CLARIN / BiG Grid Development Status Report

User Delegation in the CLARIN Metadata Infrastructure: *connecting the component registry and ISO-DCR*

Part II - Implementation

version A004; author Willem van Engen

23 December 2011

Introduction

The European project CLARIN[1] is developing a research infrastructure for e-Humanities based on a service oriented architecture. The vision of CLARIN is to have a collection of web services, accessible to the user via a number of (web) portals. Web service owners may want to restrict usage based on who is using it, and web services may need to call other web services, which can be protected as well (see Figure 1). This means an authentication and authorization infrastructure (AAI) is needed that supports this.

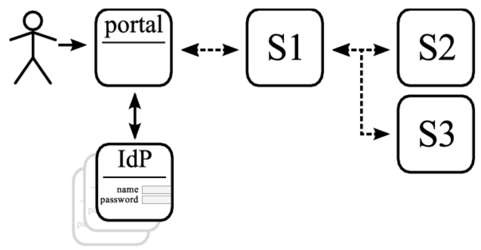

Figure 1: An example service invocation. After authenticating with an identity provider (IdP), the user invokes service S1 from within a portal. In turn S1 invokes S2 and S3 with the user's permissions.

The design of a security infrastructure has been the topic of the workshop “security for web services”[2] organised by BiG Grid[3] in April 2010. The next step was to consider a simple but practical use-case to gain some experience. This was found in a CLARIN web application (component metadata registry/editor[4]), which needs to access private data present in another web application (the ISOcat data category registry[5]). BiG Grid has investigated a number of technologies that would enable this[6], leaving a setup based on OAuth 2 and X.509 certificates as options (see the full report[6] for details).

Since CLARIN has indicated a preference for an OAuth 2 based solution, the focus lies there, with the possibility of X.509 certificates as a fallback.

OAuth 2

OAuth[7] is an open protocol for secure API authorization, used widely on the world wide web. Version two of the standard[8] provides the flexibility needed for our use-case.

To access a service (resource) which is protected by OAuth 2, one needs an access token. This is provided by an authorization service (AS). The standard specifies various ways in

1 http://www.clarin.eu/
2 http://agenda.nikhef.nl/conferenceDisplay.py?confId=993
3 The Dutch national grid initiative, http://www.biggrid.nl/
4 http://www.clarin.eu/cmdi
5 http://www.isocat.org/
6 http://www.nikhef.nl/pub/projects/grid/gridwiki/images/6/66/Clarin-security_for_web_services-research-report010.pdf

which an access token can be obtained[9], depending on the use-case.

One way is to redirect the user's web browser to a web page on the authorization service which asks for authentication and consent (see Figure 2). On success, an access token is returned. This is a simple option that provides a good user-experience when a single authorization service is present.

Another way is to use a token provided by a trusted authorization endpoint (AE), depicted in Figure 3. For example, the user authenticates with the authorization endpoint and receives a token (authorization code) with which it can obtain access tokens from one or more authorization services.

It is also possible to use assertions or custom ways for obtaining an access token from the authorization service. This is mostly similar to the authorization code scenario, but usually requires some additional client-support.

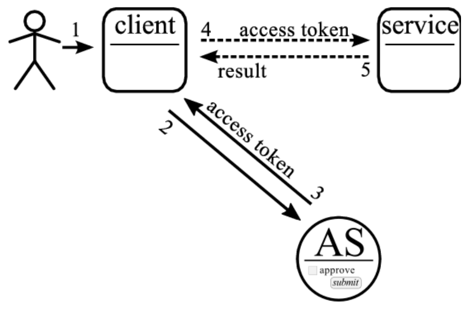

Figure 2: OAuth 2 scenario where the authorization service (AS) asks the user for confirmation directly. This involves a browser redirect to each TE involved.

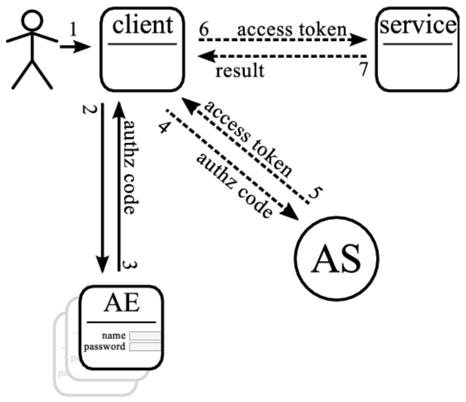

Figure 3: OAuth 2 scenario where an authorization endpoint (AE) passes an authorization to the AS. This involves a browser redirection only to the AE.

OAuth2lib

OAuth2lib[10] is a software package that provides the components for building such a security infrastructure, combined with SAML single sign-on[11] (a requirement for CLARIN).

Figure 4 shows the flow that OAuth2lib implements. The user (1) accesses the portal and logs in using a SAML identity provider (not shown), (2) resulting in a security assertion about who the user is. (3) The assertion is exchanged for an (4) access token at the AS, which is used to (5) access the service and (6) obtain the result.

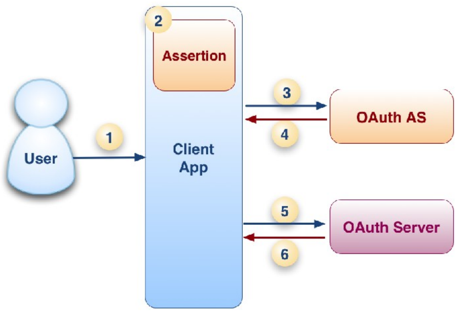

Figure 4: OAuth2lib components and their interaction.

As part of the implementation phase, an OAuth2lib test setup was created. There were

7 http://oauth.net/
8 Which is currently in draft, but already used in production for example by Facebook, Github and Google.
9 These are called "OAuth flows"; http://www.independentid.com/2011/03/oauth-flows-extended.html
10 http://www.rediris.es/oauth2/
11 A solution for web-based authentication using web redirects, that is widely available in research and education. Each institution has an identity provider (IdP) that authenticates their own users (though some institutions work together in a single IdP).

some initial problems, but with some help of its developers and debugging, a functional test setup was realised.

Issues

During the installation and configuration, more details about OAuth2lib's design became clear. This turned out to be not exactly as expected. The code shows that the authorization service accepts authentication information from the client (portal) without any verification. Technically, the SAML assertion is converted to a simple web token[12] without a signature. Whatever the client provides to the authorization service, is accepted.

This means that the portal is fully trusted by the AS, and subsequently by the service. I'm waiting for the developers to confirm this. Full client trust is not what was envisioned for our use-case.

Another issue is that all clients need to communicate with the identity providers using SAML, support of which is not present by default in most OAuth 2 client libraries. While this is provided by OAuth2lib for PHP-based web portals, it would be a barrier to the use of other OAuth 2 flows. This would affect, for example, in-browser web applications, and software written in other programming languages.

Solution A: Authorization service that authenticates too

Both problems can be solved by moving authentication from the portal to the authorization service. This is the scenario of Figure 2. When the portal needs to know who the user is, it redirects the user to the authorization service. The authorization service handles the SAML login, and asks for user consent to use the required services. The portal uses the returned access token to obtain user information, as well as to invoke services.

In this approach there is a single, centrally managed authorization service.

This is the approach that Google[13] has taken.

Solution B: Addition of an authorization endpoint

Both problems can be overcome by the development of an authorization endpoint. This provides authorization by means of a SAML identity provider, and returns an authorization code (or assertion) to the client. This is then supplied to an AS to obtain an access code.

In this way the authorization service can trust just the authorization endpoint, instead of all clients. This method can also work with general OAuth 2 client libraries out of the box.

This solution involves a centrally managed authorization endpoint, and one or more authorization services. This would allow owners of a group of services to use their own authorization services, for example. For the use-case at hand, however, both services can be run centrally on a single server.

Facebook[14] and Github[15] use this approach.

---

12 Explained at http://msdn.microsoft.com/en-us/library/windowsazure/gg185950.aspx
13 http://code.google.com/apis/accounts/docs/OAuth2.html#webserver
14 http://developers.facebook.com/docs/authentication/
15 http://developer.github.com/v3/oauth/

Solution C: SAML assertion pass-through

The security issue could also be solved by passing signed SAML assertions to the AS.

When the user logs into the client, it can obtain a signed SAML assertion from the identity provider. When this is passed to the AS, it can check that the assertion was signed by a trusted identity provider.

The OAuth 2 draft related to this[16] (SAML bearer assertion) specifically mentions that the assertion MUST contain an audience restriction identifying either the AS(es) or the service(s). This would mean that adding a service or AS would need an update of the information at all identity providers, making it harder to add a service.

This solution still requires SAML support at each client. OAuth2lib uses SimpleSAMLphp, which does not provide access to the raw assertion[17]. Also each AS needs to maintain a list of identity providers.

Other software

Other OAuth 2 implementations have been looked at as well, but most software packages are libraries, not services that one can install and go[18]. This would require some development work.

OAuth2 conclusion

OAuth2lib's implementation currently has some problems. Some development work is needed to come to a clean solution. This can be either moving authentication to the authorization service (solution A), or adding an authorization endpoint (solution B). The latter allows for more flexibility. I would estimate the development work required in the order of weeks (for either solution).

If the problems mentioned are not so much an issue for the basic use-case at hand, it is also possible to use OAuth2lib as-is. For future deployments, this is certainly not recommended.

X.509 certificates

Since the OAuth 2 approach was not completely successful yet, we plan to setup a test environment for X.509 early 2012.

This will be based on software from the CILogon[19] project, as well as the GridSite delegation service[20]. In this approach certificates are only used on portals and services, completely hidden from the user. More details can be found in the full report (see footnote 6 on page 1).

Conclusion

The current OAuth 2 setup is not conforming to the security requirements. This can be solved by adaptation of OAuth2lib.

An X.509 certificate based test setup is planned.

16 draft-ietf-oauth-saml2-bearer (version 09)
17 The assertion is not stored; http://groups.google.com/group/simplesamlphp/msg/45046408d1667ebd
18 Except maybe for commercial solutions like PingFederate. These are outside the scope of this project.
19 http://www.cilogon.org/
20 http://www.gridsite.org/