Use of SAML in the Community Authorization Service

Von Welch, Rachana Ananthakrishnan,
Sam Meder, Laura Pearlman, Frank Siebenlist

{welch, ranantha, meder, franks}@mcs.anl.gov, laura@isi.edu

August 19, 2003

Abstract

This document describes our use of SAML in the upcoming release of our Community Authorization Service. In particular we discuss changes we would like to see to SAML to address issues that have come up both with current and planned development.

1 Introduction

A virtual organization (VO) is a dynamic collection of resources and users unified by a common goal and potentially spanning multiple administrative domains [3]. VOs introduce challenging management and policy issues, resulting from often complex relationships between local site policies and the goals of the VO with respect to access control, resource allocation, and so forth. In particular, authorization solutions are needed that can empower VOs to set policies concerning how resources assigned to the "community" are used—without, however, compromising site policy requirements of the individual resources owners.

The Community Authorization Service (CAS) [5] is a system that we have developed as part of a solution to this problem. CAS allows for a separation of concerns between site policies and VO policies. Specifically, sites can delegate management of a subset of their policy space to the VO. CAS provides a fine-grained mechanism for a VO to manage these delegated policy spaces, allowing it to express and enforce expressive, consistent policies across resources spanning multiple independent policy domains. Both past and present CAS implementations build on the Globus Toolkit® middleware for Grid computing [2], thus allowing for easy integration of CAS with existing Grid deployments.

While our currently released implementation of CAS [1] uses a custom format for policy assertions, the new version currently in development uses SAML [7] to express policy statements. In this document we describe our use of SAML with some issues we have encounters with its use.

2 CAS Overview

CAS functions as a "push-model" authorization service, as shown in Figure 1. In this section we give a brief overview of how CAS is used in normal operation.

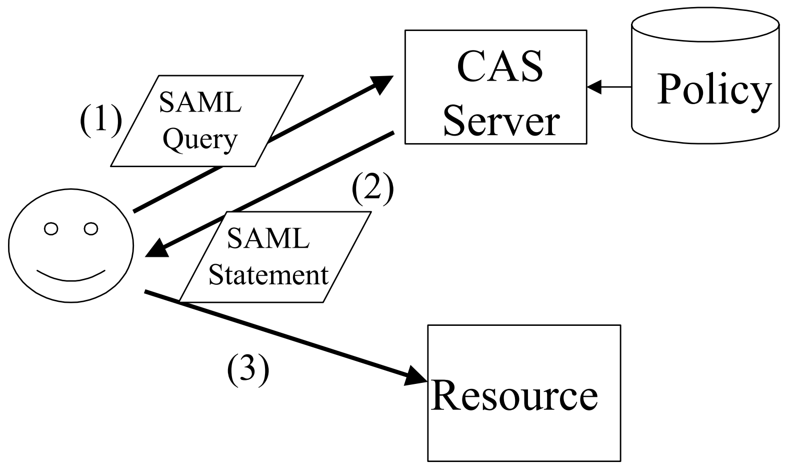

Figure 1: CAS Architecture. Steps are described in the text.

The steps in the Figure are:

1. The client, shown at left, sends a signed SAML AuthorizationDecisionQuery request to the CAS server, at right, indicating which resources they wish to access and which actions they desire to invoke.
2. The CAS server establishes the user's identity. Using the identity it determinesthe rights as established by the VO's policy. It then returns a signed SAML assertion containing an AuthorizationDecisionStatement. This assertion has the user's identity as the Subject and some subset of the user's requested actions
3. The user presents the SAML assertion to a resource along with an authenticated invocation request (the details of which are described in [6]). The resource uses the SAML assertion, subject to local policy regarding how much authority was delegated to the CAS service, to authorize the request. The user may use the assertion to potentially make multiple requests, potentially to multiple resources.

Note that it is common for a client to ask for an assertion containing a complete set of rights they may have on a given resource, set of resources, or even on all resources for which a CAS server has authority. Since the SAML statement returned is typically valid for a number of hours, a assertion with multiple rights allows the user to undertake a number of different actions, which may not be known a priori, without having to recontact the CAS server.

3 CAS Policy Statements

In this section we discuss how policy from the CAS server is expressed using SAML.

3.1 File Access

Current implementations of CAS have been focused on granting access to files, either singularly or as directories trees. For this purpose SAML has worked well - the Resource element contains a URI specifying the file or directory tree and the Action element contains the type of access being granted.

Our only extensions to SAML here were to define mechanisms for expressing Resource and Action elements that specified all resources or actions that the CAS server policy allowed for the user. This allows the user to request a statement containing all their rights on a given resource or even all rights on all resource for which the CAS server is authoritative.

3.2 Grid Service Access

Grid Services [8], as defined by the Global Grid Forum [4], are built off of Web Services, adding functionality necessary for instantiating stateful services, such as definitions of operations for instance creation, lifetime management, and access of stateful "Service Data", used by a Grid Service to expose internal state.

We are currently determining how we would use CAS, and hence SAML, to express policy on Grid Service access. This would entail two specific types of access:

- **Invocation of Grid Service operations**: For all practical purposes this would be identical to invocation of Web Service operations. Our feeling is that the Resource element would contain a URI indicating the service in question and the Action element could contain the operation being invoked. One open issue here is how we would specify policy on arguments in the operation invocation requests. While this is not required for initial use, this is something we will eventually want to support.
- **Access of Service Data:** Each Grid Service can have a unlimited number of Service Data Elements (SDEs) associated with it. SDEs are arranged in a hierarchy and have names unique to the Grid Service with which they are associated. Much like file access, we envision policies expressed both on specific SDEs and on subsets of the SDE space. Accesses will also be in form of read and write access. This raises issues since policy needs to identify both the Grid Service with which the SDE in associated and the SDE set being accessed, which is beyond the ability of the current SAML Action and Resource element. In the Action element we also see specifying policy on the new value of the SDE when an alteration is requested as a potential future enhancement.

4 Summary of SAML Issues

In this section we list the issues we have encountered with the current version of SAML (1.0/1.1) in our implementation:

- Expressing policy in regards to arguments on operations. Currently a SAML Action element is just a URI and string without any attributes.
- Expressing policy in regards to a secondary target associated with the primary target - i.e. a Service Data Element that is part of a Grid Service as described in Section 3.2. This could either be thought of as a nested resource or as an argument to an action. In either case the SAML element doesn't have the ability to express these attributes.

- Expressing wildcard Resources and Actions to either request all rights in a request or express them in an assertion. While these can be encoded in current SAML elements, ideally this would be standardized.
- Ambiguity of AuthorizationDecisionResponse: If a client requests authorization for some set of actions for which only some subset is authorized, should the response contain a single AuthorizationDecisionStatment with a DesicionType of "permit" and the permitted actions or a pair of AuthorizationDecisionStatments, one with the permitted actions and one with the unpermitted.

5 Other Suggestions

In this section we express a couple of suggestions related to the SAML specification that are note related to our current implementation.

- While not currently in development, we are seriously considering the use of XACML [10] as a means of expressing policy in CAS. This integration would be easier if the XACML and SAML standards could be brought into closer alignment through the use of common elements for specifying resources, actions and subjects.
- We are considering supporting the use of our CAS server in a pull mode where it would be directly contacted by resource to render an authorization decision. In the current SAML implementation the assertion in response to the AuthorizationDecisionQuery contains a list, possibly a subset, of allowed action from the query. A resource would then need to process this list to see which of the actions about which it queried are listed. If the resource is interested in an "all or nothing" response, this is highly inefficient. Allowing a resource to request a simple Permit/Deny/Indeterminate response would improve this.

6 References

[1] CAS AlpahR2 Web site, http://www.globus.org/Security/cas/alpha-r2/, September 2002.
[2] Foster, I. and Kesselman, C. Globus: A Toolkit-Based Grid Architecture. Foster, I. and Kesselman, C. eds. *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1999, 259-278.
[3] Foster, I. and Kesselman, C. Computational Grids. Foster, I. and Kesselman, C. eds. The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999, 2-48.
[4] The Global Grid Forum, http://www.ggf.org, 2003.
[5] Pearlman, L., Welch, V., Foster, I., Kesselman, C. and Tuecke, S., A Community Authorization Service for Group Collaboration. IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, 2002.
[6] Pearlman, L., Welch, V., Foster, I., Kesselman, C. and Tuecke, S., The Community Authorization Service: Status and Futures. Computing in High Energy Physics (CHEP03), 2003.
[7] Security Assertion Markup Language (SAML) 1.0 Specification, OASIS, November 2002.

[8] Tuecke, S., et al, Open Grid Services Infrastructure (OGSI) Version 1.0, June 2003. https://forge.gridforum.org/projects/ogsi-wg/document/Final_OGSI_Specification_V1.0/en/1/Final_OGSI_Specification_V1.0.pdf
[9] Welch, V., et al, Use of SAML for OGSA Authorization, 2003. http://www.globus.org/ogsa/security/authz/OGSA-SAML-authorization-profile-june4.pdf
[10] eXtensible Access Control Markup Language (XACML) 1.0 Specification, OASIS, February 2003. http://www.oasis-open.org/committees/xacml/