User-Managed Access to Web Resources

Maciej P. Machulak
Newcastle University
Newcastle upon Tyne, UK
m.p.machulak@ncl.ac.uk

Eve L. Maler
PayPal, Inc.
San Jose, CA, USA
eve.maler@paypal.com

Domenico Catalano
Oracle Corp.
Rome, Italy
domenico.catalano@oracle.com

Aad van Moorsel
Newcastle University
Newcastle upon Tyne, UK
aad.vanmoorsel@ncl.ac.uk

ABSTRACT

Web 2.0 technologies have made it possible to migrate traditional desktop applications to the Web, resulting in a rich and dynamic user experience and in expanded functionality. Individuals can create and manage their content online, and they are not only consumers of Web services, but also active participants on the Web platform. As a result, potentially large amounts of personal, sensitive, and valuable data is put online, spread across various Web services. Users sometimes share this data with other users and services on the Web, but are also concerned about maintaining privacy and sharing their data securely.

Currently, users must use diverse access control solutions available for each Web service to secure data and control its dissemination. When such mechanisms are used on a daily basis, they add considerable overhead, especially since these mechanisms often lack sophistication with respect to functionality as well as user interfaces. To alleviate this problem, we discuss a novel approach to access management for Web resources that includes a user as a core part of its model. The proposal puts the user in charge of assigning access rights to resources that may be hosted at various Web applications. It facilitates the ability of users to share data more selectively using a centralized authorization manager which makes access decisions based on user instructions.

Categories and Subject Descriptors

D.4.6 [**Software**]: Operating Systems - Security and Protection—*Access controls*

General Terms

Design, Security

Keywords

access control, authorization, security, Web 2.0

DIM'10, October 8, 2010, Chicago, Illinois, USA.

1. INTRODUCTION

Web 2.0 has become a platform supporting all kinds of interactions, be it business processes or collaboration between users. This has resulted in many of these interactions and their associated data being shifted from the real world to this environment [35, 36]. It has also influenced the way people engage with one another, form communities, and share information. A key trend in Web 2.0 is the inclusion of the user as a core part of any model [20]. It is the user who creates data and plays a role as a content publisher. It is also the user who disseminates this data and who shares it with other users and services on the Web.

Sharing data in the Web 2.0 environment poses various security and privacy issues which are commonly addressed using diverse access control solutions. Such solutions, however, seem to lack sophistication, simplicity and usability since they are a side issue for typical Web 2.0 applications.

Access control mechanisms are often tightly bound to the application and have limited flexibility in terms of their configuration or adaptation to a particular user's security requirements [29, 32]. These mechanisms are configured using diverse and incompatible policy interfaces at every Web application. This prevents a user from easily coordinating any changes in access settings between online services. Moreover, a user lacks a global view of all their sharing preferences, patterns and data recipients on the Web.

In order to benefit from the increasing number of services accessible over the Web, a user is forced to share data using provided access control mechanisms. As noted by the Vendor Relationship Management (VRM) movement [5], for example, a user may need to "hand over" information that can be sensitive, valuable, and personal and often has to do it in time-consuming and imprecise ways. By providing such information to requesting Web services an individual is paying a price in both privacy and convenience. A user then has limited ability to control access to such information once it is submitted, and in any case must surrender it under terms favorable only to its recipients.

Following the highly collaborative and user-centric Web 2.0 paradigm, there is a clear need for new approaches to access management which would allow a user to play a pivotal role in their model. Such approaches would allow a user to be in full control over access to their data irrespective of the location of this data. Moreover, these approaches would allow a user to apply the necessary security and privacy con-

trols while retaining all the benefits of social interactions and data sharing that the Web 2.0 environment offers.

In this paper we present a new approach to access control for Web resources based on the User-Managed Access (UMA) protocol[1]. The UMA proposal provides a method for users to control third-party access to their protected resources, residing on any number of host sites, through a centralized authorization manager that makes access decisions based on user instructions. This gives users the required flexibility in sharing their data and supports potential requesters with accessing such data.

The UMA solution is fundamentally different from existing proposals for access control that exist both within closed and open environments. As opposed to classic access management systems such as Kerberos, UMA allows all interacting entities to establish relationships dynamically and it does not rely on policy administrators but allows a user to play a pivotal role and be in full control over access rights to their Web resources. Moreover, unlike protocols such as OAuth, the UMA solution allows to aggregate the authorization for many point-to-point sharing relationships. We provide a more detailed comparison of User-Managed Access and existing proposals throughout the paper.

The remainder of the paper is organized as follows. We discuss shortcomings of existing authorization solutions in Section 2. We provide a requirements analysis for a user-managed access control solution in Section 3. In Section 4, we explain the User-Managed Access approach to authorization for Web resources which addresses discussed shortcomings and meets presented requirements. We evaluate this approach against these requirements in Section 5. We present a prototype implementation of the discussed access control solution in Section 6. We discuss progress and future work in Section 7. We examine related work in Section 8 and we conclude in Section 9.

2. PROBLEM STATEMENT

Users are the core part of the Web 2.0 model. They create and publish data using various Web applications and may then disseminate this data and share it with other users and services on the Web. Despite the fact that users now play the pivotal role in various online interactions, they are not sufficiently empowered to be in control over access to their Web resources. As discussed by Ann Cavoukian in [19], users should be able to "quickly determine what information is shared with what parties and for what purposes, and further, to control how information is shared. The user should be capable of determining the trustworthiness of these parties, how the shared information will be handled, and what the consequences of sharing this information are."

The majority of popular Web 2.0 applications appear not to provide access control at the level that would have the above mentioned functionality. Some of these applications implement only very simple sharing options. Other applications, on the other hand, implement more sophisticated access control solutions which seem to be particularly well-suited to closed environments (typically large enterprises) and do not provide direct benefits in terms of sharing functionality or usability to average Web users. We therefore discuss these authorization solutions and analyze their shortcomings in the context of the open Web environment.

Isolated authorization models implemented by today's Web applications significantly limit their configuration or adaptation to particular user's security requirements. Data-sharing and service-access relationships have to be established at each Web application separately by a user. Existing proposals for delegated authorization, well-supported by various working implementations such as the one in OpenSSO [3], do not fully alleviate this problem as these proposals are mainly targeted at internally deployed Web systems with well-defined points of control. These proposals are ill-fitted for the open Web environment and they do not give a Web user the required flexibility in configuring access rights to a user's set of resources.

Additionally, various access control solutions deployed by different Web applications force users to use their proprietary policy management tools which results in an inconsistent user experience (UX). Underlying policies at these applications often differ and are incompatible with each other, thus they cannot be easily reused for distributed resources hosted at various Web applications that may reside in different Web domains.

Another important shortcoming of access control mechanisms used by existing applications in the open Web environment is their lack of support for claim-based policies such as those discussed in [15]. Currently, access control policies used by these applications are based mostly on uniquely authenticated client identities and not on generic properties that must be met by requesters. For example, photo and calendar sharing is typically achieved through specifying email addresses in access control lists, and OAuth connections are forged through client and resource-owner authentication. Systems such as Shibboleth allow to define policies based on generic attributes but it's the policy administrator and not the end user that is supported with this task.

Identity-based policies may be acceptable in closed environments but this approach significantly limits possible sharing scenarios that may be conducted in the open Web where applications hosting resources may not know data requesters in advance. For example, a user cannot define a policy stating that anyone who can prove themselves to be a citizen of the United Kingdom can have access to a resource or convey their agreement that accessed data will not be distributed further after authorization is granted to this requester.

For another example, one of several sharing scenarios discussed in [8] and [4] relates to electronic health data. Ideally a Web user who is a medical patient could control the sharing of their health data among various service providers, such as primary-care physicians, specialists, user-centric health vaults, and other data sources and destinations. When a new health care provider system would need to be used, then such a provider could seek access to a user's health data dynamically and could be granted access based on recognized medical credentials, rather than having to prove unique identities that could not have been known in advance.

Moreover, with access control being distributed across multiple Web applications, a user is unable to easily modify the conditions of access between various Web services and terminate relationships between these services entirely. This task requires traversing applications and configuring access control appropriately. The same applies to auditing and monitoring various aspects of such relationships. With

[1]The protocol is being developed in initial form by the Kantara User-Managed Access Work Group (UMA WG). Authors Machulak, Maler, and Catalano hold leadership positions in this work group.

such limited view of the applied security over distributed Web resources, a user may fail to provide the required level of protection over possibly sensitive information.

3. REQUIREMENTS ANALYSIS

We discuss requirements for a new approach to access control that would fit precisely into the Web 2.0 environment. Some of these requirements have been initially presented in [43]. They address the shortcomings of existing access management systems such as those based on XACML [9], that we perceive as inflexible, insufficiently user-managed and ill-fitted for the user-driven and open Web environment

Dedicated access relationship service.

Access control should be delegated from Web applications and provided as a dedicated online service. Such service should be managed by 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 Web domains and they should be able to 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 which can be identified with URLs.

User-driven policies.

The solution should allow an individual to set their own policies required to make access control decisions with the use of a user's preferred policy management tool. As such, a user should be able to apply a single policy across a set of distributed Web resources and should be provided with a consistent user experience throughout the entire process of composing such a policy and linking this policy with a set of resources. For example, it should be possible to define a policy that offers access only to a person's family members, and attach that policy to photos residing in one Web application, calendars residing in another, and social status updates in another. It should also allow for policies that can take effect either with the user's real-time consent or silently, at the user's direction.

Support for claims.

Access control should not be based solely on preliminary identification and authentication of requesters in order to fit well into the highly dynamic and open Web environment where 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, as well as data storage, further usage, and further sharing on the part of requesting services. The solution should support communication of such properties between services protecting data and services accessing such data.

User management of access control.

Access relationships between services, expressed in the form of policies and terms as discussed, should be user-managed allowing an individual to modify the conditions of access and terminate relationships easily. The solution

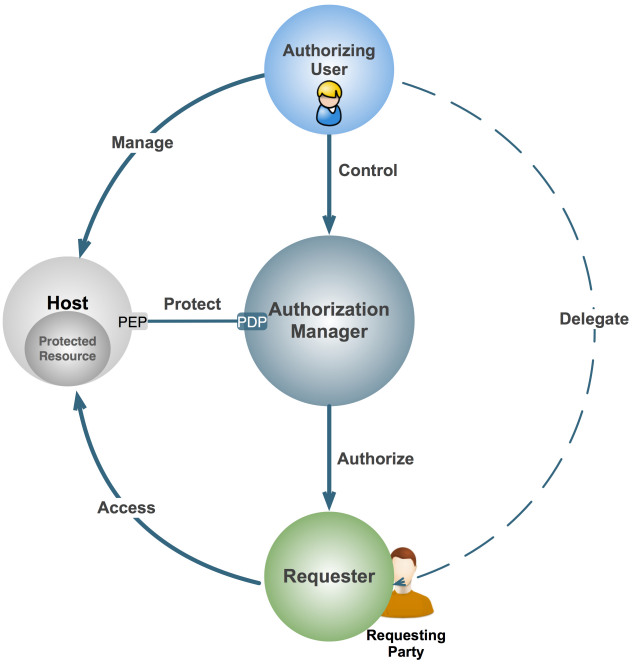

Figure 1: Interactions between entities involved in the User-Managed Access (UMA) protocol.

should also allow an individual to audit and monitor various aspects of such relationships at any time with minimum effort. It should not require a user to be directly involved in interactions between services accessing data and services hosting data. Rather, it should control such interactions in a fashion guided by a user's defined policies as managed and audited through a "digital footprint dashboard" UX.

Apart from the afore mentioned requirements, other ones include design simplicity that promotes understanding, implementation, and Internet-scale deployment; the use of REST principles to support an orientation towards protecting Web resources; independence from any one user identification system; and OAuth as a substrate for trust establishment issues. Relying on OAuth would ease adoption of the new solution among the increasing number of applications on the Web that already implement this protocol. A full list of requirements and design principles can be found in [43].

4. APPROACH

User-Managed Access (UMA) to Web resources is a novel access management solution that consists of an architecture and a new access control delegation protocol. UMA provides a method for users to control third-party application access to their protected resources, residing on any number of host sites, through a centralized authorization manager that makes access decisions based on user instructions. The protocol is designed to address shortcomings of existing authorization solutions presented in Section 2 and to satisfy requirements discussed in Section 3 and formulated in [43].

The UMA proposal consists of a dedicated service for authorizing data sharing and service access. The user is capable of imposing demands on any Web application that wishes to access a user's data. Moreover, the user is able

to monitor, change, and stop access relationships between online services from one location. With a specialized component being in charge of relationships, the user does not have to manually provision copies of data to requesting services, nor attempt to redefine essentially the same policies at multiple new hosts. Instead, the user may point requesters to authoritative sources from which they can request such data directly in a secure and efficient way, and apply policies across multiple hosts and resources.

UMA has been researched by the User-Managed Access Work Group (UMA WG) [44]. The protocol has been first proposed in [23] and [22] but has since undergone significant modifications regarding the adoption of the OAuth V2.0 protocol [14]. We discuss how OAuth V2.0 fits into our model in the paper and we show OAuth-based interactions between entities of the UMA architecture in Fig. 2.

4.1 Architecture

UMA is based on four main entities: Authorizing User, Authorization Manager, Host, and Requester (Fig. 1). We provide an overview of each of these entities in subsequent sections. For a more detailed explanation of these terms we refer the reader to [7].

4.1.1 Authorizing User

An Authorizing User delegates access control from their chosen Hosts to an Authorization Manager. Such a user is also responsible for configuring policies at an Authorization Manager which makes access decisions when a Requester attempts to access a Protected Resource at a Host. A user, therefore, serves as their own policy administrator.

4.1.2 Authorization Manager

An Authorization Manager (AM) acts on behalf of an Authorizing User. It evaluates access requests made by a Requester against applicable policies, issuing Access Tokens necessary to make authorized access requests to Protected Resources at a Host. An Authorization Manager may also evaluate such tokens on demand in case a Host chooses not to evaluate them locally. Therefore, an AM acts as a Policy Administration Point (PAP) and a Policy Decision Point (PDP), as defined in [45], and plays the conceptual role of a Security Token Service as defined in [21].

4.1.3 Host

A Host is a Web application that is used by an Authorizing User to store and manage Protected Resources and to share access to these resources with specific Requesters. A Host delegates access control to an Authorization Manager following configuration by an Authorizing User. A Host is then concerned with enforcing access control decisions issued by an Authorization Manager. Therefore, a host acts as a Policy Enforcement Point (PEP).

4.1.4 Requester

A Requester is an application that interacts with a Host in order to get access to a Protected Resource, which can be accomplished after it interacts with an AM to obtain an Access Token. A Requester is controlled by a person or a company (Requesting Party) that uses such an application to seek protected resource access on their own behalf.

For example, a Web user (Authorizing User) can arrange to authorize an online service to gain one-time or ongoing access to a set of personal data including his home address stored at a personal data service (Host). A user can achieve that by instructing the host to check with his authorization decision-making service (Authorization Manager). The requesting party might be an e-commerce company whose site is acting on behalf of the user himself to assist him in arranging for shipping a purchased item, or it might be his friend who is using an online address book service to collect addresses, or it might be a survey company that uses an online service to compile population demographics.

And in the electronic health record scenario discussed earlier, a medical patient might serve as an authorizing user who chooses a market-leading authorization manager application to coordinate sharing among a variety of applications – physician, lab, health vault – that each serve as both a host and a requesting party seeking access to information generated by other medical hosts.

We discuss an extensive set of use cases with various settings of the proposed entities of the UMA solution in [8].

4.2 Delegation Protocol

The User-Managed Access protocol describes interactions between all of the previously defined entities. It consists of the following steps, some of which are currently defined as extensions and profiles of the OAuth V2.0 protocol [14]: (1) User introduces host to AM, (2) Requester gets access token from AM, (3) Requester wields access token at host to gain access (Fig. 2). We also describe how a user may define access control policies at an AM for resources stored at a host. The protocol, however, does not impose any constraints on how the policy setting step should be performed.

4.2.1 User introduces host to AM

In this step a user establishes a trust relationship between a host and an authorization manager (Fig. 3). This can be initiated by providing the location of a user's preferred AM to a host. For example, an authorizing user may provide the URL of their AM to a host Web application by typing it into a text field on a Web page or transmitting through an information card.

When a host is provisioned with the location of the AM then it uses the host-meta discovery mechanism [12] to obtain a metadata document from the AM. Such a document defines the location of various endpoints exposed by this AM. These endpoints provide functionality for hosts and requesters. Host-related endpoints include the registration URL, the user authorization URL, the access token URL, the resource registration URL, and the token validation URL. The metadata document also defines access token formats and claim formats that this AM generates. We refer the reader to [7] for examples of such a metadata document and further explanation of AM's endpoints.

The host registration URL allows a host to obtain credentials which are later required to bootstrap communication between this host and AM. Because these credentials are used by a host only and are not user specific, a host needs to perform this step only once per authorization manager. As such, the UMA protocol supports dynamic introductions between hosts and authorization managers. The user authorization URL is used by a host leveraging the OAuth V2.0 Web Server Flow [14] to initiate the process of acquiring authorization to use a particular AM. This authorization is eventually granted when a host uses the AM's access token

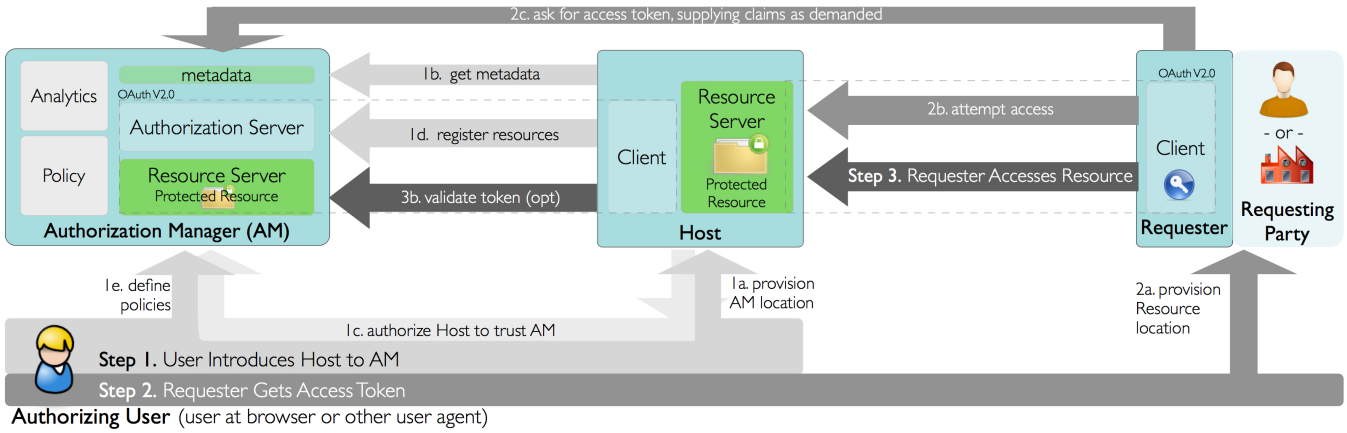

Figure 2: High-level overview of the User-Managed Access protocol.

URL to obtain the necessary access token to use this AM's functionality. From the host's perspective, such functionality includes registering resources that should be protected with this AM (resource registration URL) and validating tokens received from requesters (token validation URL).

A host may delegate access control to the AM for all resources of a particular user, for groups of resources or for individual resources only. A user on a host can select resources and groups of resources that are meant to be policy-protected at the AM. A host then registers these resources using the AM's resource registration URL. Such registration is currently under investigation by the UMA WG and is out of the scope of this paper. The protocol itself supports various granularity levels of access control delegation and does not restrict a user from using multiple AMs to protect different resources on the same host.

User defines access control policies at AM

The UMA protocol does not impose any constraints on how access control policies are composed by a user or how a policy is linked with a resource, since policy provisioning and decision-making take place solely within the AM and this information is not conveyed to other UMA entities. The AM, however, is meant to provide a consistent user experience by allowing users to manage their policies for distributed Web resources with a single user interface. Different AMs may provide different management tools with different user interfaces and underlying policy types and policy engines. We envisage that support for specific tools may dictate the choice of an authorization manager by a user.

UMA purposely does not constrain the policy composition process in order to support a variety of data sharing scenarios on the Web. A user may compose policies identifying suitable subjects and their access rights to a user's resources. More importantly, the UMA protocol supports the policy-driven ability of an AM to demand "claims" from a requester before authorization is granted. A policy may also require an authorizing user's consent to be collected in real time. We discuss how UMA uses claims to support users in composing flexible access control policies that fit well to the open Web environment further in this section. Other examples of policies are discussed in more detail in [8].

As far as linking a policy to a resource is concerned, a user may perform this in a variety of ways. We envisage that a

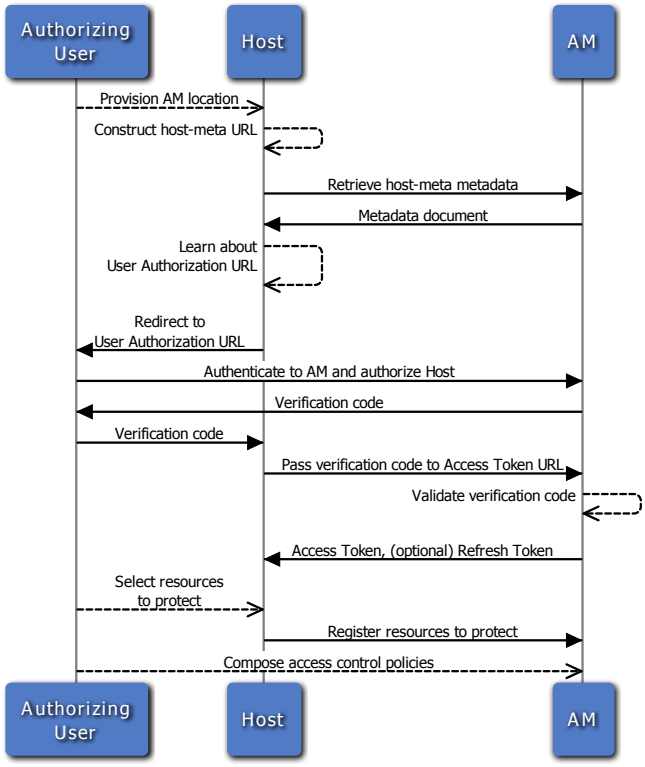

Figure 3: UMA Step 1: User introduces Host to AM.

host may offer a typical security-related user interface (e.g. a "Protect" link). When a user clicks on such a link then they are redirected to the configured authorization manager to associate a resource with an access control policy. Similarly, a user may decide to log in to an AM and manually link a policy with a resource using provided management tools. Our prototype software, as discussed in Section 6, supports both types of interaction.

At the end of this step, a set of resources is successfully associated with one or more access control policies defined

by a user at the AM. The authorization manager evaluates access requests issued by a requester against these policies.

4.2.2 Requester gets access token from AM

In order for the requester to be able to access a protected resource on a host the access request needs to be accompanied by an access token. If such a token is missing in the request then a host responds with a "HTTP 401 Unauthorized" response [25]. Such response contains information about the AM that protects this resource which the host acquired earlier from the AM's hostmeta document.

The UMA protocol uses the "WWW-Authenticate: Token" header, as defined by the OAuth V2.0 specification [14] to specify requester-related endpoints of the authorization manager. These endpoints include the requester registration URL, user authorization URL and the access token URL.

The requester may choose not to issue any unauthorized access requests to a host and may directly approach the authorization manager to acquire the token if a location of AM's endpoints is known in advance to the requester.

Once the requester learns about the location of the AM's endpoints, it registers with this AM (registration URL) and then adheres to the OAuth V2.0 protocol, with a few key UMA extensions, to obtain an access token for a protected resource at a host (user authorization URL and access token URL). UMA adopts one of the existing flows, as defined in [14], for this step of the protocol. If an authorizing user acts as a requesting party then it adopts the user delegation flow. However, if a requesting party is a different person or a company acting on its own behalf then it uses one of the autonomous client flows with the added semantic that the operator of the client is different from the authorizing user.

In both cases a requester asks for an access token by providing information to the AM's access token URL about the required type of access (method and scope) being sought at a host. A requester may provide information concerning multiple desired methods and scopes at a host. Such information is a URL-encoded JSON object. We refer the reader to [7] for an example of such an object.

The AM evaluates an access request based on the provided information. The protocol does not define how the AM performs such an evaluation. The authorization manager may, for example, use simple rules or a more sophisticated policy engine to evaluate access requests against applicable policies. We discuss a proposal for evaluating access requests against policies that contain claims in Section 6.

Once the AM decides whether the access request is valid or not, it may respond to a requester in one of three ways - a successful access response, an unsuccessful access response or a claims-requested document. The first two responses have been adopted from the OAuth V2.0 protocol [14]. The UMA protocol introduces the third option which is used when a user policy requires that one or more claims must be submitted by a requester.

Firstly, if the AM has all the required information concerning this particular access request then it may respond with a successful access response (Fig. 4 a). As defined in [14], such a response contains an access token and an optional refresh token.

An access token is a token, generally short-lived, which is used by a requester to gain access to a protected resource on a host. The refresh token can be a long-lived token that can be used by a requester to subsequently reuse already

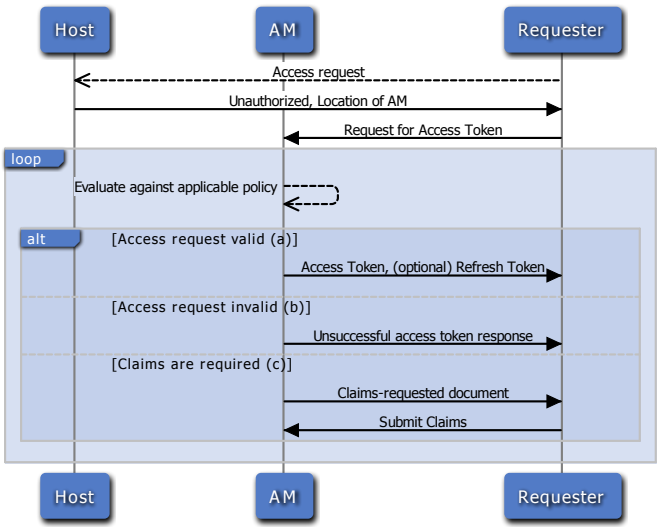

Figure 4: UMA Step 2: Requester gets access token from AM.

obtained authorization and to request fresh access tokens from AM without the need of repeating the evaluation process. The time for which both the access and the refresh tokens are valid can be controlled by a user at AM or can be determined solely by an AM, and is implementation specific.

Secondly, the AM may decide that a requester is definitively not authorized to access a particular resource according to a user's policy and it then responds with an unsuccessful access response (Fig. 4 b).

The third case is where a user's policy specifies requested claims that must be conveyed from a requesting party before an authorization is granted, such as self- or third-party-asserted identification, or a promise to adhere to access licensing terms. In such a case, AM responds with a claims-requested response containing a list of all requested claims. Upon subsequently receiving these claims from the requester, the AM performs the access request evaluation process and decides whether authorization can be granted or not. It may then respond with a successful or unsuccessful access token response, or with another claims-requested response if more claims are needed (Fig. 4 c).

At the end of this step, a requester is in a possession of an access token and an optional refresh token issued by an AM for a specific access type to a resource on a host.

4.2.3 Requester wields access token at host to gain access

This step is executed when a requester has acquired an access token from an authorization manager. A requester simply presents such a token when attempting to access a protected resource on a host as illustrated in Fig. 5.

A host can either choose to evaluate the access token locally or may use the AM for the evaluation process. In the first case, it's the host that decides whether to grant access to a resource or not, though this must be based on the received access token. If the token is valid then access to a resource is granted. In case the token is invalid then a host responds with an "HTTP 401 Unauthorized" response

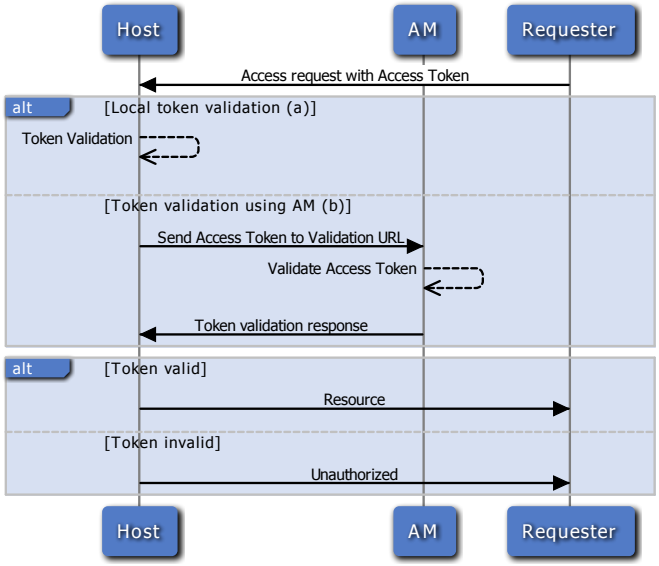

Figure 5: UMA Step 3: Requester wields access token at host to gain access.

that contains information about the location of the AM that protects this resource (recall step (2) of the protocol).

In case a host decides to use the AM for the validation process, it then sends the token to the AM's token validation URL. This request for validation is accompanied by a host's own access token previously acquired in step (1) of the UMA protocol. Additionally, a host sends information regarding the resource on which access is being attempted and the method of requested access. The AM replies with an access control decision that can be "deny" or "permit".

The remote token validation, although reducing the work that needs to be done on a host's side, may result in a significant overhead due to additional communication between a host and an AM. Therefore, the UMA protocol can be extended to support other validation models such as hybrid access token validation to address the aforementioned performance issues. In this model, the AM dynamically provisions a host with the ability to do local token validation. For example, the AM may provide a host with a piece of code that is capable of performing token validation.

After this step of the protocol, a requester gains authorized access to a protected resource or is denied access if the presented access token is invalid. In the latter case, a requester is free to seek authorization or to refresh its access token at AM using the previously acquired refresh token or by other means specified in the chosen OAuth V2.0 flow.

4.3 Claims

The UMA protocol does not constrain users in composing access control policies for their resources (subject to AM implementation limitations). Rather, it supports communicating requests for claims to requesters so that users can specify policies defining required claims which must be submitted by a requester before authorization can be granted.

Submitted claims may be affirmative, representing a statement of fact. An affirmative claim can be asserted by a requesting party or another claims issuer (e.g. can be signed by a third-party service). A statement of fact might be "The requesting party is over 18 years of age." A claim can be also promissory and can be asserted by the requesting party specifically to the authorizing user. For example, such claim may state that "The requesting party will adhere to the specific Creative Commons licensing terms indicated by the AM in accessing this resource." We envisage that the process of demanding and submitting claims would have legal enforceability consequences as necessary.

The UMA protocol uses the recently proposed Claims 2.0 specification [33] for the format of requested and submitted claims. The protocol, however, can be easily extended to make use of already established claim types such as these proposed in [24, 18, 11], and [10]. However, some of these types appear to be too complex and therefore not well-suited for Web 2.0-style implementations, and all these types lack the ability to specify required parameter values in claims; thus they may be unable to satisfy complex use cases such as these discussed in [8].

5. EVALUATION

The proposed User-Managed Access architecture and protocol address all identified shortcomings discussed in Section 2 and meet all of the requirements described in Section 3.

Requirement (1) is satisfied by including an authorization manager within our architecture and by allowing authorizing users to delegate access control from hosts to this specialized component. Moreover, all of the entities of the proposed architecture can reside in distinct Web domains. as interactions between hosts, requesters and AM are based on the standard HTTP protocol and conform to the REST architectural style [26] to the extent possible. The AM is agnostic as to the identifiers used in an individual's various hosting Web services making it possible to be deployed in "today's Web". Additionally, the AM does not impose any constraints on what resources may be protected and its functionality can be applied to arbitrary Web applications.

An authorizing user can be involved in all the steps of access control policy management such as those given in [9]. It is the user that applies access control policies to their Web resources and grants authorizations to requesters of these resources using the AM. The UMA proposal does not constrain how a user defines access control policies and the user is free to choose their preferred authorization manager that offers tools that meet a user's requirements. As such, a user can be provided with a consistent UX when composing policies and linking these policies with distributed Web resources. Hence, the UMA proposal meets requirement (2).

The UMA protocol supports communication of claims between requesters and the authorization manager before authorization can be granted. As such, a user may use the AM to define policies which demand such claims. These policies are not based solely on client credentials and fit well into the highly dynamic and open Web environment where servers may not know in advance possible identities of clients accessing data. These policies may also impose contract terms that govern ongoing access to data and may require requesting parties to convey their agreement to these terms. UMA relies on legal means to enforce such agreements. These properties of the UMA proposal meet requirement (3).

With access control being delegated from various Web applications at a user's preferred authorization manager, a user may easily monitor, change, and stop access relationships between their online services. Moreover, access control de-

cisions are performed by this central component and a user does not have to be directly involved in interactions between services accessing data and services hosting data. These UMA properties satisfy requirement (4).

6. IMPLEMENTATION

The UMA protocol has been implemented as part of the SMART project at Newcastle University [40]. The prototype implementation consists of all the components required by the protocol, i.e. the AM, host and requester. It allows an authorizing user to introduce UMA-enabled host applications to the prototype AM and to use this AM to protect resources stored at these hosts by defining access control policies and linking these policies with resources. Policies consist of a set of rules and a set of claims. Each rule represents a set of Requesting Parties and their associated access rights. Each claim represents the properties that a particular Requesting Party must prove to possess. The AM uses the Claims 2.0 specification [33] as the format for its claims.

When a requesting party makes a request for an access token for a resource or a group of resources then the AM uses its policy engine to evaluate such a request. We define rules governing policy evaluation as follows: if a policy consists of a set of rules only, then a requesting party must authenticate to the AM to prove its identity. If a policy consists of a set of claims only, then the requesting party must prove to possess the required properties. In the third case, if the policy consists of both rules and claims, then requesting parties must authenticate themselves and must present requested claims. In all three cases, the requested access type must match one of the specified types within the policy.

Claims evaluation is performed by our custom-built claims evaluation engine. This engine checks what claims have been submitted and evaluates them against the list of claims requested in the applicable policy. Each claim is evaluated by a module specific to a particular claim type. For example, a claim that specifies the age of a requesting party is evaluated by a module that is able to check whether this age is within the range specified in the policy. Our engine has been built in a modular way allowing inclusion of additional modules capable of evaluating custom claim types.

The software has been implemented in Java with the use of such frameworks as Spring, Spring MVC, Apache CXF, and Hibernate, among others. The AM has been built as a Spring application and can be deployed to such Web containers as Apache Tomcat or SpringSource tcServer. Host and Requester applications can be deployed to the Google App Engine (GAE) platform.

7. PROGRESS AND FUTURE WORK

We have described the User-Managed Access approach to authorization for Web resources. UMA addresses shortcomings that are present in other solutions to access control and aims to solve the problems discussed in Section 2 and identified in an extensive set of scenarios discussed in [8]. It also meets all of the requirements presented in Section 3.

UMA is being researched by the User-Managed Access Work Group [44] (operating at the Kantara Initiative under the charter at [2]). The aim of this work group is to develop a set of draft specifications for the protocol, and to facilitate the development of interoperable implementations of these specifications. The described protocol has been already produced as a draft specification [7] and the end result of the Kantara process is planned to be submitted to IETF [1].

Ongoing work includes developing a model for retrieving trusted identity claims from a requester to ensure access is delegated to the correct parties, and defining methods for dynamic client registration that allow for AMs to build strongly authenticated trust relationships with hosts and requesters. The group is also working on finalizing the model of resource management which would allow hosts to inform AMs which resources are meant to be protected.

The User-Managed Access group is actively collaborating with other efforts aiming to provide new approaches to authorization for Web resources such as the OAuth WG [14]. The group is continuously gathering use cases for its protocol in order to formulate further sound and concise requirements that the protocol must satisfy. The group also works on specifying design principles that the protocol should meet.

8. RELATED WORK

Access control systems for the Web have been researched extensively and aim to address mostly flexibility [17] or usability [16] challenges. These systems, however, do not appear to be well-suited to the increasingly user-driven Web 2.0 environment with an ever-growing number of resources. Recently proposed solutions aim to fit into such an environment by empowering users with more control over access rights to their data. More notable works include those discussed in [30, 41, 42, 27, 39] and [38]. In this section we provide an overview of related work that has either influenced or contributed to the User-Managed Access proposal.

OAuth V1.0 [28] is one of the early proposals to address authorization between Web services that attracted much attention in the Web community. It allows a Resource Owner to share data between two Web applications, one being a Server and the other being a Client. Access to data is authorized by a Resource Owner at the Server side which results in an access token being issued to a Client. As a result, a Client does not learn credentials of a Resource Owner and is able to make authorized access requests to resources at a Server using this acquired token.

With tokens being used for accessing protected resources, OAuth addresses the password anti-pattern [37], i.e. a user does not have to reveal their credentials to give one service access to protected resources hosted at a different service. Each OAuth service provider, however, must independently serve an authorization management function without the possibility of centralized management, defeating requirement (1) as defined in Section 3. Therefore, this specification fails to address data sharing scenarios presented in [8].

OAuth V2.0 [14] is a less complex attempt to solve the problem of authorization for Web resources. It allows a Web application, called a Resource Server, to delegate authorization to a trusted authority called an Authorization Server. A Client, when seeking access to a Protected Resource hosted on a Resource Server, must first obtain an access token from an Authorization Server and present this token along with an access request.

Despite allowing the use of an external entity to control authorization for Web resources, OAuth V2.0 fails to provide certain features required in the open Web environment. In particular, it does not support dynamic introductions by users, requires relationships between parties to be pre-established, and does not define how a Resource Server

comes to trust an Authorization Server. It also does not allow for demanding claims from Clients which we perceive as one of this protocol's major weaknesses. Therefore, OAuth V2.0 fails to address identified shortcomings (Section 2) and does not meet all requirements as discussed in Section 3.

The UMA protocol currently relies on the OAuth V2.0 protocol for its interactions between entities of the proposed architecture. It builds on two instances of this protocol to address identified shortcomings and to meet requirements as well as investigated scenarios of sharing information on the Web [8]. We show how OAuth V2.0 fits into the UMA model for access management for Web resources in Fig. 2.

Kerberos [34] is a well-established protocol that allows to delegate access control from servers, called Service Servers, to a centralized component, called the Ticket Granting Server (TGS). Clients, must first authenticate to an Authentication Server and then receive a Service Ticket from the TGS. This ticket can be later used by a client to gain access to a resource on a Service Server which validates this ticket based on a session key that was used to encrypt it.

Kerberos allows to centralize authorization for various services but it requires a well-defined point of authority, relationships between services to be pre-established and relies heavily on symmetric cryptography. As such, it is ill-fitted for the open and dynamic Web environment. Moreover, in the current form, Kerberos does little to empower users with better control over access to their online resources. The recent proposal of OAuth V2.0 support for Kerberos [13] seems to allow this protocol to be easily extendable to the Web. However, this approach inherits the previously described weaknesses of OAuth V2.0.

The SAML [18] and Liberty ID-WSF frameworks [6] are federated identity standards that enable identity information-sharing. Typical SAML deployments involve single sign-on with attribute transfer that takes place at login time, and ID-WSF defines an architecture that includes a discovery service for locating and gaining authorized token-based access to a person's various identity-enabled services. Both are able to capture and propagate a user's act of consent throughout the authorization process. Despite matching some features of SAML and ID-WSF, UMA differs from both proposals. In particular, it is independent from identifier systems and incorporates user-driven policy decision-making in addition to consent alone.

A similar approach to UMA is discussed in [32]. The proposed system, called User-Managed Access Control (UMAC), has been initially described in [31] and consists of an architecture of services and an access control protocol defining interactions between these services. A user may delegate access control from their set of Web applications to a specialized component and apply security requirements to all of their Web resources using this component.

The described architecture and protocol are closely related to User-Managed Access and have been researched virtually in parallel with our work. The authors of UMAC have adopted the terminology proposed by the UMA WG for their architecture. The discussed protocol, however, is based on redirections where a host refers a requester to the authorization component without any discovery. It also uses remote token validation which has been only recently adopted by UMA in favour of polling the authorization state of a requester from AM. A more detailed analysis of both UMA and UMAC solutions is presented in [32].

9. CONCLUSIONS

We have investigated the need for new approaches to access management for Web resources that would include a user as a core part of their model. Additionally, we have discussed the requirements for such approaches. We have presented the User-Managed Access solution and described its architecture and access control delegation protocol. We have then evaluated the UMA solution to show that it addresses identified shortcomings and satisfies all discussed requirements. We have also provided an overview of the prototype implementation of the UMA protocol.

The UMA approach provides a method for users to control third-party application access to their protected resources, residing on any number of host sites and to introduce those hosts dynamically to a user-chosen authorization manager that makes access decisions based on user instructions. UMA aims to allow users to flexibly apply the necessary security and privacy controls while retaining all the benefits of social interactions and data sharing in the Web 2.0 environment.

Acknowledgments

Authors Machulak and van Moorsel are supported by UK Technology Strategy Board grant nr. P0007E ("Trust Economics") and JISC-funded project "Student-Managed Access to Online Resources (SMART)".

The authors would like to acknowledge the work done by other members of the UMA WG towards standardization of the User-Managed Access protocol. In particular, the authors are grateful to Paul C. Bryan for his work on the protocol since its early beginning and Lukasz Moren for his help with implementing the protocol. We would also like to thank Hasan Ibne Akram for his feedback on this paper.

10. REFERENCES

[1] Internet Engineering Task Force (IETF). http://www.ietf.org. Accessed 29/06/2010.

[2] Kantara Initiative. http://kantarainitiative.org/. Accessed 29/06/2010.

[3] OpenSSO Project. http://opensso.dev.java.net/. Accessed 29/06/2010.

[4] Project hData. http://www.projecthdata.org. Accessed 29/06/2010.

[5] Project VRM - Vendor Relationship Management. http://cyber.law.harvard.edu/research/projectvrm. Accessed 29/06/2010.

[6] The ID-WSF Evolution Work Group. http://kantarainitiative.org/confluence/display/idwsf. Accessed 29/06/2010.

[7] UMA 1.0 Core Protocol. http://kantarainitiative.org/confluence/display/uma/UMA+1.0+Core+Protocol. Accessed 29/06/2010.

[8] UMA Scenarios and Use Cases. http://kantarainitiative.org/confluence/display/uma/UMA+Scenarios+and+Use+Cases. Accessed 29/06/2010.

[9] OASIS eXtensible Access Control Markup Language (XACML). http://www.oasis-open.org./committees/xacml/, 2005. Version 2.0.

[10] OpenID Attribute Exchange, Dec 2007. Version 1.0.

[11] Identity Metasystem Interoperability Version 1.0. http://www.oasis-open.org/committees/download.php/29979/identity-1.0-spec-cd-01.pdf, Nov 2008. Committee Draft 01.

[12] host-meta: Web Host Metadata. http://tools.ietf.org/html/draft-hammer-hostmeta-13, Jun 2010. (Work in Progress).

[13] OAuth 2.0 support for the Kerberos V5 Authentication Protocol. http://tools.ietf.org/html/draft-hardjono-oauth-kerberos-00, Jun 2010. (Work in Progress).

[14] The OAuth 2.0 Protocol. http://tools.ietf.org/html/draft-ietf-oauth-v2-09, Jun 2010. (Work in Progress).

[15] C. A. Ardagna et al. Expressive and Deployable Access Control in Open Web Service Applications. *IEEE Transactions on Services Computing*, Apr 2010.

[16] D. Balfanz. Usable access control for the world wide web. In *ACSAC '03: Proc. of the 19th Annual Computer Security Applications Conference*, page 406, Washington, DC, USA, 2003.

[17] L. Bauer, M. A. Schneider, and E. W. Felten. A general and flexible access-control system for the web. In *Proc. of the 11th USENIX Security Symposium*, pages 93–108, 2002.

[18] S. Cantor et al. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard, Mar. 2005.

[19] A. Cavoukian. Privacy in the clouds. *Identity in the Information Society*, 1:89–108, Dec 2008.

[20] D. H. Duane Nickull. *Web 2.0 Architectures. What entrepreneurs and information architects need to know*, volume 271. O'Reilly, 1. edition edition, May 2009.

[21] A. N. et al. WS-Trust 1.4. OASIS Standard, Feb. 2009.

[22] Eve Maler. ProtectServe draft protocol flows. http://www.xmlgrrl.com/blog/2009/04/02/protectserve-draft-protocol-flows, Mar 2009. Accessed 29/06/2010.

[23] Eve Maler. To protect and to serve. http://www.xmlgrrl.com/blog/2009/03/23/to-protect-and-to-serve, Mar 2009. Accessed 29/06/2010.

[24] S. Farrell and R. Housley. An Internet Attribute Certificate Profile for Authorization. RFC 3281 (Draft Standard), June 2002.

[25] R. Fielding et al. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Updated by RFC 2817.

[26] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[27] R. Geambasu et al. Organizing and sharing distributed personal web-service data. In *WWW '08: Proc. of the 17th Intl. Conference on World Wide Web*, pages 755–764, New York, NY, USA, 2008.

[28] E. Hammer-Lahav. The OAuth 1.0 Protocol. RFC 5849 (Draft Standard), 2010.

[29] M. Hart, R. Johnson, and A. Stent. More content - less control: Access control in the web 2.0. In *WOSP '08: Proc. of the first workshop on Online social networks*, pages 43–48, New York, NY, USA, 2008.

[30] F. Hsu and H. Chen. Secure File System Services for Web 2.0 Applications. In *CCSW '09: Proc. of the 2009 ACM Workshop on Cloud Computing Security*, pages 11–18, New York, NY, USA, 2009. ACM.

[31] M. P. Machulak and A. van Moorsel. A Novel Approach to Access Control for the Web. Technical Report CS-TR-1157, School of Computing Science, Newcastle University, Jul 2009.

[32] M. P. Machulak and A. van Moorsel. Architecture and Protocol for User-Controlled Access Management in Web 2.0 Applications. In *ICDCS-SPCC 2010: Proc. of the First ICDCS Workshop on Security and Privacy in Cloud Computing*, Genoa, Italy, Jun 2010.

[33] E. L. Maler and P. C. Bryan. Claims 2.0. http://kantarainitiative.org/confluence/display/uma/Claims+2.0. Accessed 29/06/2010.

[34] C. Neuman, S. Hartman, and K. Raeburn. The Kerberos Network Authentication Service (V5). RFC 4120 (Draft Standard), 2005.

[35] T. O'Reilly. What Is Web 2.0. Design Patterns and Business Models for the Next Generation of Software. http://oreilly.com/web2/archive/what-is-web-20.html, Sep 2005. Accessed 29/06/2010.

[36] T. O'Reilly and J. Battelle. Web Squared: Web 2.0 Five Years On. http://oreilly.com/web2/archive/what-is-web-20.html, Oct 2009. Accessed 29/06/2010.

[37] R. Paul. OAuth and OAuth WRAP: defeating the password anti-pattern. http://arstechnica.com/open-source/guides/2010/01/oauth-and-oauth-wrap-defeating-the-password-anti-pattern.ars, Jan 2010. Accessed 29/06/2010.

[38] A. Simpson. On the need for user-defined fine-grained access control policies for social networking applications. In *SOSOC '08: Proc. of the Workshop on Security in Opportunistic and SOCial networks*, pages 1–8, New York, NY, USA, 2008.

[39] D. K. Smetters. Building Secure Mashups. In *W2SP '08: Proc. of the Workshop on Web 2.0 Security and Privacy*, Oakland, CA, USA, May 2008.

[40] Student-Managed Access to Online Resources (SMART). http://research.ncl.ac.uk/smart. Accessed 29/06/2010.

[41] S.-T. Sun, K. Hawkey, and K. Beznosov. Secure web 2.0 content sharing beyond walled gardens. In *ACSAC '09: Proc. of the 25th Annual Computer Security Applications Conference*, pages 409–418, Dec 2009.

[42] A. Tootoonchian et al. Lockr: social access control for web 2.0. In *WOSP '08: Proc. of the First Workshop on Online Social Networks*, pages 43–48, New York, NY, USA, 2008.

[43] UMA Requirements. http://kantarainitiative.org/confluence/display/uma/UMA+Requirements. Accessed 29/06/2010.

[44] User-Managed Access Work Group. http://kantarainitiative.org/confluence/display/uma. Accessed 29/06/2010.

[45] R. Yavatkar, D. Pendarakis, and R. Guerin. A framework for policy-based admission control. RFC 2753 (Draft Standard), Jan. 2000.