Controlling Data Usage with User-Managed Access (UMA)

Eve Maler
W3C Privacy and Data Usage Control Workshop
October 2010

Overview

The User-Managed Access (UMA)[1] protocol is designed to give a web user a unified control point for authorizing who and what can get access to their online personal data (such as identity attributes), content (such as photos), and services (such as viewing and creating status updates), no matter where all those things live on the web. Further, UMA allows a user to make demands of the requesting side in order to test their suitability for receiving authorization. These demands can include requests for information (such as "Who are you?" or "Are you over 18?") and promises (such as "Do you agree to these non-disclosure terms?" or "Can you confirm that your privacy and data portability policies match my requirements?").

The implications of these demands quickly go beyond cryptography and web protocols and into the realm of agreements and liability. UMA values end-user convenience, development simplicity, and web-wide adoption, and therefore it eschews such techniques as DRM. Instead, it puts a premium on user visibility into and control over access criteria and the authorization lifecycle. UMA also seeks at least a minimum level of enforceability of authorization agreements, in order to make the act of granting resource access truly informed, uncoerced, and meaningful. Granting access to data is then no longer a matter of mere passive consent to terms of use. Rather, it becomes a valuable *offer* of access on user-specified terms, more fully empowering ordinary web users to act as peers in a network that enables selective sharing.

This position paper presents UMA's overall goals and architecture for user control of access to web-based data and services, reviews its proposed solutions for controlling data usage, and lists major areas still to be resolved.

Introduction to UMA

User-Managed Access (UMA) is a novel access management solution that consists of an architecture and access control delegation protocol based on OAuth 2.0.[2] It is being developed at the Kantara Initiative for ultimate contribution to the IETF. The UMA Work Group is free for anyone to join. Its draft specifications are free for anyone to implement, and several implementation efforts are under way.

Following are some key UMA requirements.

Dedicated access relationship service: Access control should be externalized from web applications and provided as a dedicated online service (think "digital footprint dashboard"). Such a service should allow a user to control data-sharing and service-access relationships between online services hosting and accessing data. All these services should be able to reside in distinct domains and establish relationships between themselves in a dynamic way. For the access relationship service to be usable across multiple web applications, it should not be required to understand the representations of resources it is charged with protecting and its functionality should be applicable to arbitrary web resources.

[1] See Kantara User-Managed Access Work Group, http://kantarainitiative.org/confluence/display/uma/Home.

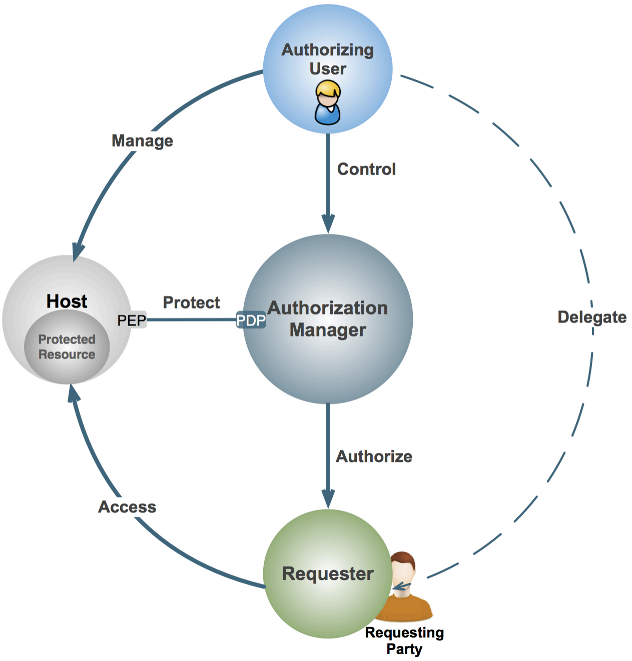
[2] See IETF Open Authentication Protocol (oauth), http://datatracker.ietf.org/wg/oauth/charter/.

User-driven policies: An individual should be able to configure their own policies required for the access relationship service to make access control decisions. As such, a user should be able to apply the same policy across distributed web resources, using a consistent interaction experience.

Support for claims-based access control: In order to fit well into the highly dynamic and open web environment, access control should not be based solely on preliminary identification and authentication of requesters; servers may not know possible identities of clients accessing data in advance. Policies should allow a user to define properties that clients must possess before authorization can be granted. Moreover, a user should be able to impose contract terms that govern access rights, as well as data storage, further usage, and further sharing on the part of requesting services.

User management of access control: An individual should be able to modify the conditions of access and terminate relationships easily. It should also be possible to audit and monitor various aspects of such relationships at any time. If they wish, the user should not have to be directly involved in interactions between services accessing data and services hosting data. Rather, it should be possible to guide such interactions by applying the user's defined policies alone.

To satisfy these and other documented design principles and requirements, UMA has the architecture depicted below.



An **authorizing user** arranges to externalize resource protection from their chosen set of resource **hosts** (policy enforcement points) to an **authorization manager** or AM (policy decision and administration point), configuring the latter with policies that control how it makes decisions about delegation of access authorization when a **requester** attempts to access a protected resource. The **requesting party** "behind" the service running the requester endpoint may be the same person as the authorizing user, a different person, or a legal entity (such as a corporation).

For example, web user Alice (authorizing user) might authorize an online service (requester) to gain one-time or ongoing access to a set of personal data including her home address stored at a personal data service (host), having already instructed the host to check with her authorization decision-making service (AM) whenever requesters come calling. The requesting party might be an e-commerce company whose site is acting on behalf of Alice herself to assist her in arranging for shipping a purchased item, or it might be her friend Bob who is using an online address book service to collect addresses, or it might be a survey company that uses a web app to find and compile broad population demographics.

Policies and Claims

A policy may, in the simplest case, allow an AM to grant access authorization to a requester without needing to gather further information from it; for example, Alice could set a policy for some resource that allows automatic access by any requester but limits their access window to one day, or a policy that tells the AM to gather her real-time consent by text message every time access is attempted. UMA treats the composition, expression, and evaluation of policies generally as being out of band, since policy provisioning and decision-making take place solely through the AM. Thus it admits a variety of solutions, both standardized and proprietary.

In more complex cases, a policy could require the AM to gather more information about the requesting party before making a policy decision, and this is where UMA introduces the notion of **claims**. A claim in the federated-identity sense is defined as "an assertion made by a claimant of the value or values of one or more identity attributes of a digital subject, typically an assertion which is disputed or in doubt."[3] UMA defines a sub-protocol whereby an AM can ask a requester to convey claims about its requesting party. In this case, the requesting party is the digital subject. It may also be the (self-asserting) claimant, or a third party trusted by the AM might have issued the claim instead. UMA does not dictate a format for communicating claims between requesters and AMs, but it does provide a candidate simple solution called Claims 2.0.[4]

UMA's claims communication protocol can readily handle self-asserted claims well-suited for lightweight access control and enforcement, such as asking Bob to say on his own recognizance whether he is over 18 years old, or asking the survey company to agree to a privacy and data usage policy that forbids reselling data to third parties. The goal is to achieve a legal enforcement bar at least as high as Alice clicking "I Agree" when presented with a site's terms of service.

However, the UMA group has collected a number of use cases that require higher security and carry higher liability consequences than this. It is clear that today's Web 2.0-style access control mechanisms, in which a user types email addresses into an access control list and the app emails a "secret link" or "private URL" to those people, won't suffice for controlling access to healthcare records, tax data, or perhaps even university transcripts. A practical system is needed for authenticating the source of third-party-asserted claims, evaluating their quality, ensuring that they apply to the requesting party in question, and transferring claims in a privacy-sensitive way only for policy decision-making purposes – all in a way that reaches web-wide scalability.

The UMA group summarizes this class of problems using the *bob-at-gmail* use case: How can Alice type "bob@gmail.com" into an ACL at her AM to govern access to her travel calendar, such that whatever requester app Bob is using can lead him down a path that ends up with Google generating a trustworthy claim that he's the guy who matches the policy? Similarly, when the survey company is the requesting party, how can Alice ensure that the party agreeing to her privacy and data usage policies is really representing that company, so she can successfully sue the company for breach of contract if they sell her data?

The group is investigating a solution, called **trusted claims**, whereby the claims themselves can be UMA-protected. The requester app shares with the AM only the location of a claims catalog, and Alice's authorization management service becomes a requester app for those claims as part of its AM duties. In the general case, the solution appears to rely on the existence of PKI-enabled trust frameworks that enable dynamic high-quality trust between certain endpoints and

[3] See Identity Commons lexicon, http://wiki.idcommons.net/Claim.

[4] See UMA Working Drafts, http://kantarainitiative.org/confluence/display/uma/Working+Drafts.

well-known semantics for identity and attribute assurance. But use cases that are able to make simplifying assumptions could be solved in lighter-weight fashion.

Scope

Discussing “data usage policies” typically carries an assumption that online data-sharing is about retrieving personally identifiable data or user-generated content and then exercising an authorized right – in some technical and/or legal context – to do something with that data independently.

Reality, however, is more complicated. Access to data and content is often a two-way street. For example, services that use OAuth-protected location APIs seek to *write* location data to those endpoints as often as *reading* it from them. Thus, any discussion of data usage should account for the extent of data manipulation involved in direct interactions between requesting and responding services.

OAuth refers to a labeled extent of access at a resource server[5] as a **scope**, and users are presented with the opportunity to consent to the particular scopes being requested by a client (such as “writing your location” and “reading your location at a city level”) when setting up an authorized connection. This model currently involves passive user consent, similar to accepting a site’s TOS, rather than an opportunity for the user to make an offer of access on more favorable terms.

The UMA group is working to incorporate a flow of scope information throughout the process of delegating authorized access that supports the user's ability to map policies to scopes and the host’s ability to map scopes to access instructions.

Discovery of Data Offerings

As can be seen in real-life examples of OAuth-protected access to APIs (as well as login-time sharing of identity attributes during single sign-on), the user is only in a position to consent – or withhold consent – to offered terms and scopes.

There are several structural barriers in the way of changing this situation. First, most web APIs are proprietary, requiring a substantial amount of integration between the requester and host for both functionality and security reasons well before the user comes along; this requires the services to offer Alice the connection opportunity, rather than waiting for her to suggest it. Second, requesting services naturally have business reasons for being proactive and requesting as wide a scope of access as possible.

UMA is attempting to build features that begin to change this equation on the technical end, for example, allowing for more dynamic introduction and registration of various parties, and supporting a way for a host to tell an AM what scopes it makes available, so the user can map scopes to policies at the AM. However, until the day when a user can, out of the blue, advertise the potential availability of access, or directly share knowledge about a resource, to requesters – in short, until requesters can dynamically **discover** that data access is even an option – there are strong forces keeping the offer-acceptance cycle stuck where it is.

UMA has chosen not to dictate mechanisms for this kind of discovery. We envision a variety of mechanisms to come into use, with the most basic being a publication/subscription model involving the direct sharing of an UMA-protected “feed” URL, much as calendars, blog entries, and photostreams are shared today.

[5] UMA’s host is an enhanced version of an OAuth resource server; its AM an enhanced version of an OAuth authorization server; and its requester an enhanced version of an OAuth client.

Conclusion

This paper has reviewed:

- The goals of the UMA effort in building improved data access and data usage control mechanisms;
- UMA's approach to policies governing access, claims for assessing requester suitability for access, scopes of access, and building dynamic trust between parties, and;
- the main business and web-ecosystem factors that lead to an over-reliance on user consent in the service connection ecosystem.

UMA is making a run at the problem of privacy-enhanced selective sharing on several fronts, but is counting on synergies with many other efforts (ongoing, planned, and speculative) to take full advantage of the opportunities presented, including, for example:

- Trust frameworks[6] in concert with identity assurance[7] and attribute assurance schemes
- Standardized policy expression and evaluation frameworks[8]
- Standardized privacy policies, data portability policies, and information sharing agreements[9]
- Standardization around web APIs and the scopes that apply to them
- Dynamic registration of OAuth clients at dynamically discovered authorization servers[10]
- Customer-centric "fourth-party" brokering services[11]

Even without all the pieces in place, however, having some basic selective-sharing flows available for experimentation and deployment could encourage the organic development of other pieces. In particular, the combination of a generalized mechanism for scoped, terms-conditioned resource protection with pub/sub has the potential to be powerful all on its own.

The UMA group looks forward to the outcome of discussions at the W3C Privacy and Data Usage Control Workshop and to future opportunities for collaboration on these topics.[12]

[6] Such as Kantara IAF, http://kantarainitiative.org/confluence/display/idassurance/Home, and Open Identity Exchange, http://openidentityexchange.org/.

[7] Such as NIST 800-63, http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf.

[8] Such as XACML, http://www.oasis-open.org/committees/xacml/.

[9] Such as work being done at the Data Portability Project, http://portabilitypolicy.org, and the Kantara Information Sharing Work Group, http://kantarainitiative.org/confluence/display/infosharing/Home.

[10] Participants in the UMA group have proposed an OAuth Dynamic Client Registration Protocol, http://tools.ietf.org/html/draft-oauth-dyn-reg-v1-00, as a starting point.

[11] See VRM and the Four Party System, http://blogs.law.harvard.edu/vrm/2009/04/12/vrm-and-the-four-party-system/.

[12] The author is grateful to Maciej Machulak and Domenico Catalano of the UMA Work Group for their many contributions to this paper.