OAuth protocol for CERN Web Applications

September 2016

Author:
Emil Kleszcz (emil.kleszcz@gmail.com)

Supervisors:
Luis Rodriguez Fernandez
Nicolas Bernard Marescaux

CERN openlab Summer Student Report 2016

15 years CERN openlab

Project Specification

The CERN Authentication service has recently increased his offer of SSO protocols/frameworks with OAuth2 standard. The purpose of the work is to explain the need for an authorization standard and explain how does OAuth2 protocol addresses our requirements.

OAuth is an open standard for authorization which provides a secure access to server resources that belongs to a user. As a wide known and relatively new standard (2012) is used by big players in the industry and is compatible and designed to work with different types of clients including native and web applications.

The aim of this project is to study how CERN Web applications can make use of OAuth 2.0 protocol and compare it with the current standard SAML2. Moreover investigate how OAuth 2.0 protocol works with Single Sign-On and integrate this solution into current infrastructure at CERN.

Abstract

The purpose of this report is to document the project that I was working on during openlab summer student program in IT-DB-IMS section.

In the first chapter of this paper you can find introduction into the project. In this part I have explained what is Single Sign-On and how it works as well as listed all the use cases for authorization protocols at CERN. In the next chapter you will find project overview with the explanation of two protocols and comparison of them. SAML2 is an authorization protocol widely used at CERN and OAuth2 is the subject of the project. In system analysis chapter, all the scenarios for OAuth2 protocol have been presented. In the next section you will find three different approaches that I have implemented to test OAuth2 protocol. Last two parts of the report are conclusions and future work.

Table of Contents

1 Introduction 7

1.1 CERN SSO 7

1.1.1 What is SSO? 7

1.1.2 Why a CERN Authentication SSO? 7

1.1.3 Authentication methods 7

1.1.4 Technical background and registration 7

1.1.5 ADFS2 8

1.2 Middleware services (IT-DB-IMS section) 8

2 Project overview 8

2.1.1 SAML2 limitations 8

2.1.2 Objectives 9

2.2 Current implementation SAML2 9

2.2.1 SAML2 configuration for WebLogic 10

2.2.2 SAML Assertion 11

2.3 OAuth2 protocol 11

2.4 Comparison 12

3 System analysis 13

3.1 Scenarios 13

3.1.1 Oracle Cloud and WebLogic 14

3.1.2 Tomcat (MWOD) 14

3.1.3 APEX (ORDS) 14

4 Implementation 15

4.1 Client Server web application 15

4.2 Simple Client 15

4.3 Indico API 16

5 Conclusions 17

6 Future work 18

References 18

Acknowledgments

I would like to thank both my supervisors Luis Rodriguez Fernandez and Nicolas Bernard Marescaux fort their support throughout the two months of my summer student internship at CERN, keeping me going when times were tough, asking insightful questions, and offering invaluable advices. In particular, I would like to thank Luis for introducing me smoothly into the project and showing me joyful side of life at CERN. I would also like to express acknowledgments to all the members of the *Infrastructure and Middleware Services* section, for the possibility of being part of such a great team from the first day. I am indebted to all of you for your invaluable support.

1 Introduction

The evolving world of Information Technology changed the way people socialize, share information and access applications. Nowadays it become necessary to exchange data between many applications in a secure way. Such process involves user authentication and resource authorization. According to that, many authorization protocols have emerged to address the integration problem.

The objective of this project is to study how CERN Web applications can make use of OAuth 2.0 protocol and compare it with the current standard SAML2. Moreover investigate how OAuth 2.0 protocol works with Single Sign-On and integrate this solution into current infrastructure at CERN.

1.1 CERN SSO

1.1.1 What is SSO?

Single sign-on (SSO) is a session and user authentication service that permits a user to use one set of login credentials to access multiple software systems. The service authenticates the end user for all the applications the user has been given rights to. SSO is also helpful for logging user activities as well as monitoring accounts. [1]

1.1.2 Why a CERN Authentication SSO?

The aim of the CERN SSO solution is to be the central point of authentication for all of the CERN web applications. The final user being authenticated only in one system, will be able to access to other applications that have been registered in the SSO system. The benefits of using SSO are for end users (only one set of credentials), system security and developers who don't have to care for authentication system anymore.

1.1.3 Authentication methods

CERN Authentication offers three ways to authenticate user:

- Classic Forms based authentication: user types credentials and confirm them.
- Certificate authentication: use CERN Certificate or SmartCard to authenticate.
- Windows Integrated authentication reuse the current Windows session for authentication, available only for Internet Explorer. [2]

1.1.4 Technical background and registration

The system is based on two main concepts:

- **Identity Provider** (IdP) - service that maintains the users information and provides authentication for them.
- **Service Provider** (SP) - component that provides resources such as Web pages to users relying on their authentication in an IdP.

The CERN implementation of the IdP is based on Active Directory Federation Services (ADFS2). ADFS is an extension of the Microsoft Active Directory central authentication database. For the

SP Shibboleth (CGI, JAVA), Oracle WebLogic SAML2 (JAVA) and Spring Security are used. [3]

The SSO solution for CERN web based applications allows any web application to authenticate users and retrieve their information including their group membership to manage authorizations. To use CERN SSO in the application that are not centrally hosted we need to register a new application on SSO Management. All centrally hosted websites are SSO enabled. [4]

Recently, a new OAuth2 authorization service has been offered for CERN clients and applications. You can register your client using the SSO management site. [5] You can also test how it works on test site configured to use the OAuth system at CERN. [6]

The CERN SSO classic service is suitable for the web profile. [7] For command line scripts you can use the *cern_get_sso_cookie*. [8]

1.1.5 ADFS2

CERN uses **ADFS2** to share users information in a secure way between applications. It creates relying relationship between systems and ADFS2 so-called **Federation**. ADFS consist of

- **Active Directory** for managing entities (such as users, machines etc.) information in domains.
- **Services** such as
  - Federation services or STS which server as back-end to perform many authentication operations.
  - Login services (front-end) serve as GUI to log in into CERN account.

In the CERN SSO ADFS2 works as Identity Provider.

1.2 Middleware services (IT-DB-IMS section)

IT-DB-IMS section manages application middleware infrastructure and provides installation, monitoring, configuration and patching of the middle tier application components such as Apache web server, WebLogic server and Apache Tomcat among many others.

We have the following use-cases for authorization protocol in IT-DB-IMS section:

- **WebLogic Applications (Java, APEX)** - applications deployed in Oracle WebLogic such as edh.cern.ch or e-groups.cern.ch and APEX applications
- **Oracle Cloud** - JEE application deployed in Oracle Cloud
- **Middleware On Demand (MWOD)** - central server infrastructure for deployment of Java web applications at CERN. The application server used for this platform is Apache Tomcat. There are two instances of the system: production [9] and test. [10]

2 Project overview

2.1.1 SAML2 limitations

The existing protocol SAML2 have some limitations for users and developers. Below you can find some of them:

- If any of the participants (Service Providers) in the session fail, the whole logout process fails. The cause is logout implementation in ADFS2 and the solution is to delete cookies or close the web browser.
- User clicks a back button and gets a 403 error. The cause is the history of a browser which is "polluted" with the SAML2 requests.
- In the standard configuration we can use only one SAML2 cluster per domain in WebLogic. WebLogic force us to use /saml2 path for federation services endpoints. In order to solve it, our section introduced one Apache web server directive that specifies the string trimmed of the original URL before the request is forwarded to WebLogic Server. [11]
- The limitation of the HTTP POST binding for native mobile apps. Mobile applications don't have access to the HTTP POST body. They only have access to the URL use to launch the application. This means that we can't read the SAML token. No SAML token, no authenticated user. Workaround can be embedded web view and scraping the HTML of the page and extract out the SAML token. [12]

You can find more limitations in the references section. [13]

2.1.2 Objectives

- Alternative solution for mobile applications that is easy to develop and maintain.
- Get rid of all the limitations with SAML2.
- Use authentication servers such as Facebook for logging into non-public CERN web application to make users life more pleasant and give them alternative approach to log in. [14]
- Study a new solution, the OAuth2 protocol that can solve the limitations and provide these features.

2.2 Current implementation SAML2

The CERN SSO system is built over two standards, **WS-Federation Language** and **SAML2** (Security Assertion Markup Language). Both systems exchange XML messages between the SP's and IdP's entities.

The Security Assertion Markup Language (SAML) standard defines a framework for exchanging security information between online business partners. [7] SAML is an XML-based open standard data format for exchanging authentication and authorization data between parties. SAML encompasses profiles, bindings and constructs to achieve: SSO, Federation and Identity Management.

SAML has three main players that participate in the authorization process.

- **Service Provider** (SP)
- **Client** (web browser, scripts, etc.)
- **Identity Provider** (IdP)

Below, on *Figure 1* you can find most common SAML flow.

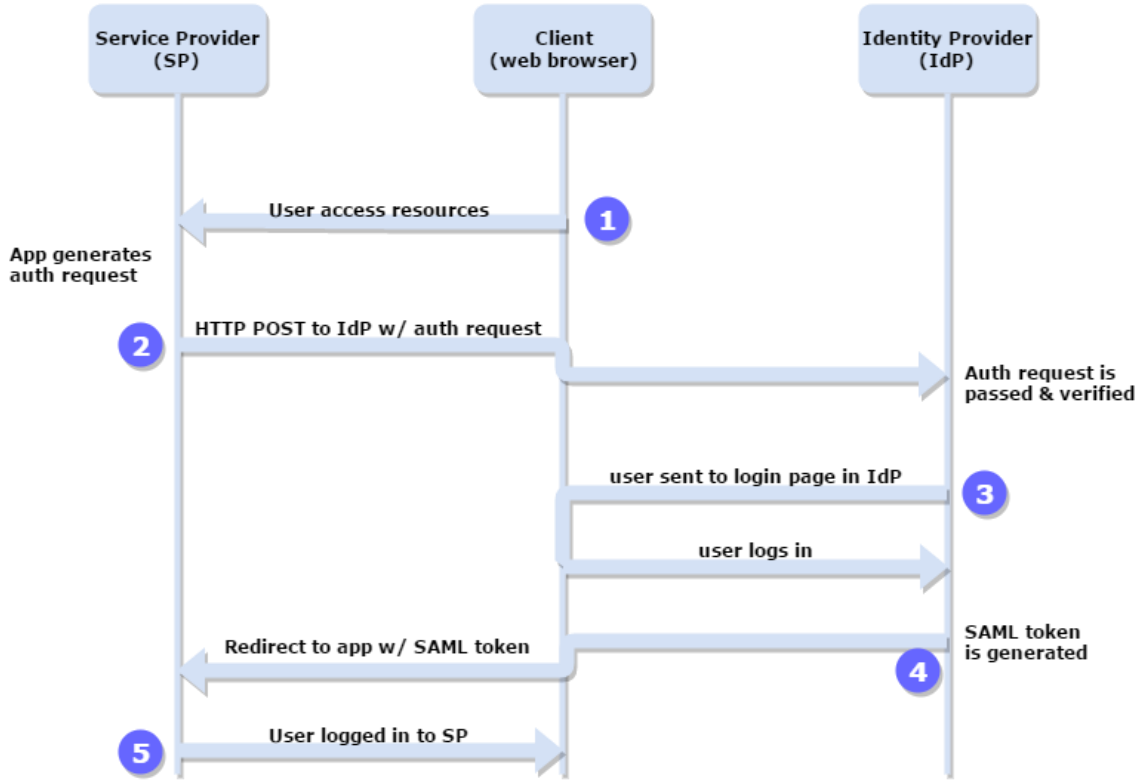

Figure 1 - Graphical representation of the SAML protocol flow.

1. A user opens a web-browser and goes to a web page provided by SP which stores resources. Web page doesn't handle authentication itself.

2. In order to authenticate the user, webpage constructs a SAML Authentication request, signs it, optionally encrypts and encodes it. Next, it redirects the user's web browser to IdP in order to authenticate. The IdP receives the request and process it.

3. With a valid Authentication request the IdP will present the user with a login page.

4. Once the user has logged in, the IdP generates a SAML token that includes identity information about the user. After that IdP redirect the user back to the Service Provider with generated token.

5. SP verifies the token and logs the user into the system, presumably with a cookie and a session.

At the end of the whole process the user is logged in and can access all the resources from SP.

2.2.1 SAML2 configuration for WebLogic

Oracle WebLogic provides a module (Web application) that implements the SAML Web Browser SSO Profile. The module allows WebLogic to work both as a IdP and SP, but we use it only as SP. We focus on Web Single Sign-On Use Case which is described as the user accessing a secure resource. Once the configuration for SAML2 module has been done, we can secure a web application deployed on the managed server. [3] [15]

2.2.2 SAML Assertion

SAML Assertion is a signed list of user attributes such as name, email etc. **Security Token Service** (STS) consumes SAML2 assertions and produce X.509 credentials in return. This functionality is based on IOTA- CA (Identifier-Only Trust Assurance Certification Authority) that issues short-living (days) X.509 certificates. At CERN we can get such certificates from CERN CA. [16]

2.3 OAuth2 protocol

OAuth2 is open standard for authorization which provides a secure access to server resources that belongs to a user. OAuth2 is compatible with different types of clients such as web server, browser based application and native applications.

OAuth 2.0 standard defines four following roles:

- **Resource owner** is the user who authorizes an application to access its account.
- **Client** is the application (web, native, mobile etc.) that wants to access the user's resources.
- **Resource server** the web server from which you are trying to get the resources.
- **Authorization server** this is the server that owns the user identities and credentials. It verifies the identity of the user.

On the *Figure 2* you can see the authorization code flow for grant code authorization in OAuth2.

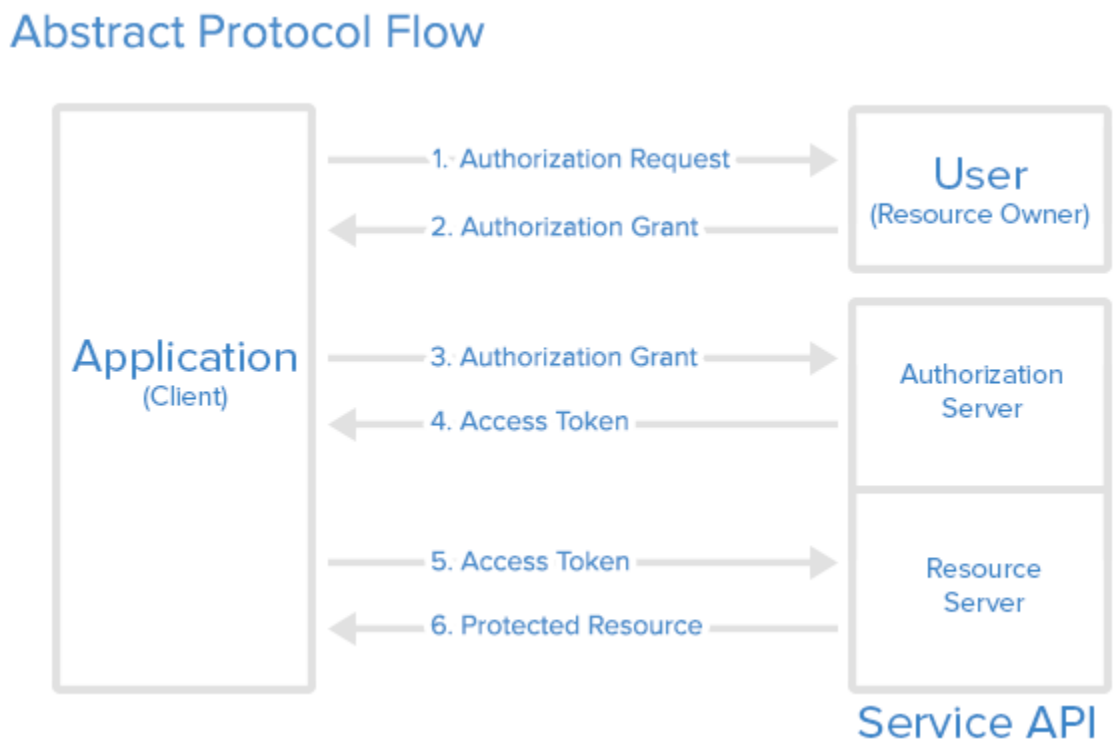

Figure 2 - Authorization code flow for OAuth2 framework and grant code authorization. **[17]**

Here is a more detailed explanation of the steps in the diagram:

1. The application requests authorization to access service resources from the user.
2. If the user authorized the request, the application receives an authorization grant.
3. The application requests an access token from the authorization server (API) by presenting authentication of its own identity, and the authorization grant.
4. If the application identity is authenticated and the authorization grant is valid, the authorization server (API) issues an access token to the application. Authorization is complete.
5. The application requests the resource from the resource server (API) and presents the access token for authentication.
6. If the access token is valid, the resource server (API) serves the resource to the application.

The actual flow of this process will differ depending on the authorization grant type in use. This is the flow for authorization code grant which is the most commonly used because it is optimized for server-side applications, where source code is not publicly exposed, and Client Secret confidentiality can be maintained. [18]

Bearer token used in the flow is security token such as **access token** or **refresh token** intended to access protected resources. **Authorization code** is suited for server-side clients and works only when client is registered with authorization code grant. **Authorization grant** is a well defined set of steps to obtain access token from an authorization server. Others authorization grants offered by OAuth standard has been presented below:

- Authorization code grant
- Implicit grant (suitable for mobile applications)
- Password grant
- Client credentials grant
- Refresh token grant

2.4 Comparison

SAML deals with XML as the data construct or token format. OAuth tokens can be binary, JSON or SAML. [19] SAML has bindings that use HTTP such as HTTP POST or HTTP REDIRECT. However there is no restriction on the transport format. It's also possible to use SOAP or JMS to send SAML tokens or messages. On the other hand, OAuth uses HTTP exclusively and is designed for internet scale. Even though SAML was designed to be applicable openly, it is typically used in Enterprise SSO scenarios. OAuth has been designed for use with applications on the internet, primarily for delegated authorization of web resources. You can use both protocols together. SAML can be use for authentication and SAML token can be used as OAuth bearer token in HTTP bearer header to access protected resources. [20] In *Table 1* you can find main differences between two standards.

	OAuth v2.0	SAML v2.0
Message format	Binary, JSON or SAML	XML
Transport	HTTP exclusively	COOKies / Bindings: HTTP POST, HTTP REDIRECT etc.
Scope	Internet Scale	Enterprise / SSO scenerios
Use case	Authorization of internet resources /mobile devices	Typically use for enterprise partners/SSO/centralized identity secure
Both SAML and OAuth	SAML token as an Oauth bearer token in the HTTP bearer header to access protected resources	For authentication
Support for SSO	Web SSO only	Yes
Token	Yes (JSON Web Token)	Yes (SAML XML Token)

Table 1 - Comparison between two authorization standards SAML2 and OAuth2.

3 System analysis

3.1 Scenarios

In this section you can find scenarios for all the use cases that can be used with OAuth2 protocol. In all the scenarios ADFS serves as authentication server (Identity provider) and enables SSO.

3.1.1 Oracle Cloud and WebLogic

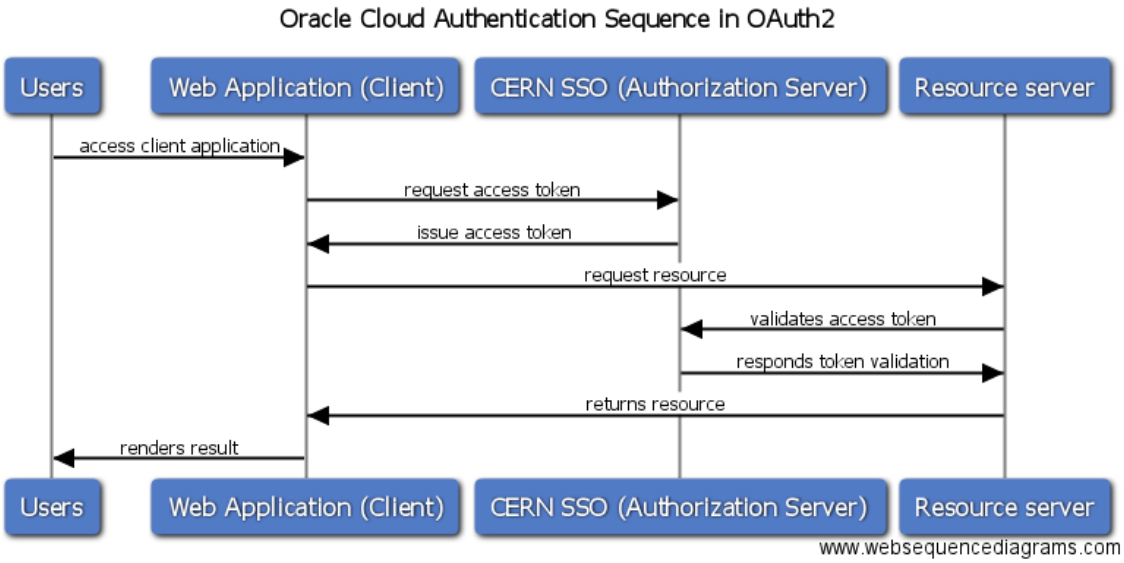

Figure 3 - Oracle Cloud and WebLogic authentication sequence diagram for OAuth2 protocol.

In this use case client web application plays the role of OAuth client. The application will request access to resources owned by the end user or access service API. Resource server is a RESTful webservice that represent the protected resources or API. Valid access token is required in order to server information back to the client. CERN SSO with ADFS is responsible for user authentication (issuing and validating OAuth token).

3.1.2 Tomcat (MWOD)

Java middleware on demand applications are deployed on Tomcat and can be requested on CERN Web Services[1]. In this case configuration of OAuth2 on the client side (web application) can be done with some OAuth protocol implementation. The good examples for Java implementations are Apache Oltu and Spring Security framework.

3.1.3 APEX (ORDS)

Oracle Application Express uses Oracle REST Data Services (ORDS). RESTful services support two kinds of authentication:

First Party Authentication. This is authentication intended to be used by the party who created the RESTful Service, enabling an APEX application to easily consume a protected RESTful Service. The application must be located with the RESTful Service, i.e. it must be located in the same Oracle Application Express workspace. The application must use the standard Oracle Application Express authentication.

Third Party Authentication. This is authentication intended to be used by third party applications not related to the party who created the RESTful Service. Third party authentication relies on the OAuth 2.0 protocol. [21]

[1] https://webservices.web.cern.ch/webservices/

4 Implementation

In this section, I present three different implementations of applications that I have used to test OAuth2 protocol. All the projects and more information you can find on my Github[2] repository and in my articles on IT-DB group blog[3].

4.1 Client Server web application

First Proof of Concept (PoC) consists of web application with both authorization and resource server. For this example I have used JAX-RS specification. There are four REST resource endpoints:

- **End User Authorization** - Authorization of the app without any authentication (to make it simpler) In response, the authorization code is redirect back to application.
- **Redirect** - Retrieve the response from authorization server and return JSON with headers and parameters (to obtain authorization code).
- **Token** - Intended to access token, check user credentials, check grant type and assigns new token.
- **Resource** - Validate token and return resources for a given user.

On Figure 4 all the classes from the project has been presented. In Database class are stored authorization codes and tokens that belongs to all the users. Common class store fake constants for user authentication and resource authorization such as client id and client secret.

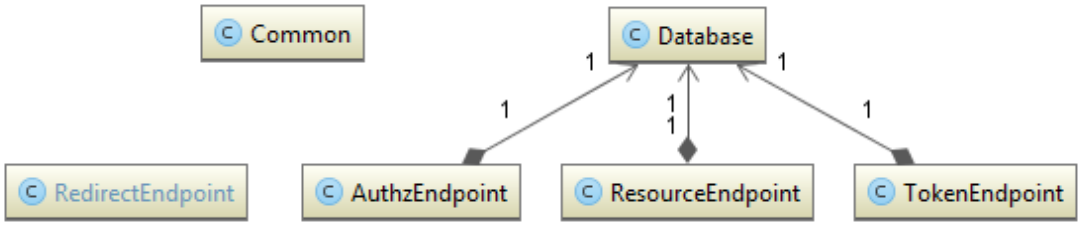

Figure 4 - Class diagram from Client Server project.

In order to test the web application I have used **Arquillian** framework. With that library I have implemented simple integration tests for the classes that are executed inside servlet container in **GlassFish4** application server. For authorization I have used **Apache Oltu** which is OAuth 2.0 protocol implementation in Java. It's very easy to use, and just require one dependency in project build manager to work.

4.2 Simple Client

I have developed an application to test OAuth2 client on the CERN SSO Management. Following steps from [22] I have registered my client on Authorization Server [23] defining client id, redirect uri and application homepage parameters.

[2] https://github.com/tr0k
[3] http://db-blog.web.cern.ch/

In order to register client it is required to define redirect uri in the cern.ch domain over HTTPS protocol. To obtain access token I set up Apache server to work on the port I have defined for redirection uri over HTTPS.

I am able to receive access token for implicit grant and authorization code for authorization code grant, however I cannot access the resources and CERN API. I receive 411 HTTP response code from the server. Example provided by CERN to test OAuth Client for CERN OAuth Resource Service works fine. [24]

Another case for the client using developers.facebook.com works correctly. With this simple example you can access information from your Facebook profile. First step is to register web application on the Facebook website and define redirect URL to obtain bearer token.

4.3 Indico API

This application is intended to test Indico API. Indico is a general-purpose event management web-based solution. Indico is used every day at CERN to manage more than 300 000 events of different complexities and 200 meeting and conference rooms. All major LHC experiments at CERN adopted Indico as their tool of choice for meeting organization and information sharing. [25] [26]

It includes a full-blown conference organization workflow as well as tools for meeting management and room booking. It provides also integration with video conferencing solutions. Indico is free software, licensed under terms of the GNU General Public License (GPL) v3. It is currently in production at CERN, as well as in many other scientific institutions around the world. Indico was created at CERN, originally as part of the European Union's InDiCo Project. [27]

Indico API provides powerful REST API that allows for easy retrieval of information such as events, room booking, user information as well as managing conference files.

In our case, Indico API serves as a resource server. In order to obtain authorization code the application asks user to grant permissions to use Indico API (*Figure 5*). Application can be either deployed on the Tomcat server on the middleware on demand infrastructure or be native and use only MWOD as redirect uri to obtain access token. CERN OAuth is an authentication server that server as identity manager and we need to log in through CERN SSO login site provided based on ADFS.

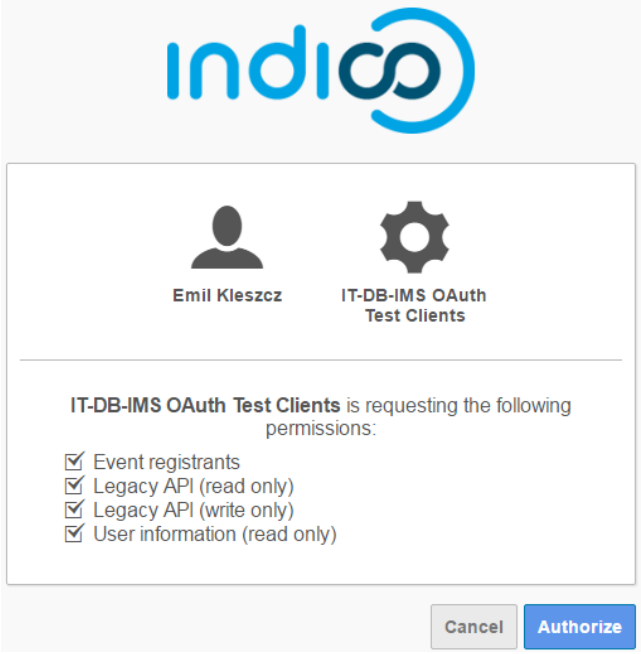

Figure 5 - Screen with option to grant permissions for the client.

Indico API offers 4 different scopes that we define in authorization request. [28]

- *read:user* (user information - read only
- *read:legacy_api* (legacy API - read only)
- *write:legacy_api* (legacy API - write only)
- *registrants* - (event registrants)

For now, this project works and you can use Indico API as authenticated user. Authorization is based on OAuth2 protocol. For instance, you can get information about a room reservation. Below on the Listing 1 you can see how to get any protected resources with Apache Oltu.

```
public OAuthResourceResponse getProtectedResources(OAuthClient
oAuthClient, String accessToken, String url)
throws OAuthSystemException, OAuthProblemException {

    OAuthClientRequest bearerClientRequest = new
    OAuthBearerClientRequest(url).setAccessToken(accessToken)
    .buildQuesryMessage();

    return oAuthClient.resource(bearerClientREquest, OAuth.HttpMethod.GET,
    OAuthResourceResponse.class);

}
```

Listing 1 - Getting protected resources from Indico API with OAuth2 protocol.

5 Conclusions

Below have been listed all the conclusions after my research on the OAtuh2 protocol and implementation of the Proof of Concepts.

- OAuth2 is very simple and more standardize solution to implement.
- We can use OAuth with WebLogic and CERN SSO.
- Cern SSO enables registration of OAuth2 clients.
- OAuth is wide known standard used by big players in the industry such as Google.
- There are the limitations that users and developers face with SAML2 protocol at CERN.
- OAuth is recommended especially for mobile applications (lightweight and no workarounds like for HTTP Bindings).
- You can easily revoke session and invalidate ad hoc access token on the Authorization Server and disable it from further access to the Resource Server.
- Both standards can work as complementary solutions.
- SAML token contains the user identity information and a digital signature, OAuth do not provide it out of the box and not require digital signature by default.
- Many different implementation of OAuth2 exists and are incompatible with each other.

Summarizing, OAuth2 is much easier in implementation than SAML2 and is suitable for mobile applications. Provides option to use other Identity Providers such as Google which is alternative option for users to log in into a web page. However, for many applications at CERN SAML2 is enough for web applications and introducing this protocol into existing infrastructure is challenging.

6 Future work

There is some further work that can be done in this project. It covers testing OAuth2 protocol on the Oracle Cloud and implementing Indico API with Java servlets or JAX-RS specification. There are also some issues with the client registered in SSO Management that have to be solved. So far I couldn't get protected resources after I have obtained access token. There is also lack of documentation on OAuth2 protocol at CERN. Further, I suggest to compare and test other authorization solutions, for instance Auth0 platform[4]. Also creation of a mobile client would be a good idea to test the OAuth2 and SAML2 protocols.

References

1. **Rouse, Margaret.** [Online] http://searchsecurity.techtarget.com/definition/single-sign-on.

2. [Online] https://espace.cern.ch/authentication/CERN%20Authentication%20Help/Home.aspx.

3. **Rodriguez, Luis.** Weblogic Single Sign On Integration. [Online] https://twiki.cern.ch/twiki/bin/view/DB/CERNonly/WeblogicSSOintegration.

4. [Online] https://espace.cern.ch/authentication/default.aspx.

5. [Online] https://espace.cern.ch/authentication/CERN%20Authentication/OAuth.aspx.

6. [Online] https://test-oauth.web.cern.ch/.

7. Security Assertion Markup Language (SAML) V2.0 Technical Overview. [Online] http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-cd-02.html.

[4] https://auth0.com

8. Linux @ CERN. [Online] http://linuxsoft.cern.ch/cern/slc53/i386/yum/extras/repoview/cern-get-sso-cookie.html.

9. [Online] https://mwctl.cern.ch/mwctl/.

10. [Online] https://mwctl-test.cern.ch/mwctl/.

11. *Oracle® Fusion Middleware Using Oracle WebLogic Server Proxy Plug-Ins 12.1.2.* [Online] https://docs.oracle.com/middleware/1212/webtier/PLGWL/plugin_params.htm.

12. **Dennis, Zach.** Choosing an SSO Strategy: SAML vs OAuth2. [Online] 09 05 2013. https://www.mutuallyhuman.com/blog/2013/05/09/choosing-an-sso-strategy-saml-vs-oauth2/.

13. *Weblogic as a Service Provider for CERN Web Applications:.* **Luis Rodriguez Fernandez, CERN IT-DB-IMS.** 2013.

14. **Team, CERN Single Sign-On Team & the Computer Security.** *Revolution when logging into CERN: the new single sign-on portal.* [Online] https://cds.cern.ch/journal/CERNBulletin/2013/05/Announcements/1507262?ln=en.

15. **Rodriguez, Luis.** Weblogic Saml2 Service Provider Configuration. [Online] https://twiki.cern.ch/twiki/bin/view/DB/CERNonly/WeblogicSAML2_SPconfiguration.

16. *WebFTS as a first WLCG/HEP FIM pilot.* **Andrea Manzi, Andrey Kiryanov, CERN IT-SDC.** 2015.

17. **Anicas, Mitchell.** An Introduction to OAuth 2. [Online] https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2.

18. **Kleszcz, Emil.** Java web application based on OAuth2. [Online] http://db-blog.web.cern.ch/blog/emil-kleszcz/2016-08-java-web-application-based-oauth2.

19. *OAuth Bearer Tokens.* [Online] https://docs.jboss.org/author/display/PLINK/OAuth+Bearer+Tokens.

20. **Saldanha, Anil.** *SAML vs. OAuth: Which One Should I Use?* [Online] 13 11 2013. https://dzone.com/articles/saml-versus-oauth-which-one.

21. **Oracle.** *Oracle REST Data Services Documentation* . [Online] http://www.oracle.com/technetwork/developer-tools/rest-data-services/documentation/index.html.

22. **CERN.** *CERN Authentication documentation.* [Online] https://espace.cern.ch/authentication/default.aspx.

23. **CERN SSO Management.** *Register a new OAuth Client to use CERN Authentication.* [Online] https://sso-management.web.cern.ch/OAuth/RegisterOAuthClient.aspx.

24. **CERN.** *Authorization Code Grant Client.* [Online] https://test-oauth.web.cern.ch/.

25. *Indico.* [Online] http://indico-software.org/.

26. **Indico Team.** *Indico's HTTP Export API.* [Online] http://indico.readthedocs.io/en/latest/http_api/.

27. *Indico API Github Repository.* [Online] https://github.com/indico/indico.

28. *Indico Github repository - class with scopes.* [Online] https://github.com/indico/indico/blob/master/indico/modules/oauth/models/applications.py#L31.

29. **Emmanuel Ormancey, IT/IS.** Single sign-on facilitates authentication at CERN. [Online] http://cerncourier.com/cws/article/cnl/31982.