Guidelines for credential delegation

Published Date:	13-06-2017
Revision:	1.0
Work Package:	JRA1
Document Code:	AARC-JRA1.4D
Document URL:	https://aarc-project.eu/wp-content/uploads/2017/03/AARC-JRA1.4D.pdf

The research leading to these results has received funding from the European Community's Horizon2020 Programme under Grant Agreement No. 653965 (AARC).

Table of Contents

The research leading to these results has received funding from the European Community's Horizon2020 Programme under Grant Agreement No. 653965 (AARC).

1 Introduction

Federated single sign-on (SSO) has focused on providing authentication and access control for websites accessed directly by the user's browser. For example, logging in to a library resource, or accessing a virtual learning environment (VLE). The user's IdP passes information to the service he/she wishes to access. This assertion is usually strictly limited to being valid only on the website it was created for.

However, in distributed environments it is often necessary for a remote service to access other services on behalf of a user, or for a software agent to act on behalf of the user. In this case, it is necessary to securely delegate the user's "rights" from the website he/she originally accessed to a wide variety of other applications, such as mobile applications, intranet services and HPC clusters.

The oldest form of credential delegation is to store and reuse the user's login password. This is neither possible nor safe in a distributed environment. Current delegation credentials include signed assertions, session tickets, "tokens" of various types, and proxy certificates.

2 Delegation

This document covers requirements for delegation, to aid (1) the selection of the optimal technology for implementing delegation, and (2) to deploy and operate the technology. While the examples use specific technologies, they should also illustrate the situations where delegation is useful.

2.1 Types of Delegation

The term "delegation" is sometimes used to cover slightly different scenarios:

A. Delegation of **rights** to another person. Combined with RBAC, this becomes entirely an authorisation question and is outside the scope of this document. The delegatee is typically a person, but could also be a service.
B. Delegation of **access** to another person - a client, service, or person requests and obtains the right from the "owner" of a resource to access the resource, as in OAuth2. The token is typically issued for a limited time and purpose.
C. **Credential** delegation (sometimes called impersonation). The most primitive example is copying the username/password as mentioned above; more useful examples include Kerberos proxiable tickets (RFC1510) and GSI (RFC3820): a remote service (typically a host/service, not a person) obtains a full or limited credential with which it can act on behalf of the user.

The research leading to these results has received funding from the European Community's Horizon2020 Programme under Grant Agreement No. 653965 (AARC).

The first point (type A) is out of scope for this document (it will be covered in AARC2); the more widely used second and third points are covered here.

2.2 Delegation Features

This section summarises feature-related points to consider when selecting technology/implementations for delegation (beyond the usual questions of maturity of technology, interoperation of implementations, etc.)

The discussion considers a scenario in which a delegatee acts on behalf of a delegator (and by assumption is authorised by the delegator), and a token is issued to the delegatee by some authority to enable it to do so.

1. Can the participants (delegator, delegatee) be humans/automated?
    a. For the authentication of human participants, is federated identity management (FIM) supported?

    (Note that the resource, to which access is delegated, is usually non-human.)
2. How does the delegation integrate with existing authorisation?
3. How is the token validated?
    a. By the delegatee?
    b. By the resource accessed by the delegatee?
    c. Can the token be revoked, such that validation by the resource will fail?
4. Is the technology web-based, or does it support non-web access?
    a. If web-based, does it work with basic HTTP clients such as curl (plus perhaps some other standard components, such as XML or JSON parsers)? Or does it require that the client be a browser (needs JavaScript, user intervention)?
    b. If it supports non-web access, does it also work with web servers/clients?
5. Does it support onward delegation (from the delegatee to a second delegatee)?
    a. With the originating user's permission?
    b. Without?
6. Can the scope of the delegated credential be limited? Options include:
    a. Limited time (possibly pre-dated, valid in the future).
    b. Locked to a particular delegate.
    c. Not further proxiable.
    d. Limited number of uses.
    e. Limited set of activities it can be used for.
7. Is the delegatee's use of the delegated token logged/audited?
8. How is the delegated credential stored and protected by the delegatee?

The research leading to these results has received funding from the European Community's Horizon2020 Programme under Grant Agreement No. 653965 (AARC).

3 Examples

3.1 Example 1 (Type A)

Alice is responsible for a VO, so has an "admin" role which gives her rights to define and modify attributes for members of the VO. However, Alice is going on holiday, so she *temporarily* assigns the admin role to Bob: during this holiday period, Bob can manage members' attributes just like Alice except he cannot delegate the admin right further. At the end of Alice's holiday, Bob's admin assignment times out and he reverts to a normal member role.

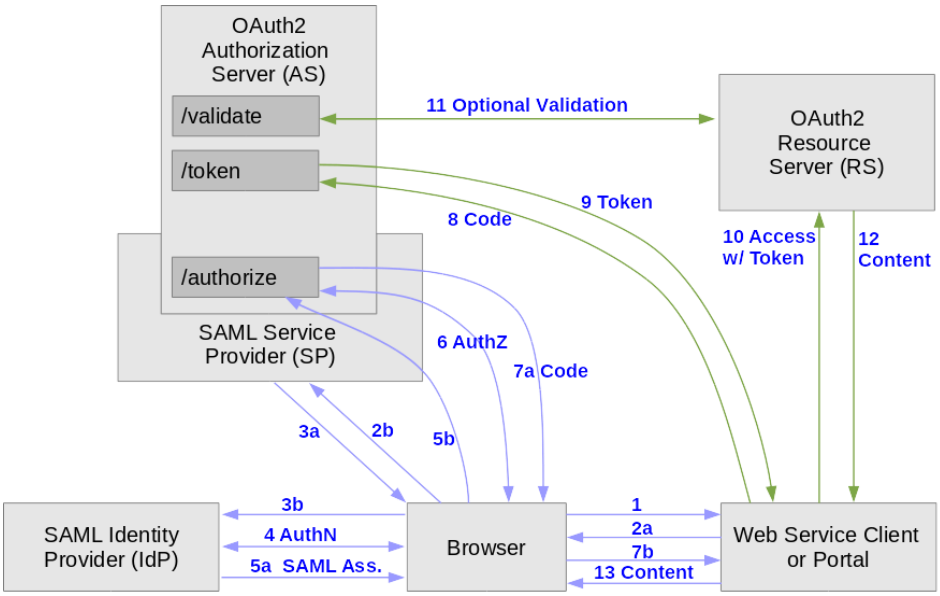
Note that this is different from Alice nominating Bob as her deputy (or of Bob impersonating Alice) in three respects: first, the delegation of authority is time limited, and secondly, Bob's rights are restricted to the delegated role; he cannot do everything that Alice can do. Finally, Bob cannot delegate the right further.

The technologies behind this are out of scope for AARC1 and will be covered in more detail in AARC2.

3.2 Example 2 - OAuth2/OIDC (Type B)

The "classic" OAuth2 delegation scenario is a user delegating rights to a printer to access their photos - this is described in the introduction to RFC 6749 [OAUTH2]. In this section, we show two extensions of delegation based on OAuth2 and/or OIDC [OIDC].

3.2.1 Example 2a - Delegation for SAML authentication (Type B):



The research leading to these results has received funding from the European Community's Horizon2020 Programme under Grant Agreement No. 653965 (AARC).

This example is based on a SAML-based authentication-infrastructure and focuses on a use-case where a client ("Web Service Client or Portal") wants to access another web-service ("OAuth2 Resource Server (RS)") on behalf of the user. There are two problems when doing this in SAML: a) the delegation step is difficult to realise with pure SAML (the assertion would have to be securely redirected to the web-service, which is not a common use-case) and b) the web-service might be REST-based. Therefore, the SAML-based infrastructure is combined with the OAuth2 authorization code flow to handle delegation. Once the client wants to access the web-service it issues an Authorization Request (step 2). A SAML SP protects the Authorization Endpoint and handles authentication (steps 3-5). Note that this is the only place where SAML-based communication takes place. After that the usual OAuth2 authorization code flow continues (steps 6-9) and finally the client can access the web-service with the access token (steps 10-12). Note that the client is mainly a OAuth2 client in this scenario but can also be a SAML SP at the same time.

The EUDAT B2ACCESS, CIlogon, and RCauth services work the same way: services can internally (within the relying party infrastructure) use OAuth for delegation, with the user authenticated through (external) SAML IdPs. See also example 3b.

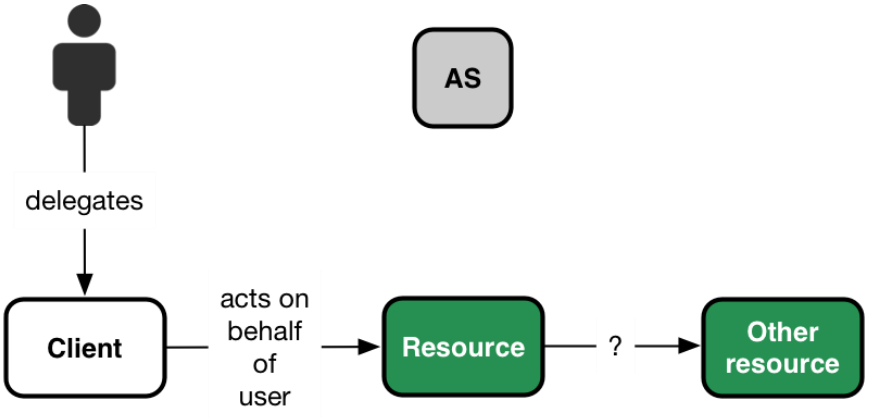
See also [RFC7522].

3.3 Example 2b - INDIGO use of OIDC (Type B)

(This example requires some knowledge of the OAuth2 protocol)

OpenID Connect is used for the authentication and OAuth2 for authorization within INDIGO. The authorization using OAuth2.0 has two major drawbacks:

1. The OAuth2.0 specification does not specify a way to redelegate (and limit) access to a third instance.
2. OAuth2.0 does not specify a flow to gain long lived access for a downstream service.



The research leading to these results has received funding from the European Community's Horizon2020 Programme under Grant Agreement No. 653965 (AARC).

The first issue results in the access token issued for service A to be unusable for service B, which from a security point of view is usually desirable. To put it differently, if service B is allowed to use the token directly, it will have the same permissions as service A, which is not a good approach from the security point of view.

The question is, what if service A needs to delegate its access to service B so that the service B can perform only a limited set of tasks on the behalf of the user?

The second issue is slightly more complicated than the first one. This time service B needs a long-lived refresh token, instead of an access token which is bound to the service (client), in this example A. The service B needs the refresh token to perform tasks longer than the lifetime of the access token, like performing long running jobs on behalf of the user.

The INDIGO DataCloud IAM (Identity and Access Management) works around these limitations by implementing the needed parts of the 'OAuth 2.0 Token Exchange' [OAUTH2-TE]. This proposal extends OAuth 2.0 to enable both clients and delegated clients to request and obtain security tokens from authorization servers (ASs). The specification defines a new grant type for a token exchange request and the associated specific parameters for such a request to be submitted to the token endpoint.

Upon successful authentication of the OAuth client (delegator) to the AS (IAM) with one token, a new token is issued to the client, which can be forwarded to the delegatee. The new token might be an access token that is more narrowly scoped for the resource, or it could be an entirely different kind of token, including, but not limited to, bearer or JSON web token (JWT).

If the requested and granted scope includes the 'offline' scope, the resource will be able to request a long-lived refresh token using the freshly minted token. The refresh token will, following the OAuth2.0 specification, be bound to that specific client (delegatee).

The implementation of this feature enables the INDIGO architecture to delegate access and refresh token in a controllable manner through IAM.

3.4 Example 3a - GSI proxies

In grid job submission, the users of the grid are authenticated using X.509 certificates. The job runs on a worker node but needs to access resources on behalf of a user, e.g. a storage element. The delegation is done with GSI proxies [RFC3820] that is, short-lived certificates that are issued and signed with the user's personal certificate or with other GSI proxies. This is normally not permitted, but GSI allows for this provided restricted naming conventions are followed (i.e. users issue delegated certificates only in their own name, not in other users' names.)

Multi-step delegation appears naturally in this scenario by creating new GSI proxies from existing GSI proxies, thus forming a proxy certificate chain. Such multi-step delegations are necessary in for example job-

The research leading to these results has received funding from the European Community's Horizon2020 Programme under Grant Agreement No. 653965 (AARC).

submission, where the job has to go through a number of services before landing on a worker node, or for so-called third-party data transfers, where large data files are transferred directly between two storage elements on behalf of the user.

GSI certificates are used not just to delegate, but also to enable use of credential stores (such as the MyProxy server), and to "decorate" certificates with authorisation attributes, or with other attributes which limit the scope, including limiting the number of onward delegations. In case of abuse, the impact can be mitigated by revoking the end-entity certificate, which will result in the verification of the proxy chain to fail.

In all cases, the general convention is that the private key does not move (see RFC 3820 section 2.7 and the MyProxy protocol [MYPROXY] section 2.4); thus, a service which wishes to obtain a delegated credential from the user generates a key pair and a request, and sends the request to the delegator (which may form part of the chain, with the user themselves being the earliest delegator).

Questions may arise when a longer delegated chain arrives at a service, with more than one of the delegated/delegating certificates "decorated" with authorisation extensions. These issues were investigated by the OGF VOMSPROC and IDEL working groups.

3.5 Example 3b - Combined use of X.509 and OIDC

In the portal delegation scenario of the US-based CIlogon service, the flow as described in example 2a is used to get a user's end-entity certificate onto a web portal (acting as OIDC client) by putting an online CA behind an OIDC protected resource (acting as OIDC Resource Server). Logging in at the Authorization Server is typically done using a SAML SSO flow. The OIDC access token is used by the client to obtain the certificate from the online CA.

This scenario has been further extended and adapted to the European infrastructure and policy requirements and let to the development of the RCauth.eu service. In the latter scenario, a full web-flow becomes feasible, where a Science Gateway can easily obtain a short-lived proxy certificate which can be used in all the use-cases where GSI certificates are used, such as job-submission on grid-infrastructure, accessing storage elements etc., but in a way that they remain completely hidden for the end-user.

4 Guidelines for Implementing Delegation

For integrator/developer/architect, faced with the need to delegate within their infrastructure, a decision needs to be made regarding the technologies to use. As can be gleaned from the examples, different projects solve the same problem in different ways, either because they have to integrate with different existing infrastructures or because they require different features. Suggested steps are:

1. Decide which features you need (see section on features)
2. Select technology, preferring standards-based and interoperable, then mature.

The research leading to these results has received funding from the European Community's Horizon2020 Programme under Grant Agreement No. 653965 (AARC).

3. Look up best practices for running it and implement them
4. Check the risks.
5. Ask for help from the experts

The key point is to select the technology based on the features required; once the selection has been made, the technology and the features (including the features not supported by the technology) may give rise to specific risks, which should be assessed.

4.1 Example of Feature Selection

The following table illustrates technology selection based on features, focusing on three very common and widely supported technologies: GSI proxies, OIDC/OAuth2 and Kerberos. There are other technologies with similar or different feature sets; in particular, the editorial decision has been taken to not cover SAML ECP[1] [SAML-ECP-1], Macaroons [MACAR], iRODS tickets [IRODS], Dynafed [DYNAF], and other relevant but less widely used technologies.

Feature Required	GSI proxies	OIDC/OAuth2	Kerberos
Human/automated participants	Both	Limited[2]	Both
Integration with existing authorisation	Fully integrated with VOMS	Own authorisation	Via LDAP/Active Directory
Token validated?	Verify digital signature of the chain and check for revocation	Call-out to AS	Digital signature
Web/non-web (primarily delegatee/resource)?	Mainly non-web	Mainly web	Both

[1] The delegation part of the specification was not included in the main Shibboleth implementation.

[2] The "Resource Owner" (cf. RFC 6749) generally has to be human (i.e. the "End-User" of [OIDC-GUIDE]), with a browser; however, see also Section **Error! Reference source not found. Error! Reference source not found.**.

The research leading to these results has received funding from the European Community's Horizon2020 Programme under Grant Agreement No. 653965 (AARC).

Onward delegation?	Yes, chained (RFC 3820)	No, but see [AARC-JRA1.4D] for discussion	Yes, at least within realm
Delegation can be restricted (in the sense of Feature 6)	Partly	Depends on token type	Partly
Logged/audited use?	Yes	Yes (in AS)	No
How is it protected?	Filesystem	Filesystem	Filesystem
Revocation	Yes	Implementation dependent	No
Time limitation	Proxies are conventionally short-lived (O(12h))	Implementation dependent	Yes

Admittedly, this table does not quite do justice to the discussion, and only covers the high-level questions; but hopefully it should illustrate the selection process for different delegation technologies, and it could point the way to further work on delegation, if needed.

5 Risks associated with delegations

The table below summarises several general risks associated with delegation. It does not cover the technology-specific risks. (For example, if OAuth is used with bearer tokens (RFC 6750), there is a risk that a stolen token could be misused, as it intentionally does not support Feature 6.b. (Locked to a particular delegate), but it thus allows delegation to unregistered clients. If the use of the token (through the validation) is logged by the Authorisation Server (Feature 7), it could help mitigate this risk.)

	Description	Type	Owner	Mitigation
0	Delegation is not supported by protocol or infrastructure	Proto	Infrastructure	See earlier subsections of this section and the guidelines on token

The research leading to these results has received funding from the European Community's Horizon2020 Programme under Grant Agreement No. 653965 (AARC).

				translation services.
1	Delegated credential is compromised	Op	Infrastructure	Use revocable and/or short-lived credentials. Adopt best practices for operational security (see [SIRTFI-1] sections OS, IR, and PR.)
2	Delegated credential used for non-user-approved activities, by being too widely applicable, or by being forwarded/delegated without the user's consent	Op	User	Credentials with limited use/ purpose/locality and/or which are revocable. Auditable (user-visible) use of delegated credential.
3	Delegated credential is delegated further (without authorisation)	Op	User	Special case of Risk 2 (or Risk 1, depending on point of view). See [AARC-JRA1.4D].
4	Lack of clarity in interpretation of rights of delegated credentials, particularly credentials delegated multiple times (combining restrictions)	Tech, Policy	Infrastructure	For example, see the work by the OGF VOMS attribute PROCessing working group [VOMS-PROC] on VOMS Attribute Certificate Parsing Rules for Chained Identity Credentials.
5	Delegated credential not capable of inheriting same (or selected) authorisations as user credential	Tech	User	Bugfix – may need changes to SP to support delegated credential
6	User cannot fine-tune limits on delegated credential so sets the most general limits or, if possible, turns them off	Tech, Usability	Infrastructure	Test with real users
7	Delegations don't work (or features	Tech	User	Needs research

The research leading to these results has received funding from the European Community's Horizon2020 Programme under Grant Agreement No. 653965 (AARC).

	are lost) in a federated environment (e.g. beyond scope of IdP)	Proto		

6 References

[AARC-JRA1.4D] Guidelines for credential delegation
https://aarc-project.eu/wp-content/uploads/2017/03/AARC-JRA1.4D.pdf

[DYNAF] Dynafed – The Dynamic Federation Project web page
http://lcgdm.web.cern.ch/dynafed-dynamic-federation-project

[IRODS] iRODS Docs – Tickets (Guest Access)
https://docs.irods.org/4.2.0/system_overview/users_and_permissions/#tickets-guest-access/

[MACAR] A. Birgisson et al., Macaroons: Cookies with Contextual Caveats for Decentralized Authorization in the Cloud, Network and Distributed System Security Symposium, 2014
https://www.internetsociety.org/doc/macaroons-cookies-contextual-caveats-decentralized-authorization-cloud/

[MYPROXY] The MyProxy Protocol
http://grid.ncsa.illinois.edu/myproxy/protocol/

[OAUTH2] The OAuth 2.0 Authorization Framework
https://tools.ietf.org/html/rfc6749

[OAUTH2-TE] OAuth 2.0 Token Exchange (Draft)
https://tools.ietf.org/html/draft-ietf-oauth-token-exchange-08

[OIDC] OpenID Connect
http://openid.net/connect/

[OIDC-GUIDE] OpenID Connect Basic Client Implementer's Guide 1.0 – draft 37
http://openid.net/specs/openid-connect-basic-1_0.html

[RFC3820] Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile
https://tools.ietf.org/html/rfc3820

[RFC7522] Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants
https://tools.ietf.org/html/rfc7522

[SAML-ECP-1] Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS, March 2005
http://docs.oasis-open.org/security/saml/v2.0/

[SIRTFI-1] T. Barton et al., A Security Incident Response Trust Framework for Federated Identity (Sirtfi)
https://refeds.org/wp-content/uploads/2016/01/Sirtfi-1.0.pdf

[VOMS-PROC] OGF VOMS attribute PROCessing Working Group
https://redmine.ogf.org/projects/voms-proc-wg

The research leading to these results has received funding from the European Community's Horizon2020 Programme under Grant Agreement No. 653965 (AARC).

7 Glossary

ECP Enhanced Client Protocol

FIM Federated Identity Management

GSI Grid Security Infrastructure

JSON JavaScript Object Notation (RFC 7519)

OIDC OpenID Connect

RBAC Role-Based Access Control

REST REpresentational State Transfer (web services design principle)

SAML Security Assertion Markup Language

VLE Virtual Learning Environment

VO Virtual Organization

The research leading to these results has received funding from the European Community's Horizon2020 Programme under Grant Agreement No. 653965 (AARC).