available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Semantic web-based social network access control

Barbara Carminati[a], *Elena Ferrari*[a], *Raymond Heatherly*[b,*], *Murat Kantarcioglu*[b], *Bhavani Thuraisingham*[b]

[a] *The University of Insubria, Via Mazzini, 5 Varese, Italy*

[b] *The University of Texas at Dallas, Computer Science, 800 W. Campbell Road, Richardson, Texas 75080, USA*

ARTICLE INFO

Article history:
Received 11 April 2010
Received in revised form
16 August 2010
Accepted 27 August 2010

Keywords:
Social network
Access control
Semantic inference
Experimentation
Implementation

ABSTRACT

The existence of online social networks that include person specific information creates interesting opportunities for various applications ranging from marketing to community organization. On the other hand, security and privacy concerns need to be addressed for creating such applications. Improving social network access control systems appears as the first step toward addressing the existing security and privacy concerns related to online social networks. To address some of the current limitations, we have created an experimental social network using synthetic data which we then use to test the efficacy of the semantic reasoning based approaches we have previously suggested.

1. Introduction

On-line Social Networks (OSNs) are platforms that allow people to publish details about themselves and to connect to other members of the network through links. Recently, the popularity of OSNs is increasing significantly. For example, Facebook now claims to have more than a hundred million active users.[1] The existence of OSNs that include person-specific information creates both interesting opportunities and challenges. For example, social network data could be used for marketing products to the right customers. At the same time, security and privacy concerns can prevent such efforts in practice (Berteau, 2007). Improving the OSN access control systems appears as the first step toward addressing the existing security and privacy concerns related to online social networks. However, most of current OSNs implement very basic access control systems, by simply making a user able to decide which personal information are accessible by other members by marking a given item as public, private, or accessible by their direct contacts. In order to give more flexibility, some online social networks enforce variants of these settings, but the principle is the same. For instance, besides the basic settings, Bebo (http://bebo.com), Facebook (http://facebook.com), and Multiply (http://multiply.com) support the option "selected friends"; Last.fm (http://last.fm) the option "neighbors" (i.e., the set of users having musical preferences and tastes similar to mine); Facebook, Friendster (http://friendster.com), and Orkut (http://www.orkut.com) the option "friends of friends"; Xing (http://xing.com) the options "contacts of my contacts" (2nd degree contacts), and "3rd" and "4th degree contacts". It is important to note that all these approaches have the advantage of being easy to be implemented, but they lack flexibility. In fact, the available protection settings do not allow users to easily specify their access

* Corresponding author.
E-mail addresses: barbara.carminati@uninsubria.it (B. Carminati), elena.ferrari@uninsubria.it (E. Ferrari), rdh061000@utdallas.edu (R. Heatherly), muratk@utdallas.edu (M. Kantarcioglu), bxt043000@utdallas.edu (B. Thuraisingham).

[1] http://www.facebook.com/press/info.php?Fstatistics.

0167-4048/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2010.08.003

control requirements, in that they are either too restrictive or too loose. Furthermore, existing solutions are platform-specific and they are hard to be implemented for various different online social networks.

To address some of these limitations, we propose an extensible, fine-grained OSN access control model based on semantic web technologies. Our main idea is to encode social network-related information by means of an ontology. In particular, we suggest to model the following five important aspects of OSNs using semantic web ontologies: (1) user's profiles, (2) relationships among users (e.g., Bob is Alice's close friend), (3) resources (e.g., online photo albums), (4) relationships between users and resources (e.g., Bob is the owner of the photo album), (5) actions (e.g., post a message on someone's wall). By constructing such an ontology, we model the Social Network Knowledge Base (SNKB). The main advantage for using an ontology for modeling OSN data is that relationships among many different social network concepts can be naturally represented using OWL. Furthermore, by using reasoning, many inferences about such relationships could be done automatically. Our access control enforcement mechanism is then implemented by exploiting this knowledge. In particular, the idea is to define security policies as rules (see Section 3), whose antecedents state conditions on SNKB, and consequents specify the authorized actions. In particular, we propose to encode the authorizations implied by security policies by means of an ontology, obtaining the Security Authorization Knowledge Base (SAKB). Thus, security policies have to be translated as rules whose antecedents and consequents are expressed on the ontology. To achieve this goal, we use the Semantic Web Rule Language (SWRL) (Horrocks et al., 2004). As consequence, the access control policies can be enforced by simply querying the authorizations, that is, the SAKB. The query can be easily directly implemented by the ontology reasoner by means of instance checking operations, or can be performed by an SPARQL query, if the ontology is serialized in RDF. In this paper, we focus on how to model such a fine-grained social network access control system using semantic web technologies. We also assume that a centralized reference monitor hosted by the social network manager will enforce the required policies. Since our proposed approach depends on extensible ontologies, it could be easily adapted to various online social networks by modifying the ontologies in our SNKB. Furthermore, as we discuss in details later in the paper, semantic web tools allow us to define more fine-grained access control policies than the ones provided by current OSNs.

The paper is organized as follows. In Section 2, we provide a brief discussion of current security and privacy research related to online social networks. In Section 3, we introduce a high level overview of the security policies we support in our framework. In addition to access control policies, we state filtering policies that allow a user (or one of her supervisors) to customize the content she accesses. We also introduce administration policies, stating who is authorized to specify access control and filtering policies. In Section 4, we discuss how security policies could be enforced. In Section 5, we give an overview of the architecture we have chosen to integrate the semantic components. In Section 6, we describe the synthetic data that we use in our experiments, and in Section 7 we discuss and provide the results of experiments using our implementation of semantic web-based access control for social networks. Finally, we conclude the paper in Section 8.

2. Related work

Past research on OSN security has mainly focused on privacy-preserving techniques to allow statistical analysis on social network data without compromising OSN members' privacy (see Carminati et al. (2008) for a survey on this topic). In contrast, access control for OSNs is a relatively new research area. As far as we are aware, the only other proposals of an access control mechanism for online social networks are works by Kruk et al. (2006), Ali et al. (2007) and Carminati et al. (2008). The D-FOAF system (Kruk et al., 2006) is primarily a Friend of a Friend (FOAF) ontology-based distributed identity management system for social networks, where access rights and trust delegation management are provided as additional services. In D-FOAF, relationships are associated with a trust level, which denotes the level of *friendship* existing between the users participating in a given relationship. Although the work by Kruk et al. (2006) discusses only generic relationships, corresponding to the ones modeled by the foaf:knows RDF property in the FOAF vocabulary (Brickley and Miller, 2007), another D-FOAF-related paper (Choi et al., 2006) considers also the case of multiple relationship types. As far as access rights are concerned, they denote authorized users in terms of the minimum trust level and maximum length of the paths connecting the requester to the resource owner. In the work by Ali et al. (2007), authors adopt a multi-level security approach, where trust is the only parameter used to determine the security level of both users and resources. In the work by Carminati et al. (2009b), a semi-decentralized discretionary access control model and a related enforcement mechanism for controlled sharing of information in OSNs is presented. The model allows the specification of access rules for online resources, where authorized users are denoted in terms of the relationship type, depth, and trust level existing between nodes in the network.

Compared to existing approaches, we use semantic web technologies to represent much richer forms of relationships among users, resources and actions. For example, we are able to represent access control rules that leverage relationship hierarchies and by using OWL reasoning tools, we can infer a "close friend" is also a "friend" and anything that is accessible by friend could be also accessible by a "close friend". In addition, our proposed solution could be easily adapted for very different online social networks by modifying the underlying SNKB. A further discussion on the differences between the proposed framework and the access control mechanism in the work by Carminati et al. (2009b) is provided in Section 3.

Semantic web technologies have been recently used for developing various policy and access control languages for domains different from OSNs. For example, in the work by Tonti et al. (2003), authors compare various policy languages for distributed agent based systems that define authorization and obligation policies. In the work by Finin et al. (2008), OWL is used to express role-based access control policies. In the work by Yague et al. (2005), authors propose a semantic access control model that separates the authorization and access control management responsibilities to provide solutions for distributed and dynamic systems with heterogeneous security requirements. None of these previous work deals with the access control issues related to online social networks. Among the existing works, the one by Elahi et al. (2008) is the most similar to our proposal. Compared to the work of Elahi et al. (2008), we provide a much richer OWL ontology for modeling various aspects of online social networks. In addition, we propose authorization, administrative and filtering policies that depend on trust relationships among various users.

3. Security in online social networks

For a detailed discussion of the use of semantic technologies in online social networks, please refer to our work in Carminati et al. (2009a). Here, we will constrain our discussions to those specific topics which impact our implementation of an access control mechanism for resources in an online social network.

In the recent past, Facebook has made significant changes to its method of defining the relationships between friends on the network. Previously, if we had two Facebook users who were friends, John and Jane for instance, either John or Jane could select one of the pre-chosen friendship types that Facebook allowed, where the other friend would be required to confirm or reject this label of their friendship. Following this, any friend of either John or Jane could see the definition that was applied to this friendship.

Now, however, instead of defining link types that are visible to others, each individual has the ability to create meaningful lists. Friends can then be added into as many lists as a user chooses. These lists can then be used to control visibility to status updates, wall posts, etc. However, there is no way to define a hierarchy of these lists. For instance, if one was to create a ‘High School Classmates’ and then a ‘College Classmates’ list, there is no way to create a ‘Classmates’ list, without individually adding each individual person to that third list.

We represent a friendship using the n-ary relation pattern, as specified by the W3C Consortium (1999). This means that each friendship is an instance of a class, which we call *FriendshipRelation*. This allows us to maintain separate information about each friendship. Specifically, we maintain a *TrustValue* for each friendship. This allows us to determine a specific strength of a friendship, even when compared to those in the same class.

Our implementation supports access control policies to regulate how resources can be accessed by the members of an online social network. In particular, the supported access control policies are defined on the basis of our previous work (Carminati et al., 2009b). Here, authorized users are denoted in terms of the type and/or trust level of the relationships between nodes in the network. For instance, an access control policy can state that the only OSN participants authorized to access a given resource are those with a direct friendship relationship with the resource owner, as long as the relationship also has a certain trust level.

Note, however, that using semantic reasoning can give some improvements over the capabilities discussed in the work by Carminati et al. (2009b). This benefit comes from our ability to specify access control policies over semantic concepts in the OSN. For example, as a default, Facebook may specify that photos can only be viewed by direct friends. In the absence of policies defined by individual users, reasoning will default to this general policy.

3.1. *Filtering policies*

In an OSN, users can publish information of very heterogeneous content, ranging from family photos to adult-oriented contents. In this sense, the access control issues arising in OSNs are similar to those we have in the web, where the availability of inappropriate information could be harmful for some users (for example, young people). To protect users from inappropriate or unwanted contents, we introduce *filtering policies*, by which it is possible to specify which data has to be filtered out when a given user browses the social network pages. By means of a filtering policy, it is, for example, possible to state that from OSN pages fetched by user Alice, all videos that have not been published by Alice’s direct friends have to be removed.

Similar to access control policies, filtering policies are defined as rules over ontologies. This implies that policy propagation is possible also in case of filtering policies. Another relevant aspect of filtering policies is related to the user that specifies the policy (i.e., the grantor). We define two methods where a filtering policy may be created. According to the first one, a filtering policy is specified by a user to state which information she prefers not to access, i.e., which data has to be filtered out from OSN pages fetched by her. Thus, in this case the grantor and the user to which the policy applies, i.e., the target user, are the same. These policies state user preferences w.r.t. the contents one wants to access and for that reason are called *filtering preferences*. However, we also support the specification of filtering policies where the target user and the grantor are different. This kind of filtering policies makes the grantor able to specify how the SN pages fetched by target users have to be filtered. By means of these filtering policies, a grantor can *supervise* the content a target user can access. In this case, we refer to the filtering policy as *supervised filtering policy*. This represents an extremely useful feature in open environments like OSNs. For example, a parent can specify a supervised filtering policy stating that her children do not have to access those videos published by users that are not trusted by the parent herself. As it will be more clear later on, semantic technologies greatly facilitate the specification of this kind of policies.

It is worth noticing that both filtering preferences and supervised filtering policies cannot be enforced by simply supporting negative access control policies, that is, policies avoiding access to resources. This is due to the fact that access control policies and filtering policies have totally different semantics. Indeed, an access control policy is specified by the resource owner to state who is authorized or denied to access her resources. Rather, a filtering policy is specified by a supervisor for a target user or by the target user herself, to specify how resources have to be filtered out when she fetches an OSN page. Note that, according to the proposed semantics, this filtering takes place even in the case the target user is authorized to access the resource, that is, even if she satisfies the access control policies specified by the resource owner.

4. Security rule enforcement

Our framework acts like a traditional access control mechanism, where a *reference monitor* evaluates a request by looking for an authorization granting or denying the request. Exploiting this principle in the proposed framework implies retrieving the authorizations/prohibitions by querying the SAKB ontology. Thus, for example, to verify whether a user *u* is authorized to specify access control policies for the read privilege on object *o*, it is necessary to verify if the instance *AdminRead*(*u*,*o*) is in the ontology, i.e., to perform an instance checking. This implies that before any possible requests evaluation all the SWRL rules encoding security policies have to be evaluated, thus to infer all access control/administrative authorizations as well as all prohibitions. For this reason, before policy enforcement it is required to execute a preliminary phase, called *policy refinement*. This phase aims to populate the SAKB with the inferred authorizations/prohibitions, by executing all the SWRL rules encoding security policies.

Once authorizations/prohibitions are inferred, security policy enforcement can be carried out. In particular, access control and filtering policies are evaluated upon an *access request* being submitted, whereas administrative policies are evaluated when an *administration request* is submitted. We do this because, in general, the number of people given administrative access to an object is far less than the number of individuals who may have access to that same object. In the following, we present both the request evaluation by showing how the corresponding policies are enforced.

4.1. *Administration request evaluation*

An administrative request consists of two pieces of information: the name of the *grantor*, i.e., the user that has submitted the administrative request, and the access control or filtering policy the grantor would like to specify, encoded as SWRL rule, that is, the *submitted SWRL*. The submitted SWRL has to be inserted in the system only if there exists an administrative authorization in the SAKB for the grantor. For example, if the submitted rule requires to specify an access control policy for the read privilege on targetObject, then there must exist an instance of <grantor, targetObject>:*Read*. Note that information about the privilege and the targetObject can be retrieved directly from the submitted SWRL. Thus, in order to decide whether the request above can be authorized or not, a possible way is to query the SAKB to retrieve the corresponding administrative authorization, if any. If there exists an instance, then the submitted SWRL can be evaluated, otherwise the framework denies to the grantor the administrative request. An alternative way is to rewrite the submitted SWRL by adding in its body also condition to verify whether there exists an administrative authorization in the SAKB authorizing the specification of the rule. The following example will clarify the underlying idea.

Example 1. *Let us assume that the system receives the following administrative request: {Bob, Tag}, where Tag is the following:*

Tag: Owns(Bob,?targetObject) ^ Photo(?targetObject) ⇒ Tag(?targetSubject,?targetObject)

That is, that if Bob is the owner of a specific photo (represented by ?targetobject), then the ?targetSubject should be added to the list of users who is allowed to tag new individuals in the photo. In order to determine the result of the administrative request, the framework has to verify the existence of <Bob,targetObject>: *AdminTag instance in the SAKB. That is, a check to ensure that Bob is allowed to convey the Tag privilege on the photograph.*

This revised check can be incorporated in the body of Tag by simply modifying it as follows:

New_Tag: AdminRead(Bob,?targetObject) ^ Photo(?targetObject) ⇒

Tag(?targetSubject, ?targetObject)

Then New_Tag is evaluated with the consequence that the individual will have the ability to tag people in the photo only if Bob is authorized to specify them by an administration policy.

4.2. *Access request evaluation*

In general, an access request can be modeled as a triple (*u*, *p*, *URI*), which means that a user *u* requests to execute the privilege *p* on the resource located at *URI*. To evaluate this request the framework has to verify whether there exists an access control authorization granting *p* on *URI* to requester *r*. However, since the proposed system also supports filtering policies, the presence of such an authorization does not necessarily imply that *r* is authorized to access *URI* because there could be a prohibition denying access to the resource to the user. Thus, to evaluate whether an access request has to be granted or denied, it is necessary to perform two queries to the SAKB. The first to retrieve authorizations and the second to retrieve prohibitions. More precisely, if *u* requires the read privilege *Read*, the system has to query the instances of object property *Read* and *PRead*. In particular, both the queries look for instance <u, URI> (i.e., <u, URI>:*Read* and <u, URI>:*PRead*). Then, the access is granted if the first query returns an instance and the second returns the empty set. It is denied otherwise.

5. Architecture

In our system, we built several layers on top of a reduced online social network application. We considered the actions of a social network (messages, wall posts, viewing profiles, viewing images, etc.) and examined those that involved the most access requests. For example, if a user, John, was to go to Jane’s profile, then in the best case, there is a single check (are John and Jane friends of an appropriate level) on permissions.

However, when you consider an image, which can easily have a dozen people tagged in it, and each of those individuals may specify their own additional constraints to the viewership of that image, then it is easy to see that this will be the more complicated example of permissions inference in a system. In effect, we built our system to test a consistent series of worst-case scenarios to test its ability to handle a testing load.

We implement our prototype using the Java based open source semantic web application development framework called JENA,[2] since it offers an easy to use programmatic environment for implementing the ideas discussed in this paper, as well as generally being a framework that is supported by most semantic products currently available. Here, we describe each of these layers independently, as well as the motivation behind choosing specific technologies in our framework. While we use specific instances of Facebook as the over-arching application utilizing the lower level semantic layers, any social network application could potentially be modified to use the design we describe here.

5.1. *RDF datastore*

We use a general RDF triple-store to hold the underlying data. In this representation, all facts about an entity are recorded as a triple of the form <Subject, Predicate, Object>. So, suppose we have an individual named John Smith, who is assigned a unique identifier 999999999, would give us the tuple <999999999, *foaf:Name*, "John Smith" >.

Due to the size of the data set we used for experimentation, we do maintain a separate datastore for each of the following: list of users, security policy for each user, resource descriptions, and each generic class of friend (see 6 for specifics on generic classes). While not strictly the desired method of maintaining RDF data, we determined that because of the large size of the data we were using, it was not feasible to store all of this in fewer datastores, due to the computational time that would have been required for even the simplest of queries. Even through the use of database indices, there are still far too many tuples in a single table to be efficient for use. Due to its ease of interface and its availability, we used MySQL as the database engine.

We note here that an RDF datastore differs from a relational database in that there is no database method of ensuring that constraints are maintained on the ontology as a whole, such as making sure that a defined *Person* has a name. Because we condense the actions of a social network, we assume that this is handled programmatically at a higher layer.

5.2. *Reasoner*

Any reasoner that supports SWRL rules can be used to perform the inferences described in this paper. However, we initially chose SweetRules[3] because it interfaces with JENA and has a rule-based inference engine, which meant that we could use both forward and backward chaining in order to improve the efficiency of reasoning for enforcing our access control policies. However, during implementation of our system, difficulties arose through the use of SweetRules (described later in Section 7) and, due to its success in other projects, chose to use Pellet.[4]

[2] http://jena.sourceforge.net/.
[3] http://sweetrules.projects.semwebcentral.org/.
[4] http://clarkparsia.com/pellet.

5.3. *RDF/OWL engine*

For the RDF/OWL interface, we chose to use the JENA API. We use this to translate the data between the application and the underlying data store. JENA has several important features that were considered in its use. First, it is compatible with Pellet. Secondly, it supports OWL-DL reasoning which we could use to verify that the data is consistent with the OWL restrictions on the ontology. The OWL restrictions are simple cardinality and domain/range constraints such as every person has to have a name and must belong to at least one network. To enforce these constraints, we plan to have the application layer pass the statements to be entered about an individual until all have been collected. We then have JENA insert these statements into the database and then check the new model for consistency. If there are any constraints that have been violated, then we pass this information back to the social network application and have it gather the required information from the user.

6. Data generation

As we began our implementation, it was apparent we would need to be able to measure the performance of our implementation on large data sets. Because the size of Facebook (at the time we started the implementation) was approximately 300 million users, we established 350 million as the required number of nodes in our data set, to ensure that our implementation could scale to match the numbers Facebook would soon reach.[5] Unfortunately, there are no publicly available data sets of this size that we could use. Instead, we generated our own data set.

That is, we created a data generator that generates n nodes. We began with creating a group of 50 nodes, and generated 50 edges that were distributed randomly across the 50 nodes in the graph. We then chose a random node, n_i with at least one friend and performed Dijkstra's Algorithm to determine if the subgraph was connected. If the subgraph was not connected, then there were j nodes which were not reachable from n_i. We then chose $0 < r_i \leq j$ to be a number of new edges to create. For each edge, we randomly chose a node from the connected subgraph containing n_i and chose a destination node in the disconnected portion of the subgraph. As long as there was a disconnected portion of the subgraph, we continued generating edges. By performing subgraph-joins in this manner, we are able to generate a social network-like graph while still maintaining randomness and not hand-picking which edges would have links between them.

It is important to note at this point that our datastore records each linked twice. Even though the graph is undirected, there is a link type (used in inference) that is directed. For instance, while the generic 'Friend' link type is bi-directional, the specific 'BestFriend' link type is not necessarily. To maintain this, we recorded each direction of the link with its associated link type. Once the initial subgraph was complete,

[5] It is important to note that at this time, Facebook reports that they have in excess of 450 million users.http://www.facebook.com/press/info.php?Fstatics.

we iterated through the nodes to assign each edge a link type. We established three generic link types: Friend, Family, and Co-Worker, and recorded them for each direction of the edge. That is, $(n_i, n_j) \in \varepsilon$, if assigned the Friend link type, would have generated the tuples {n_i, n_j, Friend}, and {n_j, n_i, Friend}.

We uniformly chose a generic type for each edge that a node is a member of. After this, we then assigned specific subtypes. For 10% of Friend generic link types, we assigned the specific (uni-directional) link type of bestFriend. That is, in the above example, if n_i declared n_j to be a bestFriend, then the tuple {n_i, n_j, Friend} would have become {n_i, n_j, bestFriend}. Note that the second tuple would have remained unchanged at this time.

Next, we used a Pareto distribution over the number of defined Family Members to determine how many relationships would be defined as 'ParentOf'. We use a Pareto distribution because while having one or two parents listed is what we may generally think of as reasonable for a child, in today's mixed families, it would not be outside the realm of the believable to have more parents listed for a child. It is also important to note that when a link was defined as ParentOf, its partner tuple was automatically assigned the inverse relationship of ChildOf. That is, suppose that our earlier example was instead, {n_i, n_j, Family} and {n_j, n_i, Family}. If n_i was determined to be the parent, in a single step, we would have the amended tuples {n_i, n_j, *ParentOf*} and {n_j, n_i, *ChildOf*}.

For the Co-worker generic link type, we did not further define a specific link type.

For each node, we also defined a security policy. For clarity, we defined three security policies, which are chosen uniformly at random:

1. **Strict** – Only BestFriends and Family can view photos of self and any children; their children may not view any videos
2. **Casual** – Anyone can see photos; no restriction on children
3. **ParentStrict** – Anyone can see photos of the parent, only family can see photos of their children; Child cannot see any videos.

We then generated *m* resources, where *m* is a random number less than 4.5 million and more than 750,000, which should allow us to model both more active and more passive social networks. A resource could be either a photo or a video. We weighted the probability of a resource being a photo to 75%. We then drew from a uniform distribution between 1 and 25, inclusive, to represent the number of people to 'tag' in a photo. This 'tag' indicated a person appearing in the photo, and we viewed this as having a 'stake' in the individuals in the network who could see the photo.

It is important here to note several things. The first is that while our uniform distribution may not reflect a true probabilistic model of the realities of photos on a social networking site such as Facebook, the inclusion of larger groups having a stake in photos represents a more difficult inference problem for such a site. The second is that we do not restrict the individuals being tagged in a photo to only those friends of the photo owner. Because this functionality exists in current online social networks, we needed to support this type of tagging. This ability can support photos of events such as weddings, where a person taking a photo may only know a subset of individuals in the photo, but people who view the photo can supply more details as to other individuals who were photographed. We also note at this point that our method generates a graph where the average number of friends that a person has is 102 users[6].

6.1. *Event generation*

We next generated a series of events that will be processed in order to examine the effect of using a semantic web-based reasoner for social network access control. We condensed the full set of actions that can be performed in a social network (such as games, posting on walls, changing one's status, etc.) to those that would most strongly affect the ability of a reasoner: Adding friends, Accessing existing resources, creating new resources, and changing permissions. When creating users, adding friends, or creating new resources we performed the task as described previously. However, when we accessed an existing resource, a randomly chosen user attempted to access a randomly selected resource.

7. Experiments

We performed two independent implementations of reasoners. Our first implementation relied upon the SweetRules inference engine. We attempted to perform inference on the entire dataset. Performing inference in this way took 17 h to load the initial model into memory, and then several seconds to perform each specific reasoning request. However, we noticed that when our model needed to be updated (through new resources, friends, or a change in security policy) these changes were not reflected in our in-memory model. This caused inference done later to become more and more incorrect.

Because of this, we implemented a reasoning solution using Pellet, which has become a very popular reasoner. Initially, we simply changed the reasoner that we used to Pellet. However, in the initial loading step with Pellet, we received out of memory errors and were unable to proceed to the reasoning segment.

We then decided that we needed to implement some type of partitioning scheme for the social network. A naïve approach of partitioning would have resulted in some friendship links that spanned partitions. These cross-partition edges would have resulted in one of two things:

1. Reconstructing partitions – Suppose that a partition, P_i, has an user u_a with n friends, u_1, …, u_n who are stored in partitions P_1,…P_n, respectively. Remember that we can determine who the friends of u_a are from our *Friends* datastore. This ability, however, does not assist us in determining which specific partition their friends are in without an additional index. We must then devote considerable resources to recombining specific partitions in order to be able to effectively infer access permissions.
2. Ignoring links – The other option is that we can simply ignore any friendship links that lead to another partition.

[6] As of April 1, 2010, Facebook reports an average of 130 friends.

This is clearly not a viable option, because it will obviously result in far too many invalid access requests.

We then realized that for any individual access request, there are only groups of users whose security policies and friends are important for determining the success or failure of the request: 1) the resource owner, 2) the person making the access request, and 3) those individuals tagged in the resource. So, we adopt an amended partitioning scheme. For any access request, we generate a temporary table that contains only the members of the three groups mentioned above, all links involving those individuals, the requestor, and the security policies of all these people.

We then use three methods of measuring trust values and perform experiments to determine how each type affects the time required to perform inference:

- Link Type Only (LTO) is the method described above, where there is only a link type for each edge.
- Trust Value Only (TVO) is a method in which we use only a trust value in place of most link types. Note, however, that because of the unique constraints held by a 'ParentOf'/ 'ChildOf' relationship, we do still maintain only this link type. We do not maintain other generic or specific link types here, however. We assume that the trust values (which would be assigned by a user) are more specific measures than a defined link type.
- Value/Trust Hybrid (VTH) is an approach where we retain all generic link type declarations, the 'ParentOf' specific type, and add on top of this a Trust Value. This provides for a finer granularity in a security policy. For instance, instead of just being a BestFriend, a specific user can define various security policies, such as only accessible to 'Friend's with a trust value greater than 7, where they may declare a BestFriend to be a trust value of 6 or higher. This allows the user to restrict even among best friends or family.

The results of our experiments are provided in Table 1. This table shows the average amount of time that it takes to perform inference tasks, as well as the longest time taken and the shortest time taken. Note that this includes the time required to generate the temporary table that is required for the request to be evaluated.

Additionally, we ran an additional series of tests to determine the effect of the number of friends in a graph, as shown in Fig. 1. For these tests, we generated a subgraph where each person had a specific number of friends, ranging between 50 and 150. We repeated tests using our three types of trust measurement (LTO, TVO, and VTH) and report the average time taken for inference. Again, we see that LTO clearly takes the least time for inference. However, additionally, we can see that there is only a slight increase in the time taken for inference at each additional group of friends. Further, there is only a slight decrease at the point where all members have 50 friends, which indicates that most of the time for our inference operation is consumed by the overhead of the inference engine, including our dynamic partitioning method.

Table 1 – Time to conduct inference (in seconds).

	Average	Low	High
LTO	0.585	0.562	0.611
TVO	0.612	0.534	0.598
VTH	0.731	0.643	0.811

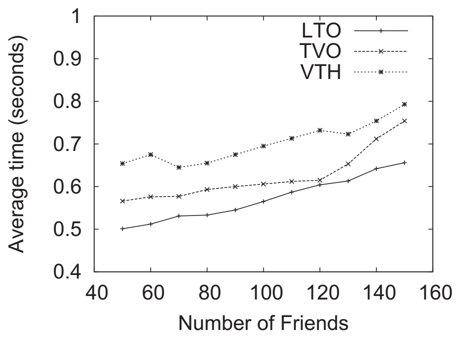

Fig. 1 – Average time for inference by number of friends.

8. Conclusions

In this paper, we have proposed an extensible fine-grained online social network access control model based on semantic web tools. In addition, we propose authorization, administration and filtering policies that are modeled using OWL and SWRL. The architecture of a framework in support of this model has also been presented. Further, we have implemented a version of this framework and presented experimental results for the length of time access control can be evaluated using this scheme. Further work could be conducted in the area of determining a minimal set of access policies that could be used in evaluating access requests in a further attempt to increase the efficiency of these requests.

Additionally, we have shown that existing social networks need some form of reasonable data partitioning in order for semantic inference of their access control to be reasonable in its speed and memory requirements, due to constraints on the memory available to perform inference. Additionally, further work can be used in determining the best method of representing the individual information of a person in a social network to determine if a hybrid semantic/relational approach or a pure approach offers the best overall system.

Acknowledgments

This work was partially supported by National Science Foundation Grants Career-0845803, CNS-0716424, and 0742477 as well as Air Force Office of Scientific Research MURI Grant FA9550-08-1-0265. Additionally, work reported in this paper is partially funded by the Italian Ministry of University, Education, and Research under the ANONIMO project (PRIN-2007F9437X 004).

REFERENCES

Ali B, Villegas W, Maheswaran M. A trust based approach for protecting user data in social networks. In: Lyons KA, Couturier C, editors. CASCON. IBM; 2007. p. 288–93.

Berteau S. Facebook's misrepresentation of beacon's threat to privacy: tracking users who opt out or are not logged in. CA Sec Advis Res Blog. Available at, http://community.ca.com/blogs/securityadvisor/archive/2007/11/29/facebook-s-misrepresentation-of-beacon-s-threat-to-privacy-tracking-users-who-opt-out-or-are-not-logged-in.aspx; 2007.

Brickley D, Miller L. FOAF vocabulary specification 0.91. Available at, http://xmlns.com/foaf/0.1; 2007.

Carminati B, Ferrari E, Heatherly R, Kantarcioglu M, Thuraisingham BM. A semantic web based framework for social network access control. In: Carminati B, Joshi J, editors. SACMAT. ACM; 2009a. p. 177–86.

Carminati B, Ferrari E, Perego A. Security and privacy in social networks. In: Encyclopedia of information Science and Technology. 2nd ed., vol. VII. IGI Publishing; 2008. 3369–3376.

Carminati B, Ferrari E, Perego A. Enforcing access control in web-based social networks. ACM Trans Info Sys Sec 2009b;13.

Choi HC, Kruk SR, Grzonkowski S, Stankiewicz K, Davids B, Breslin JG. Trust models for community-aware identity management; 2006. IRW2006/WWW2006 Workshop.

Consortium, W.W.W. Status for resource description framework (rdf) model and syntax specification. Available at, http://www.w3.org/1999/.status/PR-rdf-syntax-19990105/status; 1999.

Elahi N, Chowdhury M, Noll J. Semantic access control in web based communities; 2008:131–6. Computing in the Global Information Technology, 2008. ICCGI'08. The Third International Multi-Conference on.

Finin TW, Joshi A, Kagal L, Niu J, Sandhu RS, Winsborough WH, et al. R OWL BAC: representing role based access control in OWL. In: Ray I, Li N, editors. SACMAT. ACM; 2008. p. 73–82.

Horrocks I, Boley H, Patel-Schneider P, Tabet S, Grosof B, Dean M. Swrl: a semantic web rule language combining OWL and ruleml. Available at, http://www.w3.org/Submission/SWRL; 2004.

Kruk SR, Grzonkowski S, Gzella A, Woroniecki T, Choi HC. D-FOAF: distributed identity management with access rights delegation. In: Mizoguchi R, Shi Z, Giunchiglia F, editors. ASWC. Lecture notes in Computer Science, vol. 4185. Springer; 2006. p. 140–54.

Tonti G, Bradshaw JM, Jeffers R, Montanari R, Suri N, Uszok A. Semantic web languages for policy representation and reasoning: a comparison of KAoS, rei, and ponder. In: Fensel D, Sycara KP, Mylopoulos J, editors. International semantic web Conference. Lecture notes in Computer Science, vol. 2870. Springer; 2003. p. 419–37.

Yague MI, Gallardo María-del-Mar, Mana A. Semantic access control model: a formal specification. In: ESORICS: European Symposium on research in Computer security. LNCS, Springer-Verlag; 2005.

Barbara Carminati is an assistant professor in the Department of Information and Communication at the University of Insubria in Milan, Italy. She received her Ph.D. in Computer Science from the University of Milan in 2003. Prof. Carminati has served as the Editor and Chief of Computer Standards and Interfaces, guest editor for a special issue of Secure Semantic Web, and the editorial board for Computer Standards and Interfaces. Additionally, she has served as Program and General chair of ACM Symposium on Access Control Models and Technologies, as well as co-chair for the ICDE International Workshop on Secure Semantic Web.

Elena Ferrari is a professor of Computer Science at the University of Insubria, Italy, where she heads the Database & Web Security Group. She has been a visiting researcher at the Department of Computer Science of George Mason University, Fairfax (Virginia), Rutgers University, Newark (New Jersey). She received the IEEE Computer Society's 2009 Technical Achievement Award for "outstanding and innovative contributions to secure data management." Her research activities are related to aspects of data management systems, including web security, access control and privacy, multimedia databases, and temporal databases. She is a member of the ACM and senior member of IEEE.

Raymond Heatherly is a Ph.D. student of computer Science at the University of Texas at Dallas, working in the Data Security and Privacy research lab at the institution. Raymond has had publications in the IEEE Intelligence and Security Informatics, SACMAT, and WWW conferences.

Murat Kantarcioglu is an assistant professor of Computer Science at the University of Texas at Dallas and heads the Data Security and Privacy research laboratory. Prof. Kantarcioglu received his Ph.D. from Purdue University in 2005. Since then, Prof. Kantarcioglu has received the NSF Career award in 2009, as well as various research grants from the United States Air Force.

Dr. Bhavani Thuraisingham joined The University of Texas at Dallas in October 2004 as a Professor of Computer Science and Director of the Cyber Security Research Center in the Erik Jonsson School of Engineering and Computer Science. She is an elected Fellow the IEEE, AAAS, and the BCS for her work in data security. She received the IEEE Computer Society's 1997 Technical Achievement Award for "outstanding and innovative contributions to secure data management." Prior to joining UTD, Dr. Thuraisingham was an IPA (Intergovernmental Personnel Act) at the National Science Foundation (NSF) in Arlington, VA, from the MITRE Corporation.