SISTEM SINGLE SIGN ON UNIVERSITAS BERBASIS *CAS-LDAP*

Yesi Novaria Kunang [1)], Ilman Zuhri Yadi [2)]
[1)]Sistem Informasi, Universitas Bina Darma
Jalan Ahmad Yani no.12 Plaju, Palembang
email: yesi_kunang@mail.binadarma.ac.id
[2)] Sistem Informasi, Universitas Bina Darma
Jalan Ahmad Yani no.12 Plaju, Palembang
email: ilmanzuhriyadi@mail.binadarma.ac.id

Abstrak *– Banyaknya sistem dan aplikasi yang digunakan di suatu Universitas memberikan kendala tersendiri bagi pengguna, yaitu sulitnya pengguna sistem 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. Kendala akan terjadi dengan banyaknya user dan password pada masing-masing sistem tersebut, mengakibatkan seringnya user lupa dengan password mereka. Ditambah lagi jika si 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 suatu Universitas, yang memungkinkan pengguna hanya perlu melakukan satu kali login pada keseluruhan sistem yang ada. Hasil penelitian ini berhasil mengintegrasikan layanan sistem elearning, email dan blog di Universitas menggunakan sistem Single Sign On berbasis CAS LDAP.*

Kata Kunci: Single Sign-on, Autentikasi, Universitas, LDAP, CAS

I. PENDAHULUAN

Teknologi informasi yang berkembang pesat telah umum digunakan pada semua bidang termasuk juga pada bidang pendidikan. Di suatu Universitas penggunaan teknologi informasi sudah menjadi keharusan untuk menunjang kelancaran proses belajar mengajar yang ada di Universitas tersebut. Universitas biasanya memiliki sistem akademis, dan beberapa aplikasi atau sistem lain seperti *elearning*, *email*, *digital library*, dan lain-lain yang digunakan mahasiswa, dosen, maupun karyawan di lingkungan Universitas tersebut.

Masing-masing sistem yang ada di Universitas biasanya dikembangkan oleh pengembang yang berbeda-beda dan memiliki *platform* yang berbeda-beda. Hal ini menjadi kendala bagi Universitas sendiri dikarenakan sistem-sistem 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 sehingga *user* akan sulit untuk mengingat banyaknya *user* dan *password* yang mereka miliki. Di sisi *administrator* sendiri akan mengalami kesulitan mensinkronisasi user yang ada. Jika seorang *user* mahasiswa sudah tamat maka *administrator* harus menghapus *account user* tadi satu persatu di seluruh sistem yang ada.

Banyak pendekatan yang bisa digunakan untuk mensinkronisasikan data pengguna pada beberapa sistem yang ada, salah satu cara dengan dengan memanfaatkan teknologi *webservice* [8] dan [12]. Akan tetapi sinkronisasi yang menggunakan konsep *SOA* tersebut memiliki kendala kurang *up to date* nya data pengguna sistem autentikasi, tergantung jadwal proses sinkronisasi data tersebut. Pendekatan lain menggunakan satu portal yang biasa dikenal dengan istilah *Single Sign On*. Dengan teknologi ini user cukup hanya melakukan *login* satu kali maka user bisa mengakses ke seluruh sistem lainnya yang sudah diintegrasikan tanpa perlu melakukan *login*. 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.

Beberapa perguruan tinggi sudah mulai mengenalkan sistem *Single sign on* antara lain di ITB, UI, Universitas Padjajaran. Teknologi yang digunakan antara lain menggunakan berbasis teknologi *SAML (Security Assertion Markup Language)* dan *CAS (Central Authentication Service)*. Meskipun dalam implementasinya pada perguruan tinggi yang sudah mulai menerapkan teknologi *SSO* ini masih belum mengintegrasikan seluruh layanan yang ada di Universitas tersebut dengan kendala dan kompleksitas interoperabilitas platform interoperabilitas beberapa aplikasi yang mungkin tidak mendukung suatu teknologi *SSO*. Belajar dari pengalaman perguruan tinggi lain yang telah mencoba mengimplementasikan *SSO* dengan berbagai kendalanya tersebut, maka diharapkan dengan penelitian ini nantinya dihasilkan *prototype* sistem *SSO* yang bisa diimplementasikan di Universitas.

Permasalahan yang dibahas dalam penelitian ini adalah : (1) Mendesain hirarki direktori *LDAP* pengguna yang bisa diterapkan di Universitas.; (2)

Mendesain teknologi *SSO* yang aman yang bisa diterapkan di Universitas.

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

II. LANDASAN TEORI

2.1. *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 [2].

Keuntungan sistem *Single Sign On (SSO)*, antara lain: (1) User tidak perlu mengingat banyak username dan password.; (2) Kemudahan 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*.

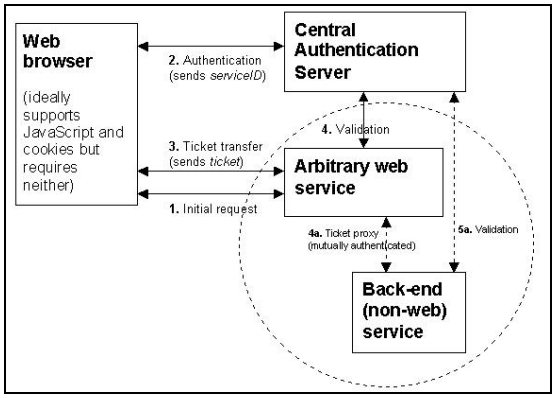
Produk-produk sistem *SSO* yang berbasis *open source* yang umum digunakan pada saat ini adalah *CAS, OpenAM (Open Access Manager)*, dan *JOSSO (Java Open Single Sign-On).*

2.2. 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 sebuah *open source* komponen server Java dan mendukung *library* dari *client* untuk Java, *PHP, Perl, Apache, uPortal*, dan lainnya. *CAS server* sebagai sebuah dasar untuk beberapa *framework* untuk keamanan dan solusi *SSO* [1].

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) 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.

Central Authentication Servise (CAS) merupakan aplikasi web yang berdiri sendiri. Untuk lebih jelasnya dapat dilihat pada gambar 1 [9].



Gambar1. 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, disebut *"ticket-granting cookie"*, yang akan mengidentifikasi user setelah user melakukan satu kali *logged in*.

2.3. *Lightweight Directory Access Protocol (LDAP)*

Lighweight Directory Access Protocol (LDAP) merupakan protokol yang mendefinisikan bagaimana data direktori dapat diakses melalui jaringan. *LDAP* biasa digunakan untuk menyimpan berbagai informasi terpusat yang dapat diakses oleh berbagai macam mesin atau aplikasi dari jaringan. Penggunaan *LDAP* di dalam sistem akan membuat pencarian informasi menjadi terintegrasi dan sangat mudah. *LDAP* seringkali digunakan untuk menyimpan nama pengguna dan sandi yang terdapat di dalam sistem secara terpusat [5].

Ada tiga definisi yang sangat penting di *LDAP*, yaitu skema, kelas, dan atribut. Ketiganya saling terkait dan menjadi tulang punggung *LDAP*. (1)

Skema: Skema bisa diibaratkan seperti sistem pemaketan untuk kelas objek dan atribut. Setiap kelas objek dan atribut harus didefinisikan di dalam skema, dan skema tersebut harus dideklarasi didalam berkas konfigurasi deamon slapd, slapd.conf. (2) Kelas Objek: Kelas objek merupakan *container* yang berfungsi untuk mengelompokkan atribut. Kelas objek akan menentukan apakah suatu atribut harus ada, atau bersifat pilihan. (3) Atribut merupakan struktur terkecil dari skema dan merupakan anggota kelas objek. Atribut memiliki nama dan juga memiliki nilai dan setiap atribut dapat memiliki lebih dari satu nilai.

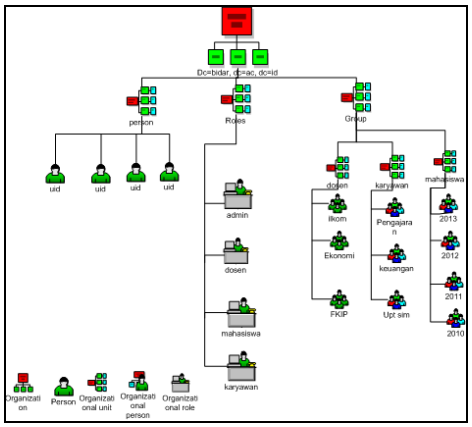
III. PEMBAHASAN

Penelitian ini merupakan penelitian tindakan (*action research*) diimplementasikan dengan mengikuti situasi aktual yang ada di Universitas. Kerangka kerja yang diikuti untuk mendapatkan pemecahan masalah adalah sebagai berikut : (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 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. Dari berbagai sistem aplikasi yang berjalan di Universitas dipelajari kemungkinan untuk diintegrasikan menggunakan teknologi *CAS* dan *LDAP*. Dari berbagai literatur dipelajari kemungkinan aplikasi tersebut menggunakan *LDAP* untuk autentikasi *login*, dan kemungkinannya diintegasikan menggunakan teknologi *SSO* berbasis *CAS* (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.

3.1. Rancangan Sistem

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

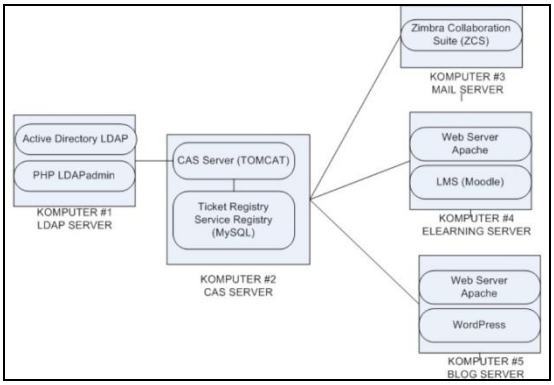
LDAP (Ligweight Directory Access Protocol) merupakan protokol yang mengatur mekanisme pengaksesan Layanan direktori *(Directory Service)*. Protokol ini yang dimanfaatkan pada sistem SSO untuk pengelolaan pengguna semua layanan/aplikasi yang ada di Universitas. Pada penelitian ini penggunaan *LDAP* sudah 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* [11].



Gambar 2. Struktur Hirarki LDAP yang dirancang

Rancangan struktur hirarki *LDAP* bisa di lihat pada gambar 2. Struktur ini menyesuaikan hirarki organisasi yang umumnya ada di Universitas. Di Universitas umumnya terdapat tiga kategori *user* 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.

Untuk Topologi *SSO CAS* yang diterapkan pada penelitian ini bisa dilihat pada gambar 3 berikut:



Gambar 3. Topologi *Logic* Teknologi *SSO- LDAP* yang dirancang

Pada gambar 3 bisa dilihat server CAS mengintegrasikan 3 layanan yaitu layanan *elearning*, *mail server* dan *blog*. Pada Komputer *SSO* didalamnya terdapat *CAS (Central Authentication Service)* berbasis server *Tomcat* yang menangani proses *Single Sign Out*. Selain itu di dalam server *SSO* terdapat juga *Ticket Registry Service* yang menangani proses *ticket registry* untuk mengatur sesi *login user*. *Service registry* dibutuhkan untuk menyimpan informasi aplikasi apa saja yang diizinkan untuk menggunakan infrastruktur *SSO*, demikian pula *user attributes* bisa diberikan untuk setiap aplikasi. Kedua *ticket registry* dan *service registry* disimpan di database *MySQL* yang disimpan di server *CAS server*. Sedangkan LDAP Server berfungsi untuk menyimpan dan memanajemen user yang menggunakan aplikasi server.

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*. Selanjutnya proses logout pengguna menuju kehalaman *logout*, 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 *OpenLDAP*. Untuk manajemen user digunakan *phpLDAPadmin*.
2. Server *SSO*; diujicobakan pada *Windows XP*. Proses instalasi server *SSO* ini membutuhkan beberapa aplikasi antara lain *Java JDK*, *Maven*, *Tomcat*, dan *MySQL*.
3. Server *Elearning*; *Server elearning* diinstal pada Ubuntu 12.04, dengan *versi moodle* yang diinstal versi 2.2.6+ yang sudah mendukung teknologi *LDAP* dan *CAS SSO*.
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.

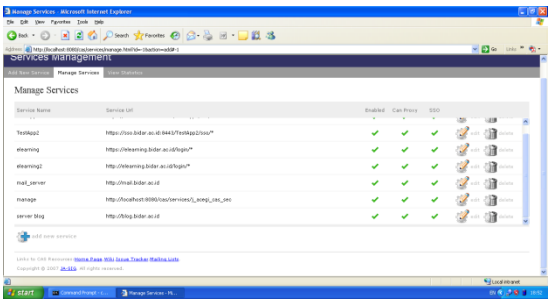
CAS menggunakan *SSL* pada proses koneksi antara *server SSO* dan server yang diproteksi saat melakukan *ticket validation*, melakukan *passing user attributes*, dan lain-lain. *SSL* ini digunakan pada semua proses koneksi antara *client* dan *server* yang proses autentikasi usernya berjalan di *cookie*.

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 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*.

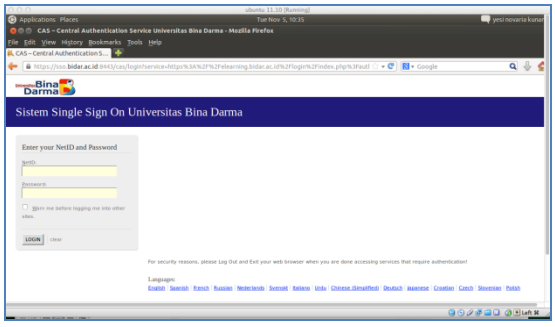
Pada *server CAS SSO*, aplikasi server yang dikelola oleh server CAS harus didaftarkan pada *Service Management* seperti pada gambar 4.



Gambar 4. Layanan pada *Service Management SSO CAS*

Di sisi aplikasi yang akan diintegrasikan perlu dilakukan konfigurasi server *elearning*, *moodle* dan *Zimbra* untuk menggunakan layanan *server SSO* berbasis *CAS*, sebagai berikut:

1. Pada *server elearning* 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*. Kunci komunikasi antara *server SSO* dan *server elearning* sebagai *client* menggunakan komunikasi *SSL* yang sangat tergantung dengan mekanisme *trust relationship.* Untuk itu sertifikat *SSOserver.cer* yang sudah dibuat menggunakan *portecle* diimpor terlebih dahulu ke masing-masing server aplikasi yang menjadi *client server SSO*.
2. 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/cas-clients/php/1.3.2/CAS-1.3.2.tgz.* Selain menginstal *library cas-client*, yang perlu dilakukan juga adalah mengimport sertifikat digital *SSOserver.pem*. Proses instalasi *plugin* otentikasi *CAS. Plugin* ini dapat diunduh pada http://downloads.wordpress.org/plugin/wpcas.zip.
3. Konfigurasi *Server Email* dengan *SSO CAS* maka 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 5. Halaman Login *elearning* setelah *redirect* ke *Server SSO CAS*

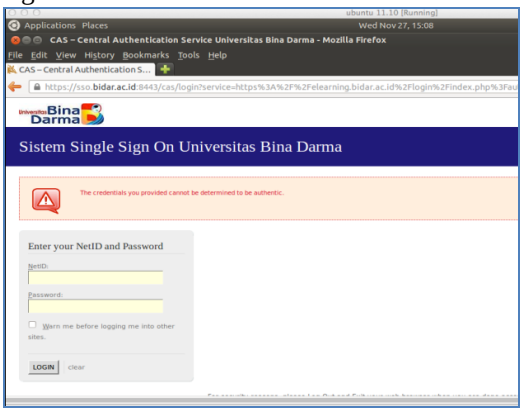
3.3. Pengujian Sistem *SSO CAS*

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.

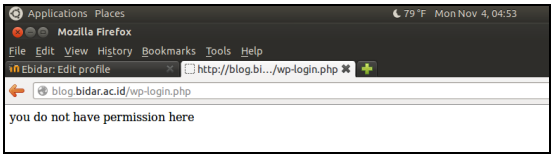
Tabel1. Skenario Pengujian dan Hasil Pengujian berdasarkan Hiraki Group User di *LDAP*

User	Otoritas	Hak Akses			Login	Akses ke server		
		elearning	blog	mail	*SSO*	*elearning*	blog	mail
User1	Tidak ada	tidak	tidak	tidak	ditolak	gagal	gagal	gagal
User2	mahasiswa	ya	tidak	ya	berhasil	berhasil	ditolak	berhasil
User3	dosen	ya	ya	ya	berhasil	berhasil	berhasil	berhasil

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 6.



Gambar 6. User 1 ditolak karena tidak ada hak akses *SSO CAS*

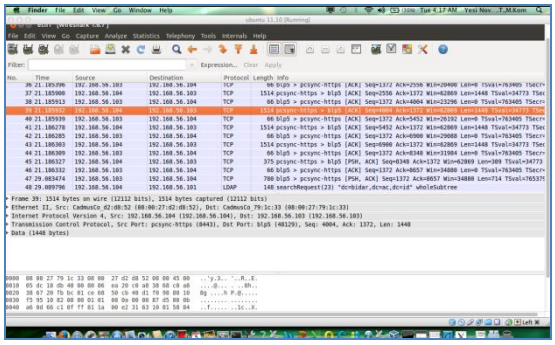


Gambar 7. User 2 yang tidak memiliki akses ke *server blog* setelah *login SSO* akan ditolak untuk mengakses *server blog.*

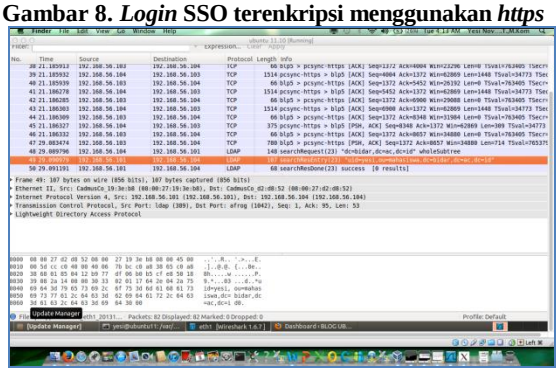
Stabilitas sistem *SSO* juga diujicobakan dengan melakukan simultan *login*, untuk beberapa aplikasi. Hasilnya user yang membuka aplikasi *elearning login* dengan *SSO CAS*, bisa langsung masuk ke halaman *blog* dan *email* tanpa melakukan *login* lagi jika user memiliki otoritas untuk mengakses aplikasi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

Pada pengujian juga dilakukan pengujian untuk meninjau keamanan sistem. Pengujian

dilakukan dengan mengcapture 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 8. *Login* SSO terenkripsi menggunakan *https*



Gambar 9. Respon *Query LDAP* yang tidak terenkripsi

Jadi mekanisme *login* menggunakan *Single sign on* berbasis *CAS LDAP* ini aman karena komunikasi dilakukan menggunakan *protocol https* yang dienkripsi. Akan tetapi untuk respon query dengan protokol *LDAP* terlihat tidak terenkripsi, sehingga bisa terlihat *query LDAP* berupa *respon LDAP* yang bisa terlihat nama *user* dan *group user* tapi tidak untuk *password*.

Untuk pengembangan perlu dipikirkan untuk mengusahakan jalur terenkripsi untuk protokol *LDAP*. Untuk itu ada 2 solusi agar saat pengiriman data *ldapserver terencypt*, yaitu menggunakan: (a) *Ldap over TLS* (*Transport Layer security*) = penggunaan saat *configure client* masih menggunakan *ldap://....* dan masih menggunkan *port* default *LDAP server* yaitu 389.; (b) *LDAP SSL* (*Secure Socket Layer*) = jika menggunakan ini saat saat *configure client* menggunakan ldaps://…….. dan akan mengubah *port* dari *LDAP server* menjadi 636.

Kemudian juga untuk website yang menggunakan Sistem *SSO CAS* ini sebaiknya dikonfigurasi untuk menggunakan protokol *https* yang terenkripsi.

IV. KESIMPULAN

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

Sistem *SSO* yang dikembangkan ini sangat memungkinkan dilakukan filterisasi berdasarkan group user di *LDAP*. *User* bisa diatur untuk akses ke beberapa aplikasi saja berdasarkan group user tersebut, meskipun login menggunakan sistem *SSO*.

Dari sisi keamanan sistem *SSO* yang dikembangkan cukup aman terutama dengan penggunaan *https* dan fitur *ldap over TLS* sehingga komunikasinya terenkripsi. Dengan demikian tidak bisa dilakukan proses *sniffing data* dan *password* oleh *hacker*.

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

DAFTAR REFERENSI

[1] Aaslund, K., Larsen, S, 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), 2007

[2] Anonim. Single sign on. http://www.opengroup.org/security/SSO. Diakses 21 Juni 2013.

[3] Central Authentication Service (CAS). http://www.jasig.org/cas. Diakses 22 April 2013.

[4] Davinson, R.M., Martinsons, M.G., Kock N, Jurnal: Information Systemsdan Principles of Canonical Action Researc, 2004.

[5] Imam Cartealy, Linux Networking, Jasakom, 2013.

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

[7] Kelly D. LEWIS, James E. LEWIS, Ph.D., Web Single Sign-On Authentication using SAML, IJCSI International Journal of Computer Science

Issues, Vol. 2, 2009.

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

[9] Rudy, dan Riechie, OdiGunadi, Integrasi Aplikasi Menggunakan Single sign on Berbasiskan Lightwight Directory Access Protocol (LDAP) dalam Portal binus@ccess (BEE-PORTAL), Jakarta, Universitas Bina Nusantara, 2012.

[10] Saputro, Muhammad Yanuar Ali, Jurnal: Implementasi Sistem Single Sign on/Single Sign Out Berbasis Central Authentication Service Protocol Pada Jaringan Berbasis Lightweight Directory Access Protocol, Universitas Diponegoro, 2012.

[11] Timothy A. Howes Ph.D., Mark C. Smith, Gordon S. Good, Understanding and Deploying LDAP Directory Services. Addison Wesley, 2003.

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

Biodata Penulis

Yesi Novaria Kunang, memperoleh gelar Sarjana Teknik (S.T), Jurusan Teknik Elektro Universitas Sriwijaya Palembang, lulus tahun 1998. Memperoleh gelar Magister Komputer (M.Kom) Program Pasca Sarjana Magister Ilmu Komputer Universitas Gadjah Mada, lulus tahun 2002. Saat ini menjadi Dosen di Universitas Bina Darma, Palembang.

Ilman Zuhri Yadi, memperoleh gelar Sarjana Komputer (S.Kom) 1999 Program Studi Manajemen Informatika STMIK Bina Darma Palembang, dan lulus magister Komputer (M.Kom.) 2011 Konsentrasi *IT Infrastructure* Universitas Bina Darma, Palembang. Saat ini menjadi Dosen di Universitas Bina Darma, Palembang.