Integrasi *Central Authentication Service* (CAS) dengan *Freeradius* Sebagai Single-Sign-On

Irvan Suryadi[1], Hartanto Tantriawan[2], Ahmad Luky Ramdani[3], Rajif Agung Yunmar[4]

Program Studi Teknik Informatika, Jurusan Teknologi Produksi dan Industri,
Institut Teknologi Sumatera, Lampung, Indonesia
[1]irvan.14115002@student.itera.ac.id
[2]hartanto.tantriawan@if.itera.ac.id
[3]ahmadluky@if.itera.ac.id
[4]rajif@if.itera.ac.id

Intisari — **Perkembangan teknologi informasi membuat banyak organisasi menggunakan aplikasi berbasis web untuk memaksimalkan kinerja mereka. Penggunaan dua atau lebih aplikasi berbasis web pada suatu organisasi membuat pengguna harus mengingat lebih dari satu akun untuk dapat masuk ke aplikasi berbasis web pada organisasi tersebut. Hal ini tentu tidak efektif, maka dibutuhkan teknologi yang dapat membuat satu akun pengguna dapat digunakan untuk mengakses seluruh aplikasi berbasis web yang ada. Teknologi yang biasa digunakan adalah teknologi *Single-Sign-On* (SSO). Pada penelitian ini protokol *Single-Sign-On* yang digunakan adalah *Central Authentication Service* atau CAS. Untuk menerapkan sistem autentikasi terpusat maka digunakan FreeRADIUS sebagai *authentication server* dan *authentication protoco*l. Agar kedua teknologi ini dapat beroperasi maka pada penelitian ini akan membahas bagaimana mengintegrasikan SSO dengan protokol CAS sebagai SSO dan FreeRADIUS sebagai *authentication Server,* untuk menciptakan SSO berbasis CAS dan FreeRADIUS. Keberhasilan ditentukan pada proses *login* dan *logout,* apabila setelah *user* melakukan *login* pada portal SSO maka *user* dapat mengakses seluruh aplikasi berbasis web yang ada.**
Kata kunci — ***Central Authentication Service* (CAS), FreeRADIUS, *Single-Sign-On* (SSO).**

Abstract — ***The evolution of information technology has made lots of organizations use web-based applications to improve their work performance. Using two or more web-based applications in an organization makes users have to remember more than one account to log in to the websites. This is certainly ineffective, and it takes technology that can make users to access all existing web applications using one account. The technology commonly used is the Single-Sign-On (SSO) technology. In this study, the Single-Sign-On protocol that was developed was the Central Authentication Service or CAS. To implement a centralized authentication system, FreeRADIUS is used as an authentication server and authentication protocol. To make both technologies work, this study will explain how to integrate SSO with CAS protocol as SSO and FreeRADIUS as authentication server, to create SSO based on CAS and FreeRADIUS. Success is determined in the login and logout process. After the user has logged into the SSO portal, the user can access all existing web applications.***
Keywords - ***Central Authentication Service* (CAS), FreeRADIUS, *Single-Sign-On* (SSO)**.

I. PENDAHULUAN

Perkembangan penggunaan teknologi informasi dan juga internet sangat masif di Indonesia. Perkembangan penggunaan teknologi informasi dan juga internet sangat masif di Indonesia. Berdasarkan hasil survei Asosiasi Penyelenggara Jasa Internet Indonesia (APJII) populasi pengguna internet di Indonesia mencapai 196,71 juta jiwa dari 266,91 juta jiwa penduduk Indonesia atau sekitar 73,7% pada kuartal II tahun 2020[1].

Peningkatan penggunaan internet yang sangat tinggi di Indonesia membuat banyak lembaga atau organisasi mulai menggunakan teknologi informasi dalam kegiatan bisnis atau kelancaran organisasi mereka. Salah satu teknologi informasi yang digunakan adalah aplikasi berbasis web. Setidaknya ada satu atau lebih aplikasi berbasis web atau sistem informasi berbasis web yang digunakan dalam satu organisasi. Sebagai contoh suatu perusahaan menggunakan lebih dari satu aplikasi berbasis web untuk menunjang proses bisnis mereka seperti sistem informasi, aplikasi *email*, blog, ftp dan lain sebagainya.

Penggunaan lebih dari satu aplikasi berbasis web membuat pengguna harus memiliki atau mengingat lebih dari satu akun, tentu hal ini tidak efisien. Untuk itu dibutuhkan suatu teknologi yang dapat membuat pengguna cukup memiliki satu akun untuk dapat masuk ke semua aplikasi berbasis web yang ada. Teknologi ini biasa disebut dengan *Single-Sign-On* atau SSO, dimana apabila pengguna sudah melakukan *login* atau *logout* satu kali maka pengguna akan otomatis sudah *login* atau *logout* pada semua aplikasi. Hal ini dikarenakan dalam proses autentikasi *Single-Sign-On* menerapkan *Server* terpusat yang menyimpan seluruh data kredensial pengguna.

Salah satu penelitian tentang teknologi SSO adalah pengembangan teknologi SSO dengan *Security Assertion Markup Language* (SAML) berbasis RADIUS yang dikembangkan oleh Purba [2]. Pada penelitian ini Purba menggunakan RADIUS sebagai *Server* autentikasi.

Keunggulan dari penggunaan RADIUS *Server* karena dinilai kuat terhadap *sniffing* dan *active attacker* karena RADIUS menggunakan kunci rahasia antara RADIUS *Server* dan *client* pada proses autentikasi [3]. Sementara itu protokol SAML digunakan sebagai portal penghubung antara pengguna dengan aplikasi berbasis web di Universitas Bina Darma. Namun, menurut Somorovsky pada jurnal yang berjudul On Breaking SAML - Be Whoever You Want sekitar 80% dari *framework* SAML dapat dipatahkan dengan menghindari *integrity protection* dengan menyerang XML *Signature Wrapping* (XSW) [4].

Sebuah penelitian yang dibuat oleh Saputro menerapkan sistem SSO berbasis protokol *Central Authentication Service* (CAS) [5]. Pada bagian keamanan CAS selalu mengirimkan paket secara terenkripsi, apabila pengguna berhasil melakukan *login*, CAS akan mengirimkan *cookie* (dihapus ketika *browser* ditutup) yang berguna sebagai tiket masuk ke *web service* yang sama sekali tidak mengandung data personal dan juga data Kredensial [6].

Pada penelitian ini Saputro menggunakan LDAP *Server* sebagai protokol autentikasi pada sistem SSO-nya. Sementara itu, menurut *Open Web Aplication Security Service* (OWASP) sebuah yayasan pengembangan keamanan perangkat lunak nonprofit, autentikasi berbasis LDAP rentan terhadap serangan berupa LDAP *injection*. Serangan LDAP *injection* ini dapat menyebabkan pemberian akses terhadap *query* yang tidak sah sehingga dapat menyebabkan konten yang ada pada LDAP dimodifikasi oleh pihak yang tidak bertanggung jawab.

Berdasarkan kelebihan dan kekurangan pada penelitian sebelumnya maka dibuatlah tugas akhir tentang Integrasi *Central Authentication Service* dengan FreeRADIUS sebagai *Single-Sign-On.* Penggunaan *Central Authentication Service* dan FreeRADIUS pada penelitian ini akan membuat suatu sistem *login* tunggal yang aman dan juga efisien. Pada penelitian ini SSO berbasis CAS digunakan karena memiliki beberapa keunggulan yaitu gratis, sistem yang aman, fleksibilitas, keandalan dan juga banyaknya *client libraries* [7].

Bahkan CAS pada versi 2.0 telah menerapkan teknologi yang membuat pengiriman paket data saat proses autentikasi sama sekali tidak menyebarkan *password* pada aplikasi pihak ketiga. Untuk membuat sistem ini lebih aman pada penelitian ini akan mengintegrasikan CAS dengan FreeRADIUS (*Free Remote Authentication Dial in User Service*) *Server* untuk digunakan sebagai *authentication Server.* RADIUS yang merupakan AAA (*authentication, Authorization and accounting*) protokol dinilai cukup andal dalam hal keamanan pada jaringan privat maupun publik dan juga manajemen administrasi pengguna.

Tujuan akhirnya penelitian ini memberikan opsi dalam mengembangkan sistem autentikasi yang efisien dari segi waktu karena menggunakan sistem login tunggal, efisien biaya karena menggunakan perangkat lunak berbasis *open source* serta sistem yang aman karena menggunakan fitur dari CAS yang tidak menyebarkan *password* pada aplikasi pihak ketiga.

II. PERANCANGAN SISTEM

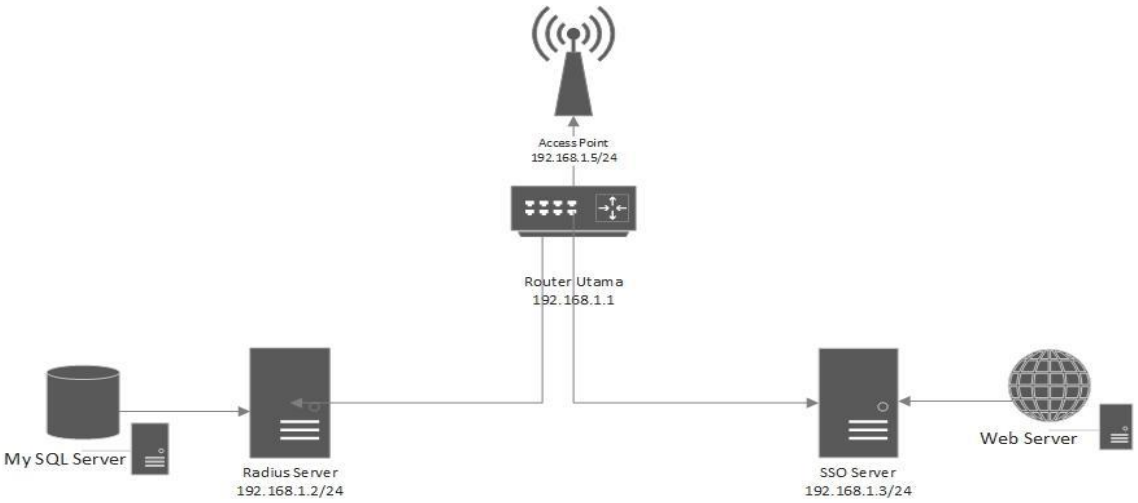
Sistem *Single-Sign-On* pada penelitian ini membutuhkan 4 buah *server*. Masing-masing akan dipasangkan pada 2 buah komputer *server*. Desain dibuat berdasarkan ketersedian perangkat yang dimiliki dan dibutuhkan. Untuk menguji keberhasilan implementasi sistem, maka akan dilakukan pengujian *basic path, fungsional, response time* dan *load testing.*

A. Rancangan Sistem Secara Umum

Gambar 1 menampilkan rancangan sistem dan konektivitas antar perangkat yang digunakan. Sistem yang dibuat pada penelitian ini menggunakan 4 buah *Server*, masing-masing *Server* yaitu *Server* Web, *Server* Database, *Server* CAS dan *Server* FreeRADIUS. Setiap *Server* akan saling terhubung dan berkomunikasi. *Server* Web digunakan sebagai *Server* yang mengandung semua aplikasi web pada perusahaan X, semua layanan aplikasi web berada pada *Server* ini.

Server CAS merupakan *Server* tersendiri yang bertindak sebagai SSO *Server*. Pada *Server* CAS administrator dapat melakukan konfigurasi untuk menambahkan dan mengurangi aplikasi yang menggunakan layanan CAS. Untuk dapat mengakses setiap aplikasi pengguna akan diarahkan ke *Server* CAS dahulu untuk melakukan proses *login*. *Server* CAS akan menampilkan portal halaman *login* yang nantinya data kredensial pengguna tersebut akan dikirimkan untuk dilakukan otentikasi oleh *Server* FreeRADIUS.

Server FreeRADIUS akan melakukan otentikasi dan pencocokan data pada *datastore*. Setiap aplikasi *server* tersebut akan dipasangkan dalam 2 komputer sebagai komputer *server*. Untuk mengetahui topologi dari sistem Gambar 1 merupakan topologi sistem :



Gbr.1 Topologi Sistem

Pengguna yang berhasil melakukan autentikasi akan langsung diarahkan ke halaman web yang diakses. Pengguna diberikan tiket yang didapatkan dari *Server* CAS sebagai tiket masuk ke aplikasi web dan aplikasi web akan mencocokkan tiket yang dimiliki pengguna ke *Server* CAS.

Apabila data yang dicocokkan benar, maka CAS akan memberikan informasi *username* pengguna pada web tersebut. CAS juga akan mengirimkan *cookie* ke browser pengguna yang membuat pengguna tidak perlu melakukan pengulangan autentikasi untuk dapat mengakses aplikasi lain.

Mekanisme *logout* yang dilakukan *user* pada aplikasi web akan diteruskan diteruskan ke *Server* CAS. Setelah berhasil maka *Server* CAS akan meneruskan ke halaman *logout* aplikasi tersebut. *Session* dan *cookie* lokal pada aplikasi web tersebut akan dimusnahkan. Pemusnahan ini akan membuat pengguna secara otomatis keluar dari setiap aplikasi yang ada.

B. Rancangan Pengujian

1) Pengujian *Basic Path*

Pengujian *Basic Path* adalah pengujian struktur program yang digunakan untuk efisiensi dan efektivitas pengujian *white box*, karena di dalam pengujian *white box* tidak mungkin seluruh jalur dieksekusi. Pengujian ini memungkinkan perancangan *test case* yang diturunkan dari *cyclomatic complexity* struktur program.

Ukuran dari *cyclomatic complexity* akan menjadi panduan dalam menentukan *basic path. Test case* yang dapat akan menjadi *test case* yang digunakan untuk membuktikan bahwa setiap *statement* dari program akan dieksekusi minimal sekali[7] Pengujian *basic path* yang digunakan pada penelitian ini difokuskan pada proses *login* SSO. Pengujian ini bertujuan untuk melihat *path-path* yang dilalui saat program dijalankan dan untuk mengetahui kompleksitas atau kerumitan dari proses *login* sistem SSO.

2) Pengujian Fungsional

Pengujian fungsional merupakan metode pengujian perangkat lunak yang digunakan untuk menguji perangkat lunak tanpa

mengetahui struktur internal kode atau program. Dalam pengujian ini, *tester* menyadari apa yang harus dilakukan oleh program tetapi tidak memiliki pengetahuan tentang bagaimana melakukannya. Pengujian fungsional yang digunakan pada penelitian ini adalah pengujian *output* dan *input* dari sistem *Single-Sign-On*. Pengujian ini ditujukan untuk mengetahui kebenaran *output* dari perintah yang dimasukkan.

3) Pengujian *Response Time*

Response Time Testing adalah pengujian waktu respon yang mengacu pada waktu yang diperlukan untuk satu *server* untuk menanggapi permintaan dari yang lain. Waktu respon dimulai ketika pengguna mengirim permintaan dan berakhir saat waktu aplikasi menyatakan bahwa permintaan tersebut telah selesai. Dalam pengujian ini perlu dipastikan bahwa pengguna situs/aplikasi mendapatkan respon dalam waktu yang dapat diterima[8].

4) Pengujian *Load Test*

Load testing software adalah alat evaluasi untuk menentukan bagaimana suatu aplikasi akan bekerja ketika tingkat kerja mendekati batas spesifikasi aplikasi. Tujuan utama *Load testing* (pengujian beban) adalah untuk menentukan jumlah maksimum pekerjaan yang dapat ditangani sistem tanpa penurunan kerja secara signifikan. Pengujian ini dilakukan untuk mengetahui kinerja *Server* SSO dalam menangani autentikasi dalam jumlah pengguna yang banyak dalam waktu bersamaan.

Pengujian *Load Test* dilakukan dengan melakukan proses autentikasi pada *Server* SSO. Komputer penguji (pengguna) dan *Server* SSO yang akan digunakan berada pada suatu *network* yang sama. Pengujian *load test* dilakukan dengan melakukan proses autentikasi pada *Server* SSO dengan jumlah 50,100 dan 200 *user* pada saat bersamaan. Target dari pengujian ini adalah 0 jumlah *error request* pada percobaan hingga 200 pengguna.

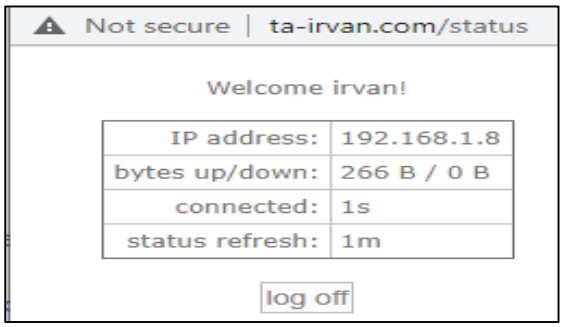
III. HASIL IMPLEMENTASI

Hasil implementasi dibagi menjadi 2 bagian yaitu hasil implementasi FreeRADIUS *Server* dan hasil implementasi SSO *server*.

A. Hasil Implementasi FreeRADIUS Server

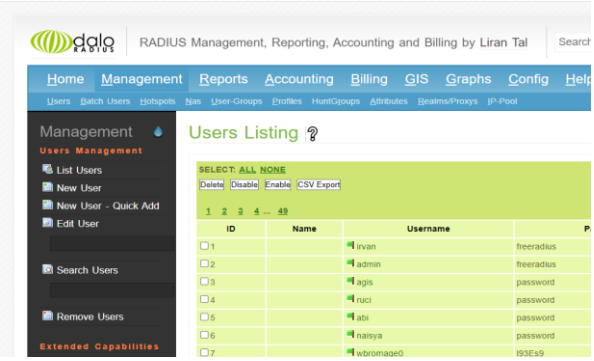
Server FreeRADIUS ditujukan khusus untuk dapat menunjang keamanan jaringan sistem SSO. Pada bagian keamanan jaringan *server* FreeRADIUS menggunakan IEEE protokol 802.1X yang menyediakan mekanisme autentikasi ke perangkat yang ingin terhubung ke suatu LAN atau WLAN. Pada penelitian ini FreeRADIUS sudah dapat menjalankan protokol 802.1X dengan baik. Gambar 2 Menunjukkan tampilan dari *login* pada suatu perangkat yang ingin terhubung ke dalam jaringan. Setelah pengguna berhasil memasukkan data kredensial yang tepat, maka *server* akan menampilkan status pengguna dalam jaringan tersebut yang dapat dilihat pada Gambar 3.

Gbr.2 Tampilan *Login Hotspot*

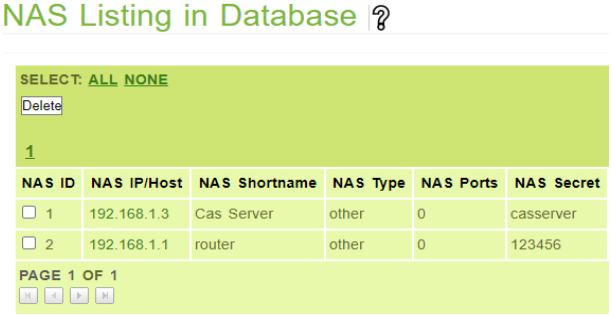


Gbr.3 Tampilan Status Pengguna

Manajemen *server* FreeRADIUS diperlukan untuk mempermudah manajemen pengguna dan klien atau NAS, pada penelitian ini dibuatlah sebuah web panel. Web panel yang digunakan adalah DaloRADIUS karena bersifat *open source.* Web panel akan terhubung langsung pada *database* FreeRADIUS. Gambar 4 merupakan tampilan daftar pengguna dari FreeRADIUS pada web DaloRADIUS dan Gambar 5 merupakan tampilan dari daftar klien atau NAS pada FreeRADIUS.



Gbr.4 Tampilan DaloRADIUS Daftar Pengguna

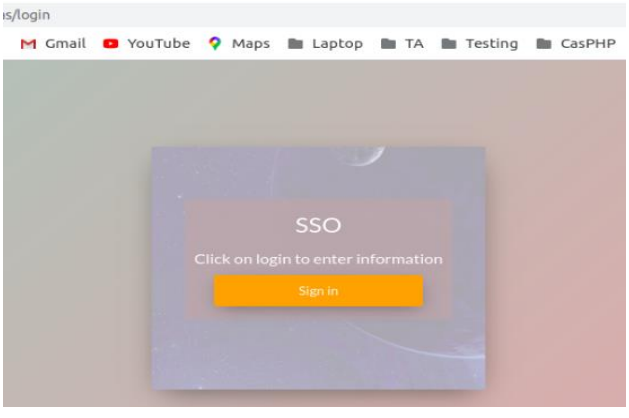


Gbr.5 Tampilan DaloRADIUS Daftar Klien

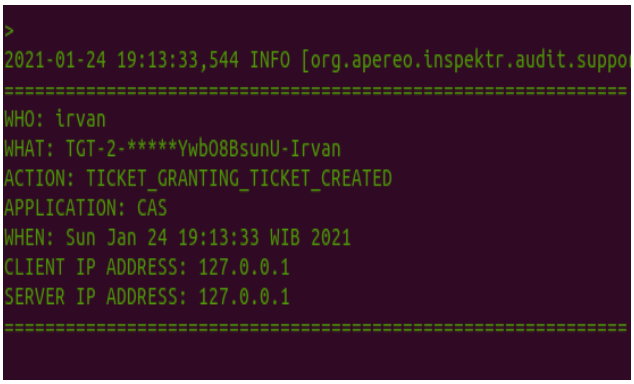
B. Hasil Implementasi SSO Server

SSO *server* menggunakan Apereo CAS telah berhasil digunakan. Hasil dari implementasi SSO adalah autentikasi SSO dapat dijalankan menggunakan *server* FreeRADIUS. Setiap user yang telah berhasil masuk pada *server* SSO akan dapat mengakses setiap web aplikasi yang telah dibuat tanpa melakukan autentikasi ulang. Apabila pengguna mencoba untuk mengakses web klien SSO tapi belum melakukan autentikasi sebelumnya maka pengguna akan langsung diarahkan ke halaman *login* CAS *server*. Halaman *login* CAS *server* sebelumnya telah di *customize* agar tidak terlihat seperti *default* dapat dilihat pada Gambar 6.

Pengguna yang mencoba *login* akan diarahkan FreeRADIUS *server*. Apabila autentikasi berhasil maka SSO *server* akan membuatkan *Ticket Granting Ticket* sebagai *session* SSO CAS. Gambar 7 menampilkan informasi dari *Ticket Granting Ticket* yang dibuat oleh *server* CAS yang berisikan nama pengguna dan ID tiket. Pada gambar dapat dilihat tidak ada pertukaran *password* antara CAS *server* dengan web aplikasi. CAS *server* hanya melakukan pertukaran data dengan FreeRADIUS *server*. Untuk validasi pengguna web klien akan diberikan oleh CAS *server*.



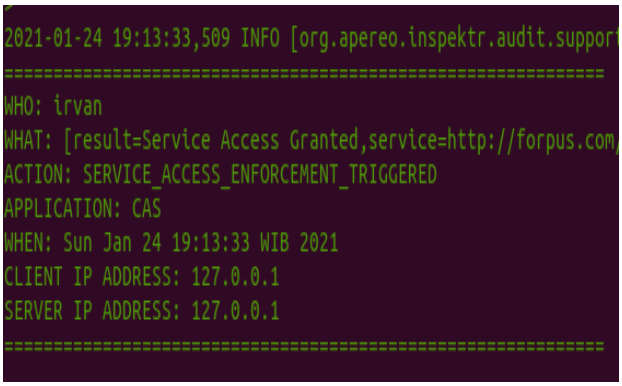
Gbr.6 Halaman *Login* SSO CAS



Gbr.7 Tampilan Informasi *Ticket Granting Cookie* Pada *Server* CAS

Apabila pengguna telah memiliki *Ticket Granting* dalam hal ini berhasil *login* ke SSO maka pengguna akan dapat mengakses web *server* tanpa melewati proses autentikasi lagi. Gambar 8 menampilkan proses pengecekan SSO CAS terhadap *user* yang telah berhasil *login* mencoba mengakses web "forpus.com".

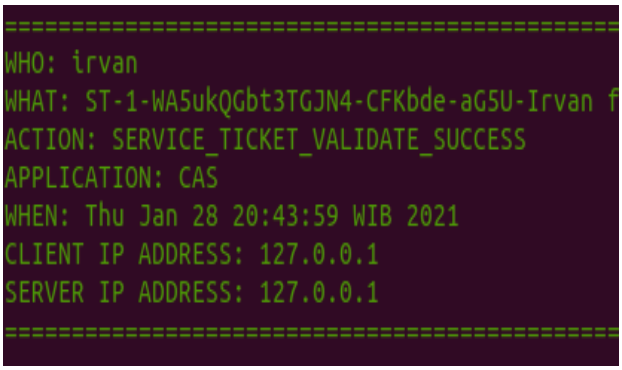
Pada Gambar 9 menampilkan pengguna dengan nama pengguna "irvan" telah diberikan akses untuk *service* web "forpus.com".



Gbr.8 Tampilan Proses Pengecekan Tiket Pengguna

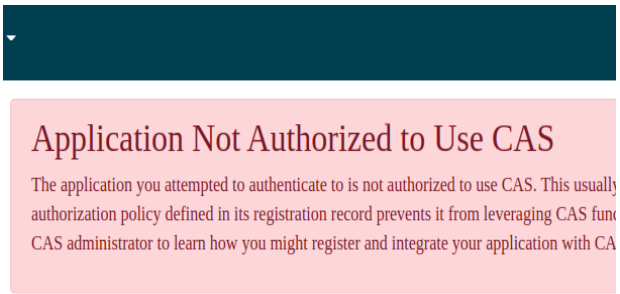
Pengguna yang telah diberikan akses ke aplikasi menggunakan TGT selanjutnya akan dibuatkan *service ticket* atau ST yang akan digunakan sebagai tiket untuk mengakses aplikasi tersebut. 1 servis tiket hanya dapat diakses oleh satu pengguna untuk satu aplikasi.

Berikut merupakan Validasi ST untuk user yang telah memiliki TGT atau berhasil *login* ke dalam SSO dapat dilihat pada Gambar 9.

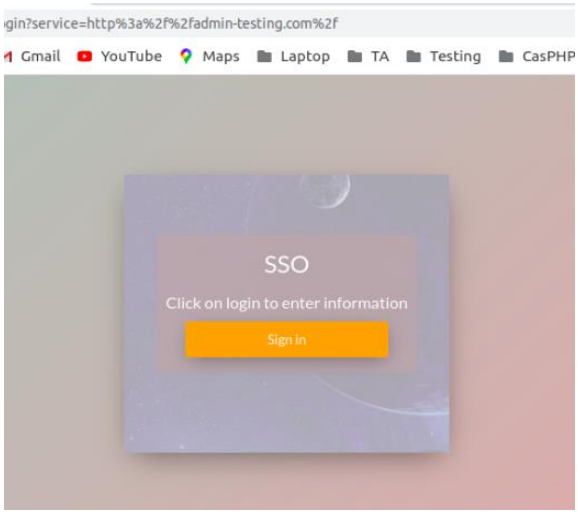


Gbr.9 Proses Validasi *Service Ticket*

Untuk mendapatkan layanan SSO CAS setiap web harus didaftarkan dan dibuatkan sebuah *Service Registry*. Web klien yang belum terdaftar ke dalam *service registry server* CAS dan mencoba untuk menggunakan modul autentikasi maka akan dilakukan pengecekan *service registry* dan akan ditampilkan sebagai "unauthorized application". Tampilan aplikasi yang belum terotorisasi oleh *server* CAS dapat dilihat pada Gambar 10. Aplikasi klien yang telah didaftarkan pada CAS *server* akan langsung diarahkan ke SSO *Server*. Gambar 11 merupakan tampilan aplikasi yang telah terdaftar sebagai klien CAS dapat dilihat pada URL aplikasi yang terdapat tulisan "?service= "nama aplikasi".



Gbr.10 Tampilan Aplikasi Yang belum Terotorisasi Oleh CAS



Gbr.11 Tampilan Login SSO Pada Aplikasi klien CAS

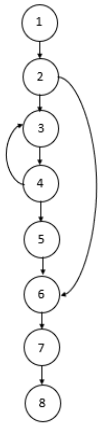
IV. HASIL PENGUJIAN

Berdasarkan rancangan pengujian yang telah di tentukan hasil dari pengujian akan menunjukan keberhasilan dari sistem yang telah diterapkan. Pengujian ini bertujuan untuk mendapatkan hasil dari alur sistem *login* SSO serta kemampuan *server* dalam melayani pengguna.

A. Hasil Pengujian Basic Path

Pengujian *basic path* akan dilakukan pada skenario *login* SSO saat mengakses suatu web klien SSO CAS. Pengujian ini ditujukan untuk melihat *path-path* yang dilalui saat menjalankan proses *login*. Gambar 12 akan menampilkan *flow graph* pada proses *login*. Berikut merupakan detail keterangan dari nomor-nomor pada *flow graph login* SSO:

1. Aplikasi web diakses
2. Pengecekan TGT
3. Proses *Login* SSO
4. Validasi data kredensial
5. *Redirect* aplikasi web
6. Validasi ST
7. Get Username
8. Halaman utama web



Gambar 1. *Flow graph login SSO*

Perhitungan *Cyclomatic Complexity*:

$V(G) = E - N + 2$

V(G) = 9 – 8 + 2

V(G) = 3

Nilai V(G) yang didapat adalah 3, dimana terdapat 3 jalur pengujian yang didapat berdasarkan perhitungan dari *Cyclomatic complexity*. Berdasarkan nilai dari *cyclomatic complexity* yang didapatkan maka jumlah *basic path* yang harus diidentifikasi adalah sebanyak 3 buah. Adapun jalur pengujian ini akan diterapkan pada pengujian fungsional yang diujikan berdasarkan *basic path* yang telah didapatkan.

B. Hasil Pengujian Fungsional

Hasil dari pengujian fungsional pada sistem *login* SSO telah berhasil dan memenuhi setiap *expect result* yang telah ditentukan. Pada pengujian *login* tanpa memiliki *ticket granting* pengguna akan diarahkan ke halaman *login* SSO dan hanya dapat memasukkan data kredensial yang benar. Sedangkan pengguna yang telah memiliki *ticket granting* berhasil untuk mengakses web aplikasi tanpa harus melakukan *login* terlebih dahulu. Pengguna

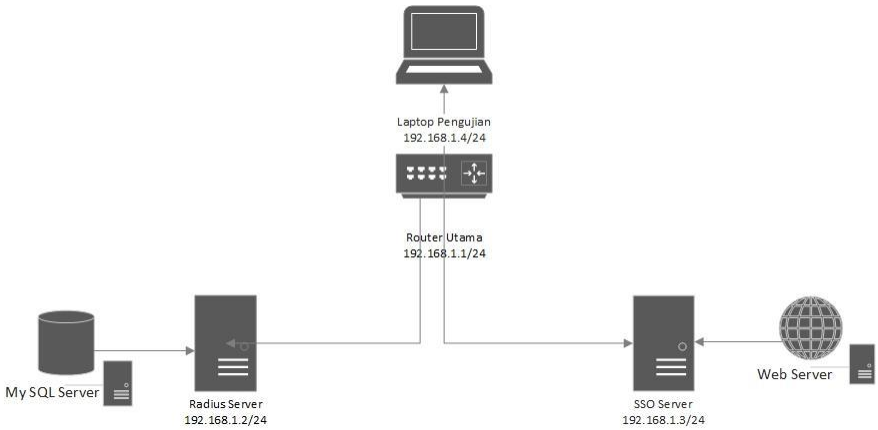
yang memiliki *ticket granting* yang sudah kadaluarsa juga tidak dapat mengakses web aplikasi klien CAS.

C. Performance Testing

Pengembangan suatu *server* web perlu untuk mengetahui kemampuan *server* dalam menangani pengguna. Salah satu metode yang digunakan adalah metode pengujian performance testing. Untuk mendapatkan hasil pengujian yang lebih baik maka digunakan sebuah tools khusus yang dibuat untuk menguji performa suatu web aplikasi ataupun database *server*. Tools yang digunakan pada penelitian ini merupakan Apache JMeter.

Aplikasi ini merupakan aplikasi open source berbasis JAVA yang dipergunakan untuk melakukan performance test. Pada penelitian ini pengujian akan dibagi menjadi 2 bagian, yaitu pengujian Response Time dan pengujian Load Test.

Pengujian dilakukan menggunakan sebuah komputer laptop yang berada di dalam jaringan yang sama dengan CAS dan RADIUS *server*. Untuk mengetahui lingkungan pengujian, topologi jaringan dari skema pengujian ini dapat dilihat pada Gambar 13.



Gbr.13 Topologi Pengujian

1) Hasil Pengujian Response Time Testing

Pengujian ini menggunakan dua metode GET dan POST untuk menguji proses login pada sistem SSO pada waktu yang bersamaan. Metode GET digunakan untuk mengakses halaman login SSO. Sedangkan metode POST digunakan untuk mengirimkan data kredensial pengguna yang di-export langsung dari database FreeRADIUS. Berikut merupakan hasil dari pengujian response time dengan sampel 1,2 dan 4 virtual user ditampilkan pada Tabel 1:

Tabel 1. Hasil Pengujian *Response Time*

Jumlah Pengguna	***Average Response Time* (ms)**
1 Pengguna	204
2 Pengguna	241
4 Pengguna	360

Berdasarkan Tabel 1 *Server* berhasil mencapai waktu respon yang diharapkan yaitu dibawah 2 detik. Bahkan dapat dikategorikan cepat dikarenakan berada pada interval 0,1-1 detik.

2) Hasil Pengujian *Load Testing*

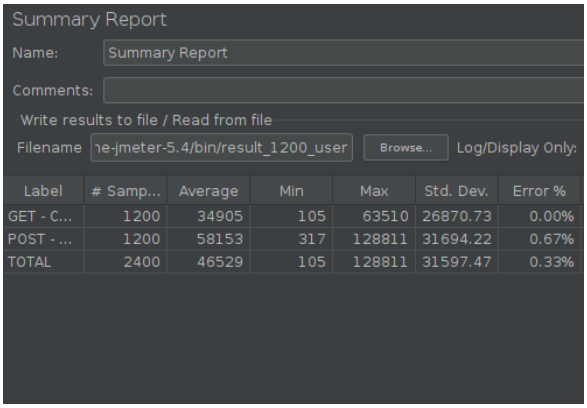
Pengujian *load testing* dibutuhkan untuk mengetahui berapa beban maksimal yang dapat ditangani oleh *server*. Sama dengan pengujian *response time*. Pengujian *Load Test* dibuat menggunakan metode GET untuk mengakses

halaman login dan POST untuk mengirimkan data kredensial. Pada pengujian ini akan diuji apakah *server* dapat menangani 50, 100 dan 200 *virtual user* mengakses *server* dalam satu waktu tanpa terjadinya *error*. Berdasarkan rancangan pengujian sebelumnya *server* berhasil untuk menangani hingga 200 *virtual user* tanpa terdapat *error*. Hasil pengujian dapat dilihat pada Tabel 2.

Tabel 2. Hasil Pengujian *Load Test*

Jumlah Pengguna	Average Response Time (s)	Error %
50	12.2	0.00
100	3.3	0.00
200	7.2	0.00

Untuk mengetahui beban maksimal yang dapat ditangani oleh *server* SSO CAS maka pengujian dilanjutkan dengan mencoba untuk menguji performa *server* menggunakan 400, 800, 1000 hingga 1200 *virtual user*. *Error* baru terdapat pada pengujian menggunakan 1200 *virtual user* terdapat error sebanyak 0.67% pada *method* POST atau 8 dari 1200 *user* dengan *response code* "java.net.SocketTimeoutException" dan pesan *error* "Read Time Out" ini disebabkan karena *server* tidak dapat merespon permintaan. Hasil dari pengujian 1200 *virtual user* terdapat pada Gambar 14.



Gbr.14 Pengujian 1200 *Sample User*

V. PENUTUP

Berdasarkan hasil implementasi dan pengujian penelitian Integrasi *Central Authentication Service* (CAS) dengan FreeRADIUS Sebagai Sistem *Single-Sign-On* (SSO), maka didapatkan kesimpulan sebagai berikut:

1. Sistem *Single-Sign-On* menggunakan *Central Authentication Service* dapat diterapkan menggunakan FreeRADIUS pada jaringan RADIUS. Metode ini juga dapat membuat *login* aplikasi web dan jaringan internet hanya menggunakan satu akun.
2. Setiap Aplikasi web yang diakses akan diarahkan ke *Single-Sign-On* CAS menggunakan konfigurasi PHP dan *Server* APACHE. Integrasi web aplikasi menggunakan modul phpCAS dan mod_auth_cas yang telah disediakan oleh Apereo CAS.
3. *Server* CAS dapat dihubungkan ke *Server* FreeRADIUS dengan mengganti *authentication handler* CAS menggunakan *dependency* RADIUS yang disediakan oleh Apereo CAS. Pada konfigurasi *Server* FreeRADIUS *server* CAS akan didefinisikan sebagai sebuah *Network Access Service* atau NAS.
4. SSO CAS dapat dijalankan pada protokol HTTPS pada port 8443 dengan menggunakan konfigurasi *Secure Socket Layer* atau SSL pada Apache Tomcat. Pada penelitian ini Sertifikat SSL yang digunakan masih menggunakan *Self Signed Certificate*.
5. Pengujian performa *server* yang dilakukan pada jaringan lokal menggunakan aplikasi JMeter menggunakan metode POST dan GET *Request* menghasilkan *avarage response time* sebesar 204 *milisecond* pada 1 *virtual user* dan dapat menangani 200 *virtual user* yang mencoba *login* dalam 1

waktu tanpa terjadinya error. Pengujian maksimal menggunakan 1200 *virtual user* mendapatkan *error* sebesar 0.67% atau 8 *virtual user* pada penelitian ini

REFERENSI

[1] APJII, "Laporan Survei Internet APJII 2019 – 2020," *Asos. Penyelenggara Jasa Internet Indones.*, vol. 2020, hal. 1–146, 2020, [Daring]. Tersedia pada: https://apjii.or.id/survei.

[2] T. J. Purba, Y. N. Kunang, dan A. Muzakir, "Pengembangan Sistem Otentikasi SSO dengan SAML Berbasis Radius," *Student Colloq. Sist. Inf. Tek. Inform.*, no. August 2015, hal. 13–17, 2015.

[3] M. H. Rehman, D. A. Govardhan, dan T. V. Narayana Rao, "Design and Implementation of RADIUS – An Network Security Protocol GJCST Computing Classification," vol. 10, no. October 2014, hal. 48–54, 2014.

[4] J. Somorovsky, A. Mayer, J. Schwenk, M. Kampmann, dan M. Jensen, "On breaking SAML: Be whoever you want to be," *Proc. 21st USENIX Secur. Symp.*, hal. 397–412, 2012.

[5] M. Y. A. Saputro, K. I. Satoto, dan A. F. Rochim, "Implementasi Sistem Single-Sign-on / Single-Sign Out Berbasis Central Authentication Service Protocol Pada Jarigan Lightweight Directory Access Protocol Universitas Diponegoro," vol. 1, no. 3, hal. 36, 2012.

[6] P. Aubry, V. Mathieu, dan J. Marchal, "Open-source Single-Sign-On with CAS (Central Authentication Service)," *Actes of EUNIS*, hal. 1318--1882, 2004.

[7] T. A. Kurniawan, "Pengujian Struktur Program Dengan Pengujian Jalur Dasar (Basis Path Testing) : Teori Dan Aplikasi," *Eeccis*, vol. 1, no. 1, hal. 29–32, 2007, [Daring]. Tersedia pada: http://jurnaleeccis.ub.ac.id/index.php/eeccis/article/viewFile/357/266.

[8] https://www.loadtestingtool.com, "Response Time." https://www.loadtestingtool.com/help/response-time.shtml (diakses Apr 25, 2021).