PENGEMBANGAN TEKNOLOGI *SINGLE SIGN ON* BERBASIS *CAS-LDAP* DI UNIVERSITAS BINA DARMA

Yesi Novaria Kunang[1], Ilman Zuhri Yadi[2]
Dosen Universitas Bina Darma
Jalan Ahmad Yani No.3 Plaju, Palembang
Sur-el: yesi_kunang@mail.binadarma.ac.id[1], ilmanzuhriyadi@mail.binadarma.ac.id[2].

Abstract: *Universitas Bina Darma has many systems and applications used by students, faculty and staff at the University. The number of systems and applications that are used provide an obstacle for users to remember the username and password to log on each of these systems. In addition, the admin should manage all users that exist in each of the distributed system, so that if a user no longer active in an application or in the entire system, then the admin should make changes to the user data in each system. For this study aims to develop a prototype of a secure single sign-on and can be implemented at the University of Bina Darma that integrate elearning systems, blogs and mail servers. With the developed system allows the users only need to log on to one of the three systems.*

Keywords: *Single Sign-on, Authentication, Bina Darma University, LDAP, and CAS*

Abstrak: *Universitas Bina Darma memiliki banyak sistem dan aplikasi yang digunakan oleh mahasiswa, dosen dan karyawan di lingkungan Universitas. Banyaknya sistem dan aplikasi yang digunakan tersebut memberikan kendala tersendiri bagi pengguna untuk mengingat user dan password login pada masing-masing sistem tersebut. Selain itu juga admin harus mengelola semua user yang ada pada masing-masing sistem yang terdistribusi tersebut, sehingga jika user tidak lagi aktif di suatu aplikasi ataupun di seluruh sistem, maka si admin harus melakukan perubahan data user di masing-masing sistem. Untuk itu penelitian ini bertujuan untuk mengembangkan prototype Single sign on yang aman dan bisa diimplementasikan di Universitas Bina Darma yang mengintegrasikan sistem elearning, blog dan mail server. Dengan sistem yang dikembangkan memungkinkan pengguna hanya perlu melakukan satu kali login pada ke tiga sistem tersebut.*

Kata Kunci*: Single Sign-on, Autentikasi, Universitas Bina Darma, LDAP, dan CAS*

1. PENDAHULUAN

Universitas Bina Darma sebagai salah satu perguruan tinggi swasta di kota Palembang yang menjadikan teknologi informasi sebagai unggulan, memiliki rencana strategis untuk mewujudkan seluruh layanan yang terintegrasi dalam satu sistem. Saat ini Universitas Bina Darma memiliki sistem akademis, dan beberapa sistem lain seperti *elearning*, *email*, *digital library*, *blog*, *eprints* dan lain-lain yang digunakan oleh civitas akademika mahasiswa, dosen, maupun karyawan. Akan tetapi sistem yang ada di Universitas Bina Darma tersebut memiliki *database* dan sistem autentikasi/*login* sendiri-sendiri. Dengan banyaknya sistem autentikasi tersebut mengakibatkan pengguna memiliki *user* dan *password* yang berbeda-beda pada masing-masing sistem tersebut. Hal ini tentu saja sangat berkendala yaitu user akan sulit mengingat *user* dan *password* yang mereka miliki. Selain itu juga kendala di sisi administrator yang harus mengelola user-user yang ada di seluruh sistem, sehingga sangat sulit bagi administrator untuk mensinkronisasi user yang ada. Sebagai contoh jika *user* yang merupakan mahasiswa yang sudah tamat maka sulit bagi administator untuk menghapus *account user* tadi di seluruh sistem yang ada.

Untuk mensinkronisasikan data pengguna sistem yang ada di Universitas Bina Darma, sejauh ini telah dilakukan antara lain dengan memanfaatkan teknologi *webservice* (Yesi, 2012 dan Nasir, 2012). Akan tetapi sinkronisasi tersebut baru dilakukan untuk di sistem autentikasi *hotspot* dan sistem otomasi perpustakaan dengan sistem akademis, belum menyeluruh ke seluruh layanan. Selain itu juga dengan sinkronisasi yang menggunakan konsep *SOA* tersebut memiliki kendala kurang *up to date* nya data pengguna sistem autentikasi, tergantung jadwal proses sinkronisasi data tersebut. Untuk itulah dalam penelitian ini peneliti tertarik untuk mencoba mengembangkan *prototype sistem Single sign on* untuk yang memudahkan pengguna layanan aplikasi yang tersedia di Universitas Universitas Bina Darma tempat diadakannya penelitian ini. Sehingga cukup dengan melakukan proses 1 kali *login* maka pengguna bisa menggunakan aplikasi/layanan lain yang tersedia di Universitas.

Permasalahan yang dibahas dalam penelitian ini adalah: 1) Merencanakan pengembangan integrasi seluruh sistem yang ada di Universitas Bina Darma dengan sistem *Single sign on* untuk autentifikasi pengguna; 2) Memetakan seluruh sistem yang ada di Universitas dan kendala integrasi sistem tersebut ke *SSO*; 3) Mendesain hirarki direktori *LDAP* pengguna yang bisa diterapkan di Universitas Bina Darma; 4) Mendesain teknologi *SSO* yang aman yang bisa diterapkan di Universitas.

Dalam penelitian ini permasalahan dibatasi pada Sistem *SSO* yang dikembangkan di Universitas Bina Darma ini berbasis teknologi *LDAP* dan CAS yang akan diujicobakan pada sistem *elearning*, *blog* dan *mail server*.

2. METODOLOGI PENELITIAN

2.1 Metode Penelitian

Penelitian ini merupakan penelitian tindakan (*action research*) yang diimplementasikan dengan mengikuti situasi aktual yang ada di Universitas. Pemilihan metoda ini dikarenakan action research bersifat praktis dan bisa langsung diimplemetasikan. Kerangka kerja yang diikuti (menurut Davinson, 2004) untuk mendapatkan pemecahan masalah adalah 1) Melakukan diagnosa (*Diagnosing*). Pada tahapan ini dilakukan identifikasi masalah-masalah pokok yang ada. Serta membuat hipotesa awal; "bisakah dikembangkan *prototype SSO* yang aman di Universitas"; 2) Membuat rencana tindakan (*Action Planning*). Pada tahapan ini kita memahami pokok masalah dengan melakukan studi pustaka yang berhubungan dengan teknologi *SSO* serta kelebihan dan kekurangannya yang ada, serta menyusun rencana tindakan yang tepat untuk menyelesaikan masalah yang ada; 3) Melakukan tindakan (*Action Taking*). Pada tahapan ini dibuat perancangan (desain) dari sistem *SSO* yang akan diterapkan di Universitas; 4) Melakukan evaluasi (*Evaluating*). Pada tahapan ini kita evaluasi hasil dari implementasi. Setelah dirancang dan dibuat *prototype SSO* dilakukan evaluasi untuk melihat kestabilan sistem dan keamanan *SSO* yang dibuat; 5) Pembelajaran

(*Learning*). Pada tahap ini kita melakukan review tahapan-tahapan yang telah berakhir dan mempelajari kriteria dalam prinsip pembelajaran. Dari hasil evaluasi bisa didapatkan kelebihan dan kekurangan prototype sistem *SSO* yang dikembangkan.

2.2 Teknologi *Single Sign On (SSO)*

Single Sign On (SSO) adalah suatu mekanisme dimana masing-masing user hanya memiliki satu akun yang berfungsi sebagai identitas user satu-satunya. Satu akun ini dapat digunakan untuk meminta izin dari sistem supaya *user* dapat mengakses berbagai aplikasi dengan *username* dan *password* yang sama dalam *session* tertentu. *Single Sign On* mengurangi jumlah *human error* yang merupakan alasan kegagalan utama dari sebuah sistem.
(http://www.opengroup.org/security/SSO/)

Keuntungan sistem *Single Sign On (SSO)*, antara lain: 1) User tidak perlu mengingat banyak *username* dan *password*; 2) Kemudahan dalam pemrosesan data. Jika setiap server memiliki data user masing-masing, maka pemrosesan data *user* (penambahan, pengurangan, perubahan) harus dilakukan pada setiap server yang ada. Sedangkan dengan menggunakan *SSO*, cukup hanya melakukan 1 kali pemrosesan.

Arsitektur Sistem *SSO* memiliki dua bagian utama yaitu *agent* yang berada di web server / layanan aplikasi dan sebuah server *SSO* yang akan dijelaskan sebagai berikut: 1) *Agent :* permintaan setiap HTTP yang masuk ke web server akan diterjemahkan oleh *agent*. Di tiap-tiap *web server* ada satu *agent* sebagai *host* dari layanan aplikasi. *Agent* ini akan berinteraksi dengan server *SSO* dan berinteraksi dengan *web browser* dari sisi pengguna; 2) SSO server: Dalam menyediakan fungsi manajemen sesi *cookies temporer* (sementara) menggunakan server *SSO*. *User-id, session creation time, session expiration time* dan lain sebagainya adalah informasi ada pada *cookies.*

Produk-produk sistem *SSO* yang berbasis *open source* yang umum digunakan pada saat ini adalah *CAS, OpenAM (Open Access Manager)*, dan *JOSSO (Java Open Single Sign-On)*. (Kelly, 2009 dan Ionut, 2011)

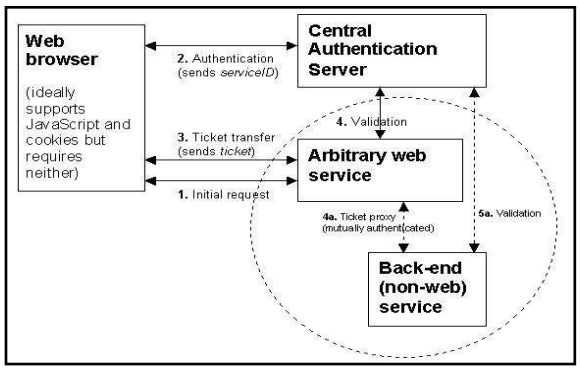
2.3 *Central Authentication Service (CAS)*

CAS adalah merupakan sebuah sistem autentikasi yang aslinya dibuat oleh Universitas Yale untuk menyediakan sebuah jalan yang aman untuk sebuah aplikasi untuk meng-autentikasi seorang user. *CAS* kemudian diimplementasikan sebagai komponen server *Java open source* dan mendukung *library client Java, PHP, Perl, Apache, uPortal*, dan lainnya. CAS server sebagai dasar untuk *framework* keamanan dan solusi *SSO* (Kristian Aaslund et al, 2007).

Dalam tahap pembuatannya, *CAS* memiliki beberapa fitur dasar layanan, diantaranya adalah: 1) Untuk memfasilitasi *Single Sign On* untuk berbagai aplikasi web. Sebagai sebuah servis inti, *CAS* tidak memerlukan *web-based* tetapi memiliki *front-end web*; 2) Memungkinkan layanan yang tidak memiliki akses ke suatu *organization* selain *ITS* (servis yang memiliki akses) untuk

mengautentikasi *user* tanpa memiliki akses pada *password*-nya; 3) Untuk mempermudah prosedur pada aplikasi untuk melakukan autentikasi; 4) Untuk membatasi autentikasi menjadi hanya pada satu aplikasi *web* yang utama, yang mempermudah *user* untuk menjaga keamanan *password*-nya dan mengijinkan aplikasi yang dipercaya untuk mengubah logika autentikasinya jika diperlukan, tanpa harus mengubah banyak aplikasi. (Sumber: http://www.jasig.org/cas).

Central Authentication Servise (CAS) merupakan aplikasi web yang berdiri sendiri. Untuk lebih jelasnya dapat dilihat pada gambar 1 (Sumber: Rudy, 2009):



Gambar 1. Arsitektur *CAS*

Halaman *URL login* utama akan menangani autentikasi. *CAS* akan memaksa user dengan *NetID* dan *password* untuk divalidasi ketika melakukan autentikasi. *CAS* menggunakan *PasswordHandler* untuk memvalidasi *username* dan *password* untuk menselarakan autentikasi.

Untuk mencegah kemungkinan dari pengulangan autentikasi, *CAS* juga mengirimkan *user* dan *password* dalam *memory cookie* (dihapus ketika *browser* ditutup). *Cookie* ini, bisa disebut "*ticket-granting cookie*", yang akan mengidentifikasi *user* setelah *user* melakukan satu kali *logged in*.

2.4 Analisis Sistem

Tujuan dalam analisa sistem yang berjalan ini adalah untuk mendapatkan informasi tentang sistem atau aplikasi apa saja yang biasanya digunakan di suatu Universitas dan kemungkinan sistem atau aplikasi tersebut diintegrasikan menggunakan teknologi *SSO*.

Berdasarkan hasil wawancara dengan UPT SIM Universitas Bina Darma tempat diadakannya penelitian, ada beberapa sistem atau aplikasi yang digunakan di Universitas tersebut antara lain:

Tabel 1. Sistem/Aplikasi yang Berjalan di Universitas

No	Jenis Aplikasi	Platform Pengembangan	Platform Server
1	Sistem Informasi Akademis	bahasa pemrograman PHP, Java, dan database MySQL	Linux
2	Sistem *elearning*	Moodle	Linux
3	Sistem *email*	Zimbra	Linux
4	Sistem *blog*	PHP, MySQL dan Wordpress	Linux
5	Sistem HRIS	PHP, MySQL dan CMS	Linux
6	Sistem Digilib	PHP dan MySQL, GDLN versi 4	Linux
7	Sistem Otomasi Perpustakaan	PHP dan MySQL	Linux
8	Website Fakultas	PHP dan MySQL	Linux
9	Website Program Studi	PHP dan MySQL	Linux
10	Sistem intranet akademis	PHP dan MySQL	Linux
11	Sistem intranet keuangan	PHP dan MySQL	Linux
12	Sistem Autentikasi Hotspot	freeradius, PHP, MySQL	Linux
13	Sistem eprint	PHP dan MySQL,	Linux
14	website unit kerja	PHP & MySQL	Linux

Dari berbagai sistem aplikasi yang berjalan di Universitas dipelajari kemungkinan untuk diintegrasikan menggunakan *LDAP*. Dari berbagai literatur dipelajari kemungkinan

aplikasi tersebut menggunakan *LDAP* untuk autentikasi *login*. Hasil dari analisis literatur untuk kesemua aplikasi yang ada di Universitas tersebut memungkinkan untuk diintegrasikan dengan teknologi *LDAP*, dengan cara modifikasi string *login* yang diarahkan ke *LDAP* dengan menambah library *LDAP*.

Dari aplikasi yang berjalan di Universitas berdasarkan platform dipelajari juga kemungkinan penerapan teknologi *SSO* berbasis CAS. Hasil analisis dari mempelajari berbagai literatur, dari semua aplikasi pada tabel 1, penerapan *SSO* CAS di berbagai aplikasi sangat memungkinkan terutama untuk aplikasi yang pernah diujicobakan berdasarkan referensi di https://wiki.jasig.org/display/CAS/CASifying+Applications. Untuk aplikasi dengan platform *zimbra, moodle, wordpress*, *elearning* dan *joomla* sangat mungkin untuk menggunakan teknologi *SSO* berbasis *CASIfy*. Demikian juga untuk aplikasi berbasius RADIUS pada referensi yang dipelajari juga sangat memungkinkan untuk menggunakan teknologi *SSO* berbasis CAS (https://wiki.jasig.org/display/CASUM/ RADIUS).

Demikian juga untuk aplikasi lain, memungkinkan penggunaan teknologi *SSO* berbasis *CAS* dengan melakukan modifikasi *script* dan menambahkan *library CAS* seperti *phpCAS* pada *script* berbasis *PHP, CAS JSP* untuk aplikasi berbasis *Java, ASP.NET, CpldFusion, Perl Client, VBScript, Symfony* dan lain-lain. Akan tetapi pada penelitian ini hanya membahas pengujian implmentasi pada beberapa aplikasi saja yaitu *elearning* berbasis *moodle, mailserver* berbasis *zimbra* dan *blog* berbasis *wordpress*. Untuk aplikasi lain terutama yang *coding*nya dibuat oleh *developer* tertentu tidak diujicobakan karena *coding* dari aplikasi tersebut yang bersifat *closesource.*

3. HASIL DAN PEMBAHASAN

3.1 Desain Perancangan Sistem

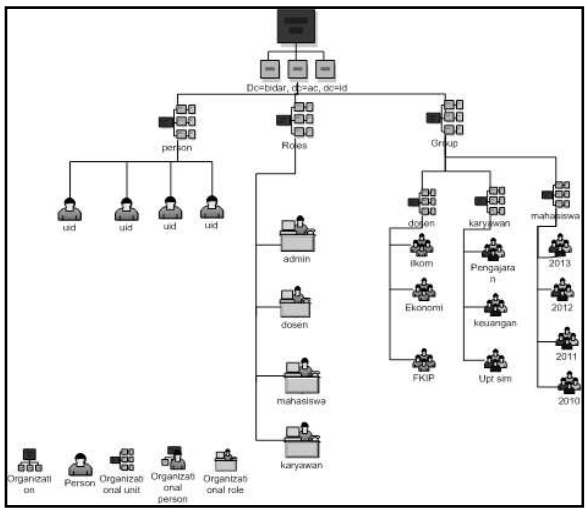
Ada 3 hal yang dirancang dan didesain perancangan (desain) dari sistem *SSO* yang akan diterapkan di Universitas, yaitu struktur hirarki *LDAP* yang akan diterapkan, topologi *SSO* yang akan diuji cobakan dan perancangan alur proses *SSO*.

3.1.1 Pengembangan Struktur Hirarki *LDAP* yang bisa Diimplementasikan untuk Sistem *SSO*

LDAP (Ligweight Directory Access Protocol) merupakan sebuah protokol yang mengatur mekanisme pengaksesan Layanan direktori (*Directory Service*). Protokol ini yang dimanfaatkan pada sistem *SSO* untuk pengelolaan pengguna semua layanan/aplikasi yang ada di Universitas.

Jika dilihat hasil pemetaan sistem aplikasi yang ada di Universitas, semua aplikasi tersebut memungkinkan untuk dimigrasikan menggunakan sistem autentikasi berbasis *LDAP*. Terutama untuk aplikasi yang berbasis *CMS*. Pada penelitian ini penggunaan *LDAP* sudah diuji cobakan pada sistem autentikasi *wireless* berbasis *radius*, sistem *elearning* berbasis *moodle*, sistem berbasis *CMS Wordpress*, berbasis *Joomla*, sedangkan untuk aplikasi yang

dikembangkan sendiri perlu melakukan perubahan *script* untuk halaman *login* agar bisa menggunakan *LDAP* (Timothy, 2003).

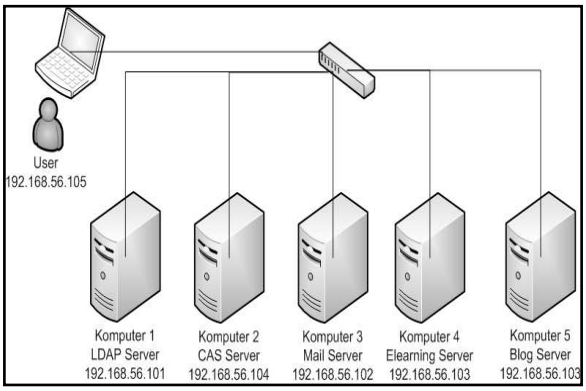


Gambar 2. Struktur Hirarki *LDAP* yang Dirancang

Perancangan struktur hirarki *LDAP* menyesuaikan hirarki organisasi yang umumnya ada di Universitas. Di Universitas umumnya terdapat tiga kategori *user* (pengguna) yaitu dosen, mahasiswa dan karyawan. Selain itu juga ada pembagian mahasiswa berdasarkan tahun anggkatan, pembagian mahasiswa berdasarkan program studi. Untuk dosen pengelompokan biasanya berdasarkan fakultas dan program studi, sedangkan untuk karyawan pengelompokannya berdasarkan unit kerja. Pengelompokan pengguna ini dimaksudkan untuk pengklasifikasian user untuk penggunaan aplikasi/ sistem. Karena biasanya beberapa aplikasi tidak ditujukan untuk semua pengguna, misalnya *email* yang hanya untuk dosen dan karyawan, *blog* yang hanya untuk untuk dosen dan lainnya.

3.1.2 Perancangan topologi *SSO* yang akan diterapka

Topologi *SSO* CAS yang diterapkan pada penelitian adalah sebagai berikut:



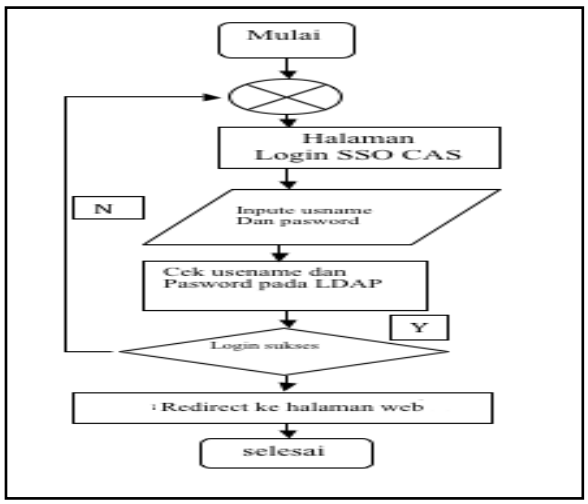
Gambar 3. Topologi Jaringan Teknologi *SSO-LDAP* yang Dirancang

Pada penelitian ini menggunakan lima buah server, dan 1 *pc notebook* sebagai *client.* Server yang dibangun terdiri dari server *SSO CAS* untuk Layanan *single sign on* sangat membantu pengguna untuk mengakses banyak layanan tanpa repot-repot memasukkan nama dan kata sandi lagi. Server yang akan diintegrasikan dengan server CAS *SSO* ini terdiri dari 3 server yaitu server *elearning* yang berbasis *moodle*, *mail server* berbasis *ZCS (Zimbra Collaboration Suite)* dan server *blog* berbasis *Wordpress*.

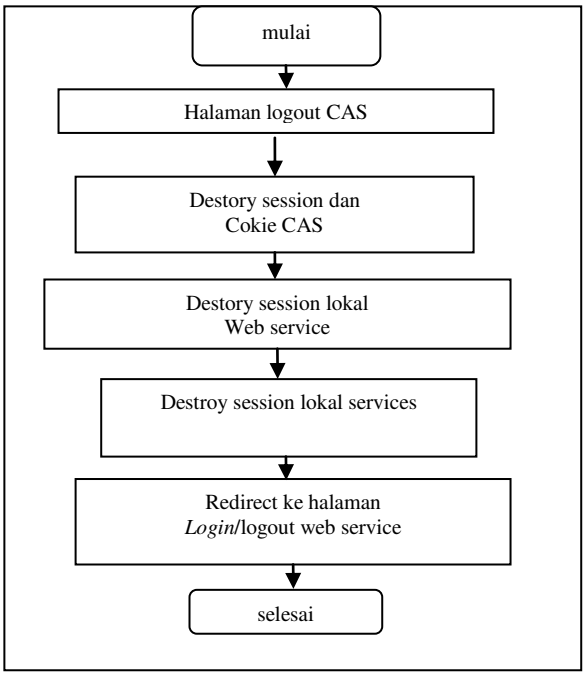
Pada Komputer server *SSO* didalamnya terdapat *CAS (Central Authentication Service)* berbasis server *Tomcat* yang menangani *proses Single Sign Out*, dan terdapat juga *MySQL* yang menyimpan *ticket registry* dan *service registry*. Sedangkan *LDAP* Server berfungsi untuk menyimpan dan memanajemen user yang menggunakan aplikasi server.

3.1.3 Perancangan Alur proses *SSO*

Perancangan alur proses *SSO* bisa dilihat pada gambar 4.



Gambar 4. Alur Proses *Login SSO*



Gambar 5. Diagram Alir Proses *logout* Aplikasi *CAS* Server

Gambar 4 adalah proses dari *login* pengguna agar dapat masuk kedalam portal web. Dimana pengguna masuk ke halaman *login* CAS, kemudian melakukan *login* dengan memasukan username dan *password* lalu dilakukan pengecekan username dan *password* pada *LDAP*. Jika *login* sukses maka proses selesai, jika tidak maka akan kembali kehalaman *login*.

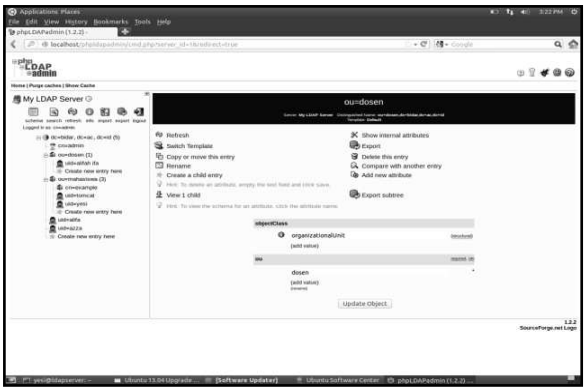
Gambar 5 adalah proses logout pengguna dimana pengguna menuju kehalaman logout dimana pada proses ini dilakukan penghancuran *session* dan *cookie* dari *local web service* kemudian *local service* dan dikembalikan ke halaman *login CAS*.

3.2 Pengembangan *Prototype SSO*

Pada penelitian ini ada beberapa server yang disiapkan yaitu:

1) Server *LDAP*

   Server *LDAP* ini digunakan untuk menampung seluruh user yang akan diautentikasi menggunakan aplikasi. Pada penelitian ini menggunakan Server Ubuntu 12.10. Untuk *LDAP* digunakan Open*LDAP*. Untuk manajemen *user* digunakan *phpLDAPadmin*

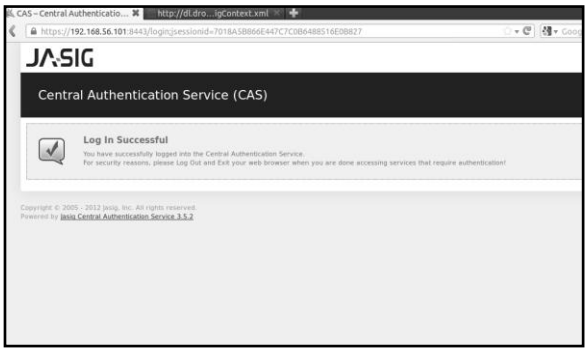


Gambar 6. php*LDAP*admin dan Open*LDAP* yang Sudah diinstal

2) Server *SSO*

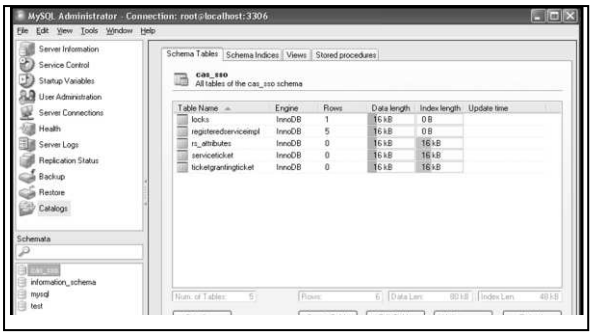
   Proses instalasi dan konfigurasi server *SSO* ini merupakan proses yang paling penting

pada tahapan penelitian ini. Pada penelitian ini server *SSO* diujicobakan pada Server Ubuntu 12.10 dan Server berbasis Windows XP. Masalah utama pada proses instalasi server *SSO* berbasis CAS ini adalah masalah sertifikat SSL yang dibuat sendiri yang tidak dikenali pada server *SSO* maupun server aplikasi yang menggunakan *SSO* sehingga proses komunikasi aplikasi server sebagai *client* dengan server *SSO* tidak berhasil. Untuk itu perlu dilakukan proses *import certificate* yang dibuat ke masing-masing server, dan karena proses pembuatan dan *import certificate* yang lebih mudah di sistem Operasi Windows maka pada penelitian ini server *SSO* yang akhirnya digunakan adalah yang berbasis Windows XP. Adapun langkah/proses instalasi server *SSO* ini membutuhkan beberapa aplikasi antara lain *Java JDK, Maven, Tomcat, dan MySQL.*



Gambar 7. Tampilan CAS *SSO login* Sucsess

Setelah proses install beberapa aplikasi tersebut tahapan selanjutnya adalah proses pembuatan *X.509 Certificates*. *CAS* menggunakan *SSL* pada proses koneksi antara server *SSO* dan server yang diproteksi saat melakukan *ticket validation*, melakukan *passing user attributes*, dan lain-lain. *SSL* ini digunakan pada semua proses koneksi antara *client* dan *server* yang proses autentikasi usernya berjalan di *cookie*. *SSO* server dan semua server yang diproteksi sebaiknya masing-masing memiliki *certificate* server. *Certificate* ini digunakan untuk melakukan proses koneksi yang dipercaya antara *clients* dan servers. Prinsip kerja *trust relation* antara server *SSO* dan server yang diproteksi sebagai berikut: 1) Server *CAS* harus percaya pada server yang diproteksi, sehingga *certificate server* tersebut harus diinstal pada SSO server's trust store, misalnya pada *JDK's cacerts keystore*. Tanpa ini server tidak akan menerima permintaan validasi dari server lain; 2) Server yang diproteksi juga harus percaya pada *CAS SSO server*, sehingga *certificate SSO* server juga harus diinstal pada masing-masing *truststore server*, misalnya pada *JDK's cacerts keystore*. Tanpa hal ini *SSO client* tidak akan menerima pesan *single sign-out*, dari *server CAS*; 3) Semua server harus dikonfigurasi mengirim *certificate* masing-masing sebagai bagian dari *SSL connection requests*. Konfigurasi kersebut dilakukan pada file server.xml tomcat. Pada penelitian ini menggunakan *self-signed certificate*. Pada proses implementasi yang sebenarnya sebaiknya menggunakan *certificate server* yang dikeluarkan oleh local *Certification Authority* atau menggunakan *commercial CA*. *Tools Portecle* digunakan untuk membuat *keystore* dan *server certificate* untuk *CAS SSO server.*



Gambar 8. Tabel *service* dan *Registry* Setelah Proses *CAS* Berhasil

3) Server *Elearning*

Server *elearning* diinstal pada Ubuntu 12.04, dengan versi *moodle* yang diinstal versi 2.2.6+ yang sudah support *LDAP* dan CAS *SSO*. *Moodle* bisa di download http://download.moodle.org/. Yang perlu dilakukan adalah memastikan bahwa di server juga sudah diinstal *mysql, library php5-curl, php5-gd, php5-intl, php5-xmlrpc, php5-LDAP,* dan *phpmyadmin* untuk mepermudah manajemen *mysql* dan *elearning*.

4) Mail Server

Pada penelitian ini menggunakan Server *Ubuntu Server 10.04.2 64 bit* dan *ZCS 7.2.2*.

5) Server *Blog*

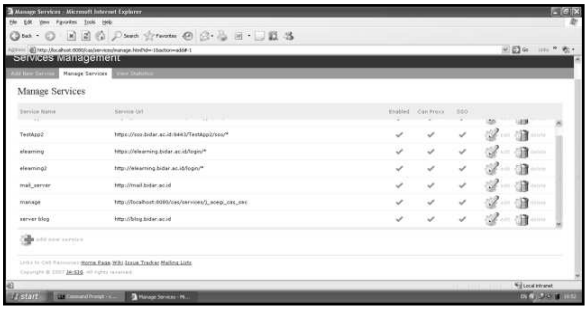
Pada penelitian yang dibuat ini Server *Blog* yang dibuat berupa *blog* berbasis *CMS wordpress* yang diinstal pada Server Ubuntu 12.04.

3.2.1 Konfigurasi Server dengan *LDAP*

Sebelum melakukan pengujian dengan Server *SSO* CAS yang sudah dibangun terlebih dahulu dilakukan pengujian autentikasi sistem server yang sudah dikembangkan dengan user autentikasi *LDAP* yang sudah dibuat pada Server Open *LDAP*. Akan tetapi yang perlu diperhatikan meskipun konfigurasi server *elearning*, zimbra dan wordpress yang dibuat dialihkan mekanisme *login*nya menggunakan *LDAP* maupun *SSO* berbasis CAS, pada masing-masing server terlebih dahulu tetap dibuatkan user *account* di masing-masing server tersebut untuk pembuatan *mailbox* pada Zimbra, pembuatan *dashboard* di *wordpress* dan di *LMS moodle*, hanya saja kita tidak perlu membuat *password* karena authentikasi akan diproses oleh *LDAP*.

3.2.2 Meng-*enable Client Services* pada *Services Management SSO*-CAS

Pada server *CAS SSO*, aplikasi server yang dikelola oleh *server CAS* harus didaftarkan pada *Service Management*. Untuk mengelola *Service Management* maka dilakukan dengan membuka aplikasi *Services Management* pada halaman *http://localhost:8080/cas/services/manage.html.* Untuk *login*nya digunakan user yang sudah ditentukan pada file *deployerConfigContext.xml*. Pada saat *login* maka tampil halaman pesan yang mengingatkan untuk menambahkan *Management service* sebagai layanan pertama yang harus dimasukkan. Setelah semua proses berjalan layanan aplikasi server lain bisa ditambahkan pada halaman *ServiceManagement*.



Gambar 9. Layanan pada *Service Management SSO CAS*

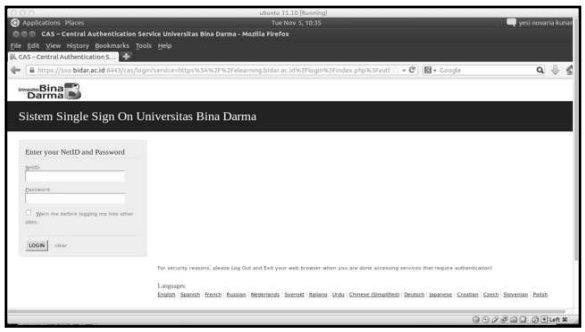
3.2.3 Konfigurasi Server dengan *SSO*

Setelah layanan aplikasi server semuanya didaftarkan pada Service Management, maka langkah berikutnya melakukan konfigurasi server *elearning*, moodle dan Zimbra untuk diintegrasikan menggunakan layanan server *SSO* berbasis CAS.

1) Konfigurasi Server *Elearning SSO- CAS*.

   Untuk mengaktifkan Autentikasi menggunakan Server *CAS SSO*, maka perlu mengaktifkan *plugin CAS Server SSO* di bagian *Manage Authentication* setelah *login* sebagai *administrator elearning*. Pada bagian Setings lakukan konfigurasi menyesuaikan konfigurasi di server *SSO* maupun di *LDAP*.

   Yang perlu diperhatikan di sini kunci komunikasi antara server *SSO* dan server *elearning* sebagai *client* menggunakan komunikasi *SSL* yang sangat tergantung dengan mekanisme *trust relationship*. Untuk itu sertifikat *SSO*server.cer yang sudah dibuat menggunakan *portecle* diimpor terlebih dahulu ke masing-masing server aplikasi yang menjadi *client server SSO*.

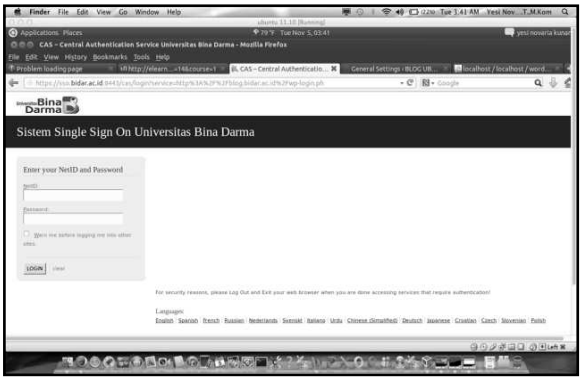


Gambar 10. Halaman *Login Elearning* Setelah *Redirect* ke Server *SSO* CAS

2) Konfigurasi Server *Blog Wordpress* dengan *SSO CAS*.

   Untuk mengintegrasikan *CAS* dengan klien php di server *blog wordpress*, maka diinstal pustaka untuk menghubungkan *CAS* server dengan klien berbasis php. Pustaka *phpCAS* dapat diunduh pada url http://www.ja-sig.org/downloads/casclients/php/1.3.2/CAS-1.3.2.tgz

   Selain menginstal *library cas-client*, yang perlu dilakukan juga adalah mengimport sertifikat digital *SSOserver.pem*. Setelah sebelumnya menginstal *Wordpress*, langkah selanjutnya adalah instalasi *plugin* otentikasi *CAS.Plugin* ini diunduh pada *http://downloads.wordpress.org/plugin/wpcas.zip* Aktifkan *plugin* melalui halaman administrasi pada bagian menu *Plugins* pilih *wpcas* kemudian klik *Activate*. Setelah mengaktifkan *plugin* tersebut, maka setelah *logout* dilakukan ujicoba *login* ke *wordpress* akan di *redirect* ke server SSO CAS seperti pada gambar 11.

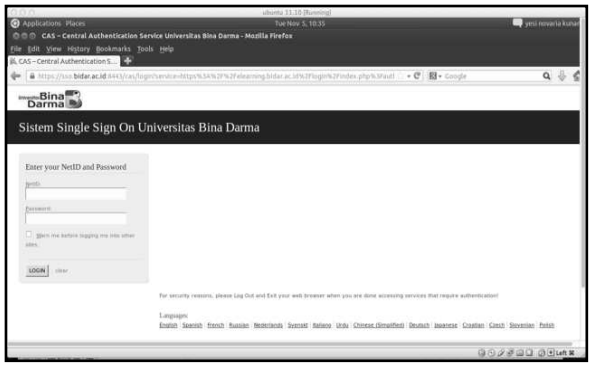


Gambar 11. Halaman *Login* Server *Blog* Setelah *Redirect* ke Server *SSO CAS*

3) Konfigurasi Server *Email* dengan *SSO CAS*.

   Untuk bisa menggunakan server *SSO CAS* make perlu diimport *CAS Server certificates* yang sudah dibuat ke dalam *Zimbra CACerts Keystore*. *Import Java CAS Client library* yang di download dari http://www.ja-

sig.org/downloads/cas-clients/. *Library* ini digunakan untuk mengimplementasikan fungsi *CAS* dan menyederhanakan aplikasi web *CASifying* menggunakan aplikasi filter. Setelah dilakukan konfigurasi maka *login* server mail akan di*redirect* juga ke server *SSO CAS*.

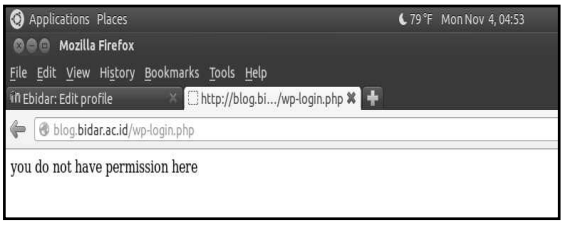


Gambar 12. Halaman *Login* Server Mail Zimbra Setelah *redirect* ke Server *SSO* CAS

3.3 Pengujian Sistem *SSO* CAS

Pada penilitian ini dilakukan Pengujian *SSO* berdasarkan hirarki user. Pengujian ini akan lebih fokus ke hirarki Group *LDAP* dengan asumsi tidak semua group user di Universitas memiliki hak ases ke suatu sistem aplikasi. Misal aplikasi yang ditujukan hanya untuk dosen seperti *blog* yang dtujukan hanya untuk dosen. Di sini akan dipelajari apakah hal tersebut memungkinkan filterisasi user di *LDAP*.

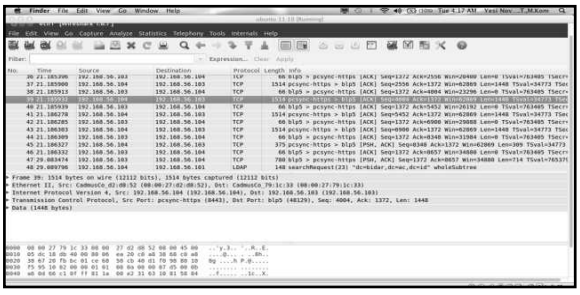
Hasil pengujian menunjukan teknologi *SSO* CAS ini sangat memungkinkan memfilterisasi user berdasarkan user group yang ditentukan. Diujicobakan dengan membuat group user dosen dan mahasiswa. Pada pengujian dicoba server *blog* yang hanya ditujukan untuk user dosen. Pada saat *login* diujicobakan *login SSO* menggunakan *account* user mahasiswa. Dari file log *LDAP* menunjukan bahwa proses *login* menemukan *user* ditemukan akan tetapi respon *LDAP* memberitahukan user tidak berhak seperti pada gambar 12.



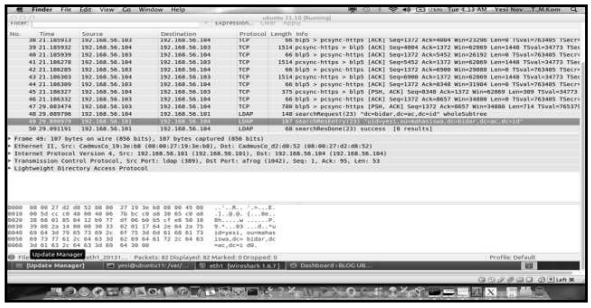
Gambar 12. Halaman *Login* Server *Blog* Setelah Redirect ke Server *SSO* CAS

Stabilitas sistem *SSO* juga diujicobakan dengan melakukan simultan *login*, untuk beberapa aplikasi. Hasilnya user yang membuka aplikasi *elearning login* dengan *SSO* CAS, bisa langsung masuk ke halaman *blog* dan *email* tanpa melakukan *login* lagi jika user memiliki otoritas untuk mengakses aplikasi tersebut.

Pada pengujian juga dilakukan pengujian untuk meninjau keamanan sistem. Pengujian dilakukan dengan meng*capture* paket menggunakan *tools wireshark*. Pada saat *login* menggunakan server *SSO CAS* terlihat proses *login* dienkripsi menggunakan protokol https. Akan tetapi setelah proses *login* maka server *LDAP* akan mengirim respon query yang di dalamnya akan terlihat user dan domain dari query *LDAP* yang tidak dienkripsi menggunakan protokol *LDAP* 389. Tetapi hasilya hanya berupa *success* atau *bindresponse LDAP* bukan *password*.



Gambar 13. Paket *Login SSO* Terenkripsi Menggunakan https



Gambar 14. Respon Query *LDAP* yang Tidak Terenkripsi

Jadi mekanisme *login* menggunakan *Single sign on* berbasis CAS *LDAP* ini aman karena komunikasi dilakukan menggunakan protokol *https* yang dienkripsi.

4. SIMPULAN

Kesimpulan pada penelitian ini adalah Sistem *Single sign on* ini sangat memungkinkan diimplementasikan pada Universitas berdasarkan hasil yang sudah diujikan pada server *SSO* berbasis CAS yang digunakan untuk mengintegrasikan layanan *elearning*, *email* dan *blog*.

Saran pada penelitian ini agar pengujian sistem *SSO* bisa dujicobakan pada sistem akademis dan sistem lainnya yang didevelop oleh developer tertentu. Selain juga sistem *SSO* ini bisa dikembangkan penelitiannya untuk sistem berbasius Radius, Shiboleth dan lainnya.

DAFTAR RUJUKAN

Aaslund, K., Larsen, S. 2007. *OTS-Wiki: A Web Community for Fostering Evaluation and Selection of Off-The-Shelf Software Components*. Department of Computer and Information Science. Norwegian University of Science and Technology (NTNU).

Anonim. *Single Sign On*. (Online). (Diakses dari http://www.opengroup.org/security/*SSO*/, 22 April 2013).

Central Authentication *Service (CAS)*. (Online). (Diakses dari http://www.jasig.org/cas., 22 April 2013).

Davinson, R.M., Martinsons, M.G., Kock N. 2004. *Information Systems dan Principles of Canonical Action Research.*

Ionut Andronache, Claudiu Nisipasiu. 2011. *Web Single Sign-On Implementation Using the SimpleSAMLphp Application*. Journal of Mobile, Embedded and Distributed Systems, vol. III, no. 1, 2011

Kelly D. LEWIS, James E. LEWIS, Ph.D.2009. *Web Single Sign-On Authentication using SAML*, IJCSI International Journal of Computer Science Issues, Vol. 2, 2009.

Nasir , M. 2012. *Sinkronisasi Data User Antara Sistem Informasi Perpustakaan dengan Sistem Informasi Akademik.* Jurnal Matrik 2012.

PPTiK UGM. 2011. *Buku Panduan TIK 2011.* Pusat Pelayanan Teknologi Informasi & Komunikasi (PPTIK) Universitas Gajah Mada. Yogyakarta.

Rudy, dan Riechie, Odi Gunadi. 2009. *Integrasi Aplikasi Menggunakan Single sign on Berbasiskan Lightwight Directory Access Protocol (LDAP) dalam Portal binus@ccess (BEE-PORTAL).* Universitas Bina Nusantara. Jakarta.

Timothy A. Howes Ph.D., Mark C. Smith, Gordon S. Good.*Understanding and Deploying LDAP Directory Services,* Second Edition. Addison Wesley.

Yesi, N.K. dan Ilman Z.Y. 2012. *Pengembangan Sistem Autentikasi Hotspot Akademis Terpusat berbasis Teknologi Web Service.* Prosiding SNATI 2012.