

# PrplAc: Attribute-based Access Control in PRPL for Fine Grained Information Sharing using Semantic Web

---

*Team: Greg Bayer, Debangsu Sengupta, Tom Wang, Antone Vogt*

## **Abstract**

PrplAc provides opportunity for an empirical study on how to provide secure, fine-grained, attribute-based access control for a personal distributed file system (eg. PRPL [1]). The system provides an intuitive interface to hide complexity and express user's sharing choices as access control rules represented as links between attributes in an OWL ontology. PrplAc utilizes semantic information harvested from a content owner's data stored in various services and applications on the Internet. A reasoner performs inference on the ontology to match incoming user requests for resources against existing rules in order to grant access. Since semantic attributes are compatible across data from diverse systems, expressing access control rules this way allows a virtual data system such as PRPL to provide fine-grained access control without exposing the user to the complexity of the underlying systems. A discussion of the approach and prototype implementation is presented along with future work.

## **1. Introduction**

The PrplAc project is motivated by the idea that content owners desire to share their information using rich, yet easy-to-use mechanisms. The target user of PrplAc has information stored in a variety of servers, services and applications. For example, her personal e-mail might be stored in Gmail, school e-mail in Stanford's IMAP/webmail servers, and her work e-mail in a corporate Exchange server accessible via IMAP. Similarly, she maintains personal and work calendars and task lists in Google and Exchange/Outlook calendar. Further, she participates in one or more social networks such as Facebook and LinkedIn.

PRPL (Public/Private Data System) provides a personal distributed file system that allows a content owner to get a unified view of her data across services. This system is virtual – it provides a layer of indirection to otherwise distributed and potentially diverse data sources. Content harvesters periodically fetch information from various services to create a unified index of the owner's data, which is stored in her personal index server. The unencrypted index, protected by perimeter security, redirects the content requesters to one or more data servers.

A typical problem is to provide secure, fine-grained sharing capabilities to the data. Current sharing mechanisms are too coarse-grained in their settings. Typically, they limit access to no one, everyone, or only friends.

The project hypothesis is that it is possible to utilize semantic data (tags) present in the user's harvested data to suggest access control choices to the content owner. The owner is presented an easy-to-use UI where she establishes links among various categories of her data and people to express sharing choices. For example, vacation photos should be shared with family and friends but not colleagues. These links are used to create rules that are applied when members of her network try to access her data via PRPL. Once she has built a library of such rules, she can query to see what her rules actually map to. For instance, she will be able to verify at a glance which groups of people can actually view a specific photo, or which of her photos her family members (e.g., grandma) may access. This provides transparency that is essential in any access control system where the user is not required to reason about the underlying implementation. An OWL ontology is used to store the content owner's rules.

When an incoming user request comes in for the content owner's index or a specific resource, a reasoner performs inference to match the requester's attributes and the resource attributes with the rules in the system. If a match is found, the requester is redirected to the appropriate resource.

## **1.1 Motivating Scenarios**

PrplAc is designed to facilitate information sharing among individuals in a variety of domains. For example, Debangsu and Greg may be students in different departments at a research university. Tom and Antone may be employees in different departments of a company. After meeting at a research conference, they exchange ideas and add each other to their social networks (e.g., LinkedIn). Eventually, they agree to collaborate on a project. It is likely that that the above group utilize a diverse range of systems within and outside their organizations to manage emails, calendars, task lists, photos, documents, and other types of content. Controlled information sharing must be facilitated across a variety of applications and services. For example, they would like to exchange a diverse range of content, such as project-related documents (e.g. specifications, design documents, project timeline sheets stored in a project management application) and otherwise (e.g. photos from their conference).

Another goal is to recognize and utilize context in information sharing scenarios. For example, the project members above exchange numerous emails, attachments and organize multiple meetings to share their ideas. Upon harvesting Debangsu's emails and calendar data, prplAc's group generation system infers the data and suggests the formation of an ad-hoc coalition whose members will have access to the shared group content. The system also suggests resources that should be considered common group content (e.g., shared emails, attachments, project documents, upcoming meeting requests, and photos).

## **1.2 Problem Statement**

The Semantic Web and the PRPL project present a new domain for which to provide access control. The Semantic Web provides potentially rich data, but this data is usually not formatted for access control purposes.

This project is further motivated by the need to provide fine grained access control. A simple example, such as a user who would like to share some documents with industry, photos with friends, and project files with the Stanford community, clearly shows that coarse-grain sharing, such as sharing with everyone, friends, no one, places severe limits on data control.

Finally, in building a system to address data stored in diverse systems across the web, data sources with incompatible access control systems are the norm, not the exception.

PrplAc aims to build a secure, easy-to-use attribute-based access control system for fine-grained information sharing. It allows content owners from diverse organizations to build access control rules intuitively to facilitate easy sharing of content (document, pictures etc.) among each other. Such information is expected to be hosted across a wide range of applications, services, and locations (e.g. Gmail, Google Calendar, Facebook, LinkedIn, Exchange mail servers, Outlook calendars, etc).

In general, every access control system must strike a balance between security and functionality, such as ease of sharing. PrplAc aims to strengthen security by providing arbitrarily fine-grained control to the user and by maintaining transparency into any system abstracted access control decisions. It attempts to increase ease of sharing by allowing data owners (the access control users) to create sharing rules based on the most intuitive choices available (the semantic attributes of their data) and by allowing sharing rules to be created at every level of the attribute hierarchy, supporting coarse-grained sharing, if desired by the user.

### **1.3 Purpose**

The purpose of the study is to discover whether a content owner's data spread across multiple service providers on the Internet provides sufficient semantic context to perform reasoning and access control decisions. The study seeks to explore a new access control approach based on this semantic information and implement the proposed ideas on an open, virtual data system developed at Stanford called PRPL.

### **1.4 Significance of Study**

The study is important to researchers in the field of data portability. Today, a large amount of one's personal and professional data is distributed by applications running on proprietary, third-party systems. These systems were built with divergent goals in mind, and a common characteristic is the limited amount of interoperability and sharing among them. The PRPL system aims to provide a unified view of one's data across various systems like email servers, calendar servers, and social networks. Harvesters to the various content sites fetch data that is indexed on the content owner's indexed server and stored on data servers.

With wide scope of content ranging from proprietary work data to highly personal information, basic access control options that include sharing with no one, friends-only, and everyone are simply too coarse-grained. Typically, proprietary work data should only be shared with people on the project, while highly personal information would only be shared with family and close friends.

Thus, a fine-grained access control system capable of operating on data from diverse sources is needed in order to provide the expressive power demanded by end users. An intuitive user interface that provides sharing suggestions and relevant feedback to the content owner about the impact of her access control policies, aims to maximize system usability.

The study benefits researchers and practitioners interested in sharing data with subsets of their social network. When sharing new data (e.g., photos), a semantic-based access control system can intelligently apply existing rules to the new data. When one's social network expands, the system can use existing rules and past preferences to suggest additional sharing options to the data owner. Further, collaborating teams across different companies and organizations can use this system to facilitate federation scenarios.

## **1.5 Research questions / hypotheses**

As part of this study, the researchers explored the state of the art in access control in open, distributed systems. They identify an appropriate access control system for a personal distributed file system like PRPL. The hypothesis is that an attribute-based access control system that utilizes tags from the semantic web (user's social networks, etc.), coupled with an intuitive user interface can produce a useful system with adequate features without sacrificing usability.

## **2. Background**

### **2.1 Related Work**

#### **2.1.1 Topic**

The Semantic Web [2] and its associated technologies attempt to create semi-structured, machine-readable content out of the rich, unstructured resources that are the current world wide web. RDF/XML [3] and RDFS provide a rich way to express resource hierarchies and attributes. Further, using OWL provides a natural way to establish ontologies and infer based on them. Toolkits like Jena [4] and query languages like SPARQL [5] provide effective technologies to explore semi-structured information. SPARQL allows one to query RDF graphs (models in Jena terminology), and make inferences.

Karger et al. [6] described the Haystack system where users build adaptive views by linking various aspects of their semantic metadata. While it successfully demonstrates the power of linking resources from different sources, it is possible to become "lost" in the system. Further, the users may make destructive changes to their objects as traditional file hierarchies get blurred. The PrplAc researchers believe that providing context specific views with loose bounds is preferred to a universally adaptable system. Context drives most user tasks and is helpful to make some guiding decisions. The paper mentions that access control in this space is an open question. It highlights the challenges of dealing with semi-structured data, lack of object bounds, and the need to reasonably ensure users are not accidentally publishing private information.

Warner et al. [7] described an automatic approach to assign roles to new members of dynamic, ad-hoc coalitions. Mining attributes of members of existing roles, the system automatically match these attributes with the attributes of new members to assign their roles. The focus of this paper is to enable collaboration among teams from organizations with mismatched legacy RBAC systems. Their premise is that while the RBAC systems may be mismatched, the semantics drawn from the hierarchies may match up better across organizations. PrplAc does not presume the presence of static hierarchical information among systems. Instead, it uses a standard access control ontology for people and resources as a starting point. Then, harvesters can extend this ontology with arbitrary attributes, mined from the semantic context of the information they are harvesting.

### **2.1.2 Methodology**

Gray [8] describes the building of an access control system based on an OWL ontology upon which he implements of a prototype. He also built a promising REST-based web service called Quaestor, which can utilize OWL ontologies built by users to provide access control decisions, based on class subsumption reasoning, in simple cases. The study does not leverage semantic information.

Dersingh et al. [9] present an approach for leveraging semantic web information to represent context aware policies. Such policies can be used in federation scenarios among organizations. They demonstrate the model using the context of a SIP-based call. Toninelli et al. [10] develop an OWL based model for driving a semantic context aware policy model that is useful for determining policy decisions in collaborative wireless environments.

Priebe et al. [11] demonstrate the idea of using ontologies for access control by extending the XACML standard [12]. Inferences on the Owl ontology is used to determine group membership and add additional attributes to the requester's XACML request body. The researchers, on the other hand, aim to model complex rules and dynamically add to the rules ontology. PrplAc also does not depend on XACML for modeling access requests but may certainly support that in the future.

Cao et al. [13] present a prototype that demonstrates the potential of an access control system that attempts to capture content owner's intentions and execute them. The prototype asks the users which of the pre-defined tasks they want to execute, shows the different approaches and the side effects on other principals the approaches would have, and then executes the users' choice of approach. Cao et al built their system on top of a legacy system, but we will build a brand new system that is transparent to the user – we will make sure users know the exact effects of their actions.

Mori et al. [14] develop a plugin for a community Wiki site to demonstate the application of social network analysis as means to richer ontology for access control. The plugin extracts social network information from the Web, E-mail, social networks, and physical sensors. Then the plugin lets users attach access control list to content. While Mori et al utilizes solely external relationships in its ontology, PrplAC's attributes will also include relationships to the user.

## 2.2 Attribute-based Access Control

The idea of attribute based Access Control becomes important when considering large, diverse, loosely coupled systems on the Internet. In such an environment, traditional hierarchical or role-based access control systems (RBAC) are inappropriate. Such systems depend on common, previously defined hierarchies or roles, which are used to define permissions between people and resources. Today's large systems (in our context, various social networking, web-mail, and other online data sources) have been independently designed with divergent goals. As a result, it is unlikely that they will have complementary roles defined.

Attribute-based access control attempts to solve the problem by instead relying on attributes to perform access control decisions. Because these attributes are based on the semantic meaning of the data, they are much more likely to match up across diverse systems. For example: the semantic concept of "Photos" is likely to be the same across most systems such as Flickr, Picasa, Facebook, etc.

## 2.3 Definitions of terms

*PRPL (Private/Public):* A semantic-web based virtual data system.

*Harvester:* A module plugged into the PRPL system to collect data from an external source. A harvester uses its knowledge of the data it is collecting to insert tags and other semantic attributes that may or may not be relevant for access control into the owner's PRPL index.

*Tags / metadata:* Resources (files, photos, music) stored at different service providers are associated with various tags and metadata that describe the resources. Examples include author, description, date/time taken, keywords, people in picture etc. In this paper, tags/metadata refer to raw information harvested from various sources that may or may not be meaningful for storing rules or inferring access control decisions.

*Attributes:* In the context of PrplAc, people and resources are marked with attributes. These attributes provide information that is meaningful to the access control decision. For example, a person's group membership is described as an attribute. Attributes subsume the term group as used in other access control frameworks (eg. RBAC). Typically, elements belonging to the access control ontology are referred to as attributes. Inference performed for the access control task will depend on attributes, rather than the raw tags harvested from content. These attributes are defined by harvesters, since they know the context of the raw data they are harvesting and can decide which semantic concepts are relevant for access control.

*Access control ontology (AC Ontology):* Provides the structure of attributes used to create and enforce access control rules. It consists of a basic access control ontology defining the starting point for resource and person attributes. This basic ontology is then extended by each harvester to create the attributes for each harvester's data. Together, these ontologies form the access control ontology for a data owner's index in the PRPL system. This overall access control ontology will be dynamically modified by harvesters as the owner's data sources change and may also be modified

based on owner preferences and system suggestions. Future improvements may include: The AC ontology's resource and person trees may be constrained to contain groups/attributes that are a subset of publicly available / standard people and resource ontologies (e.g. FOAF [15] and Nepomuk [16] ontologies respectively). Harvester ontologies would contain a subset from the aforementioned standard people and resource ontologies, i.e., the groups/attributes relevant to the harvester. Translation module would map harvester attributes to AC attributes and import accordingly. Since both sets come from the same standard ontology, this translation should be feasible.

### **3. Methodology**

The purpose of the study is to identify an appropriate access control system for PRPL, implement a prototype using the approach, and discuss its success and failures. This section presents a short end-to-end overview of the system and describes the security models, underlying PRPL platform, and PrplAc architecture. Specifics about the prototype implementation are covered in section 4.

#### **3.1 Basic paradigm**

The content owner maintains an index server and a data server that store the index of her data across the Internet and the actual data itself, respectively. She shares her credentials with PRPL and allows content harvesters to fetch and/or index the data present at multiple providers like social networks, Gmail, Plaxo, etc. She uses the PrplAc sharing view to retrieve the access control ontology. It provides hierarchical views of her resources (e.g., files), people (social network), and the attributes associated with them. She draws connections between files and persons at potentially different levels to form links and express share/don't share decisions. These decisions are persisted to the index. Over time, the harvesters will index more data and present more specific sharing choices to the content owner. When a requester wants to access a specific resource, inference engines run on the ontology to gather his membership attributes and tries to match with the resource's attributes. If a successful match is found, the requester is allowed to access the resource.

#### **3.2 Security Models**

PrplAc proposes three security models that meets most user needs. The first model is the white list (or closed) model. The default state of this model is that no one gets access to the data except for the specific groups or individuals that are granted access to it.

The second model is referred to as black list (or open model), where the default state allows everyone in the content owner's network to get access to the data. The owner can then tweak the model by specifically black listing select groups or individuals.

The third model is called the smart (or adaptive) security. The idea here is that the system will utilize data mining and machine learning techniques to periodically suggest groups that the user belongs to. It uses this information to identify content that is relevant to the above groups. These group and link suggestions are presented to the content owner for approval or further tuning. This model runs

periodically as the system harvests newer data from the owner's activities in the different networks.

### 3.3 Platform

The PrplAc implementation runs on top of the PRPL (Public-Private) Data System that provides a personal distributed file system for sharing one's content. It provides a unified view of data across multiple service providers and makes use of the semi-structured data on the semantic web [1]. The system implements a distributed content indexing and sharing system using the Jena framework and Semantic Web technologies like OWL, RDF, RDFS, etc. The PRPL system provides an excellent platform to gather attributes available from the content owner's personal Semantic Web from which to draw inferences and build an access control system.

### 3.4 Architecture

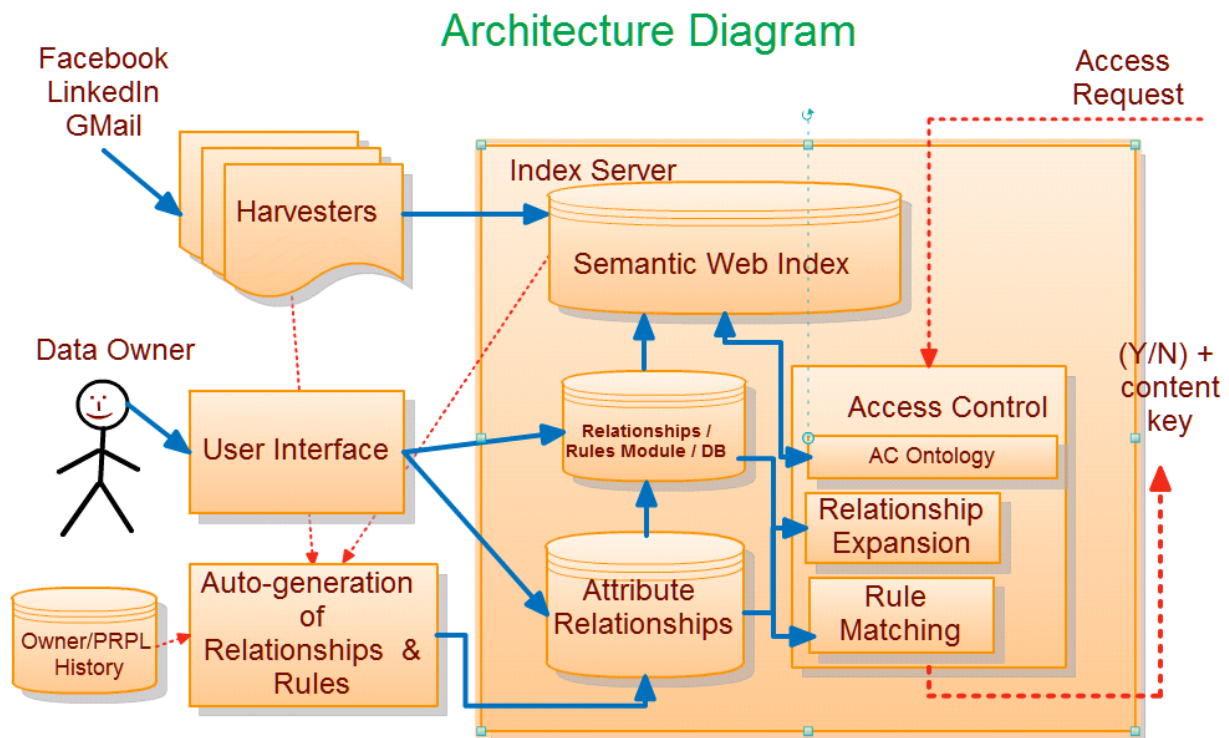


Figure 1 PrplAc architecture. Depicts a) Content/data owner's data being harvested in her index. b) Data owner specifying rules via the sharing view interface, c) Rules persisting to index, d) Auto rules generation, e) Content requester access request matching.

Figure 1 depicts the overall architecture of PrplAc and how it integrates with the PRPL data system. There are several interesting aspects of the architecture.

### 3.4.1 Access Control Ontology

The system relies on a common access control ontology that is used to link persons (subjects/principals) and resources (files, URI's etc). The ontology is based on OWL with an RDF/XML backend for extensibility. The purpose is to identify the main semantic attributes that are relevant for defining access control rules.

Our prototype has built a simple ontology to describe resources and people. It is designed to be extended by harvesters and in the future may incorporate standard ontologies, including the popular FOAF ontology to richly describe people [15]. The idea is that general ontologies can be extended to add access control specific classes and attributes for improved manual access control or to support more sophisticated inferences. A critical aspect of the ontology design is that as a data owner adds new harvesters to include new document types, people, etc., the ontology can be easily and dynamically extended.

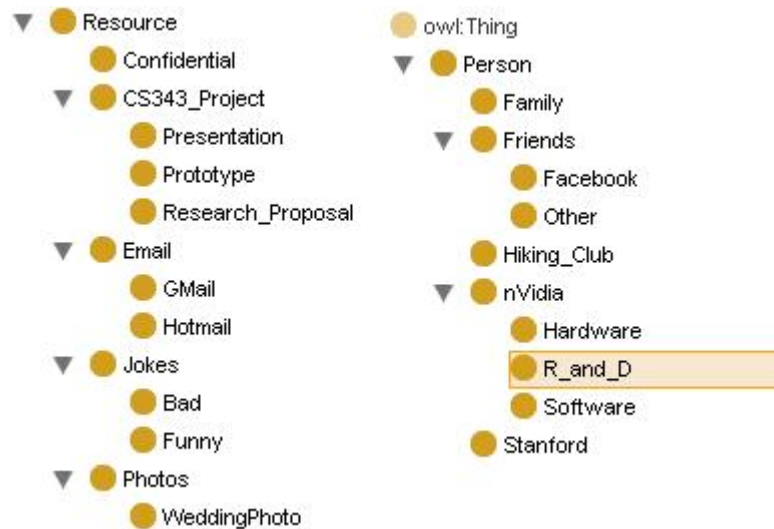


Figure 2 Access Control ontology linking Resource to Persons

### 3.4.2 Content Harvesting

Content harvesting involves reading data from the content/data owner's public and private services and applications. Typically, the user will share credentials with the PRPL system, and special purpose harvesters will gather relevant data from services like Gmail, Facebook, LinkedIn and other sources. A harvester uses its knowledge of the data it is collecting to insert tags and other semantic attributes that may or may not be relevant for access control into the owner's PRPL index. A combination of XML feeds and screen scraping is used to gather the data. The combination is suggested to gather necessary data that is, perhaps, not part of the service provider's XML feed. The data is stored in the user's PRPL data server in encrypted form, while the index is stored unencrypted in the index server.

The index server is protected by perimeter security, which is provided by PrplAc. This module may also invoke the auto relationship and generation component for access control purposes (left to future work).

### **3.4.3 Auto Relationship and Rules Generation:**

Statically updating the access control ontology with attributes is not scalable. For example, the researchers envision that each content owner will have a different set of requirements about the type of attributes and granularity at which they wish to control access. For this reason, it is critical that the owner's access control ontology can dynamically update to include new attributes based on her interactions in her social network. For example, the user may join a new social group, or enter into a new collaboration (per the motivating scenarios above), and wish to control access for a subset of documents based on these rules. Also, new types of resources will certainly emerge. It is clear that a static access control ontology would prove to be deficient for these needs.

Therefore, an essential component of PrplAc's design is support for dynamic update of attributes and relationships by the harvesters that generate them. In addition, future work could improve on this concept via system-provided relationships / rules, auto generation of relationships / rules, and other mechanisms to augment the ability of the owner to express her sharing intentions.

#### **System-provided rules and relationships**

First, the basic access control ontology provides sufficient guidance for common documents types and groups for a few specific sets of users (e.g., students, professors, industry professional). Such groups will be used to suggest system-provided relationships and rules. The advantages are manifold. Firstly, the researchers hope that the system-provided rules prove to be sufficient for a large percentage of novice users and drive system usage. Secondly, the system-provided rules provide examples for the content owner to become comfortable with the system so that the content owner can then customize the relationships and rules to her needs. Thirdly, as the system grows, data mining techniques can be used to derive popular rules used in the system. Such data can be used to suggest better rules. The goal is to provide a useful system that requires minimal tweaking from the content owner should she choose not to. Implementation is left to future work.

#### **Auto-generated rules and relationships**

Auto-generation of relationships and rules is important for the system to create suggestions that are customized to the content owner's usage. One can envision this to be a custom access control ontology that is augmented from the basic one and personalized for the content owner.

Data mining and machine learning techniques are used to detect new relationships that are emerging in the content owner's social network. This module runs periodically after new content is harvested. In addition to extending the ontology to include new attributes on the Person and Resource sides each time a harvester runs, detecting new groups/coalitions based on the content owner's usage patterns supports creation of attributes of which no single harvester is aware. Also as an owner's access

control ontology grows, it may be advantageous to prune certain attributes from the user interface based on usage patterns.

Using a concrete example, the system can use the information about the content owner joining a Ski group in Facebook to create such a group on the person side of the AC ontology. Similarly, a Ski photos group can be created on the Resource side.

The next module utilizes the newly created groups to automatically generate rule suggestions for the content owner. For example, a rule can suggest photos tagged with ski keywords to be part of the Ski photos group. An additional rule can suggest linking the Ski photos resource attribute to the Ski club membership attribute.

Such choices are presented to the user for her to accept/reject or customize on a finer granularity.

Implementation is left to future work.

### **3.4.4 Rules / relationships module**

The relationships generated from harvesting the content owner's data are persisted to the AC ontology. Conceptually, one can view this ontology to be a merger of two distinct ontologies – one provided by the system/harvesters, and the other one being customized to the content owner. This module is responsible for translating simple tags present in harvested resources to attributes that are meaningful in the access control context.

The rules (approved by the content owner) are persisted to a rules module from where it is applied to the semantic index by adding the relevant attributes to express the rule. The researchers believe that as the index grows, it might be necessary to implement a separate rules database for performance and caching reasons. It is a problem faced by other similar semantic web implementations [6].

Implementation of data owner customizations is left to future work.

### **3.4.5 User Interface**

The PrplAC user interface provides a clear display of the content owner's Resource and Person attributes. It provides a tree view of the current hierarchical structure present in the two attribute groups. The content owner gets a high level view of her data and can explore deeper in the structure to the individual person and resource levels. The attribute information is obtained from the semantic index and the AC ontology.

The PrplAc UI allows the user to specify the security model she wants to implement (i.e. white list, black list, or smart security), as described previously. The content owner is shown suggestions for sharing the data if using the smart security model. She draws links between the Subject and Object trees to specify permissions. Both accept and deny rules are allowed. While the user interface currently supports a single link per rule, the architecture can support multiple links (and links with attribute conjunctions on either end) to express more complex rules related to attributes.

The idea is that the user does not need to learn about Semantic web technology specifics (e.g. SPARQL, RDF, XML) to be able to specify her needs. The researchers feel that a very high percentage of common rules can be described in the system using more intuitive semantics such as drawing lines between attributes, and possibly a subset of easily understood attributes.

The specified rules are persisted to the access control ontology and the semantic index via the rules/relationship module described above. Simple rules can directly be persisted, while more complex rules may need to be translated in terms of system attributes, validated (for conflicts), before being added to the system.

The user interface also supports resource and subject specific views. The idea is to provide transparency into how the system is interpreting sharing rules, while abstracting away the complexity of the underlying SPARQL queries. The content owner should be able to see at a glance exactly which people have access to a specific file. Alternatively, the content owner may wish to learn exactly which files a specific individual (e.g., grandma) can view on the system.

### **3.4.6 Content request checks and matching**

The section describes the series of actions that occur in the system when a content requester wants to survey the content owner's files. The researchers assume that requester is authenticated using a policy-based encryption scheme used in PRPL [1]. Typically, the requester asks the owner's index server for resources that he has access to. The index server passes on the request to the access control module for checks and approval.

The AC module includes an OWL [17] reasoner and SPARQL query engine. It looks up the requester identity and gathers/infers the attributes relevant to the user. It performs a similar lookup on the requested resource side to gather its groups (attributes). We refer to this as relationship expansion on both the person and resource ontologies. The module performs a match for each object against the requester's attributes and returns the matching subset of the content owner's index to the requester. If no matches are found, the requester is returned an empty index.

At this point, the requester can choose a specific (group of) files that he has access to, and make a request to the PRPL index server. The index server performs a check similar to the one described above, against the specific file. It is possible that the permissions may have changed between time of the index request, and time of the resource request. If a match is found, then the index server grants access to the file. The requester is redirected to the data server with a key, and the file is retrieved using that key.

## **4. Prototype Implementation**

### **4.1 User Interface**

This section describes the prototype UI (PRPL Sharing View) implemented in PrplAC. The researchers believe that the current prototype lays the groundwork for rapidly adding new features to

the access control system. The design is that the data structures are flexible enough to incorporate new extensions easily.

#### 4.1.1 UI models

There are primarily three important models in the user interface: a) Resources tree, b) People tree, and c) Rule Links graph.

The objects and principals trees model the access control ontology as presented to the content owner. It includes the harvester-provided ontology as well as elements added as user customized ontology, i.e., the attributes are system as well as automatically generated (see above). The tree representation is adequate as a hierarchical view of resources and persons and is a paradigm that users are familiar with. It also allows for multi-level navigation that is easy to depict on screens of different sizes and resolution.

While we have chosen a tree representation, the system is not tied to this implementation. It can easily be changed to different graph views to show alternative connections, if found useful. The access control ontology in Figure 2 offers the structure represented in the trees.

The Rule Links graph represents the rules as specified by the content owner. This is done in the view by dragging connections between principals and objects. Links can be drawn between elements at any levels in either of the hierarchies. The nodes in the graph are either objects or principals. The edge represents the connection and its properties. For example, it can be used to support accept relationships (green in the view), or deny relationships (red in the view). The edge can be augmented to add more attributes as properties.

The graph structure is chosen in order to be extensible. The researchers envision that it can represent complicated connections. For instance, an principal -> object connection can be augmented by drawing connections back to the principal hierarchy with annotations. As a concrete example, one can envision a connection between “Mom” and “Photos.” A further connection can be drawn from “Photos” back to the “Family” group with edge annotated with the “takenBy” predicate. This represents a rule to share photos with mom that are taken by members of the content owner’s family. Arbitrary predicates could also be represented in other ways, such as a context menu or a separate predicate tree/graph.

Figure 3 provides an example representation of the Rule Links graph. The rule links graph uses the standardized GraphML format in order to facilitate graph exchange and usage of common tools and parsers [18].

Both the attribute trees and the rule graph are generated dynamically from their representation in the owner’s semantic index. Newly created rules are persisted back to the index.

```

<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
  http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">

  <key id="name" for="node" attr.name="name" attr.type="string"/>
  <key id="accept" for="edge" attr.name="accept" attr.type="boolean"> <default>true</default> </key>

  <graph edgedefault="undirected">
    <!-- nodes -->
    <data key="name">CS343 Project</data> </node>
    <node id="5"> <data key="name">CS343</data> </node>
    <node id="6"> <data key="name">CS343 Project</data> </node>
    <node id="7"> <data key="name">Basketball</data> </node>
    <node id="8"> <data key="name">Funny</data> </node>
    <node id="9"> <data key="name">Basketball</data> </node>

    <!-- edges -->
    <edge id="1" source="4" target="5"> <data key="accept">true</data> </edge> <!--Accept rule between nodes-->
    <edge id="2" source="6" target="7"> <data key="accept">>false</data> </edge> <!--Deny rule between nodes-->
    <edge id="3" source="8" target="9"> <data key="accept">true</data> </edge>

  </graph>
</graphml>

```

**Figure 3 Example listing of links graph showing accept and deny links.**

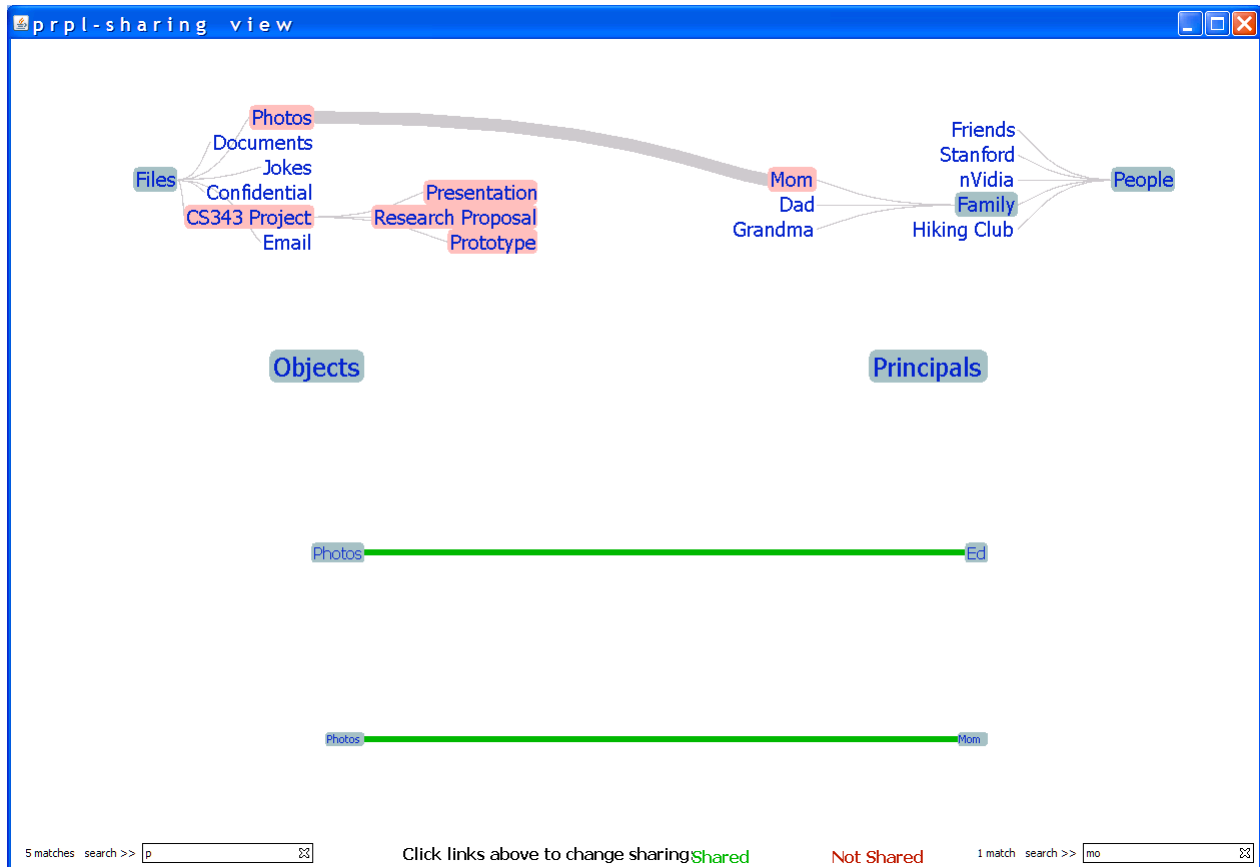


Figure 4 PRPL sharing view. Showing accept links between objects. Search and expand feature on either side.

## 4.1.2 User Interface Views

### Sharing View

As Figure 4 shows, the PRPL Sharing View is the primary mechanism for the content owner to describe her rules. The two trees rooted at Files and People correspond to the Resource and Person hierarchies. The Prefuse visualization toolkit [19] and Java AWT packages [24] are used for the prototype. The tree view expands to two levels by default and allows the user to expand the hierarchy to drill down to the desired level of detail.

As system usage increases, the researchers expect there to be lots of levels created. A search facility has been implemented in both hierarchies to allow the user to quickly find the correct group or individual file within many nested levels.

A very intuitive mechanism to create rules has been presented. The content owner simply drags a connection between resources and people to describe a rule. The legend mentions that green links represent “accept” or “shared” rules, while red represents the opposite. Figure 5 shows an example of a “not shared” rule along with a view of a typical interface with rules of different levels.

In the case of conflicting rules, simple heuristics will be used to resolve conflicts or point them out to the content owner. Some example tie-breaking heuristics can include specific rules override more general rules from the hierarchy. Alternatively, deny rules take precedence over accept rules, thereby allowing the system to be conservative in sharing.

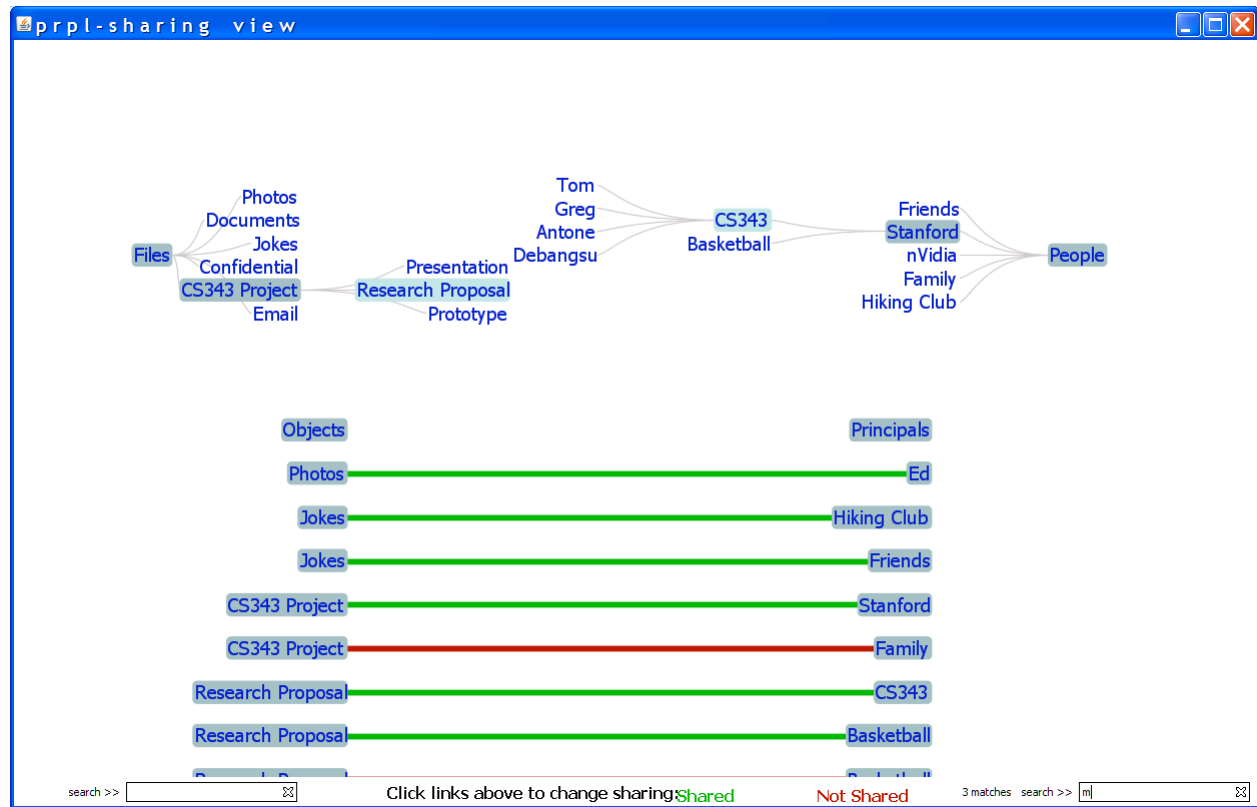


Figure 5 PRPL Sharing view depicting multiple level tree expansion, rules at different levels and accept/deny links

## Other views

With increased system usage and auto-generation, it is possible to develop an intricate set of rules for the discriminating user. Researchers envision that the tradeoff between functionality and simplicity will emerge. It is important to provide ways to hide the complexity of the design. The goal is that the content owner must be able to understand/predict what the system is doing; otherwise she will not trust it. To meet that goal, the following views are suggested.

### Resource Specific View

In this view, the content owner selects a specific file or object group and selects a “List viewers option.” The system will produce a list of all individuals and groups that are allowed to view the particular file or object group after computing all the rules. At this point, she may create additional rules specific to the file, or modify her existing rules to achieve different effects. This view is always available and helps to provide the owner with dynamic transparency into how the system is interpreting and implementing her intentions (rules).

This is currently implemented by updating the “Shared With” column in the current PRPL prototype resource view. This view is supported in the prototype access control module but is not yet implemented in the user interface module.

### **Subject Specific View**

This view essentially answers the question: “Which files can grandma view?” The content owner selects a specific group or individual and selects a “List files” option. The system generates a list of all the files and object groups that are visible to the specific users per the given rules. The content owner can then modify the rules to achieve her desired effect. As with the resource specific view, this view should always be available and in order to provided the owner with dynamic transparency into how the system is interpreting and implementing her intentions (rules).

The underlying access control modules supports this feature. Implementation in the user interface is left to future work.

## **4.2 Back end**

On the back end of the prototype is the arbiter, the software that interprets rules and answers either yes or no to a request. In effect, the arbiter searches the index for a set of rules that apply to the requester and the requested resource, then grants or denies according to those rules. To accomplish this, each rule connects two attributes, one for the requester and one for the resource. These attributes map the access control ontology into the classes and properties of the external ontology. If a match exists, then the rule is deemed applicable and added to the set. The attributes are defined such that they simplify the external ontology into a concept hierarchy, a taxonomy that may be presented to the user in a meaningful way, and provide a level of indirection enabling access control to leverage virtually any external ontology. Thus we meet our goal of access control that is fine-grained and easy to use.

Since the purpose of this paper is to discuss the larger proposal, the specific methodology of the back end implementation is out of scope. As such, the reader is referred to a separate paper [25] by the same authors.

## **5. Results**

PrplAC demonstrates attribute-based access control using attributes harvested from the Semantic Web. The few scenarios that have been implemented are: a) individual files to individual persons, b) individual files to group of persons (attribute), c) group of files (attribute) to individual persons, and d) group of files (attribute) to group of persons (attribute).

### **5.1 Current approach benefits**

This section summarizes the benefits of the current approach, as it pertains to both the content owner as well as the requester.

#### **5.1.1 Attribute based access control system using OWL**

Semantic attributes are a strong foundation on which to build an access control system for open systems like PRPL. The lack of common, static, hierarchical access control information means that system designers cannot rely on such sources to make access control decisions. An attribute-based system, backed by an OWL ontology along with relevant inference engines provide an appropriate set of tool for solving this problem. For most common cases, the problem of granting access is one of class subsumption, and the OWL tools readily provide this capability. Secondly, at least for simple (most common) rules, allowing access to a resource ultimately becomes a question of whether the requester has Type of an attribute as defined by the content owner. For more complex rules, the hasType predicate may also be specified in the rule.

#### **5.1.2 Semantic information in today's unstructured web**

Sufficient semantic information is present in today's rather unstructured web to use this information as a basis for an access control system. Typical content owners have published an enormous amount of their personal data across various services including social networks, email, calendars, task lists, photo sharing sites. The context of the sources can be unified to create a map of the content owner's personal network and her files. The researchers believe that it is sufficient to start providing access control decisions based on this data. The FOAF ontology [15] presents a rich source of data about people, while the Nepomuk File ontology [16] similarly provides a rich classification of resources. These pre-built ontologies can be used by harvesters to simplify or augment the attributes that are mined from the content owner's data to produce a customized ontology on which to perform inference.

### **5.1.3 Extending attribute based access control**

By publishing to and augmenting a harvester/system-provided access control ontology as described above, one takes the first step toward making it more useful for the user. The system becomes more useful for cross-organization collaboration when one can plug in static information to augment the model among individuals. The shared information in the form of legacy RBAC [22] systems can be plugged into our ontology to improve the performance of inference engines. Even though the implementation does not depend on it, PrplAc should be able to function alongside and benefit from RBAC and other similar access control information sources.

## **5.2 Cons of alternative approaches**

This section describes the downsides of a few alternative design approaches. It also helps understand some of the choices made by the researchers.

### **5.2.1 Coarse grained sharing**

Sharing is an important feature of many Web 2.0 systems. However, doing this well is a hard problem, and most solutions choose a simplified approach. Generally, the choices revolve around sharing with everyone, no one, or friends-only. While this is a good initial step, this level of sharing is too coarse-grained and proves to be inadequate when one tries to share in more sophisticated manner. It also makes such solutions inflexible and inappropriate for collaborative activities in dynamic, ad-hoc coalitions between members of multiple organizations. A typical use case involves an interdisciplinary project team that wants to share data with members belonging to industry, university community, and outside stakeholders, as well as a user with identities in multiple systems.

### **5.2.2 Traditional RBAC, HRBAC, ACLs**

Because of prior work, relying on traditional access control systems would greatly simplify the problem space. However, there are many problems with this approach in our domain. Firstly, as Cao et al. [13] describe, traditional systems like ACL's do not make any attempts to hide the complexity from the user. Relatively simple tasks become complicated for users who may not desire to become an expert in ACL systems. Secondly, RBAC type systems prove to be inappropriate in our case because such systems typically only exist in corporate environments and do not back open systems of the type our target users are interested in. Thirdly, as Warner et al. [7] state, it is likely that different collaborating organizations will implement the systems in non-complementary ways, which will make the mapping challenging.

For that reason, it is more intuitive to rely on attribute-based access control systems. One is able to hide the complexity by providing users, from novice to expert, the paradigm of linking groups that they are familiar with. In our system, this translates to linking resource attributes to person attributes [13].

### 5.2.3 Manually assigned attributes

Building a static attribute-based access control system will prove to be insufficient for such a dynamically changing place as the Web. Manually assigned attributes are reasonable if the system is merely a front-end for an RBAC system. Or perhaps, it is being used in a controlled domain, like within a company. However, this does not scale to the web, and our ontologies need to be dynamic. Basic dynamic ontologies can be maintained by having harvesters update attributes and their relationships after periodic harvest.

### 5.2.4 “Too” smart security

The researchers believe that the system must not go overboard in creating complex rules and abstracting them away from the content owner. The researchers agree with Cat et al’s views on the algorithmic relationship between goal and configuration [13]. The user of the system must also have the “appropriate level of confidence in the system.” A user must understand what the system is doing and be able to verify her understanding. Otherwise, she will not trust or use the system [23]. For that reason, our system aims to be transparent by letting the content owner see the effect of her rules by providing resource and subject-specific views, along with visibility of all rules in the system.

## 6. Future Work/Recommendations

There is significant opportunity for future research in this area and for extension of the PrplAc prototype:

1. Automatic relationship and rules generation: There are a lot of opportunities for applying data mining and machine learning techniques to suggest automatic creation of sophisticated “groups” . The groups can be expressed as OWL attributes or classes on either the Person (Subjects/Principals) or Resources (Objects) side of the Access Control ontology. An example is the ability to suggest formation of ad-hoc coalitions from email and calendar information. The email content and attachments become shared content for the group which will typically include at least the members in the to: and cc: lists. Such ideas may involve links between subjects, objects, along with back links to either group to suggest context. (see section 3.4.3)

Developing a set of heuristics for group creation, beyond what any one harvester knows, is the first step towards this promising area of research.

2. Integrate other projects with PRPL data system and AC:
  - a. Movie Cards [24] - secure who can see your preferences / where you are
  - b. pFS [25] - only sync files as permitted by access control
  - c. Collaborative version of iToDo [26] - The content owner enter todo items and family/friends/colleagues can see them (like Outlook calendar).

- d. Open Flow [27] - Can flow configuration be abstracted as PRPL data, controlled and indexed? Target application: user defines who can use their machine to host computation for nearby mobile devices.
3. Authentication: One can study open standards like OAuth and OpenID to identify integration opportunities.
  4. Usability studies: Comparison studies need to be performed to validate whether the fine-grained access control is meeting users' needs as envisioned.
  5. Scalability of the data owner's index: Over time, the index will become quite large with large amounts of resources and OWL/RDF data. If OWL DL is used, the extra expressivity leads to a performance tradeoff. Studies are needed to establish the limits of the system and compare them to the demands of a responsive user interface. Alternatively, caching techniques and standard databases may be needed to convert and store data in structured form. Also, performance tuning, such as in the Prefuse UI, would be useful.
  6. Common rules and queries: PrplAc UI can suggest common rules and queries for typical tasks like sharing family photos, work documents, etc. Such queries help improve the system's usability, and allow the content owner to confidently begin sharing her content using our system. The user can view the result of the rules using our UI and over time, create and edit complex rules herself.
  7. Publishing content owner's choices to their existing services: The idea is to allow the content owner to specify their access control choices even when the requester is not using a PRPL compliant client, such as visiting Facebook directly. The current approach involves the writing of a PRPL-compliant client for various devices and platforms in order to take advantage of the unified view and fine-grained data sharing. Researchers are building several lightweight clients that work on multiple desktop and mobile platforms. In addition, existing service providers might wish to reap the benefits of such a system. This can be achieved in several ways. They can build a PRPL-compliant client that works with their service using a common access control ontology.

Another way is to have the PRPL system "push" out the content owner's choices across to her service providers using XML-based feeds and services. This can be done by providing the PRPL service credentials to perform limited updates on the content owner's behalf. A service provider specific harvester ontology can be mapped to the standard access control ontology to provide a close approximation of the user's choices.

Since PRPL creates a uniform view of the content owner's data, it can be leveraged when the user decides to move her photos from Flickr to the new version of Facebook's photo sharing application. The rich set of sharing choices backed up by the current access control ontology should be leveraged to suggest a compatible set of sharing choices on the new service provider's location.

8. Other remaining work (from above)
  - a. System-provided rules and relationships (see section 3.4.1)

- b. Data owner customization of system provided rules / attribute relationships (see 3.4.4)
  - c. Subject specific view: transparency of what resources can a given user see (see section 4.1.2)
9. Other future improvements to prototype
- a. Merge code branch with latest PRPL prototype code: some RPC work, more scalable inference engine, etc.
  - b. “Deny” rules. Whitelist / Blacklist / Smart Security options.
  - c. How to express more complex rules in easy-to-use UI (arbitrary predicates, etc.)
  - d. Investigate PRPL/AC client design on mobile devices (look at using GWT, Android).

Estimated timeframe for selected future work:

1. Demonstrate more complex rules in the system and UI: 1 - 2 months
2. Automatic relationship and rule generation: 3 months
3. Publish content owner’s rules to a service external to PRPL: 1 month
4. Improve performance / scalability: 2 - 3 months
5. Authentication: 2 - 3 months
6. Usability studies: 1 – 2 months

## 7. Conclusions

PrplAc demonstrates that Semantic Web attributes can be used as a basis on which to build a successful access control system. By building on top of this layer of semantic abstraction, the PrplAc achieves interoperability across diverse systems. Leveraging the contextual knowledge of the harvesters required to populate any virtual data system such as PRPL, PrplAc provides data owners with structured attribute hierarchies that support intuitive, arbitrarily fine-grained sharing control. Security and user trust are improved by supplying transparency mechanisms whenever abstraction potentially obscures the result of user access control decisions. Much work remains to fully understand the implications of this approach to access control, as well as to continue the development of additional features.

## 8. References / Bibliography

- [1] POMI/PRPL proposal and presentations. Stanford University, 2008.
- [2] T Berners-Lee, J Hendler, O Lassila, "The Semantic Web," Scientific American, 2001, Available: [www-personal.si.umich.edu](http://www-personal.si.umich.edu).
- [3] D. Becket and B. McBride, "RDF/XML Syntax Specification (Revised): W3C Recommendation," W3C, Feb 10 2004, Available: RDF/XML <http://www.w3.org/TR/rdf-syntax-grammar/>.
- [4] "Jena: A Semantic Web Framework for Java," HP Labs Semantic Web Research, 2008, Available: <http://jena.sourceforge.net/>.
- [5] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF: W3C Working Draft," W3C, 2006, Available: <http://www.w3.org/TR/rdf-sparql-query/>.
- [6] E. Adar, D. R. Karger, and L. Stein, "Haystack: Per-User Information Environments," *Proceedings of the 1999 Conference on Information and Knowledge Management (CIKM)*, 1999.
- [7] J. Warner, V. Atluri, R. Mukkamala, J. Vaidya, "Using Semantics for Automatic Enforcement of Access Control Policies among Dynamic Coalitions," 2007, Available: <http://cimic.rutgers.edu/~jsvaidya/pub-papers/vaidya-sacmat07coll.pdf>.
- [8] N. Gray. "ACLs in OWL: practical reasoning about access," 2006, <http://www.eswc2006.org/poster-papers/FP33-Gray.pdf>.
- [9] Dersingh et al., "Using semantic policies for ad-hoc coalition access control," 2006, Available: <http://ieeexplore.ieee.org/iel5/4141733/4141734/04141777.pdf?tp=&isnumber=&arnumber=4141777>.
- [10] Toninelli et al., "A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments," 2007, Available: <http://people.csail.mit.edu/lkagal/papers/toninelli-CameraReadyFinal.pdf>.
- [11] Priebe et al., "Supporting Attribute-based Access Control with Ontologies," 2006, Available: <http://ieeexplore.ieee.org/iel5/10823/34117/01625344.pdf?tp=&isnumber=&arnumber=1625344>.
- [12] OASIS XACML, [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=xacml](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml).
- [13] Cao et al., "Intentional access management: making access control usable for end-users," 2006, Available: <http://portal.acm.org/citation.cfm?id=1143120.1143124&coll=portal&dl=ACM&type=series&idx=SERIES10714&part=series&WantType=Proceedings&title=AICPS>.

- [14] Mori et al., “Real-world Oriented Information Sharing Using Social Networks,” Available: <http://portal.acm.org/citation.cfm?id=1099203.1099215&coll=ACM&dl=GUIDE&CFID=64463517&CFTOKEN=79708528>
- [15] FOAF ontology, 2007, Available: <http://xmlns.com/foaf/spec/>.
- [16] Nepomuk File Ontology, Available: <http://www.semanticdesktop.org/ontologies/nfo/>.
- [17] OWL, 2004, Available: <http://www.w3.org/TR/owl-features/>.
- [18] GraphML, 2006, Available: <http://graphml.graphdrawing.org/>.
- [19] Prefuse Visualization Toolkit, Available: <http://prefuse.org/>.
- [20] Java AWT, Available: <http://java.sun.com/j2se/1.5.0/docs/api/java/awt/package-summary.html>.
- [21] A. Vogt, T. Wang, G. Bayer and D. Sengupta, “PRPL Access Control Back-End Methodology,” June 2008, unpublished.
- [22] D. F. Ferraiolo and D.R. Kuhn, “Role Based Access Control,” *15th National Computer Security Conference*, Oct 1992, Available: <http://csrc.nist.gov/groups/SNS/rbac/>.
- [23] S. Klemmer, Stanford University, Discussions, May 2008.
- [24] C. Brigham, S. Hangal, S. Seong, “Movie Cards Project,” Stanford University, 2008, Available: <http://suif.stanford.edu/~courses/cs343/pmwiki/pmwiki.php?n=Groups.MobileApps>.
- [25] A. Mehdi and S. Polu, “Personal Storage Cloud from Portable Components,” Stanford University, 2008, Available: <http://suif.stanford.edu/~courses/cs343/pmwiki/pmwiki.php?n=Groups.Caching>.
- [26] S. Gopalan and P. Schindler, “iTodo,” Stanford University, 2008, Available: <http://suif.stanford.edu/~courses/cs343/pmwiki/pmwiki.php?n=Groups.InternetBrowsing>.
- [27] C. Duhadway and D. Erickson, “Virtualization of Mobile Devices,” Stanford University, 2008, Available: <http://suif.stanford.edu/~courses/cs343/pmwiki/pmwiki.php?n=Groups.MobileVirtualization>

# PRPL Access Control Back-End Methodology

## CS343 Research Proposal

Antone Vogt <ubiquity@cs.stanford.edu>

Tom Wang <tsewen@cs.stanford.edu>

Greg Bayer <gbayer@cs.stanford.edu>

Debangsu Sengupta <debangsu@cs.stanford.edu>

## 1. Introduction

Our goal is to design and implement an access control methodology that is usable by nontechnical persons and permits fine-grained rules to be defined. To accomplish that task, we augmented the existing PRPL framework [1] to leverage semantic web information and created an experimental prototype. As such, we are able to interface with ontologies while making minimal assumptions about those ontologies. As a result, we have a methodology, which we call *Attribute-Based Access Control* (ABAC) capable of translating an abstract ontology into a taxonomy that is presentable to a nontechnical user.

On the back end of the prototype is the arbiter, which interprets access control rules and answers either yes or no to a request. In effect, the arbiter searches for a set of rules that apply to the requester and the requested resource, then grant or deny according to those rules. Each rule contains two attributes, one for the requester and one for the resource. These attributes map the access control ontology into the classes and properties of the external ontology. If the match exists, then the rule is deemed applicable and added to the set. The arbiter then interprets the set and either grants or denies the request.

The scope of this paper is the back end implementation of the prototype and its underlying theory. For a more general overview of the project proposal, the reader is referred to [2].

The motivating examples in Figure 1 are referenced throughout this paper.

- $x_1$ : Susan shares a particular photo that she owns, myiphone.jpg, with her friend Janet.
- $x_2$ : Susan shares all wedding photos with all of her family members.
- $x_3$ : Susan shares any email with its recipients.
- $x_4$ : Susan shares any email attachment with the recipients of that email.

Figure 1. Motivating examples.

## 2. Definitions and Theory

### 2.1. Preliminaries

There are multiple ontologies at play. When we refer to the *PRPL ontology*, we are implicitly including the access control classes presented in Figure 2 and their properties. We refer to an abstract ontology from which attributes are inferred as an *external ontology*. There may be many external ontologies. The ontologies are represented as triples, which contain a *subject*, *predicate*, and *object* [5]. The subject is said to be in the *domain* of the predicate and the object is said to be in the *range* [10].

We define an *attribute* as a class whose membership is inferred from classes, properties, and relations of an external ontology. An *access control attribute* is an attribute that is interesting in the context of access control. A *concept hierarchy* is the taxonomy implied by the subclass relations among access control attributes. All access control attributes are subattributes (either directly or indirectly) of either the *PersonAttribute* or *ResourceAttribute* class.

In the context of this paper, a *resource* is a member of the *Resource* class defined by the existing PRPL framework, which implies that it is sharable (although not necessarily shared). A *requester* is a member of the *Person* class that may request access to a resource. An *owner* of a resource

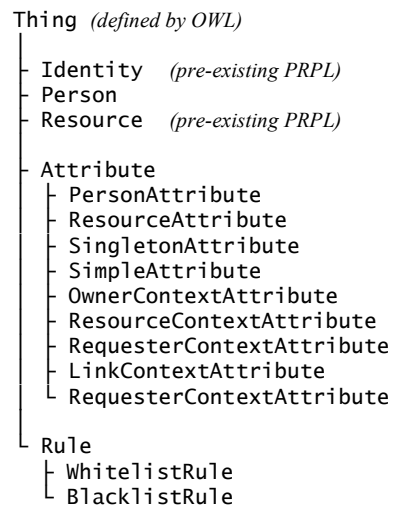


Figure 2. Classes defined in the PRPL and PRPL Access Control ontologies.

$\lambda_{owner}$ :	the resource owner	bound by arbiter
$\lambda_{req}$ :	the requester	bound by arbiter
$\lambda_{res}$ :	the requested resource	bound by arbiter
$\lambda_{link}$ :	inter-attribute link	bound via ?link

Figure 3. Variables left unbound by attributes.

has the authority to grant or revoke access to a resource (i.e. create or destroy rules regarding the resource), but does not necessarily have to be a member of the Person class. Each resource has exactly one owner.

A *uniform resource identifier* (URI) is a globally unique compact string that identifies a resource or person [3]. The existing PRPL ontology defines *identities*, which map a key (usually a person’s email address) to a URI. Since the identity is used in the existing PRPL infrastructure to identify a person [1], the owner and requester must each have an identity.

An *access control rule* (or simply *rule*) is a relation between two access control attributes, a *PersonAttribute* instance and a *ResourceAttribute* instance. For brevity, we respectively refer to these as the *Person attribute* and the *Resource attribute*. An access control rule must be a member of either the *WhitelistRule* or *BlacklistRule* classes, which respectively indicate that access is to be granted or denied. Our prototype supports only whitelisting at this time.

A *request* is a pair, the requester and the requested resource. All access under PRPL is read-only so there is no need for access level. A rule is *applicable* to a particular request if the requester is a member of its Person attribute and the resource is a member of its Resource attribute. The *arbiter* is the software that interprets the applicable rule(s) for a request and responds yes or no to the request.

## 2.2. Attribute Order and Variables

A *singleton attribute* is an attribute that is defined to have exactly one member. Every resource and person implies a singleton attribute. Both attributes in example  $x_1$  are singletons, one for myiphone.jpg and one for Janet.

In contrast, the attributes in example  $x_2$ , Family and Wedding Photos, are not singletons. Since the cardinality of these attributes may vary after a rule has been created, the rule automatically adjusts. In this example, as Susan gets more wedding photos, access is automatically granted to her family. Similarly, if Susan’s family grows or shrinks, the Family attribute adjusts without intervention from her. This usability feature is in contrast to the system of user-defined static groups used in Role-Based Access Control (RBAC) [4] and less sophisticated access control systems.

An attribute is *simple* if it can be inferred directly from the underlying class or property of the external ontology and are

not singleton. In example  $x_2$ , the Wedding Photo attribute is likely to be simple. That is, the external ontology probably defines a Wedding Photo class or property that directly maps to the Wedding Photo attribute.

However, the Family attribute is an *owner-context* attribute, which means that it requires a specific owner to infer membership. For example, suppose Mary and Lisa are sisters of each other, but not Susan. The Family attribute would apply to Mary if Mary requests access for a resource owned by Lisa, but not a resource owned by Susan.

This implies the need for *unbound variables* in the attribute definition, which are listed in Figure 3. When the arbiter queries for the rule and its attributes, it binds the variable,  $\lambda_{owner}$ , to the owner before submitting the query.

In fact, there are three variables that are bound by the arbiter:  $\lambda_{owner}$ ;  $\lambda_{req}$ , which is bound to the requester; and  $\lambda_{res}$ , which is bound to the resource. All Person attributes must reference  $\lambda_{req}$  and all Resource attributes must reference  $\lambda_{res}$ .

An attribute whose unbound variables are only those described above are referred to as *zeroth-order attributes*. The attribute classes already discussed, singleton, simple, and owner-context, are zeroth-order. A rule that contains only zeroth-order attributes is a *zeroth-order rule*.

Zeroth-order rules are insufficient to express example  $x_3$  since the requester (i.e. the email recipient) is defined in terms of the resource (i.e. the email). The closest approximation with zeroth-order rules would allow any email recipient to access any email Susan has sent – clearly not the same intent.

A *first-order attribute* is either a Person attribute that references  $\lambda_{res}$  or a Resource attribute that references  $\lambda_{req}$ . A rule that contains either two first-order attributes or a first-order attribute and a zeroth-order attribute is called a *first-order rule*.

In contrast, a *second-order attribute* is one that references the variable  $\lambda_{link}$ , which is bound by the rule. As such, both attributes must be second order if either is second order. A *second-order rule* is one that relates two second-order attributes.

Example  $x_4$  requires a second-order rule since the email itself is the inter-attribute link. The closest approximation with a first-order rule would grant access for any recipient of any email from Susan to all attachments of any email from Susan. The arbiter of our prototype contains logic for zeroth-, first- and second-order rules. However, second-order rules have not been tested as of the writing this document.

This approach implies higher-order attribute and rules, where there may be multiple links in a chain between pairs of attributes. However, such is beyond the scope of this document.

### 3. Arbiter Implementation

#### 3.1. Query

SPARQL can be used to query an RDF-encoded ontology [6]. We found SPARQL useful in coding the prototype since the existing PRPL uses the Jena framework [7] to store semantic information as an RDF and the Jena framework can execute SPARQL queries via ARQ [8].

The SPARQL query at the heart of the arbiter is shown in Figure 4. To incorporate the context, the arbiter replaces  $\lambda_{owner}$  with the URI of the resource owner,  $\lambda_{req}$  with the URI of the requester, and  $\lambda_{res}$  with the URI of the resource. The query then answers with the applicable rule set where the URI of each rule is returned in `?rule`.

A slight modification to the query answers with the list of resources available to a particular person. The modification adds the variable `?resource` to the `select` clause and replaces instances of `< $\lambda_{res}$ >` in the `where` clause with references to `?resource`. The result is a list of pairs of resource and the applicable rules.

Similarly, a query that determines the persons who can access a particular resource is achieved by adding the variable `?requester` to the `select` clause and replacing instances of `< $\lambda_{req}$ >` in the `where` clause with references to `?requester`. This query answers with a list of pairs of requesters' identities and applicable rules. Since the prototype does not support blacklisting at this time, it ignores the `?rule` portion of the pair.

#### 3.2. Classes and Predicates

Two properties are defined for the attribute classes: the *Target* (with value  $\tau$ ) and the *Predicate* (with value  $\pi$ ). The purpose of these properties is to define a triple  $\langle s, p, o \rangle$  of the external ontology that applies to the attribute. For Person attributes, the subject  $s$  must be the requester. For Resource attributes,  $s$  must be the resource being requested. The predicate  $p$  must equal  $\pi$  and the object  $o$  must equal  $\tau$ .

The *SingletonAttribute* class requires only the Target property, in which case  $\tau$  is the single requester or resource. In the  $x_1$  example,  $\tau$  would be a URI representing `myiphone.jpg` in the Resource attribute and `Janet` in the Person attribute.

The *SimpleAttribute* class is the only one that requires both a Predicate and a Target property so that  $\pi$  and  $\tau$  fully specify the external triple. As such, any simple property of an external ontology may be converted into a simple attribute so long as the  $s$  is the requester or resource. In the common case, the  $\pi$  is `<rdf:type>` and the  $\tau$  is a class of the external ontology. In the  $x_2$  example, if the external ontology represents wedding photos with a `WeddingPhoto` class, then  $\tau$  is that class. However, if instead of a class, the external ontology defines a property `TakenAt` and a keyword `Wedding`, then these respectively are  $\pi$  and  $\tau$ .

In the *OwnerContextAttribute* class,  $\tau$  is implied to be the owner ( $\lambda_{owner}$ ) and  $\pi$  specifies the relationship. For example  $x_2$ , the external ontology may define an `IsFamilyWith` property. In this case,  $\pi$  is `IsFamilyWith` and  $\tau$  is bound by the arbiter to the owner's URI.

*ResourceContextAttribute* and *RequesterContextAttribute* work in a similar way, except that  $\tau$  is implied to be the resource or requestor (respectively). This allows for first-order attributes such as the Person attribute of example  $x_3$ .  $\pi$  could be set to an externally defined `IsRecipientOf`, a property of the requester.

Since second-order attributes come in pairs, the  $\tau$  of the *LinkContextAttribute* class is implicitly the inter-attribute link represented by the `?link` variable in Figure 4. We let  $\pi_\psi$  be the  $\pi$  of the Person attribute and  $\pi_\phi$  be the  $\pi$  of the Resource attribute.  $\pi_\psi$  and  $\pi_\phi$  can be any predicate that is meaningful to the external ontology. For example  $x_4$ ,  $\pi_\psi$  may be `IsRecipientOf` and  $\pi_\phi$  may be `IsAttachedTo` so that  $\tau$  is the linking email.

We consistently require that the requestor or resource be the subject  $s$  and  $\tau$  to be the object  $o$  of the externally defined triple. An early version of the prototype defined classes that reversed these two so that  $\tau$  equals  $s$  and the requester or resource is  $o$ . Although this inverse attribute feature improves expressibility, we judged that it would likely be redundant since we assume that ontologies would generally define inverse properties. For example, we assume that an ontology that defines `IsAttachedTo` would also define `HasAttachment`. Such inverse properties are easily inferred assuming the ontology is in the Web Ontology Language (OWL) family [9].

#### 3.3. Rule Ordering

In our prototype, we use the Jena framework [7], which provides a useful interface for inference engines. In particular, the OWL inference engine included with Jena is more than sufficient to infer the concept hierarchy. In fact, our prototype requires only Subclass property defined in RDFS [10] to implement the concept hierarchy.

The set of applicable rules may contain a combination of whitelist and blacklist rules. In this case, a rule priority decides which rule is to be used by the arbiter. The priority is based on the priority of its attributes as follows.

For two attributes  $\alpha_1$  and  $\alpha_2$ , we have that  $\alpha_1 < \alpha_2$  iff  $\alpha_1$  is a distinct subclass of  $\alpha_2$  (i.e. if  $\alpha_1$  is less general than  $\alpha_2$  in terms of the concept hierarchy). Additionally,  $\alpha_1 \leq \alpha_2$  iff  $\alpha_1 < \alpha_2 \vee \alpha_1 = \alpha_2$ . Since not all attributes are directly related, we have a partial ordering of attributes.

Let  $\psi_i$  be the Person attribute and  $\phi_i$  be the Resource attribute of rule  $\rho_i$ . When comparing two rules  $\rho_1$  and  $\rho_2$  we define  $\rho_1 < \rho_2$  iff  $\psi_1 \leq \psi_2 \wedge \phi_1 \leq \phi_2 \wedge (\psi_1 \neq \psi_2 \vee \phi_1 \neq \phi_2)$ . We have a partial ordering makes no distinction between whitelist and blacklist rules.

The arbiter grants access only if for every blacklist rule  $\beta$  in the applicable rule set, there exists a whitelist rule  $\gamma$  such that  $\gamma < \beta$ . That is, each applicable blacklist rule must be overridden by a more specific whitelist rule. The arbiter adds a general blacklist rule relating the PersonAttribute and

ResourceAttribute classes so that an empty applicable rule set results in denial. Since blacklisting is not part of the current prototype, it contains no logic to compare rules.

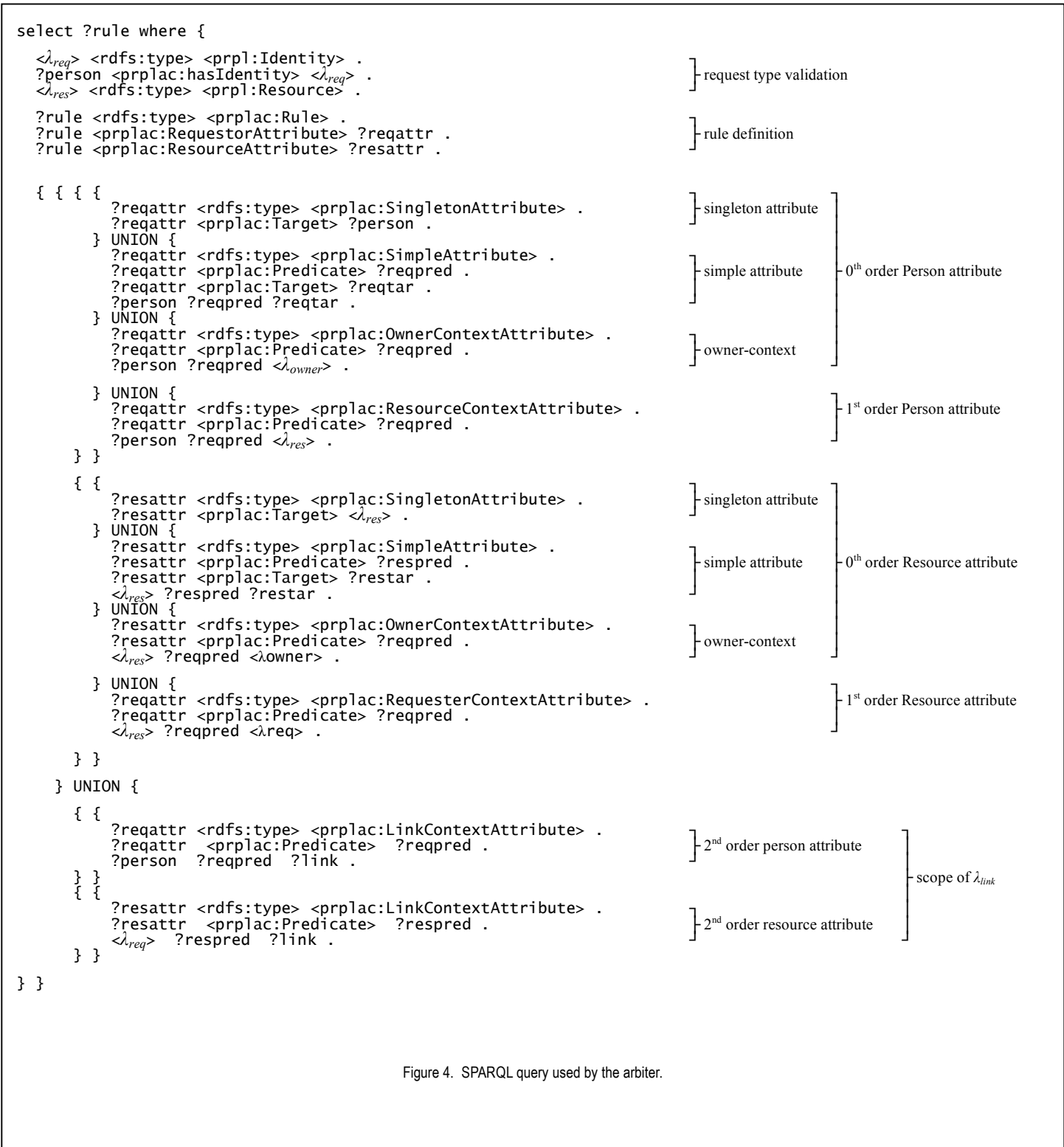


Figure 4. SPARQL query used by the arbiter.

## 4. Future Work

### 4.1. Access Control Attribute Inference

We defined an access control attribute as an attribute that is interesting in the context of access control without defining what *interesting* means. *Interesting* could be viewed as either a boolean or as a full order converted to a boolean by comparison to a threshold. The advantage of the latter is that a user interface application could adjust the threshold in response to user input. In effect, the user may select the level of granularity to his or her tastes.

The interest-level property may come from the external ontology itself if it is aware of PRPL Access Control. Alternatively, inference engines that interface to the Jena framework could utilize machine-learning techniques to set the interest-level. As such, these inference engines would not have to be specific to the external ontology.

These techniques may benefit from usage data such as most-recently-used (MRU) and most-frequently-used (MFU). It may also be feasible to apply MRU and MFU data only from the user, from the user's friends, from friends of friends, and so on. It may also be useful to leverage MFU and MFU information from aggregate user experience.

Such inference engines may use more domain-specific information as well. For example, attributes identifying an owner's clique could be inferred from an analysis of their strong ties in a social network [11].

Clearly, any attribute previously or currently referenced in a rule should be marked as interesting since it would be frustrating for the user to lose the ability to create a rule based on attributes she had already used.

Furthermore, the OWL ontology can be utilized. For example, a Person attribute cannot be considered interesting unless its domain (as defined by OWL) is a superset of the Person class. Similarly, the domain of a Resource attribute should be a superset of the Resource class. Since the arbiter enforces such inferences in effect, such a mechanism would prove essential.

### 4.2. Concatenation

It may be useful for the owner to create attributes that are a concatenation of two or more attributes. For example, Susan may want to share wedding photos with family members who attended the wedding. Since the ontology may represent attendance at an event as a separate triple from the family relationship, two separate Person attributes are required such that membership in both attributes is required to grant access.

Concatenation implies that the class structure of the concept hierarchy is no longer a tree but a directed acyclic graph. Any user interface that displays the concept hierarchy to the user, however, can flatten the graph into a tree either (a) by duplicating concatenation attributes or (b) by showing

concatenation attributes only if the user selects multiple operand attributes. For example, the user interface may show Susan "Family who Attended Wedding" by requiring that she select either "Attended Wedding" or "Family" (case a) by requiring that she select both "Attended Wedding" and "Family" (case b).

The methodology described in this paper is not sufficient since rules pair exactly one Person attribute with one Resource attribute. Multiple rules imply disjunction in common cases (e.g. when all rules are whitelist). Two approaches to solve this problem are worth investigating.

The first is to add the ability to define a concatenation attribute that references two (or more) operand rules. If this logic implemented in the SPARQL query, it would not scale well since the query would have to reference the concatenation rule and operand rules. This observation suggests that an inference engine rather than the arbiter's SPARQL query should be used to imply membership in the concatenation rule from membership in the operand rules. Zeroth- and first-order attributes would be supported by this scheme since  $\lambda_{owner}$ ,  $\lambda_{req}$ , and  $\lambda_{res}$  may be left unbound in the concatenation rule.

However, the use of two or more second-order concatenation attributes implies the need for two or more instances of  $\lambda_{link}$ , which in turns suggests a modification to the arbiter's query. To deal with this, rather than define strict concatenation per se, the rules may be augmented with two integer properties: *RequesterAttributeCount* and *ResourceAttributeCount*. The former specifies the minimum number of Person attributes that must match the requester in order for the rule to apply. Similarly, the latter specifies the number of Resource attributes. Concatenation is achieved when the number of Person and Resource attributes referenced by the rule equals the numbers in these properties.

## 5. Conclusion

We have demonstrated that it is feasible to leverage semantic web information for access control and designed the Attribute-Based Access Control (ABAC) methodology to support virtually any abstract external ontology. ABAC defines attributes such that they simplify the external ontology into a taxonomic concept hierarchy that may be presented to the nontechnical user in a meaningful way while allowing for fine granularity.

When the taxonomy seen by the user is the same as the taxonomy used by the arbiter, there is certainty that the rules the user views will be interpreted as she defines. Since the attributes are interpreted in a context with several variables, they are also reasonably flexible, allowing ABAC to leverage virtually any external ontology. Thus we meet our goal of access control that is fine-grained and easy to use. As the amount of information available in the semantic web

becomes abundant, some form of ABAC will become indispensable.

Finally, The authors would like to acknowledge Chris Brigham and Seok-Won Seong for their many hours of help and Monica Lam for her guidance and vision.

## 6. References

- [1] The PRPL Java code repository, June 2008, unpublished.
- [2] G. Bayer, D. Sengupta, T. Wang, A. and Vogt, "PrplAc: Attribute-Based Access Control in PRPL for Fine Grained Information Sharing using Semantic Web, June 2008, unpublished.
- [3] T. Berners-Lee, R. Fielding, L. Masinter, "STD66 RDF3986: Uniform Resource Identifier (URI): Generic Syntax," IETF Full Standard RFC, Jan 2005, Available: <http://www.ietf.org/rfc/rfc3986.txt>.
- [4] D. F. Ferraiolo and D.R. Kuhn, "Role Based Access Control," 15th National Computer Security Conference, Oct 1992, Available: <http://csrc.nist.gov/groups/SNS/rbac/>.
- [5] P. Hayes, "RDF Semantics: W3C Recommendation," W3C, Feb 10, 2004, Available: <http://www.w3.org/TR/2004/REC-rdf-mt-20040210/>.
- [6] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF: W3C Working Draft," W3C, 2006, Available: <http://www.w3.org/TR/rdf-sparql-query/>.
- [7] "Jena: A Semantic Web Framework for Java," HP Labs Semantic Web Research, 2008, Available: <http://jena.sourceforge.net/>.
- [8] "ARQ: A SPARQL Processor for Jena," HP Labs Semantic Web Research, 2008, Available: <http://jena.sourceforge.net/ARQ/>.
- [9] M. K. Smith, C. Welty, and D. L. McGuinness, "OWL Web Ontology Language Guide: W3C Recommendation," W3C, Feb 10, 2004, Available: <http://www.w3.org/TR/2004/REC-owl-guide-20040210/>.
- [10] D. Brickley, R.V. Guha, and B. McBride, "RDF Vocabulary Description Language 1.0: RDF Schema: W3C Recommendation," W3C, Feb 10, 2004, Available: <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>.
- [11] M. Granovetter, "The Strength of Weak Ties: A Network Theory Revisited," *Sociological Theory*, Vol. 1, 1983, pp 201-233, Available: <http://www.jstor.org/stable/202051>.