BAB II

LANDASAN TEORI

2.1 Tinjauan Pustaka

Kendali akses adalah cara bagaimana administrator mengendalikan siapa saja yang boleh mengakses server dan layanan apa yang diperbolehkan bagi mereka (CISCO, 2010). Keamanan jaringan dengan *authentication* (otentikasi), *authorization* (otorisasi), dan *accounting* (AAA) menyediakan *framework* utama bagaimana administrator mengendalikan pengaksesan terhadap *router* atau server. Otentikasi menyediakan metode pengidentifikasian pengguna, termasuk dialog *login* dan *password*, *challenge* dan *response*, dukungan komunikasi pesan *(messaging)* dan enkripsi (tergantung protokol keamanan yang dipilih).

Otentikasi pada organisasi dengan berbagai aplikasi dilakukan dengan mekanisme *Single Sign On (SSO)*. SSO merupakan komponen yang sangat penting pada arsitektur keamanan sebuah organisasi (Ponnapalli, 2005). Dalam teknologi informasi diyakini bahwa implementasi SSO pada perusahaan mahal dan mungkin terus melebar. Bagaimanapun, perhatian perusahaan terhadap keuntungan pengimplementasian SSO semakin baik. Sebagian besar produk SSO yang ada di pasar, berdasarkan arsitekturnya, dapat dikategorikan menjadi dua tipe yaitu:

a. *Web-based* (juga dikenal sebagai enterprise SSO atau ESSO)

b. *Non web-based* (juga dikenal legacy SSO)

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015

Seiring dengan berkembangnya banyak aplikasi web, dimana setiap aplikasi menyediakan fasilitas otentikasi bagi anggotanya, maka setiap pengguna cenderung memiliki banyak *account*. Otentikasi dengan *Lightweight Directory Access Protocol* (LDAP) memungkinkan setiap aplikasi berbasis web dapat secara terpadu menggunakan satu informasi identifikasi pengguna yang tersimpan di direktori server LDAP. Metode ini telah berhasil diimplementasikan pada Sistem Informasi Kepegawaian Universitas Indonesia (Sari, 2006).

SSO juga telah berhasil diimplementasikan si Universitas Bina Nusantara untuk menggabungkan aplikasi yang sudah ada di binus-access ke dalam sebuah web portal (Rudy, 2009). Dengan adanya web portal yang menggunakan SSO ini berarti pengguna hanya memiliki sebuah *username* dan *password*. Pengguna hanya butuh *login* sekali untuk mendapatkan semua layanan di web portal. Hal ini mempermudah pengguna dalam menggunakan aplikasi dan tidak perlu mengingat banyak account.

Pengujian keamanan jaringan komputer sudah banyak dilakukan dengan berbagai macam platform yang ada, maka dalam tinjauan pustaka ini hanya meninjau beberapa penelitian awal yang digunakan untuk menguji keamanan sistem jaringan komputer terintegrasi dengan SSO menggunakan LDAP. Beberapa penelitian terkait pengujian keamanan sistem jaringan komputer SSO menggunakan LDAP antara lain adalah penelitian yang dilakukan oleh Yana Hendriana (2012).

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015

2.2 *Single Sign On (SSO)*

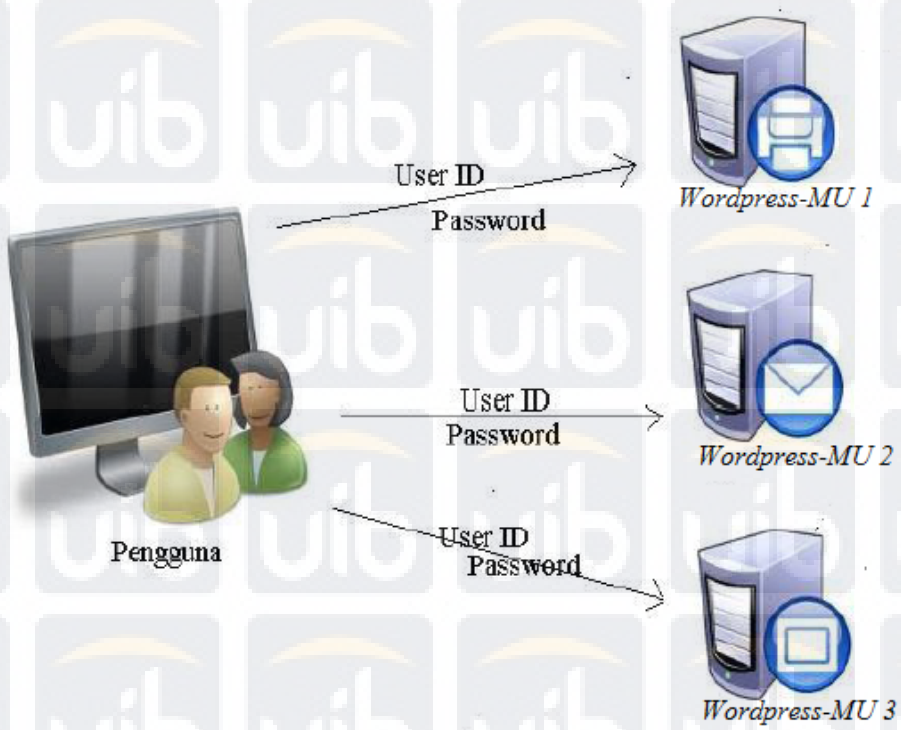
Seiring berkembangnya teknologi informasi yang mendukung proses bisnis, pengguna dan administrator dihadapkan pada semakin banyaknya aplikasi yang digunakan dalam pekerjaan mereka. Pengguna biasanya harus sign-on beberapa kali, sebanyak aplikasi yang dibutuhkan, masing-masing mungkin dengan data *login (account)* yang berbeda. Administrator juga harus mengelola banyak *account* pada masing-masing aplikasi (Open Group, 2011). *SSO* merupakan teknologi yang memiliki kemampuan untuk memasukkan *id* dan *password* yang sama untuk *login* ke beberapa aplikasi dalam suatu perusahaan. Seperti *password* adalah mekanisme otentikasi paling aman, *SSO* kini telah dikenal sebagai *reduced sign on* (*RSO*) sejak lebih dari satu jenis mekanisme otentikasi yang digunakan sesuai dengan model risiko perusahaan.

Untuk jaringan yang sangat besar dan bersifat heterogen, dimana pengguna diminta untuk mengisi informasi dirinya pada setiap aplikasi yang hendak di akses dibutuhkan *SSO*. Sistem *SSO* tidak memerlukan interaksi yang manual, untuk mengakses seluruh layanan aplikasi tanpa harus melakukan *login* dan mengetikan *password*-nya berulang kali.

SSO meng-otentikasi pengguna pada semua aplikasi yang telah di-*authorized* untuk diakses. Ini menghilangkan permintaan *authentication* lagi ketika pengguna mengganti aplikasi selama *session* berlaku (Rudy, 2009). *SSO* juga memperkenankan informasi autentikasi dan mengidentifikasi subjek secara ketat guna menghindari *login* ganda pada sistem atau kelompok sistem yang terpercaya. Sistem *SSO* juga dapat memusatkan penglolaan dari parameter sistem

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015

yang relevan pada saat bersamaan dan meningkatkan penggunaan secara keseluruhan. Pengguna layanan bisa lebih menyukai sistem *SSO* dari pada sistem *sign-on* biasa. Gambaran perbedaan sistem *sign-on* dengan sistem *Single sign-on* dapat dilihat pada gambar berikut.



Gambar 2.1 Sistem *Sign On*

UIB Repository(c)2015

Gambar 2.2 Sistem *Single Sign On*

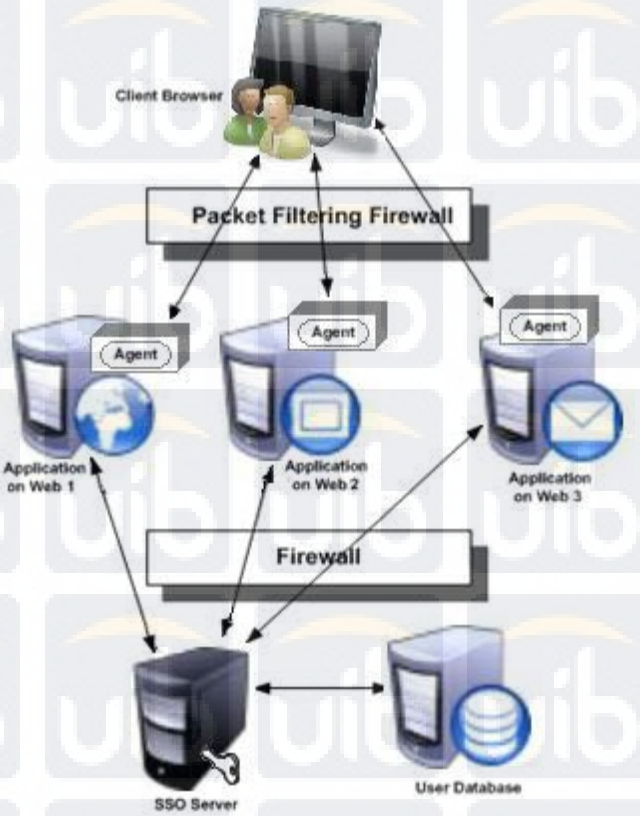
2.2.1 Arsitektur *Single Sign On (SSO)*

Solusi sistem SSO didasarkan pada salah satu dari dua tingkat pendekatan, yaitu pendekatan script dan pendekatan agent. Sistem *SSO* memiliki dua bagian utama yaitu *agent* yang berada di *web server* / layanan aplikasi dan sebuah *server SSO* berdedikasi yang akan dijelaskan sebagai berikut:

1. Agent: Sebuah agent akan menterjemahkan setiap permintaan HTTP yang masuk ke web server. Hanya ada satu agent di tiap-tiap web server, yang mana host bagi layanan aplikasi. Agent tersebut akan berinteraksi dengan web browser pada sisi pengguna, dan dengan server SSO pada sisi layanan aplikasi.
2. SSO server: Server SSO menggunakan cookies temporer (sementara) untuk menyediakan fungsi manajemen sesi. Sebuah cookies terdiri dari informasi

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015

seperti user-id, session-id, session creation time, session expiration time dan lain-lain.



Gambar 2.3 Arsitektur Sistem SSO

Beberapa arsitektur dari sistem SSO telah muncul, masing-masing dengan berbagai keunggulan dan infrastruktur yang berbeda. Pada umumnya sistem SSO memiliki beberapa keuntungan, antara lain :

1. Pengguna tidak perlu mengingat banyak username dan password.

   Cukup dengan satu *credential*, sehingga pengguna cukup melakukan proses otentikasi sekali saja untuk mendapatkan izin akses terhadap semua layanan aplikasi yang tersedia di dalam jaringan.

2. Kemudahan pemrosesan data.

   Jika setiap layanan aplikasi memiliki data pengguna masing-masing, maka

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015

pemrosesan data pengguna (penambahan, pengurangan, perubahan) harus dilakukan pada setiap aplikasi yang ada. Sedangkan dengan menggunakan sistem SSO, cukup hanya melakukan sekali pemrosesan pada server database backend-nya. Hal ini menyatakan bahwa penggunaan sistem SSO meningkatkan efisiensi waktu dan kepraktisan dalam memproses data.

3. Tidak perlu membuat data pengguna yang sama di setiap aplikasi.

   Karena setiap layanan aplikasi dalam jaringan dapat terhubung langsung dengan server database backend ini, maka hanya dengan sekali saja menginput data kedalam database, credential pengguna akan valid di seluruh layanan aplikasi.

4. Menghemat biaya untuk pemeliharaan password.

   Ketika harus me-reset password karena pengguna lupa pada password-nya, pengelola layanan tidak perlu menghabiskan waktu dan bandwith untuk menemukan data credential pengguna.

Selain mendatangkan manfaat, sistem SSO juga dapat mendatangkan kerugian, antara lain:

1. Pentingnya kesadaran pengguna untuk merahasiakan data *credential* dan menjaga keadaan *login*-nya. Bila masih dalam keadaan *login*, pengguna yang tidak sah dapat memakai mesin yang ditinggalkan pengguna sahnya.
2. Kerumitan mengimplementasikan sistem SSO kedalam sebuah jaringan yang heterogen dan multiplatform, sehingga banyak pengelola layanan jaringan kurang begitu giat dalam mengimplementasikannya.

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015

3. Kelemahan dalam hal keamanan. Jika *password* sistem pengelola layanan jaringan diketahui oleh orang yang tidak berhak, maka orang tersebut dapat melakukan perubahan terhadap semua data yang ada didalam sistem.
4. Titik Kegagalan Tunggal *(Single point failure).* Karena setiap layanan aplikasi bergantung kepada sistem *Single Sign-On*, sistem ini dapat menjadi suatu titik kegagalan bila tidak dirancang dengan baik. Kondisi apapun yang dapat menyebabkan sistem SSO padam, mengakibatkan pengguna tidak dapat mengakses seluruh layanan aplikasi yang dilindungi oleh sistem SSO tersebut.

2.2.2 Pendekatan *Single Sign-On*

Secara umum SSO diimplementasikan sebagai sebuah model otentikasi yang independen yang mana seluruh aplikasinya menggunakan modul otentikasi berbasis SSO untuk mengesahkan pengguna. Ketika pengelola layanan memilih untuk mengaplikasikan sistem SSO, maka dapat dipilih salah satu dari tiga pendekatan SSO berikut:

1. Pendekatan Terpusat *(Centralized Approaches)*

   Pendekatan ini mempunyai sebuah lokasi yang terpusat dimana seluruh identifikasi disimpan. Server SSO bertindak sebagai perantara untuk mendistribusikan identifikasi ketika dibutuhkan dan sampai pada otentikasi/otorisasi pengguna jika diperlukan. Biasanya hal ini membutuhkan pergantian aplikasi untuk mengintegrasikan dengan server SSO. Teknologi pada Microsoft's.NET Passport menggunakan pendekatan ini.

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015

2. Pendekatan Pendekatan Distribusi *(Distributed Approaches)*

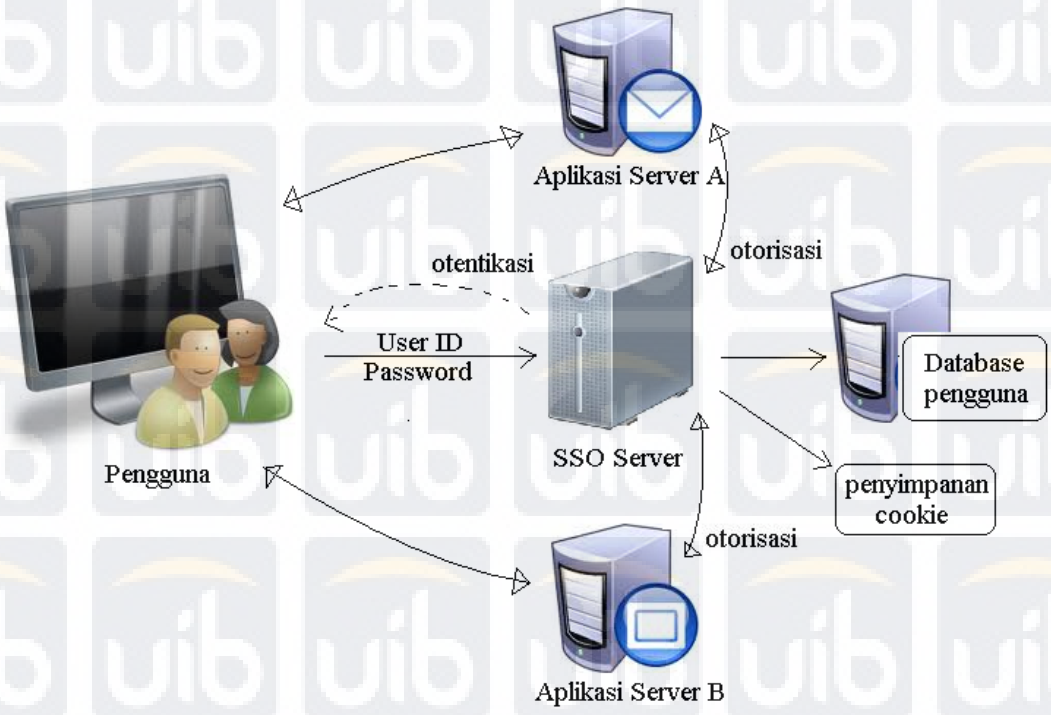
   Pendekatan ini mengijinkan kumpulan pernyataan identifikasi yang dilokalisasikan dari tiap-tiap aplikasi dengan diteruskan menggunakan komponen *client-based* (berbasis klien). Oleh karena itu pengguna memiliki kendali penuh terhadap komponen klien dan profil/password disinkronkan ke pusat server. Satu dari keuntungan pendekatan ini adalah bahwa jika seorang penyerang (attacker) mendapatkan akses informasi dari sistem, dia hanya akan menemukan akses ke informasi dari database yang sedang berjalan dari sistem tersebut.

3. Pendekatan Federasi *(Federation Approches)*

   Pendekatan ini menyediakan identifikasi terpusat dan layanan manajemen otentikasi yang sejalan dengan kumpulan pernyataan identifikasi yang dilokalisasikan. Hampir seluruh dari produk dan arsitektur SSO yang sekarang berdasarkan pada model ini. Pendekatan ini menyediakan manfaat yang paling besar dari kedua pendekatan terpusat dan distribusi.

Model yang dari sistem SSO yang digunakan berdasarkan pada pendekatan federasi, dan pendekatan ini menggunakan konsep *cookie* untuk aplikasi berbasis web. Karakteristik *cookie* memperbolehkan web server untuk mengendalikan pengguna mereka *(web browser)* secara lengkap. Pendekatan dari sistem SSO berbasis konsep *cookie* dapat ditunjukkan pada Gambar 2.3.

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015



Gambar 2.4 Pendekatan sistem SSO Berbasis Konsep *Cookie*

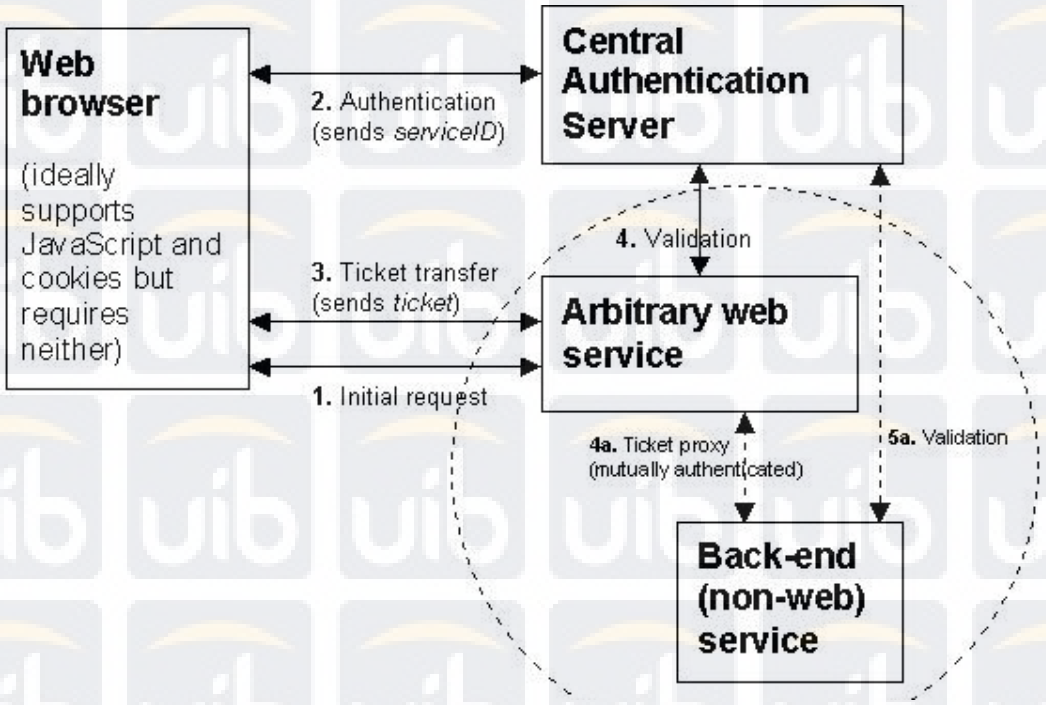
Produk-produk sistem SSO yang berbasis open source yang umum digunakan pada saat ini adalah CAS *(Central Authentication Service),* OpenAM *(Open Access Manager),* dan JOSSO *(Java Open Single Sign-On).*

2.3 Central Authentication Service (CAS)

CAS merupakan protokol SSO yang tujuannya adalah untuk memberikan ijin pada pengguna dalam mengakses beberapa aplikasi, sekaligus menyediakan *credential* pengguna (seperti *user id* dan *password*) hanya sekali, dan mengizinkan aplikasi *web* untuk meng-otentikasi pengguna tanpa mendapatkan akses ke *security credential* pengguna. CAS kemudian dikembangkan sebagai sebuah *software open source* dengan komponen *server* Java dan mendukung *library* dari *client* untuk Java, PHP, Perl, dan lainnya (Jasic, 2010).

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015

Central Authentication Servise (CAS) dibentuk sebagai sebuah aplikasi web yang berdiri sendiri. Untuk lebih jelasnya dapat dilihat pada gambar dibawah ini.



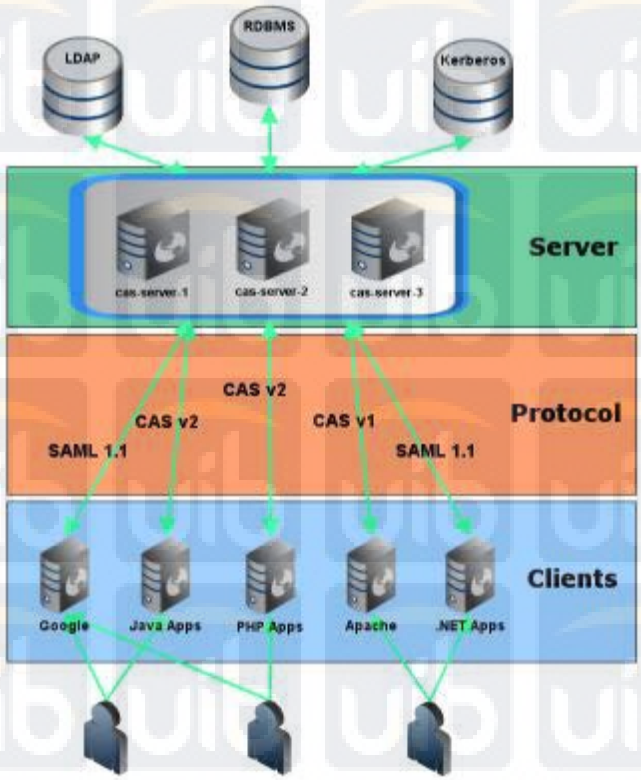
Gambar 2.5 Sistem CAS

URL login menangani otentikasi yang utama. Karena itu CAS mendorong pengguna dengan sebuah NetID dan password dan validasinya melawan provider yang mendukung autentikasi. Pengembang CAS yang berbeda akan menghubungkan bermacam PasswordHandler untuk mengesahkan username dan password melawan dukungan mekanisme otentikasi manapun yang selaras. CAS juga melakukan usaha untuk mengirimkan sesuatu dalam memory cookie (ini akan hancur ketika browser ditutup) kembali ke browser, untuk mencegah kemungkinan dari pengulangan autentikasi. Cookie ini, bisa disebut "*ticket-granting cookie"*, mengidentifikasi pengguna sebagai satu yang telah melakukan *logged in.*

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015

2.3.1 Arsitektur Dan Otentikasi CAS

Arsitektur CAS dapat digambarkan dengan sistem komponen termasuk server, klien yang berkomunikasi melalui beberapa jenis protocol. CAS server dan klien CAS merupakan dua komponen system arsitektur CAS yang mengkomunikasi melalui berbagai protocol (Addison dkk, 2011).



Gambar 2.6 Arsitektur CAS

2.3.2 *CAS Server* dan *Web Browser*

Otentifikasi yang terpusat pada mesin yang unik, inilah yang disebut CAS Server. Mesin ini hanyalah aktor yang mengetahui password pengguna. Mesin ini memiliki 2 peran utama, yaitu :

- Proses otentikasi pengguna

- Mengirim sertifikasi pengguna yang telah terotentifikasi (ke klien CAS)

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015

Aplikasi CAS Server merupakan aplikasi *Java servlet* yang mana dibangung dengan *Spring Framework* yang memiliki fungsi utama untuk otentikasi pengguna dan memberikan akses ke klien CAS dengan memvalidasi tiket. *Session* SSO dibuat saat tiket divalidasi ke pengguna pada kesuksesan login. *Service Ticket* (ST) diberikan ke pengguna melalui browser yang diteruskan menggunakan TGC sebagai token.

Web browser sendiri harus memenuhi beberapa persyaratan agar dapat menggunakan fitur CAS, diantaranya adalah :

1. Dapat mendukung SSL, dikarenakan CASberjalan dengan baik di HTTPS.
2. Dapat menampilkan HTTP *redirection* dan mengerti *Javascript* dasar.
3. Menggunakan cookie dan dalam keadaan aktif, fungsinya untuk menyimpan tiket. Ketiga persyaratan tersebut tidak perlu dikhawatirkan lagi saat ini, karena mayoritas browser saat ini sudah memiliki ketiga fitur tersebut.

2.3.3 Klien CAS

Sebuah aplikasi web yang dilengkapi dengan library / fungsi CAS klien disebut sebagai klien CAS. Berguna mengirimkan resource hanya ke klien yang sebelumnya telah terotentikasi oleh CAS server. Selain itu klien CAS memiliki satu arti lagi yang berbeda pada penggunaan umumnya, yaitu aplikasi web yang dapat diintegrasikan dengan berbagai macam variasi software. Ada 3 jenis klien CAS :

1. Library / fungsi yang berdasarkan bahasa pemrograman yang digunakan oleh klien (PHP,ASP .NET, Perl, JSP, Java), yang berfungsi mengalihkan ke halaman login terpusat CAS, dan memproses hasil otentikasi oleh CAS Server

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015

2. Sebuah modul Apache, digunakan untuk melindungi dokumen statis.
3. Modul PAM, untuk menjembatani otentikasi pada level sistem / aplikasi desktop.

2.3.4 Otentikasi CAS dengan TGC *(Ticket Granting Cookie)*

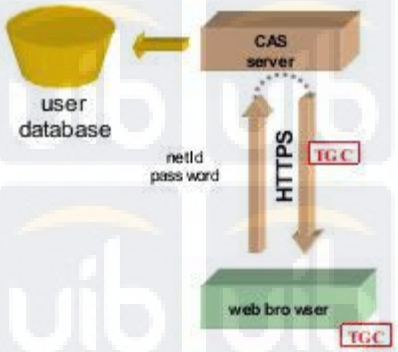
Proses otentikasi CAS yang paling sederhana adalah menggunakan Ticket Granting Cookie (TGC). Yaitu dimana jika otentikasi dengan CAS berhasil, maka akan diteruskan kembali ke halaman aplikasi web klien CAS dengan tiket sebagai bukti otentikasi yang disimpan di cookie. Tiket ini hanya berlakupada periode waktu tertentu, dan nilai tidak pernah sama / atau berubah setiap kali login. Prosesnya secara rinci adalah sebagai berikut :

1. Pertama pengguna yang belum terotentikasi berusaha mengakses aplikasi web yang telah terpasang aplikasi klien CAS dengan cara login, maka akan diteruskan ke halaman login CAS Server yang meminta pengisian username dan password.

Gambar 2.7 Otentikasi Pengguna

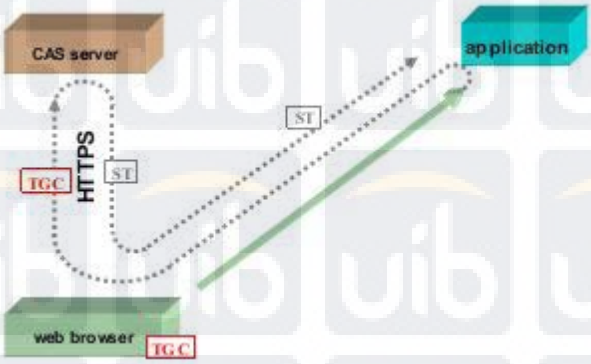
Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015

2. Jika username dan password benar maka CAS Server akan mengirimkan TGC ke browser.



Gambar 2.8 Pengiriman TGC

3. TGC merupakan passport pengguna terhadap CAS Server. Masa berlakunya terbatas dan ini merupakan cara dimana web browser mendapatkan tiket tanpa perlu adanya otentikasi kembali. Ini hanya merupakan identitas session antara CAS Server dengan web browser.
4. Sebagai presentasi dari TGC, CAS Server memberikan *Service Ticket* (ST) yang hanya dapat digunakan untuk aplikasi web dengan klien CAS yang memerlukannya tadi, yang mana dikirimkan bersamaan dengan diteruskan kembali ke halaman aplikasi web dengan klien CAS tersebut.



Gambar 2.9 Validasi dengan ST

5. Selanjutnya ST divalidasi oleh klien CAS dandidapatkan data username yang dapat digunakan untuk mengakses aplikasi web dengan klien CAS.

Gambar 2.10 Penerusan ke Klien

Melalui mekanisme *ticketing* ini tidak ada lagi informasi *password* yang disimpan dalam session maupun cookie, karena telah digantikan oleh tiket, yang mana masa berlakunya singkat dan hanya dapt digunakan satu kali saja untuk setiap aplikasi (One-Time-Ticket).

2.3.5 *Central Authentication Services Protocol (CAS Protocol)*

Klien selalu berkomunikasi dengan server melalui beberapa protokol yang didukung. Semua protocol yang didukung pada prinsipnya adalah sama, akan tetapi hanya beberapa fungsi atau karakteristik yang membuatnya sesuai untuk aplikasi atau kondisi tertentu. Sebagai contoh CAS Protocol mendukung otentikasi dengan proxy dan protocol SAML mendukung pelepasan attribute dan Single-Sign-Out.

CAS Protocol merupakan protokol berbasis tiket yang sederhana dan kuat yang dikembangkan secara eksklusif untuk CAS. Protokol CAS memiliki 2 versi, versi 1 merupakan protokol berbasis teks sederhana, sedangkan versi 2 merupakan

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015

protokol berbasis XML yang mendukung bentuk otentikasi yang tidak diketahui yaitu “Otentikasi berbasis Proxy”. CAS merupakan salah satu dari berbagai produk SSO. Perangkat lunak CAS Server sejak versi 3.4 meliputi protocol CAS v1 dan CAS v2. Sedangkan untuk proses Single-Sign-Out nya menggunakan Protokol SAML 1.1 melalui pemanggilan kembali aplikasi yang berpartisipasi dalam session *Single-Sign-On.*

2.4 Lightweight Data Access Protocol (LDAP)

LDAP merupakan *protocol* 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* didalam sistem akan membuat pencarian informasi menjadi terintegrasi dan sangat mudah. Sebagai contoh, *LDAP* seringkali digunakan untuk menyimpan nama pengguna dan sandi yang terdapat didalam sistem secara terpusat (Imam Cartealy,2013).

Pada dasarnya untuk penyimpanan informasi *LDAP* dengan penyimpanan pada aplikasi database relasional seperti *PHPMyadmin* adalah sama. *LDAP* juga melakukan proses perubahan dan permintaan pada data atau informasi seperti yang dilakukan oleh aplikasi *RDBMS*. Akan tetapi *LDAP* bukanlah aplikasi database relasional, karena *LDAP* hanya dioptimasi untuk mencari informasi cepat dan bukan untuk memproses perubahan data dalam skala besar dan cepat secara nyata.

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015

Menurut Imam Cartealy (2013) Ada tiga definisi yang sangat penting di *LDAP* yaitu skema, kelas objek dan atribut. Dimana ketiga definisi tersebut saling berkaitan 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 didalam skema, dan skema tersebut harus di deklarasikan didalam berkas konfigurasi *deamon slapd, slapd.conf*.
2. Kelas Objek: Kelas objek merupakan container yang berfungsi untuk mengelompokan atribut. Kelas objek akan menentukan apakah suatu atribut harus ada, atau bersifat pilihan.
3. Atribut: Atribut merupakan truktur terkecil dari skema yang merupakan anggota kelas objek. Atribut memiliki nama dan juga memiliki nilaidan setiap atribut dapat memliki lebih dari satu nilai.

Model informasi LDAP adalah berdasarkan entri. Sebuah entri adalah koleksi atribut yang mempunyai nama yang terbedakan (Distinguished Name/DN) secara global. DN ini digunakan sebagai referensi ke entri yang secara unik berbeda dengan nilai DN yang lainnya. Setiap atribut entri mempunyai sebuah tipe dengan satu nilai atau lebih. Tipe biasanya string singkatan khusus, seperti “cn” untuk common name,atau “mail” untuk alamat e-mail. Sintaks dari nilai bergantung kepada tipe atribut. Contoh, atribut cn mungkin berisi kata-kata “repo”. Atribut mail mungkin berisi alamat email “repo@server.com”.

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015

Menurut Carter (2003), pengertian LDAP dibagi menjadi 3 bagian, yaitu:

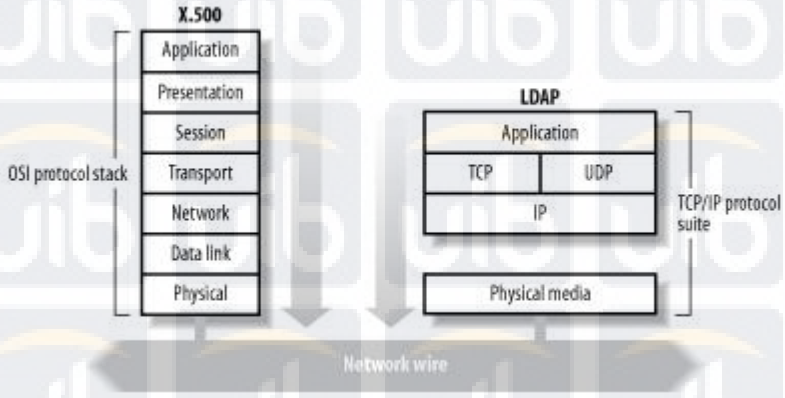
- *Lightweight*
- *Directory*
- *Access Protocol*

Lightweight

LDAP pada mulanya dibuat sebagai protocol *desktop* yang lebih muda yang digunakan sebagai *gateway* untuk X.500 *server*. X.500 merupakan sebuah standard. X.500 mendapatkan gelar *Heavyweight* karena membutuhkan *client* dan *server* untuk berkomunikasi menggunakan *Open System Interface (OSI)* protokol. Tujuh layer OSI ini bagus dalam mengaplikasikan jaringan *protocol suite*, tetapi ketika dibandingkan dengan *TCP/IP protocol suite*, ini serupa dengan bepergian jauh dengan barang bawaan yang sangat berat.

LDAP dikatakan ringan ("*lightweight"*) karena menggunakan sedikit pesan diatas udara yang dipetakan secara langsung pada TCP *layer* (biasanya *port 389*) dari protokol TCP/IP (Carter, 2003). Karena X.500 merupakan *layer* aplikasi protokol (dalam OSI layer) maka ini akan membawa lebih banyak bawaan, seperti *network header* yang dipasang pada paket pada setiap *layer* sebelum akhirnya dikirimkan ke jaringan.

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015



Gambar 2.11 X.500 dengan OSI Vs LDAP dengan TCP/IP

LDAP v3 hanya memiliki sembilan operasi utama dan menyediakan model sederhana untuk *programmer* dan *administrator*. Menyediakan satu set operasi yang lebih kecil dan sederhana yang mengijinkan pengembang untuk fokus pada seluk-beluk dari program tanpa harus mengerti keunggulan dari protokol yang jarang digunakan. Dengan jalan ini pembuat LDAP berharap untuk meningkatkan penggunaan dengan menyediakan pengembangan aplikasi yang lebih mudah.

Directory

Perlu diingat bahwa servis direktori dan basis data memiliki karakteristik masing-masing, seperti pencarian cepat dan schema yang dapat diperluas. Perbedaannya adalah direktori dibuat untuk dibaca lebihbanyak dari pada ditulis. Sedangkan untuk basis data diasumsikan untuk operasi baca dan tulis memiliki frekuensi yang sama. Asumsi bahwa direktori dibaca lebih sering dari pada ditulis merupakan suatu keunggulan yang spesifik untuk sebuah basis data, seperti dukungan untuk transaksi dan menulis *lock* merupakan sesuatu hal yang tidak penting untuk servis direktori seperti LDAP.

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015

Titik penting untuk membedakan antara LDAP dengan *backend* yang digunakan untuk menyimpan data. Ingat bahwa LDAP hanya sebuah protokol, yang penting adalah LDAP adalah sebuah set dari pesan yang digunakan untuk mengakses data yang spesifik. Protokol ini tidak mengatakan apapun tentang dimana data disimpan.

Gambar 2.12 Hubungan antara LDAP *Client,Server* dan data *Storage*

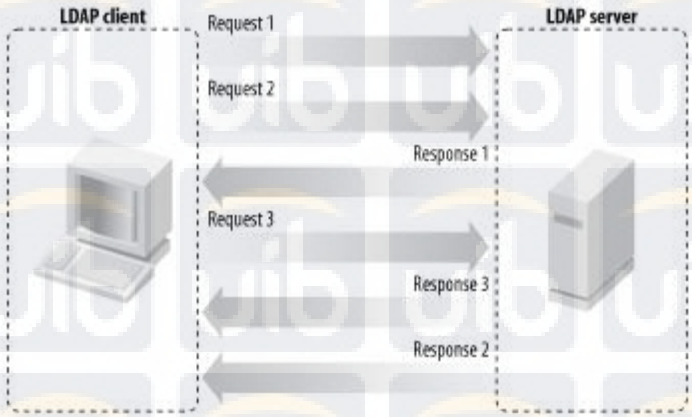
LDAP server dapat digunakan sebagai backend storage untuk web server. Semua HTML dan berkas grafis dapat disimpan dalam direktori dan dapat dipertanyakan *(query)* oleh banyak web server. Disamping semuanya, sebuah web server biasanya hanya membaca file dan mengirimkannya ke client, file itu sendiri tidak akan sering diubah. Ketika ini mungkin untuk mengimplementasikan web server yang menggunakan LDAP untuk mengakses *backend storage*, ada sebuah tipe spesial dari direktori yang telah ada, dan ini merupakan suite yang lebih baik untuk memenuhi kebutuhan melayani file, namanya adalah *file system*.

Access protocol

LDAP merupakan *message-based, client/server protckol* yang didefinisikan dalam RFC 2251. LDAP itu *asynchronous* (meskipun banyak peralatan development menyediakan baik *blocking* dan *nonblocking* API), ini berarti bahwa sebuah klien dapat menimbulkan banyaknya permintaan dan

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015

response-nya mungkin datang dalam urutan yang berbeda ketika permintaan itu dimunculkan.



Gambar 2.13 *LDAP request* dan response

2.4.1 Konsep Dasar *Lightweight Directory Access Protocol (LDAP)*

Untuk mempelajari LDAP, sangatlah penting untuk memahami maksud dari direktori dan apa fungsinya. Direktori dapat berupa personal address book, phone book, yellow pages bahkan web direktori seperti Yahoo. Direktori dapat membantu untuk menemukan informasi yang dibutuhkan, sebagai contoh yellow pages.

Di dalam yellow pages dapat dicari alamat lengkap, nomor telepon, alamat website dan email dari suatu perusahaan hanya dengan mencarinya berdasarkan ‘nama’ dari perusahaan yang telah disusun secara alphabetis pada direktori yellow pages. Dalam terminologi komputer, directory service bisa dikatakan sebagai suatu database tempat penyimpanan data, yang dapat digunakan untuk memberikan informasi-informasi yang berkaitan dengan objeknya. Bagian direktori mungkin dapat berisi kumpulan informasi tentang pengguna seperti sure

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015

name, first name, phone number, User ID, mail address dan lain sebagainya. Untuk memudahkan pemahaman tentang konsep direktori ini, dijelaskan melalui model seperti Gambar 3.4.

Gambar 2.14 Konsep Direktori

Secara prinsip struktur database pada suatu directory service adalah hierarki seperti yang ditunjukkan pada Gambar 2.6. Suatu directory service akan memiliki item yang dijadikan sebagai root. Untuk sebuah titik root, secara umum ditunjukkan dengan suatu attribut dc (Domain Component) atau o (Organization) mungkin juga ou (Organization Unit). Kemudian pada titik daun (leaf) biasanya akan berisi item dengan attribut uid (User ID) ataupun cn (Common Name). Directory service biasanya menyimpan informasi dalam bentuk struktur tree yang dinamakan Directory Information Tree (DIT). Setiap titik pada DIT diberi suatu alamat, baik secara relatif maupun secara absolut. Untuk suatu alamat relatif sering disebut sebagai RDN (Relative Distinguish Name) sedangkan alamat yang absolut di sebut sebagai DN (Distinguish Name).

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015

Jadi apabila ingin mendapatkan informasi tentang pengguna rsukmana seperti contoh pada Gambar 3.4 diatas, dapat dituliskan hasil pencariannya dengan "dn=uid=rsukmana,ou=people,dc=smartbee,dc=com". Konsep seperti inilah yang digunakan oleh direktori LDAP.

2.4. 2 Konsep Kerja *Lightweight Directory Access Protocol (LDAP)*

1. *Authentication*

a. *User and Password Authetication*

Melakukan pengecekan terhadap *username* dan *password* pada saat *login* pertama kali

b. *Generating and Retrieving Token Identity*

User mendapat seperti tiket dari sistem setelah berhasil login. 1 User kemungkinan hanya dapat masuk ke beberapa sistem saja tergantung dari hak akses yang dimiliki oleh user tersebut.

2. *Authorization*

a. *Group Authorization*

Setelah melakukan pengecekan terhadap hak akses dari user tersebut, kemudian hak akses tersebut dinyatakan dalam grup dan dalam grup tersebut memiliki hak akses untuk semuanya.

b. *Group Permission*

disini dicek halaman apa saja yg bisa diakses user yang logged-in. setelah bisa mengakses (view), user dapat melakukan apa saja terhadap halaman tersebut. kalau bisa, nanti akan ada tombol-tombol untuk tiap aksi yang muncul. tapi kalau tidak bisa, biasanya di halaman itu cuma ada tulisan-tulisan saja untuk dibaca.

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015

kalau hak view pun tidak bisa, biasanya, menu untuk mengakses halaman itu jg di hidden.

2.5 Apache Tomcat

Menurut Hidayat (2007) Tomcat, atau lengkapnya Apache Jakarta Tomcat Adalah Servlet/JSP *cotainer* yang dibuat oleh *Apache Software Foundation* (ASF). Saat ini tomcat menjadi reference implementation dari spesifikasi *Servlet /* JSP (*Java Server Pages)* dari *Sun Microsystem*. Versi terakhir dari Tomcat adalah 6.0.14. saat ini Tomcat bahkan telah dapat dijalankan sebagai *service* pada sistem operasi berbasis *Windows*. Tomcat versi 6.0.x adalah implementasi dari spesifikasi Servlet versi 2.5, dan spesifikasi JSP versi 2.1 situs web resmi dari Tomcat adalah http://jakarta.apache.org/tomcat/index.html. Tomcat merupakan Servlet/JSP container yang dibangun dalam bahasa pemrograman java dan dijalankan di atas *Java* web server *Virtual Machine* (JSM). Untuk terhubung ke sebuah diperlukan komponen yang disebutkan sebagai '*connector'*. Tomcat dapat berjalan sendiri (*Standalone*) tanpa *webserver* lain dengan menggunakan *connector Coyote.* Apabila dihubungkan dengan *Apache HTTP Server*.diperlukan *connector* JK2.

2.6 JAVA

JAVA merupakan bahasa pemograman yang dikembangkan *Sun Microsystem* yang dirilis pada tahun 1995 sebagai komponen utama dari *sun microsystem* lingkungan (*Platform*) java. Bahasa ini dikembangkan dengan model

UIB Repository(c)2015

yang mirip dengan bahasa C++ dan *smalltalk*, namun dirancang agar lebih mudah di pakai dan platform *independent*, yaitu dapat dijalankan di berbagai jenis sistem operasi dan arsitektur computer. Bahasa ini juga dirancang untuk pemograman di internet sehingga dirancang agar aman dan *portable*.

Proyek java dimulai pada bulan Juni tahun 1991 oleh james Gosling. Pada mulanya bahasa ini disebut **Oak** yang berasal dari pohon oak yang berada di luar kantor Gosling, selain itu juga pernah berubah menjadi *Greendon* dan akhirnya dinamakan Java yang berasal dari kumpulan kata acak. Gosling bertujuan untuk mengimplementasikan sebuah mesin virtual dan bahasa yang mirip dengan notasi C /C++. Sun merilis implementasi public pertamanya Java 1.0 di tahun 1995 yang Menjanjikan "*Write Once, Run Anywhere"* (WORA) dimana dapat dijalankan dengan baik pada *platform* popular, cukup aman, dan menyediakan fitur keamanan yang dapat dikonfigurasi.

Benny Handoko Dharma, Analisis dan Implementasi single sign on (sso) pada web Menggunakan Server Central Authentication Aervice, 2015
UIB Repository(c)2015