Extending OAuth2 to Join Local Services into a Federative SOA

M. Politze[1]

[1]IT Center RWTH Aachen University, Seffenter Weg 23, 52074 Aachen, politze@itc.rwth-aachen.de

Keywords
BYOD, SOA, OAuth2, federated services, eScience, eLearning

1. ABSTRACT

New approaches to teaching and research require machine readable interfaces as provided by service oriented architectures. Many scenarios in teaching as well as in research demand cooperation between members of universities. Identity federations and federated applications provide some means for collaboration. Current federation services for single sign on however are not compatible with common applications arising from service oriented architectures. The OAuth2 workflow allows personalized access to service oriented architectures for applications but lacks a definition for federated deployment. The presented extensions are based on the current implementation of OAuth2 They allow establishing a federation of OAuth2 token services that builds on top of the existing federated infrastructures in higher education.

2. INTRODUCTION

For funding and research organizations as well as governments, cooperation between universities and research organizations became a mandatory requirement. As projects are involving more cooperation, especially over long distances, supporting IT services are becoming more important. After all, many IT services that were exceptional five or ten years ago have emerged to be a part of the daily life for universities' members: students, researchers and employees alike. Consequently, universities want to rise their attractiveness by offering more innovative IT services.

Closer collaboration between researchers of different origins means increased mobility and in turn requires IT services to be transitive between organizations. Consequently, also funding for IT infrastructures and services converges to competence centers offering elaborated services in regional, national or even international federations (RfII – German Council for Scientific Information Infrastructures, 2016). Local providers have often swiftly tailored offered services to existing processes. Federative services, however, are often more basic and therefore may require levels of adoption to replace local infrastructures.

Ubiquitous access, standardized, machine readable and programmable interfaces (APIs) are becoming even more important due to the rising number of smart devices and data intensive applications: Not only in scientific context but also as parts of every users' daily routines. The Horizon report, one of the most regarded studies concerning the development of education lists Mobile Learning, Internet of Things and Artificial Intelligence among the six most important developments in higher education in the coming years (Adams Becker, et al., 2017).

Federated IT services, especially web applications, are available but most do not offer, APIs for the user to integrate services into local processes. Clearly one of the reasons is that, apart from some local implementations, the current federative authentication and authorization infrastructure (AAI) often does not offer out-of-the-box solutions for securing APIs and identifying users. Using the existing infrastructure, a set of extensions to the OAuth2 workflow aims to lift authorizations to the federated level.

3. RELATED WORK

Identity federations are well established and widely used to secure internet applications in educational and scientific context. These federations offer single sign on capabilities and allow exchanging user information between participating services (Grabatin, Hommel, Metzger, & Pöhn, 2016). National federations organize hierarchically to inter federations such as eduGAIN on a worldwide level. This does not only provide a standard way to authenticate users but also allows easy sharing and reuse of services across universities and other organizations. The underlying Security Assertion Markup Language (SAML) protocol (Cantor, Kemp, Philpott, & Maler, 2005) combined with its most widely deployed implementation Shibboleth (Knight, et al., 2014) however is technically restricted to interactive sessions in a web browser. Other applications, like apps installed on a smartphone require additional protocols.

Other software suites like OpenID Connect are more widespread in commercial applications. Major IT companies like Google, Microsoft and PayPal offer complying endpoints. OpenID Connect uses the OAuth2 workflow as a basis to exchange user information and to provide sign on capabilities for remote services (Sakimura, Bradley, Jones, Medeiros, & Mortimore, 2014) and thus allows usage beyond web applications. The underlying workflow is easier to understand than SAML, which leads to a decent fragmentation of client and server implementations supporting OpenID. While the specifications imply a certain security, recent analysis of applications revealed that many applications are prone to security problems due to implementation errors (Li & Mitchell, 2016). In the current specification, OpenID connect allows remote services to log in via a single identity provider or the service has to maintain a set of viable identity providers. There exists, however, a draft on how federations of several identity providers are realized using OpenID Connect (Hedberg, Gulliksson, Jones, & Bradley, 2016). While OpenID Connect uses the OAuth2 workflow, its intention is to authenticate users and not authorization for applications and web resources needed in the underlying scenario.

A prominent example of a federated service is the campus cloud “sciebo” (Vogl, et al., 2015). While the service itself is working properly, it shows where the current federation using SAML and Shibboleth fall short: upon first usage, users need to create a new account and set a separate password after authorizing via Shibboleth. The user then has to log on to the service using the newly created credentials. Nevertheless, this allows the user to use the file synchronization APIs of the software using the same credentials. Applications using the APIs for a user need to store these credentials. While users mostly accept this for locally installed applications, this is legitimately uncommon for third party applications that want to integrate sciebo into existing processes. In addition, lifecycle management of eligible user accounts is an issue. The system therefore requires a login via Shibboleth every six months to identify old user accounts.

Several single universities approach interfaces that integrate existing systems into new applications for their students and employees. The works of Mincer-Daszkiewicz and Barata et al. show two practical examples of APIs in the field of university administration (Mincer-Daszkiewicz, 2014) (Barata, Silva, Martinho, Cruz, & Guerra e Silva, 2014). The architecture presented by Politze et al. generalizes for several personalized eLearning applications but aims to cover more university processes (Politze, Schaffert, & Decker, 2016).

Together with the project STApps Lehmann et al. have developed Viadrina Core, a middleware that abstracts data from student lifecycle management into a generic interface that can be used across university boundaries, for example in a federation, but has currently no means to identify users (Lehmann & Huber, 2015).

The goal to offer web services for eLearning and eScience basing on existing AAI federations is the starting point for this extension of the OAuth2 workflow.

4. THE CURRENT OAUTH2 IMPLEMENTATION

The OAuth2 workflow as described in RFC 6749 (Hardt, 2012) allows secured, personalized access to web services or resources and handles the user’s authorization without supplying credentials to the application itself. This also paves the way for third party developers accessing central IT services. Generally, it follows the steps 1-4 shown in Figure 1.

At first, the user accesses the application (1). To access the web service resources for the user the application needs an access token. It directs the user to the token service passing along a set of scopes

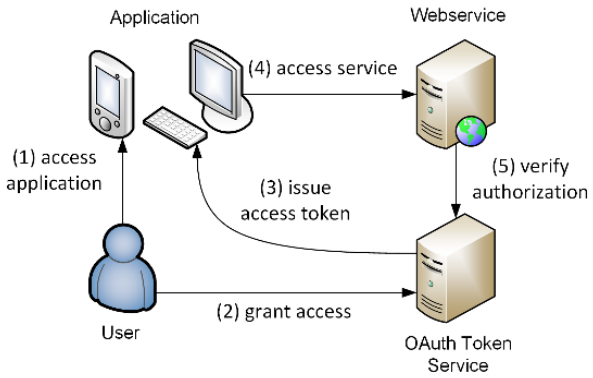

Figure 1: Schematic of the OAuth2 workflow

that defines which services to access. The token service then requires the user's credentials before granting access for the application (2). The token service then issues the access token (3). The application can now use this token to access the web service resources (4). Upon incoming requests, the web service has to verify the authorization (5). Due to tight coupling of web services and the token service, this step remains mostly internal. To decrease coupling of the services, modelling of this step is key for a cooperative scenario where multiple web services access a single token service.

Detaching authorizations and token handling from the business logic allows decentralization of services within a cooperation. All web service resources, however, are able to process the same tokens. The user has to authorize an application only once and not for every service in the cooperation which is a requirement for seamless integration of APIs into one application.

In terms of operability and security key implications of this model are:

- *The token service is the authority*
  Within the OAuth2 workflow, the token service clearly identifies as an endpoint for all redirections of the user and backchannel communication of web services.
- *The token service is trusted*
  Both the users and web services know and trust the token service. Users trust in the service to authorize applications only with their consent. Web services trust that the authorization information provided by the users is legitimate.
- *Users are known*
  Verifying the authorization only transfers minimal information about user context of the application. Web services have to determine the user's permissions to access certain resources.
- *Applications and web services are separated*
  Applications rely on resources provided by web services. Web services however should only rely on their own data about the user rather than requesting other resources of other web services. Thus, security issues as mentioned by Yang et al. (Yang, Lau, & Liu, 2016) are reduced.

Apart from providing the pure functionality, when lifting OAuth2 based SOAs from a local cooperative scenario to a federated scenario, it is crucial to address concerns arising from these implications.

5. ADAPTING OAUTH2 TO THE FEDERATIVE SCENARIO

The adoption of OAuth2 should of course consider already established national and international AAI federations. Furthermore, the OAuth2 workflow should not replace existing parts of the infrastructure, but extends them and to enable authorization of applications. Before granting access to an application, the user needs to supply credentials in order to log in. Integrating the token service as a service provider into the federation allows using sign on capabilities of the existing federation.

In a very centralized scenario, a federation could simply set up one authorization server. By looking at the key implications from the previous section, even this simple scenario reveals challenges of federative authorization:

- *The token service is the authority*
  Since only a single instance of the token service exists, it remains the authority for all interactions with the user and backchannel communication.
- *The token service is trusted*
  The organization behind the federation provides the token service. Local implementations may be trusted more. In addition, there may be issues with someone else managing authorizations for local users.
- *Users are known*
  In federated scenarios, this is generally not the case. The authorization needs to convey additional user information.
- *Applications and web services are separated*
  The security considerations from the cooperative implementation remain the same. The federation has to endorse service providers and application developers to use the workflows correctly.

Even though this solution is simple, two problems arise: (1) Setting up only a single authorization server, subverts the idea of many federations and their members to remain responsible for their own set of users. Consequently, the infrastructure should remain decentralized and allow each member of the federation to provide their own token service. (2) Additionally federated services usually do not meet the condition to know users in advance. It might thus be necessary to deliver a more detailed set of information about the user.

5.1. Providing Decentralization

In a second approach, several authorization servers exist in the federation. Either Users or web services need to establish a trust relationship between them and every single authorization server. In a case where users use different token services based on the resources they want to access applications might require authorizations from multiple servers and usage of web services across organization boundaries is hardly possible. In case of web services, this would require to keep a list of trusted authorization servers. A federation then is required to keep an additional directory of authorization servers. Many commercial services implement the one or the other pattern using accounts and token services of Google, Facebook, GitHub or the like by maintaining their own list of trusted services. Considering that the German federation DFN-AAI-Basic already consists of 230 identity providers this seems hardly possible.

A third approach tries to make better use of the existing federative infrastructure. The general idea is that users and web services will always interact with their local authorization server. The authorization server then handles the communication and redirection to other peers to start or verify an authorization. Users and web services keep the local authorization server as the authority; in turn, the authorization server then establishes the trust relationship to the remote server.

Either way, the federation has to maintain a directory of authorization servers, possibly by extending existing metadata directories by token service endpoints.

5.2. Providing User Information

For most web services, it is most likely necessary to provide basic user information. The existing federations participating in eduGAIN already address this second issue. Table 1 shows the results of a survey on the published metadata of the German DFN-AAI federation. The table shows the number of times a service provider requires a certain attribute. Being able to serve the top attributes generally should suffice for services to be able to establish the user context. Apart from user information like `displayName` or `mail`, attributes from the `eduPerson` schema obviously have a high relevance.

Existing federations clearly define these attributes and the service providers accessing them. Technically conveying the information as such is simple. It should however be noted that some of these attributes like `mail`, `sn` (surname) or `givenName` are personal data which should not be shared with relaying parties without the users consent. Furthermore, operators of identity providers need to verify that requested attributes are legitimate and sometimes enable them upon request for single service providers. Again, federations and local operators already acquire this information. Extending the

Table 1: Attributes required by 347 service providers in the German DFN-AAI federation

Attribute	Count
eduPersonPrincipalName	167
mail	117
eduPersonScopedAffiliation	80
sn	73
givenName	67
eduPersonEntitlement	63
eduPersonTargetedID	37
cn	18
uid	18
displayName	14
eduPersonScopedAffiliation	13
o	12

Attribute	Count
persistentId	10
eduPersonUniqueId	7
schacHomeOrganization	6
schacHomeOrganizationType	3
schacPersonalUniqueCode	3
bwidmOrgId	1
eduPersonOrgUnitDN	1
eduPersonPrimaryAffiliation	1
ou	1
dfnEduPersonTermsOfStudy	1
uniqueIdentifier	1

service provider metadata by the OAuth2 scope used by the web service allows distribution of this information in the federation.

6. IMPLEMENTATION OF FEDERATED OAUTH

Based on the findings specified in the previous sections, it is now possible to deduce a recommendation on how to implement an OAuth2 federation to comply with the existing federative scenario. Using the key implications, the proposed solution is evaluated:

- *The token service is the authority*
  Every local token service is the authority. They are the only *token* service known to users and applications. Users use a single token service to manage their authorizations.
- *The token service is trusted*
  Users and web services only trust their local token service. Federative metadata established trust between local and remote token services.
- *Users are known*
  A small set of attributes from existing federations is contemplable. Transferring them during validation provides user information.
- *Applications and web services are separated*
  The federation hast participating token services, web services distributes metadata needed to connect the services. Applications remain subject to local administration. This effectively separates services and applications.

Federative authorities are then required to keep directories of participating token and web services, reusing existing infrastructures. However, the presented findings are merely a first step towards actually establishing a federation of OAuth2 service providers. Starting from the presented model it is now possible to deduce a proposal for implementation

Figure 2 shows an overview of the participating parties and necessary communication in this workflow. Bold arrows show parts additionally needed by the federative workflow when compared to the initial workflow. These three steps, token service resolution during authorization and validation (1), token issuing (2) and remote token validation (3) therefore need closer investigation.

6.1. Token service resolution

When a user wants to authorize an application, the application redirects to the token service within its organization. This token service can then validate the legitimacy of the application and then

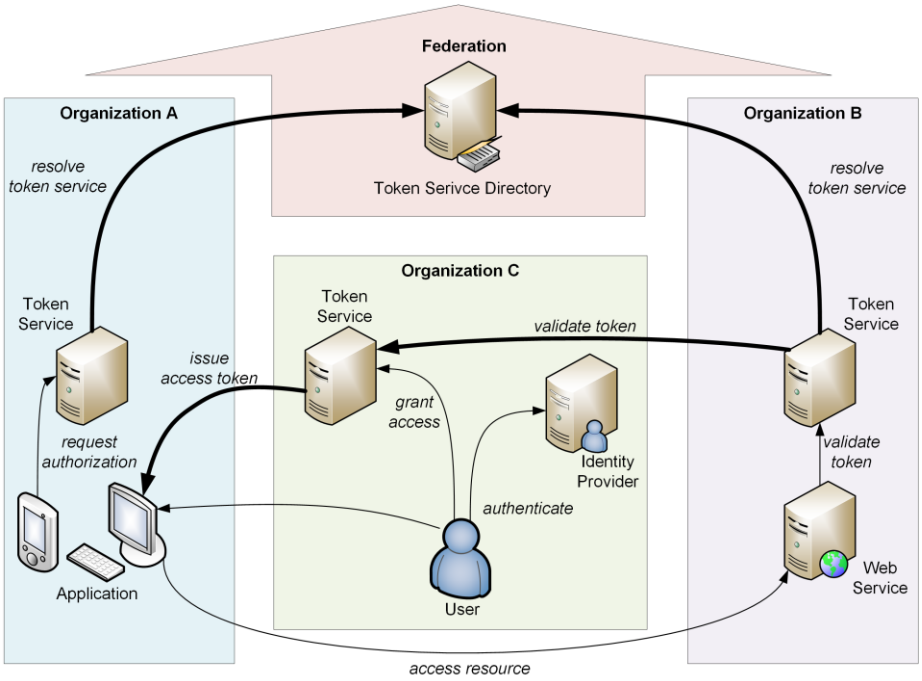

Figure 2: Schematic of the federated OAuth2 workflow

accesses the token service directory offered by the federation. For each token service in the federation, this directory should at least contain:

- *Name of the token service*
  Much like in current federative scenarios, during authentication, the user has to determine a home organization. This name should serve as a hint to find the correct organization.
- *Namespace of the token service*
  The eduPerson scheme offers several attributes, which are unique within a namespace (scoped). This namespace determines valid scopes of a token service. Comparing `eduPersonTargetedID` and `eduPersonPrincipalName` it is obvious that different kinds of techniques are used. While `eduPersonTargetedID` uses the entity ID of the identity provider, the service provider and a unique id to form a three tuple, `eduPersonPrincipalName` and other scoped attributes make use of a postfix in the form `user@namespace` (Internet2, 2012).
  In the OAuth2 federation, this namespace also applies to separate applications and web services connected to the different token services.
- *Signing key of the token service*
  While local services and their respective token services generally have a strong trust relationship, public key cryptography leverages trust among token services. By signing messages with these keys, token services confirm the authenticity of requests.
- *Endpoints of the token service*
  The five endpoints already used by the cooperative OAuth2 workflow support various authorization flows:
  - The *Authorize* endpoint authorizes tokens for server side and web applications.
  - The *Code* endpoint to request authorization codes that shown to the user used for installed applications.
  - The Token endpoint to manipulate access tokens during and after the authorization process. This includes extending the lifetime of the token or invalidating a token.
  - The *TokenInfo* endpoint supplies information about a token. Application should verify that the token is valid and actually belongs to the application.
  - The *Context* endpoint for other token services validates tokens and resolves user attributes.

According to the already existing conventions in the OAuth2 workflow, this directory could be encoded as a JSON file hosted by the federation. A single file therefore may include all token services in the federation. The format leaves room for later additions and metadata added by the federation:

```
{
  ...
  "token_services" : {
    "https://oauth.example.com" : {
      "displayName" : "Example University",
      "namespace" : "example.com",
      "key" : "-----BEGIN PUBLIC KEY-----\nMIGfM...",
      "endpoints" : {
        "authorize" : "https://oauth.example.com/authorize",
        "code" : "https://oauth.example.com/code",
        "token_info" : "https://oauth.example.com/token_info",
        "context" : "https://oauth.example.com/context"
      }
    },
    ...
  }
}
```

6.2. Token Issuing

After the user selected a home organization, the token service signs the authorization request and, depending on the authorization flow, redirects the user to the home token service or relays the response to the application that in turn redirects the user. The users' home token service validates the signature and starts its authentication workflow. The user may now log in and grant access for the application. Based on the currently used workflows for authorizing installed applications (e.g. apps on a smartphone) and web applications additions to the workflow arise:

In the device workflow, the token service issues a temporary device code and a user code as in the sample below. The user then has to enter the code on a web page. For devices that do not have rich input abilities the user has to copy the verification URL and user code manually to a computer. Many devices, like smartphones, offer to open a web browser directly. If the user still has to copy a code, this results in very bad usability.

```
{
  "device_code" : "BaUAJHPFYFi6wKU0WY5xLC",
  "user_code" : "SFW7WZXK7G",
  "verification_url" : "https://oauth.example.com/verify",
  "expires_in" : 1800,
  "interval" : 5
}
```

In order not to undermine cases where the user has to type the verification URL, it should remain as short as possible. Allowing user friendly transmission of the code, token services should however also accept URLs of the form:

```
https://oauth.example.com/verify?user_code=SFW7WZXK7G
```

In contrast to the device workflow, the web application authorization obviously requires a web browser. It relies on HTTP redirects and ends by directing to a page of the application. For the security of the OAuth2 workflow, it is very important that the last redirection URL actually is valid for the application since authorization tokens could otherwise be hijacked. The user's home organization however does not know the application in this case. The token service in the applications organization therefore has to verify that the redirect URL actually belongs to the application prior to signing and redirecting the authorization request.

In order to identify the token service that issued the token, it should also include the namespace of the token service in the form `token@namespace`.

6.3. Remote Token Validation

When accessing web service resources, applications pass the authorization token. As in the cooperative workflow, the web service validates the token against the token service in its own organization. The token service validates the request and its origin. Using the namespace, the token service can now resolve the user's home organization and the associated token service. The token service signs and then forwards it to the remote token service.

Using the signature, the token service in the user's home organization, is able to validate that the request is legitimate. It verifies, furthermore, that the token is actually valid. Additional attributes to establish the user context at the web service complete the response. By default there should be at least one attribute uniquely identifying the user for the service. The aforementioned `eduPersonPrincipalName` therefore seems like a viable choice. Many services however need additional attributes, commonly `mail` or `givenName`. If web services require further attributes operators of the organization on demand can add them. Based on the findings in table 1 and if compatible with local privacy conventions a "zero configuration" attempt for locally unknown web services the three mentioned attributes accompanied with `eduPersonScopedAffiliation` are a tradeoff to reduce the need for manual intervention.

A valid response from the token server of the user's home organization in JSON format could look like this

```
{
  "isValid" : true,
  "application" : "ahcndwlsajcnalfejalsd@example.com",
  "mail" : "max.power@example.com",
  "displayName" : "Max Power",
  "eduPersonPrincipalName" : "anpqr7d@example.com",
  "eduPersonScopedAffiliation" : "student@example.com"
}
```

The token service receiving the validation of the token then needs to verify that all scoped attributes actually belong to the same namespace the token suggested. Only then, it should forward the validation response to the web service.

7. CONCLUSION AND FUTURE WORK

In order to complete the presented workflow, it is still necessary to address and fully define some security considerations. For example, the actual technology for signing responses and requests is still undefined. JSON Web Tokens (JWT) (Jones, Bradley, & Sakimura, 2015) seem like a natural choice. Furthermore, additional security measures may be necessary to protect the token service directory from manipulation.

A "zero configuration" policy like in the proposed implementation intentionally sacrifices strong privacy requirements in order to remain as simple as possible for application and web service developers but also for operators of the token service. At the current stage all applications, web services and token services are well known and go through the same privacy review. As the number of associated services increases, these policies should undergo a review. Still the threshold to access to this kind of services should be as low as possible to allow innovative ideas and concepts to focus on the core of their implementation. A strict separation of applications offered by third parties and web services offered by organizations is the starting point for a reasonable compromise between privacy and flexibility.

The presented extension of the OAuth2 workflow gives an example on how to integrate OAuth2 services into current federations. Analyzing the main challenges, *Token Service Resolution* and *Providing of User Data* yielded necessary information for the practical implementation. An implementation of the prototype is currently in review for bugs and security issues. Another project furthermore evaluates the prototype for joint services of RWTH Aachen University and Jülich Super Computing Centre. Further projects are in a planning phase. This furthermore underlines the importance to find a viable solution for personalized access to SOAs.

8. REFERENCES

Adams Becker, S., Cummins, M., Davis, A., Freeman, A., Hall Giesinger, C., & Ananthanarayanan, V. (2017). *NMC Horizon Report: 2017 Higher Education Edition.* Austin, Texas: The New Media Consortium.

Barata, R., Silva, S., Martinho, D., Cruz, L., & Guerra e Silva, L. (2014). Open APIs in Information Systems for Higher Education. Umea.

Cantor, S., Kemp, J., Philpott, R., & Maler, E. (Eds.). (2005). *Security Assertion Markup Language (SAML) V2.0.* OASIS Standard.

Grabatin, M., Hommel, W., Metzger, S., & Pöhn, D. (2016). Improving the Scalability of Identity Federations through LevelofAssurance Management Automation. Bonn.

Hardt, D. (2012). *The OAuth 2.0 Authorization Framework.* RFC Editor.

Hedberg, R., Gulliksson, R., Jones, M. B., & Bradley, J. (2016). *OpenID Connect Federation 1.0 - draft 01.* (The OpenID Foundation, Producer) Retrieved from http://openid.net/specs/openid-connect-federation-1_0.html

Internet2. (2012). eduPerson Object Class Specification. (Internet2 Middleware Architecture Committee for Education, Directory Working Group, Ed.)

Jones, M., Bradley, J., & Sakimura, N. (2015). *JSON Web Token (JWT).* RFC Editor.

Knight, J., Cantor, S., Fisher, D., Putman, B., Widdowson, R., Young, I., & Zeller, T. (2014). *Shibboleth.* (Shibboleth Consortium, Producer) Retrieved 2 22, 2017, from https://shibboleth.net/

Lehmann, A., & Huber, R. (2015). Das Studierenden-App Projekt StApps. In H. Pongratz, *DeLFI 2015 - die 13. E-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V.* Bonn: Ges. für Informatik.

Li, W., & Mitchell, C. J. (2016). Analysing the Security of Google's Implementation of OpenID Connect. In J. Caballero, *Detection of intrusions and malware, and vulnerability assessment.* Cham; Heidelberg: Springer.

Mincer-Daszkiewicz, J. (2014). We Publish, You Subscribe — Hubbub as a Natural Habitat for Students and Academic Teachers. Umea.

Politze, M., Schaffert, S., & Decker, B. (2016). A secure infrastructure for mobile blended learning applications. In J. Bergström, *European Journal of Higher Education IT 2016-1.* Umeå.

RfII – German Council for Scientific Information Infrastructures. (2016). Performance through Diversity - Recommendations regarding structures, processes, and financing for research data management in Germany. Göttingen.

Sakimura, N., Bradley, J., Jones, M. B., Medeiros, B., & Mortimore, C. (2014). *Openid connect core 1.0.* (The OpenID Foundation, Producer)

Vogl, R., Angenent, H., Rudolph, D., Thoring, A., Schild, C., Stiegliz, S., & Meske, C. (2015). "sciebo — theCampuscloud" for NRW. Dundee, Scotland.

Yang, R., Lau, W. C., & Liu, T. (2016). Signing into One Billion Mobile App Accounts Effortlessly with OAuth2.0. In *Black Hat Europe.*

9. AUTHOR'S BIOGRAPHY

Marius Politze, M.Sc. is research associate at the IT Center RWTH Aachen University since 2012. His research is focused on service oriented architectures supporting university processes. He received his M.Sc. cum laude in Artificial Intelligence from Maastricht University in 2012. In 2011, he finished his B.Sc. studies in Scientific Programming at FH Aachen University of Applied Sciences. From 2008 until 2011, he worked at IT Center as a software developer and later as a teacher for scripting and programming languages.