User-Managed Access to Web Resources

Maciej P. Machulak, Eve L. Maler, Domenico Catalano and Aad van Moorsel

TECHNICAL REPORT SERIES

No. CS-TR-1196 **March, 2010**

TECHNICAL REPORT SERIES

No. CS-TR-1196 **March, 2010**

User-Managed Access to Web Resources

M.P. Machulak, E.L. Maler, D. Catalano, A. van Moorsel

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 in creating, enriching and personalizing these services. As a result, potentially large amounts of personal, sensitive, and valuable data is put online, spread across various Web services. Users willingly share this data with other users and services on the Web, but are also concerned about maintaining privacy and keeping their personal data secure.
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 in this paper 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 authorization 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. It also supports requesters in accessing such data.

Bibliographical details

MACHULAK, M.P., MALER, E.L., CATALANO, D., VAN MOORSEL, A.

User-Managed Access to Web Resources
[By] M.P. Machulak, E. L. Maler, D. Catalano, A. van Moorsel
Newcastle upon Tyne: University of Newcastle upon Tyne: Computing Science, 2010.

(University of Newcastle upon Tyne, Computing Science, Technical Report Series, No. CS-TR-1196)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE
Computing Science. Technical Report Series. CS-TR-1196

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 in creating, enriching and personalizing these services. As a result, potentially large amounts of personal, sensitive, and valuable data is put online, spread across various Web services. Users willingly share this data with other users and services on the Web, but are also concerned about maintaining privacy and keeping their personal data secure. 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 in this paper 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 authorization 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. It also supports requesters in accessing such data.

About the authors

Maciej Machulak received his MSc in Computing Engineering from Wroclaw University of Technology in Poland in 2007. During his studies he was an Erasmus Exchange Student at Newcastle University. Maciej additionally completed the Advanced MSc degree in "System Design for Internet Applications" (SDIA) in 2007 at Newcastle University and his thesis was awarded Best 2007 SDIA Thesis. This degree included an industrial placement at Red Hat UK Ltd. Maciej's main task at Red Hat was to develop a framework for transactional Web Services. Before commencing his PhD studies Maciej was also employed as an intern at Red Hat and worked on embedded tools for transaction monitoring for the JBoss Application Server. Maciej Machulak is currently a PhD student working with Prof. Aad van Moorsel on the Trust Economics project, funded by the UK Technology Strategy Board (TSB). Maciej's project is concerned with building a novel authorization solution for the Web which allows users to flexibly adapt access control for their Web resources to their particular security requirements. Maciej is also involved in the User-Managed Access (UMA) Work Group where he is a Specification Editor, an Implementation Coordinator and a Use Case Editor. He is also a Project Manager for the JISC-funded project "Student-Managed Access to Online Resources" (SMART) that aims to develop a User-Managed Access compliant system.

Eve Maler is a Distinguished Engineer in PayPal’s Identity Services group, where she drives the development of security and identity strategies for enabling consumer choice in permissioning of personal data sharing. Eve was one of the inventors of XML; she also co-founded the SAML effort and has made major leadership, technical, and educational contributions to many other standards and technical communities. In recent times she has focused primarily on consumer trust, privacy, and empowerment issues in Web identity and permissioned data-sharing. She launched an open effort called User-Managed Access to explore long-term solutions in this area. Eve is a sought-after public speaker, and serves as the chair of the Web Services and Identity track of the annual XML Summer School held at University of Oxford. Eve co-authored Developing SGML DTDs: From Text to Model to Markup, a book that provided a unique methodology for information analysis and SGML schema design. Eve’s blog, Pushing String at xmlgrrl.com, touches on topics both technical and whimsical.

Domenico Catalano is a Senior Architect in the Software Line of Business at Sun Microsystems Italy. He focuses on designing Identity and Access Management solutions, and he is also an evangelist for federation and Web 2.0 identity emerging technologies. Before joining Sun in July 1999 he worked in a Security Consulting company. He received his BSc in Computer Science at the University of Salerno (Italy), with a thesis on Data Security and Cryptography. Domenico is the author or co -author of several journal articles regarding Identity Management, and he blogs about identity and security at blogs.sun.com/domcat. Domenico is a leadership team member of the Kantara User-Managed Access (UMA) Work Group.

Aad van Moorsel joined the University of Newcastle in 2004. He worked in industry from 1996 until 2003, first as a researcher at Bell Labs/Lucent Technologies in Murray Hill and then as a research manager at Hewlett-Packard Labs in Palo Alto, both in the United States. Aad got his PhD in computer science from Universiteit Twente in The Netherlands (1993) and has a Masters in mathematics from Universiteit Leiden, also in The Netherlands. After finishing his PhD he was a postdoc at the University of Illinois at Urbana-Champaign, Illinois, USA, for two years. Aad has worked in a variety of areas, from performance modelling to systems management, from web services to cloud computing and on issues of security and trust. In his last position in industry, he was responsible for HP's research in web and grid services, and worked on the software strategy of the company. His research agenda aims at establishing an intelligent enterprise, with a specific focus on trust, privacy and security. The goal is to provide tools to improve IT decision making, if possible based on objective, quantitative methods, eventually fully automated. This involves mathematical modelling, algorithms and service-oriented software implementations The recent research is highly interdisciplinary, using ideas from social and business sciences, to gain a deeper understanding of issues of trust in, for instance, cloud computing. DTI, EPSRC and EU-funded collaborations are ongoing with Hewlett-Packard, Merrill-Lynch, Warwick, Bath, UCL, various universities throughout Europe, and the Business School as well as the Medical School in Newcastle.

Suggested keywords

WEB 2.0
SECURITY
PRIVACY
ACCESS CONTROL
AUTHORIZATION

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
Sun Microsystems
Rome, Italy
domenico.catalano@sun.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 in creating, enriching and personalizing these services. As a result, potentially large amounts of personal, sensitive, and valuable data is put online, spread across various Web services. Users willingly share this data with other users and services on the Web, but are also concerned about maintaining privacy and keeping their personal data secure.**

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 in this paper 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 authorization 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. It also supports requesters in accessing such data.

I. 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 to this environment [31], [32]. It has also influenced the way people engage with one another, collaborate, 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 [18]. 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, often 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 [27], [30]. These mechanisms are configured using policies that are specified in diverse and often incompatible policy languages using various tools at every Web application. This prevents a user from easily monitoring, changing, or stopping access relationships 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 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 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 controls 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 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. This gives users the required flexibility in sharing their data and supports them in their participation in interactions and collaboration on the Web. It also supports potential requesters with accessing a user's data.

The remainder of the paper is organized as follows. We provide a requirements analysis for a user-managed access control solution in Section II. In Section III, we explain the User-Managed Access approach to authorization for Web resources which meets all presented requirements. We evaluate this approach against these requirements in Section IV. We

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

discuss progress and future work in Section V. We examine related work in Section VI and we conclude in Section VII.

II. Requirements Analysis

New approaches to access control for Web resources should allow the user to quickly determine what information is shared with what parties and for what purposes [17], 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. These properties, however, appear not to be fully covered in existing authorization solutions.

Therefore, there is a need to formulate sound and concise requirements for a novel access management solution that would fit precisely into the Web 2.0 environment. These requirements have been initially presented in [41]. They address the shortcomings of existing access management systems, such as those based on XACML [8], that we perceive as either inflexible or insufficiently user-managed. These systems seem to lack the sophistication, simplicity and usability required to respond to security and privacy challenges in the highly user-driven Web environment.

We discuss each of the formulated requirements in more detail:

1) Access relationship service: A successful user-managed access control solution should support the notion of a distinct online service for managing data-sharing and service-access relationships between an individual and their online services that request such access.

2) User-driven policies and terms: The solution should allow an individual to select policies and enforceable contract terms that govern access, as well as data storage, further usage, and further sharing on the part of requesting services.

3) User-managed access relationships: It should allow an individual to conduct short-term and long-term management of access relationships, including modifying of the conditions of access or terminating the relationship entirely.

4) Auditing: It should be possible for an individual to audit and monitor various aspects of access relationships for the purpose of data sharing analytics.

5) Requester-Host direct access: The access control solution should allow requesting services to interact directly with hosting services in a fashion guided by policy without the user being involved in these interactions. Real-time user approval should be reserved for extraordinary circumstances.

6) Multiple hosting services: Requesting services should be able to interact with multiple data hosting services associated with the same individual.

7) Entity separation: A user should be able to store resources at a host in one Web domain and protect these resources with an access relationship service residing in a different domain. Correspondingly, the requester could reside in a different domain as well.

8) Resource orientation: User data access and service access should be enabled through accessing Web resources that have URLs.

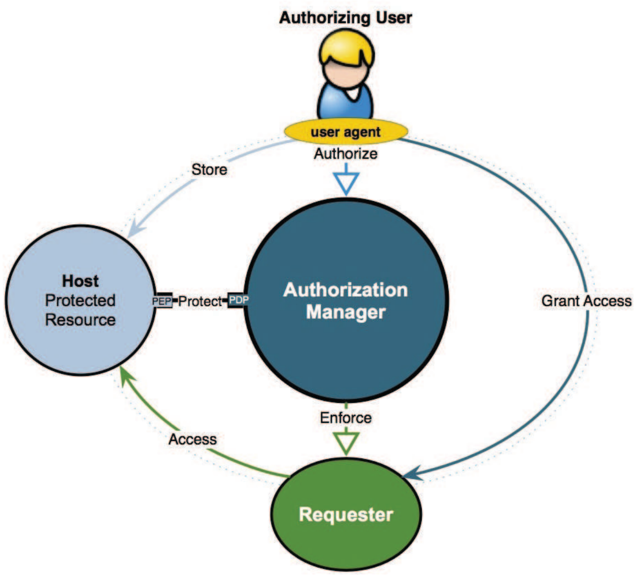

Fig. 1. Interactions between entities involved in the User-Managed Access (UMA) protocol.

9) Representation agnostic access control: The access relationship service should not be required to understand the representations of resources it is charged with protecting. As such, the functionality of this service should be applicable to arbitrary resources on the Web.

10) Preservation of user's privacy: For resources at hosting service X and resources at hosting service Y, neither X nor Y should be able to find out, through their relationship with the access relationship service, that the same individual uses the other service.

In the next section, we discuss a new access control solution for Web resources that meets all formulated requirements.

III. Approach

User-Managed Access (UMA) to Web resources is a novel access management solution based on a new access control delegation protocol. This protocol 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 satisfy requirements presented in Section II and formulated in [41]. A high-level view of entities involved in UMA is depicted in Fig. 1.

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 provide data to requesting services. Instead, the user may provide authoritative sources from which Web services can request such data directly in a secure and efficient way.

The UMA protocol has been researched by the User-Managed Access Work Group (UMA WG) [42]. It has been initially proposed in [22] and defined in [21] but has since undergone significant modifications regarding the adoption of the OAuth Web Resource Authorization Profiles (WRAP) [14]. We discuss how WRAP fits into our model throughout the paper. Additionally, we show WRAP-based interactions between entities of the UMA architecture in Fig. 2.

A. Architecture

As shown in Fig. 1, User-Managed Access is based on four main entities: Authorizing User, Authorization Manager, Host, and Requester. We provide a brief overview of each of these entities in subsequent sections. For a more detailed explanation of these terms we refer the reader to [40] and [6].

1) Authorizing User: An Authorizing User delegates access control from their chosen set of Hosts to an Authorization Manager. Such a user is also responsible for configuring an Authorization Manager with policies that control how this component makes access decisions when a Requester attempts to access a Protected Resource at a Host, thus serving as their own policy administrator.

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 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 [43], and plays the conceptual role of a Security Token Service as defined in [19].

3) Host: A Host is a Web application that is used by an Authorizing User to store and manage Protected Resources and to share 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) 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 Requesting Party that can be a person or a company 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. We discuss an extensive set of use cases with various settings of the proposed entities of the UMA solution in [7].

B. Delegation Protocol

The User-Managed Access protocol describes interactions between all of the previously defined entities. It consists of the following steps, which are currently defined as extensions and profiles of the WRAP protocol [14] while the OAuth V2.0 protocol [4] is under development: (1) User registers host at AM, (2) Requester gets access token from AM, (3) Requester wields access token at host to gain access (Fig. 2). For the sake of completeness, we also describe how a user may define access control policies at AM for resources stored at a host. The protocol, however, does not impose any constraints on how this step should be performed.

1) User registers host at AM: In this step a user establishes a trust relationship between a host and an authorization manager (Fig. 3). This can be achieved 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 [11] to obtain a metadata document from the AM. Such a document defines the location of a user authorization URL, an access token URL, and a token validation URL. It also defines access token formats and claim formats that this AM generates. We refer the reader to [6] for examples of such a metadata document.

The user authorization URL is used by a host to initiate the process of acquiring authorization to use a particular AM. The access token URL is used by a host to obtain an access token for this AM. A host provides this URL to requesters so they can acquire access tokens necessary to access protected resources on this host. The token validation URL can be used by the host to validate access tokens received from requesters.

When a host receives the metadata document from the AM, it then uses one of the user delegation profiles (e.g. Web App Profile), as defined in the WRAP specification [14], to obtain the user's authorization to use this AM. This is achieved by receiving an access token authorized by a user from an authorization manager. This token allows a host to make authorized access requests to the AM. We discuss access tokens in more detail when describing step (2) of the UMA protocol.

A host may delegate access control to the AM for all its resources, for resources of a particular user or for a specific subset of resources only. This, however, is implementation specific and the protocol itself supports all three granularity levels of access control delegation.

At the end of this step, a host is capable of making authorized access requests to the AM in order to validate access requests to protected resources issued by requesters.

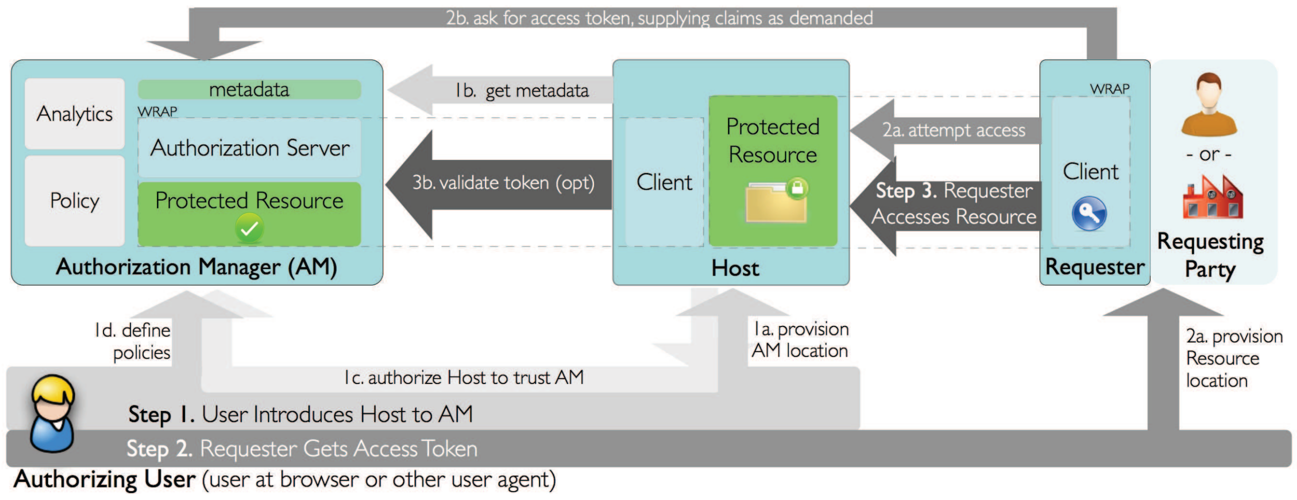
2b. ask for access token, supplying claims as demanded
Analytics
metadata
WRAP
Authorization Server
Policy
Protected Resource
Authorization Manager (AM)
1b. get metadata
3b. validate token (opt)
Client
Protected Resource
Host
2a. attempt access
Step 3. Requester Accesses Resource
WRAP
Client
Requester
- or -
Requesting Party
1d. define policies
1c. authorize Host to trust AM
1a. provision AM location
2a. provision Resource location
Step 1. User Introduces Host to AM
Step 2. Requester Gets Access Token
Authorizing User (user at browser or other user agent)

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

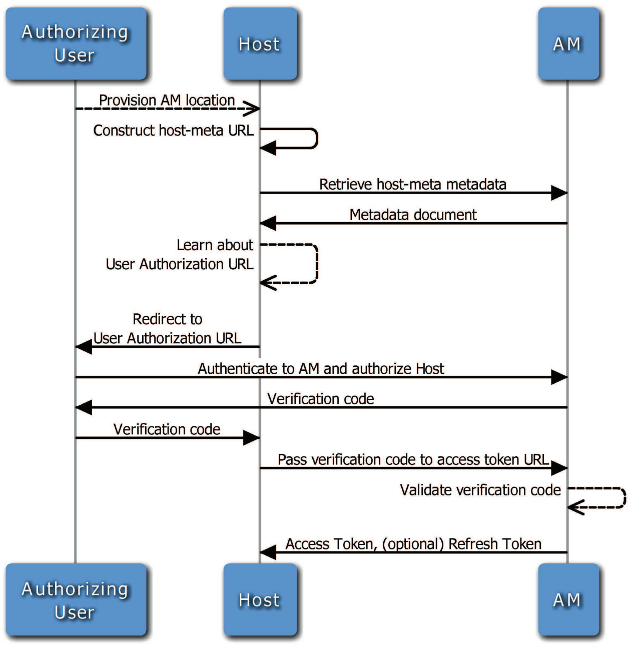
Authorizing User
Host
AM
Provision AM location
Construct host-meta URL
Retrieve host-meta metadata
Metadata document
Learn about User Authorization URL
Redirect to User Authorization URL
Authenticate to AM and authorize Host
Verification code
Verification code
Pass verification code to access token URL
Validate verification code
Access Token, (optional) Refresh Token
Authorizing User
Host
AM

Fig. 3. UMA Step 1: User registers Host at AM.

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. The AM may support simple access control matrix type policies or may provide a variety of flexible policy languages and policy engines to support policy composition and evaluation respectively. We envisage that support for specific policy languages and management tools may dictate the choice of a specific AM 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 defining subjects and their access rights to a user's resources. Moreover, 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 a user's consent to be provided in real time. Different examples of policies are discussed in more detail in [7].

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 host may provide 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.

At the end of this step, a set of resources is successfully associated with one or more access control policies defined by a user. The AM evaluates access requests issued by a requester against these policies.

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 standard "HTTP 401 Unauthorized" response as defined in [24]. Such response also contains information about the location of the AM that protects this resource. We use the "WWW-Authenticate: WRAP" header to specify the URL of the authorization manager where an access token can be obtained. 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 is known in advance to the requester.

Once the requester learns about an access token URL at AM, it adheres to the WRAP protocol, with a few key UMA extensions, to obtain an access token for a protected resource at a host. UMA adopts one of the existing profiles, 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 profile. However, if a requesting party is a different person or a company then it uses one of the autonomous client profiles with the added semantic that the operator of the client is different from the authorizing user.

In both cases a requester issues a request to the access token URL of the AM. Such a request contains information about the method that the requester wants to execute on a protected resource and the scope to which access should be granted. A requester may provide information concerning multiple methods and multiple scopes at a host. Such information is a simple URL-encoded JSON object. We refer the reader to [6] 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.

Once the AM decides whether the access request is valid or not, it may respond to a requester in one of three ways. It can respond with a successful access response, an unsuccessful access response or a claims-required document. The first two responses have been adopted from the WRAP protocol and conform to those defined in [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, on the other hand, can be a long-lived token that can be used by a requester to subsequently reuse already 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 required 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-required response containing a list of all required claims. Upon receiving these claims, 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-required response if more claims are needed (Fig. 4 c).

The User-Managed Access protocol does not define the

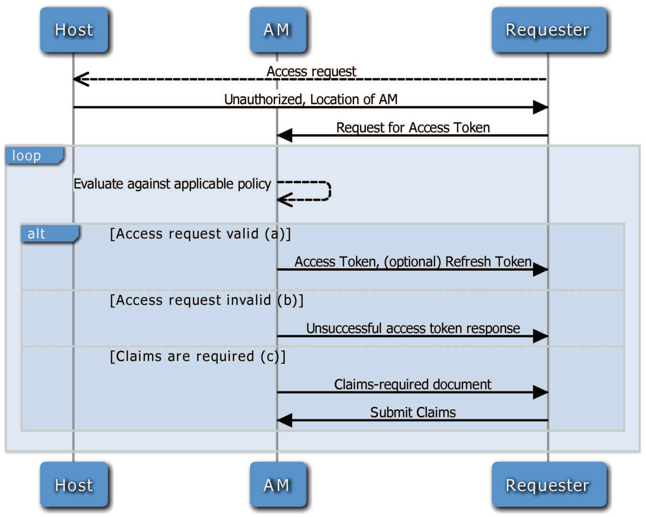
Host
AM
Requester
Access request
Unauthorized, Location of AM
Request for Access Token
loop
Evaluate against applicable policy
alt
[Access request valid (a)]
Access Token, (optional) Refresh Token
[Access request invalid (b)]
Unsuccessful access token response
[Claims are required (c)]
Claims-required document
Submit Claims
Host
AM
Requester

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

format of the claims-required document and the claims themselves. We envisage that protocol may use already existing claim types such as Information Cards [10] or SAML assertions [20]. However, we do see potential shortcomings of these and other claims specifications and we are currently working on providing a more flexible way of expressing claims as discussed further in this section.

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

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 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 may respond in two ways to a host: the token is valid, or the token is invalid. Based on the AM's response, a host allows or rejects the access request.

The UMA WG is also researching a hybrid token validation model where AM dynamically provisions a host with the

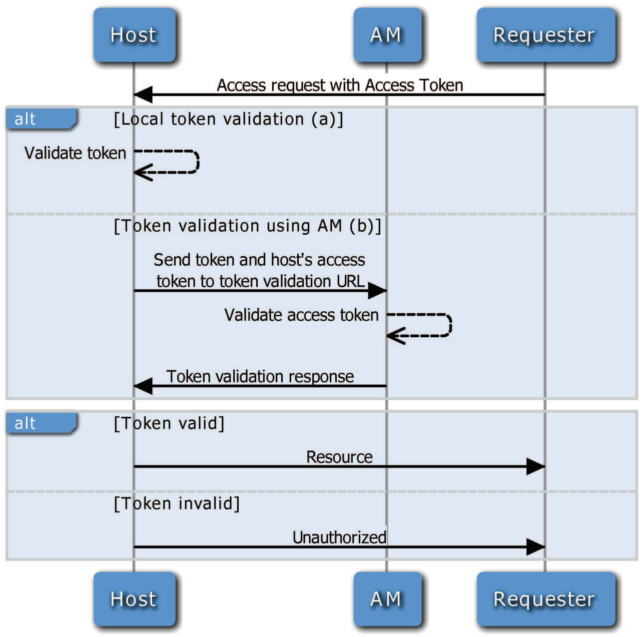
Host
AM
Requester
Access request with Access Token
alt
[Local token validation (a)]
Validate token
[Token validation using AM (b)]
Send token and host's access token to token validation URL
Validate access token
Token validation response
alt
[Token valid]
Resource
[Token invalid]
Unauthorized
Host
AM
Requester

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

ability to do local token validation. Such approach would address performance issues related to remote token validation.

After this step of the protocol, a requester gains authorized access to a protected resource or is denied access to a resource 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 WRAP profile.

C. Claims

The User-Managed Access protocol does not constrain users in composing access control policies for their protected resources (subject to AM implementation limitations). Hence, a user may specify policies defining required claims which must be submitted by a requester before authorization can be granted. These claims may refer to data storage, further usage, and further sharing on the part of requesting services.

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." We envisage that the process of demanding and submitting claims would have legal enforceability consequences as necessary.

The User-Managed Access protocol, as previously discussed, does not depend on any specific claim type that a requester may need to submit to an authorization manager. The protocol can make use of already established types such as these proposed in [23], [20], [10], [9], and [13]. However, some of these types are too complex and therefore not well-suited for Web 2.0-style implementations, and all these types appear to 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 [7]. Therefore, the UMA WG plans to define a new simple and extensible claim type that would fit precisely into the needs of User-Managed Access. This prototype would use JSON as its format and would allow for parameterized claims to be defined.

IV. Evaluation

The proposed User-Managed Access architecture and delegation protocol meet all of the requirements described in Section II. 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.

An authorizing user can be involved in all the steps of access control policy management such as those given in [8]. It is the user that applies access control policies to their distributed set of Web resources and grants authorizations to requesters of these resources. The UMA proposal does not constrain how a user defines access control policies. It supports the policy-driven ability to demand "claims" which may govern access, as well as data storage, further usage and sharing by requesters. Moreover, the user is able to monitor, change, and stop access relationships between their online services. Hence, the UMA proposal meets requirements (2), (3) and (4) respectively.

A requester is able to access protected resources only when wielding an access token issued by an authorization manager that protects these resources. The step of obtaining such a token does not oblige a user to be directly involved in these interactions (unless the user chooses policies that demand their consent in real time) and it therefore satisfies requirement (5). Additionally, a requester is free to obtain more than one access token to access protected resources on different hosts associated with the same authorizing user. This property of the UMA proposal meets requirement (6).

All of the entities of the proposed architecture can reside in distinct Web domains. Interactions between these entities are based on the standard HTTP protocol and conform to the REST architectural style [25] to the extent possible. This makes the User-Managed Access approach satisfy the formulated requirement (7). Moreover, the described solution is agnostic as to the identifiers used in an individual's various Web services making it possible to be deployed in "today's Web". Additionally, it does not impose any constraints on what resources may be protected by an authorization manager. These properties of the discussed UMA solution satisfy requirements (8) and (9) respectively.

An authorizing user may establish a trust relationship between multiple hosts and a single authorization manager. In the presented approach, none of these hosts can recognize that an individual has established a trust relationship between a different host and the same AM and is therefore using a particular service on the Web. As such, the User-Managed Access protocol preserves the user's privacy, which meets the formulated requirement (10).

V. 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 identified in an extensive set of scenarios discussed in [7]. Moreover, it meets all of the requirements presented in Section II.

The UMA proposal is being researched by the User-Managed Access Work Group [42] (operating at the Kantara Initiative under the charter at [3]). 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 [6] and the end result of the Kantara process is planned to be submitted to IETF [1].

We plan to empirically test the UMA protocol using multiple independent implementations [39]. One of the authors is currently working on an open source prototype Java implementation under the Student-Managed Access to Online Resources (SMART) project [36] funded by JISC [2]. The project will develop an UMA-compliant access control solution, deploy it within Newcastle University and evaluate the entire system through user studies. Another UMA group participant also plans to have an open source C# implementation of the protocol jointly conducted by Technical University of Munich and Fraunhofer Institute of Secured Information Technology.

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 [4]. 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.

VI. Related Work

Access control systems for the Web have been researched extensively and aim to address mostly flexibility [16] or usability [15] 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 [28], [37], [38], [26], [35] and [34]. In this section we provide an overview of related work that has either influenced or contributed to the User-Managed Access proposal.

OAuth [12] 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 [33], 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 II. Therefore, this specification is unable to address data sharing scenarios presented in [7].

The OAuth Web Resource Authorization Profiles (WRAP) [14] is a less complex attempt to solve the problem of authorization for Web resources that has been recently submitted to OAuth V2.0 work in IETF. WRAP allows a Web application, called a Protected Resource, to delegate authorization to a trusted authority called an Authorization Server. A Client, when seeking access to a Protected Resource, must first obtain an access token from an Authorization Server and present this token along with an access request. However, WRAP leaves unspecified how a Protected Resource comes to trust an Authorization Server, and does not allow for demanding claims from Clients. This defeats requirements (3) and (2) respectively that are discussed in Section II.

The User-Managed Access protocol currently relies on the WRAP protocol for its interactions between entities of the proposed architecture. It builds on two instances of this protocol to meet formulated requirements [41] and to satisfy investigated scenarios of sharing information on the Web [7]. We show how WRAP fits into the UMA model for access management for Web resources in Fig. 2.

A similar approach to User-Managed Access is discussed in [30]. The proposed system, called User-Managed Access Control (UMAC), has been initially described in [29] 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 the User-Managed Access solution 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 can refer a requester to the authorization component without any discovery mechanisms. 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 [30].

VII. 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 satisfies all discussed requirements.

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 "SMART - Student-Managed Access to Online Resources".

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. We would also like to thank Hasan Ibne Akram for his feedback on this paper.

REFERENCES

[1] Internet Engineering Task Force (IETF). http://www.ietf.org. Accessed 18/03/2010.
[2] Joint Information Systems Committee (JISC). http://www.jisc.ac.uk/. Accessed 18/03/2010.
[3] Kantara Initiative. http://kantarainitiative.org/. Accessed 18/03/2010.
[4] OAuth WG. https://www.ietf.org/mailman/listinfo/oauth. Accessed 18/03/2010.
[5] Project VRM - Vendor Relationship Management. http://cyber.law.harvard.edu/research/projectvrm. Accessed 18/03/2010.
[6] UMA 1.0 Core Protocol. http://kantarainitiative.org/confluence/display/uma/UMA+1.0+Core+Protocol. Accessed 18/03/2010.
[7] UMA Scenarios and Use Cases. http://kantarainitiative.org/confluence/display/uma/UMA+Scenarios+and+Use+Cases. Accessed 18/03/2010.
[8] OASIS eXtensible Access Control Markup Language (XACML). http://www.oasis-open.org./committees/xacml/, 2005. Version 2.0.
[9] OpenID Attribute Exchange. http://openid.net/specs/openid-attribute-exchange-1_0.html, December 2007.
[10] Identity Metasystem Interoperability Version 1.0. http://www.oasis-open.org/committees/download.php/29979/identity-1.0-spec-cd-01.pdf, November 2008. Committee Draft 01.
[11] host-meta: Web Host Metadata. http://tools.ietf.org/html/draft-hammer-hostmeta-05, November 2009. (Work in Progress).
[12] OAuth Core 1.0 Revision A. http://oauth.net/core/1.0a/, June 2009.
[13] Simple Web Token. http://oauth-wrap-wg.googlegroups.com/web/SWT-v0.9.5.1.pdf, November 2009. Version 0.9.5.1.
[14] OAuth WRAP - Web Resource Authorization Profiles. http://tools.ietf.org/html/draft-hardt-oauth-01, January 2010. (Work in Progress).
[15] Dirk Balfanz. Usable access control for the world wide web. *Computer Security Applications Conference, Annual*, 0:406, 2003.
[16] Lujo Bauer, Michael A. Schneider, and Edward W. Felten. A general and flexible access-control system for the web. In *Proceedings of the 11th USENIX Security Symposium*, pages 93–108, Berkeley, CA, USA, 2002. USENIX Association.
[17] Ann Cavoukian. Privacy in the clouds. *Identity in the Information Society*, 1:89–108, December 2008.
[18] Dion H. Duane Nickull. *Web 2.0 Architectures. What entrepreneurs and information architects need to know*, volume 271. O'Reilly, 1. edition edition, May 2009.
[19] A. Nadalin et al. WS-Trust 1.4. OASIS Standard, 2009.
[20] S. Cantor et al. Assertions and protocols for the oasis security assertion markup language (saml) v2.0. OASIS Standard, 2005.
[21] Eve Maler. ProtectServe draft protocol flows. http://www.xmlgrrl.com/blog/2009/04/02/protectserve-draft-protocol-flows/, March 2009. Accessed 18/03/2010.
[22] Eve Maler. To protect and to serve. http://www.xmlgrrl.com/blog/2009/03/23/to-protect-and-to-serve/, March 2009. Accessed 18/03/2010.
[23] S. Farrell and R. Housley. An internet attribute certificate profile for authorization. RFC 3281 (Draft Standard), June 2002.
[24] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Updated by RFC 2817.
[25] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, IRVINE, 2000.
[26] Roxana Geambasu, Cherie Cheung, Alexander Moshchuk, Steven D. Gribble, and Henry M. Levy. Organizing and sharing distributed personal web-service data. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 755–764, New York, NY, USA, 2008. ACM.
[27] M. Hart, R. Johnson, and A. Stent. More content - less control: Access control in the web 2.0. In *WOSP '08: Proceedings of the first workshop on Online social networks*, pages 43–48, New York, NY, USA, 2008. ACM.
[28] Francis Hsu and Hao Chen. Secure File System Services for Web 2.0 Applications. In *CCSW '09: Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, pages 11–18, New York, NY, USA, 2009. ACM.
[29] 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, July 2009.
[30] M. P. Machulak and A. van Moorsel. Architecture and protocol for user-controlled access management in web 2.0 applications. Technical Report CS-TR-1191, School of Computing Science, Newcastle University, March 2010.
[31] Tim 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, September 2005. Accessed 18/03/2010.
[32] Tim OReilly and John Battelle. Web Squared: Web 2.0 Five Years On. http://oreilly.com/web2/archive/what-is-web-20.html, October 2009. Accessed 18/03/2010.
[33] Ryan 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, January 2010. Accessed 18/03/2010.
[34] Andrew Simpson. On the need for user-defined fine-grained access control policies for social networking applications. In *SOSOC '08: Proceedings of the workshop on Security in Opportunistic and SOCial networks*, pages 1–8, New York, NY, USA, 2008. ACM.
[35] D. K. Smetters. Building secure mashups. In *W2SP '08: Proceedings of the Workshop on Web 2.0 Security and Privacy*, Oakland, CA, USA, May 2008.
[36] Student-Managed Access to Online Resources (SMART) Project. http://www.jisc.ac.uk/whatwedo/programmes/aim/smart.aspx. Accessed 18/03/2010.
[37] San-Tsai Sun, Kirstie Hawkey, and Konstantin Beznosov. Secure web 2.0 content sharing beyond walled gardens. In *ACSAC '09: Proceedings of the 25th Annual Computer Security Applications Conference*, pages 409–418. IEEE Computer Society, December 2009.
[38] Amin Tootoonchian, Kiran Kumar Gollu, Stefan Saroiu, Yashar Ganjali, and Alec Wolman. Lockr: social access control for web 2.0. In *WOSP '08: Proceedings of the first workshop on Online social networks*, pages 43–48, New York, NY, USA, 2008. ACM.
[39] UMA Implementations. http://kantarainitiative.org/confluence/display/uma/Implementations. Accessed 18/03/2010.
[40] UMA Lexicon. http://kantarainitiative.org/confluence/display/uma/Lexicon. Accessed 18/03/2010.
[41] UMA Requirements. http://kantarainitiative.org/confluence/display/uma/UMA+Requirements. Accessed 18/03/2010.
[42] User-Managed Access Work Group. http://kantarainitiative.org/confluence/display/uma. Accessed 18/03/2010.
[43] R. Yavatkar, D. Pendarakis, and R. Guerin. A framework for policy-based admission control. RFC 2753 (Draft Standard), January 2000.