ANALISIS PERBANDINGAN UNJUK KERJA SISTEM AUTENTIKASI *SINGLE SIGN-ON* DENGAN PROTOKOL LDAP DAN RADIUS

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:

EUGENIUS TITO VALADDO MAHENDRA
NIM. 115060807111065

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2016

repository.ub.ac.id

UNIVERSITAS BRAWIJAYA

PENGESAHAN

ANALISIS PERBANDINGAN UNJUK KERJA SISTEM AUTENTIKASI *SINGLE SIGN-ON* DENGAN PROTOKOL LDAP DAN RADIUS

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:

EUGENIUS TITO VALADDO MAHENDRA
NIM. 115060807111065

Skripsi ini telah diuji dan dinyatakan lulus pada

15 Januari 2016

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Adhitya Bhawiyuga, S.Kom., M.S.	**Eko Sakti Pramukantoro, S.Kom., M.Kom.**
NIK. 201405 890720 1 1 001	**NIK. 860805 06 1 1 0252**

Mengetahui

Ketua Program Studi Informatika / Ilmu Komputer

Drs. Marji, M.T.

NIP. 19670801 199203 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata di dalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (Sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 15 Januari 2016

Eugenius Tito Valaddo M.

NIM. 115060807111065

KATA PENGANTAR

Puji syukur ke hadirat Tuhan Yang Maha Esa yang selalu memberikan rahmat dan bimbingan-Nya sehingga penulis dapat menyelesaikan penyusunan penelitian skripsi ini pada keminatan Jaringan Komputer dengan judul **"Analisis Unjuk Kerja Sistem Autentikasi *Single Sign-On* Dengan Protokol *Lightweight Directory Access Protocol* dan *Remote Authentication Dial-In User Service*"**.

Penyusunan skripsi ini tidak terlepas dari bimbingan dan bantuan dari berbagai pihak yang mendukung hingga akhir penulisan skripsi. Penulis mengucapkan terima kasih kepada mereka, yaitu:

1. Tuhan Yang Maha Esa atas rahmat dan anugerah-Nya sehingga skripsi ini dapat terselesaikan dengan baik dan lancar.
2. Joseph Sami Jananto dan Caecilia Nanik Pudjiati selaku orang tua penulis yang turut mendoakan dan memberi semangat sampai kelulusan kuliah.
3. Robert, Ronald, Merryza, Edoardus selaku kakak kandung penulis yang turut mendoakan dan memberi semangat sampai kelulusan kuliah.
4. Adhitya Bhawiyuga, S.Kom., M.S. dan Eko Sakti P., S.Kom., M.Kom. selaku Dosen Pembimbing Skripsi yang membimbing pengerjaan skripsi dari awal pengerjaan aplikasi sampai akhir penulisan laporan.
5. Aswin Suharsono, S.T., M.T. selaku Dosen Pembimbing Akademik yang selalu membimbing kegiatan perkuliahan dari awal masuk sampai lulus kuliah.
6. Teman-teman kelas I Informatika 2011 yang memberikan semangat dan doa, terutama Lingga, Heddy, Andhika, Irfandi, Rosikhan, Billy, Murtadho, Rio, Adi, Fikhi, Rizki, Alfian, Harris, Zein, Iffa, Ayu, Ami, Dyah, Rahma, dkk.
7. Teman-teman sesama pejuang skripsi keminatan Jaringan Komputer yang memberikan semangat dan doa, yaitu Ndhar, Nurtria, Pratiwi, Danny, Bayu, Raka, Adit, Ega, Rani, Zella, Gigih, Raymond.
8. Teman-teman dari keminatan lain yang memberikan semangat dan doa, serta mengerjakan skripsi bersama di lab, yaitu Zidna, Fikar, Ian, Demmy, Galang, Ratri, Anhar, Aristiawan, Dayat, Lazu.
9. Dan teman-teman lain yang belum saya sebutkan, yang turut memberikan semangat dan doa selama pengerjaan skripsi.

Akhir kata, penulis mengharapkan saran dan kritik yang membangun, sehingga skripsi ini dapat dijadikan kajian pustaka pada penelitian berikutnya.

Malang, Januari 2016

Penulis

valm.tito@gmail.com

ABSTRAK

Perkembangan teknologi di bidang jaringan komputer berkembang pesat untuk meningkatkan kenyamanan dan efisiensi dari penggunaan sumber daya bersama melalui internet, dengan aspek penting yang digunakan yaitu sistem autentikasi. Hal tersebut harus diimbangi dengan keamanan yang baik untuk meminimalkan resiko serangan, karena data yang terkirim ketika melakukan *login* merupakan data *credential* pengguna. Dengan banyaknya aplikasi jaringan yang ada, dibutuhkan sistem autentikasi yang handal dan aman serta mampu mengakses beberapa aplikasi jaringan cukup melakukan satu kali proses autentikasi sehingga lebih efisien.

Single Sign-On yaitu sistem yang mampu mengakses beberapa layanan dalam jaringan menggunakan satu akun pengguna dengan cara memusatkan proses autentikasi. Penelitian ini membandingkan dua protokol jaringan yaitu *Lightweight Directory Access Protocol* (LDAP) dan *Remote Authentication Dial-In User Service* (RADIUS). Pada *password* pengguna dienkripsi menggunakan kriptografi SHA256 dan aplikasi *web* autentikasi menggunakan protokol HTTPS.

Hasil pengujian antara LDAP dan RADIUS, perbedaan pemakaian CPU 0,22% dan pemakaian RAM 25,65 kB. Perbedaan *Throughput* 12,0 *req/sec* dan *Response Time* 0,37 sec. Pengujian *Stress Load* terdapat perbedaan signifikan yaitu 894,45 *milisec*. Ketika melakukan *sniffing* yang didapatkan yaitu nilai *hash*, karena nilai asli *password* sudah dikonversi pada *client-side*. Ketika melakukan *dictionary attack*, didapatkan hasil serangan tidak berhasil karena *password* disimpan pada *database* berupa nilai *hash* 32-bit (SHA256).

Kata kunci: autentikasi, *single sign-on*, LDAP, RADIUS, protokol.

repository.ub.ac.id

ABSTRACT

Technological developments in the field of computer networks rapidly growing to improve the comfort and efficiency use of resources shared via the Internet, with the important aspect that is used is the authentication system. This must be balanced with good security to minimize the risk of attack, because the sent data when doing login is the data credential. With many existing network applications, authentication system needs reliable and secure and be able to access to multiple network applications simply do once the authentication process so that more efficient.

Single Sign-On is a system that is capable of accessing multiple services in a network using single user account in centralized authentication process. This study compared two network protocols are Lightweight Directory Access Protocol (LDAP) and Remote Authentication Dial-In User Service (RADIUS). At the user's password is encrypted using SHA256 cryptographic and authentication web application using HTTPS protocol.

The test results between LDAP and RADIUS, a difference of CPU usage 0.22% and RAM usage 25.65 kB. Difference of Throughput 12.0 req/sec and Response Time 0.37 sec. Load Stress testing occur significant difference 894.45 milisec. When doing sniffing obtained hash value, because the value of the original password has been converted in the client-side. When performing a dictionary attack, showed the attack was unsuccessful because the password is stored in a database form 32-bit hash value (SHA256).

Keyword: authentication, single sign-on, LDAP, RADIUS, protocol.

UNIVERSITAS BRAWIJAYA

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINALITAS iii
KATA PENGANTAR iv
ABSTRAK v
ABSTRACT vi
DAFTAR ISI vii
DAFTAR TABEL ix
DAFTAR GAMBAR x
BAB 1 PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Tujuan Penelitian 2
1.4 Manfaat Penelitian 2
1.5 Batasan Masalah 2
1.6 Sistematika Penulisan 2
BAB 2 LANDASAN PUSTAKA 4
2.1 Kajian Pustaka 4
2.2 *Single Sign-On* (SSO) 4
2.3 *Lightweight Directory Access Protocol* (LDAP) 6
2.4 *Remote Authentication Dial-In User Service* (RADIUS) 7
2.5 *Hypertext Transfer Protocol Secure* (HTTPS) 9
2.6 *Secure Hash Algorithm* (SHA) 9
BAB 3 METODOLOGI 10
3.1 Tipe Penelitian 10
3.2 Strategi Penelitian 10
3.2.1 Studi Literatur 10
3.2.2 Perancangan Sistem 10
3.2.3 Peralatan Pendukung 11
3.2.4 Implementasi Sistem 12
3.2.5 Pengujian dan Analisis 13

3.2.6 Kesimpulan dan Saran 13
BAB 4 PERANCANGAN 14
4.1 Rancangan Sistem LDAP dan RADIUS 14
4.2 Konfigurasi Aplikasi *Web* Accounts 16
BAB 5 IMPLEMENTASI 19
5.1 Persiapan Instalasi dan Konfigurasi *Server* 19
5.1.1 Instalasi dan Konfigurasi LAMP 19
5.1.2 Instalasi dan Konfigurasi LDAP 20
5.1.3 Instalasi dan Konfigurasi RADIUS 22
5.2 Pengisian Data Pengguna Baru 23
5.2.1 Pengisian Data pada LDAP 23
5.2.2 Pengisian Data pada RADIUS 24
BAB 6 PENGUJIAN 26
6.1 Pengujian Awal Autentikasi 26
6.2 Pengujian Kinerja 29
6.2.1 Hasil Pengujian Kinerja Pemakaian CPU dan RAM *Server* 33
6.2.2 Hasil Pengujian Kinerja *Throughput* dan *Response Time* 34
6.2.3 Hasil Pengujian Kinerja *Stress Load* 35
6.3 Pengujian Keamanan 36
6.3.1 Hasil Pengujian Keamanan Data *Sniffing* 38
6.3.2 Hasil Pengujian Keamanan *Dictionary Attack* 38
BAB 7 PENUTUP 40
7.1 Kesimpulan 40
7.2 Saran 40
DAFTAR PUSTAKA 41

DAFTAR TABEL

Tabel 4.1 Potongan *source code index* aplikasi *web* Accounts 16
Tabel 4.2 Potongan *source code result* aplikasi *web* Accounts pada LDAP 17
Tabel 4.3 Potongan *source code result* aplikasi *web* Accounts pada RADIUS..... 17
Tabel 6.1 Perbedaan fungsi autentikasi antara LDAP dan RADIUS 26
Tabel 6.2 *Stress Load* autentikasi LDAP dengan Python 31
Tabel 6.3 *Stress Load* autentikasi RADIUS dengan Python 32

DAFTAR GAMBAR

Gambar 2.1 Proses autentikasi *Single Sign-On* secara umum 5
Gambar 2.2 Google dengan aplikasi *web*-nya menggunakan sistem SSO 6
Gambar 2.3 Contoh struktur *Directory Information Tree* 6
Gambar 2.4 Proses autentikasi pada LDAP 7
Gambar 2.5 Proses autentikasi pada RADIUS 8
Gambar 2.6 Contoh sertifikat HTTPS Google yang muncul di browser 9
Gambar 3.1 Rancangan sistem *Single Sign-On* yang dibangun 11
Gambar 4.1 Diagram alir sistem *Single Sign-On* yang dibangun 14
Gambar 4.2 Rancangan sistem menggunakan LDAP 15
Gambar 4.3 Rancangan sistem menggunakan RADIUS 15
Gambar 5.1 Konfigurasi SSL pada Apache Server 21
Gambar 5.2 Tampilan awal phpLDAPadmin sebelum login 24
Gambar 5.3 Tampilan phpLDAPadmin saat melihat daftar pengguna 24
Gambar 5.4 Tampilan awal daloRADIUS sebelum login 25
Gambar 5.5 Tampilan daloRADIUS saat melihat daftar pengguna 25
Gambar 6.1 Tampilan awal aplikasi Accounts 27
Gambar 6.2 Tampilan Accounts ketika gagal *login* 27
Gambar 6.3 Tampilan Accounts ketika berhasil *login* 28
Gambar 6.4 Tampilan aplikasi *web* sederhana Rakit 28
Gambar 6.5 Tampilan aplikasi *web* sederhana Sikapen 28
Gambar 6.6 Mengatur jumlah *user* dan batas rentang waktu 29
Gambar 6.7 Mengisi IP *Address Server* 29
Gambar 6.8 Pengaturan untuk menghapus *auth* pada tiap iterasi 29
Gambar 6.9 Pengaturan untuk menghapus *cookies* pada tiap iterasi 30
Gambar 6.10 Mengisi *link* target dan variabel *credential* pengguna 30
Gambar 6.11 Mengukur pemakaian CPU dan RAM pada *server* 30
Gambar 6.12 Tampilan *Stress Load* dengan *multithreading* Python 33
Gambar 6.13 Pemakaian CPU pada *server* (*percent*) 33
Gambar 6.14 Pemakaian RAM pada *server* (*kiloBytes*) 34
Gambar 6.15 *Throughput* pada aplikasi *web* (*request/second*) 34
Gambar 6.16 *Response Time* pada aplikasi *web* (*second*) 35
Gambar 6.17 *Stress Load* pada protokol *server* (*milisecond*) 35
Gambar 6.18 *Request* HTTP *history* yang akan dikirim pada *Intruder* 37
Gambar 6.19 Parameter yang digunakan pada *Intruder* 37
Gambar 6.20 *Sniffing* ketika *login* pada aplikasi *web* Accounts 38
Gambar 6.21 *Dictionary attack* pada aplikasi *web* Accounts 39

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi terutama di bidang jaringan komputer sangat berkembang pesat untuk meningkatkan kenyamanan dan efisiensi dari penggunaan sumber daya bersama melalui internet, dengan salah satu aspek penting yang digunakan yaitu sistem autentikasi jaringan. Hal tersebut harus diimbangi dengan keamanan yang baik untuk meminimalkan resiko serangan yang terjadi, karena data yang terkirim pada saat melakukan proses autentikasi merupakan data *credential* pengguna. Dengan banyaknya aplikasi jaringan yang ada sekarang ini dibutuhkan sebuah sistem autentikasi jaringan yang handal dan aman, dan mampu mengakses ke beberapa aplikasi jaringan cukup melakukan satu kali proses autentikasi sehingga lebih efisien.

Single Sign-On (SSO) merupakan teknologi yang mampu mengakses beberapa layanan dalam jaringan cukup menggunakan satu akun pengguna dengan cara memusatkan proses autentikasi (Ade, 2009). Sistem SSO ini sudah sukses diimplementasikan secara luas oleh perusahaan internasional Google, yang memiliki banyak aplikasi berbasis web seperti GMail, Youtube, GDrive, dan yang lain. Pengguna cukup melakukan satu kali *login* dan otomatis mendapatkan hak akses ke semua aplikasi *web* milik Google tersebut.

Penelitian yang terkait dengan SSO menggunakan protokol LDAP, yaitu *sniffing* dapat digunakan untuk mendapatkan *username* dan *password* yang digunakan untuk *login* ke *server* yang terintegrasi dengan SSO menggunakan LDAP. Disarankan untuk meningkatkan keamanan pada jaringan, misalnya dengan implementasi protokol lain. Perlu dilakukan penelitian lebih lanjut dalam mendapatkan *username* dan *password* (Amarudin, 2014).

Kemudian penelitian yang terkait dengan SSO menggunakan protokol RADIUS, yaitu bahwa proses optimalisasi RADIUS sebagai sistem autentikasi dan otorisasi untuk proses *login* multi aplikasi *web* membuat pengguna hanya memiliki satu akun tunggal untuk beberapa aplikasi yang berbeda. Pada *password* pengguna sudah menggunakan metode CHAP, membuat nilai *random* yang disebut *challenge* dan diubah menjadi MD5 *hash* lalu dikombinasikan pada *challenge* dengan *password* (Yuliansyah, 2011).

Pada penelitian ini dilakukan analisis terhadap dua protokol yaitu LDAP dan RADIUS pada sistem autentikasi *Single Sign-On*. Untuk mendukung dari sisi keamanan, dirancang satu aplikasi *web* khusus untuk proses autentikasi yang menggunakan protokol HTTPS, serta melakukan enkripsi pada *password* menggunakan algoritma *hash* SHA256 pada *client-side*.

1.2 Rumusan Masalah

Berdasarkan pada latar belakang, terdapat beberapa rumusan masalah dalam penelitian ini, antara lain:

1. Bagaimana merancang sistem autentikasi *Single Sign-On* menggunakan protokol LDAP dan RADIUS yang aman?
2. Bagaimana perbandingan kinerja dari protokol LDAP dan RADIUS?

1.3 Tujuan Penelitian

Pada penulisan penelitian tugas akhir ini, terdapat beberapa tujuan yang terkait, antara lain:

1. Membuat perancangan sistem autentikasi *Single Sign-On* yang aman.
2. Melakukan analisis perbandingan cara kerja dari protokol autentikasi LDAP dan RADIUS pada sistem *Single Sign-On*.

1.4 Manfaat Penelitian

Berdasarkan sistem yang akan dirancang ini, memiliki beberapa manfaat dari sudut pandang yang berbeda, antara lain:

1. Bagi pembaca: Memahami proses sistem autentikasi *Single Sign-On* yang aman, serta mengetahui hasil analisis sistem yang didapatkan.
2. Bagi penulis: Mampu melakukan perancangan dan implementasi protokol LDAP dan RADIUS pada sistem autentikasi *Single Sign-On*.

1.5 Batasan Masalah

Mengacu pada permasalahan yang sudah dirumuskan sebelumnya, ada beberapa batasan masalah yang ada, antara lain:

1. Penelitian ini merupakan simulasi proses autentikasi *Single Sign-On* yang dilakukan menggunakan *prototype* aplikasi berbasis *web*.
2. Penelitian ini fokus pada dua protokol jaringan LDAP dan RADIUS yang digunakan pada sistem autentikasi *Single Sign-On*.

1.6 Sistematika Penulisan

Secara garis besar sistematika penulisan ini bertujuan menguraikan dan menggambarkan detil penyusunan secara keseluruhan, antara lain:

BAB 1 PENDAHULUAN

Berisi tentang penjelasan latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, dan sistematika penulisan, sehingga sesuai dengan topik penelitian yang akan dibahas.

BAB 2 LANDASAN PUSTAKA

Membahas berbagai landasan teori dasar yang diambil dari kajian pustaka dan referensi literatur pendukung yang terkait dengan *Single Sign-On*, LDAP, RADIUS, serta aplikasi pendukung lain yang digunakan.

BAB 3 METODOLOGI

Menjelaskan tentang tipe penelitian, perancangan, peralatan pendukung, dan implementasi sistem yang digunakan pada penelitian *Single Sign-On*.

BAB 4 PERANCANGAN

Menjelaskan tentang tahap perancangan sistem *Single Sign-On* menggunakan protokol LDAP dan RADIUS, serta konfigurasi aplikasi *web* autentikasi.

BAB 5 IMPLEMENTASI

Menjelaskan tentang tahap implementasi sistem dari perancangan sebelumnya, serta aplikasi *web* yang digunakan.

BAB 6 PENGUJIAN

Membahas proses pengujian sistem dari awal hingga akhir untuk mengetahui bagaimana kinerja dan keamanan dari sistem autentikasi *Single Sign-On* yang sudah dirancang dan diimplementasikan sebelumnya.

BAB 7 PENUTUP

Bab terakhir sebagai penutup laporan berisi tentang kesimpulan yang didasarkan atas pengujian serta analisis yang telah dilakukan, dan berisi saran supaya sistem tersebut dapat dikembangkan untuk penelitian berikutnya.

BAB 2 LANDASAN PUSTAKA

Pada bab ini akan membahas landasan teori serta kajian pustaka yang berkaitan dengan penelitian ini. Landasan pustaka membahas berbagai macam teori yang diperlukan untuk menyusun penelitian yang dilakukan penulis dan menjadi acuan sebagai usulan penelitian berikutnya. Kajian pustaka ini terkait dengan sistem autentikasi terpusat menggunakan sistem *Single Sign-On*.

2.1 Kajian Pustaka

Dalam kajian pustaka ini akan membahas secara singkat penelitian yang dilakukan penulis dengan penelitian yang sudah ada sebelumnya, yaitu sistem autentikasi terpusat *Single Sign-On* dengan masing-masing menggunakan protokol jaringan LDAP dan RADIUS.

Penelitian yang terkait dengan SSO menggunakan protokol LDAP, yaitu *sniffing* dapat digunakan untuk mendapatkan *username* dan *password* yang digunakan untuk *login* ke *server* yang terintegrasi dengan SSO menggunakan LDAP. Disarankan untuk meningkatkan keamanan pada jaringan, misalnya dengan implementasi protokol lain. Perlu dilakukan penelitian lebih lanjut dalam mendapatkan *username* dan *password* (Amarudin, 2014).

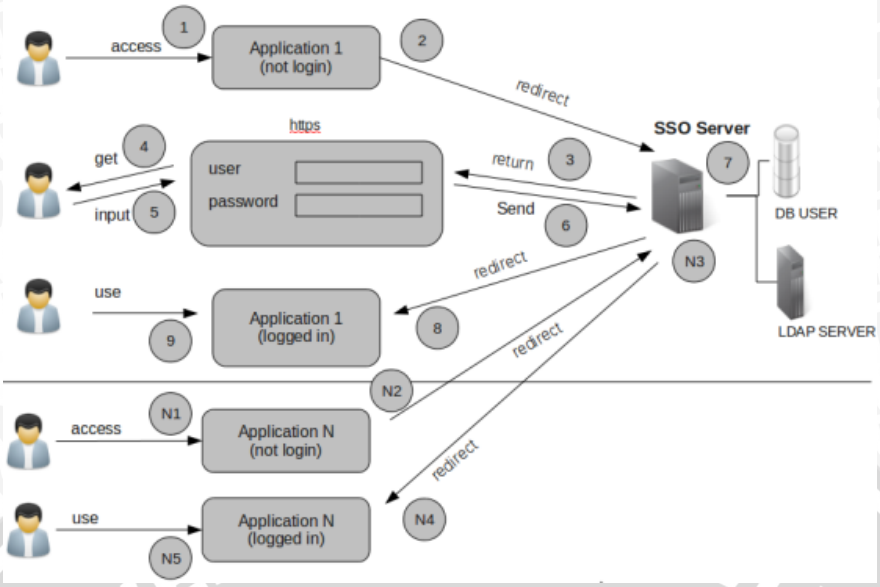
Kemudian penelitian yang terkait dengan SSO menggunakan protokol RADIUS, yaitu bahwa proses optimalisasi RADIUS sebagai sistem autentikasi dan otorisasi untuk proses *login* multi aplikasi *web* membuat pengguna hanya memiliki satu akun tunggal untuk beberapa aplikasi yang berbeda. Pada *password* pengguna sudah menggunakan metode CHAP, membuat nilai *random* yang disebut *challenge* dan diubah menjadi MD5 *hash* lalu dikombinasikan pada *challenge* dengan *password* (Yuliansyah, 2011).

2.2 *Single Sign-On* (SSO)

Single Sign-On merupakan teknologi yang mampu mengakses beberapa layanan dalam jaringan cukup menggunakan satu akun pengguna dengan cara memusatkan proses autentikasi (Ade, 2009). Pengguna hanya melakukan satu kali proses autentikasi untuk mendapatkan izin akses terhadap semua layanan aplikasi meskipun berpindah aplikasi lain, dengan ketentuan aplikasi yang bersangkutan sudah terhubung dengan sistem SSO (Patil, 2013).

Pada proses autentikasi, terdapat tiga hal penting yang harus dimiliki oleh pengguna, antara lain (Machado, 2010):

a. Sesuatu yang diketahui pengguna, seperti *password* atau PIN.
b. Sesuatu yang dimiliki pengguna, seperti token atau *software certificate*.
c. Sesuatu yang berada di pengguna, seperti sidik jari atau retina.



Gambar 2.1 Proses autentikasi *Single Sign-On* secara umum
Sumber: Hilmi (2012)

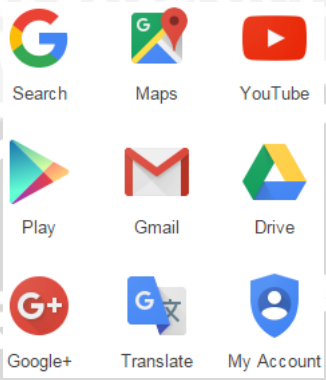
Sistem SSO memiliki beberapa keuntungan yang tidak dimiliki oleh sistem autentikasi biasa, antara lain (Ade, 2009):

a. Pengguna bisa berpindah ke aplikasi lain tanpa perlu *login* lagi.
b. Tidak perlu mengingat atau membuat banyak akun yang sama di setiap aplikasi yang berbeda.
c. Menghemat waktu karena hanya membutuhkan satu kali proses autentikasi.
d. Menyediakan *login form* yang dapat diakses terhubung ke semua aplikasi.
e. Menghemat biaya untuk pemeliharaan *password* secara signifikan.
f. Memungkinkan kontrol dan pengelolaan sumber daya yang lebih baik.

Hal-hal di atas menyatakan bahwa penggunaan SSO meningkatkan kenyamanan dan efisiensi. Namun di sisi lain menimbulkan kelemahan dalam hal keamanan, antara lain (Findlay, 2011):

a. Pentingnya kesadaran pengguna untuk merahasiakan data *credential* dan menjaga keadaan *login* pada aplikasinya.
b. Kerumitan mengimplementasikan sistem SSO ke dalam sebuah jaringan yang heterogen dan *multiplatform*, sehingga banyak pengelola layanan jaringan tidak begitu giat dalam melakukan implementasi.
c. Jika data *credential* sistem pengelola layanan jaringan diketahui oleh orang yang tidak berhak, maka orang tersebut dapat melakukan perubahan terhadap semua data yang ada di dalam sistem.
d. Sistem ini dapat menjadi suatu titik kegagalan tunggal (*single point failure*) bila tidak dirancang dengan baik karena setiap layanan aplikasi bergantung kepada sistem *Single Sign-On*.

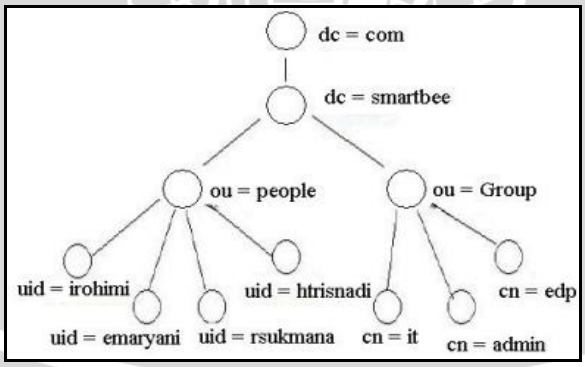
Sistem *Single Sign-On* ini sudah sukses diimplementasikan secara luas oleh perusahaan internasional Google. Pengguna cukup melakukan satu kali *login* dan otomatis mendapatkan hak akses ke semua aplikasi web milik Google tersebut. Sistem autentikasi Google dipusatkan pada satu *login page* dengan link https://accounts.google.com.



Gambar 2.2 Google dengan aplikasi *web*-nya menggunakan sistem SSO

2.3 *Lightweight Directory Access Protocol* (LDAP)

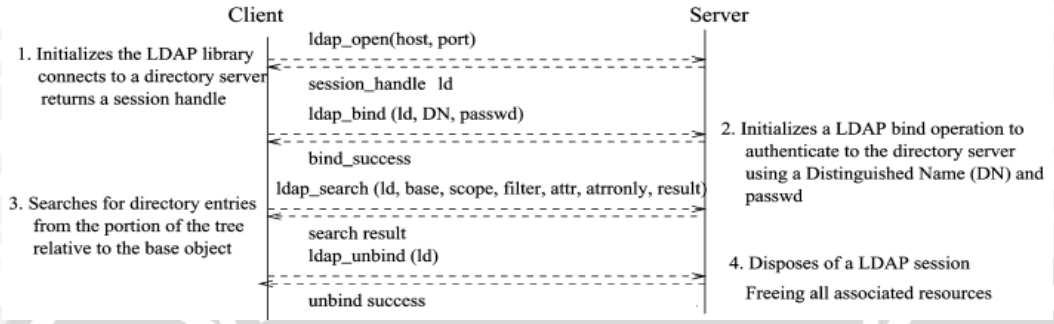
LDAP adalah sebuah protokol kelas ringan untuk mengakses protokol X.500 *Directory Service* dan berjalan melalui protokol TCP/IP. Protokol LDAP membentuk sebuah direktori berisi hierarki pohon yang memiliki cabang, disebut sebagai *Directory Information Tree* (DIT). Informasi pada LDAP disimpan dalam suatu data yang memiliki beberapa atribut. Jika pada *relational database* memiliki *primary key* untuk membedakan antar data, maka pada LDAP memiliki *Distinguished Name* (DN) yang bernilai unik untuk tiap data. DN didapat dengan mengurutkan lokasi data dari *root* DIT sampai pada *directory* data tersebut (Widiharso, 2011).



Gambar 2.3 Contoh struktur *Directory Information Tree*
Sumber: Hilmi (2012)

DN terpisah dengan tanda koma yang menandakan tiap *directory* dalam DIT, urutan dibaca dari kanan ke kiri, contoh: cn=admin, ou=group, dc=smartbee, dc=com. Secara umum, DIT terdiri dari *Domain Component* (dc), *Organizational Unit* (ou), *Unique Identification Number* (uid), dan *Common Name* (cn). Untuk *uid* dan *cn* sama saja, hanya merupakan nama lain (Obimbo, 2011).

LDAP mengakses *Directory Service* yang digunakan untuk menyimpan, mengorganisir, dan menyediakan akses ke informasi dalam sebuah direktori. Terdapat perbedaan *Directory Service* dengan *Relational Database* yaitu penggunaan DS lebih ke proses pembacaan data daripada penulisan data. Karena DS lebih sederhana hanya menyimpan data *username*, *password*, dan beberapa data penting saja, serta tidak mendukung proses transaksi rumit yang biasa ditemukan pada *Relational Database* seperti MySQL (Widiharso, 2011).



Gambar 2.4 Proses autentikasi pada LDAP
Sumber: Yuliansyah (2011)

2.4 *Remote Authentication Dial-In User Service* (RADIUS)

RADIUS merupakan sebuah protokol *access-control* yang melakukan verifikasi dan autentikasi pengguna berdasarkan pada metode *challenge / response*, termasuk di mana autentikasi terpusat, pengatur otorisasi, dan rincian *accounting* pengguna yang diperlukan. Dikembangkan Livingston Enterprises pertengahan tahun 1990, kemudian dibeli Lucent Technology (IETF, 2014).

RADIUS memiliki fungsi utama yaitu melakukan autentikasi pengguna sebelum mengakses ke jaringan, memberikan otorisasi pengguna untuk beberapa layanan jaringan tertentu, dan mencatat pemakaian penggunaan layanan pengguna tersebut. Beberapa keunggulan yang dimiliki, yaitu:

a. Bersifat *open-source* dan *scalable.*
b. Pemisahan proses antara keamanan dan komunikasi data.
c. Mudah beradaptasi dengan banyak sistem keamanan.

AAA singkatan dari *Authentication, Authorization, and Accounting*, yang mendefinisikan sebuah arsitektur yang melakukan autentikasi dan memberikan otorisasi dan mencatat aktivitas pengguna. Namun bisa hanya menggunakan sebagian dari sistem AAA tersebut, seperti hanya melakukan autentikasi dan otorisasi pengguna namun mengabaikan aktivitas pengguna tanpa *accounting*, ataupun bisa juga sebaliknya (IETF, 2014). RADIUS merupakan salah satu protokol AAA yang sebagian besar sering digunakan, terdapat pesaingnya yaitu TACACS+ dan Kerberos. Namun yang membuat RADIUS lebih baik daripada protokol AAA yang lain yaitu vendor dari RADIUS bersifat independen, tidak dikontrol oleh vendor tunggal. Hal tersebut berlawanan dengan TACACS+ (Cisco) dan Kerberos (Merit) (Yuliansyah, 2011).

Paket data yang dikirim melalui protokol RADIUS dienkapsulasi, yang terdiri dari 5 bagian penting, antara lain (Dekok, 2015):

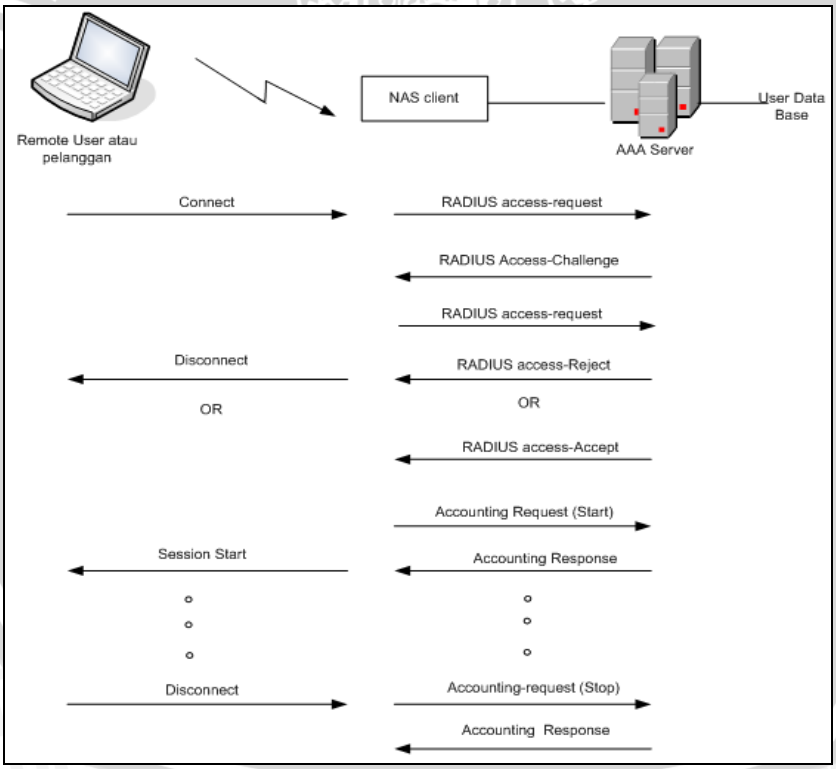
a. *Code*, digunakan untuk membedakan tipe pesan RADIUS yang dikirimkan pada paket dan memiliki ukuran satu *octet*, yaitu: (1) *Access-Request*, (2) *Access-Accept*, (3) *Access-Reject*, (4) *Accounting-Request*, (5) *Accounting-Response*, (11) *Access-Challenge*, (12) *Status-Server*, (13) *Status-Client*, (255) *Reserved*.

b. *Identifier*, untuk melakukan identifikasi permintaan dan memiliki ukuran satu *octet*. Jika permintaan cocok maka akan terjadi *Access Request*.

c. *Length*, memberikan informasi mengenai panjang paket data dan memiliki ukuran dua *octet*.

d. *Authenticator*, digunakan untuk membuktikan balasan dari RADIUS Server dan digunakan untuk algoritma *password*, memiliki ukuran 16 *octet*.

e. *Attributes*, berisi informasi yang dibawa pesan RADIUS, setiap pesan dapat membawa satu atau lebih atribut. Contoh: nama pengguna, *password*, CHAP-*password*, alamat IP *Access Point* (AP), atau pesan balasan.

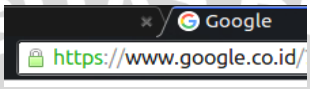


Gambar 2.5 Proses autentikasi pada RADIUS
Sumber: Dekok (2015)

Pengguna melakukan koneksi mengirimkan kode *Access-Request* ke *server*. Jika autentikasi sukses maka *server* membalas *Access-Accept*, jika gagal maka mendapat *Access-Reject*, dan jika dibutuhkan tambahan data maka mendapat *Access-Challenge* yang harus mengirim *request* ulang. Untuk kode *Accounting* boleh digunakan ataupun tidak.

2.5 *Hypertext Transfer Protocol Secure* (HTTPS)

HTTPS adalah protokol untuk komunikasi yang aman melalui jaringan komputer dan banyak digunakan di Internet. HTTPS terdiri dari komunikasi melalui HTTP dalam koneksi terenkripsi oleh Transport Layer Security (TLS) atau pendahulunya, Secure Sockets Layer (SSL). Fungsi HTTPS yaitu untuk melindungi privasi dan integritas data dalam komunikasi jaringan seperti proses autentikasi pada website yang dikunjungi. HTTPS menyediakan autentikasi *website* dan komunikasi yang terjadi untuk melindungi dari serangan *man-in-the-middle* (MITM). Selain itu juga menyediakan enkripsi komunikasi dua arah antara *client* dan *server* yang melindungi terhadap *eavesdropping* atau merusak isi data. Hal tersebut memastikan bahwa isi data dari komunikasi antara *client* dan *server* tidak dapat dibaca oleh pihak ketiga (Netscape, 2015).



Gambar 2.6 Contoh sertifikat HTTPS Google yang muncul di browser

HTTPS Uniform Resource Identifier (URI) memiliki sintaks identik dengan skema HTTP standar. Namun, HTTPS menginstruksikan *browser* untuk menggunakan lapisan enkripsi tambahan SSL/TLS untuk melindungi lalu lintas jaringan. *Web browser* membaca situs HTTPS berdasarkan otoritas sertifikat yang datang dari *server* sebelum melakukan instalasi pada *browser* milik *client*.

2.6 *Secure Hash Algorithm* (SHA)

Secure Hash Algorithm merupakan kelompok fungsi *hash* kriptografi yang diterbitkan oleh National Institute of Standards and Technology (NIST). SHA-2 adalah satu set fungsi *hash* kriptografi yang dirancang oleh National Security Agency (NSA). Sebuah aspek kunci dari algoritma *hash* kriptografi adalah bersifat *collision resistance*, tidak mungkin menemukan dua nilai input berbeda yang menghasilkan keluaran *hash* yang sama. Algoritma *hash* merupakan operasi matematika yang dijalankan pada data *digital* dengan membandingkan nilai "*hash*" yang dihitung (*output* dari eksekusi algoritma) menjadi nilai hash yang diketahui. Berikut jenis *hash* yaitu (NSA, 2015):

a. SHA-0: Fungsi yang diterapkan pada versi asli dari fungsi *hash* 160-bit yang diterbitkan pada tahun 1993 dengan nama "SHA".
b. SHA-1: Fungsi *hash* 160-bit menyerupai algoritma MD5 sebelumnya, dirancang oleh NSA untuk menjadi bagian dari Digital Signature Algorithm.
c. SHA-2: Gabungan dari dua fungsi *hash* yang sama, dengan ukuran blok yang berbeda, dikenal sebagai SHA-256 (32-bit) dan SHA-512 (64-bit).
d. SHA-3: Fungsi *hash* yang disebut Keccak dipilih tahun 2012, mendukung panjang hash yang sama seperti SHA-2, namun struktur internal berbeda secara signifikan dari keluarga SHA.

BAB 3 METODOLOGI

Pada bab ini akan membahas lebih lanjut mengenai tipe penelitian, perancangan, peralatan pendukung, dan metode yang digunakan pada penelitian *Single Sign-On* sehingga akan sesuai dengan kaidah penelitian.

3.1 Tipe Penelitian

Penelitian ini menggunakan tipe implementatif perancangan karena merupakan kegiatan penelitian dengan melakukan implementasi protokol LDAP dan RADIUS, serta merancang *prototype* aplikasi *web* autentikasi Accounts untuk membuat simulasi proses sistem *Single Sign-On*.

3.2 Strategi Penelitian

Strategi penelitian menjelaskan langkah-langkah yang digunakan dalam proses penelitian dari awal hingga akhir. Berikut ini merupakan urutan langkah-langkah yang digunakan dalam penelitian:

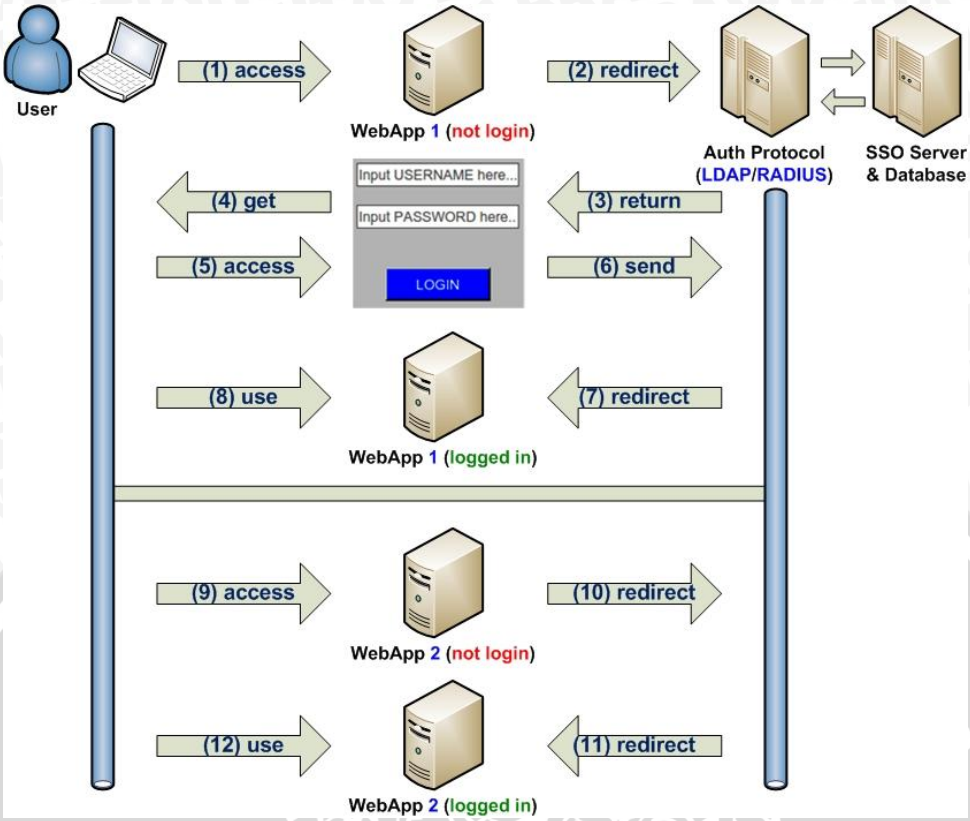
1. Studi Literatur
2. Perancangan Sistem
3. Peralatan Pendukung
4. Implementasi Sistem
5. Pengujian dan Analisis
6. Kesimpulan dan Saran

3.2.1 Studi Literatur

Studi literatur dilakukan untuk mempelajari mengenai dasar teori yang digunakan sebagai panduan penunjang penelitian ini. Dasar teori pendukung tersebut diperoleh dari materi yang bersifat resmi seperti jurnal, *e-book*, forum, dan penelitian sebelumnya. Referensi yang diperlukan untuk menunjang penelitian ini yaitu teori mengenai pengertian umum, dasar sistem, serta cara kerja dari *Single Sign-On*, LDAP, dan RADIUS secara keseluruhan.

3.2.2 Perancangan Sistem

Proses perancangan sistem pada penelitian ini akan menjelaskan tentang alur kerja dari sistem autentikasi terpusat *Single Sign-On*. Secara umum sistem SSO memusatkan proses autentikasi dalam satu *server*, bukan pada masing-masing aplikasi. Karena itu terlebih dahulu *server* autentikasi harus dilakukan instalasi dan konfigurasi hingga *server* tersebut bekerja dengan baik dan lancar. Setelah itu penulis menyiapkan minimal dua aplikasi *web* sebagai objek sistem SSO tersebut dan dikonfigurasi supaya terkoneksi dengan sistem SSO.



Gambar 3.1 Rancangan sistem *Single Sign-On* yang dibangun
Referensi: Hilmi (2012)

Pada gambar 3.1 di atas, pengguna mengakses (1) aplikasi *web* dalam keadaan belum *login* kemudian diarahkan (2) ke protokol autentikasi dan memberikan respon (3) berupa halaman *login*. Pengguna mendapat respon (4) tersebut dan mengakses (5) pada *browser*. Pengguna mengisi data *credential* dan melakukan *login* (6) ke *server*. Jika autentikasi berhasil pengguna diarahkan (7) menuju aplikasi *web* tersebut untuk digunakan (8), namun jika gagal maka diarahkan kembali ke halaman *login*.

Setelah proses autentikasi pada aplikasi *web* pertama berhasil, pengguna mengakses (9) aplikasi *web* kedua yang terhubung ke sistem *Single Sign-On* dan sudah terdapat *session* dari aplikasi pertama tadi (10) maka pengguna diarahkan (11) langsung ke aplikasi kedua tanpa perlu *login* lagi (12).

3.2.3 Peralatan Pendukung

Untuk mendukung proses implementasi dan pengujian penelitian ini, dibutuhkan peralatan pendukung seperti perangkat keras dan perangkat lunak yang digunakan. Sistem ini dirancang sebagai simulasi autentikasi yang bekerja menggunakan satu *server* yang dipasang pada *laptop* penulis dan satu komputer *client* milik laboratorium, kemudian dihubungkan menggunakan koneksi kabel LAN. Berikut ini peralatan pendukung yang digunakan:

- *Server*
  - *Operating System* : Linux Ubuntu 14.04 LTS
  - CPU / *Processor* : Intel Atom N2800 @ 1,8 GHz
  - RAM / *Memory* : 2048 MB
  - *Network Adapter* : Realtek PCIe Gigabit Ethernet
  - *Application* :
    - ✓ Aplikasi *web* minimal dua, rancangan penulis yaitu Rakit dan Sikapen.
    - ✓ Aplikasi *web* khusus autentikasi rancangan penulis yaitu Accounts.
    - ✓ LAMP sebagai *web server*.
    - ✓ LDAP dan RADIUS sebagai protokol autentikasi.
    - ✓ Sublime Text, Google Chrome, dan Terminal sebagai pendukung.
- *Client*
  - *Operating System* : Linux Kali 2.0
  - CPU / *Processor* : Intel Core i3 CPU550 @ 3,2GHz
  - RAM / *Memory* : 2048 MB
  - *Network Adapter* : Realtek PCIe Gigabit Ethernet
  - *Application* :
    - ✓ Apache JMeter untuk menguji kinerja aplikasi *web*.
    - ✓ Wireshark untuk menangkap data (*sniffing*) saat proses autentikasi.
    - ✓ Burp Suite untuk melakukan *penetration testing*.
    - ✓ Aplikasi Python rancangan penulis untuk menguji kinerja protokol.
    - ✓ Google Chrome atau Mozilla Firefox untuk menjalankan aplikasi *web*.

3.2.4 Implementasi Sistem

Pada tahap implementasi sistem SSO, *server* menggunakan Linux Ubuntu dan *client* menggunakan Kali Linux, kemudian melakukan instalasi perangkat lunak yang dibutuhkan. Setelah itu *server* dan *client* dihubungkan dalam satu jaringan menggunakan kabel LAN. Semua aplikasi di *server* dipastikan sudah berjalan dengan lancar. Aplikasi *web* diakses dari *client* dengan mengetik IP *Address server* pada *browser*. Protokol LDAP dan RADIUS dijalankan secara independen untuk melakukan analisis perbandingan kinerja dari masing-masing protokol pada sistem SSO.

Pada *password* akun pengguna menggunakan kriptografi *Secure Hash Algorithm* 256 di mana algoritma *hash* merupakan kriptografi satu arah yang tidak bisa didapatkan kembali nilai aslinya dengan sifatnya *collision resistance*, dan pada aplikasi *web* autentikasi menggunakan protokol HTTPS. Diharapkan penggunaan kriptografi SHA256 dan protokol HTTPS dapat meningkatkan sisi keamanan pada sistem autentikasi SSO tersebut.

3.2.5 Pengujian dan Analisis

Setelah *server* dan *client* sudah siap digunakan, dilakukan pengujian awal untuk melakukan autentikasi melalui aplikasi *web* autentikasi yang sudah dirancang. Terdapat skenario berhasil dan gagal dalam proses autentikasi ini. Jika berhasil *login* maka akan mendapatkan *session* baru dan bisa mengakses *web* Rakit dan Sikapen secara bersamaan, jika gagal *login* maka akan muncul peringatan dan harus kembali ke halaman *login*. Ketika *logout* akan menghapus semua *session* pada aplikasi *web*.

Kemudian dilakukan pengujian dari dua sisi yaitu kinerja dan keamanan. Pada sisi kinerja menggunakan beberapa parameter yang akan diukur yaitu pemakaian CPU, pemakaian RAM, *Throughput*, *Response Time*, dan *Stress Load*. Pada sisi keamanan akan dilakukan serangan pasif (*sniffing*) dan serangan aktif (*dictionary attack*). Sesudah itu akan didapatkan hasil dari pengujian tersebut dan dianalisis perbandingan dari kinerja pemakaian *server*, aplikasi *web*, dan protokol yang digunakan pada implementasi protokol LDAP dan RADIUS.

3.2.6 Kesimpulan dan Saran

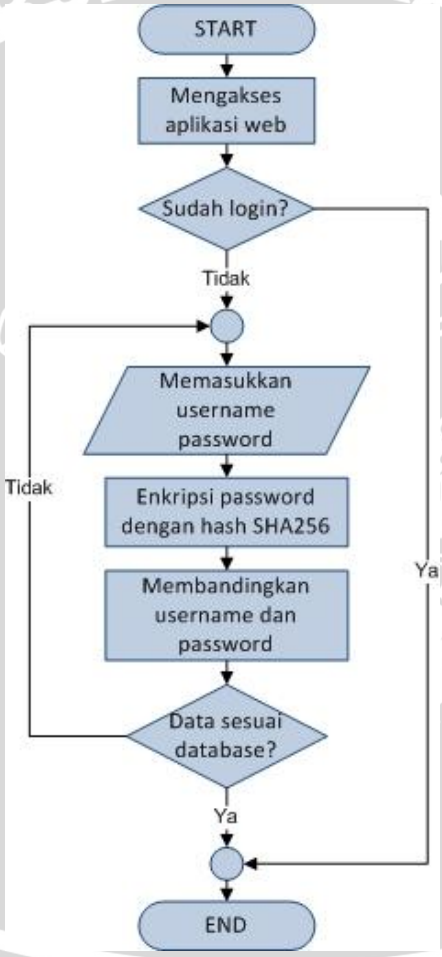
Pengambilan kesimpulan dan saran dilakukan setelah selesai melakukan pengujian dan analisis hasil. Kesimpulan merupakan hasil akhir yang didasarkan pada kesesuaian dengan dasar teori dan implementasi sistem yang telah dilakukan, diharapkan dapat menjadi acuan untuk pemilihan protokol jaringan yang sesuai dengan lingkungan jaringan. Penulisan saran digunakan sebagai acuan pustaka untuk penelitian yang akan dilakukan selanjutnya.

BAB 4 PERANCANGAN

Pada bab ini akan dijelaskan mengenai tahapan perancangan sistem yang akan dilakukan sebelum implementasi, yaitu merancang skema protokol LDAP dan RADIUS serta aplikasi *web* autentikasi sesuai sistem *Single Sign-On*.

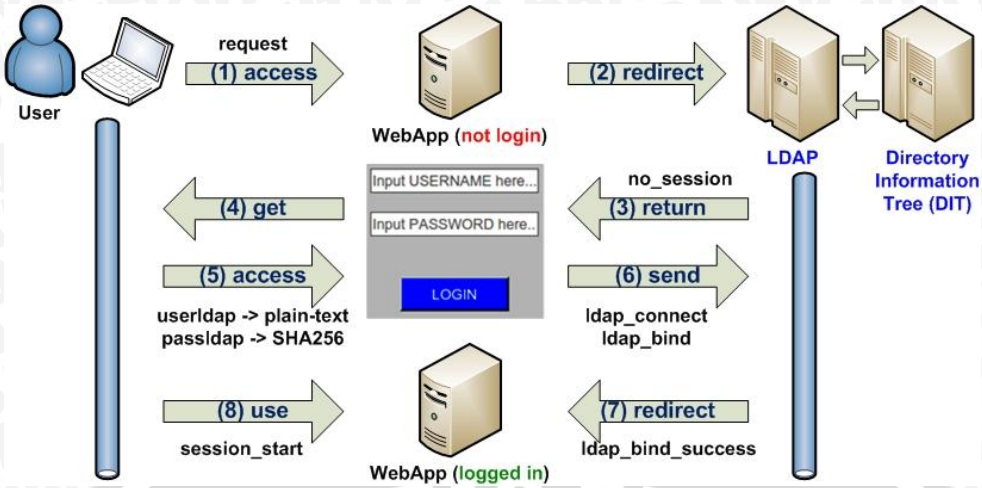
4.1 Rancangan Sistem LDAP dan RADIUS

Sistem autentikasi terpusat *Single Sign-On* harus dilakukan perancangan terlebih dahulu untuk memudahkan pada saat tahap implementasi sistem. Pada tahap ini dilakukan perancangan menggunakan protokol LDAP dan RADIUS yang dikombinasikan dengan aplikasi *web* sebagai objek sistem.



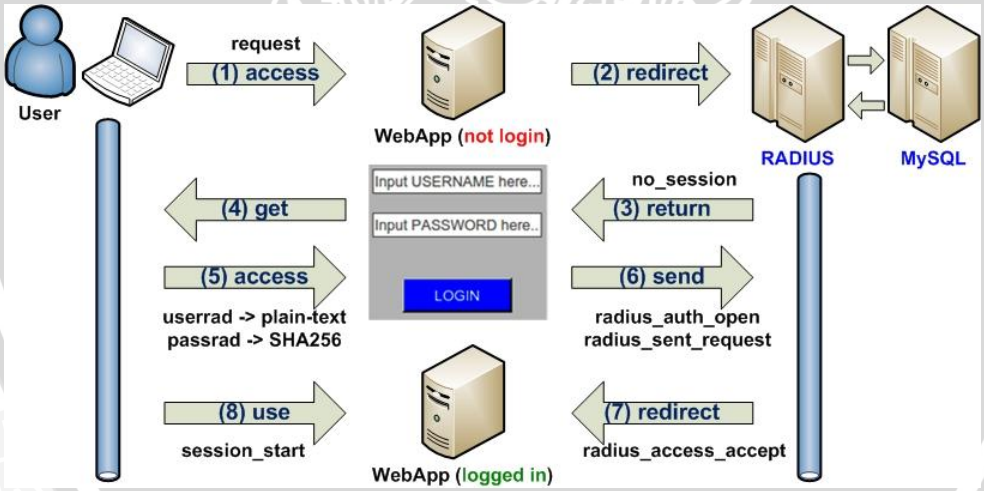
Gambar 4.1 Diagram alir sistem *Single Sign-On* yang dibangun

Pengguna melakukan akses ke aplikasi *web* dan sistem mengecek apakah sudah *login* atau belum. Jika belum *login* langsung diarahkan menuju halaman *login* dan harus memasukkan *username* dan *password*. Kemudian data tersebut dikirim ke server untuk dilakukan pengecekan dengan data di *database*. Jika data *credential* tersebut benar maka langsung mendapatkan akses ke aplikasi *web* yang dituju sebelumnya.



Gambar 4.2 Rancangan sistem menggunakan LDAP

Pada gambar 4.2 merupakan proses autentikasi pada protokol LDAP dengan fungsi yang berjalan di dalamnya. LDAP menggunakan *database* bawaan yaitu *Directory Information Tree* (DIT).Setelah memasukkan data *credential*, kemudiansistem melakukan koneksi ke *server* menggunakan fungsi *ldap_connect* dan proses mengirim data *credential* menggunakan fungsi *ldap_bind*. Jika sukses maka *server* memberikan respon *ldap_bind_success* dan memulai *session* baru.



Gambar 4.3 Rancangan sistem menggunakan RADIUS

Pada gambar 4.3 merupakan proses autentikasi pada protokol RADIUS dengan fungsi yang berjalan di dalamnya. RADIUS menggunakan *database* MySQL. Setelah memasukkan data *credential*, kemudian sistem melakukan koneksi ke *server* menggunakan fungsi *radius_auth_open* dan proses mengirim data *credential* menggunakan fungsi *radius_sent_request*. Jika sukses maka *server* memberikan respon *radius_access_accept* dan memulai *session* baru.

4.2 Konfigurasi Aplikasi *Web* Accounts

Pada simulasi ini, penulis menggunakan dua aplikasi *web* yang dibangun sederhana yaitu Rakit dan Sikapen. Karena sistem SSO memusatkan proses autentikasi, maka harus dibangun aplikasi *web* baru khusus untuk autentikasi yang berfungsi sebagai penghubung antara aplikasi *web* Rakit Sikapen dengan protokol LDAP RADIUS, dengan nama "Accounts". Penelitian ini menggunakan dua protokol *server* yang berjalan secara independen, maka terdapat dua versi yaitu "accounts-ldap" dan "accounts-rad".

Tabel 4.1 Potongan *source code index* aplikasi *web* Accounts

```
<?php session_start();
if(!isset($_SERVER['HTTPS'])){
    header('Location:
https://'.$_SERVER['SERVER_NAME'].$_SERVER['REQUEST_URI']);}?>
<html><head>
<script src="sha256.js"></script>
<script>
    function mySubmit(aForm){
        var inputPass = aForm["passldap"];
        inputPass.value=CryptoJS.SHA256(inputPass.value);
        return true; }
</script>
</head>
<body>
<?php
if(isset($_SESSION['userldap']) && isset($_SESSION['passldap']) &&
isset($_SESSION['agent']) && isset($_SESSION['ipaddr'])){
    include "applist.php";
}else{
?>
<form action="result.php?continue=<?php echo $target; ?>"
method="post" onsubmit="return mySubmit(this);">
<input type="text" name="user" required>
<input type="password" name="pass" required><br>
<input type="submit" value="LOGIN">
</form>
<?php
} ?>
</body></html>
```

Pada tabel 4.1 aplikasi *web* Accounts memulai *session* dan menjalankan protokol HTTPS. Pada *tag head* memuat fungsi *JavaScript* CryptoJS untuk melakukan konversi *hash* menggunakan *script* fungsi *mySubmit(aForm)*. Fungsi *isset* akan mengecek *session* apakah pengguna sudah melakukan *login* atau belum. Jika sudah *login* maka menampilkan daftar aplikasi yang terkoneksi dengan sistem SSO, namun jika belum *login* maka menampilkan *login form* dan harus mengisi *username* dan *password* dengan benar.

Ketika pengguna melakukan klik *Login* menjalankan fungsi konversi *hash* pada *password*, setelah itu menjalankan *file script* result.php dan dikirim menuju *server* untuk dilakukan pengecekan data *credential* di sisi *server*. Proses tersebut merupakan konversi pada sisi *client*. Untuk mendukung hal tersebut, unduh *library* CryptoJS. Ekstrak *file* CryptoJS lalu masuk *folder rollups*, kemudian *copy file* sha256.js ke dalam *folder* aplikasi *web* Accounts.

Tabel 4.2 Potongan *source code result* aplikasi *web* Accounts pada LDAP

```
<?php session_start();
if(!isset($_SERVER['HTTPS'])){
    header('Location:
https://'.$_SERVER['SERVER_NAME'].$_SERVER['REQUEST_URI']);}?>
<html><body>
<?php
$userldap = $_POST[userldap];
$passldap = $_POST[passldap];
$ip = "192.168.1.1";
$port = 389;
$ldap = ldap_connect($ip, $port);
ldap_set_option($ldap, LDAP_OPT_PROTOCOL_VERSION, 3);
if($ldap){
    $bind =
ldap_bind($ldap,"cn=$userldap,ou=users,dc=vm,dc=com",$passldap);
    if($bind){
        $_SESSION......
        include "applist.php";
    }
else echo "<b>Incorrect username and password.</b>";
}
else echo "<b>Can't connect to the LDAP Server.</b>";
ldap_close($ldap); ?>
</body></html>
```

Pada tabel 4.2 merupakan proses pengiriman data *credential* pengguna pada LDAP. Ketika pengguna melakukan klik *Login* akan menjalankan *script* ini dan memulai *session* serta menjalankan protokol HTTPS, lalu menerima *input* dan dimasukkan dalam variabel *userldap* dan *passldap*. Setelah itu membuat koneksi baru dengan protokol LDAP menggunakan fungsi *ldap_connect*, serta fungsi *ldap_set_option* perlu didefinisikan untuk mengecek versi LDAP.

Jika berhasil membuat koneksi, maka dilakukan proses autentikasi dengan fungsi *ldap_bind*. Namun jika gagal, maka menampilkan peringatan tidak bisa terkoneksi ke LDAP *server*. Jika data *credential* benar, maka memulai *session* pengguna dan menampilkan daftar aplikasi yang terkoneksi dengan sistem SSO. Namun jika data salah, maka menampilkan peringatan salah *username* dan *password*, dan harus kembali ke *login form*.

Tabel 4.3 Potongan *source code result* aplikasi *web* Accounts pada RADIUS

```
<?php session_start();
if(!isset($_SERVER['HTTPS'])){
    header('Location:
https://'.$_SERVER['SERVER_NAME'].$_SERVER['REQUEST_URI']);}?>
<html><body>
<?php
$userrad = $_POST[userrad];
$passrad = $_POST[passrad];
$ip = "192.168.1.1";
$port = 1812;
$secret = "rad2307";
$radius = radius_auth_open();
radius_add_server($radius, $ip, $port, $secret, 5, 3);
radius_create_request($radius, RADIUS_ACCESS_REQUEST);
radius_put_attr($radius, RADIUS_USER_NAME, $userrad);
$chall = mt_rand();
```

```
17 $chapval = pack('H*', md5(pack('Ca*',1 , $passrad . $chall)));
18 $passchap = pack('C', 1) . $chapval;
19 radius_put_attr($radius, RADIUS_CHAP_PASSWORD, $passchap);
20 radius_put_attr($radius, RADIUS_CHAP_CHALLENGE, $chall);
21 $result = radius_send_request($radius);
22 switch ($result){
23     case RADIUS_ACCESS_ACCEPT:
24         $_SESSION......
25         include "applist.php";
26     break;
27     case RADIUS_ACCESS_REJECT:
28         echo "<b>Incorrect username and password.</b>";
29     break;
30     default:
31         echo "<b>Can't connect to the RADIUS Server.</b>";
32     break;
33 }
34 radius_close($radius); ?>
35 </body></html>
```

Pada tabel 4.3 merupakan proses pengiriman data *credential* pengguna pada RADIUS. Ketika pengguna melakukan klik *Login* akan menjalankan *script* ini dan memulai *session* serta menjalankan protokol HTTPS, lalu menerima *input* dan dimasukkan dalam variabel *userrad* dan *passrad*. Fungsi *radius_auth_open* untuk membuat koneksi baru ke *server*, lalu fungsi *radius_add_server* untuk memasukkan data IP, *port*, dan *secret key server*. Fungsi *radius_create_request* untuk membuat *request* autentikasi. Fungsi *radius_put_attr* untuk memasukkan semua *input credential* ke dalam satu paket data baru. Setelah itu menjalankan fungsi *radius_sent_request* untuk mengirim paket data tersebut ke *server*.

Proses autentikasi menggunakan metode CHAP-*Password*, membuat nilai *challenge* berupa nilai *random* dengan fungsi *mt_rand*. Kemudian *password* dikombinasikan dengan *challenge* lalu dikonversi menjadi nilai MD5, digabung menggunakan fungsi *pack* dan dimasukkan dalam variabel *passchap*.

Jika berhasil membuat koneksi, maka dilakukan proses autentikasi dengan fungsi *radius_auth_open*. Namun jika gagal, maka menampilkan peringatan tidak bisa terkoneksi ke RADIUS *server*. Jika data *credential* benar, maka *server* mengirim respon *radius_access_accept* lalu memulai *session* dan menampilkan daftar aplikasi yang terkoneksi dengan sistem SSO. Namun jika data salah, maka *server* mengirim respon *radius_access_reject* lalu menampilkan peringatan salah *username* dan *password*.

BAB 5 IMPLEMENTASI

Pada bab ini akan dijelaskan mengenai tahapan implementasi sistem yang sesuai dengan perancangan yang telah dibuat, yaitu melakukan persiapan kebutuhan serta instalasi dan konfigurasi *server* dari awal hingga bisa berjalan dengan lancar sesuai dengan sistem *Single Sign-On*.

5.1 Persiapan Instalasi dan Konfigurasi *Server*

Untuk memulai penelitian ini, perlu disiapkan *server* yang menggunakan Ubuntu 14.04 LTS dan *client* dengan Kali 2.0 dengan masing-masing memiliki keuntungan yaitu Ubuntu banyak didukung tutorial dan aplikasi mengenai instalasi *server* yang tersebar di internet, sedangkan Kali memiliki banyak aplikasi untuk berbagai macam pengujian yang dibutuhkan.

Setelah itu *server* dan *client* dihubungkan dengan kabel LAN tipe *cross* karena langsung menghubungkan dua komputer tanpa perangkat jaringan perantara. Lakukan konfigurasi IP Address dalam satu *network*, kemudian tes ping untuk mengecek komunikasi data apakah sudah lancar. Penulis menggunakan IP *Server* 192.168.1.1 dan IP *Client* 192.168.1.2.

Sebelum melakukan instalasi, hubungkan *server* ke internet karena Linux harus menjalankan perintah yang membutuhkan koneksi internet. Perintah dasar berikut (dicetak miring) akan memudahkan dalam tahap instalasi, jalankan pada Terminal dan lakukan pada setiap awal tahapan, yaitu:

- Masuk tingkat *superuser* memberikan hak akses administrator. *#sudo su*
- *Update local repository* sebelum melakukan instalasi. *#apt-get update*

5.1.1 Instalasi dan Konfigurasi LAMP

Berikut ini merupakan tahapan instalasi dan konfigurasi aplikasi LAMP *Server* hingga dapat berjalan dengan lancar, yaitu:

a) Instalasi aplikasi LAMP. *#apt-get install lamp-server^*

b) Membuat *file* baru, isi dengan "ServerName localhost" lalu simpan. *#nano /etc/apache2/conf-available/fqdn.conf*

c) Aktifkan konfigurasi *fully qualified domain name*. FQDN merupakan nama *domain* yang menentukan lokasi dalam hierarki DNS. *#a2enconf fqdn*

d) Aktifkan konfigurasi 000-default. *#a2ensite 000-default*

e) Aktifkan modul php5. *#a2enmod php5*

f) Instalasi aplikasi database MySQL. *#apt-get install mysql-server libapache2-mod-auth-mysql php5-mysql mysql-workbench*

g) Nonaktifkan baris *bind address* dengan memberi tanda pagar supaya MySQL dapat terkoneksi dinamis. *#nano /etc/mysql/my.cnf*

h) Instalasi aplikasi grafis MySQL, pilih "apache2", pilih yes, isi *password* dengan "ubuntu". *#apt-get install phpmyadmin*

i) Tambahkan "Include /etc/phpmyadmin/apache.conf" pada baris paling bawah lalu simpan. *#nano /etc/apache2/apache2.conf*

j) Ganti "www-data" pada baris *apache_run_user* dan *apache_run_group* sesuai *user* dan *group* komputer, lalu simpan. *#nano /etc/apache2/envvars*

k) Lakukan konfigurasi hak akses pada *folder localhost* supaya dapat diisi dengan file-file aplikasi *web*. *#chmod 755 -R /var/www/html*

l) Menerapkan semua konfigurasi Apache. *#/etc/init.d/apache2 restart*

m) Buat file baru dan isi dengan "<?php phpinfo(); ?>" lalu simpan. *#nano /var/www/html/test.php*

n) Buka *browser* dan ketik "localhost/test.php" akan muncul konfigurasi PHP.

o) Ketik "localhost/phpmyadmin" akan muncul halaman *login* MySQL, gunakan *username* "root" dan *password* "ubuntu".

5.1.2 Instalasi dan Konfigurasi LDAP

Berikut ini merupakan tahapan instalasi dan konfigurasi aplikasi LDAP Server hingga dapat berjalan dengan lancar, yaitu:

a) Instalasi aplikasi LDAP. *#apt-get install slapd ldap-utils*

b) Lakukan konfigurasi awal LDAP. *#dpkg-reconfigure slapd*

- ✓ *Omit OpenLDAP server configuration? No*
- ✓ *DNS domain name? vm.com*
- ✓ *Organization name? VM*
- ✓ *Administrator password? ubuntu*
- ✓ *Database backend to use? HDB*
- ✓ *Do you want the database to be removed when slapd is purged? No*
- ✓ *Move old database? Yes*
- ✓ *Allow LDAPv2 protocol? No*

c) Masuk ke *folder* instalasi Apache. *#cd /etc/apache2*

d) Buat *symbolic link* konfigurasi Apache. *Symlink* yaitu penghubung antara *file* target dengan nama *link* yang ditentukan. *#ln -s conf-enabled conf.d*

e) Instalasi aplikasi grafis LDAP. *#apt-get install phpldapadmin*

f) Ganti pada baris berikut lalu simpan. *#nano /etc/phpldapadmin/config.php*

- ✓ $servers->setValue('server','host','127.0.0.1');
- ✓ $servers->setValue('server','base',array('dc=vm,dc=com'));
- ✓ $servers->setValue('login','bind_id','cn=admin,dc=vm,dc=com');

g) Buat *folder* baru di instalasi Apache. *#mkdir /etc/apache2/ssl*

h) Konfigurasi SSL Apache lalu isi pertanyaan yang muncul. *#openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt*

```
root@titovm:/home/titovm# openssl req -x509 -nodes -days 365 -newkey rsa:2048 -k
eyout /etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt
Generating a 2048 bit RSA private key
.....+++
.............+++
writing new private key to '/etc/apache2/ssl/apache.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ID
State or Province Name (full name) [Some-State]:East Java
Locality Name (eg, city) []:Malang
Organization Name (eg, company) [Internet Widgits Pty Ltd]:VM
Organizational Unit Name (eg, section) []:Admin
Common Name (e.g. server FQDN or YOUR name) []:localhost
Email Address []:
root@titovm:/home/titovm#
```

Gambar 5.1 Konfigurasi SSL pada Apache Server

i) Aktifkan modul SSL pada Apache. *#a2enmod ssl*

j) Ganti nama alias "phpldapadmin" dengan "ldap" lalu simpan. *#nano /etc/phpldapadmin/apache.conf*

k) Tambahkan di bawah baris *DocumentRoot* dengan "ServerName localhost" lalu simpan. *#nano /etc/apache2/sites-enabled/000-default.conf*

l) Aktifkan konfigurasi default-ssl. *#a2ensite default-ssl.conf*

m) Tambahkan di bawah baris *DocumentRoot* dengan "ServerName localhost" dan ganti *file* target pada baris SSLCertificate. *#nano /etc/apache2/sites-enabled/default-ssl.conf*

- ✓ SSLCertificateFile /etc/apache2/ssl/apache.crt
- ✓ SSLCertificateKeyFile /etc/apache2/ssl/apache.key

n) *Copy* seluruh isi *folder* phpldapadmin pada folder baru ldap. *#cp -R /usr/share/phpldapadmin /var/www/html/ldap*

o) Hapus *file* config.php pada *folder* ldap untuk diganti dengan *symbolic link*. *#rm /var/www/html/ldap/config/config.php*

p) Buat *symbolic link* config.php pada phpldapadmin dengan *folder*ldap. *#ln -s /etc/phpldapadmin/config.php /var/www/html/ldap/config/config.php*

q) Menerapkan semua konfigurasi Apache. *#/etc/init.d/apache2 restart*

r) Menerapkan semua konfigurasi LDAP. *#/etc/init.d/slapd restart*

s) Buka *browser* dan ketik "localhost/ldap" akan muncul halaman *login* LDAP, gunakan *login* DN "cn=admin,dc=vm,dc=com" dan *password* "ubuntu".

5.1.3 Instalasi dan Konfigurasi RADIUS

Berikut ini merupakan tahapan instalasi dan konfigurasi aplikasi RADIUS Server hingga dapat berjalan dengan lancar, yaitu:

a) Instalasi aplikasi RADIUS. #*apt-get install freeradius freeradius-mysql freeradius-utils freeradius-common*

b) Instalasi modul php5 supaya bisa dijalankan dengan bahasa PHP. #*apt-get install php5-radius php-pear php5-gd php-db*

c) Aktifkan baris ke 700 dan 712 yaitu "$INCLUDE sql.conf" dan "$INCLUDE sql/mysql/counter.conf" dengan cara hapus tanda pagar, lalu simpan. #*nano /etc/freeradius/radiusd.conf*

d) Aktifkan "sql" baris ke 177, 406, 454, 475, dan 563 dengan cara hapus tanda pagar, lalu simpan. #*nano /etc/freeradius/sites-available/default*

e) Pada baris *secret*, ganti nilai *default* "testing123" sesuai keinginan yaitu "rad2307". #*nano /etc/freeradius/clients.conf*

f) Aktifkan baris "readclients = yes" dengan cara hapus tanda pagar, dan ganti pada baris berikut lalu simpan. #*nano /etc/freeradius/sql.conf*

- ✓ login = "root"
- ✓ password = "ubuntu"
- ✓ radius_db = "radiusdb"

g) Tambahkan dengan "extension=radius.so" pada baris *Dynamic Extension* untuk memuat modul php5-radius. #*nano /etc/php5/apache2/php.ini*

h) Buka link berikut pada browser "http://sourceforge.net/projects/daloradius" kemudian unduh dan ekstrak *file* tersebut lalu pindahkan ke *folder* /var/www/html dan ganti nama folder menjadi "radius".

i) Masuk ke *folder* radius tersebut. #*cd /var/www/html/radius/contrib/db*

j) Ketik *password* MySQL dan masuk ke *console* mysql. #*mysql -u root -p*

k) Buat *database* baru bernama radiusdb. #*create database radiusdb;*

l) Keluar dari console mysql. #*quit*

m) Masukkan skema sql ke *database* radiusdb. #*mysql -u root -p radiusdb < fr2-mysql-daloradius-and-freeradius.sql*

n) Konfigurasi pada baris berikut untuk koneksi dengan *database* lalu simpan. #*nano /var/www/html/radius/library/daloradius.conf.php*

- ✓ $configValues['CONFIG_DB_ENGINE'] = 'mysql';
- ✓ $configValues['CONFIG_DB_HOST'] = 'localhost';
- ✓ $configValues['CONFIG_DB_USER'] = 'root';
- ✓ $configValues['CONFIG_DB_PASS'] = 'ubuntu';
- ✓ $configValues['CONFIG_DB_NAME'] = 'radiusdb';

o) Lakukan pengecekan konfigurasi *syntax*. #*apachectl configtest*

p) Menerapkan semua konfigurasi Apache. #*/etc/init.d/apache2 restart*

q) Menerapkan semua konfigurasi RADIUS. #*/etc/init.d/freeradius restart*

r) Buka *browser* dan ketik "localhost/radius" akan muncul halaman *login* RADIUS, gunakan *username* "administrator" dan *password* "radius".

5.2 Pengisian Data Pengguna Baru

Setelah melakukan instalasi dan konfigurasi protokol *server*, saatnya memasuki tahap pengisian data pengguna baru yang akan digunakan untuk proses autentikasi pada aplikasi *web*. Untuk *password* pengguna menggunakan *hash* 32-bit yang lebih aman daripada enkripsi. Karena itu, pengisian *password* pada *database server* berupa nilai SHA256, bukan *plain-text*. Namun untuk *username* tetap menggunakan *plain-text*.

5.2.1 Pengisian Data pada LDAP

Pengisian data pengguna baru dapat dilakukan melalui aplikasi grafis phpLDAPadmin. Buka browser dan berikut tahapannya:

a) Ketik pada URL browser "localhost/ldap" lalu isikan *login* DN dengan "cn=admin,dc=vm,dc=com" dan *password* dengan "ubuntu".

b) Klik tanda plus untuk *expand tree* dari dc=vm,dc=com.Klik Create new entry here, pilih Generic: Organizational Unit.

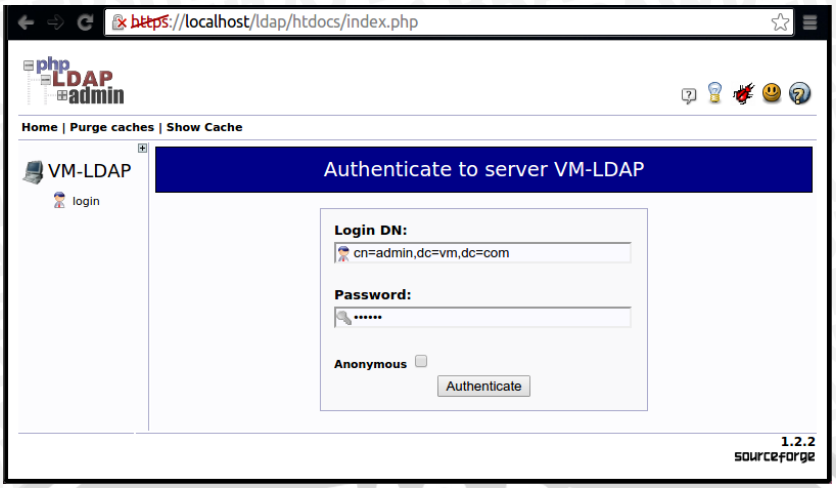
c) Isikan dengan "users", klik Create Object, klik Commit.

d) Kemudian klik ou=users pada kolom kiri.Klik Create a child entry, pilih Generic: User Account.

e) Isikan hanya pada Common Name dengan "titovm" dan Password dengan "c9d1aba96f5598dd824728477ea209985959bd2bbbb66b6c9ad04e8c7fd47137" yaitu nilai SHA256 dari titovm. Klik Create Object, klik Commit.

f) Klik lagi ou=users pada kolom kiri untuk melihat semua daftar pengguna.

g) Jika sudah selesai menambahkan user, klik Logout.



Gambar 5.2 Tampilan awal phpLDAPadmin sebelum login

Gambar 5.3 Tampilan phpLDAPadmin saat melihat daftar pengguna

5.2.2 Pengisian Data pada RADIUS

Pengisian data pengguna baru dapat dilakukan melalui aplikasi grafis daloRADIUS. Buka browser dan berikut tahapannya:

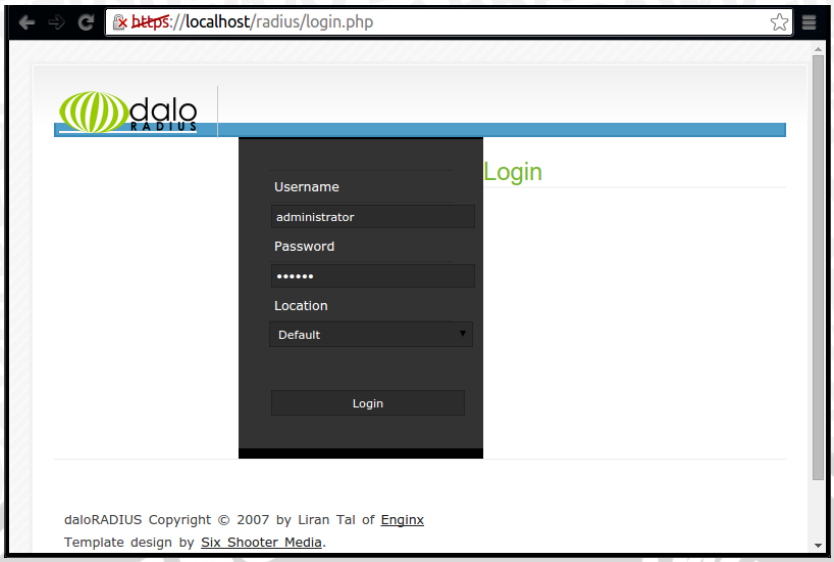
a) Ketik pada URL browser "localhost/radius" lalu isikan *username* dengan "administrator" dan *password* dengan "radius".

b) Klik tab Management, klik tab Users, pilih New User.

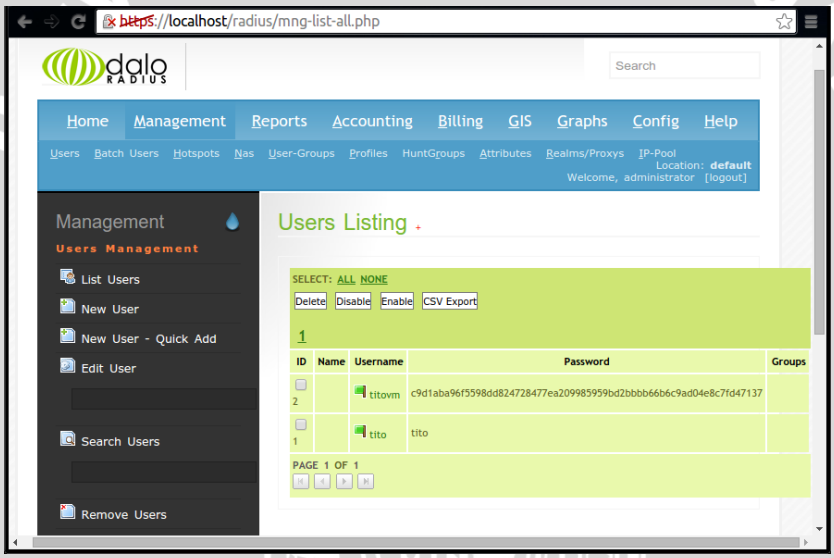
c) Isikan pada Username dengan "titovm" dan Password dengan "c9d1aba96f5598dd824728477ea209985959bd2bbbb66b6c9ad04e8c7fd47137" yaitu nilai SHA256 dari titovm. Klik Apply.

d) Klik List Users untuk melihat semua daftar pengguna

e) Jika sudah selesai menambahkan user, klik Logout.



Gambar 5.4 Tampilan awal daloRADIUS sebelum login



Gambar 5.5 Tampilan daloRADIUS saat melihat daftar pengguna

BAB 6 PENGUJIAN

Pada bab ini akan dijelaskan mengenai pengujian sistem yang telah dibuat. Pengujian dilakukan dari dua sisi yaitu sisi kinerja dan sisi keamanan dari masing-masing *server* LDAP dan RADIUS dengan paramater pengujian yang sudah ditentukan sebelumnya.

6.1 Pengujian Awal Autentikasi

Pengujian awal diperlukan untuk melakukan percobaan apakah sistem yang dirancang sudah diterapkan dengan baik dan apakah berjalan dengan lancar. Setelah melakukan konfigurasi aplikasi *web* Accounts pada masing-masing LDAP dan RADIUS, penulis sudah menyiapkan aplikasi *web* Rakit dan Sikapen yang sudah ada sebelumnya. Proses autentikasi yang terjadi secara umum memang sama, namun berbeda pada tiap fungsi yang digunakan antara LDAP dan RADIUS yang berfungsi sebagai *server* sekaligus protokol jaringan.

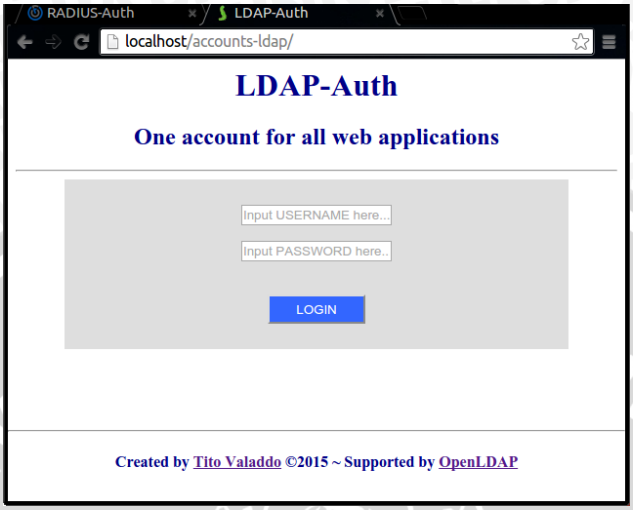
Tabel 6.1 Perbedaan fungsi autentikasi antara LDAP dan RADIUS

Fungsi LDAP	Fungsi RADIUS
$_POST["userldap"] => plain-text $_POST["passldap"] => SHA256	$_POST["userrad"] => plain-text $_POST["passrad"] => SHA256
ldap_connect => IP, port	**radius_auth_open** => IP, port, secretkey
distinguished_name => user, pass	**radius_put_attr** => user, CHAP-pass
ldap_bind => auth	**radius_sent_request** => auth
ldap_bind_success => response session_start => logged-in	**radius_access_accept** => response session_start => logged-in
ldap_bind_failed => response **ldap_close** => not logged-in	**radius_access_reject** => response **radius_close** => not logged-in

Pada tabel 6.1 terdapat perbedaan fungsi autentikasi yang digunakan antara protokol LDAP dan RADIUS. Untuk *input* awal menggunakan fungsi PHP yang sama, username dengan nilai *plain-text* dan *password* dengan nilai *hash* SHA256. Kemudian *client* membuat koneksi baru menuju *server* menggunakan fungsi yang berbeda, *ldap_connect* dengan parameter IP dan *port server* sedangkan *radius_auth_open* ditambahkan *secretkey server*. Setelah itu data *credential* dikirim ke *server*, pada LDAP dimasukkan fungsi *distinguished_name* sedangkan RADIUS dimasukkan fungsi *radius_put_attr*. Kemudian melakukan pengecekan *credential* dengan fungsi *ldap_bind* atau *radius_sent_request* yang akan memberikan balasan sukses lalu memulai *session* baru jika data benar, dan balasan gagal lalu menutup koneksi menuju *server* jika data salah.

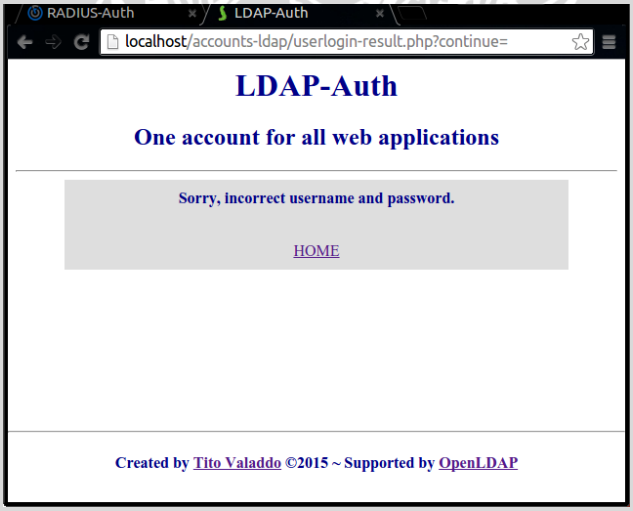
Berikut ini beberapa tampilan dari aplikasi autentikasi *web* "Accounts" dilakukan dengan berbagai skenario pengujian, yaitu:

a. Aplikasi *web* secara otomatis mengarah menuju aplikasi autentikasi Accounts yang berisi *login form* karena pengguna belum *login*.



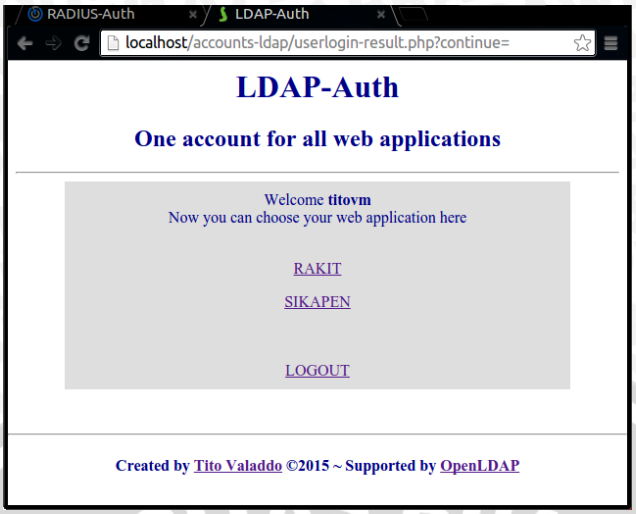
Gambar 6.1 Tampilan awal aplikasi Accounts

b. Pengguna gagal *login* karena salah memasukkan *username* dan *password* yang tidak sesuai pada *database*, kemudian muncul peringatan.



Gambar 6.2 Tampilan Accounts ketika gagal *login*

c. Pengguna berhasil *login* dengan memasukkan *username* dan *password* yang sesuai pada *database server*, kemudian muncul daftar aplikasi *web* yang sudah dikoneksikan dengan sistem SSO.



Gambar 6.3 Tampilan Accounts ketika berhasil *login*

d. Setelah pengguna berhasil *login* maka dapat mengakses aplikasi *web* Rakit dan Sikapen yang sudah terkoneksi pada sistem SSO. Ketika *logout* maka mengarah ke *login form* aplikasi *web* Accounts seperti sebelumnya.

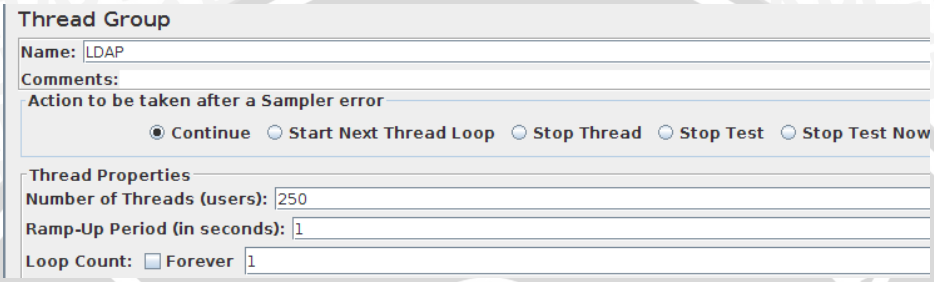
Gambar 6.4 Tampilan aplikasi *web* sederhana Rakit

Gambar 6.5 Tampilan aplikasi *web* sederhana Sikapen

6.2 Pengujian Kinerja

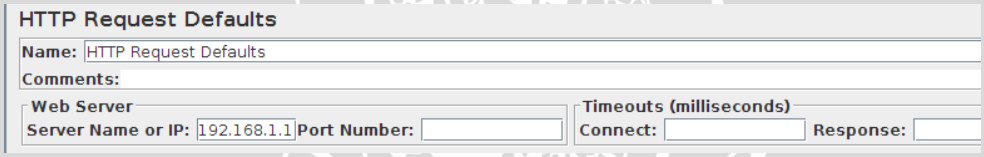
Pada tahap ini, pengujian kinerja menggunakan aplikasi Apache JMeter untuk mengukur parameter pada LDAP dan RADIUS yaitu *Throughput* dan *Response Time*. Skenario pengujian memberikan beban proses autentikasi pada aplikasi *web* yang dilakukan sebanyak empat kali pengujian. Berikut ini cara melakukan pengujian dengan JMeter, yaitu:

a) Buka JMeter, klik kanan *Test Plan*, pilih *Add*, *Threads*, *Thread Group*. Pada *Number of Threads* diisi "250", *Ramp-Up* diisi "1", *Loop Count* diisi "1".



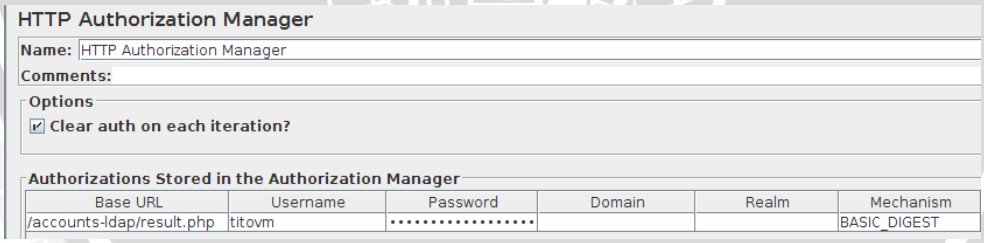
Gambar 6.6 Mengatur jumlah *user* dan batas rentang waktu

b) Klik kanan *Thread Group*, *Add*, *Config Element*, *HTTP Request Defaults*. Pada *Server Name* diisi IP Server yaitu "192.168.1.1".



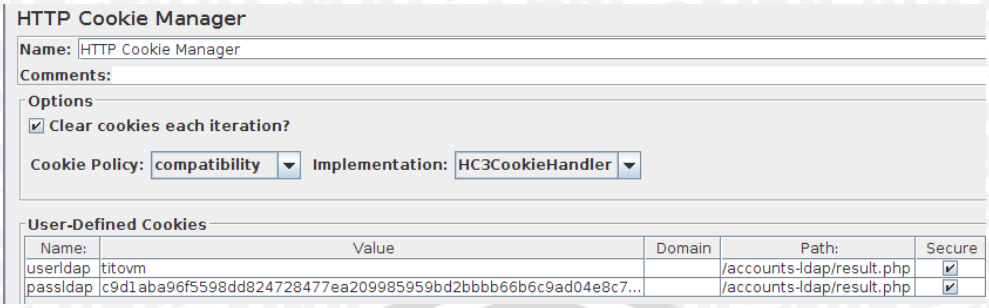
Gambar 6.7 Mengisi IP *Address Server*

c) Klik kanan *Thread Group*, *Add*, *Config Element*, *HTTP Authorization Manager*. Pada *Base* URL isi dengan *link* setelah *login* yaitu "/accounts/result.php", pada *Username* diisi "titovm" dan *Password* diisi nilai *hash* dari "titovm".



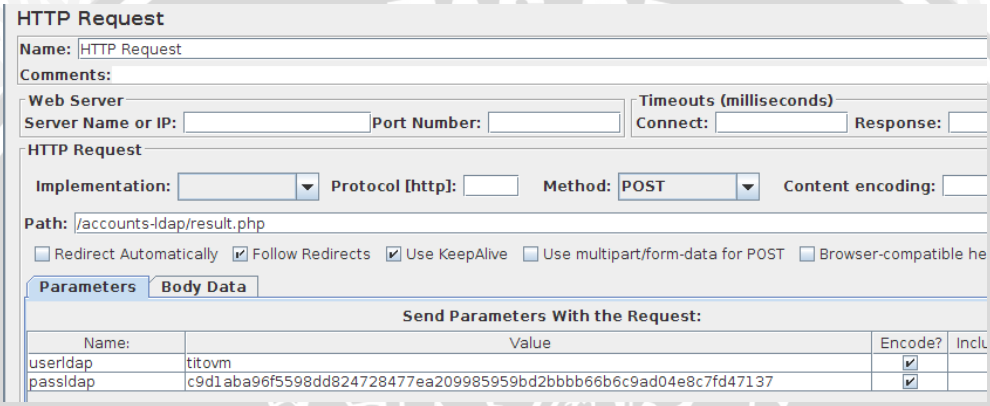
Gambar 6.8 Pengaturan untuk menghapus *auth* pada tiap iterasi

d) Klik kanan *Thread Group*, *Add*, *Config Element*, *HTTP Cookie Manager*. Pada *Name* isi dengan variabel *user* dan *pass*, pada *Value* diisi dengan nilai variabelnya, pada *Path* diisi dengan *link* "/accounts/result.php".



Gambar 6.9 Pengaturan untuk menghapus *cookies* pada tiap iterasi

e) Klik kanan *Thread Group*, *Add*, *Sampler*, *HTTP Request*. Pada *Path* isi dengan *link* setelah *login* yaitu "/accounts/result.php", pada *Parameter* kolom *Name* diisi "user" dan "pass", lalu kolom *Value* diisi dengan nilai dari variabelnya.



Gambar 6.10 Mengisi *link* target dan variabel *credential* pengguna

f) Klik kanan *Thread Group*, *Add*, *Listener*, pilih *View Results Tree* dan *Summary Report* untuk menampilkan hasil.

g) *Test Plan* untuk LDAP sudah selesai, sekarang buat *Test Plan* kembali untuk RADIUS dengan langkah yang sama.

h) Klik *Start* untuk memulai pengujian, lihat hasil pengujian pada *Listener*.

Pengujian kinerja berikutnya yaitu menggunakan perintah "iostat" dan "vmstat" yang dijalankan pada Terminal untuk mengukur parameter pemakaian CPU dan pemakaian RAM pada komputer *server*. Berikut contoh tampilannya.

```
File  Edit  View  Search  Terminal  Help
titovm@titovm:~$ iostat
Linux 3.13.0-32-generic (titovm)        29/12/15        _i686_  (4 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           9,68    0,15    2,49    3,01    0,00   84,66

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda              22,95       677,85        68,26     525333      52904

titovm@titovm:~$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 889940  58932 488280    0    0   174    17  136  349 10  3 85  3  0
titovm@titovm:~$
```

Gambar 6.11 Mengukur pemakaian CPU dan RAM pada *server*

Pengujian kinerja berikutnya menggunakan aplikasi bahasa Python untuk mengukur parameter *Stress Load* dengan *multithread*. Penulis membangun sendiri program *tester* menggunakan Python karena mendukung *threading* yang langsung menguji *server* tanpa perantara. Skenario pengujian ini memberikan beban 50 *req/sec* pada masing-masing protokol dilakukan empat kali pengujian.

Program *tester* ini dijalankan melalui *command line* "python filename.py result.csv" dengan filename.py diganti dengan nama *file* dan hasil keluaran tersimpan dengan format csv yang menggunakan fungsi *csv.writer* dari *library* csv milik Python untuk memudahkan analisis hasil pengujian. Berikut *source code* program untuk melakukan *Stress Load*.

Tabel 6.2 *Stress Load* autentikasi LDAP dengan Python

```
 1  import ldap
 2  import time
 3  import datetime
 4  import hashlib
 5  import threading
 6  import csv
 7  import sys
 8  def auth(u):
 9        row = reader.next()
10        uname = row[0]
11        passwd = "titovm"
12        for i in range(50):
13              start = datetime.datetime.now()
14              l = ldap.open("127.0.0.1",389)
15              l.protocol_version = ldap.VERSION3
16              h = hashlib.sha256()
17              h.update(passwd)
18              passhash = h.hexdigest()
19              l.simple_bind(uname,passhash)
20              end = datetime.datetime.now()
21              delta = end - start
22              sec = (str(delta))[-9:]
23              print (u),uname,(i),"-",sec
24              writer.writerow(((u),uname,(i),sec))
25              time.sleep(0.02)
26  o = open('user.csv','rb')
27  reader = csv.reader(o)
28  f = open(sys.argv[1],'wt')
29  writer = csv.writer(f)
30  writer.writerow(('Loop','User','Thread','Time'))
31  for u in range(400):
32        t = threading.Thread(target=auth,args=(u,))
33        t.start()
```

Pada tabel 6.2 program *tester* ini memuat semua *library* yang dibutuhkan untuk melakukan autentikasi pada LDAP. Membuat fungsi auth(u) diisi dengan proses autentikasi (*thread*) dengan pengulangan 50 kali dan delay 0,02 *second* sehingga beban menjadi 50 *request/sec*. Fungsi *datetime.datetime.now* untuk menampilkan waktu pada saat itu, diletakkan di awal dan akhir *thread* untuk mencatat waktu yang digunakan. Fungsi *ldap.open* untuk melakukan koneksi ke server, dan versi LDAP harus didefinisikan. Lalu memanggil fungsi *hashlib.sha256* untuk melakukan konversi *password* menjadi *hash*, kemudian melakukan autentikasi dengan fungsi *simple_bind*.

Untuk variabel *uname* diambil dari daftar pengguna pada file "user.csv" menggunakan fungsi *csv.reader*, namun menggunakan isi variabel *passwd* yang sama. Daftar *username* dan *password* tersebut harus sama pada *database* masing-masing protokol *server*. Penulis menggunakan *username* dari "user1" hingga "user1000" dengan menggunakan *password* yang sama yaitu nilai hash dari "titovm". Pengujian ini melakukan autentikasi dengan *user* yang berbeda sehingga didapatkan hasil pengujian yang valid.

Setelah selesai membuat fungsi auth(u), pada program utama melakukan pengulangan sebanyak 100, 200, 300, dan 400 *user*. Pada pengulangan tersebut memanggil fungsi auth(u) dengan metode *threading* Python.

Tabel 6.3 *Stress Load* autentikasi RADIUS dengan Python

```
 1  from radius import RADIUS
 2  import time
 3  import datetime
 4  import hashlib
 5  import threading
 6  import csv
 7  import sys
 8  def auth(u):
 9        row = reader.next()
10        uname = row[0]
11        passwd = "titovm"
12        for i in range(50):
13              start = datetime.datetime.now()
14              r = RADIUS("rad2307","127.0.0.1",1812)
15              h = hashlib.sha256()
16              h.update(passwd)
17              passhash = h.hexdigest()
18              r.authenticate(uname,passhash)
19              end = datetime.datetime.now()
20              delta = end - start
21              sec = (str(delta))[-9:]
22              print (u),uname,(i),"-",sec
23              writer.writerow(((u),uname,(i),sec))
24              time.sleep(0.02)
25  o = open('user.csv','rb')
26  reader = csv.reader(o)
27  f = open(sys.argv[1],'wt')
28  writer = csv.writer(f)
29  writer.writerow(('Loop','User','Thread','Time'))
30  for u in range(400):
31        t = threading.Thread(target=auth,args=(u,))
32        t.start()
```

Pada tabel 6.3 program *tester* ini memuat semua *library* yang dibutuhkan untuk melakukan autentikasi pada RADIUS. Fungsi *radius* untuk melakukan koneksi ke server. Lalu memanggil fungsi *hashlib.sha256* untuk melakukan konversi *password* menjadi *hash*, kemudian melakukan autentikasi dengan fungsi *authenticate*. Untuk fungsi-fungsi yang lain, sudah dijelaskan sebelumnya.

Untuk menyimpan hasil pengujian menjadi format csv gunakan fungsi *open(sys.argv[1],'wt')*, kemudian fungsi *csv.writer* dan *writer.writerow* untuk proses menulis isi *file*. Jalankan program ini pada *client* yang sudah terhubung ke *server*. Fungsi *threading* saat dijalankan akan menampilkan hasil secara acak tidak berurutan (*random*).

repository.ub.ac.id

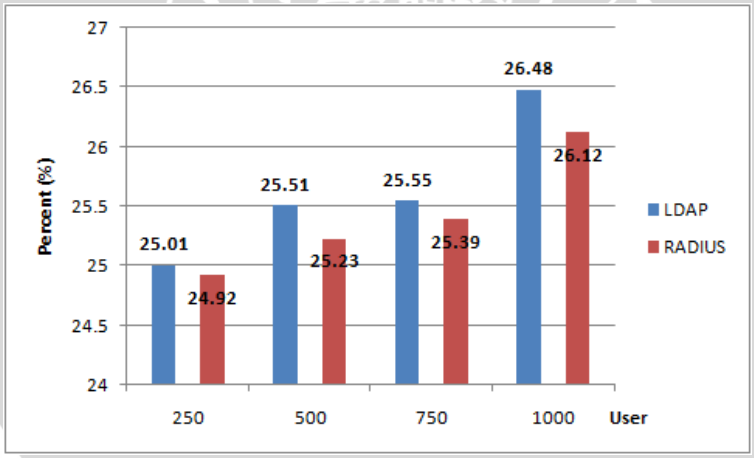
```
File  Edit  View  Search  Terminal  Help
titovm@titovm:~$ python authldap.py ldap200.csv
user 0 0 - 00.006119
user user 23  0 - 00.006897
user 0 - 400.005138 0
 - user 1 0 -user 5  000.00725700.011591
 - 00.002946

user 6 0 - 00.002646
user 7 0 - 00.002979
useruser 9 8 0 - 00.013098
 0 - 00.012382user 10 0 - 00.014153

user 16 0 user 13 0 - 00.020444
- 00.008462
user 15 0 - 00.015303
  useruser  20 1 - 00.014616
  user 4user 111 - user 00.016048 0 1 - 00.014542

 1 1 - 00.015265
- 00.028322
user 6 1 - 00.016539
user 12user  70 1 - 00.017429
 user -  1400.033329 0
 - user 5 1 - 00.018142
```

```
File  Edit  View  Search  Terminal  Help
user 154 48 - 00.005607
user 186 48 - 00.008001
 user 195user user 49  173 49 - 00.006649
-190 47 - 00.004850
user 177 49 - 00.010050
 user 00.006249
 160 49 - 00.007316
user 170 48 - 00.006733
 user user 174 48 - 00.005601
185 49 - 00.006747
user 188 47 - 00.009862
user 197 48 - 00.007697
 user 180 48 - 00.010228
user 154 49 - 00.006176
user 186 49 - 00.003478
user 190 48 - 00.008308
user 170 49 - 00.004886
user 188 48 - 00.002990
 user 174 49 - 00.005736
user 197 49 - 00.002895
user 180 49 - 00.002939
user 190 49 - 00.004844
user 188 49 - 00.004893
titovm@titovm:~$
```

Gambar 6.12 Tampilan *Stress Load* dengan *multithreading* Python

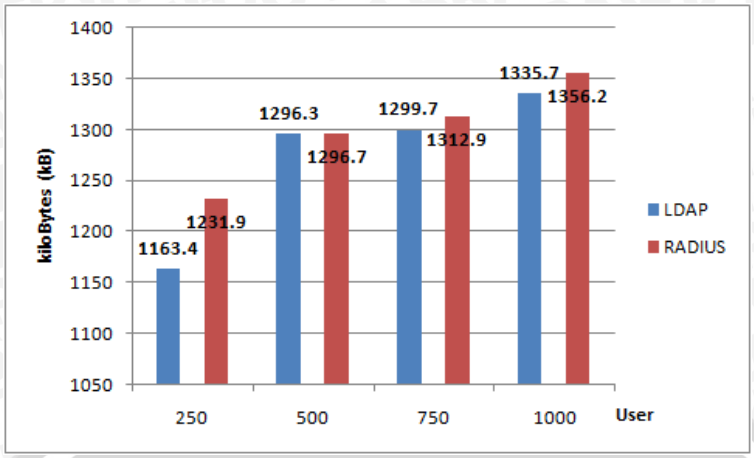
6.2.1 Hasil Pengujian Kinerja Pemakaian CPU dan RAM *Server*

Pengujian pemakaian CPU dan pemakaian RAM menggunakan parameter dari banyak *user* yang mengakses ke *server* autentikasi, dan dilakukan empat kali percobaan secara bertahap pada *server*. Berikut hasil pengujian yang didapatkan dalam bentuk grafik.



Gambar 6.13 Pemakaian CPU pada *server* (*percent*)

UNIVERSITAS BRAWIJAYA

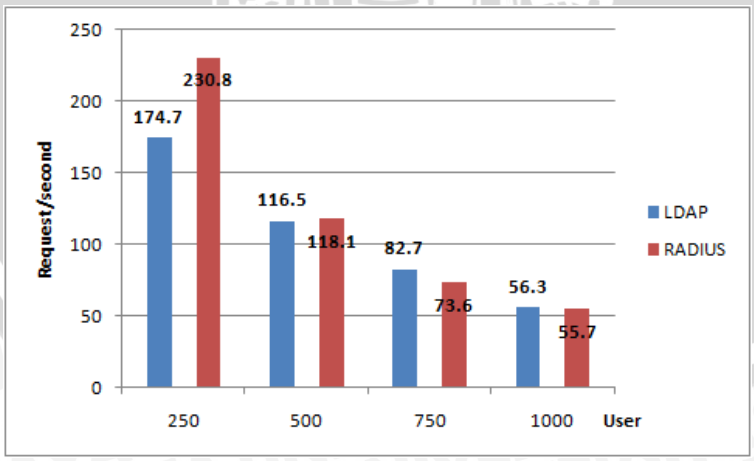


Gambar 6.14 Pemakaian RAM pada *server* (*kiloBytes*)

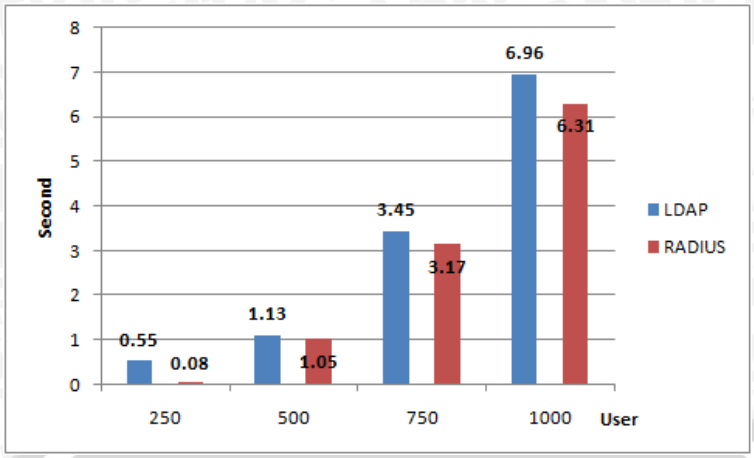
Pada gambar 6.13 pemakaian pada LDAP dengan rata-rata 25,63% dan pada RADIUS dengan rata-rata 25,41%. Kemudian pada gambar 6.14 pemakaian pada LDAP dengan rata-rata 1273,77 kB dan pada RADIUS dengan rata-rata 1299,42 kB. Terdapat perbedaan yang tipis, pada CPU berbeda 0,22% dan pada RAM berbeda 25,65 kB. Hal tersebut karena LDAP dan RADIUS merupakan protokol dengan fungsi yang hampir sama yaitu digunakan untuk autentikasi, sehingga berbeda tipis saat mengunakan sumber daya CPU dan RAM *server*.

6.2.2 Hasil Pengujian Kinerja *Throughput* dan *Response Time*

Pengujian *Throughput* dan *Response Time* menggunakan parameter dari banyak *user* yang mengakses ke *server* autentikasi dan dilakukan empat kali percobaan secara bertahap pada aplikasi *web*. Berikut hasil pengujian yang didapatkan dalam bentuk grafik.



Gambar 6.15 *Throughput* pada aplikasi *web* (*request/second*)

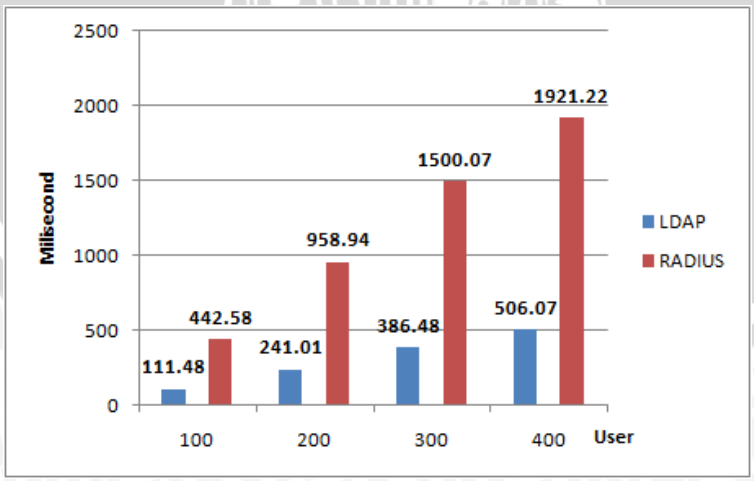


Gambar 6.16 *Response Time* pada aplikasi *web* (*second*)

Pada gambar 6.15 *Throughput* pada LDAP dengan rata-rata 107,55 *req/s* dan pada RADIUS dengan rata-rata 119,55 *req/s*. Kemudian pada gambar 6.16 *Response* pada LDAP dengan rata-rata 3,02 *sec* dan pada RADIUS dengan rata-rata 2,65 *sec*. Terdapat perbedaan yang tipis, pada *Throughput* berbeda 12,0 req/s dan pada *Response* berbeda 0,37 *sec*. Hal tersebut karena pada aplikasi *web* autentikasi versi LDAP dan RADIUS memiliki konten yang hampir sama.

6.2.3 Hasil Pengujian Kinerja *Stress Load*

Pengujian *Stress Load* memberikan beban proses autentikasi pada masing-masing protokol LDAP dan RADIUS dengan beban 50 *request/sec* yang dilakukan empat kali percobaan menggunakan bahasa Python. Pengujian ini juga berpengaruh dari spesifikasi komputer *client* yang digunakan karena bisa membatasi banyaknya jumlah *thread* yang berjalan, disarankan memiliki spesifikasi yang tinggi. Berikut hasil pengujian dalam bentuk grafik.



Gambar 6.17 *Stress Load* pada protokol *server* (*milisecond*)

Pada gambar 6.17 pengujian *Stress Load* pada LDAP dengan rata-rata 311,26 *milisec* dan pada RADIUS dengan rata-rata 1205,71 *milisec*. Terdapat perbedaan signifikan yaitu 894,45 *milisec*. Hal tersebut karena LDAP memiliki *database* sendiri yaitu *Directory Information Tree* (DIT) yang memiliki fitur sederhana dan memang dirancang pada kinerja pembacaan yang sangat cepat daripada proses penulisan. Berbeda dengan RADIUS yang menggunakan *database* MySQL yang memiliki fitur kompleks sehingga akan mempengaruhi kinerja pada proses pembacaan maupun penulisan. LDAP juga merupakan protokol ringan dengan fiturnya yang sederhana, sedangkan RADIUS merupakan protokol dengan fitur yang rumit (AAA) serta memiliki fitur tambahan *secret key* dan metode CHAP-*Password*.

6.3 Pengujian Keamanan

Pada tahap ini, pengujian keamanan akan dilakukan *sniffing* (serangan pasif) atau menangkap paket data saat proses autentikasi sedang berlangsung pada masing-masing *server* menggunakan aplikasi Wireshark. Berikut ini cara pengujian dengan Wireshark, yaitu:

a) Buka Wireshark, pilih *Capture*, *Interfaces*. Pilih *interface* "eth0", klik *Start*.

b) Buka *browser* dan ketik *link* aplikasi *web* "/rakit" atau "/sikapen", secara otomatis akan mengarah ke *login form* aplikasi "/accounts".

c) Isi *username* "titovm" dan *password* "titovm" lalu klik *Login*, muncul pilihan aplikasi *web* Rakit atau Sikapen, pilih salah satu.

d) Setelah masuk *web* yang bersangkutan, klik *Logout*.

e) Lihat lagi Wireshark dan klik *Stop Capture*.

f) Pada Filter, ketik "*frame contains* titovm", pilih pada protokol HTTP dengan info berisi *link* "/accounts/result.php".

g) Pada kolom bawah, pilih *Line-based text data*, data yang tertangkap yaitu *plain-text* dari *username* dan nilai *hash password*.

Pengujian keamanan juga menggunakan aplikasi Burp Suite untuk melakukan uji coba *dictionary attack* (serangan aktif) pada masing-masing *server*. Berikut ini cara pengujian dengan Burp Suite, yaitu:

a) Buka *browser* Firefox, buka *Preferences*, *Advanced*, *Network*, *Settings*. Pilih *Manual Proxy*, lalu HTTP Proxy diisi "127.0.0.1" dan port diisi "8000" (port 8080 dipakai Tomcat), pada isian *No Proxy for* dikosongkan, klik OK.

b) Buka aplikasi Burp Suite, buka *Proxy*, *Options*.Ganti pada *Proxy Listeners*, port "8000" dan *bind address loopback*, klik OK, klik centang kolom *Running*.

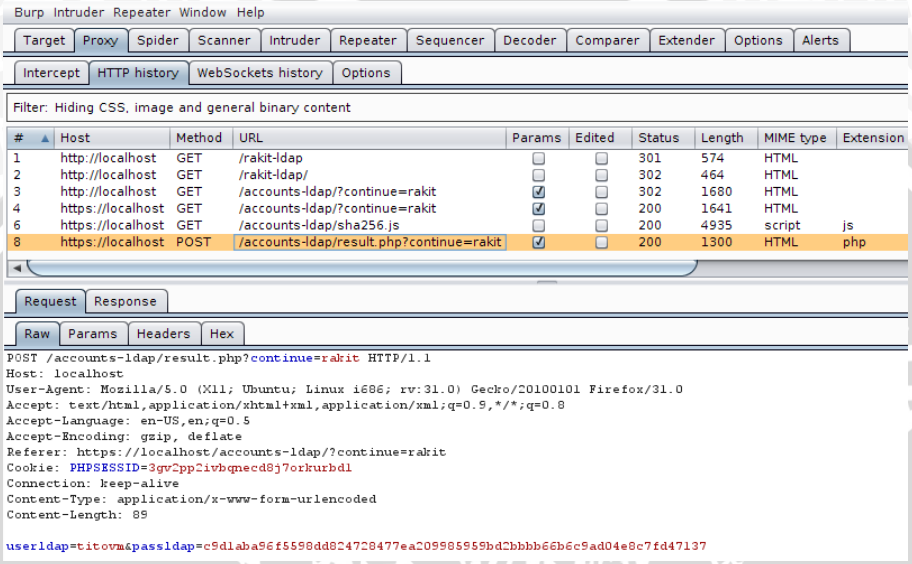
c) Buka *Proxy*, *Intercept*, pastikan statusnya "*Intercept is on*".

d) Lihat Firefox, ketik *link* aplikasi *web* "/rakit" atau "/sikapen". Kemudian lihat Burp Suite, klik *Forward* sampai selesai.

e) Lihat Firefox, *login form* aplikasi Accounts muncul, isi *username* "titovm" dan *password* "titovm", klik *Login*.

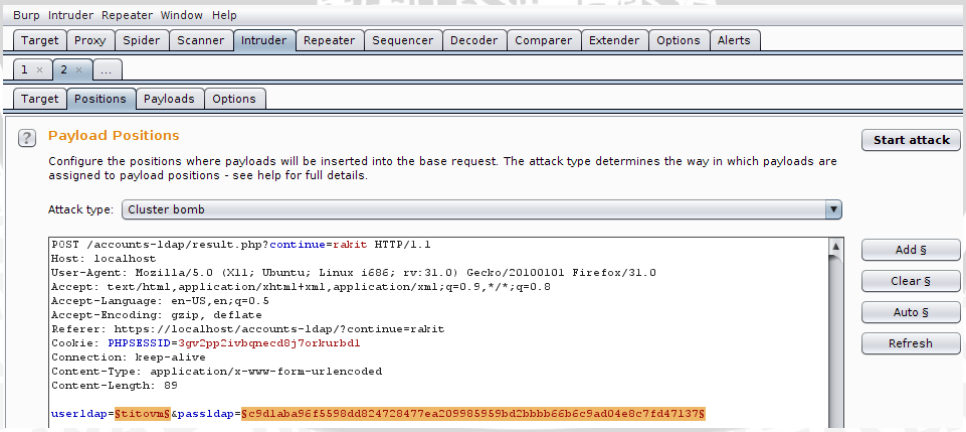
f) Lihat Burp Suite, klik Forward.

g) Buka *Proxy*, *HTTP History*, klik *link* paling akhir akan muncul detail *Request* dan *Response* di bawahnya, klik kanan pada *Request*, pilih *Send to Intruder*.



Gambar 6.18 *Request* HTTP *history* yang akan dikirim pada *Intruder*

h) Buka *Intruder*, *Positions*. Pilih *Attack type* "Cluster Bomb", klik *Clear all payload markers* di kolom kanan, lalu *block* pada nilai *username* dan *hash password* yang ada, klik *Insert new payload marker* di kolom kanan.



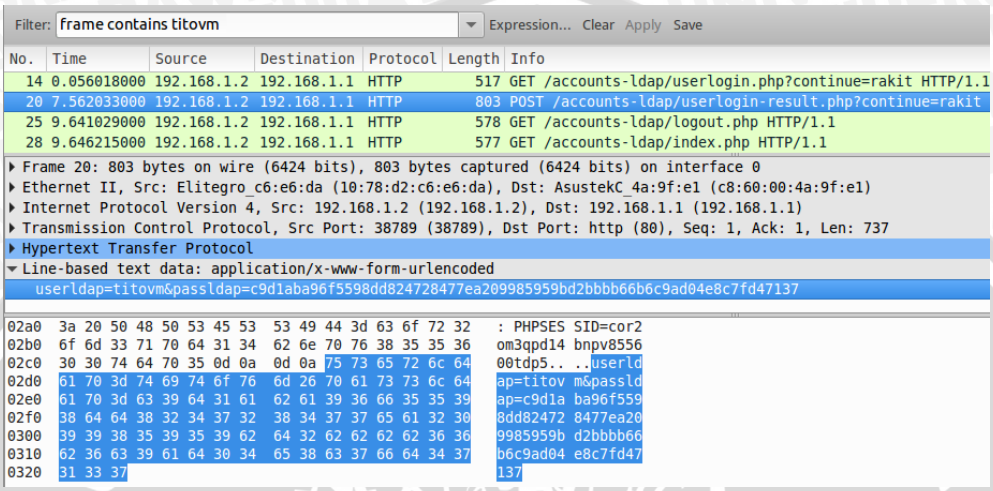
Gambar 6.19 Parameter yang digunakan pada *Intruder*

i) Buka *Intruder*, *Payloads,* isikan daftar kata pada *payload set* 1 (*username*) dan *set* 2 (*password*) untuk *dictionary attack*.

j) Klik *Intruder* dan *Start attack*, akan muncul hasil apakah serangan berhasil dapat dilihat pada *Response*.

6.3.1 Hasil Pengujian Keamanan Data *Sniffing*

Pengujian ini akan menangkap semua data yang terjadi saat proses autentikasi berjalan, kemudian dilakukan *filtering* terhadap data yang penting. Dari hasil *sniffing* tersebut, data yang tertangkap yaitu *username* berupa *plain-text* dan *password* berupa nilai *hash*. Berikut tampilan hasil pengujian data *sniffing* menggunakan Wireshark.

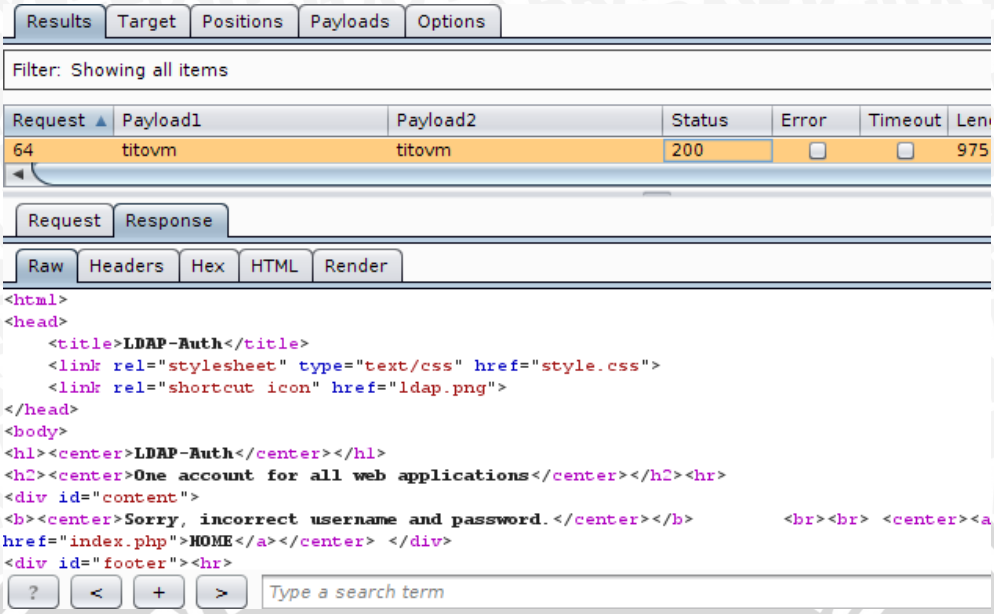


Gambar 6.20 *Sniffing* ketika *login* pada aplikasi *web* Accounts

Pada gambar 6.20 data *credential* pengguna terlihat dengan jelas ketika dilakukan *sniffing* saat proses autentikasi berlangsung. Pada *password* pengguna yang tertangkap berupa nilai *hash*, karena sebelum data *credential* dikirim ke *server* sudah dilakukan konversi *hash* pada sisi *client*. Hal tersebut meningkatkan keamanan karena nilai asli *password* tidak diketahui. Algoritma *hash* sendiri merupakan kriptografi satu arah yang tidak bisa didapatkan kembali nilai aslinya, berbeda dengan algoritma enkripsi yang masih bisa didekripsi (dua arah).

6.3.2 Hasil Pengujian Keamanan *Dictionary Attack*

Pengujian ini merupakan serangan langsung pada aplikasi *web* Accounts dengan *dictionary attack* yang menggunakan sekumpulan daftar kata untuk dimasukkan pada kolom *username* dan *password* di halaman *login*. Berikut tampilan hasil pengujian *dictionary attack* menggunakan Burp Suite.



Gambar 6.21 *Dictionary attack* pada aplikasi *web* Accounts

Pada gambar 6.21 hasil yang didapatkan yaitu *dictionary attack* tidak berhasil melakukan *login*. Meskipun sudah dimasukkan data *credential* yang benar, dapat dilihat pada *Response* yang menampilkan gagal *login*. Hal tersebut terjadi karena *password* pengguna sudah diubah menjadi nilai *hash* SHA256 pada sisi *client*, di dalam *database* juga diisi dengan nilai *hash*, bukan nilai *plain-text*. Dengan hasil tersebut, diharapkan mampu meningkatkan keamanan.

BAB 7 PENUTUP

Pada bab ini berisi kesimpulan dari hasil penelitian sistem autentikasi terpusat *Single Sign-On* yang sudah dirancang, serta saran untuk digunakan sebagai kajian pustaka pada penelitian selanjutnya.

7.1 Kesimpulan

Ditinjau dari hasil implementasi, pengujian, dan analisis penelitian sistem autentikasi *Single Sign-On* ini, dapat disimpulkan bahwa:

a. Sistem autentikasi terpusat *Single Sign-On* berhasil dilakukan implementasi menggunakan protokol LDAP dan RADIUS, dengan dukungan aplikasi *web* autentikasi menggunakan protokol HTTPS dan enkripsi *password* pengguna menggunakan kriptografi *hash* 32-bit SHA256.

b. Penggunaan protokol HTTPS dan kriptografi SHA256 berhasil meningkatkan keamanan sehingga sulit mendapatkan data *credential* pengguna.

c. Didapatkan hasil perbandingan kinerja protokol LDAP dan RADIUS yaitu:

   - Hasil pengujian dari pemakaian CPU, pemakaian RAM, *Throughput*, dan *Response Time* hanya terdapat sedikit perbedaan.
   - Hasil pengujian *Stress Load* terdapat perbedaan signifikan yaitu setara dengan kinerja pembacaan LDAP empat kali lebih cepat daripada RADIUS.
   - LDAP merupakan protokol ringan yang memiliki *database* sendiri yaitu *Directory Information Tree* dengan struktur pohon sehingga kinerja pembacaan menjadi sangat cepat.
   - RADIUS merupakan protokol AAA yang menggunakan *database* MySQL dengan fitur kompleks sehingga mempengaruhi kinerja pembacaan, serta terdapat tambahan *secret key* dan metode CHAP-*Password*.

7.2 Saran

Saran yang dapat penulis sampaikan terkait pada penelitian ini untuk penelitian yang lebih lanjut, yaitu:

a. Melakukan kombinasi implementasi protokol LDAP dengan RADIUS sehingga dapat meningkatkan keuntungan.

b. Implementasi protokol yang lain, misalnya protokol Kerberos yang berbasis *ticket* atau protokol Diameter yang merupakan penerus RADIUS.

c. Dapat menambahkan parameter pengujian atau menggunakan algoritma kriptografi yang lain.

DAFTAR PUSTAKA

Amarudin, Widyawan, 2014. Analisa Keamanan Jaringan Single Sign-On Dengan LDAP Menggunakan Metode MITMA. Indonesia

Yuliansyah, H., 2011. Optimalisasi RADIUS Server sebagai Sistem Otentikasi dan Otorisasi untuk Proses Login Multi Aplikasi Web. Indonesia

Patil, A., Pandit, R., Patel, S., 2013. Analysis of Single Sign On for Multiple Web Applications. India

Machado, B., Nascimento, A.,Tonicelli, R., 2010. Framework for Secure Single Sign-On. Brazil

Ade, D., Suadi, W., 2009. Implementasi Teknologi Single Sign-On di Lingkungan Teknik Informatika ITS. Indonesia

Hilmi, F., Rumani, R., Irawan, B., 2012. Analisis Performansi Autentikasi Single Sign-On pada Web Menggunakan LDAP. Indonesia

Obimbo, C., Ferriman, B., 2011. Vulnerabilities of LDAP as an Authentication Service. Canada

Wang, X., Schulzrinne, H., Kandlur, D., Verma, D., 2008. Measurement and Analysis of LDAP Performance. IEEE. USA

Huntington, G., 2006. 101 Things to Know About Single Sign On. USA

Wang, R., Chen, S., Feng, X., 2012. Traffic-Guided Security Study of Commercially Deployed Single Sign-On Web Services. USA

Findlay, A., 2011. Best Practices in LDAP Security. UK

Yanuar, M.,Iman, K., Fatchur, A., 2012. Implementasi Single Sign-On Berbasis Central Authentication Service Protocol Pada Jaringan Berbasis LDAP. Indonesia

Rudy, Riechie, Gunadi, O., 2007. Integrasi Aplikasi Menggunakan Single Sign-On Berbasis Lightweight Directory Access Protocol Dalam Portal Binus. Indonesia

Internet Engineering Task Force, 2014. The FreeRADIUS Technical Guide - How RADIUS Works. USA

Dekok, A., The FreeRADIUS Project. <http://freeradius.org> [diakses Juli 2015]

Netscape Communications, Hypertext Transfer Protocol Secure. <https://en.wikipedia.org/wiki/HTTPS> [diakses Juli 2015]

National Security Agency, Secure Hash Algorithm. <https://en.wikipedia.org/wiki/Secure_Hash_Algorithm> [diakses Juli 2015]