.

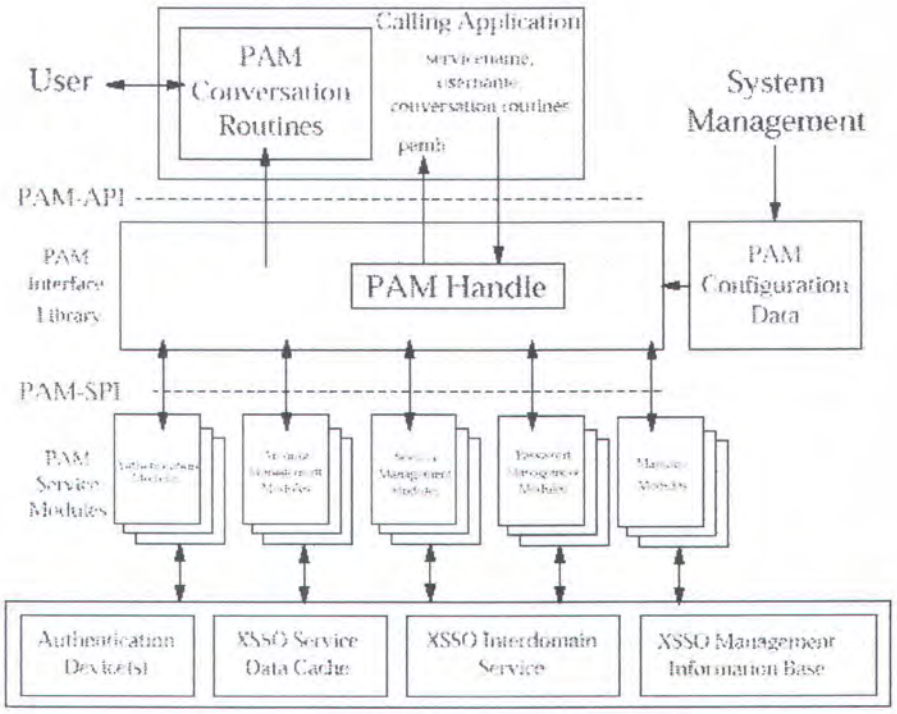
PAM digunakan oleh banyak aplikasi UNIX seperti AIX, HP-UX, dan Solaris, demikian juga oleh versi Unix yang gratis seperti FreeBSD. Hampir semua distro Linux juga menggunakan PAM, terkecuali Slackware.

PAM menyediakan sistem administrator fleksibilitas untuk

- Memilih *service* otentikasi manapun yang tersedia dalam sistem untuk melakukan otentikasi akhir untuk menjalankan aplikasi.
- Menggunakan banyak *service* otentikasi dan menyediakan teknologi otentikasi yang terintegrasi dengan service sistem entry.
- Menambah modul *service* otentikasi ke dalam suatu sistem dan membuat hal itu dapat dpakai tanpa harus merubah aplikasi.
- *Service* Pemetaan incorporate untuk menjelajahi nama *user* dan token otentikasi antara otentikasi *domain* yang berbeda.

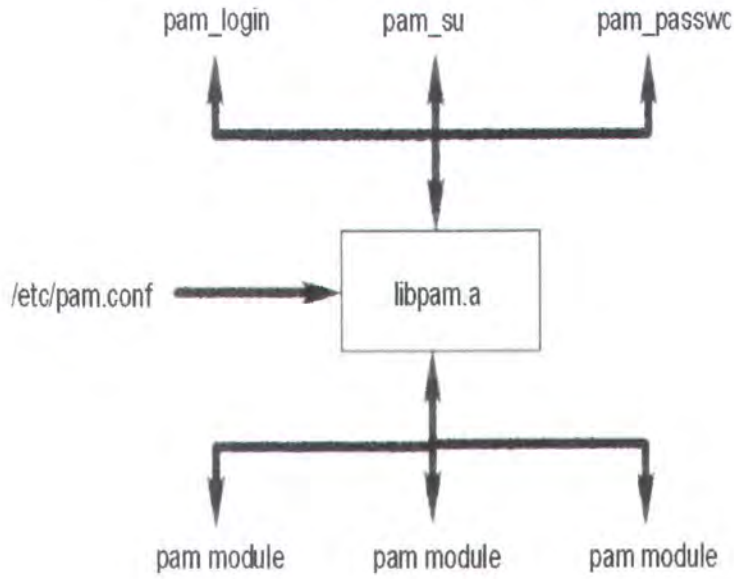
PAM *framework* menggabungkan antara *library interface* dan berbagai service modul PAM. PAM *interface library* adalah bagian implementasi dari Application Praogramming *Interface*(PAM-API). Dan *service* modul PAM

adalah bagian dari objek dinamis yang dibangun oleh PAM-SPI untuk menyediakan beberapa tipe otentikasi.



Gambar 2.13. Modul-modul PAM

Gambar berikut menunjukkan interaksi antara aplikasi, *library* PAM, *file* konfigurasi, dan modul PAM. Aplikasi PAM (pam_login, pam_su, dan pam_passwd) membangkitkan PAM API di *library* PAM. *Library* ini menentukan modul yang cocok berdasarkan aplikasi yang dimasukkan dalam *file* konfigurasi dan memanggil PAM SPI dalam modul. Berhasil atau tidaknya modul dan kebiasaan ditentukan dalam *file* konfigurasi dan dari modul lain yang diperlukan bila ada. Jika ada, proses diteruskan, jika tidak, data akan dikembalikan ke aplikasi.



Gambar 2.14. Diagram modul PAM

2.8.2 Konfigurasi PAM

Disini akan dibahas konfigurasi PAM yang diimplementasikan di Linux. Pada Solaris dan UNIX komersil lainnya memiliki sedikit perbedaan model konfigurasi, tapi tetap berkisar dalam sebuah *file*, `/etc/pam.conf`. Walaupun secara konsep kedua implementasi adalah sama, model pada Linux menggunakan *file* konfigurasi yang berbeda untuk tiap *service* yang menggunakan PAM. Dalam kebanyakan sistem Linux, *file* konfigurasi ini terdapat di `/etc/pam.d`, dan dinamakan menggunakan nama dari *service* tersebut. Contoh, untuk *service* login, *file* konfigurasi. Contoh *file* tersebut dapat dilihat dibawah ini :

Tabel 2.5. Tabel Contoh File

Service	Module_type	Control_flag	Module_path	Options
login	mapping	sufficient	libmap.so	
login	mapping	sufficient	libmapfoo.so	
login	auth	required	pam_unix_auth.so	nowarn
login	session	required	pam_unix_session.so	
login	account	required	pam_unix_account.so	
login	password	required	pam_unix_passwd.so	
ftp	auth	required	pam_skey_auth.so	debug
ftp	session	required	pam_unix_session.so	
telnet	session	required	pam_unix_session.so	
passwd	mapping	required	libmap.so	
passwd	mapping	required	libmapfoo.so	
passwd	password	required	pam_unix_passwd.so	
OTHER	auth	required	pam_unix_auth.so	
OTHER	session	required	pam_unix_session.so	
OTHER	account	required	pam_unix_account.so	
OTHER	password	required	pam_unix_passwd.so	

2.8.3 Management Zona PAM

Hal yang pertama kali perlu diperhatikan dalam *file* tersebut adalah kolom yang paling kanan, dimana semuanya memiliki 4 kata unik, yang mewakili keempat zona management PAM: auth, *account*, *password*, dan *session*. Walaupun terdapat banyak module yang mendukung lebih dari satu jenis zona ini (seperti pam_unix yang mendukung semuanya), yang lain seperti pam_cracklib hanya bisa satu. Dengan mengetahui lingkungan keempat zona ini bekerja, sangat penting untuk menggunakan PAM dengan efektif . Berikut adalah penjelasan dari keempat zona tersebut.

a. auth

zona auth (dapat dikatakan zona karena dalam dokumen dikatakan sebagai managemen grup atau fasilitas) bekerja untuk memeriksa bahwa *user*

adalah sesuai dengan yang digunakan. Modul yang bisa didaftarkan dalam area ini umumnya mendukung pengecekan *password*.

b. *account*

zona ini bekerja untuk verifikasi *account* dalam skala luas. Terdapat banyak modul yang tersedia untuk fasilitas ini. Hal-hal yang berhubungan dengan pemeriksaan keanggotaan grup, waktu dan hari, apakah *account* itu *local* atau *remote*, daln lain-lain. Biasanya dikerjakan oleh modul yang mendukung fasilitas ini.

c. *password*

Modul dalam area ini bertanggung jawab atas kebutuhan fungsional dalam *service* yang mengubah *password*. Prosesnya cukup sederhana,memanggil modul untuk memasukkan *password* saat ini, dan apabila berhasil, akan membuatkan *password* baru. Modul lainnya dapat ditambahkan untuk menambah kompleksitas *password* atau pengecekan kamus, seperti yang dilakukan oleh modul pam_cracklib dan pam_pwcheck.

d. *session*

Modul di area ini melakukan berbagai hal yang terjadi saat *setup* dan *cleanup* sebuah *service* dari seorang pemakai. Beberapa diantaranya antara lain: menjalankan *script* inisialisasi sistem secara luas, menjalankan *logging* khusus, *mounting* direktori asal pemakai, atau penyetingan pembatas *resources*.

2.8.4 Kontrol Modul PAM

Kolom bagian tengah berisi kata kunci yang penting untuk menentukan apa yang dilakukan oleh PAM apabila modul tersebut gagal atau berhasil. Kata kunci ini disebut "kontrol" dalam bahasa PAM. Perlu dicatat bahwa kata kunci ini bukanlah indikator modul, tapi hanya menunjuk pada *library* PAM. Kemungkinan dalam 90% kasus, kita dapat menggunakan satu dari beberapa kata kunci yang umum (*requisite*, *required*, *sufficient*, atau *optional*). Tapi, ini hanyalah saran dari *iceberg* untuk melepas fleksibilitas dan kekuatan dari PAM. Tapi tergantung pada situasi maka *syntax* dapat diperumit. Berikut adalah penjelasan beberapa kata kunci PAM:

1. Required

Jika sebuah modul *required* mengembalikan status gagal, maka operasi akan seluruhnya gagal, tapi hanya setelah modul yang sesuai dijalankan. Ini terlihat biasa pada awalnya, namun hal ini juga berlaku sama untuk *service-service* yang lain. Dengan adanya ini maka akan mustahil bagi hacker untuk mengetahui modul yang mana yang gagal, dan semakin sedikit informasi yang diterima *user* yang usil, maka akan semakin baik. Penting untuk dicatat, apabila seluruh modul dalam antrian jalan, namun salah satunya ada yang gagal, maka seluruh operasinya akan dianggap gagal. Dan operasi masih dapat gagal juga apabila jika setelahnya ada "*required*" modul gagal.

]

2. Requisite

Jika sebuah modul "*requisite*" gagal, maka operasi tersebut itu tidak hanya sekedar gagal, tapi operasi juga akan secepat mungkin dihancurkan dengan kegagalan tanpa menjalankan modul lain.

3. Sufficient

Jika sebuah modul *sufficient* berhasil, itu cukup untuk memenuhi kebutuhan modul *sufficient* di zona itu yang bersusuaian *service*nya, dan modul *sufficient* berikutnya tidak akan dijalankan. Jika ini gagal, operasi akan gagal kecuali sebuah modul menjalankannya setelah berhasil. Penting untuk dicatat bahwa apabila *required* modul gagal sebelum *sufficient* berhasil, operasi akan gagal, mengabaikan status modul *sufficient*.

4. Optional

Sebuah modul yang bersifat opsional atau pilihan, yang akan menyebabkan operasi gagal apabila modul satu-satunya dalam fasilitas ini gagal.

2.8.5 Option

Dan dalam bagian yang paling kanan, dalam beberapa bagian terdapat informasi tambahan. Informasi tambahan ini adalah option. Option ini merupakan tambahan pilihan dari module yang dikerjakan (disebelah kirinya). Option-option yang terdapat dalam PAM antara lain:

Debug

Mengaktifkan metode untuk debugging

Config_*file*

Untuk menspesifikasi dari *file* konfigurasi tersebut (umumnya `/etc/pkcs11/pam_pkcs11.conf`)

Option berikut ini diambil dari *file* konfigurasi, tapi terserah dari *user* untuk menspesifikasikan dari command line. Jadi dapat diaktifkan diluar *file* konfigurasi

Nullok

Mempersilahkan *password* kosong

Use_first_pass

Jangan meminta *user* untuk memasukkan *password*, tapi ambil saja dari PAM_items.

Try_first_pass

Jangan meminta *user* untuk memasukkan *password* kecuali PAM_(OLD)AUTHOK telah dilepas.

Use_authok

Sama seperti try_first_pass, tapi gagal apabila PAM_AUTHOK yang baru sebelumnya belum diatur (ditujukan hanya untuk modul penumpukan *password*)

2.8.6 PAM Dalam Linux Debian

Sebelumnya telah dijelaskan secara umum tentang implementasi PAM. Berikut akan dijelaskan sedikit tentang PAM dalam sistem operasi Linux Debian.

Perbedaannya adalah dari penempatan *file* konfigurasi dan pengaturan *file* konfigurasi. *File* konfigurasi PAM ditempatkan dalam *folder* `/etc/pam.d`. Berikut *file-file* yang terdapat dalam *folder* tersebut yang akan sering digunakan dalam pengerjaan Tugas Akhir ini:

- Common-auth → Ini adalah *file* konfigurasi zona auth yang umum digunakan dalam *file* konfigurasi pam lainnya. Dalam beberapa *file* konfigurasi lainnya, *file* ini dimasukkan sebagai tambahan.
- Common-*account* → Ini adalah *file* konfigurasi zona *account* yang umum digunakan dalam *file* konfigurasi pam lainnya. Dalam beberapa *file* konfigurasi lainnya, *file* ini dimasukkan sebagai tambahan.
- Common-*session* → Ini adalah *file* konfigurasi zona *session* yang umum digunakan dalam *file* konfigurasi pam lainnya. Dalam beberapa *file* konfigurasi lainnya, *file* ini dimasukkan sebagai tambahan.
- Common-*password* → Ini adalah *file* konfigurasi zona *password* yang umum digunakan dalam *file* konfigurasi pam lainnya. Dalam beberapa *file* konfigurasi lainnya, *file* ini dimasukkan sebagai tambahan.
- Login → Ini adalah *file* konfigurasi PAM untuk aplikasi login.
- Gdm → Ini adalah *file* konfigurasi PAM untuk aplikasi gdm.
- Ssh → Ini adalah *file* konfigurasi PAM untuk aplikasi ssh.

2.9. KERBEROS

2.9.1 Penjelasan

Kerberos adalah sistem otentikasi yang dikembangkan oleh MIT(Masachuset Institute Technology) sebagai bagian dari proyek Athena. Kerberos menggunakan teknologi enkripsi dan pihak ketiga yang terpercaya, sebuah *arbitrator*, untuk melakukan otentikasi yang aman di jaringan yang terbuka. Secara spesifik, Kerberos menggunakan *cryptographics tickets* untuk menghindari mengirim *password* yang berupa teks biasa lewat jaringan. Kerberos bekerja berdasarkan protokol Needham-Schroeder.

Terdapat dua versi Kerberos yang sekarang digunakan, versi 4 dan 5. Versi 1, 2, dan 3 merupakan versi pengembangan internal dan tidak pernah di *release*. Kerberos versi 4 memiliki banyak kelemahan dan sebaiknya tidak digunakan. Dan yang digunakan oleh Windows *Server* 2003 adalah versi 5.

2.9.2 Cara Kerja Kerberos

Kerberos adalah protokol otentikasi yang menggunakan sebuah pihak ketiga yang rahasia dan terpercaya untuk memvalidasi identitas *client*. Di Kerberos *client* dapat berupa *user*, *server*, atau program. Pihak ketiga *arbitrator* yang terpercaya adalah *server* yang disebut *Key Distribution Center*(KDC) yang menjalankan *daemon* Kerberos. Rahasia yang dibagi adalah *password user* yang telah di-*cryptographi*.

Di Kerberos, *user* dikatakan sebagai *principals*. KDC memiliki *database principal* dan kunci rahasianya yang digunakan untuk melakukan otentikasi. Di Kerberos pengetahuan kunci rahasia dikatakan cukup sebagai bukti

identitas, *server* Kerberos dapat mempercaya *client* manapun untuk diotentikasi. Otentikasi di kerberos dilakukan tanpa mengirim *password* berupa teks di jaringan. Berikut akan dijelaskan bagaimana pemetaan protokol Kerberos di sistem GNU/Linux.

KDC menjalankan 2 *daemon* Kerberos yang penting. *Daemon* ini adalah kadmin dan krb5kdc. Kedua *daemon* ini jalan sebagai root.

Kadmins adalah *daemon* administrasi Kerberos *server*. Kadmins digunakan oleh program kadmin untuk menjaga *database principals* dan konfigurasi *policy*. Jika dipilih untuk melarang *remote* login melalui ssh dari perangkat Kerberos, kadmin akan mengijinkan kamu untuk me-*remote* administrasi komponen Kerberos yang ada di *server*.

Krb5kdc bertugas untuk menjalankan peran sebagai *arbitrator* pihak etiga yang dipercaya di otentikasi Kerberos. Saat *user* mau meng-otentikasi dirinya ke sebuah sistem atau *service*, *user* meminta tiket dari KDC. Tiket adalah sebuah *datagram* yang berisi identitas *client*, *session key*, *timestamp*, dan informasi lainnya. *Datagram* tersebut dienkripsi dengan kunci rahasia *server*.

Berikut prosesnya secara detail, pertama permintaan untuk otentikasi dikirim ke daemon krb5kdc. Saat daemon menerima permintaan ini, dy melihat ke *client*, si principal, mencoba me-otentikasi database principal. Dia membaca kunci rahasia *client* dari database dan meng-encrypt sebuah tiket khusu bernama Ticket Granting Ticket(TGT) yang kemudian dikirim ke *client*. *Client* menerima TGT yang ter-encript ini yang mengandung *session* key. Jika *client* mengetahui *password*nya(kunci rahasia yang tersimpan di

database principal) dan sukses men-decrypt, ini dapat merepresentasikan tiket yang ter-enkripsi dengan enclosed *session* key ke TGS. TGS akan meng-isukan tiket berikutnya yang menyediakan *client* dengan otentikasi yang dibutuhkan untuk menggunakan sistem atau service tersebut.

2.9.3 Instalasi Kerberos di GNU/Linux

Pertama, lakukan instalasi yang benar-benar dibutuhkan oleh kerberos. Ini termasuk sistem operasi dasar dan paket kerberos. Kemudian *install* X atau aplikasi GUI lainnya, SSH terserah. SSH dapat diinstal apabilaanda ingin bisa mengatur *server* denagn *remote*.

Di Fedora core berbasis GNU/Linux, paket yang dibutuhkan untuk service kerberos adalah :

1. ***Krb5-server***
2. ***Krb5-libs***

Kemudian adalah memastikan tidak ada port yang tidak perlu yang terbuka dan *install security patches* yang dibutuhkan. Metode untuk menentukan *patch security* apa saja yang dibutuhkan tergantung dari paket *software* apa saja yang telah ter-*install*. Untuk menentukan port apa saja yang sedang di-*listen*, dapat digunakan perintah `netstat`. Sebagai contoh, dalam sistem yang menjalankan SSH terlihat seperti ini:

```
bash$ netstat -an | grep -i listen | less
tcp        0      0 0.0.0.0:22                  0.0.0.0:*
LISTEN
```

Gambar 2.15. Netstat ssh

2.9.4 Konfigurasi Kerberos

Berhubung yang dikerjakan dalam pengerjaan tugas akhir ini hanya menggunakan Kerberos dari sisi *client*, maka konfigurasi kadmin dan *server* Kerberos tidak dibahas.

Konfigurasi Kerberos dapat dilakukan melalui *file* konfigurasi /etc/krb5.conf. Dalam *file* ini harus di set realm yang digunakan, tambahkan definisi realm dengan menspesifikasi *server* Kerberos, dan akhirnya *setting domain realm*. Contohnya dapat dilihat sebagai berikut:

```
default_realm = GNUD.IE

[realms]
 GNUD.IE = {
  kdc = kerberos1.gnud.ie:88
  kdc = kerberos2.gnud.ie:88
  admin_server = kerberos1.gnud.ie:749
  default_domain = gnud.ie
 }

[domain_realm]
 .gnud.ie = GNUD.IE
 gnud.ie = GNUD.IE
```

Gambar 2.16. Spesifikasi Server Kerberos

Kemudian Kerberos dapat dijalankan secara manual dengan perintah :

```
{Kerberos1}bash# /etc/rc.d/init.d/krb5kdc start
```

Gambar 2.17. Menjalankan Kerberos

2.10. SUPERUSER DO (SUDO)

2.10.1 Penjelasan

Sudo mengijinkan sistem administrator untuk memberi beberapa *user* atau grup dari beberapa *user* kemampuan untuk menjalankan beberapa atau semua perintah sebagai root. Sudo beroperasi berdasarkan per-perintah, dan tidak menggantikan *shell*. Fitur yang dimiliki antara lain:

Kemampuan untuk membatasi perintah apa yang dapat dijalankan oleh *user*.

Sudo memiliki daftar *log* dari tiap perintah, memberikan laporan yang jelas tentang siapa yang menjalankan dan menjalankan apa. Saat digunakan bersamaan dengan syslogd (sistem log daemon), sudo dapat mencatat semua perintah ke pusat dan *client*.

Sudo menggunakan *file* timestamp untuk mengimplementasikan sistem tiket. Saat *user* menggunakan sudo dan memesukkan *password* mereka, mereka juga diberikan tiket selama 5 menit (waktu ini dikonfigurasi saat di-*compile*). Tiap pemakain perintah sudo akan menambah tiket untuk 5 menit lagi. Ini untuk menghindari masalalah saat meninggalkan *shell* root dan orang lain dapat mengakses keyboard.

File konfigurasi sudo (*file* sudoers), adalah *file* untuk mengkonfigurasi dan mengatur pemakain sudo juga dapat diakses dibanyak komputer. Sehingga mendukung administrasi terpusat dan juga mebiarkan fleksibilitas *user* dalam tiap *client*.

2.10.2 Konfigurasi sudo

File sudoers terdiri dari 2 tipe masukan: alias (dasarnya adalah variabel) dan spesifikasi *user* (yang menspesifikasikan siapa menjalankan apa).

Saat banyak masukan yang sesuai dengan satu *user*, daftar diaplikasikan sesuai urutan. Saat ada nilai yang konflik, maka yang terkhir sesuai digunakan (yang belum tentu paling sesuai).

2.10.2.1 Alias

Terdapat 4 jenis alias: *user*_alias, runas_alias, host_alias, cmnd_alias. Dan bentuk umum dari alias tersebut adalah:

Alias_Type *NAME* = item1, item2, ...

Dimana Alias_Type adalah salah satu dari keempat jeins alias tersebut. Dan *NAME* adalah kata-kata huruf besar, nomor, dan karakter *underscore* ('_'). Sebuah *NAME* harus dimulai dengan huruf besar. Dan memungkinkan untuk menaruh beberapa definisi alias bersamaan dalam satu baris dan dipisahkan oleh titik dua (':'). Contoh:

Alias_Type *NAME* = item1, item2, item3 : *NAME* = item4, item5

Pemakain sudo dalam pengerjaan TA ini akan lebih fokus kepada pemakaian spesifikasi *user*. Sehingga pemakaian alias tidak dibahas terlalu dalam buku ini.

2.10.2.2 Spesifikasi *User*

Spesifikasi *user* menentukan perintah apa yang dapat dijalankan oleh *user* dan siapa *user* itu dalam tiap tempat. Secara default, perintah berjalan sebagai root. Tapi ini dapat dirubah dengan menambahkan per baris. Bentuk umum spesifikasi user ini dapat dilihat dalam gambar 2.18.

```
User_Spec ::= User_List Host_List '=' Cmnd_Spec_List \
              (':' Host_List '=' Cmnd_Spec_List)*
 Cmnd_Spec_List ::= Cmnd_Spec |
                    Cmnd_Spec ',' Cmnd_Spec_List
 Cmnd_Spec ::= Runas_Spec? Tag_Spec* Cmnd
 Runas_Spec ::= '(' Runas_List ')'
 Tag_Spec ::= ('NOPASSWD:' | 'PASSWD:' | 'NOEXEC:' | \
'EXEC:')
```

Gambar 2.18. Bentuk umum spesifikasi user

Bagian yang paling kiri adalah nama *user*, dimana contoh pemakaiannya dapat dilihat dalam gambar 2.19.

```
dgb    boulder = (operator) /bin/ls, (root) /bin/kill,
/usr/bin/lprm
```

Gambar 2.19. Contoh pemakaian user

Dimana artinya adalah *user* dgb dapat menjalankan /nin/ls sebagai grup operator, tapi untuk /bin/kill dan /usr/bin/lrpm sebagao grup root.

2.11. SHADOW

2.11.1 Penjelasan

Di dalam Linux yang tidak menggunakan shadow, informasi *user* termasuk *password* disimpan dalam *file* `/etc/passwd`. *Password* disimpan dalam format terenkripsi. Dan *file* ini dapat dibaca oleh siapapun karena beberapa aplikasi membutuhkan *file* ini, namun ini memancing resiko *password* dapat dipecahkan enkripsinya. Dan shadow ini menyelesaikan masalah tersebut dengan merelokasi *password* ke *file* lain (biasanya `/etc/shadow`). Dan *file* shadow ini telah diatur agar tidak bisa dibaca oleh siapapun kecuali root yang dapat mengaksesnya. Dengan menyembunyikan *file* ini, kita dapat mencegah penyerangan dari hacker karena tidak dapat mengakses *password*.

Terlebih lagi shadow juga menawarkan fitur-fitur lain antara lain:

1. Sebuah *file* konfigurasi untuk mengatur *default* login (`/etc/login.defs`)
2. Utilitas untuk menambahkan, merubah, dan menghapus *user* dan grup
3. Usia dan pengkadaluarsa *password*
4. Pengkadaluarsaan dan penguncian *user*

5. *Password* grup yang menggunakan shadow (tambahan)
6. Ukuran *password* 2 kali ukuran biasa (16 karakter)
7. Kontrol yang lebih baik terhadap pemilihan *password user*
8. *Password dial-up*

2.11.2 Format *File* /etc/passwd

Berikut adalah format *file* `/etc/passwd`:

```
username:passwd:UID:GID:full_name:directory:shell
```

Gambar.2.20. Bentuk umum isi file /etc/passwd

Dan atribut-atributnya berisi:

- *Username* → Nama *user* dari login tersebut
- Passwd → Apabila tidak menggunakan shadow maka bagian ini akan berisi *password* yang dienkripsi. Kalau menggunakan shadow maka ini hanya berisi karakter "x".
- UID → Nomor id dari *user* tersebut
- GID → Nomor grup standar dari *user* tersebut
- Full_*name* → Nama lengkap dari *user* tersebut
- Directory → Direktori *home* dari *user* tersebut
- Shell → Shell dari login tersebut (*path* yang lengkap)

Berikut adalah contoh salah satu baris dalam *file* `/etc/passwd` yang tidak menggunakan shadow dan yang menggunakan shadow

```
username:Npge08pfz4wuk:503:100:Full
Name:/home/username:/bin/sh
```

Gambar 2.2.21. Contoh /etc/passwd tanpa menggunakan shadow

```
username:x:503:100:Full
Name:/home/username:/bin/sh
```

Gambar 2.2.22. Contoh /etc/passwd dengan menggunakan shadow

2.11.3 Format *File* /etc/shadow

Berikut adalah format *file* /etc/shadow:

```
username:passwd:last:may:must:warn:expire:disable:reserved
```

Gambar 2.23. Bentuk umum isi /etc/shadow

Dan atribut-atributnya berisi:

- *Username* → Nama *user* yang digunakan untuk login
- *Passwd* → *Password user* yang telah terenkripsi
- *Last* → Umur *password* dari perubahan terakhir
- *May* → Jumlah hari sebelum passwrod dapat dirubah
- *Must* → Hari saat perubahan *password*
- *Warn* → Jumlah hari sebelum *password* kadaluarsa
- *Expire* → Hari saat *password* kadaluarsa dan *account* di *disable*
- *Disable* → Jumlah hari setelah *account* di *disable*
- *Reserved* → Field yang di pesan

Contohnya dalam *file* mungkin aka seperti ini:

```
username:Npge08pfz4wuk:9479:0:10000:::
```

Gambar2. 2.24. Contoh isi file /etc/shadow

2.12. OpenSSH

2.12.1 Penjelasan

OpenSSH adalah versi gratis dari paket protokol ssh dari tools konektifitas jaringan yang meningkat jumlah pemakai di internet yang menggunakannya. Banyak orang menggunakan telnet, rlogin, ftp, dan beberapa program lain tidak menyadari bahwa *password* mereka di lewatkan di internet tidak di-enkripsi. OpenSSH meng-enkripsi tiap arus (termasuk *password*) to menghilangkan penyerangan dan pembajakan dari jaringan. Terlebih lagi, OpenSSH menyediakan banyak metode keamanan *tunneling*, sebagai variasi dari metode otentikasi.

Paket OpenSSH menyediakan program ssh yang menggantikan rlogin dan telnet, scp yang menggantikan rcp, dan sftp yang menggantikan ftp. Juga termasuk didalamnnya sshd yang merupakan bagian *server* dari paket ini, dan berbagai *utility* dasar lain seperti ssh-add, ssh-agent, ssh-keysign, ssh-keyscan, ssh-keygen, dan sftp *server*. OpenSSH mendukung protocol ssh versi 1.3, 1.5, dan 2.0.

2.12.2 Fitur OpenSSH

Berikut adalah fitur-fitur yang ditawarkan oleh OpenSSH:

1. Proyek *Open Source*
2. Lisensi gratis
3. Enkripsi yang kuat (3DES, Blowfish, AES, Arcfour)
4. *Forwarding* X11 (Sistem arus X Window yang di-enkripsi)
5. *Forwarding Port* (JAlur yang di-enkripsi untuk Legacy Protocol)

6. Otentikasi yang kuat (*Public Key*, *One-Time Password*, dan otentikasi Kerberos)
7. *Forwarding Agent* (*Single-Sign-On*)
8. Sesuai dengan standar protokol SSH 1.3, 1.5, dan 2.0.
9. Mendukung *client* dan *server* sftp dalam kedua protokol ssh1 dan ssh2
10. Kompresi Data

BAB 3

PERANCANGAN PERANGKAT LUNAK

BAB 3

PERANCANGAN PERANGKAT LUNAK

Bab ini membahas tentang perancangan perangkat lunak untuk melakukan otentikasi dari Sistem Operasi Linux Debian menggunakan login *domain* Windows *Server* 2003. Dimana pada awalnya otentikasi di Linux hanya dapat meminta informasi *user* dari database *user* yang ada di dalam console Linux. Namun Pada dengan ditambahkan beberapa konfigurasi tambahan, otentikasi di Linux dapat meminta informasi dari database *user* yang lain. Dan aplikasi ini memanfaatkan sistem seperti itu, dengan menggunakan database *user* yang berada di Active Directory milik Windows *Server* 2003. Perancangan perangkat Lunak ini meliputi deskripsi, perancangan sistem *domain*, perancangan koneksi Linux dengan Window, perancangan sistem otentikasi, perancangan aplikasi otentikasi.

3.1. DESKRIPSI PERANGKAT LUNAK

Sistem login komputer lokal dengan login *domain* yang telah ada biasanya hanya dilakukan pada komputer yang memiliki OS sama. Hal ini dikarenakan adanya perbedaan proses *autentifikasi* antara OS yang berbeda. Secara umum proses login pada komputer *client* yang ber-OS Linux dapat digambarkan seperti pada gambar 3.1 di bawah ini.

Gambar 3.1. Proses login dalam Linux secara umum

Pada gambar 3.1 ditunjukkan bahwa proses login dibagi menjadi 2 bagian besar yaitu pengecekan *user* dan *autentifikasi*. Kedua proses inilah yang menjadi jembatan komunikasi *login* komputer *client* dan komputer *server*. Melalui kedua proses inilah komputer *client* dapat login masuk dengan menggunakan *user* pada komputer *domain* (*server domain*). Setelah proses pengecekan *user* berhasil, maka *user* dapat menggunakan komputer tersebut sesuai *account*-nya.

Kemudian terdapat masalah. Pemilik *account domain* tidak memiliki *account* dalam komputer client. Sehingga saat *account domain* digunakan di komputer client. *Client* harus dapat membuatkan *account* yang dapat digunakan di lingkungan lokal mereka sendiri berdasarkan konfigurasi *user* pada *account domain*.

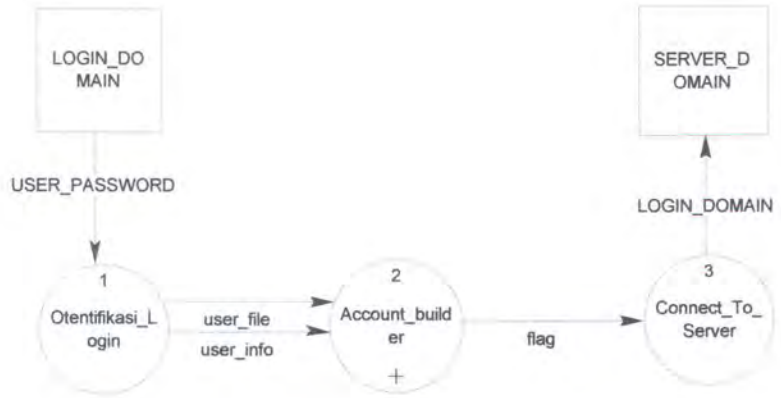
Namun permasalahan tersebut telah terselesaikan apabila OS *server* dan client yang terdapat dalam *domain* berasal dari *vendor* yang sama. Semisal Windows. Namun tetap menjadi masalah apabila diterapkan pada komputer *client* dan komputer *server* yang memiliki OS berbeda. Hal ini disebabkan karena proses login antar OS memiliki perbedaan. Sebagai uji coba pada tugas akhir ini akan digunakan komputer *client* ber-OS linux dan komputer *server* ber-OS Windows *server* 2003.

Masing-masing OS memiliki cara yang berbeda dalam melakukan proses login, karenanya agar proses login dengan OS yang berbeda dapat berjalan maka perlu adanya semacam jembatan yang menghubungkan komunikasi antara keduanya. Dan berhubung dari sisi *client* menggunakan OS yang berbeda, maka aplikasi yang meminta dan menerima otentikasi juga perlu dirubah. Pada gambar

2.2 akan dijelaskan desain dari jembatan dan aplikasi yang menghubungkan antara Windows dengan linux.

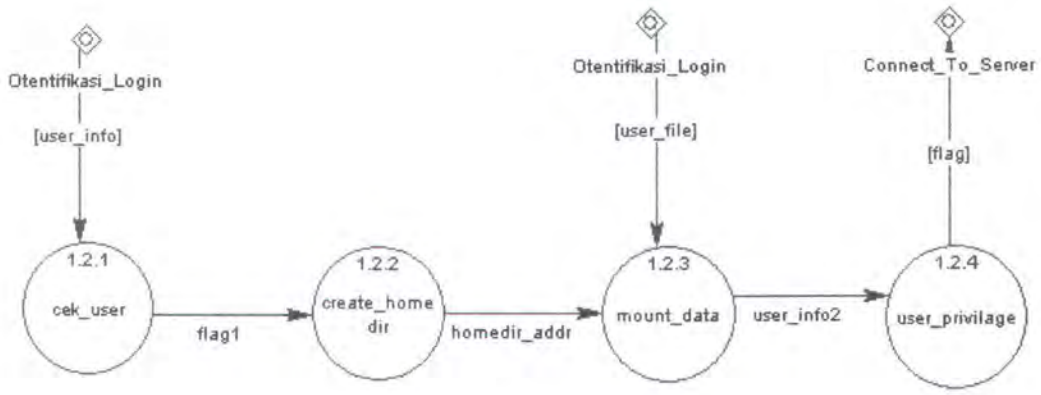
Gambar 3.2. DFD level 0

Dari DFD level 0 pada gambar 3.2 di atas, proses dapat dibagi menjadi 3 bagian yaitu *autentifikasi* login, pembangunan *user account*, dan menghubungkan dengan *server domain*. Untuk lebih jelasnya dapat dilihat DFD level 1 pada gambar 3.3.



Gambar 3.3. DFD level 1

Dan dari DFD level 1 pada gambar 3.3 hanya proses 2 yang dapat di pecah menjadi beberapa proses lagi. Proses 2 dapat dipecah menjadi 4 proses lagi, yaitu pengecekan user, pembuatan *home* direktori, pengambilan data user, dan pengaturan hak user. Untuk lebih jelasnya dapat dilihat DFD level 2 pada gambar 3.4.



Gambar 3.4. DFD level2

3.2. PERANCANGAN SISTEM *DOMAIN*

Dalam sub bab ini akan dijelaskan tentang perancangan sistem *domain* yang akan digunakan dalam pengerjaan TA ini. Ini merupakan satu-satunya perancangan yang diterapkan dalam sisi *server*. Perancangan sistem *domain* ini dibutuhkan karena tidak semua sistem *domain* milik AD dapat diterapkan untuk *domain* dengan *client* Linux.

Berikut akan dijelaskan sistem yang perlu diatur dalam AD ini agar sesuai dengan kemampuan *client*.

3.2.1 Perancangan Sistem *User*

Sistem *user* yang dirancang dalam *domain* ini dirancang untuk membuat semua *user* memiliki *home* direktori masing-masing. Dan semua *home* direktori tersebut berada dalam sebuah *folder* yang di-*share* oleh AD.

3.2.2 Perancangan Sistem Akses *Domain*

Pada pengaturan awal AD, OS selain Windows tidak dapat mengakses AD. Untuk itu perlu dirancang agar AD dapat diakses oleh OS selain Windows. Dimana dalam pengerjaan TA ini OS ini digunakan OS Linux

sebagai *client*. Sehingga hasil perancangan ini *client* dapat melakukan proses otentikasi dengan AD.

3.3. PERANCANGAN KONEKSI LINUX DENGAN WINDOWS

Untuk dapat memulai suatu proses otentikasi dalam suatu jaringan dibutuhkan suatu metode komunikasi yang digunakan untuk meminta dan menerima otentikasi. Dalam kasus suatu *domain* dimana seluruh anggotanya (*server* dan *client*) menggunakan OS Windows. Tidak perlu dilakukan suatu sistem khusus karena telah digabungkan dalam sistem *domain* Windows. Namun berhubung dalam pengerjaan TA ini *client* menggunakan OS Linux. Dibutuhkan suatu sistem khusus yang dimiliki oleh *client* agar dapat meminta dan menerima otentikasi.

Untuk itu agar komunikasi antara komputer *client* yang menggunakan linux dapat meminta otentikasi dan menerima informasi *user* pada *domain* yang menggunakan Windows, maka perlu semacam jembatan atau penghubung. Salah satu penghubung login *domain* yang berbeda OS ini yaitu menggunakan paket-paket pada komputer *client* yang telah dikonfigurasi berbeda dengan konfigurasi pada umumnya. Paket-paket tersebut dapat dibagi menjadi 2 bagian yaitu :

1. Samba dan Winbind.
2. Kerberos

3.3.1 Samba dan Winbind

Samba, seperti yang telah dijelaskan pada bab II, merupakan salah satu penghubung komunikasi jaringan antara komputer yang menggunakan linux dengan komputer yang menggunakan OS selain Linux. Aplikasi samba ini

adalah aplikasi tambahan yang berjalan pada linux. Sehingga pada intalasi linux biasanya belum terdapat aplikasi samba di dalamnya.

Paket samba ini dikonfigurasi untuk dapat menggunakan salah satu modulnya yaitu winbind. Dan dapat membuka jalur komunikasi dengan AD. Dan dapat melakukan pertukaran informasi.

3.3.2 Kerberos

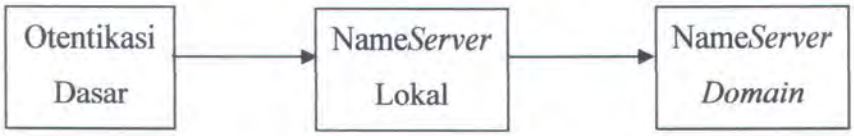
Server domain yang digunakan pada aplikasi yang dibangun ini menggunakan OS Windows *server* 2003. Proses otentifikasi login pada Windows 2003 menggunakan Kerberos. Oleh karena itu agar komunikasi proses login pada *client* (linux) dengan *server domain* (Windows) dapat berjalan, maka pada sisi *client* juga harus menggunakan proses otentifikasi yang sama dengan Windows 2003, yaitu kerberos. Untuk itu pada *client* harus memiliki Kerberos *client* agar dapat mengirim data yang dapat diterima oleh AD dan mengolah data yang dikirim oleh AD.

3.4. Perancangan Sistem Otentikasi

Setelah komunikasi antara *client* dengan *server* telah dijalankan maka tahap berikutnya dapat dijalankan yaitu tahap otentikasi. Pada dasarnya sistem otentikasi milik Linux melakukan otentikasi dengan *name server* miliknya sendiri (console). Sehingga perlu dirancang suatu sistem dalam Linux agar dapat melakukan otentikasi dengan *name server* lain yang terhubung dalam jaringan.

3.4.1 Perancangan Sistem Otentikasi Dasar

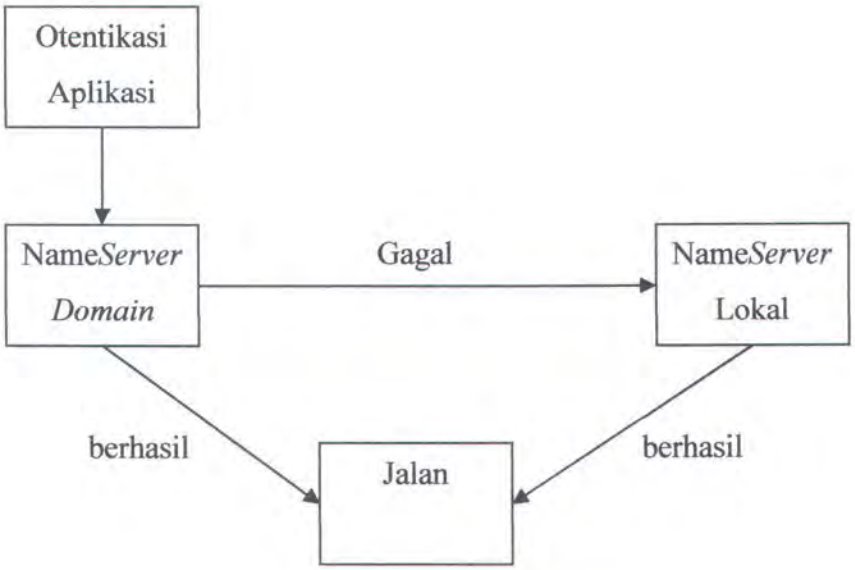
Yang dimaksud sistem otentikasi dasar di sini adalah otentikasi yang digunakan Linux untuk mengetahui siapa saja yang merupakan *user* (termasuk juga group) dari *client* tersebut. Otentikasi dasar ini memegang peranan dalam menetukan siapa yang dapat menggunakan *client*. Sehingga harus diarahkan agar otentikasi in dapat melakukan otentikasi dengan AD dan lokal. Gambaran atau skema dari otentikasi dasar ini dapat dilihat pada gambar 3.5.



Gambar 3.5. Skema otentikasi dasar

3.4.2 Perancangan Sistem Otentikasi Aplikasi

Sistem otentikasi yang dimaksud disini adalah otentikasi yang dilakukan oleh aplikasi yang dijalankan oleh Linux untuk memeriksa siapa yang menjalankan aplikasi tersebut. Sehingga harus diarahkan agar otentikasi in dapat melakukan otentikasi dengan AD. Gambaran atau skema dari proses otentikasi aplikasi ini dapat dilihat pada gambar 3.6.



Gambar 3.6. Skema otentikasi aplikasi

3.5. PERANCANGAN APLIKASI OTENTIFIKASI

Setelah dapat melakukan otentikasi. Maka aplikasi dalam linux sendiri juga perlu dirancang agar dapat menerima input otentikasi yang diluar keadaan standar Linux. Yaitu apabila menggunakan *account domain*.

Dari hasil perancangan sebelumnya, dapat diasumsikan bahwa *client* yang menggunakan sistem operasi Linux Debian sudah dapat login menggunakan login *domain*nya. Karena koneksi Linux dengan Windows sudah berjalan dengan baik, dan proses autentifikasi sudah bisa diarahkan menggunakan aplikasi lain, yakni winbind. Hal ini dapat dilihat seperti dalam gambar berikut ini.

Namun hal tersebut masih memiliki beberapa kekurangan, antara lain:

1. *User* yang login menggunakan *account domain* tidak memiliki *home* directory pada *client*.
2. Direktori milik *user* yang berada di dalam *server* belum dapat diakses secara langsung oleh *user*.

3. *Domain* administrator tidak memiliki hak khusus dalam *client*.

Untuk mewujudkan ketiga hal tersebut. Perlu dilakukan modifikasi khusus dalam sistem login yang dimiliki oleh Linux Debian. Modifikasi yang dilakukan yaitu dengan cara meng-*compile* ulang paket yang dimiliki Linux Debian. Modifikasi ini dibagi menjadi 3 bagian yaitu :

- Pembangunan sistem login *remote*.
- Pembangunan sistem login dan administrasi.
- Pembangunan sistem login GUI.

3.5.1 Perancangan Pembangunan Sistem Terminal Login

Sistem terminal login merupakan login pada Linux yang dilakukan di komputer *client* langsung yang menggunakan login berupa *text based* dalam window linux (tty) yang tidak menggunakan GUI.

Untuk membangun Aplikasi *terminal login* maka perlu modifikasi pada source paket shadow. Paket ini menangani tentang *terminal login* Linux dan aplikasi-aplikasi lain yang berhubungan dengan otentikasi dan manajemen user.

3.5.1.1 *File* dan paket yang dibutuhkan

File milik paket shadow yang didapat dari internet ada 2 yaitu:

- shadow_4.0.3.orig.tar.gz → *File* kompresi yang berisi *cource code* dari paket shadow.
- shadow_4.0.3-31sarge5.diff.gz → *File* kompresi yang berisi patch dari paket shadow.

Dan paket-paket tambahan yang dibutuhkan untuk dapat meng-*compile* dan meng-*install* shadow antara lain:

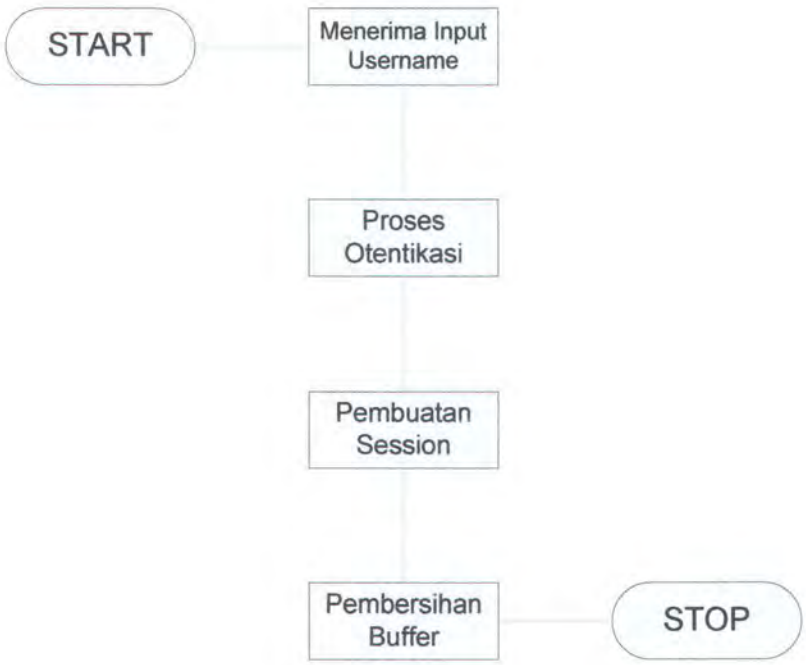
- Libpam-dev
- Zlib1g-dev

3.5.1.2 Keterangan isi paket

Paket shadow-4.0.3 berisi dari *cource code* dari aplikasi dan header dari proses-proses login atau otentikasi dan manajemen user. Termasuk diantaranya aplikasi adduser, deluser, addgroup, delgroup, chpasswd , dan lain-lain. Dan header-header otentikasi, pembuatan *session,* penutupan *session*, pembuatan user, penghapusan user, dan lain-lain. Dan total isi paket shadow-4.0.3 yang telah di*compile* ulang berisikan 12,9 MB data yang teridiri dari 1064 *file* dan 36 folder.

3.5.1.3 Proses aplikasi terminal login

Secara umum proses *terminal* login dapat dibagi menjadi 3 bagian utama dengan pembagian dan alur yang dapat dilihat pada gambar.



Gambar 3.7. Diagram proses aplikasi terminal login

Dalam *file* login .c dapat dilihat bahwa proses untuk meminta username dapat dilihat fungsinya seperti dalam gambar.

```
        if (optind < argc) {        /* get the user name */
                if (rflg || (fflg && username[0]))
                        usage ();

#ifdef SVR4
                /*
                 * The "-h" option can't be used with a command-line
username,
                 * because telnetd invokes us as: login -h host
TERM=...
                 */

                if (!hflg)
#endif
                {
                        STRFCPY (username, argv[optind]);
                        strzero (argv[optind]);
                        ++optind;
                }
                }
```

Gambar 3.8. Contoh potongan dari file login.c

Dan untuk proses otentikasinya. Login memiliki 3 metode dengan modul masing-masing, yaitu PAM, shadow, dan Radius. PAM adalah metode aplikasi yang menggunakan modul PAM sebagai metode otentikasi dimana proses otentikasi yang berada diluar paket shadow ini. Shadow adalah metode aplikasi yang menggunakan modul shadow yang masih berada dalam paket ini. Sedangkan Radius adalah metode otentikasi terakhir apabila kedua metode sebelumnya belum berhasil. Namun dalam paket ini shadow ini disebutkan bahwa metode radius ini hanya sebagai tambahan yang secara standar tidak digunakan.

Dan setelah melakukan proses otentikasi. Maka berikutnya akan dilakukan pembuatan *session*. Dalam pembuatan sssion ini termasuk diantaranya adalah masalah pencatatan aktifitas login. Waktu login dan modul login yang digunakan.

Setelah melewati ketiga proses diatas. Maka user telah login ke dalam *client*. Namun aplikasi ini belum selesai. Karena semua variabel dan *buffer* dari pengaksesan *file* harus dihapus untuk keamanan. Karena banyak nilai-nilai variabel yang bahaya apabila dibiarkan dan pengaksesan *file-file* yang berbahaya.

3.5.1.4 Pengecekan *user*

Setelah login dilakukan oleh *user*, maka akan dilakukan pengecekan *user* yang melakukan login, apakah login yang dilakukan oleh *user* tersebut adalah menggunakan *account* pada komputer lokal atau *domain*.

Proses pengecekan user ini cukup memakan waktu, padahal dalam aplikasi login. Saat user telah melewati proses otentikasi. User harus sesegera mungkin untuk membuka *session* yang dimiliki. Apabila proses pengecekan user ini ditempatkan di antara proses otentikasi dan proses pembuatan *session*, maka proses otentikasi gagal. Karena itu, proses pengecekan user diletakkan pada saat user memasukkan *username*.

3.5.1.5 Pembuatan *home* direktori

Setiap user yang login membutuhkan *home* direktori dalam setiap komputer yang digunakan. Baik menggunakan *account* lokal ataupun *account domain*. Home direktori ini dibutuhkan sebagai tempat penyimpanan *profile* dan data milik *user*. Dan saat melakukan login, *user* akan langsung diarahkan ke dalam *home* direktorinya.

Seperti pada login *domain* pada umumnya, ketika *user* menggunaka *account domain* untuk login pada komputer *client*, maka secara otomatis akan dibuatkan *profile* sendiri. *Profile* ini dapat berupa konfigurasi maupun *folder* sesuai yang berada pada *domain*. Karena proses login antara *client* (linux) dan *server domain* (Windows 2003) memiliki perbedaan sehingga mengharuskan dilakukan konfigurasi manual secara berbeda, maka pembuatan *profile* atau direktori inipun harus dikonfigurasi secara manual.

Pada komputer yang ber-OS linux, secara umum *profile* atau *folder* pribadi tiap *user* diletakkan pada direktori `/home/*` (* berarti adalah nama *user* tersebut). Untuk itu pada aplikasi yang dibangun ini *folder user*

yang melakukan login *domain* akan dibuatkan *folder* sendiri pada *folder user* linux pada umumnya.

Dimana dalam kasus ini home direktori para *user domain* berada di folder `/home/TA-FANNI/*`. Maka terlebih dahulu dibuat folder `/home/TA-FANNI` saat penggabungan *domain*. Dan proses ini akan membuat folder khusus bagi para user dalam folder yang telah dibuat sebelumnya.

Fungsi pembuatan folder *home* direktori ini dilakukan setelah proses otentikasi selesai. Dan ketiga fungsi dibawah ini juga bekerja berdasarkan urutan setelah dibuat *home* direktori.

Bagian yang menyatakan bahwa user telah diotentikasi dalam proses login dan akan dimasukkan fungsi-fungsi tambahan dapat dilhat pada gambar 3.9 dengan tanda "Otentikasi sukses".

```
#ifndef USE_PAM
                if (pwd && getdef_bool("FAILLOG_ENAB") &&
                    ! failcheck (pwent.pw_uid, &faillog, failed)) {
                  SYSLOG((LOG_CRIT,    FAILURE_CNT,    failent_user,
fromhost));
                  failed = 1;
                }
#endif

                if (!failed)
                {
//→               Otentikasi sukses
                  break;
                  }

                fprintf(stderr,"Login incorrect\n\n");
#ifndef USE_PAM
                if (pwd && getdef_bool("FAILLOG_ENAB"))
                  failure (pwent.pw_uid, tty, &faillog);
#endif
                if (getdef_str("FTMP_FILE") != NULL)
```

Gambar 3.9. Contoh potongan file login.c

3.5.1.6 Pengambilan *file* dari *server*

Setelah komunikasi antara linux dan Windows dapat dilakukan, yaitu menggunakan aplikasi samba pada linux, maka beberapa *file* konfigurasi pada *server domain* (Windows 2003) dapat diambil. *File* konfigurasi tersebut berisi informasi antara lain :

- Nama *server* (IP).
- *Username* yang login.
- *Path* tujuan di *client*.
- *Password* yang di-*input*-kan.

Pada awalnya, *password* yang digunakan untuk mengambil *file* dari *server*. Akan menggunakan *password* yang user masukkan saat login. Namun, dari hasil penelitian, didapat bahwa variabel *password* yang beredar atau yang digunakan dalam *file* login.c merupakan *password* yang telah di-enkripsi. Dan dari *library* yang lain dan *file cource code* yang lain dalam paket ini, didapat hasil bahwa variabel yang berisi *password* yang belum terenkripsi tidak dapat diketemukan. Hal ini dapat disebabkan karena hal-hal berikut.

- Untuk keamanan, maka *password* yang dimasukan user oleh aplikasi login langsung di-enkripsi secara instan.
- Terdapat beberapa *library* dan hasil *compile* dari *cource code* ini tidak dapat menggantikan keberadaan yang telah di-*install* terlebih dahulu karen dapat mengganggu stabilitas sistem. Hal ini terbukti pada saat instalasi banyak *file* yang tidak dapat ditulis ulang.

Sehingga disini aplikasi didesain untuk meminta ulang *password* khusus untuk mengambil *file* dari *server*.

3.5.1.7 Pemberian hak akses

Pada sub bab 3.5.1.3 telah dijelaskan bahwa tiap *user* yang melakukan login *domain*, maka secara otomatis akan dibuatkan *folder* pribadi yang dapat digunakan oleh *user*. Karena yang melakukan pembuatan *folder* ini adalah dari sistem yang memiliki hak akses root, maka *folder* yang telah dibuat tadi tentunya juga adalah milik *user* root. Oleh karena itu hak akses ini harus dirubah menjadi milik *user* yang melakukan login *domain* tadi.

3.5.1.8 Pengecekan hak akses *user* dalam *domain*

Hak akses *user* pada login *domain* terdapat bermacam-macam (misal : administrator, *power user*, *guest*, dsb). Hak akses *user* pada *domain* ini dapat diintegrasikan dengan hak akses pada komputer *client*. Dengan begini tiap-tiap *user* yang melakukan login *domain* pada komputer *client* memiliki hak akses sesuai dengan konfigurasi *user* pada komputer *server domain*.

Untuk menentukkan hak akses ini, maka login *domain* pada *client* akan dikelompokkan pada *group-group* yang telah ada. Karena *username* antara *server* (Windows 2003) dan *client* (linux) berbeda, maka perlu dibuatkan *group* yang tersendiri yang lain dari *group* yang telah ada pada komputer *client* tersebut. Dan fungsi ini akan memasukkan user kedalam grup-grup yang bersesuaian antara di *server* dan di *client*.

Dan masing-masing grup tersebut memiliki hak akses yang berbeda-beda. Perbedaan hak akses ini menggunakan aplikasi sudo, dan perbedaannya terlihat dalam *file* sudoers. Contoh perbedaan dalam *file* sudoers dapat dilihat pada gambar 3.10.

```
%admindomain          (ALL)NOPASSWD:ALL
```

Gambar 3.10. Contoh potongan file sudoers

3.5.1.9 Proses terminal login setelah dilakukan perubahan

Dan dalam gambar 3.11 dapat dilihat alur dari aplikasi terminal login yang berubah karena mendapat beberapa fungsi tambahan.

Gambar 3.11. Diagram akhir proses terminal login

3.5.2 Perancangan Pembangunan Sistem Login *Remote*

Sistem login *remote* adalah login Linux yang dilakukan melalui jaringan dengan komputer lain yang bukan dirinya sendiri.

Untuk membangun sistem *login remote* maka perlu modifikasi pada source paket "**openssh**". Paket openssh ini menangani tentang *remote* login Linux dengan cara menjalankan suatu daemon yang bernama sshd yang selalu me-listen port 22 (pada konfigurasi standar) untuk dapat di-*remote* oleh *user* melalui workstation lain.

3.5.2.1 *File* dan paket yang dibutuhkan

File milik paket openssh yang didapat dari internet ada 2 yaitu:

- Openssh_3.8.1p1.orig.tar.gz → *File* kompresi yang berisi *cource code* dari paket openssh.
- Openssh_3.8.1p1-31sarge5.diff.gz → *File* kompresi yang berisi patch dari paket openssh.

Paket-paket tambahan yang dibutuhkan untuk dapat meng-*compile* dan meng-*install* openssh antara lain:

- Libpam-dev
- Zlib1g-dev
- Libssl-dev

3.5.2.2 Keterangan isi paket

Paket openssh-3.8.1p1 berisi dari *cource code* dari aplikasi-aplikasi yang dimiliki oleh openssh. Termasuk diantaranya aplikasi ssh, sshd *server*, otentikasi *remote*, pembuatan sesi, dan lain-lain. Dan total isi paket shadow-4.0.3 yang telah di-*compile* ulang berisikan 33 MB data yang terdiri dari 631 *file* dan 13 folder.

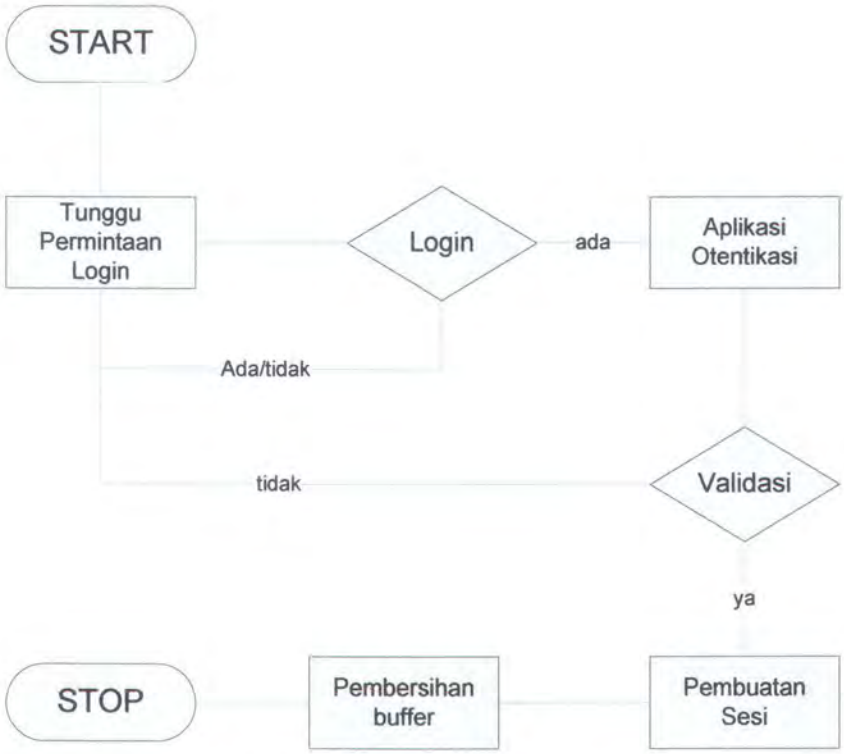
3.5.2.3 Proses aplikasi *remote* login tanpa perubahan

Selain cara pemakaian aplikasi login yang berbeda dari terminal login. Dari prosesnya *remote* login yang menggunakan paket ssh ini memiliki alur proses yang berbeda jauh.

Apabila aplikasi login dalam *terminal* login dijalankan hanya pada saat ada *user* yang melakukan login. Aplikasi *remote* login ini berjalan setiap saat karena berjalan sebagai *daemon*. *Daemon remote* login ini adalah sshd *server*, yang terus berjalan menunggu adanya permintaan *remote* login.

Dan saat terdapat permintaan *remote* login. Maka sshd *server* akan mengaktifkan aplikasi lain yang akan menangani permintaan ini dan kembali menunggu permintaan *remote* login yang lain. Sshd *server* akan memanggil aplikasi otentikasi memeriksa vailiditas dari permintaan login tersebut.

Dan saat permintaan login itu valid. Maka dipanggil aplikasi yang akan membuat sesi untuk user tersebut yang melakukan *remote* login. Dan membersihkan sisa-sisa variabel dan fungsi yang dijalankan pada saat dilakukan otentikasi. Gambaran dari proses-proses tadi akan dijelaskan dalam gambar 3.12.



Gambar 3.12. Diagram proses aplikasi remote login

Aplikasi otentikasi dari openssh memanggil aplikasi dari *file* auth-passwd.c, auth1.c, auth2.c, auth-pam.c, dan beberapa aplikasi otentikasi lain yang mendukung kesemua aplikasi otentikasi diatas dijalankan secara sekuensial untuk mecari asal user yang login. Apabila user tersebut menggunakan *account domain*. Maka yang memberikan hasil positif adalah auth-pam.

Kemudian fungsi pembuatan sesi adalah aplikasi yang menyediakan sesi bagi para user yang login dan mengarahkan user tersebut menuju *home direktori*-nya. Untuk fungsi pembuatan sesi akan dijelaskan lebih lanjut dalam fungsi-fungsi tambahan berikutnya.

3.5.2.4 Pengecekan *user*

Seluruh fungsi tambahan dalam aplikasi login ini ditempatkan pada saat awal pembuatan *session*. Hal ini dikarenakan proses validasi dan pembuatan sesi dalam openssh ditempatkan dalam dua aplikasi yang berbeda. Sehingga untuk mencegah ketidakstabilan dan pertukaran variabel yang salah. Maka seluruh fungsi tambahan ditempatkan dalam satu *file* dan aplikasi yang sama.

Pembuatan *session* dalam paket openssh dilakukan dalam *file* sesion.c, dan awal pembuatan *session* dilakukan dalam fungsi do_authenticated. Jelasnya dari fungsi tersebut dapat dilihat dalam gambar 3.13.

```
do_authenticated(Authctxt *authctxt)
{
    setproctitle("%s", authctxt->pw->pw_name);

→ tempat ditambahkan fungsi tambahan
    /*
     * Cancel the alarm we set to limit the time taken for
     * authentication.
     */
    alarm(0);
    if (startup_pipe != -1) {
        close(startup_pipe);
        startup_pipe = -1;
    }
```

Gambar 3.13. Contoh isi file dari aplikasi pembuatan session remote

Dan tempat untuk ditambahkan fungsi telah ditandai dalam gambar 3.13 dengan adanya tanda panah.

Setelah *remote* login dapat dilakukan dan user telah diotentikasi. Maka akan dilakukan pengecekan *user* yang melakukan login, apakah login yang dilakukan oleh *user* tersebut adalah login pada komputer lokal atau login *domain*. Dan hasilnya akan dipergunakan oleh fungsi berikutnya.

3.5.2.5 Pembuatan direktori

Setiap user yang login membutuhkan *home* direktori dalam setiap komputer yang digunakan. Baik menggunakan *account* lokal ataupun *account domain*. Home direktori ini dibutuhkan sebagai tempat penyimpanan *profile* dan data milik *user*. Dan saat melakukan login, *user* akan langsung diarahkan ke dalam *home* direktorinya.

Seperti pada login *domain* pada umumnya, ketika *user* menggunakan *account domain* untuk login pada komputer *client*, maka secara otomatis akan dibuatkan *profile* sendiri. *Profile* ini dapat berupa konfigurasi maupun *folder* sesuai yang berada pada *domain*. Karena proses login antara *client* (linux) dan *server domain* (Windows 2003) memiliki perbedaan sehingga mengharuskan dilakukan konfigurasi manual secara berbeda, maka pembuatan *profile* atau direktori inipun harus dikonfigurasi secara manual.

Pada komputer yang ber-OS linux, secara umum *profile* atau *folder* pribadi tiap *user* diletakkan pada direktori `/home/*` (* berarti adalah nama *user* tersebut). Untuk itu pada aplikasi yang dibangun ini *folder user* yang melakukan login *domain* akan dibuatkan *folder* sendiri pada *folder user* linux pada umumnya.

Dimana dalam kasus ini home direktori para *user domain* berada di folder `/home/TA-FANNI/*`. Maka terlebih dahulu dibuat folder `/home/TA-FANNI` saat penggabungan *domain*. Dan proses ini akan

membuat folder khusus bagi para user dalam folder yang telah dibuat sebelumnya.

3.5.2.6 Pengambilan *file* dari *server*

Setelah komunikasi antara linux dan Windows dapat dilakukan, yaitu menggunakan aplikasi samba pada linux, maka beberapa *file* konfigurasi pada *server domain* (Windows 2003) dapat diambil. *File* konfigurasi tersebut berisi informasi antara lain :

1. Nama *server* (IP).
2. *Username* yang *login*.
3. *Path* tujuan di *client*.
4. *Password* yang di-*input*-kan.

Pada awalnya, *password* yang digunakan untuk mengambil *file* dari *server*. Akan menggunakan *password* yang user masukkan saat login. Namun, dari hasil penelitian, didapat bahwa variabel *password* yang beredar atau yang digunakan dalam *file* login.c merupakan *password* yang telah di-enkripsi. Untuk mencari *password* yang belum di-enkripsi. Maka dilakukan penelusuran dalam *file-file* yang digunakan untuk autentikasi. Antara lain: auth1.c, auth2.c, auth-pam.c, auth-passwd, dan lain-lain. Namun dari hasil penelitian, *password* yang tidak di-enkripsi yang ditemukan hanyalah *password* milik *account* lokal *client*. *Password* yang tidak di-enkripsi milik *account domain* tidak ditemukan, sehingga fungsi pengambilan *file* dari *server* tidak dapat dijalankan secara otomatis. Hal ini dikarenakan oleh hal-hal berikut ini:

- Dalam menggunakan aplikasi openssh. Tidak dapat diatur untuk dapat meminta *password* untuk kedua kalinya.
- Saat meminta ulang *password*, aplikasi yang dibuat tidak menjalankan perintah tersebut. Sehingga *password* tidak didapatkan.
- Terdapat beberapa *library* dan hasil *compile* dari *cource code* ini tidak dapat menggantikan keberadaan yang telah di-*install* terlebih dahulu karena dapat mengganggu stablitas sistem. Hal ini terbukti pada saat instalasi banyak *file* yang tidak dapat ditulis ulang.

3.5.2.7 Pemberian hak akses

Pada sub bab 3.5.2.5 telah dijelaskan bahwa tiap *user* yang melakukan login *domain*, maka secara otomatis akan dibuatkan *folder* pribadi yang dapat digunakan oleh *user*. Karena yang melakukan pembuatan *folder* ini adalah dari *sistem* yang memiliki hak akses root, maka *folder* yang telah dibuat tadi tentunya juga adalah milik *user* root. Oleh karena itu hak akses ini harus dirubah menjadi milik *user* yang melakukan login *domain* tadi.

3.5.2.8 Pengecekan hak akses *user* dalam *domain*

Hak akses *user* pada login *domain* terdapat bermacam-macam (misal : administrator, *power user*, *guest*, dsb). Hak akses *user* pada *domain* ini dapat diintegrasikan dengan hak akses pada komputer *client*. Sehingga tiap-tiap *user* yang melakukan login *domain* pada komputer *client* memiliki hak akses sesuai dengan konfigurasi *user* pada komputer *server domain*.

Untuk menentukan hak akses ini, maka login *domain* pada *client* akan dikelompokkan pada *group-group* yang telah ada. Karena *username* antara *server* (Windows 2003) dan *client* (linux) berbeda, maka perlu dibuatkan *group* yang tersendiri yang lain dari *group* yang telah ada pada komputer *client* tersebut. Dan fungsi ini akan memasukkan user kedalam grup-grup yang bersesuaian antara di *server* dan di *client*.

Dan masing-masing grup tersebut memiliki hak akses yang berbeda-beda. Perbedaan hak akses ini menggunakan aplikasi sudo, dan perbedaannya terlihat dalam *file* sudoers. Contoh perbedaan dalam *file* sudoers dapat dilihat pada gambar 3.14. Dalam contoh tersebut terdapat grup admindomain yang merupakan adaptasi dari grup "admin domain" milik AD yang memiliki hak akses keseluruhan terhadap *client* tanpa perlu memasukkan ulang *password*.

```
%admindomain          (ALL)NOPASSWD:ALL
```

Gambar 3.14. Contoh potongan file sudoers

3.5.2.9 Proses dalam *remote* login setelah dilakukan perubahan

Dalam gambar 3.15 dapat dilihat alur dari aplikasi *remote* login yang berubah karena mendapat beberapa fungsi tambahan yang telah dijelaskan sebelumnya.

BAB 4

IMPLEMENTASI PERANGKAT LUNAK

BAB 4

IMPLEMENTASI PERANGKAT LUNAK

Bab ini menguraikan tentang implementasi keseluruhan dari rancangan perangkat lunak yang telah diuraikan pada bab III. Pembahasannya meliputi implementasi sistem *domain*, implementasi koneksi Linux Dengan Windows, implementasi sistem otentikasi, dan implementasi aplikasi otentikasi.

4.1. IMPLEMENTASI SISTEM *DOMAIN*

Pada sub bab ini akan dijelaskan mengenai implementasi koneksi proses login pada komputer *client* yang menggunakan Linux dengan login pada komputer *server domain* yang menggunakan Windows. Dimulai dari aplikasi dari sisi *server* yaitu Active Directory. Kemudian dari sisi *client*, yaitu samba, kerberos, NSS, dan PAM. Dari sisi *client*, komputer yang digunakan dalam implementasi ini bernama yudhistira.

Dari sisi *server* instalasi dan konfigurasi dilakukan oleh *user* dengan login administrator, dan dari sisi *client* instalasi dan konfigurasi dilakukan oleh *user* dengan login root. Namun di sisi *client*, instalasi dan konfigurasi juga dapat dilakukan *user* dengan login selain root, namun harus menggunakan fasilitas sudo. Dan *user* tersebut harus terdaftar dalam daftar sudoers.

4.1.1 Konfigurasi *User* dan *Home Directory*

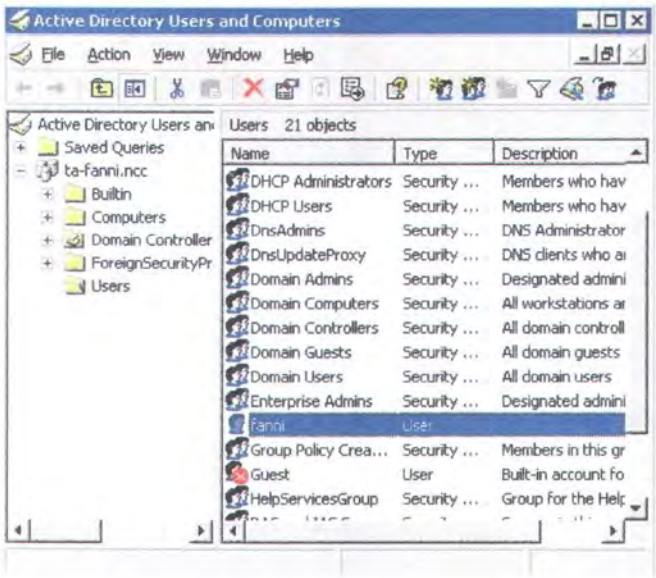
Berikut tidak membahas mengenai instalasi dan konfigurasi AD dalam *server domain*, karena hal tersebut merupakan hal yang sangat umum dalam implementasi suatu *domain* yang menggunakan Windows *Server* 2003. Pembahasan tentang implementasi tugas akhir ini mulai dari instalasi AD di *server domain*. Dengan nama komputer yang akan menjadi PDC (*Primary Domain Controller*) adalah quaestor dan nama *domain* yang dibuat adalah ta-fanni.ncc.

Setelah AD di-*instal*, maka komputer yang di-*instal* menjadi PDC dari *domain* yang digunakan dalam pengerjaan TA ini. Namun, hak akses *user* dalam *domain* tersebut masih perlu diatur lebih lanjut. Hal ini berhubungan dengan *home directory user* yang diakses pada saat *user* login pada komputer *client*. Berikut dijelaskan tentang konfigurasi *user*, konfigurasi pengaturan akses *home directory* (konfigurasi *folder redirection*) dan konfigurasi akses PDC pada saat *user* login di komputer *client*.

4.1.1.1 Konfigurasi *User*

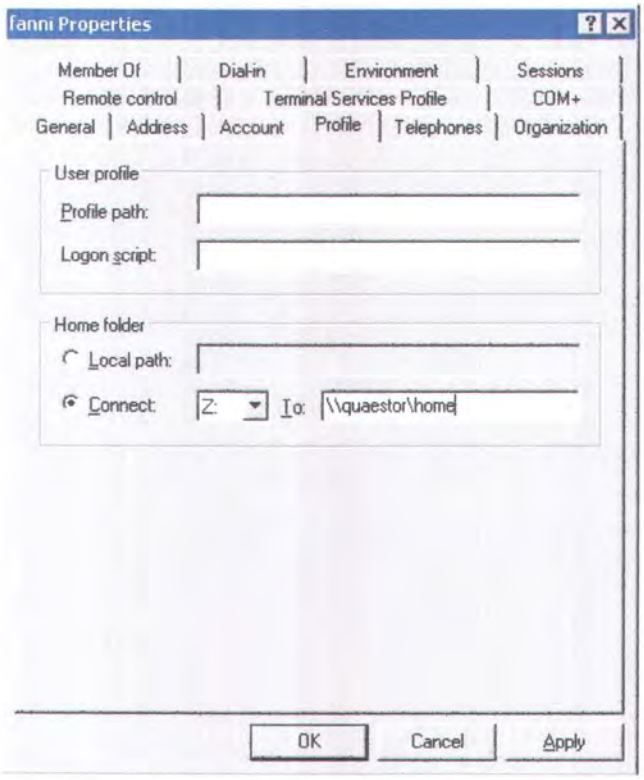
Pembuatan *user* dalam suatu *domain* tidak dibahas di dalam buku ini karena merupakan hal yang umum dalam administrasi *domain*. Yang akan dijelaskan berikut ini adalah konfigurasi *user* untuk meletakkan *folder* pribadi mereka yang ada di dalam *server* sebagai salah satu drive dalam komputer *client* yang digunakan. *Step by step* konfigurasi *user* yaitu :

1. Untuk dapat melihat daftar *user* yang ada, pilih Administartive Tools → Active Directory *User*s and Computers. Yang akan memunculkan *window* seperti dalam gambar 4.1



Gambar 4.1. Contoh window Active Directory User

2. Kemudian cari daftar *user* dalam Active Directory *User*s and Computers → ta-fanni.ncc → *User*s. Dan klik kanan *user* yang akan di konfigure. Dan pilih properties. Dalam hal ini yang dipilih adalah *user* fanni. Kemudian akan muncul window seperti berikut pada gambar 4.2, dan pilih tap Pro*file*.



Gambar 4.2. Contoh window User Properties

3. Dalam bagian "*Home folder*" pilih Connect, kemudian isi *drive*-nya sesuai kebutuhan dan pada *textbox* setelahnya isi dengan *path* yang menyimpan data para *user* dalam. Dalam kasus ini quaestor adalah nama komputer yang menjadi *server* data di *domain* ta-fanni.ncc kemudian *home* adalah nama share *folder* dalam *server* yang berisi data *user*. Sehingga isinya adalah //quaestor/*home*.

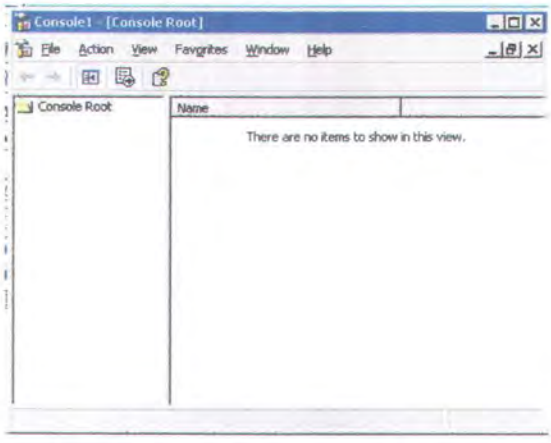
4.1.1.2 Konfigurasi Folder Redirection

Folder redirection adalah pengaturan untuk membuat *folder-folder* umum milik *user* dalam *domain* menuju ke *folder* apa. Dalam kasus ini *folder* yang dimaksud antara lain :

1. Application Data

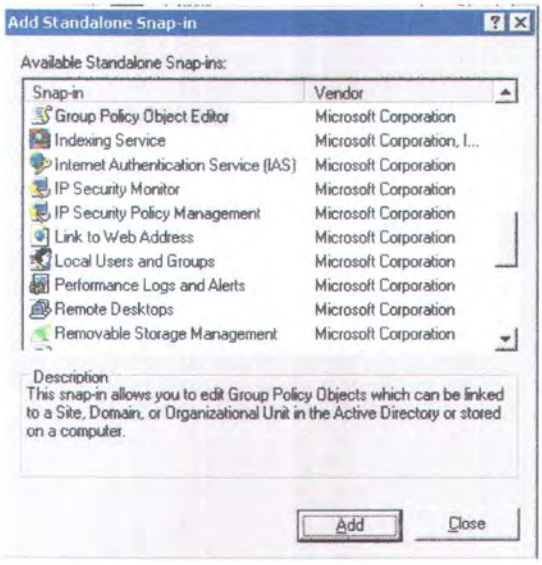
2. Desktop
3. My Documents
4. Start Menu

Untuk dapat mengatur *folder* redirection, dapat dibuka "management console" dengan cara mengetikkan "mmc" di run. Kemudian akan muncul window "console" (gambar 4.3). Kemudian di tap "*file*" pilih "add/remove snap-in". Untuk memunculkan window "snap-in" (gambar 4.4). Kemudian klik tombol "add". Tombol "add" ini akan memunculkan window "add standalone snap-in" (gambar 4.5). Kemudian pilih "Group Policy Object Editor" dan klik "add" dan akan muncul window "Select Group Policy Editor" (gambar 4.6) dan klik tombol "browse". Dan akan muncul window "Browse for a Group Policy Object" (gambar 4.7), pilih "Default *Domain* Policy" dan pilih "Default *Domain* Policy" dan klik OK. Dan kembali ke window sebelumnya klik finish, close dan close lagi. Urutan tampilan window dapat dilihat dibawah ini.

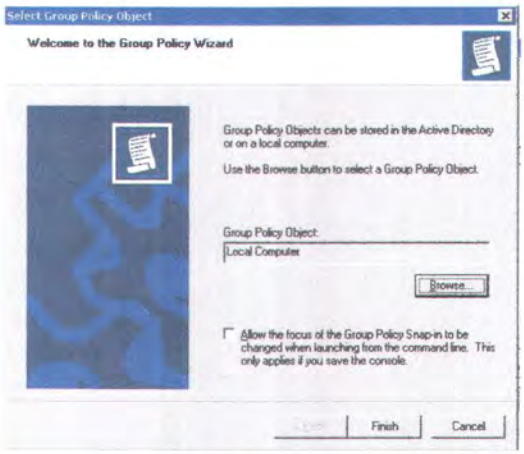


Gambar 4.3. Contoh window Console Management

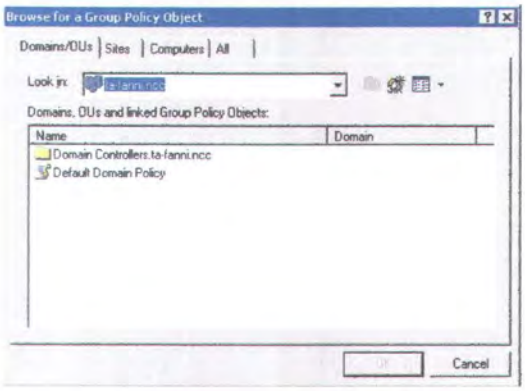
Gambar 4.4. Contoh window Add/Remove Snap-in



Gambar 4.5. Contoh windowaAdd standalone Snap-in



Gambar 4.6. Contoh window Select Group Policy Object

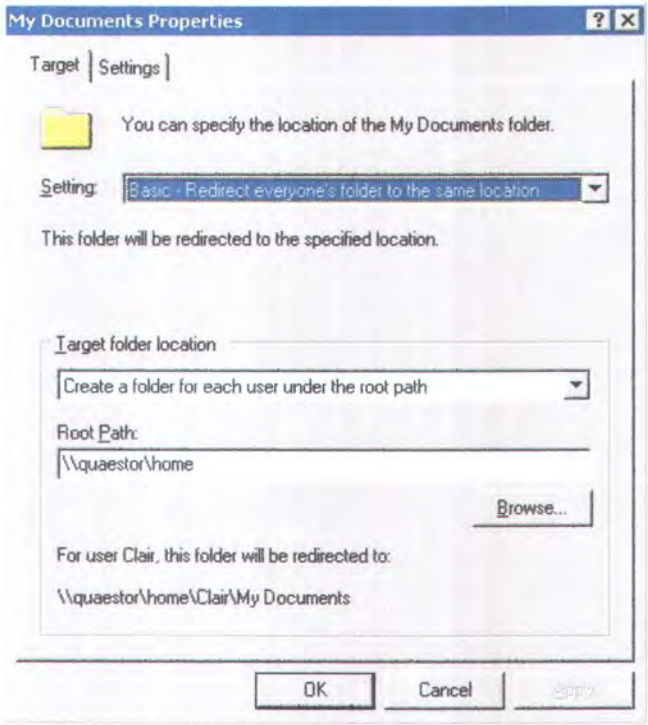


Gambar 4.7. Contoh window Browse Group Policy Object

Kemudian pilih Console Root → Default *Domain* Policy → *User* Configuration → Windows Setting → My Document sehingga menampilkan tree yang berbentuk seperti pada gambar 4.8.

Gambar 4.8 Contoh Tree untuk Folder Redirection

Klik kanan di "My Document" dan pilih properties dan akan muncul window seperti pada gambar 4.9. Pada bagian "Setting" pilih "Basic – Redirect everyone *folder* to same location". Kemudian pada bagian "Target *Folder* Redirection" pilih "Create a *folder* for each *user* under the root path" dan isikan "Root Path" dengan tempat My Document berada sesuai pengaturan *user* sebelumnya yakni "//quaestor/*home*"

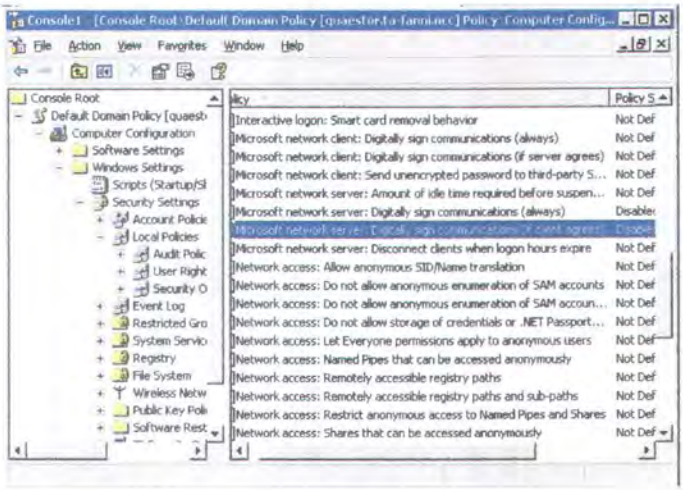


Gambar 4.9 Contoh windowMy DocumentProperties

4.1.1.3 Konfigurasi akses PDC

Akses terhadap PDC dari *client* yang menggunakan OS Linux membutuhkan konfigurasi lebih daripada *client* yang menggunakan OS Windows. Yaitu dengan cara mengizinkan akses smb.

Hal ini dapat diatur melalui window yang sama saat mengatur *folder* redirection namun berbeda tree. Arahkan tree menuju "Computer Configuration → Windows Settings → Security Settings → Local Policies → Security Options". Kemudian cari bagian "Microsoft network *server*: Digitally sign communications (always)" dan "Microsoft network *server*: Digitally sign communications (if *client* agrees)". Dan ubah kedua bagian tersebut menjadi disabled. Dimana hasilnya seperti terlihat dalam gambar 4.10.



Gambar 4.10. Contoh gambarConsole Security Options

4.2. IMPLEMENTASI KONEKSI LINUX DENGAN WINDOWS

Pada bagian ini akan dijelaskan implementasi dari konektivitas antara Linux dengan Windows yang ditangani oleh aplikasi Samba dan Kerberos.

4.2.1 Instalasi dan Konfigurasi Samba dan Winbind

Samba yang berfungsi sebagai media penghubung antara *client* yang menggunakan OS Linux Debian dengan sever yang menggunakan OS Windows *Server* 2003, merupakan aplikasi tambahan milik Linux yang tidak terdapat dalam aplikasi awal Linux. Sehingga harus diinstalasi terlebih dahulu dalam *client*. Untuk memulai tahap ini paket-paket yang perlu di-*install* antara lain:

- Samba
- Samba-*client*
- Smbfs
- Smb*client*
- winbind

Setelah semua paket diatas ter-*install* kita memasuki tahapan berikutnya, yaitu konfigurasi samba untuk bergabung dalam *domain* yang telah dibuat sebelumnya. Konfigurasi dapat dilakukan melalui perubahan pada *file* /etc/samba/smb.conf . Dan tambahkan beberapa baris konfigurasi seperti yang terlihat pada gambar 4.11.

```
workgroup = TA-FANNI
security = ads
password server = *
realm = TA-FANNI.NCC
pam password change = yes
idmap uid = 10000-20000
idmap gid = 10000-20000
template shell = /bin/bash
template homedir = /home/%D/%U
winbind use default domain = yes
```

Gambar 4.11. Gambar Potongan konfigurasi smb.conf

Setelah perubahan dilakukan. *Restart* samba dan winbind dengan perintah sebagai berikut seperti terlihat dalam gambar 4.12.

```
/etc/init.d/samba restart
/etc/init.d/winbind restart
```

Gambar 4.12. Contoh Perintah restart samba dan winbind

Kemudian periksa kembali apakah samba dan winbind telah berjalan seperti yang terlihat pada gambar 4.13.

```
root        3502  0.0  0.3   5604  1968 ?    Ss    Jan12    0:03
/usr/sbin/nmbd -D
root        3504  0.0  0.6   8888  3592 ?    Ss    Jan12    0:00
/usr/sbin/smbd -D
root        3533  0.0  0.7   8160  3788 ?    Rs    Jan12    0:00
/usr/sbin/winbindd
root        3535  0.0  0.6   7776  3440 ?    S     Jan12    0:00
/usr/sbin/winbindd
```

Gambar 4.13. Contoh proses samba dan winbind

4.2.2 Konfigurasi Kerberos

Mengingat disini kita menggunakan Windows *Server* 2003 sebagai sistem operasi *domain server*. Maka dibutuhkan kerberos untuk menyesuaikan *transfer* antar paket. Hal ini dikarenakan AD yang dimiliki Windows *Server* 2003 menggunakan kerberos untuk keamanan jaringannya. Sehingga *client* pun harus meiliki kerberos *client* untuk menerima informasi dari *server*. Paket yang dibutuhkan untuk implementasi ini adalah krb5-kdc.

File konfigurasi yang dimiliki oleh kerberos ini ada 2 antara lain: krb5.conf dan kdc.conf. *File* krb5.conf adalah konfigurasi untuk mengatur *domain* apa saja yang tersedia dan kdc.conf adalah konfigurasi untuk mengatur dimanakah PDC berada.

Dalam kasus ini konfigurasi dilakukan dengan menambahkan konfigurasi seperti dalam gambar 4.14 dan 4.15 dalam *file* krb5.conf sesuai pada tempatnya.

```
TA-FANNI.NCC = {
     kdc = 10.126.13.55
    admin_server = 10.126.13.55
    default_domain = TA-FANNI
}
```

Gambar 4.14. Contoh penambahan domain yang tersedia

```
.ta-fanni.ncc = TA-FANNI.NCC
ta-fanni.ncc = TA-FANNI.NCC
```

Gambar 4.15. Contoh pengaturan untuk penamaan domain

4.3. IMPLEMENTASI SISTEM OTENTIKASI

Sistem otentikasi milik Linux pun terbagi dalam 2 bagian. Yaitu sistem otentikasi dasar dan sistem otentikasi aplikasi. Berikut akan dijelaskan lebih lanjut tentang keduanya dan perancangan dalam kedua sistem tersebut.

4.3.1 Implementasi Sistem Otentikasi Dasar

Otentikasi ini menggunakan NSS sebagai skema untuk mencari data *user* dari *name server* yang tersedia. NSS mengarahkan ke *name server* mana Linux mencari data *user*. Sehingga dalam kasus ini NSS akan mengarahkan Linux menuju *name service* milik sendiri kemudian memeriksa *name* service AD yang menggunakan koneksi winbind.

Berikut akan dijelaskan tentang otentikasi dasar milik Linux yang diatur oleh konfigurasi oleh NSS.

4.3.1.1 Konfigurasi NSS

Seperti yang kita telah disebutkan dalam bab-bab sebelumnya, maka perlu dilakukan perubahan pada nsswitch agar aplikasi ini juga mengecek winbind. Perubahan dilakukan dalam *file* /etc/nsswitch.conf dan lakukan perubahan hingga terdapat potongan seperti pada gambar 4.16.

```
passwd:         compat winbind
group:          compat winbind
```

Gambar 4.16. Contoh potongan file konfigurasi NSS

4.3.2 Implementasi Sistem Otentikasi Aplikasi

Dalam Linux, aplikasi dijalankan menggunakan PAM sebagai media otentikasi. Modul-modul dalam PAM dipanggil dalam aplikasi tersebut untuk melakukan otentikasi apabila aplikasi tersebut membutuhkan otentikasi.

4.3.2.1 Konfigurasi PAM

Setelah sampai pada bagian ini berarti *user* dan group yang ada dalam *domain server* telah dianggap sebagai *user* dan group dalam *client*. Berikutnya adalah bagaimana membuat aplikasi yang dijalankan oleh

client memeriksa atau meng-otentikasi *user* yang menjalankan aplikasi. Dalam kasus ini dimana aplikasi yang difokuskan adalah login, ssh, dan gdm. Hal ini merupakan tugas dari PAM (Plugable Authentication Module).

PAM merupakan bagian dari spesifikasi minimum dari Sistem Operasi Linux, sehingga tidak perlu dilakukan instalasi tambahan. Sehingga kita hanya perlu melakukan konfigurasi yang dapat dilakukan dengan mengubah isi *file* dalam *folder* `/etc/pam.d`

4.3.2.2 /etc/pam.d/common-auth

Konfigurasi otentikasi umum yang digunakan oleh seluruh servide PAM.

Tambahkan dalam *file* tersebut perintah dibawah ini pada awal baris *file* tersebut.

```
auth sufficient      pam_winbind.so
```

Gambar 4.17.Contoh potongan file /etc/pam.d/common-auth

Dari hasil konfigurasi tersebut berarti PAM selain menggunakan otentikasi dengan sistem unix yang berarti utentikasi linux yang menggunakan *user* yang ada dalam console itu saja, tapi juga menggunakan sistem winbindd, dimana winbindd sudah dihubungkan dengan *domain*.

4.3.2.3 /etc/pam.d/common-*account*

Konfigurasi *account* umum yang digunakan oleh seluruh servide PAM.

Tambahkan dalam *file* tersebut perintah dibawah ini pada awal baris *file* tersebut.

```
account sufficient  pam_winbind.so
```

Gambar 4.18. Contoh potongan file /etc/pam.d/common-account

Dari hasil konfigurasi tersebut berarti PAM selain menggunakan otentikasi dengan sistem unix yang berarti utentikasi linux yang menggunakan *user* yang ada dalam console itu saja, tapi juga menggunakan sistem winbindd, dimana winbindd sudah dihubungkan dengan *domain*.

4.3.2.4 /etc/pam.d/common-*session*

Konfigurasi *session* umum yang digunakan oleh seluruh service PAM. Dalam kasus ini tidak perlu ditambahkan konfigurasi tambahan. Karena otentikasi yang digunakan dalam pengerjaan TA ini tidak membutuhkan *session* khusus dari PAM.

4.4. Implementasi Aplikasi Otentifikasi

Sampai pada bagian ini, komunikasi antara *server domain* dan *client* sudah dapat dilakukan. Akan tetapi agar sistem *domain* atau proses login dapat berjalan seperti pada umumnya, perlu ada konfigurasi lebih lanjut. Konfigurasi secara detail akan dijelaskan pada sub bab berikut di bawah ini.

4.4.1 Pembangunan Aplikasi *Terminal login*

Untuk membangun Aplikasi *terminal login* maka perlu modifikasi pada source paket shadow. Paket ini menangani tentang *terminal login* Linux

4.4.1.1 Pengecekan *user*

Tujuan modifikasi ini adalah untuk mengetahui apakah *user* yang login merupakan *user client* atau *user domain*. Hal ini dibutuhkan karena *home folder* yang belum ada hanyalah milik *user domain*.

Untuk dapat melakukan hal ini harus ditemukan variabel yang berisi nama *user* yang login di *client* tersebut menggunakan fasilitas terminal login. Setelah nama *user* ditemukan, maka akan dibandingkan dengan *file* `/etc/passwd` untuk mencari apakah *user* terdaftar atau tidak. Apabila tidak maka pada tahap berikutnya perlu dibuatkan direktori untuk *user* tersebut apabila ia belum memiliki *home* direktori di *client* tersebut.

Fungsi ini diletakkan di setelah aplikasi login meminta *username*. Fungsi permintaan username dalam *file* login.c dapat dilihat dalam gambar 4.19.

Dan pseudocode dari implementasi ini dapat dilihat dalam gambar 4.20

```
Cek_lokal (username)

    A = open_file(passwd_file)
    WHILE s != EOF
        B = read_file(A)
        IF B = username
            Return 0
        ENDIF
    ENDWHILE
    RETURN username
EXIT
```

Gambar 4.19. Gambar pseudocode pengecekan user

4.4.1.2 Pembuatan direktori

Setelah diketahui bahwa yang *user* yang login merupakan *user domain*, maka perlu dibuatkan direktori yang akan menjadi *home* direktori *user*. Misal dalam kasus ini, *user domain* yang login adalah tester.

Kemudian dalam *client* dibuatkan direktori yang sesuai dengan nama */home*/TA-FANNI/tester. Setelah direktori tersebut dibuat maka perlu dilakukan pengambilan *file-file* dari *server* yang berisi data *user* dan juga untuk penyerahan hak akses terhadap direktori tersebut dengan login. Jelasnya hal tersebut dapat dilihat dalam gambar 4.21.

```
Bikin_dir (username)

    A = Set_addr(username)
    Create_dir(A)
    RETURN 0
EXIT
```

Gambar 4.20. Gambar contoh pseudocode pembuatan direktori

4.4.1.3 Pengambilan *file* dari *server*

Berdasarkan informasi berikut, maka kita dapat melakukan pengambilan *file* dari *server*:

- Nama *server* (IP)
- *Username* yang login
- Path tujuan di *client*
- *Password* yang diinputkan

Dan dengan informasi diatas, maka dapat dilakukan pengambilan *file* dari *server* menggunakan fasilitas samba.

4.4.1.4 Pemberian hak akses

Setelah direktori untuk *user* telah dibuat dan *file-file* milik *user* telah diambil dari *server*, maka *user*t perlu diberikan akses untuk dapat mengakses *folder* ini karena perintah sebelumnya dilakukan oleh *user* root.

Dengan memiliki informasi *username* dan path *folder* direktori *user*, maka dapat dilakukan pemberian hak akses ini.

4.4.1.5 Pengecekan hak akses *user* dalam *domain* dan implementasi dalam *client*

Hak yang dimiliki oleh *user* dalam *domain* bermacam-macam. Dan hal ini berpengaruh terhadap hak-hak yang dimiliki *user* dalam *client*. Dalam sudo dapat ditentukan hak *user domain* yang tersedia apa saja. Dan *user* yang memiliki hak akses tertentu tergabung dalam grup yang sama. Sehingga akses *user* dapat ditentukan berdasarkan grup *user* tersebut. Dan permasalah yang muncul adalah, linux tidak dapat menerima informasi nama atau grup yang mengandung unsur spasi. Sehingga perlu dibuat grup baru untuk menampung hak akses para *user* ini.

Dalam pengerjaan TA ini hak *user* yang ada adalah "*domain* admins". Yaitu dapat melakukan apa saja. Implementasi dalm sudo terlihat dalam *file* sudoers seperti berikut ini:

Dan implementasi penggabungan grup, psedeucode terlihat dalam gambar 4.21.

```
Hak_domain (username)

    A = open_file(passwd_file)
    WHILE s != EOF
        B = read_file(A)
        IF B = "domain admins"
            Addgroup(username, group_admin);
        ENDIF
    ENDWHILE
    RETURN username
EXIT
```

Gambar 4.21. Gambar contoh pseudocode penyesuaian hak

4.4.1.6 Implementasi Menggunakan Modul PAM

Namun dalam proses pengerjaan TA ini, ditemukan suatu metode yang lebih efektif dan lebih mudah untuk diaplikasikan untuk menangani

masalah pembuatan *home* direktori dan pengambilan *file* dari *server*. Sehingga implementasi bagian 4.4.1.2 sampai 4.4.1.6 dapat dilewati.

Metode ini menggunakan modul PAM untuk menjalankan sistem tersebut. Modul yang digunakan antara lain:

1. Pam_mkhomedir → Modul PAM untuk membuat *home* direktori
2. Pam_mount → Modul PAM untuk mengambil *file* dari *server* (*mount*)

Kedua modul tersebut dimasukkan ke dalam *file* konfigurasi PAM untuk aplikasi login. Contoh penggunaan kedua modul tersebut dapat dilihat pada gambar 4.22.

```
auth    sufficient      pam_mount.so
session sufficient      pam_mount.so
session required        pam_mkhomedir.so skel=/etc/ skel/ %
umask=0077
```

Gambar 4.22. Contoh gambar pemakain modul PAM

Karena dalam metode ini menggunakan modul pam_mount. Maka harus modul pam_mount harus di-instalasi lebih dahulu karena modul ini tidak terdapat dalam paket standar PAM. Nama modul yang perlu di-*install* adalah libpam-mount.

Setelah modul pam_mount ini ter-*install*. Perlu dilakukan konfigurasi pam_mount untuk dapat mengambil *file-file* dari *server* sesuai spesifikasi seperti dalam bab 4.4.1.4. *File* konfigurasi terdapat di /etc/security/pam_mount.so. Dan baris konfigurasi yang perlu ditambahkan dapat dilihat dalam gambar 4.23.

```
volume * smb 10.126.13.55 home /home/TA-FANNI/& \
uid=&,gid=&,dmask=0750,workgroup=TA-FANNI - -
```

Gambar 4.23. Contoh konfigurasi tambahan /etc/security/pam_mount.conf

4.4.2 Pembangunan Sistem *Remote* Login

Untuk membangun sistem *login remote* maka perlu modifikasi pada source paket "openssh". Paket openssh ini menangani tentang *remote* login Linux dengan cara menjalankan suatu daemon yang bernama sshd yang selalu me-listen port 22 (pada konfigurasi standar) untuk dapat di-*remote* oleh *user* melalui workstation lain.

4.4.2.1 Pengecekan *user*

Tujuan modifikasi ini adalah untuk mengetahui apakah *user* yang login merupakan *user client* atau *user domain*. Hal ini dibutuhkan karena *home folder* yang belum ada hanyalah milik *user domain*.

Untuk dapat melakukan hal ini harus ditemukan variabel yang berisi nama *user* yang login di *client* tersebut menggunakan fasilitas login ssh. Setelah nama *user* diketemukan, maka akan dibandingkan dengan *file* /etc/passwd untuk mencari apakah *user* terdaftar atau tidak. Apabila tidak maka pada tahap berikutnya perlu dibuatkan direktori untuk *user* tersebut apabila ia belum memiliki *home* direktori di *client* tersebut.

4.4.2.2 Pembuatan direktori

Setelah diketahui bahwa yang *user* yang login merupakan *user domain*, maka perlu dibuatkan direktori yang akan menjadi *home* direktori *user*. Misal dalam kasus ini, *user domain* yang login adalah tester. Kemudian dalam *client* dibuatkan direktori yang sesuai dengan nama `/home/TA-FANNI/tester`. Setelah direktori tersebut dibuat maka perlu dilakukan pengambilan *file-file* dari *server* yang berisi data *user* dan

juga untuk penyerahan hak akses terhadap direktori tersebut dengan login. Pseudocode dari aplikasi sama seperti dalam bab 4.4.1.4.

4.4.2.3 Pengambilan *file* dari *server*

Berbeda dengan terminal login, aplikasi *remote* login tidak dapat melakukan pengambilan *file* dari *server*. Hal ini dikarenakan karena dalam pengambilan *file* dari *server* membutuhkan *password* yang dimasukkan oleh user. Dan string yang dalam pengerjaan TA ini, belum dapat diketemukan string yang menyimpan *password* ini.

Selama proses pengerjaan, *string password* hanya berhasil ditemukan apabila sistem otentikasi yang digunakan merupakan otentikasi dengan *name server* lokal. Sedangkan yang digunakan untuk otentikasi *domain account* menggunakan PAM untuk otentikasi, sehingga *string password* berada dalam *library* PAM yang tidak ikut dalam pengerjaan TA ini.

4.4.2.4 Pemberian hak akses

Setelah direktori untuk *user* telah dibuat dan *file-file* milik *user* telah diambil dari *server*, maka *user*t perlu diberikan akses untuk dapat mengakses *folder* ini. Karen perintah sebelumnya dilakukan oleh *user* root. Dengan memiliki informasi *username* dan path *folder* direktori *user*, maka dapat dilakukan pemberian hak akses ini.

4.4.2.5 Pengecekan hak akses *user* dalam *domain* dan implementasi dalam *client*

Hak yang dimiliki oleh *user* dalam *domain* bermacam-macam. Dan hal ini berpengaruh terhadap hak-hak yang dimiliki *user* dalam *client*. Dalam

sudo dapat ditentukan hak *user domain* yang tersedia apa saja. Dan *user* yang memiliki hak akses tertentu tergabung dalam grup yang sama. Sehingga akses *user* dapat ditentukan berdasarkan grup *user* tersebut. Dan permasalah yang muncul adalah, linux tidak dapat menerima informasi nama atau grup yang mengandung unsur spasi. Sehingga perlu dibuat grup baru untuk menampung hak akses para *user* ini.

Dalam pengerjaan TA ini hak *user* yang ada adalah "*domain* admins". Yaitu dapat melakukan apa saja. Implementasi dalm sudo terlihat dalam *file* sudoers seperti berikut ini:

Dan implementasi penggabungan grup, psedeucode sama seperti dalam bab 4.4.1.6.

4.4.2.6 Implementasi Menggunakan Modul PAM

Namun dalam proses pengerjaan TA ini, ditemukan suatu metode yang lebih efektif dan lebih mudah untuk diaplikasikan untuk menangani masalah pembuatan *home* direktori dan pengambilan *file* dari *server*. Sehingga implementasi bagian 4.4.1.2 sampai 4.4.1.6 dapat dilewati.

Metode ini menggunakan modul PAM untuk menjalankan sistem tersebut. Modul yang digunakan antara lain:

- Pam_mkhomedir → Modul PAM untuk membuat *home* direktori
- Pam_mount → Modul PAM untuk mengambil *file* dari *server* (*mount*)

Kedua modul tersebut dimasukkan ke dalam *file* konfigurasi PAM untuk aplikasi login. Contoh penggunaan kedua modul tersebut sama seperti yang telah ditunjukkan dalam bab4.4.1.7.

BAB 5

UJI COBA DAN EVALUASI

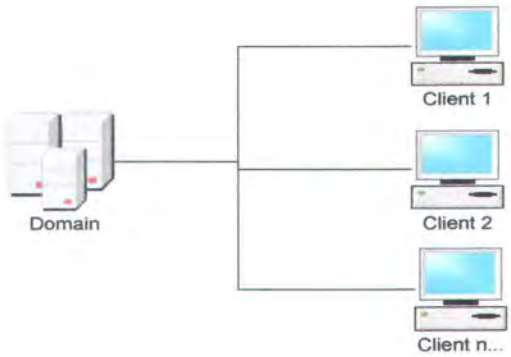
BAB 5

UJI COBA DAN EVALUASI

Pada bab ini dibahas mengenai uji coba terhadap aplikasi yang telah dibangun. Uji coba ini dilakukan dengan membangun sebuah *domain* baru yang berdiri sendiri yang terdiri dari sebuah *server domain* dengan sistem operasi Windows *Server* 2003 Standar Edition dan beberapa *client* dengan sistem operasi Linux Debian. Uji coba dan evaluasi ini meliputi lingkungan pelaksanaan uji coba, konfigurasi dan Instalasi Aplikasi pada *client*, pembangunan *domain* beserta anggotanya, dan pelaksanaan uji coba dalam keseluruhan sistem *domain*.

5.1. Lingkungan Pelaksanaan Uji Coba

Pada sub bab ini akan dijelaskan mengenai lingkungan uji coba aplikasi ini. Untuk melakukan uji cobanya, diperlukan minimal 3 komputer dengan 1 komputer sebaga *domain server* 1 komputer yang digunakan sebagai *client* untuk penelitian dan pengembangan, dan komputer yang lain sebagai *client* untuk implementasi dari aplikasi tersebut. Gambaran jelas tentang komputer yang dibutuhkan dapat dilihat pada gambar 5.1.



Gambar 5.1. Gambar kompute ryang dibutuhkan dalam uji coba

Untuk *detail* spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam pengujian ini dapat dilihat pada tabel 5.1, 5.2, dan 5.3.

Table 5.1. Spesifikasi komputer Domain server

Perangkat Keras	Prosesor : Intel Pentium IV 2400 MHz (VMware) Memory : 480 MB
Perangkat Lunak	Sistem Operasi : Microsoft Windows *Server* 2003 Standard Edition Aplikasi penunjang : Active Directori

Table 5.2. Spesifikasi komputer client untuk pengembangan

Perangkat Keras	Prosesor : AMD barton 2.6 MHz Memory : 512 MB
Perangkat Lunak	Sistem Operasi : Debian linux Aplikasi penunjang : -

Table 5.3. Spesifikasi komputer client uji-coba

Perangkat Keras	Prosesor : Intel Pentium IV 2400 MHz (VMware) Memory : 100 MB
Perangkat Lunak	Sistem Operasi : Debian linux Aplikasi penunjang : -

5.2. KONFIGURASI DAN INSTALASI APLIKASI PADA *CLIENT*

Setelah dilakukan riset dan percobaan untuk menentukan konfigurasi yang cocok dan pembuatan aplikasi untuk dapat membuat Linux debian dapat login menggunakan login Windows *domain* seperti menggunakan di Windows. Maka hasil tersebut diujicobakan dengan *client* lain yang juga menggunakan sistem operasi Linux Debian. Jumlah komputer *client* yang diujicobakan dalam percobaan ini berjumlah 7 buah. Sehingga total terdapat 1 komputer *server domain* dan 8 komputer *client* dalam *domain* TA-FANNI.

5.2.1 Konfigurasi *Client*

Sistem Operasi komputer *client* yang digunakan adalah Linux Debian dengan instalasi paket yang paling minim. Sehingga dibutuhkan beberapa paket tambahan yang perlu di-*install* agar konfigurasi dapat dilakukan. Paket-paket tersebut antara lain:

- Samba → Paket asli samba, paket yang dibutuhkan oleh linux untuk berhubungan dengan sistem selain linux.
- Smbfs → Paket yang digunakan untuk mengakses tipe koneksi yang menggunakan fasilitas samba.
- Smb*client* → Paket samba yang berfungsi sebagai sisi *client* dari koneksi samba.
- Samba-*client* → Paket samba yang berfungsi sebagai sisi *client* yang dapat membuka koneksi dengan samba.
- Winbind → Paket samba yang digunakan dalam urusan otentikasi dengan Windows.

- Krb5-kdc → Paket instalasi kerberos.

Setalah proses instalasi diatas dilakukan, kemudian tiap-tiap *client* melakukan mapping ke dalam *folder* khusus yang telah disiapkan oleh penulis dalam komputer *server*. Didalam *folder* tersebut telah disiapkan beberapa hal berikut:

- *File-file* konfigurasi yang dibutuhkan
- *File-file* aplikasi yang dibutuhkan.

Kemudian *file-file* konfigurasi tersebut di-*copy*-kan ke dalam komputer *client* dengan menindih *file-file* konfigurasi aslinya. Dan menjalankan ulang service samba dan winbind seperti yang telah dijelaskan dalam implementasi.

5.2.2 Instalasi Aplikasi

Dan untuk *file-file* aplikasi, *file-file* tersebut di-*copy*-kan ke dalam tiap-tiap komputer *client*. *File-file* aplikasi tersebut terbagi dalam 2 *folder* yang mewakili masing-masing aplikasi, *folder* tersebut antara lain:

- openssh-3.8.1p1
- shadow-4.0.3

Kemudian masing-masing aplikasi tersebut di-*compile* ulang semua agar ter*install*.

5.3. Pelaksanaan Uji Coba dan Evaluasi

Uji coba yang akan dilaksanakan terdiri dari 3 macam yaitu uji coba keberhasilan, uji coba ketahanan dan uji coba dalam keadaan tidak normal. Dibawah ini akan dilaporkan hasil dari kedua uji coba tersebut.

5.3.1 Uji Coba dan Evaluasi Keberhasilan

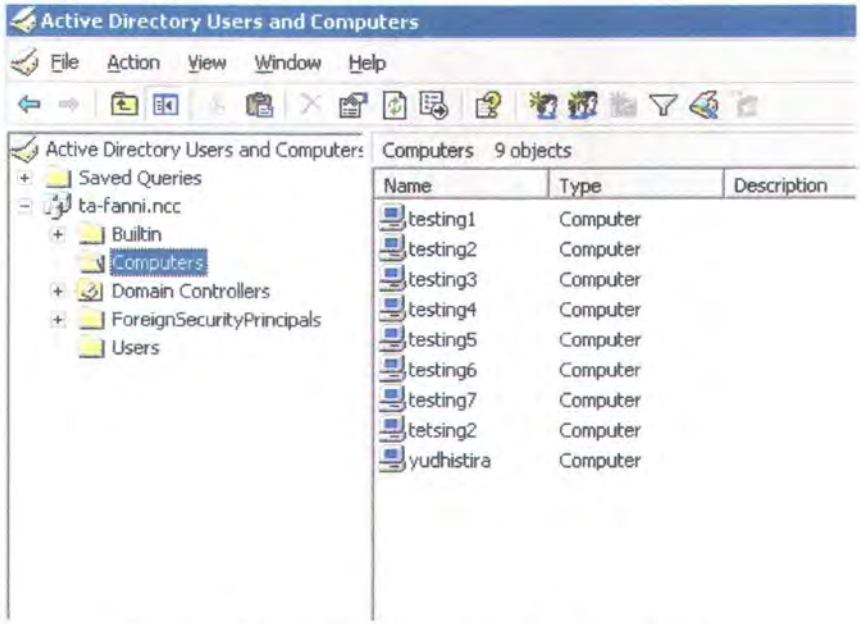
Uji coba keberhasilan dari pengerjaan TA ini dibagi dalam 4 bagian. Yaitu uji coba penggabungan anggota dalam *domain*, uji coba penggunaan login *domain* di terminal komputer *client*, dan uji coba penggunaan login *domain* menggunakan *remote login* (ssh). Dalam uji coba penggunaan *domain*, parameter yang digunakan adalah keberhasilan login dan kemampuan untuk mengambil data-data user yang ada di dalam *server*.

5.3.1.1 Uji coba penggabungan anggota kedalam *domain*

Setelah komputer *client* telah dikonfigurasi, *client* dapat mendaftarkan diri kedalam *domain* seperti yang telah ditunjukkan dalam bab implementasi. Mengingat keterbatasan resources, dalam uji coba ini hanya dapat dilakukan dengan jumlah semua *client* untuk uji coba sebanyak 7 komputer dan client yang dipakai untuk pengembangan yang sebanyak 1 komputer yang akan digabungkan dalam *domain* TA-FANNI. Nama masing-masing komputer tersebut antara lain:

- Testing1
- Testing2
- Testing3
- Testing4
- Testing5
- Testing6
- Testing7
- yudhistira

Kemudian hasil akhir penggabungan dari semua *client* tersebut terlihat dalam *domain server* seperti yang terlihat dalam gambar 5.2:



Gambar 5.2Contoh hasil penggabungan anggota domain

5.3.1.2 Evaluasi penggabungan anggota kedalam *domain*

Dari hasil uji coba diatas 8 dari 8 komputer yang digunakan dalam pengerjaan TA ini berhasil seluruhnya bergabung dan terdaftar dalam *domain* yang telah disiapkan. Berarti uji coba telah berjalan dengan baik.

5.3.1.3 Uji coba login terminal

Setelah menjalankan uji coba penggabungan *domain* diatas. Maka *client-client* ber-OS Linux yang telah terdaftar telah dapat melakukan proses login.

Uji coba berikut ini adalah uji coba login menggunakan terminal login yang dilakukan oleh masing-masing *client* berikut adalah hasil percobaan login dari ketujuh *client* beserta keterangan paket yang beredar dalam jaringan untuk melakukan otentikasi.

Dalam tabel 5.1 terlihat bahwa ketujuh client yang dijadikan client berhasil memnuhi parameter percobaan dengan dapat melakukan login menggunakan login *domain*. Dalam skenario uji coba ini login *domain* yang digunakan adalah administrator.

Tabel 5.1Tabel keberhasilan login di terminalmenggunakan login domain

client	Login		logout	
	Jumlah	waktu	jumlah	waktu
Testing1	40	0.651	20	0.401
Testing2	40	0.815	20	0.272
Testing3	40	0.556	20	0.261
Testing4	40	0.713	20	0.285
Testing5	40	0.449	20	0.277
Testing6	40	0.515	20	0.387
Testing7	40	0.619	20	0.428

Kemudian contoh hasil login terminal yang berhasil menggunakan login *domain* dapat dilihat dalam gambar 5.3.

```
Login incorrect

testing2 login:
Login timed out after 60 seconds.

Debian GNU/Linux 3.1 testing2 tty1

testing2 login: administrator
Password:
The user 'administrator' is already a member of 'admindomain'.
mkdir: cannot create directory '/home/TA-FANNI/administrator': File exists
Password:
Last login: Sun Jan  8 16:12:54 2006 on tty1
Linux testing2 2.6.8-1-386 #1 Mon Sep 13 23:29:55 EDT 2004 i686 GNU/Linux

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
administrator@testing2:~$ ls
123            All Users     fanni          NetworkService
Administrator  Default User  LocalService
administrator@testing2:~$ _
```

Gambar 5.3. Contoh hasil login di terminal menggunakan login domain

5.3.1.4 Evaluasi login Terminal

Dari hasil uji coba diatas terlihat bahwa aplikasi terminal login yang telah berhasil di-*install* di seluruh client uji coba. Dan hasil instalasi berjalan dengan baik secara keseluruhan. Terbukti dari semua client berhasil melakukan login di terminal dengan *account domain*. Dan berhasil menjalankan aplikasi dengan baik.

5.3.1.5 Uji coba *remote* login

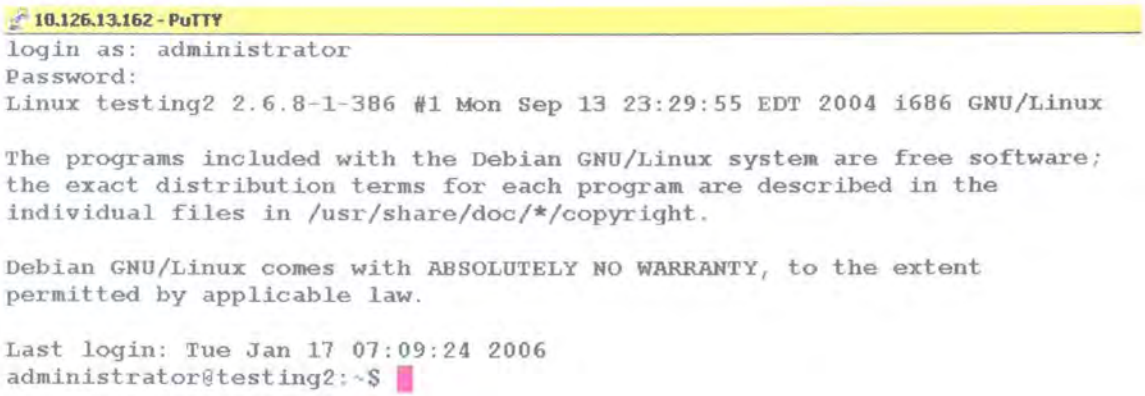
Uji coba dilanjutkan dengan cara login yang berbeda. Yaitu login menggunakan *remote* login menggunakan ssh. Berikut adalah hasil percobaan login *domain* menggunakan fasilitas login *remote*.

Dalam tabel 5.2 terlihat bahwa ketujuh client yang dijadikan client berhasil memnuhi parameter percobaan dengan dapat melakukan login menggunakan login *domain*. Dalam skenario uji coba ini login *domain* yang digunakan adalah administrator.

Tabel 5.2. Tabel keberhasilan remote login dengan login domain

client	Login		logout	
	Jumlah	waktu	jumlah	waktu
Testing1	40	0.619	20	0.582
Testing2	40	0.588	20	0.427
Testing3	40	0.631	20	0.532
Testing4	40	0.548	20	0.523
Testing5	40	0.568	20	0.428
Testing6	40	0.547	20	0.588
Testing7	40	0.684	20	0.708

Dan contoh hasil login *domain* yang menggunakan *remote* login dapat dilihat pada gambar 5.4



Gambar 5.4 Contoh hasil remote login menggunakan login domain

5.3.1.6 Evaluasi *remote* Login

Dari hasil uji coba diatas terlihat bahwa aplikasi *remote* login yang telah berhasil di-*install* di seluruh *client* uji coba. Dan hasil instalasi berjalan dengan baik secara keseluruhan. Terbukti dari semua client berhasil melakukan login dengan *account domain*. Dan berhasil menjalankan aplikasi dengan baik.

5.3.2 Uji Coba Ketahanan

Yang dimaksud dengan uji coba ketahanan adalah uji coba seberapa tahan sistem yang telah dibuat untuk menerima keadaan yang lebih berat dibandingkan dengan keadaan normal. Dimana dalam kasus ini adalah keadaan dimana banyak *user* yang melakukan login secara spontan menggunakan login *domain*.

Mengingat keterbatasan fasilitas, maka pelaksanaan uji coba ini dilakukan menggunakan satu nama *user* dan satu *client*. Dengan asumsi bahwa masing-

masing permintaan login dianggap berasal dari *client* yang berbeda-beda dengan *user* yang berbeda-beda juga. Dan hasilnya dapat dilihat sebagai berikut:

Jumlah login yang secara simultan 10 *user dapat dilihat dalam tabel 5.3*

Tabel 5.3. Percobaan dengan jumlah user 10

Waktu masuk (detik)	Waktu keluar (detik)	Selisih waktu (milidetik)
51.169	51.865	696
51.239	52.360	1.121
51.293	52.960	1.667
51.324	52.139	815
51.512	52.183	671
51.548	52.215	667
51.588	52.234	646
51.613	52.280	667
51.657	52.380	723
51.741	52.399	658
Rata-rata		833

Jumlah login yang secara simultan 20 *user dapat dilihat dalam tabel 5.4*

Tabel 5.4. Percobaan dengan jumlah user 20

Waktu masuk (detik)	Waktu keluar (detik)	Selisih waktu (milidetik)
18.683	19.560	877
18.732	19.286	554
18.783	20.276	1.493
18.998	20.288	1.290
19.480	20.350	870
19.145	20.385	1.240
19.192	20.434	1.242
19.269	20.476	1.207
19.348	20.519	1.171
19.383	20.562	1.179
19.443	20.606	1.163
19.468	20.647	1.179
19.527	20.690	1.163
19.622	20.725	1.103
19.626	20.741	1.115
19.747	20.788	1.041
19.753	20.891	1.138
19.778	20.971	1.193
19.840	21.600	1.760
19.912	21.100	1.188

	1.158

Jumlah login yang secara simultan 30 *user dapat dilihat dalam tabel 5.5*

Tabel 5.5. Percobaan dengan jumlah user 30

Waktu masuk (detik)	Waktu keluar (detik)	Selisih waktu (milidetik)
47.897	48.357	460
47.948	48.359	411
48.100	48.382	282
48.520	48.446	-74
48.406	48.672	266
48.450	48.906	456
48.605	49.627	1.022
48.657	50.331	1.674
48.708	50.389	1.681
48.772	50.430	1.658
48.809	50.480	1.671
48.841	50.525	1.684
48.886	50.570	1.684
49.500	50.607	1.107
49.161	50.652	1.491
49.166	50.694	1.528
49.200	50.739	1.539
49.217	50.785	1.568
49.220	50.814	1.594
49.387	50.834	1.447
49.404	51.210	1.806
49.447	51.790	2.343
49.488	51.810	2.322
49.557	51.142	1.585
49.719	51.236	1.517
49.725	51.283	1.558
49.750	51.311	1.561
49.773	51.422	1.649
49.774	51.442	1.668
49.774	51.465	1.691
		1.362

Jumlah login yang secara simultan 40 *user dapat dilihat dalam tabel 5.6*

Tabel 5.6. Percobaan dengan jumlah user 40

Waktu masuk (detik)	Waktu keluar (detik)	Selisih waktu (milidetik)
11.140	11.336	196
11.425	11.604	179

11.568	12.105	537
11.611	12.233	622
11.698	13.208	1.510
11.741	14.227	2.486
11.787	14.277	2.490
11.858	14.291	2.433
11.920	14.359	2.439
11.944	14.400	2.456
11.974	14.444	2.470
12.370	14.488	2.118
12.142	14.530	2.388
12.225	14.574	2.349
12.260	14.615	2.355
12.263	14.660	2.397
12.354	14.704	2.350
12.416	14.748	2.332
12.468	14.791	2.323
12.514	14.824	2.310
12.544	15.380	2.836
12.685	15.560	2.875
12.778	15.134	2.356
12.899	15.201	2.302
12.986	15.264	2.278
13.530	15.304	1.774
13.540	15.347	1.807
13.139	15.392	2.253
13.168	15.437	2.269
13.225	15.478	2.253
13.241	15.519	2.278
13.243	15.566	2.323
13.406	15.601	2.195
13.466	15.615	2.149
13.500	15.664	2.164
13.557	15.711	2.154
13.606	15.904	2.298
13.727	15.905	2.178
13.799	15.912	2.113
13.804	15.993	2.189
Rata-rata		2.095

Jumlah login yang secara simultan 50 *user dapat dilihat dalam tabel 5.7*

Tabel 5.7. Percobaan dengan jumlah user 50

Waktu masuk (detik)	Waktu keluar (detik)	Selisih waktu (milidetik)
40.457	41.158	701
40.481	41.159	678
40.562	41.180	618

40.606	41.397	791
40.668	42.740	2.072
40.731	42.770	2.039
40.768	42.748	1.980
40.835	43.472	2.637
41.175	44.117	2.942
41.185	44.158	2.973
41.266	44.169	2.903
41.315	44.198	2.883
41.391	44.334	2.943
41.445	44.452	3.007
41.504	44.472	2.968
41.505	44.523	3.018
41.538	44.680	3.142
41.638	44.723	3.085
41.759	44.763	3.004
41.799	44.807	3.008
41.802	44.849	3.047
41.893	44.892	2.999
41.911	44.933	3.022
41.979	44.980	3.001
42.470	45.230	2.760
42.520	45.620	3.100
42.349	45.145	2.796
42.437	45.224	2.787
42.450	45.271	2.821
42.555	45.319	2.764
42.639	45.357	2.718
42.674	45.395	2.721
42.823	45.441	2.618
42.823	45.475	2.652
42.851	45.492	2.641
42.852	45.531	2.679
42.876	45.577	2.701
43.450	45.619	2.169
43.530	45.683	2.153
43.171	45.701	2.530
43.172	45.745	2.573
43.208	45.887	2.679
43.294	45.963	2.669
43.343	46.111	2.768
43.354	46.111	2.757
43.554	46.115	2.561
43.592	46.257	2.665
43.618	46.306	2.688
43.656	46.383	2.727
43.837	46.423	2.586

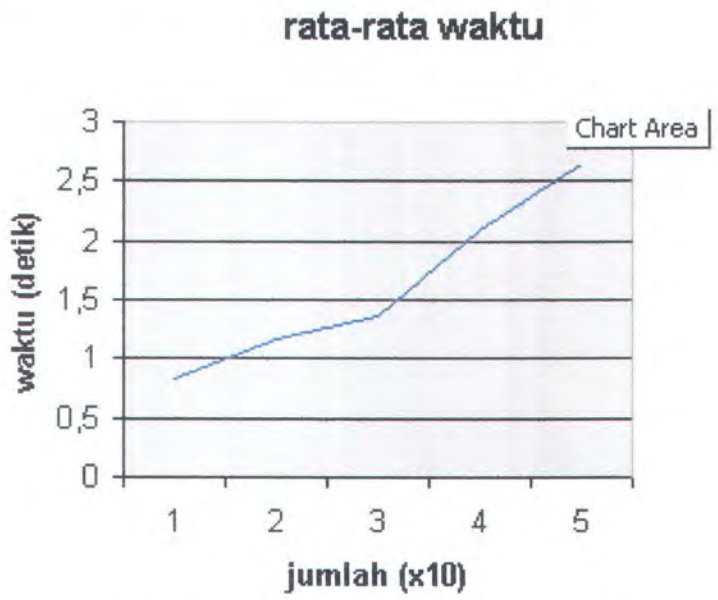
Rata-rata	2.643

Keterangan :

- Waktu masuk adalah detik dari waktu komputer client pada saat mulai aktifitas login.
- Waktu keluar adalah detik dari waktu komputer client pada saat aktifitas login selesai
- Selisih waktu adalah pengurangan dari waktu keluar dan waktu masuk yang bersesuaian.

5.3.3 Evaluasi Ketahanan

Dari hasil diatas kita dapatkan grafik waktu rata-rata yang dibutuhkan oleh setiap percobaan login apabila dalam waktu yang hampir bersamaan terjadi banyak permintaan login.



Gambar 5.5. Grafik hasil percobaan login simultan

Dari hasil percobaan terlihat bahwa semakin banyak jumlah permintaan login dalam satu waktu maka waktu proses login juga semakin besar. Dan grafik peningkatannya hampir linier.

5.3.4 Uji Coba Keadaan Tidak Normal

Yang dimaksud uji coba dalam keadaan tidak normal ini adalah kondisi login dimana data-data yang dimasukkan tidak sesuai. Keadaan ini dapat dibagi dalam 2 keadaan. Yaitu login yang dimasukkan tidak terdaftar dan *password* yang dimasukkan tidak sesuai.

5.3.4.1 Uji coba login dengan user tidak terdaftar

Uji coba user tidak terdaftar disini adalah uji coba login dengan menggunakan username yang tidak terdaftar. Baik di *server* AD ataupun di database client. Hasil dari percoban adalah gagal semua. Dan peredaran paket atas proses ini dapat dilihat pada tabel

Tabel 5.8. Tabel hasil uji coba dengan user tidak terdaftar

username	*password*	Jumlah paket	Interval (detik)
hehehe	hehehe	117	9.782
Salah	salah	32	11.813
benar	benar	12	11.301
Ilegal	Ilegal	9	8.115
Capek	capek	9	11.766

Keterangan:

- Seluruh username yang digunakan dalam percobaan merupakan username yang tidak terdaftar baik di dalam *domain* dan di dalam client.
- Jumlah paket adalah jumlah paket yang melewati jaringan antar *server* dan client saat dilakukan usaha login.
- Interval adalah selisih waktu antara paket yang pertama kali lewat hingga paket yang terakhir lewat dalam jaringan *server* dan client.
- Peredaran paket-paket dideteksi menggunakan aplikasi Ethereal

5.3.4.2 Evaluasi Login dengan user tidak terdaftar

Dari hasil uji coba uji coba login menggunakan username salah, percobaan ini berhasil semua dengan hasil bahwa tidak ada usaha login yang berhasil jika menggunakan username yang salah. Sehingga dapat diambil kesimpulan bahwa uji coba berhasil dengan baik.

5.3.4.3 Uji coba login dengan *password* salah

Uji coba dengan *password* yang salah maksudnya user login menggunakan username yang ia miliki. Namun Tidak menggunakan *password* yang benar. Baik di *server* AD ataupun di database client. Hasil dari percoban adalah gagal semua. Dan peredaran paket atas proses ini dapat dilihat pada tabel

Tabel 5.9. Tabel hasil uji coba dengan password yang salah

percobaan	username	Jumlah paket	Waktu interval (detik)

Administrator	hehehehe	123	15.859
Administrator	capek	17	15.567
123	pegel	9	10.918
123	Ngantuk	9	12.194
123	tidur	9	12.401

Keterangan :

- User Administrator memiliki *password* "administrator" dan user 123 memiliki *password* "123".
- Keterangan dari kolom yang lain sama seperti keterangan tabel dalam bab 5.3.4.1.

5.3.4.4 Evaluasi login dengan *password* salah

Demikian juga dengan uji coba login menggunakan username yang benar namun menggunakan *password* yang salah, percobaan ini pun berhasil semua dengan hasil bahwa tidak ada usaha login yang berhasil jika menggunakan *password* yang salah. Sehingga dapat diambil kesimpulan bahwa uji coba berhasil dengan baik.

BAB 6
KESIMPULAN DAN SARAN

BAB 6

KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan yang diambil berdasarkan uji coba yang telah dilakukan. Selanjutnya diberikan beberapa saran yang mungkin dapat digunakan untuk mengembangkan hasil dari aplikasi Tugas Akhir ini.

6.1. KESIMPULAN

Setelah dilakukan serangkaian uji coba dan analisa terhadap aplikasi, maka dapat diambil kesimpulan sebagai berikut :

1. Dalam Linux dapat melakukan login *domain* pada Windows *Server* 2003.
2. Login *domain* pada Linux dapat dilakukan melalui *terminal* login, *remote* login, dan login GUI.
3. Ketahanan aplikasi login *domain* ini berada pada tingkat yang baik. Karena tetap dapat melakukan otentikasi dengan baik dalam kondisi 50 permintaan logihn yang simultan.
4. Penggunaan OS Linux dalam *domain* Windows tetap menjamin keamanan data dalam jaringan tersebut. Karena Linux pun memiliki protokol enkripsi kerberos seperti yang digunakan oleh Windows *Server* 2003. Dan dari hasil uji coba terlihat bahwa tingkat keamanannya menyamai dengan keadaan jika OS *client* adalah Windows.
5. Penggunaan login *domain* di Linux tidak mengganggu performa Linux itu sendiri dan tidak menggangu *account* yang dimiliki oleh *name server* lokal.

Sehingga sistem ini baik untuk diterapkan dalam suatu *domain* dimana banyak terdapat *client* yang menggunakan OS Linux.

6.2. SARAN

Dari yang telah dikerjakan selama pembuatan TA ini ada beberapa saran yang bisa diberikan untuk pengembangan selanjutnya adalah.

Seperti yang telah diterangkan sebelumnya bahwa dalam pengembangan aplikasi login, ternyata ditemukan bahwa terdapat metode yang lebih efisien dan global. Yaitu penggunaan modul PAM. Sehingga kedepannya diharapkan adanya pengembangan dan penelitian dari modul PAM yang lain.

DAFTAR PUSTAKA

DAFTAR PUSTAKA

http://us2.samba.org/samba/

http://www.tldp.org

http://web.mit.edu/kerberos/www/

http://www.kernel.org/pub/linux/libs/pam/

http://www.die.net/doc/linux/man/man8/Winbindd.8.html

http://www.gnu.org/software/libc/manual/html_node/Name-Service-Switch.html

http://www.billboswellconsulting.com/mnpg_Active_Directory_Info.html

http://www.redmondmag.com/columns/article.asp?EditorialsID=858

http://www.microsoft.com/windows2000/*server*/evaluation/features/adlist.asp

http://www.microsoft.com/windows*server*2003/technologies/directory/activedirectory/default.mspx

Jhon H. Terpstra. *Samba-3 by example*. http://us2.samba.org/samba/. 8 Februari 2005.

Jelmer R. Vernooji, Jhon H. Terpstra, dan Gerald (Jerry) Carter. The *Official Samba-3 HOWTO and Refference*. http://us2.samba.org/samba/. 4 Februari 2005.