Detlef Hühnlein et al. (Eds.): Open Identity Summit 2015,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015 43

Proxied Authentication in Single Sign-On Setups with Common Open Source Systems – an Empirical Survey

René Peinl[1] Florian Holzschuher[1]

Abstract: The paper presents results from an empirical study about the use of a single sign-on (SSO) system in an integrated open source system landscape for supporting team collaboration. A portal solution, enterprise content management system, groupware, business process management and enterprise search engine are used. The investigation shows that although it is easy to achieve SSO with the Web-based user interfaces of the information systems used, none of the systems was prepared to pass authentication tokens to the API of an integrated system or accept SSO tokens instead of username / password pairs for authentication against the API respectively. Different alternatives for achieving the desired functionality are presented and a recommendation for improvement of the affected information systems is derived.

Keywords: single sign-on, double-hop problem, proxied authentication, open source systems

1 Introduction

A modern digital workspace often consists of a number of specialized software systems, capable of solving different problems. In order to minimize overhead, these systems can use a single sign-on (SSO) authentication system, eliminating the need for separately logging into each system. While big vendors' ecosystems, such as Microsoft's, ship with full SSO support throughout, independent open source software's support for SSO is often limited and does not cover APIs. A number of systems supports using an external SSO system for accessing the Web user interface (UI), but still requires passing username and password to integrated systems in the back-end. In a non-SSO setup for example, when a Liferay Portal connects to an external document management system (DMS), the user's credentials entered while logging into Liferay are replayed to authenticate with the DMS. However, in an SSO setup, the user authenticates with Liferay via an SSO token which is only valid for Liferay and cannot be replayed. Furthermore, username and password of the user are not known by Liferay, but only by the SSO system. Therefore, those cannot be used either. A proxied SSO authentication in the user's name would be required.

[1] Hof University, Institute of Information Systems, Alfons-Goppel-Platz 1, 95028 Hof, Germany,
{rene.peinl | florian.holzschuher}@iisys.de

2 Usage scenario

Our target setup for the Social Collaboration Hub (SCHub[2]) project contains, among others, Central Authentication Service (CAS[3]) as a SSO server, Liferay[4] as a portal, Nuxeo[5] as an enterprise content management system (ECMS) and Camunda business process management (BPM)[6] as a workflow engine. Furthermore we use Open-Xchange[7] (OX) as a groupware, with Postfix[8] and Dovecot[9] as its backend. All services are connected to an LDAP server for user account information. Fig. 1 gives a graphical overview of the setup and the communication relations, especially those between server systems. CAS and LDAP have connections to every other system except Dovecot. These are omitted in the figure in order to make it clearer.

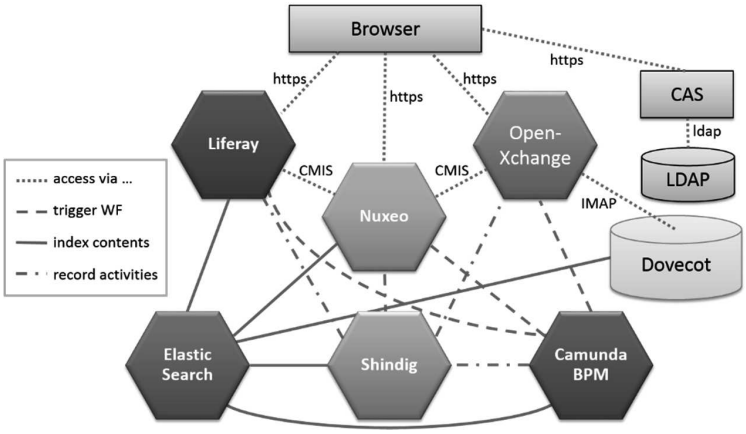

Fig. 1. Communication between systems. CAS relations are omitted (own illustration)

We have several use cases where the problem described above occurs.

1. Access to the ECMS Nuxeo via CMIS[10] from Liferay and OX
2. Triggering workflows in Camunda from Liferay, Nuxeo and OX
3. Storing activities in Shindig from Liferay, Nuxeo, OX and Camunda
4. Accessing emails in Dovecot via IMAP[11] from OX

[2] https://www.sc-hub.de/
[3] http://jasig.github.io/cas/
[4] http://www.liferay.com/
[5] http://www.nuxeo.com/
[6] http://camunda.org/
[7] http://www.open-xchange.com/en/home
[8] http://www.postfix.org/
[9] http://www.dovecot.org/
[10] Content Management Interoperability Services
[11] Internet Mail Access Protocol

In principle, there are also problems when accessing Elasticsearch from the search UI in Liferay, but these can be circumvented by directly accessing the Elasticsearch REST API from the portlet in the browser. Another challenge is indexing the contents of systems that require SSO authentication, but this is not discussed here. In the remainder of the paper the challenges of proxied SSO authentication are investigated, wide-spread authentication protocols are analyzed regarding their support for this scenario and different approaches to securely pass on authentication information are evaluated. Finally the above listed use cases are discussed and a suggestion for enhancements of common open source projects is derived in the conclusion.

3 Challenges

The requirements in the case described are the same as with multi-tiered applications [Hi00]. We need the possibility for a front-end system to access a back-end service under the authenticated user's identity [Au04]. The problem already starts with a commonly accepted term describing it. Microsoft dubbed it "the double hop issue" [Py08], in SAML the feature addressing it is known as delegated authentication [RD10], other authors call it impersonation [FF12] or proxy authentication [Sp11]. However, proxy authentication is an ambiguous term since it usually denotes a proxy that authenticates the user before passing his requests to the application, whereas here it refers to an application passing authentication information to a second application via its API. Delegated authentication is also ambiguous as it also denotes delegating the task of authentication to an external system like CAS. In this paper, the term "proxied authentication" is therefore used to denote the case where a user authenticates with an external single sign-on system for a Web-based application server system. This system in turn passes authentication information to a second back-end server system in order to access its application programming interface (API) with the name and permissions of the logged in user. Usually, common open source systems (OSS) as the ones used in SCHub rely on direct logins into their system. For accessing services of another system, the username and password accepted from the logged in user are often replayed to the second system. However, every application managing its own password replay feature is not only a security flaw but also thwarts the purpose of an SSO setup. Additionally, this replay functionality can be required by libraries used by the system, which open connections to or receive connections from other systems and do not support SSO-compatible authentication methods as is the case with the version of Apache Chemistry library used for CMIS support in Nuxeo. This makes it even more complicated to modify an integrated system so that all components use SSO. Especially the micro-service architectural style [LF14] frequently suggested in the last years [NS14] for Software as a Service (SaaS) scenarios is suffering from this problem. Although the problem is solved from a scientific point of view, the challenge of getting it to work in practice is demanding since it not only requires the usage of an authentication protocol that supports the feature. Both the SSO solution and the server systems must implement that protocol including this specific feature of the protocol as well as be prepared to pass on SSO tokens to external systems and use them for authenticating incoming API requests respectively. Fig. 2 is visualizing the

complex interplay. The user is accessing server system 1 using the browser (1) in order to view information stored in server system 2. System 1 detects that the user is not logged in and redirects the browser (2) to the external SSO system (3). After successful login there (not shown in the figure) the SSO system delivers a user ticket back to the browser (4) that in turn uses it to access system 1 as originally intended (5). However, system 1 needs a proxy ticket instead of the user ticket in order to pass credentials to system 2. Therefore, it has to request it from the SSO system (6) using the user ticket and a process described in section 4.2 but not shown in the figure. Once it got the proxy ticket (7) it can pass it on to system 2 (8) in order to retrieve the data from there (9) or invoke an action and present the result to the user (10). System 2 should be able to validate the ticket without a need to access the SSO system. Ideally, the ticket includes information about the user ID. Otherwise, system 2 would still need to query the SSO system for retrieving user information, which slows down the whole process.

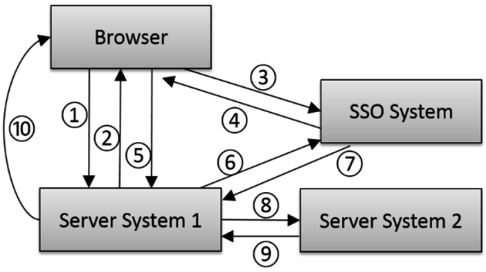

Fig. 2. Communication flow during proxied authentication

The described solution is the cleanest way of achieving the desired result and implemented in SAML 2.0 (section 4.1), Kerberos (section 4.3), as well as CAS proxy tickets (see section 5.2). Another solution, but more a workaround that circumvents the problem, is the usage of system users with fixed credentials. Despite this being relatively easy to implement and partial support for this solution in existing systems, there are several drawbacks. For one, these credentials need to be stored in the configuration of several services and only a part of them may support password encryption (as is the case in SCHub). Moreover, for requests to be handled properly and securely they need to be executed as the right user. But executing requests in a user's name using admin credentials is not widely supported and thus authorization constraints are not in effect and actions may be attributed to an admin, not the user triggering a request. One example for this mechanism is indexing of content stored in Dovecot using the IMAP river plugin of Elasticsearch. We have extended the original plugin[12] provided by Hendrik Saly in order to support the feature of Dovecot to access a user mailbox with an admin account. Such a mechanism is often called impersonation [Ve06]. Normal authentication tokens provided by users cannot be stored and passed on to another service, primarily because they are only valid for the service they were issued to. Furthermore,

[12] https://github.com/salyh/elasticsearch-river-imap

acquiring them through automated logins in the backend without a proper browser session has proven to be very difficult in our tests. The CAS documentation hints[13] at this being on purpose, since passwords have to be known outside of CAS. Otherwise, this mechanism could be easily used for brute force attacks on passwords.

4 Authentication Protocols

4.1 OAuth 2.0

OAuth 2.0 (http://oauth.net/2/) is an authorization protocol/framework, not only supporting Web but also desktop applications. It will be called OAuth in the rest of the paper. The normal authentication flow in OAuth for Web applications is called authorization code grant and works as follows. The server system, the user is trying to access (client in OAuth terms) redirects the user's browser (user agent) to the authorization server. The user is authenticated there presenting an authorization grant (e.g., username / password) and in case of valid credentials gets an authorization code. This code is delivered back to the client which can request the access token and use it to access the resources of the resource server (server system 2), which should be able to verify the access token without needing to contact the authorization server [Ha12]. Optionally, the authorization server can deliver a refresh token together with the access token, which the client can use afterwards to get new access tokens without the need for re-authentication.

In the case of Internet systems, the OAuth authorization server and the resource server are usually the same (e.g. Facebook, Google+). The client is usually a third party Web application that both uses the authorization server as an SSO system as well as the resource server to access additional information about the user or post messages in the name of the user [cf. SB12]. In the use case presented here, however, the authorization server is the SSO system whereas the resource server would be server system 2 and the client server system 1 (see figure 2). Throughout the whole OAuth specification, only the supplement on "bearer token usage" refers to our use case by stating that the bearer token scheme "is intended primarily for server authentication using WWW-Authenticate and Authorization HTTP headers but does not preclude its use for proxy authentication" [JH12]. It seems that OAuth is still missing the clarification of the SAML addendum (see below). Another issue is that the structure of tokens in OAuth is not prescribed and therefore it cannot be guaranteed that two systems will really be interoperable, although both are implementing the specification. Liferay includes an OAuth provider, but it seems to be version 1 only and additionally is working in the wrong direction for SCHub, since it is used to grant other applications access to Liferay [Li00]. There is a plugin for Liferay called oxAuth[14] which allows users to authenticate against an external OAuth provider in order to access Liferay. However, it seems to require OpenID connect support by the authorization server,

[13] https://wiki.jasig.org/display/CAS/Using+CAS+without+the+CAS+login+screen
[14] http://liferay.pbworks.com/w/page/83464783/oxAuth%20plugin%20for%20LifeRay

which is only supported by CAS in the client role up to now[15]. CAS currently supports OAuth 2.0 with the authorization code grant[16]. Consequently, our tests did not succeed to authenticate a user for Liferay using CAS and OAuth. Nuxeo also directly supports OAuth 2.0 and esp. the provider role, which can be used to access Nuxeo from third party applications. However, the CMIS server used in Nuxeo (Apache Chemistry) does not. Astonishingly, we managed to access the Nuxeo CMIS interface using OAuth authentication from a test client. It seems that Nuxeo has linked its own pluggable authentication to the Chemistry library. However, we did not manage to connect the OAuth functionality of Nuxeo with the CAS server and therefore, the use case is not supported.

4.2 SAML 2.0

Security Assertion Markup Language (SAML) 2.0[17] is an XML-based authentication and authorization standard. It was developed to accompany SOAP-based Web services, although it is versatile and can be used to transport arbitrary tokens, e.g. from Kerberos. It supports several profiles[18] for different use cases, but a widely used SSO profile is the Web browser SSO profile introduced in SAML 2.0 [AC08]. The same applies to transport bindings which include SOAP over HTTP as well as HTTP redirect and POST [MR08]. The "Delegated SAML Authentication" described in [MM12] could be used to achieve proxied authentication. It uses SOAP messages, WS-Addressing, public key encryption and signatures to exchange authentication information and user identity assertions. The original SAML specification was not precise enough in this special case, so that an addendum was released later on [Oa09]. CAS fully supports SAML 1.1 and support SAML 2.0 to some extent, especially those features needed for cooperation with Google apps[19]. Liferay is more elaborate in this case and provides a marketplace plugin[20] for the enterprise version that allows Liferay to act both as an Identity Provider (IdP) or Service Provider (SP). Nuxeo provides some SAML support via integration of CAS or Shibboleth and is actively working on providing native support in Nuxeo 7.4 scheduled for September 2015[21]. However, even if all three supported SAML 2.0, Liferay would still need to implement the delegated SAML authentication like uPortal does[22] in order to support our use case.

4.3 Kerberos

Kerberos[23] is an authentication protocol that can be used for SSO. It was not tailored for Web applications, but rather operates on an operating system level, retrieving tickets for

15 https://github.com/Jasig/cas/pull/910
16 http://jasig.github.io/cas/4.0.x/protocol/OAuth-Protocol.html
17 http://saml.xml.org/saml-specifications
18 http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf
19 http://jasig.github.io/cas/4.0.x/protocol/SAML-Protocol.html
20 http://www.liferay.com/de/marketplace/-/mp/application/15188711
21 https://jira.nuxeo.com/browse/NXP-14595
22 https://wiki.jasig.org/display/UPM31/00+-+Delegated+SAML+Authentication
23 http://web.mit.edu/kerberos/

supported applications [NT94]. This is not surprising given its year of invention even before the advent of the www [St88]. However, it is versatile enough to be used in modern usage scenarios as well. If the user logs on to her PC using Kerberos and browsers are forwarding the respective ticket, it is possible to achieve SSO on an even deeper level than with Web-based protocols. Internet Explorer does this by default for the local intranet (Windows domain). In Firefox and Chrome it is available, but only after configuring the respective server names in a whitelist. Kerberos is widely used in Windows-based intranets [Py08]. On Linux, it seems less popular, although there are two open source implementations available, which both support Kerberos v5: MIT Kerberos server and Heimdal [Ga03]. Kerberos uses a rather complex, but effective mechanism to solve the problem, similar to the solution described in Fig. 2. Clients get a session ticket, once the user has authenticated successfully against the authentication server. With this, they can obtain a ticket granting ticket (TGT), which in turn allows to get service tickets for the respective services that should be invoked [Ga03, S.21]. It therefore supports proxied authentication. CAS supports Kerberos for clients running in a Windows Active Directory (AD) domain[24]. The CAS mechanism uses the MIT Kerberos stack and is called SPNEGO. Liferay does not support Kerberos by default, but can be configured to use it with an Apache module [Pa14]. Nuxeo even supports it out of the box using a free plugin and the MIT Kerberos server[25]. However, since there is no Windows AD domain in the project setup, this option is not feasible in our use case.

5 Proprietary approaches

CAS supports all of the protocols mentioned above, but uses an own proprietary protocol by default[26]. It uses CAS clients to protect the "casified" applications and retrieve the identity of the authenticated user from the CAS server. The current CAS server v4 uses CAS protocol v3. Key concepts are service tickets (ST) transmitted as parameter in the URL that grant access to a service and ticket granting tickets (TGT) stored in a cookie as a representation of the user session and a means to request new tickets. Thus, it is similar to Kerberos, but uses XML over HTTPS for transport.

5.1 Proxy Authentication

CAS Proxy Authentication[27] is a mechanism through which an application, with a valid TGT, can request a proxy granting ticket (PGT) which it can then use to retrieve proxy tickets for other services [Sp11]. This proxy ticket includes a validation chain that the service must validate in order to prevent impersonation attacks. This mechanism has many advantages, beginning with the fact that no password has to be replayed, making this approach much more secure. Furthermore, the actual user who made the original request is

[24] http://jasig.github.io/cas/4.0.x/installation/SPNEGO-Authentication.html
[25] https://doc.nuxeo.com/display/ADMINDOC/Using+Kerberos
[26] http://jasig.github.io/cas/4.0.x/protocol/CAS-Protocol.html
[27] http://jasig.github.io/cas/development/installation/Configuring-Proxy-Authentication.html

authenticated using a request that transparently shows, by which application the authentication was proxied (authentication chain). This proxy authentication can be configured per service as well. The downside to this approach is the high implementation effort required. Backend systems need to be able to validate proxy tickets (in addition to user tickets) including the proxy chain[28] and proxying systems need retrieval logic for proxy tickets and an additional REST endpoint to receive these additional tickets and attach them to the right request. The callback is used for validating the service using its URL and the certificate for the HTTPS connection.

5.2 CAS ClearPass

CAS ClearPass[29] is the password replay feature of CAS, which can provide services with the user's password captured during login to CAS. This setup is only slightly more secure than leaving password capture to individual services since password storage is centralized and encrypted. It reuses the first part of the proxy ticket mechanism, so that the proxy application needs to retrieve a PGT and a proxy ticket first in order to get the clear text password. Afterwards, this password can be sent to the protected service. Using ClearPass eliminates the need to modify backend applications as long as they authenticate against the same LDAP directory or use the same username/ password combination. Still, the frontend application needs to request the password from the CAS server, extract it and attach it to the user's relayed request. Storing the user's password as an encrypted session variable can help to prevent the SSO system from becoming a performance bottleneck. The use of this feature can be authorized for individual services in order to minimize the potential for abuse.

6 Discussion

6.1 CMIS

Apache Chemistry is the reference implementation for CMIS and provides both a server and a client implementation in Java. Since all systems used in SCHub are mainly developed in Java, it is not surprising that existing implementations in Nuxeo and Liferay both use Chemistry and the developers of Open-Xchange have also decided to use Chemistry as a basis for their own CMIS interface, which is currently under development as part of the SCHub project. Although both Nuxeo and Liferay natively support CAS as well as several other authentication protocols, Chemistry had no working authentication mechanism besides username/password-based authentication at the time of writing. Liferay consequently explicitly states it on the Website[30], that SSO is not supported for connecting to a remote CMIS repository. A review of included Chemistry versions in historic Nuxeo and Liferay versions

[28] http://jasig.github.io/cas/4.0.x/planning/Security-Guide.html
[29] http://jasig.github.io/cas/4.0.x/integration/ClearPass.html
[30] http://www.liferay.com/de/community/wiki/-/wiki/Main/CMIS+Repository

made clear that both systems include fairly recent builds of Chemistry. Given that information, it was clear that the right way to go is contributing to Apache Chemistry instead of making project-specific developments. Fortunately, our discussions with representatives of the Deutsche Wolke working group within the Open Source Business Alliance led us to Grau Data GmbH who are already an active contributor to Chemistry and agreed to implement OAuth 2.0 functionality for Chemistry in order to make it available for their own CMIS-based products. However, we are exploring the use of CAS ClearPass in parallel, since this seems to require the least effort.

6.2 Workflows

The Camunda BPM solution is primarily designed for being embedded into its customers' information systems, although it can be used as a standalone workflow engine as well. Therefore, it doesn't pay special attention to authentication and only provides basic HTTP authentication which is even switched off by default[31]. However, the authentication provider is exchangeable. The cleanest way would be to implement a CAS proxy ticket validation chain as a pluggable provider there. However, the effort to implement the proxy retrieval mechanism in Liferay, Nuxeo and OX is high. Starting from version 7.6.1, OX uses OAuth 2.0 to access subscribed Google calendars[32]. This mechanism could be extended to access Camunda as well. As Camunda is already prepared to work with REST frameworks such as RESTeasy or RESTlet, this could be used to implement OAuth 2.0 for Camunda[33]. Another option would be to embed Camunda into Liferay and use Liferay's authentication features as a wrapper around Camunda similar to Nuxeo embedding Chemistry and combining it with its own authentication mechanism. However, this would counter the project's intention of achieving a micro-service approach, which requires separation of concerns on the level of the deployment unit. Although OX supports SAML 2.0 starting from version 7.8.0[34], it doesn't seem like the right way to go in this case. Kerberos isn't an option here as well.

6.3 OpenSocial

Shindig is the OpenSocial reference implementation and included in both Liferay and Nuxeo as a gadget container and rendering server. Starting with OpenSocial 2.0, OAuth 2.0 was specified as the primary authentication mechanism [Hinc11]. In principle, CAS supports OAuth 2.0 and authentication protocols can be specified per connected system. Shindig currently supports OpenSocial 2.5 and has an OAuth 2.0 service provider implementation[35]. However, we felt it would be a good idea to make our setup more uniform and "casified" Shindig, so that it can be used with the same protocol that was used with

[31] http://docs.camunda.org/latest/api-references/rest/#overview-configuring-authentication
[32] https://oxpedia.org/wiki/index.php?title=Google
[33] http://docs.jboss.org/resteasy/docs/3.0.9.Final/userguide/html/oauth2.html
[34] http://oxpedia.org/wiki/index.php?title=AppSuite:SAML_SSO_Integration
[35] http://bit.ly/1NeU8uE

the other information systems. It therefore seems to be a similar case as for CMIS and workflows. However, the problem is a bit different here. For storing activities in Shindig, it wouldn't be required for the proxy system to impersonate the user. It would be enough to authenticate the server system in order to prevent malicious entries being sent to Shindig. It even doesn't make sense for certain cases to use the logged in user here, since it cannot be guaranteed that there is one in every case (e.g. workflow completed events triggered by a timer event). Therefore, we propose using 2-legged OAuth 2.0 authentication for storing activities in Shindig. For showing the activities, the problem doesn't occur at all, since this is done in the browser using JavaScript and therefore, the service ticket can be used directly without the need to retrieve a proxy ticket.

6.4 IMAP

Although OX is used in many large installations (e.g. at Strato) and it is a common requirement to provide SSO to all services offered by an internet service provider, it seems that none of the existing OX customers demanded it yet. Otherwise, it would have become obvious that OX is not able to access the IMAP server in an SSO scenario, because none of the available IMAP servers is supporting common SSO protocols made for the Web. Dovecot already was the preferred IMAP server for OX before and is now even more after OX has acquired the company behind Dovecot[36]. Out of the above listed protocols, Dovecot supports only Kerberos[37], but OX doesn't. OX provides OAuth 2.0 and recently SAML 2.0 support as stated above. According to Carsten Dirks from OX during our project meeting, customers are trusting on a secure channel between OX server and IMAP server and are using the above mentioned feature to access the mailbox with a root account, but in the name of a normal user.

7 Conclusion

While there are some options for getting a working proxied authentication setup using open source software, compromises sacrificing some security may have to be made for some systems. With sufficient time and effort, at least the Java-based systems and libraries we are using could be modified to fully support CAS proxy tickets. But such in-depth modifications will probably be out of scope for most real world projects. OAuth 2.0 would be a good candidate for a solution, but only recently OpenID connect filled the gap of specifying the exchanged tokens (JSON Web Tokens) and therefore ensuring interoperability in our scenario[38]. CAS is going to support OpenID connect server role in the upcoming version 4.1. Hopefully, Liferay with oxAuth will then work with CAS as well as Nuxeo. If so, it OpenID connect would be our preferred authentication solution. In the

[36] http://www.open-xchange.com/en/dovecot
[37] http://wiki2.dovecot.org/Authentication/Kerberos
[38] http://openid.net/specs/openid-connect-core-1_0.html#IDToken

meantime, we are going to use CAS ClearPass as a workaround. The problem of authentication for scheduled tasks on a user's behalf can be solved with password replay, although this remains an undesirable solution. Thus, the problems of proxied authentication are solveable, but without support from the developers of the individual systems and libraries, the effort required to implement them is immense. With CAS being only one of many open source SSO solutions (e.g., Forgerock OpenAM, Shibboleth, FreeIPA), widespread support for all of its features is unlikely to be implemented in the near future. Although many popular open source systems support one or even several wide-spread authentication protocols, this support often doesn't cover every detail of the specification. Mostly, only one or two profiles or flows are implemented, which is enough for a specific user requirement, but not for general interoperability. We strongly recommend outsourcing the authentication task to third party systems like the application server or at least using some common libraries like RESTeasy or PAC4J that include support for common authentication protocols. Otherwise, the burden to support flexible authentication is too high for a small OSS. The use of JAAS[39] in Nuxeo shows, that this approach is the right way to go and leads to versatile systems. However, including third party components like Apache Chemistry or Apache Shindig requires additional attention in order to prevent limitations regarding overall availability of all authentication options. This wouldn't be the case, if these third party components supported JAAS as well. The recent trend in cloud computing towards "everything as a service" [Scha09] led to "Identity and Access Management as a Service" (IDaaS) [NuAg14] with players such as Octa and PingIdentity [KrWy15] which seem more suitable for use cases such as the one presented here than public Internet providers like Google and Facebook as identity providers. There is also an award winning project in Germany called SkIDentity [KuÖF14], which we are looking to integrate for allowing secure access using the new German identity card or a health insurance card. However, this won't solve problem presented here either.

References

[Ar08] Armando, A.; Carbone, R.; Compagna, L.; Cuellar, J.; Tobarra, L.: Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for google apps. In: *Proc. of the 6th ACM workshop on formal methods in security engineering* : ACM, 2008, S. 1–10

[Au04] Aubry, P.; Mathieu, V.; Marchal, J.: ESUP-Portail: Open Source Single Sign-On with CAS (Central Authentication Service). In: . Ljubljana, Slovenia, 2004, S. 172–178

[FF12] Fan, Z.; Fan, Z.: Security Research about Asp.net Web Application. In: *Proc. of the 2012 Intl. Conf. on Computer Application and System Modeling* : Atlantis Press, 2012

[Ga03] Garman, J.: *Kerberos: The Definitive Guide: The Definitive Guide* : O'Reilly, 2003

[Ha12] Hardt, D.: The OAuth 2.0 authorization framework. http://bit.ly/1L6BYKz

[39] Java Authentication and Authorization Service

[Hi00] Hill, P. B.: Kerberos interoperability issues. In: *Proc. of the 3rd Large Installation System Administration of Windows NT 2000 Conference*, 2000, S. 35–42

[Hi11] Hinchcliffe, D.: *OpenSocial 2.0: Will key new additions make it a prime time player in social apps?* http://zd.net/1LQ6o1f

[JH12] Jones, M. ; Hardt, D.: *The OAuth 2.0 authorization framework: Bearer token usage*, http://bit.ly/1NeUzVK

[KW15] Kreizman, G.; Wynne, N.: *Magic Quadrant for Identity and Access Management as a Service, Worldwide* (Analyst Report). Stamford, CT, USA : Gartner, 2015

[Ku14] Kubach, M.; Özmü, E.; Flach, G.: Secure cloud computing with SkIDentity: A cloud-teamroom for the automotive industry. In: Open Identity Summit 2014. Stuttgart, 2014

[LF14] Lewis, J.; Fowler, M.: *Microservices*. http://bit.ly/1dI7ZJQ

[Li15] *OAuth - User Guide - Liferay.com*. http://bit.ly/1RoZWqN

[MM12] Masi, M.; Maurer, R.: On the usage of SAML delegate assertions in an healthcare scenario with federated communities. In: *Electronic Healthcare* : Springer, 2012, S. 212–220

[MR08] Maler, Eve ; Reed, Drummond: The venn of identity: Options and issues in federated identity management. In: *IEEE Security & Privacy* (2008), Nr. 2, S. 16–23

[NS14] Namiot, Dmitry ; Sneps-Sneppe, Manfred: On Micro-services Architecture. In: *International Journal of Open Information Technologies* Bd. 2 (2014), Nr. 9, S. 24–27

[NT94] Neuman, B.C.; Ts' O, T.: Kerberos: An authentication service for computer networks. In: *Communications Magazine, IEEE* Bd. 32 (1994), Nr. 9, S. 33–38

[NA14] Nuñez, D.; Agudo, I.: BlindIdM: A privacy-preserving approach for identity management as a service. In: *Intl. J. of Information Security* Bd. 13 (2014), Nr. 2, S. 199–215

[Oa09] OASIS: SAML V2.0 Condition for Delegation. http://bit.ly/1X7BnOC

[Pa14] Patou, M.: *Kerberos SSO with Liferay 6.1*. http://bit.ly/1X7BvO5

[Py08] Pyle, N.: *Understanding Kerberos Double Hop*. http://bit.ly/1LnbP8z

[RD10] Rybicki, A.; Dalquist, E.: uPortal 3.1 Manual - Chapter 02-05-06-03-00 - Delegated SAML Authentication, http://bit.ly/1L6wsdK

[Sc09] Schaffer, H. E.: X as a service, cloud computing, and the need for good judgment. In: *IT professional* Bd. 11 (2009), Nr. 5, S. 4–5

[Sp11] Spencer, David: *Proxy CAS Walkthrough*. http://bit.ly/1LwiVLo

[St88] Steiner, J. G.; Neuman, B. C.; Schiller, J. I.: Kerberos: An Authentication Service for Open Network Systems. In: *USENIX Winter*, 1988, S. 191–202

[SB12] Sun, S.-T.; Beznosov, K.: The devil is in the (implementation) details: an empirical analysis of OAuth SSO systems. In: *Proc. of the 2012 ACM Conf. on Computer and Communications Security* : ACM, 2012, S. 378–390

[Ve06] Veeramani, N.: Smart clients versus web forms. In: *Computer* Bd. 39 (2006), Nr. 8, S. 93–95