PENERAPAN MEKANISME *AUTHENTICATION* DENGAN METODE *OPEN AUTHENTICATION* (OAuth) (STUDI KASUS : belajardisini.com)

Afrianda Cahyapratama[1)], Eko Sakti P., S.Kom., M.Kom[2)], Adhitya Bhawiyuga, S.Kom, M.S [3)]

Informatika/Ilmu Komputer, Fakultas Ilmu Komputer Universitas Brawijaya, Malang, Indonesia.

Email : afriandacahyapratama@gmail.com[1)], ekosakti@ub.ac.id[2)], bhawiyuga@ub.ac.id [3)]

Abstrak - OAuth merupakan mekanisme yang berperan dalam mengatur hak akses sumber daya yang dilindungi, karena sumber daya ada yang bersifat terbuka dan ada yang terbatas. OAuth menawarkan mekanisme otentikasi untuk mendapatkan hak akses. Pada sistem terdahulu belajardisini.com, web service yang digunakan belum menerapkan mekanisme otentikasi. Oleh karena itu pada penelitian ini diterapkan mekanisme kerja teknologi OAuth 2.0 pada situs belajardisini.com dengan melibatkan adanya server otorisasi yang merupakan sumber daya server itu sendiri dalam melakukan otentikasi dan otorisasi *credential* (*client_id*, *client_secret*, *redirect_uri*, *api_key*) dari *client*. Otentikasi OAuth 2.0 melalui peran server otorisasi akan dilakukan validasi mandat dari *client*, di proses dengan mengeluarkan sebuah halaman otorisasi untuk diarahkan ke halaman utama setiap aplikasi *client*. Otorisasi akhir dari kebenaran mandat milik *client* adalah di hasilkan sebuah kode otorisasi yang bekerja pada URL (*uniform resource locator*) pada setiap aplikasi *client*. Berdasarkan hasil penelitian dapat disimpulkan bahwa mekanisme otentikasi dengan menggunakan OAuth terbukti dapat menjembatani peminta layanan dan pemilik sumber daya dengan adanya mekanisme otentikasi dan otorisasi dari peran serta pengguna dalam mengakses layanan belajardisini.com.

Kata Kunci - OAuth 2.0, Authorization Server, Otentikasi dan Otorisasi.

Abstract - OAuth is a mechanism whose role in regulating as a right of access to a protected resource, because there are resources that are open and that are limited. OAuth offer an authentication mechanism to get a permission. In the previous system used by belajardisini.com, the web service have not implemented an authentication mechanism. Therefore, in this study the OAuth 2.0 technology mechanisms is implemented on belajardisini.com site involving the authorization server which is a resource server itself in the authentication and authorization credentials (client_id, client_secret, redirect_uri, api_key) from the client. OAuth 2.0 Authentication mechanism via Authorization Server will validate the role of client credential, in the process by issuing an authorization page (Authorize Request) to be redirected to the main page of each respective client application. Final authorization of the client credential validity is generate an authorization code that works on a URL (uniform resource locator) on each client application. Based on the results of this study concluded that using OAuth authentication mechanism is proven to connect the requester and resource owners with their authentication and authorization mechanisms of the participation of users in accessing services belajardisini.com.

Keywords - OAuth 2.0, Authorization Server, Authentication and Authorization.

1 PENDAHULUAN

Web service muncul untuk mendukung sistem terdistribusi yang berjalan pada infrastruktur yang berbeda. Saat ini *web service* tidak hanya dapat diakses melalui komputer saja, tetapi juga dapat diakses melalui perangkat *mobile*, sehingga memungkinkan untuk diciptakannya aplikasi *mobile* yang memanfaatkan layanan *web service*[2].

Seiring dengan perkembangan *web service* tersebut juga telah diciptakan beberapa arsitektur *web service* dan salah satunya yang dibahas di sini adalah arsitektur REST (*Representational State Transfer*). REST memiliki 2 format pertukaran data, yaitu dalam format XML dan format JSON (*Java Script Object Notation*). Dimana perbandingan performa dari keduanya, *web service* dengan format pertukaran data JSON memiliki performa yang lebih baik karena memiliki ukuran data lebih kecil dibandingkan dengan format XML sehingga dalam proses parsing data menjadi lebih cepat[4].

OAuth (*Open Authentication*) adalah protokol *authentication* yang bersumber dari layanan penyedia API, yang memberikan kuasa kepada seseorang untuk mendapatkan hak akses mereka dengan menukarkan *credential* (*username dan password*) menjadi *code accses token*[14]. *Authentication* dengan menggunakan OAuth mengatur sedemikian rupa agar pengunjung tetap aman menggunakan layanan aplikasi dengan mempercayakan kehadiran pihak ketiga (*third party*) yang telah dipercayai sebagai sumber server (*resource server*) yang mempunyai kuasa untuk menertibkan dan mengatur segala *credential* yang ada. *Resource server* juga merupakan *authorization server* yang melakukan proses otorisasi *client* dalam melakukan permintaan API dengan menjadikan *authorization code* untuk menjadi *code accses token*[15]. Pada intinya, *OAuth* adalah

mekanisme *authentication* untuk aplikasi saat peminta layanan akan mengakses API atas nama pengguna tanpa harus memiliki *username* dan *password*. Sebaliknya, aplikasi mendapatkan *token* yang dapat mereka gunakan dengan mandat aplikasi mereka sendiri untuk membuat panggilan API[13].

2 DASAR TEORI

2.1 Definisi *Web Service*

World Wide Web Consortium (W3C) mendefinisikan *web service* sebagai suatu sistem perangkat lunak yang dirancang untuk mendukung interoperabilitas dan interaksi antar mesin/sistem pada suatu jaringan[11].

2.2 REST *Web Service*

REST adalah singkatan dari *Representational State Transfer*. Istilah REST atau RESTful pertama kali diperkenalkan oleh Roy Fielding pada disertasinya di tahun 2000[7]. REST bukan sebuah standar protokol *web service*, melainkan hanya sebuah gaya arsitektur. Ide dasar dari arsitektur REST adalah bagaimana menghubungkan jalur komunikasi antar mesin/aplikasi melalui HTTP sederhana.

2.3 Komunikasi Data JSON

JSON adalah singkatan dari *JavaScript Object Notation* yang merupakan objek asli bawaan *JavaScript*[10]. Pada aplikasi berbasis *JavaScript*, penggunaan JSON sebagai format pesan pertukaran akan membawa dampak performa yang cukup signifikan dibandingkan bila menggunakan XML, karena penggunaan XML melibatkan *library* tambahan untuk membaca data dari *Document Object Model* (DOM) [12].

2.4 Authentication

Dalam sebuah *web application* dibutuhkan sebuah mekanisme yang dapat melindungi data dari pengguna yang tidak berhak mengaksesnya. Mekanisme tersebut yaitu proses *authentication* yang pada prinsipnya berfungsi sebagai kesempatan pengguna dan pemberi layanan dalam proses pengaksesan *resource*[6].

2.5 *OAuth 2.0*

OAuth 2.0 lebih menekankan pada kemudahan *client* sebagai pemilik dan pengembang aplikasi dengan memberikan otorisasi khusus di berbagai aplikasi. OAuth berada dalam pengembangan IETF OAuth WG dan didasarkan pada usulan WRAP OAuth. WRAP (*Web Resource Authorization Protocol*) adalah profil OAuth yang memiliki sejumlah kemampuan penting yang tidak tersedia di versi OAuth sebelumnya. Spesifikasi terbaru dari OAuth disumbangkan kepada IETS OAuth WG dan merupakan dasar dari terciptanya OAuth versi 2[14]

2.5.1 *Authorization Grant*

Authorization Grant (Hibah Otorisasi) adalah mandat yang mewakili otorisasi sumber daya pemilik (*Resouce Owner*) untuk dapat mengakses sumber daya yang dilindungi yang digunakan oleh *client* (pengunjung) untuk mendapatkan *code access token*. Ada beberapa metode hibah otorisasi : *authorization code*, *implicit*, *resource owner password credential*, *client credential*[14].

2.5.2 *Web Server Application*

Aplikasi web pada bahasa pemrograman *server-side* seperti php, asp.net, jsp melakukan proses kerjanya pada sisi server yang tidak dipublikasikan untuk umum[18]. Proses yang terjadi dan berlaku saat seorang pengunjung melakukan kegiatan otentikasi dengan menekan tombol "*Sign-In*" adalah pengunjung akan menerima rangkaian *link* pada alamat browser seperti pada gambar berikut[19] :

```
http://oauth2server.com/auth?response_
type=token&client_id=CLIENT_ID&redirec
t_uri=REDIRECT_URI&scope=photos
```

Gambar 2.1. Proses otentikasi Web Server Application

2.5.3 *Browser Based Application*

Aplikasi *browser* dijalankan sepenuhnya dalam *browser* setelah memuat suatu kode sumber dari suatu halaman web. Karena seluruh kode sumber tersedia untuk dan pada *browser*, sehingga kerahasiaan dari rahasia *client* tidak digunakan pada aplikasi ini[19]. Proses yang tercipta apabila seorang pengunjung melakukan kegiatan otentikasi dengan menekan tombol "Sign In" yang tersedia pada aplikasi *browser*, seperti yang ditunjukkan pada gambar berikut :

```
http://oauth2server.com/auth?response_
type=token&client_id=CLIENT_ID&redirec
t_uri=REDIRECT_URI&scope=photos
```

Gambar 2.2. Proses Otentikasi Browser Based Application

2.5.4 *Mobile Based Application*

Aplikasi *mobile* hampir sama seperti *browser based application*, aplikasi *mobile* juga tidak bisa menjaga kerahasiaan rahasia *client* mereka. Karena itu, aplikasi *mobile* juga harus menggunakan aliran OAuth yang tidak memerlukan rahasia *client*[19]. Misal pada aplikasi berbasis android, aplikasi dapat mendaftarkan

pola pencocokan URL yang akan meluncurkan aplikasi asli jika URL cocok pola akan dikunjungi[19]. Gambar 2.3 akan menunjukan URL untuk otorisasi aplikasi.

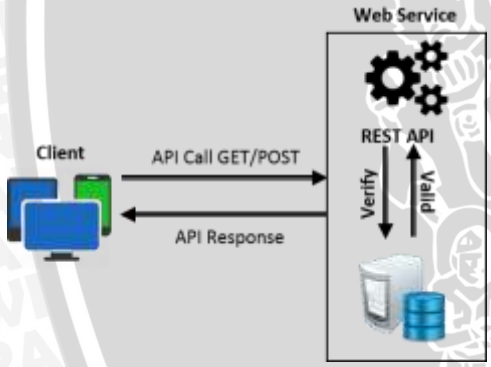
```
http://facebook.com/dialog/oauth?response_type=token&client_id=CLIENT_ID&redirect_uri=REDIRECT_URI&scope=email
```

Gambar 2.3. Proses Otentikasi Mobile Based Application

3 PERANCANGAN

3.1 Arsitektur Awal Sistem

Belakangan ini REST API banyak dibutuhkan, seiring populernya pemrograman *smartphone* yang menggunakan data dari server. Begitu juga pada sebuah aplikasi web dengan situs belajardisini.com yang sudah menerapkan *web service engine* dengan menggunakan REST. REST API yang digunakan pada situs belajardisini.com menggunakan *framework CodeIgniter*. Berikut adalah sebuah gambar arsitektur awal sistem yang sudah dianalisa dari situs belajardisini.com :

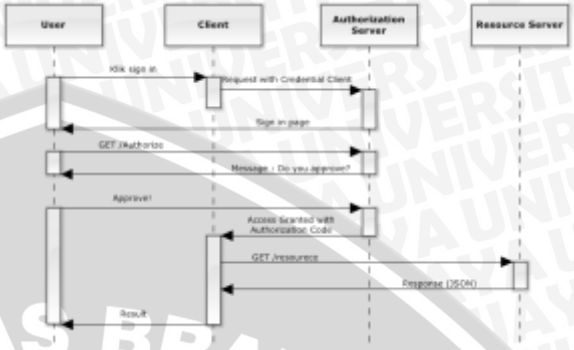


Gambar 3.1. Arsitektur Awal Sistem

3.2 Pengembangan Sistem Dengan OAuth 2.0

OAuth (*Open Authentication*) merupakan *authorization framework* yang memungkinkan aplikasi pihak ketiga untuk memperoleh *limited access* ke http service, OAuth ini bertindak sebagai perantara interaksi *approval* antara pemilik sumber daya dan layanan http. Dengan OAuth, aplikasi pihak ketiga tidak perlu melemparkan *username* dan *password* untuk mendapatkan akses ke aplikasi pemilik *resource*. Jadi, fungsi OAuth pada intinya ialah memberikan izin akses kepada aplikasi pihak ketiga untuk mengakses informasi yang terdapat di penyedia layanan tanpa harus membagi keseluruhan data.

Dengan melihat kondisi teknologi saat ini berdasarkan yang sudah dibahas sebelumnya, penulis telah melakukan percobaan untuk menerapkan proses keamanan layanan dengan memanfaatkan metode OAuth (*Open Authentication*) agar penyedia layanan lebih aman dengan adanya proses *authentication* sebelum peminta akses mulai menggunakan layanan yang tersedia. Berikut adalah gambar arsitektur baru sistem setelah di implementasikan OAuth pada *web service* belajardisini.com :



Gambar 3.2. Diagram Alur Sistem Dengan OAuth

3.3 Alur Teknologi OAuth 2.0

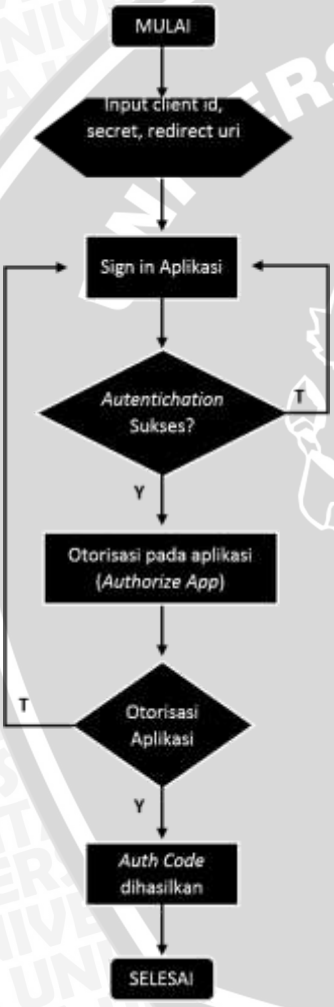
Teknologi *authentication* yang diterapkan pada prosedur mekanisme *authentication* "sign-in" setiap *client* menggunakan teknologi OAuth 2.0. Pada gambar sebelumnya yaitu gambar 3.2 di atas adalah alur kerja mekanisme *authentication* OAuth 2.0 yang diilustrasikan berjalan pada aplikasi yang sudah diterapkan menggunakan mekanisme *authentication* dengan metode OAuth 2.0 seperti aplikasi yang ada pada situs belajardisini.com.

Tabel 3.1. Keterangan Alur Teknologi OAuth 2.0

Proses	Keterangan
A	*Client* yang mengakses aplikasi milik *Resource Owner* melalui perantara internet menuju *authorization server* untuk melakukan "*sign-in*" pada aplikasi.
B	*Authorization Server* (Server Otorisasi) melakukan otentikasi *credential* kepemilikan *client*, memberikan persetujuan apakah pemilik sumber daya memberi atau menolak permintaan *client* untuk bisa mengakses layanan.
C	Dengan asumsi *client* mendapatkan kuasa melakukan akses, server otorisasi (*authorization server*) menampilkan halaman *sign-in*.
D	Apabila *client* yang sudah menerima kode otorisasi (*authorization code*) sebelumnya, maka akan mendapatkan perintah yang di terima berupa *authorize*

	app, persetujuan melakukan otorisasi yang di kirim ke *authorization server*.
E	Jika *client* melakukan otorisasi, *authorization code* akan dihadirkan bersamaan dengan dihantarkannya *client* pada halaman selanjutnya dari aplikasi.
F	Aplikasi *client* sudah berhasil mendapatkan hak akses dan sudah bisa masuk untuk mengakses layanan yang ada.

3.4 Flowchart *Authentication* OAuth 2.0



Gambar 3.3. Flowchart Sistem Dengan OAuth 2.0

Prosedur *flowchart* di atas di gunakan untuk memproses *credential* dari *client* apakah valid atau tidak, menggunakan prosedur teknologi *authentication* OAuth. Di bawah ini akan di jelaskan prosesnya secara terperinci :

1. Mulai
2. *Client* menyertakan *client id*, *client secret* dan *redirect uri* pada saat melakukan *load url*.
3. *Client* melakukan tindakan "*sign-in*" menuju aplikasi. Jika *client* sesuai maka proses *authentication* dilanjutkan menggunakan teknologi OAuth.
4. Proses *authentication* OAuth dilakukan untuk mengecek kesesuaian dari data *client*, Dengan melakukan otorisasi untuk menghadirkan halaman *Authorize App*.
5. Halaman persetujuan otorisasi pada halaman *Authorize App* merupakan halaman melakukan tindakan persetujuan otoritas untuk mendapat *authorization code*.
6. *Client* yang melakukan tindakan persetujuan akan di arahkan menuju halaman utama sistem aplikasi, ditandai dengan keluarnya *authorization code* pada layar.
7. Jika *client* tidak melakukan tindakan persetujuan maka sistem akan kembali kehalaman awal yang berarti *client* tidak dapat mengakses aplikasi utama.
8. Selesai.

4. IMPLEMENTASI

Pada gambar dibawah dapat dilihat beberapa tampilan yang telah diimplementasikan tentang penerapan mekanisme otentikasi menggunakan OAuth 2.0 dengan media berbasis android untuk *client*.

Pada penelitian ini disimulasikan dengan mengimplementasikan sebuah aktivitas *sign-in* menuju penyedia layanan yang dimana proses *sign-in* nantinya akan dilakukan oleh pengguna aplikasi dan pengguna melakukan proses persetujuan apakah pengguna tersebut menyetujui data dirinya dapat diakses oleh aplikasi atau tidak. Berikut adalah gambar simulasi *sign-in* yang dilakukan oleh pengguna pada aplikasi :

Gambar 4.1. Tampilan Awal Aplikasi Android

Ketika pengguna menekan tombol *sign-in* maka proses *authentication credential client* (*client_id* dan *client_secret*) diproses menggunakan teknologi *authentication* OAuth 2.0. Setelah itu maka dilanjutkan untuk proses *load url* berdasarkan semua komponen yang telah didefinisikan pada *client* untuk dapat masuk ke layanan penyedia API.

Gambar 4.2. Tampilan Sign-in OAuth

192.168.56.1/apioauth/index.php/oauth/?X-API-KEY=k408s8ggk000gsggsscwcwc00o8kg0ososg0w84k&response_type=code&client_id=2bfe9d72a4aae8f06a31025b7536be80&&client_secret=9d667c2b7fae7a329f32b6df17926154&scope=user&redirect_uri=http://192.168.56.1/apioauth/index.php/apis/

Gambar 4.3. Url load Aplikasi Android

OAuth 2.0 melakukan kebijakan memeriksa validnya data dari seorang *client* melalui *authorization server*, sehingga hasil akhir *client* yang terpercaya dapat menerima akses ke aplikasi dengan *authorization server* menghasilkan *authorization code* yang muncul pada http (*hypertext transfer protocol*). Apabila dijalankan pada browser akan muncul pada kolom URL. *Authorization Code* yang diberikan oleh *authorization server* adalah jika *client* melakukan tindakan persetujuan "Apakah Melanjutkan Otorisasi Pada Aplikasi ?".

Gambar 4.4. Tampilan Halaman Authorization Access Request

Pada gambar 4.5 yang muncul dengan kotak berwarna abu-abu di bawah adalah gambar saat *authorization code* dihasilkan yang terletak pada http (*hypertext transfer protocol*). Setelah itu kode tersebut dapat dimanfaatkan oleh *client* untuk mengakses data dari server.

Gambar 4.5. Tampilan Authorization Code

5. PENGUJIAN

Penerapan mekanisme *authentication* ini dapat dilakukan pengujian dan analisa kelemahan sistem yang dikembangkan dengan beberapa cara seperti yang telah dilakukan oleh penulis dan nantinya dapat ditarik kesimpulan berdasarkan hasil implementasi dan analisa yang telah dilakukan.

5.1 Pengujian Validasi

Pengujian ini untuk mengetahui tingkat kesuksesan dan keberhasilan dari sistem yang sudah dikembangkan berdasarkan sistem sebelumnya. Pengujian validasi dengan metode *black box* merupakan pengujian yang terfokus pada spesifikasi fungsional dari perangkat lunak, menemukan kesalahan dalam penggunaan struktur data, kesalahan data yang tidak valid, memeriksa dan memastikan hasil yang diinginkan apakah sudah sesuai atau tidak pada sistem.

Adapun skenario yang telah dilakukan dalam pengujian validasi sistem di tunjukkan seperti pada tabel 5.1 di bawah :

Tabel 5.1. Skenario Pengujian Validasi

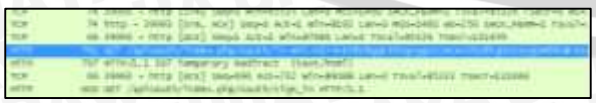
Kasus Uji	Nilai Masukan	Skenario Pengujian	Hasil Yang Diharapkan	Hasil Uji
Generate token	Salah	Belum mendaftarkan nama aplikasi untuk *token* yang baru.	Terdapat pesan "*please fill out this field*"	Sukses
	Benar	Syarat pendaftaran diisi dengan lengkap.	*Token* yang telah terdaftar akan langsung	Sukses

			masuk dalam database	
Sign-in	Salah	Pengguna mengisi form *sign-in* tidak sesuai dengan data yang ada	Muncul *dialog box* yang memberikan informasi bahwa data yang diisi tidak valid	sukses
	Salah	Pengguna tidak mengisi salah satu atau kedua *field* form *sign-in*	Muncul *dialog box* yang memberikan informasi bahwa *field* tidak boleh kosong	Sukses
	Benar	Pengguna mengisi form dengan lengkap dan benar sesuai data yang tersimpan	Pengguna akan diarahkan pada halaman *Authorize App Request*, dan jika menyetujui *authorize app akan mendapat authorization code*	Benar

5.2 Pengujian *Request* HTTP

Request HTTP dilakukan dengan menerapkan protokol HTTP pada aplikasi *client* saat akan mengakses layanan dengan melakukan percobaan pada aplikasi *client* yang dijalankan langsung pada *smarthphone* untuk dapat dilihat komunikasi yang terjadi ketika menggunakan proses *request* HTTP.

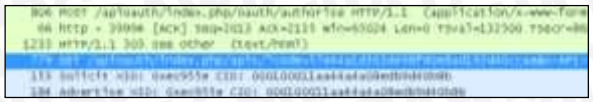
Ketika aplikasi *client* mulai dijalankan, maka disaat bersamaan juga dilakukan *capture* komunikasi yang terjadi pada saat adanya *request* yang datang dari aplikasi *client*. Dapat dilihat pada gambar dibawah menandakan bahwa penggunaan protokol HTTP pada proses *request* sebuah aplikasi masih dapat terlihat karena tidak adanya enkripsi yang diterapkan, sehingga masih ada kemungkinan pengguna lain bisa mengambil informasi yang didapat dari hasil capture komunikasi yang terjadi pada *wireshark* untuk digunakan kembali dalam melakukan proses *request*.



Gambar 5.1. Tampilan Request Pada Wireshark

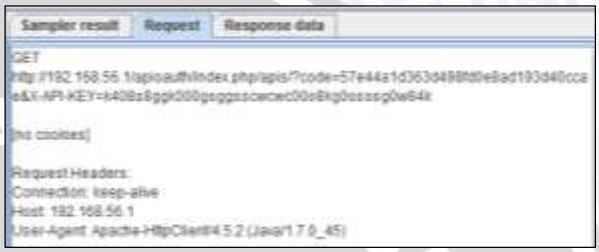
5.3 Penggunaan Kembali *Credential Client*

Tahap awal dalam pengujian ini adalah melakukan *capture* pada *wireshark* untuk melihat *credential client* yang telah dihasilkan dari proses otorisasi.



Gambar 5.2. Tampilan Authorization Code Pada Wireshark

Berdasarkan *authorization code* yang sudah dihasilkan tersebut, dilakukan percobaan dengan memanfaatkan *authorization code* yang sudah ada untuk mengakses layanan. Pada tahap ini simulasi dilakukan dengan menggunakan jmeter dalam proses mengakses url.



Gambar 5.3. Request Akses Menggunakan Jmeter

Gambar 5.4. Response Dari Request Akses

Pesan error diatas memberitahukan bahwa terdapat parameter yang tidak valid karena parameter yang digunakan adalah milik *client* yang sudah terotorisasi sebelumnya. Karena proses otorisasi tersebut harus dengan menyerahkan beberapa parameter yang menjadi bagian dari komponen OAuth untuk bisa mendapatkan akses ke layanan dan harus ditandai dengan persetujuan pengguna aplikasi yang sudah terdaftar pada server.

5.4 Pengujian *Decompile* APK

Hasil dari proses *decompile* APK yang sudah dilakukan dengan membuka file *MainActivity.java* yang dapat dilihat bahwa ketika file APK di *decompile* ternyata komponen-komponen OAuth yang menjadi parameter dalam proses meminta akses terhadap layanan masih dapat terlihat dan masih bisa dipergunakan kembali untuk membuat aplikasi yang serupa dengan memanfaatkan komponen-komponen tersebut. Seperti yang terlihat pada gambar dibawah.

Gambar 5.5. Hasil Decompile APK

6. KESIMPULAN

1 Dapat di implementasikan teknologi *authentication* OAuth 2.0 pada *web service* yang berperan dalam proses *authentication credential* milik *client* yang merupakan pengguna aplikasi pada aplikasi belajardisini.com. Melalui mekanisme *authentication* OAuth 2.0 yang melibatkan *authorization server*, berperan dalam menghadirkan halaman proses otorisasi aplikasi (*authorize app*). *Authorize App* yang disetujui oleh *client* akan dikembalikan lagi oleh *authorization server* dalam bentuk *authorization code* untuk mendapatkan hak akses menuju layanan. *Authorization code* yang dihasilkan setiap kali *client* berhasil melakukan aktivitas *sign-in* serta menyetujui permintaan otorisasi akan menghasilkan kode unik dalam bentuk *string* yang berbeda dengan yang sebelumnya.

2 Berdasarkan hasil pengujian sistem apabila dilihat dari sisi fungsional, sistem sudah berhasil dijalankan dengan memanfaatkan mekanisme *authentication* dengan sebuah fitur *sign-in* yang menggunakan teknologi OAuth 2.0, namun dilihat dari sisi keamanan *credential client* masih belum dilakukan penerapan mekanisme yang mengatur untuk proses mengamankan *credential client* tersebut.

7. DAFTAR PUSTAKA

[1] Deviana, Hartati. (2011). *"Penerapan XML Web service Pada Sistem Distribusi Barang"*. Jurnal Generic, Vol. 6, No. 2, Juli 2011, pp. 61-70.

[2] Wellem, Theophilus. (2009). *"PERANCANGAN PROTOTYPE APLIKASI MOBILE UNTUK PENGAKSESAN WEB SERVICE"*. Seminar Nasional Informatika 2009.

[3] Castillo, P. A. (2011) "*SOAP vs REST : Comparing a master-slave GA implementation*". Neural and Evolutionary Computing Cornell University Library. [Online]: http://arxiv.org/abs/1105.4978. Diakses pada 28 November 2015.

[4] Nurseotiv, N., Paulson, M., Reynolds, R., & Izurieta, C. (2009) "*Comparison of JSON and XML Data Interchange Formats: A Case Study*". ICSA 22nd International Conference on Computer Application in Industry and Engineering.

[5] Owasp. (2013). *"OWASP Periodic Table of Vulnerabilities - Weak Authentication Methods"*. [Online]: https://www.owasp.org. Di akses pada 28 September 2015.

[6] Agarwal, Tejaswi dan Mike Leonetti. (2013). "*Design and Implementation of an IP based authentication mechanism for Open Source Proxy Servers in Interception Mode*". Advanced Computing: An International Journal (ACIJ), Vol.4, No.1, January 2013.

[7] Fielding, R. T. (2000). "*Architectural Styles and the Design of Network based Software Architectures*". Doctor of Philosophy Dissetation in Information and Computer Science, University of California, Irvine.

[8] Burke, Bill. (2014). "*RESTful Java with JAX-RS 2.0*". [Online]. Google Books. Diakses pada 29 November 2015.

[9] Hammer-Lahav , E. (2010). "*RFC 5849: The OAuth 1.0 Protocol*". ISSN: 2070-1721.

[10] Json. (2014). "*JSON*". [Online] : http://www.json.org. Diakses pada 29 November 2015.

[11] W3. (2004). "*Web Services Architecture*". [Online] : http://www.w3.org/TR/wsSarch/. Diakses pada 28 November 2015.

[12] W3. (2005). "*W3C Document Object Model*". [Online] : http://www.w3.org/DOM. Diakses pada 28 november 2015.

[13] Asana. (___). *"Authentication"*. [Online] : https://asana.com/developers/documentation/getting-started/auth. Diakses pada 29 September 2015.

[14] Kaur, G, Aggarwal, D. (2013). "*A Survey Paper On Social Sign On Protocol OAuth 2.0*". Journal. Blue Ocean Research Journals, Punjab.

[15] D. Hardt, Ed. (2012). "*The OAuth 2.0 Authorization Framework*". ISSN: 2070-1721.

[16] S, Chiragsh. (2012). "*OAuth 2*". [Online] : https://code.google.com/p/google-api-php-client/wiki/OAuth2. Diakses pada 29 November 2015.

[17] Developers, Google. (2016). "*Using OAuth 2.0 to Access Google APIs*". [Online] : https://developers.google.com/accounts/docs/OAuth2. Diakses pada 29 November 2015.

[18] Brail, Graig & Ramji, Sam. (2012). "*OAuth - The Big Picture*". [Online] : http://apigee.com. Diakses pada 29 November 2015.

[19] Parecki, Aaron. (2012). "*OAuth 2 Simplified*". [Online] : http://aaronparecki.com/articles/2012/07/29/1/oauth2-simplified. Diakses pada 29 November 2015.

[20] Byod, Ryan. (2012). "*Getting Started With OAuth 2*". [Ebook]. O'Reilly Media.