The WebID Protocol Enhanced With Group Access, Biometrics, and Access Policies

[*]Cory Sabol, [€]William Nick, [€]Maya Earl, [€]Joseph Shelton, and [€]Albert Esterline
[*]Dept. of Computer Science & [€]Dept. of Computer Science
[*]University of North Carolina at Greensboro & [€]North Carolina A&T State University
Greensboro, North Carolina
cssabol@uncg.edu, {wmnick, mnearl, jashelt1}@aggies.ncat.edu, esterlin@ncat.edu

Abstract

The WebID protocol solves the challenge of remembering usernames and passwords. We enhance this protocol in three ways. First, we give it the ability to manage groups of agents and control their access to resources on the Web. Second, we add support for biometric access control to enhance security. Finally, we add support for OWL-based policies that may be federated and result in flexible access control.

Introduction

The primary way of authenticating users on the Web is via username and password. This is a rather secure and time-tested method, but it is cumbersome and error prone. A user may forget their username or password, requiring them to choose the ever so prevalent "forgot password" option. If the way a password reset is handled is insecure, then this opens up an entirely new attack surface for malicious actors to gain information about a user. This is where alternatives such as OpenID (Wang Chen & Wang, 2012), EMCSSL (Emercoin, 2015), and WebID (W3C, 2015) come into play. These systems are distributed platforms that provide a user with, ideally, a simple way of asserting their identity on the Web. In this paper, we focus on the WebID protocol, identification for groups and some ways in which the security of WebID can be improved for applications in which it is deemed necessary.

The remainder of this paper is organized as follows. The next section presents Semantic Web standards (such as RDF and OWL), which are used for representation in our protocols. Next is a section that introduces the WebID protocol and its use. Next, we introduce the problem of group access with WebIDs, and the following section presents our solution. The next section discusses our use of access policies, reasoning about policies, and enforcing policies. Next, we present biometrics (the use of a person's physiological and behavioral characteristics for authentication) and our biometric enhancement of the WeID protocol. The penultimate section briefly discusses implementation technologies, and the last section concludes with a summary of this paper.

Semantic Web Standards

RDF/RDFS/OWL

Our enhancements make heavy use of the Resource Description Framework (RDF). RDF is a specification for managing metadata of resources on the Web. In RDF, all information is represented as triples of the form *subject, predicate, object* (Broekstra, Kampman & Harmelen, 2002). The subject is the resource in question, and the predicate might describe a characteristic that the subject has, in which case the object is the value of that characteristic. Or the predicate describes a relationship between the subject and the object, as in *Tim knows Sarah*, where *Tim* is the subject, *knows* is the predicate, and *Sarah* is the object.

RDF lets us create a model of a resource and the relationships it has with other resources. RDF is also machine-readable so that we can build applications that leverage it and the power of linked data (see below) to make the use and discovery of data much easier and consistent across platforms. RDF is also expressible in various serializations, such as RDF/XML, for better machine consumption. The Turtle serialization is a subset of the N3 serialization and is meant for ease of human reading and writing. All RDF examples throughout this document are written in N3 syntax for ease of reading.

The Resource Description Framework Schema (RDFS) provides a way to construct RDF vocabularies by allowing one to define RDF classes, properties, and relations tailored to one's needs. One such RDFS vocabulary used in the research reported here is FOAF, which stands for Friend Of A Friend. Another such vocabulary is ACL, or the Access Control Language. Both vocabularies are discussed below.

Linked Data

Linked-data is a set of best practices regarding the publication and connection of structured data on the Web (Broekstra, Kampman & Harmelen, 2002). Linked-data aims to provide a consistent and uniform way for agents on the web to consume and publish data, machine and human alike. The key to this is that data is published in a consistent format and that all data links to other appropriate data so that the discovery and consumption of further data is easy and encouraged. The main format in which linked-data is published in RDF.

FOAF

FOAF is a powerful RDF vocabulary that allows for the concise description of people and their social networks in a semantic and machine-readable manner (OpenSSL, 2007), It allows us to compute over the relationships and programmatically make conclusions about a person and those to which they are connected. The conventional namespace prefix for FOAF is `foaf:`.

FOAF is, of course, applicable to more than just people and their social networks. It is applicable to any agent on the web. An agent could be a person or it could be a software agent that acts on the behalf of a person or organization. FOAF is also useful for representing the subtle status of a group of agents. We leverage the semantics of the `foaf:Group` class in order not only to allow a group to act as a collection of agents but also for the group itself to be treated as, as well as act as, an individual within an application or service. This is the crux of allowing for authentication of a group's agents based on the group asserting its identity as an individual.

ACL

The Access Control Language (ACL) is a simple RDF vocabulary that allows the specification of access control rules to be tied to resources on the web and presented in a machine-readable manner. It allows for a fairly flexible and broad range of rules to be specified regarding a resource. It also allows for the rules to be specified and determined in a decentralized manner. A resource can have its access control rules easily derived from another resource's ACL file on the web. The rules can even be maintained simply via a separate service, thanks to the power of linked data. Table 1 depicts the various modes of access offered by ACL.

Table 1 – ACL access modes

Mode:	**Actions Allowed:**
Read	Read the contents (including querying, etc.)
Write	Overwrite contents (including deleting or overwriting parts of it)
Append	Add information to the end without overwriting any part
Control	Set the access control rules for the resource

This ACL-RDF vocabulary, which is defined with RDFS, includes properties of authorization (W3C, 2015). It allows the administrator to specify access control modes, which are essentially classes of operations. The WebAccessControl protocol states that the user authenticates to the server through the implementation of WebID+TLS.

However, ACL can in some cases be rather rigid, which is why we introduced a policy ontology written in OWL (web ontology language) that we developed. The main advantage of this ontology is that it can be used in tandem with a reasoner service such as Apache Jena, which can make inferences based on the rules derived from the ontology.

OWL

OWL (Antoniou & van Harmelen, 2004) extends the power of RDFS and is used to represent ontologies on the Semantic Web. "Ontology" is a term that is borrowed from philosophy referring to a conceptualization of a domain, describing entities in the world and how they relate to each other.

WebID and Its Use

WebID

WebID is a W3C proposed protocol which provides a way for an agent to be identified using standard web technologies (W3C, 2015). At its most basic level, a WebID is simply a unique URI (uniform resource identifier) that represents some resource or agent on the web. When the URI is dereferenced, an RDF or human readable format of the document that describes the resource or agent in question is returned. (Derefencing means to retrieve the resource to which the URI points.) The version of the file that is returned is determined by content negotiation and CoolURIs (Ayers & Völkel, 2008), along with fragment identifiers or 303 redirects, to determine and return the appropriate rep-

resentation of the document representing the resource that the WebID represents. Typically, HTML is served to human users and RDF to machines. The service that is implementing WebID can of course decide how this is to be handled; it could be that a human agent has reason to view the RDF representation of a resource and a machine needs to consume the HTML representation.

WebID + TLS

WebID + TLS is the protocol that describes the implementation of WebID. It has the following general steps (W3C, 2015):

- Initially, the client must establish a TLS connection, which the server authenticates itself, using the standard TLS protocol.
- Next, the client on a specific resource can perform an HTTP GET, PUT, POST, or DELETE action.
- At this stage, the guard, or server agent, can grant or deny access according to the access control rules.
- Now, if the resource requires authentication for access, the client's identity is linked to a private key and public key pair. This pair should be embedded in the client's certificate to be verified. The client has the option to automate which certificate to send from its browser when the TLS agent makes a certificate
- The TLS agent matches the certificate's public key with the public key in the user's profile document on the server.
- The Verification Agent, defined in (W3C, 2015), verifies that the WebID in the WebID certificate knows the given public key.
- The Verification Agent extracts the public key and the URIs from the *Subject Alternative Name* property field in the certificate.
- With the WebID collected, the guard can check if one of the URIs is authorized by the access control rules. If access is granted, then the guard can pass on the request to the protected resource.

The Group Problem

There are several problems that this work addresses, including group access control using WebID as well as strengthening access control on the web by coupling the policy ontology we have developed with a Jena-based reasoner to help decentralize the security logic of determining access to resources on the web. We also present a solution to one of the main security flaws of WebID by creating a system that couples a user's biometric data with their WebID to create a two-step authentication process.

We start by describing the problem surrounding group access control on the web. As it stands, delegating access to agents on the web is a process that relies heavily on static access control lists and security logic that is heavily integrated into the application logic of the service. Furthermore, we have a natural notion of trust in a group's head member, often allowing them to assert their identity and subsequently allowing for the knowledge to validate the identity of the group's other members. Capturing this in a computational environment is a challenge, but we have several solutions. We implemented a system that can allow for group owners to authenticate themselves as well as establish the trust needed to more easily authenticate the group's members. This is primarily accomplished by making use of the FOAF term **`foaf:Group`** and the subtlety that allows a **`foaf:Group`** also to be an individual agent. Also, one can specify a class associated with a group so that elements of that class are automatically of that group and vice versa.

We now address decentralizing web access control further than was done in our initial implementation. Developing secure code is often difficult, especially when the security logic of the application is tightly knit into the implementation. In our access to resources based on semantic rule lists, decoupling some of the security functionality from the main application logic allows for the application logic to be developed and maintained much more easily. We remove the rigidity of our original solution to authenticating groups and expand it into a more generalized and extensible system with the use of OWL and a Jena reasoning service that can also easily apply to individual agents. The OWL policy ontology that has been developed in our lab is also more robust and flexible than ACL as a means of expressing access control rules.

Our Solution to the Group Problem

To solve the problems of group access control, we began experimenting with the notion of authenticating a group of agents by dereferencing the WebIDs of the group's members, which are presented in the group's profile document. In order for this to work, however, we must make certain that a group is explicitly defined to avoid ambiguities in discerning whether or not the agent that is asserting its identity is indeed a group. This is where we make use of the **`foaf:Group`** term. Figure 1 is a simple example of a **`foaf:Group`** defined in RDF using N3 syntax. The profile explicitly asserts that the agent is a **`foaf:Group`**. The profile document also contains the agent's public key, as all WebID profiles must. And it contains a list of the WebIDs belonging to its member agents. This is critical to the veri-

fication process, which allows the group to assert its identity and authenticate the group members too.

```
1: @prefix foaf: <http://www.xmlns.com/foaf/spec/#> .
2: @prefix cert: <http://www.w3.org/ns/auth/cert/#> .
3:
4: _:webidcommunitygroup a foaf:Group;
5:    foaf:name “webidcommunitygroup”;
6:    foaf:member[/*group member WebID's here*/];
7:    cert:PublicKey “MIICQ...EstKg==” .
```

Figure 1 – Simple Group Profile Document

We can allow a WebID that represents a group to serve as an entry point to the authentication process. We do this by first dereferencing the WebID to get the profile document. With this we determine whether or not the agent is a group by checking if the profile contains the `foaf:Group` term. From there, we parse the WebIDs contained in the `foaf:member` list and individually verify the WebID of each member and determine the access to the resource in question based on the access rules tied to the resource. We then use server-side caching techniques to cache this access information for a limited time to the service so that, when an agent that is a member of a previously authenticated group requests access to the resource, the service may simply validate their WebID and then perform a simple lookup against the cached access data to determine what type of access should be allowed.

We expanded our initial algorithm to include use of the policy ontology and a reasoner, which was implemented using Apache Jena. This got rid of the rigidity of the algorithm and its close ties to the limited ACL vocabulary in favor of the flexibility that our OWL-based policy ontology offers. Figure 2 shows the updated version of the high-level group access algorithm, which makes use of a response returned from the reasoner.

```
1: define function grantAccess(AgentURI, ResourceURI):
2:   if (Agent's WebID not valid) then
3:     Deny access to ResourceURI, prompt resource owner that
       AgentURI
4:     has requested access to the resource.
5:     return access denied
6:   end
7:
8:   Define AProile := dereference(AgentURI)
9:   Define ResACL := dereference(ResourceURI)
10:  Define AGraph := toGraph(AProfile)
11:  Define ResACLGraph := toGraph(ResACL)
12:
13:  if (AgentURI is in ResACLGraph OR
14:      Agraph.class is in ResACLGraph) then
15:    return access type specified in ResACL
16:  end
17:  else
18:    Deny access to ResourceURI, prompt resource owner
19:    that AgentURI has requested access to the resource.
20:    return access denied
21:  end
22:end
23:
24:forEach (Member m in Group g) do
25:  accessCache.add(m, grantAccess(m, Resource.URI))
26:end
```

Figure 2 – Group Access Delegation Algorithm

Defining and Enforcing Policies

OWL-based Policy Ontology

The major advantage of using XML to represent policies is that it is straightforwardly extensible (Bradshaw, Uszok and Montanari, 2014), but the problem with XML, as with many other representations, is that its semantics is mostly implicit. One problem with implicit semantics is that the conventions on which it is based are potentially ambiguous. For that reason, semantic technologies are being used. There are many advantages to using OWL to represent policies. Ontologies allow for policies to be easily extended by simply adding new concepts.

Using ontologies to describe policies enables the system to use concepts that describe the entities and environments that are being controlled. This simplifies their descriptions and improves the system’s analyzability. The result is that policy frameworks can take advantage of powerful features such as policy conflict detection and harmonization. As with databases, there is the possibility of accessing the information provided by querying the ontology based on the ontology schema.

The KAoS policy service framework (Uszok, Bradshaw & Jeffers, 2004) is a mature general-purpose policy management system developed by the Florida Institute for Human and Machine Cognition (IHMC). It makes use of OWL and OWL-based tools for specification, analysis and enforcement of policies that are constrained across a variety of distributed computing platforms. KAoS has support for obligations and authorization policies as well as mechanisms for other kinds of policies. This enables analysis, management, specification, and enforcement of policies. Use of OWL-based policies allows term and policy federation. We are importing the KAoS policies and extending it in order to create our own policies.

Jena/Reasoning

Jena is a Java-based Semantic Web framework (McBride, 2002). It can read all RDF serializations and add data to RDF graphs, and it can write RDF graphs to all common serializations. A number of inference engines and reasoners can be used in Jena. With these engines, additional RDF assertions can be derived from an original RDF

document. Languages such as RDFS and, for possibly sophisticated ontologies, OWL are supported; additional facts can be inferred from instance data using these languages. SWRL rules (Horrocks et al., 2004) are supported if one installs the Pellet reasoner (Parsia, 2004). Pellet is an open-source Java-based OWL 2 reasoner. One of the main points of this framework is that it is intentionally general so as to be applicable for virtually any problem.

Apache Thrift

Thrift is an interface definition language and binary communication protocol that is used to define and create services for numerous languages. It is used as a remote procedure call (RPC) framework and was developed at Facebook for "scalable cross-language services development" (Apache, 2007). It is incorporated into our code through the use of RDF and triples. The RDF file information goes to the server, which infers triples and then returns more triples. Whatever access the person has will be inferred by the reasoner. The Thrift service uses Pellet for reasoning about our policies.

Extension Using Biometrics

Biometrics

Biometrics is the study of identifying an individual based off of their physical characteristics (Jain and Ross 2008). It is useful in many authentication systems due to its inherent advantages over other types of authentication. Unlike a knowledge-based system (Zhang and Li 2011) that uses a password or some other key phrase to grant access, a biometric system uses an individual's biometric, such as their face, iris or fingerprint. These biometrics cannot be as easily misplaced or stolen as a password can. A token-based systems (Hao & Yu, 2010) use some physical token such as a key or smart card to grant access. However, tokens can also be misplaced or stolen.

A biometric system is composed of some sensor, such as a camera or scanner, a module that extracts features from what was taken from the sensor, a module that stores the extracted features in a database, and a module to compare extracted features to enrolled features (Jain and Ross 2008).

WebID + Biometric protocol

This section describes the WebID+Biometrics protocol, which is built off the WebID protocol but uses the disposable feature extractors created by the GEFE technique developed by Shelton et al. (2012) for biometric verification. WebID+Biometrics also uses OWL-based policies, as described above. One possible scenario where this protocol might be used is where a person tries to login to a social network service. For example, William is a user of Joseph's social networking site. Figure 3 shows the sequence diagram for the WebID+Biometrics protocol for this scenario. The scenario assumes authentication for a RESTful web service. A RESTful web service communicates using the HTTP protocol in a way similar to (but more involved than) the way a web browser gets a web page from a server.

William's client sends a TLS request to Joseph's social networking service. Then Joseph's social networking service requests a certificate from William's client. William's client sends a certificate to Joseph's social networking service and dereferences William's profile from William's server from the URI that was provided in William's certificate. Joseph's social networking service then queries the RDF that is William's profile and checks whether or not the modulus and exponent from the profile on William's server match the modulus and exponent from William's certificate. If they match, then Joseph's service will send the RDF for the particular feature extractor that has been generated at random. William's client will ask William to submit, for example, a picture from which it will extract the feature vector and serialize it to an RDF serialization. That RDF will be sent to Joseph's service, which will dereference a document on William's server that contains William's biometric information. Joseph's service will then verify that William's biometrics matches the enrolled feature vectors. Finally, if the biometrics submitted to Joseph's social networking service matches the biometrics from the RDF document on William's server, then Joseph's service will notify William's client that he may access it.

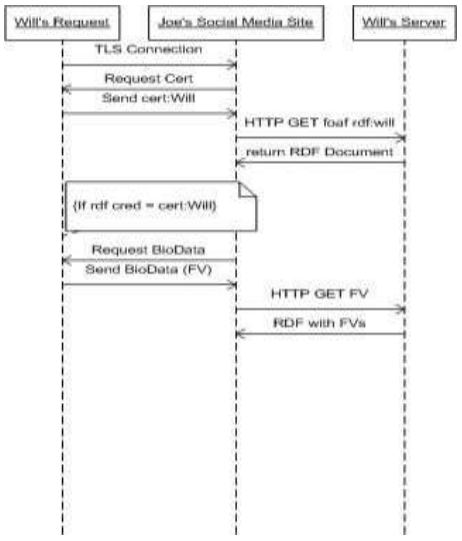

Figure 3- The sequence diagram for the WebID+Biometrics Protocol

Unlike the WebID protocol, we use policies, which are expressed in OWL. This allows us to infer roles and to infer how access is delegated. To illustrate this, consider another example, where a user, Renee, is trying to access a resource controlled by William. Renee is part of the same organization as William, and William has also delegated access to a resource to Renee. Renee's client requests to login and access a resource controlled by William. The service requests Renee's certificate and gets the RDF profile document from Renee's server from the URI in the certificate. The service checks whether the modulus and exponent in the certificate match the modulus and exponent in the profile document. The Service requests Renee's client to submit her biometric credentials using a particular disposable FE. If the biometrics for Renee matches the biometrics on Renee's server, then the server will pass Renee's policy document to a reasoning service. The reasoning service will reason about the document based on a policy ontology. The result will determine whether Renee will gain access to the system.

Like the WebID protocol, our protocol allows for profiles to be decentralized. Unlike the WebID protocol, however, our protocol is able to handle biometric verification. This allows users to bring their own biometrics. Unlike the WebID protocol, our protocol reasons about permissions using policy documents expressed in OWL.

Implementation

Node.js was used to build a client application from which agents may try to authenticate for access to resources that have access rules expressed using the OWL based policy ontology. The client also acts as the WebID authentication service. We paired the Node.js client with an Apache Jena based reasoner service, which accepts the access rules and makes inferences based on them, returning the type of access to a resource that is to be granted to the client. The reasoner used is Pellet, and Apache Thrift provides RESTful access to it.

Conclusion

We addressed the way group access control and group based authentication might be handled on the web and created a high-level algorithm for authenticating a group and its members and caching the access decisions to the server for a limited time. This allows for a more natural notion of a group to exist and interact with web services, just as in real life a group owner's identity is validated and then trust is established and propagated to the group's members. We also improved the security of the WebID protocol by coupling it with biometrics. This mitigates the threat of somebody having access to a device in which an agent's WebID certificate is installed. We also add support for OWL-based policies that may be federated and result in flexible access control. What is reported here leads to future work aimed at the notion of a network of trust.

References

Antoniou, G., & van Harmelen, F. (2004). Web ontology language: Owl. In *Handbook on ontologies* (pp. 67-92). Springer Berlin Heidelberg.

Apache Software Foundation. *Apache Thrift*. 2007.http://thrift.apache.org/.

Ayers, D. and Völkel, M. 2008. *Cool URIs for the Semantic Web*. Cool URIs for the Semantic Web.

http://www.w3.org/TR/cooluris/.

Bizer, C., Heath, T., and Berners-Lee, T. 2009. *Linked Data—The Story So Far*. International Journal on Semantic Web and Information Systems. pp. 1–22.

Bradshaw, J.M., A. Uszok, and R. Montanari. 2014. *Policy-Based Governance of Complex Distributed Systems: What Past Trends Can Teach Us about Future Requirements*. Adaptive, Dynamic, and Resilient Systems, CRC Press.

Broekstra, J., Kampman, A. and Harmelen, F. V. 2002. *Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema*. Proceedings of the First International Semantic Web Conference on the Semantic Web. Springer-Verlag, London, UK, UK, 54-68.

Emercoin. 2015. *EMCSSL. Decentralized identity management, passwordless logins, and client SSL certificates using Emercoin NVS*. Emercoin International Development Group.

Hao, Z., & Yu, N. 2010. *A Security Enhanced Remote Password Authentication Scheme Using Smart Card*. 2010 Second International Symposium on Data, Privacy, and E-Commerce, 56-60.

Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean. M. 2004. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*, W3C Member Submission, W3C. http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/.

Jain, A. K., Ross, A. 2008. *Introduction to Biometrics*. Handbook of Biometrics. Springer. pp. 1–22.

McBride, B. 2002. *Jena: A Semantic Web Toolkit*. IEEE Internet Computing 6(6): 55-59 (2002)

OpenSSL Software Foundation. *OpenSSL Project*. The OpenSSL Project Homepage, 2007.

< https://www.openssl.org/>

Parsia, B., Sirin, E. 2004. *Pellet: an OWL DL reasoned*. Proc. DL-2004, 2004.

Shelton, J., Bryant, K., Abrams, S., Small, L., Adams, J., Leflore, D., & Dozier, G. 2012. *Genetic & Evolutionary Biometric Securi-*

ty: Disposable Feature Extractors for Mitigating Biometric Replay Attacks. Procedia Computer Science, 8, 351-360.

Uszok, A., Bradshaw, J. M., & Jeffers, R. 2004. *KAoS: A policy and domain services framework for grid computing and semantic web services*. Trust management: Second international conference, iTrust 2004, Oxford, UK, March 29–April 1, 2004. Proceedings. Vol. 2995 of Lecture Notes in Computer Science., (pp. 16–26). Springer-Verlag, Berlin, Heidelberg

W3C. ed. 2015. *WebID-TLS*. Authentification over TLS.

Wang, R,. Chen, S., Wang, X. 2012. *Signing me onto your accounts through Facebook and Google: A traffic-guided security study of commercially deployed single-sign-on web services*. IEEE Symposium on Security and Privacy.

Zhang, H., & Li, M. 2011. *Security vulnerabilities of an remote password authentication scheme with smart card*. 2011 International Conference on Consumer Electronics, Communications and Networks (CECNet), 698-701.