*I*nternational

*V*irtual

*O*bservatory

*A*lliance

IVOA credential-delegation protocol

Version 0.1

IVOA WD 2007 August 23

This version:

Not posted yet

Latest version:

Not posted yet

Previous version(s):

None issued

Author(s):

Ray Plante

Guy Rixon (editor)

Giuliano Taffoni

Abstract

The credential-delegation protocol allows a client programme to delegate a user's credentials to a service such that that service may make requests of other services in the name of that user. The protocol defines a REST service that works alongside other IVO services.

Status of This Document

This is a working draft. The first release of this document was YYYY Month DD.

This is an IVOA Working Draft for review by IVOA members and other interested parties. It is a draft document and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use IVOA Working Drafts as reference materials or to cite them as other than "work in progress".

A list of current IVOA Recommendations and other technical documents can be found at http://www.ivoa.net/Documents/.

Acknowledgements

The concept of delegation by impersonation was promoted by the grid-computing movement and particularly by the Globus project. The protocol described below is derived from the delegation service in Globus Toolkit 4.

Contents

1 Introduction

Some services in the IVO have restricted access; some users have access rights. When a client programme makes a requests to one of these secured services, it is typically doing so in the name of the user running the client. I.e., the client presents to the service credentials authenticating the user's identity.

"Presents to the service credentials" means that the client sends the public credentials (an X.509 certificate) but not the private credentials (the private key matching the public key in the certificate). The client authenticates its right to use the identity in the certificate by proving that it holds the private key. It does this using one of the approved authentication methods [1], either TLS or digital signature.

Now consider a secured service -- call it the “agent” -- which needs to drive other secured services. This might be a “broker” service accessing restricted archives, or it might be a DAL service storing query-results in VOSpace. The agent has the certificate for the user's identity but it does not have the private key, so it cannot authenticate the user's identity.

We need a way for the the agent to get a private key. Sending the user's own private key across to the network is too dangerous and vulnerable to interception. The alternative, on which this IVOA delegation protocol is based, is to generate a proxy identity tied to the user's identity, with a certificate based on a key pair generated by the agent. The agent can then authenticate the proxy identity to other services. Those other services recognize a proxy identity by the annotations in its certificate and accord it the same access rights as the primary identity from which the proxy is derived.

This method of delegation is called “delegation by impersonation” and is common in grid computing [2].

The certificate for a proxy identity is commonly called a proxy certificate or simply “a proxy”. IETF RFC 3820 [3] defines the content and encoding of these certificates.

The subject or “distinguished name” (DN) in a proxy certificate is based on the subject of the certificate from which the proxy is derived. E.g. if an end-entity certificate (EEC; the long-term certificate issued to a user) authenticates the identity

> *C=UK, O=AstroGrid, OU=IoA Cambridge, CN=Guy Rixon*,

then a proxy made from this will authenticate

> *C=UK, O=AstroGrid, OU=IoA Cambridge, CN=Guy Rixon, CN=proxy*.

To trust the identity in a certificate, an entity must be able to trace a chain of signatures back to a trust anchor. When a proxy certificate is made from an EEC, then the EEC becomes part of this chain. Therefore, whenever a proxy is sent to a service to authenticate an identity, the EEC from which the proxy is derived must also be sent.

A proxy certificate may be derived from another proxy instead of directly from an EEC. This is normal in the IVO, since many desktop applications get their credentials as proxies; they never see a private key matching an EEC.

In IVOA protocols, an agent must delegate credentials *before* calling a service that needs to use delegated credentials. The agent can find out the need for delegation from the service registration.

The delegation process has these steps.

1. The client commands the agent to generate and store a key pair for a particular identity.
2. The client retrieves from the agent a certificate-signing request (CSR) containing the public key.
3. The client generates from the CSR a certificate, signs it and gives the certificate to the agent.

2 Delegation protocol

2.1 Required web-resources

2.1.1 Specification

An agent that needs to receive delegated credentials shall provide these web resources, accessible by HTTP:

- a list of delegated identities
- one resource for each delegated identity
- for each identity, a CSR
- for each identity, a certificate.

The URIs of these resources shall satisfy the following constraints on their paths.

- The delegated identities shall be children of the list of identities.
- The CSR shall be a child of its delegated identity, and shall be named *CSR*.
- The certificate shall be a child of its delegated identity, and shall be named *certificate.*

2.1.2 Commentary

The delegation protocol is RESTful and concerns a tree of web resources, most of which are created in the process of delegation. Only the delegation list is a static resource.

The delegation-list resource can have any name.

The resources for the identities can have any name, but that name should be easy to encode into the URI; X.500 DNs, as taken from the certificates, are hard to read when URL-encoded. A better choice is a hash of the DN that is unique within the delegation service. Java implementations that cache an object for each identity can use the result of the *hashcode()* method to name the identity resource.

A typical tree of resources might look like this:

/delegations	(delegation list)
/delegations/012345678/	(identity)
/delegations/012345678/CSR	(CSR)

/delegations/012345678/certificate (proxy certificate)

2.2 Representations of resources

2.2.1 Specification

The CSR shall be represented as a PKCS#10 [4] CSR with PEM encoding.

The certificate shall be represented as an X.509v3 [5, 6] certificate with PEM encoding. More specifically, the certificate shall be a proxy certificate following the rules of RFC 3820 [3]. The *proxyPolicy* field of the certificate shall be set to the special value *id-ppl-inheritAll*, as defined in section 3.8.2 of RFC 3820.

“PEM encoding” mean that the text of the credential is written out as a byte stream according to the Distinguished Encoding Rules [7] of ASN.1 [8] and that stream is re-encoded in base 64.

The representation of the delegations list is not fixed by this standard. An implementation should include in the representation the URI for each identity web-resource.

The representation of an identity shall be the distinguished name written out as a string according to RFC 2253 [9], with MIME type *text/plain*.

2.2.2 Commentary

The client and agent actually exchange CSR and certificate, so the representation of these resources is fixed. The other resources only need to be read for debugging purposes, so their representations should be human readable rather than machine readable. MIME type *text/plain* is suggested.

The *proxyPolicy* constraint means that the proxy identity represented by the certificate inherits all access rights of the identity from which it is derives; this kind of proxy certificate is colloquially called an “impersonation proxy”.

2.3 Operations on the resources

2.3.1 Specification

All resources shall respond to HTTP GET with the representations described above.

An HTTP POST to the delegations list shall create a proxy identity based on the identity passed in the request-parameter named *DN*; the identity in the parameter shall have the same representation as the identity web-resource, described above, except that the characters shall be escaped as necessary according to the URL-encoding rules of RFC 1738 [10]. On creating the identity, the agent shall create and store an RSA key-pair. The agent shall then create the web resource for the proxy identity and the matching CSR. The agent shall return HTTP status 201 “created” and shall include in the response the HTTP header named *Location* whose value is the absolute URI of the resource representing the created identity.

An HTTP DELETE to the resource for an identity shall delete that resource, its child resources and the associated private key.

An HTTP PUT to the certificate resource of an identity shall upload the certificate. The agent shall store this certificate.

The agent shall reject any POST, PUT or DELETE requests for other resources in the delegations tree.

2.3.2 Commentary

The natural sequence of operation is:

1. POST the delegated identity to the list;
2. GET the CSR;
3. PUT the certificate;
4. (later, after delegated powers have been used) DELETE the identity.

Trying to get the CSR or put the certificate before posting the identity (or after deleting it) naturally fail with a 404 “not found” error, since the resources do not exist. Getting a certificate before putting it has an undefined result: it can't work, but the result might be “not found” or “no content”.

It is left to the implementor to decide for how long to keep the credentials. The likely choices are

- until the next restart of the service;
- until the credentials expire and become invalid (as noted in the certificate);
- for ever.

Note, however, that delegating new credentials for the same identity necessarily replaces the old credentials and that the client can choose to delete the delegated credentials when they are no longer needed.

2.4 Using the delegated credentials

2.4.1 Specification

The service which receives delegated credentials must authenticate the sender of any requests that would use those credentials. The sender must have the use of the identity from which the proxy credentials are derived or else the request shall be denied.

The implementation must choose the interfaces by which the delegated credentials are propagated to the software that uses them. These interfaces must protect the private keys from copying by improper persons.

2.4.2 Commentary

In a Java implementation, the delegation resources and stored credentials might be managed by one servlet and the science service that uses them by another. These servlets would occur in the same virtual-machine and can pass credentials as objects in memory, never committing them to disc. This is generally quite secure enough.

If the science code cannot receive the credentials from memory, then precautions against copying are needed. Possible approaches are

1. A shared file that is an encrypted key-store.
2. Transfer via a MyProxy service, protected by TLS.

References

[1] G. Rixon, *IVOA Single-Sign-On Profile: Authentication Mechanisms*, http://www.ivoa.net/Documents/latest/SSOAuthMech.html

[2] *GT 4.0 Security: Key Concepts*, http://www-unix.globus.org/toolkit/docs/4.0/security/key-index.html

[3] S. Tuecke, V. Welch, D. Engert, L. Pearlman, M. Thompson, *Internet X.509 Public Key Infrastructure (PKI): Proxy Certificate Profile,* IETF RFC 3820, http://www.ietf.org/rfc/rfc3820.txt

[4] M. Nystrom, B. Kaliski, *PKCS #10: Certification Request Syntax Specification*, IETF RFC 2986, http://www.ietf.org/rfc/rfc2986.txt

[5] R. Housley, W. Ford, W. Polk, D. Solo, *Internet X.509 Public Key Infrastructure: Certificate and CRL Profile,* IETF RFC 2459, http://www.ietf.org/rfc/rfc2459.txt

[6] *Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks*, ITU-T recommendation X.509, http://www.itu.int/rec/T-REC-X.509/en

[7] *ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)* , ITU-T Recommendation X.690, http://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf

[8] *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*, ITU-T Recommendation X.680, http://www.itu.int/ITU-T/studygroups/com17/languages/X.680-0207.pdf

[9] M. Wahl, S. Kille, T. Howes, *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*, IETF RFC 2253, http://www.ietf.org/rfc/rfc2253.txt

[10] T. Berners-Lee, L. Masinter, M. McCahill, *Uniform Resource Locators*, IETF RFC 1738, http://www.rfc-editor.org/rfc/rfc1738.txt