Case Study: Access Management 2.0 for the Enterprise

Case Study: Access Management 2.0 for the Enterprise

Introduction

Although UMA's primary use cases have centered on individual people (that is, the "users" who manage access to their own online resources), the UMA notion of **authorization as a service** also has relevance to modern enterprises that must secure APIs and other web resources in a developer-friendly way.

Problem Scenario

Where once web access management (WAM) and single sign-on (SSO) were sufficient for many purposes in the enterprise context, a new requirement has surfaced: managing access to an enterprise's web APIs, not just web apps. Today's systems for managing this type of access have a number of challenges.

Using current WAM solutions to provide API security can be **unfriendly to developers**, **complex**, **expensive**, and likely **proprietary**. **Mobile clients struggle** to deal with XML-based and SOAP-based security mechanisms. **Enterprise IT struggles to deploy agents or proxies.**

Since it's currently overly complex to centralize access authorization, we find **too much authorization code** in applications, which slows service delivery by forcing developers to redevelop authorization logic, as well as hindering effective auditing and policy administration.

WAM solutions are **not fully agnostic as to authentication method**. WAM solutions previously could make simplifying assumptions about how users authenticate (typically username and password into a web app). With mobile applications, game consoles, and other device types, and with strong authentication needs increasing, old assumptions are no longer viable.

APIs by their nature are subdivisions of functionality exposed at a single domain, and would map well to arbitrarily fine-grained policy, for example, at the method or even parameter level. However, outside the use of XACML, **authorization policy granularity is coarse** in traditional solutions: group filtering or URL regular expression matching at most. Further, it is often **impractical to act according to policies from multiple authoritative sources**.

Proposed Improvements

UMA makes the following solutions possible.

As a profile of OAuth 2.0 (IETF RFCs 6749 and 6750) that is complementary to OpenID Connect, UMA defines **RESTful**, **JSON-based**, **standardized** flow s and constructs for coordinating the protection of any API or web resource in a way that will be familiar to any developer already acquainted with OAuth. **M obile developers accept** technologies that use HTTP and JSON at their core.

UMA's notion of machine-readable resource set and scope descriptions creates an access control mechanism that enables **control over specific API scopes** (customizable buckets of API functionality), not just domains. With UMA, client app developers can handle authorization tasks by calling simple REST/JSON endpoints; administrators **don't have to deploy a web server agent** or reverse proxy to enable centralization.

UMA defines interfaces between authorization servers and resource servers that, by default, **enable centralized policy decision-making** for improved service delivery, auditing, policy administration, and accountability, even in a very loosely coupled "public API" environment. Custom profiles enable flexibility to move the decision-making line outward to distributed applications, to account for local preferences in API ecosystems. UMA does not standardize a policy expression language, enabling **flexibility in policy expression and evaluation** through XACML, other declarative policy languages, or procedural code as warranted by conditions. It also has a **fluid way to handle federated authorization policy**.

UMA inherits **authentication agnosticism** from OAuth. It concentrates on authorization, not on authentication. It has been profiled to work with OpenID Connect to gather identity claims from whoever is attempting access, and enables true **claims-based authorization** (with simple group- or role-based policies a natural subset).

Solution Scenario

An organization – say, BusinessCo -- publishes numerous APIs. BusinessCo authorizes which people, clients, and networks can access a subset of the APIs. BusinessCo, as a subsidiary of ParentCo, relies on ParentCo to provide some of the required authorization policies.

When BusinessCo publishes and UMA-protects an API, its resource server acts as a policy enforcement point (PEP). The RS relies on an UMA authorization server to as a policy decision point (PDP) for it. The PEP can query multiple authorization servers to ensure that all required authorizations have been granted, for example the policies of both BusinessCo and ParentCo. In UMA trust model terminology, this scenario is in the category **non-person entity (NPE) sharing** because the resource owner is a corporate entity that also operates its own authorization server.

Enterprise resource users such as employees and partners operate client applications, such as web and mobile apps, in order to request access. How the respective PDPs make authorization decisions about these users and clients is up to the implementation. Each authorization server may itself be a policy information point (PIP) at which policies reside, or it may be a client of one or more PIP services. To understand the nature of the requesting party, a PDP might be sent SAML- or OpenID Connect-based tokens by the requesting party's client, or may itself need to call out to an identity provider.

The PDP service would expose policy administration point (PAP) functions to IT administrators in some fashion. BusinessCo would need the capability to express their access policies; for example, BusinessCo or ParentCo might interface a standard enterprise entitlements management system that expresses polices in XACML or some other standard language.

Solution Flow

This scenario uses ordinary UMA flows, noting that it is a human "Authorizing Party Agent", not BusinessCo, that sets policy and possibly performs any policy-dictated manual intervention to enable access requests to go through. The company Gluu has produced a swimlane diagram to illustrate.

Solution Demo

Gluu gave a demo of an enterprise UMA scenario at an UMA WG meeting on Jan 10, 2013; see the meeting notes. Also see their series of demo videos on YouTube and information on their open-source implementation.