Usable Access Control in Collaborative Environments: Authorization Based on People-Tagging

Qihua Wang[1,⋆], Hongxia Jin[2], and Ninghui Li[1]

[1] Department of Computer Science, Purdue University
[2] IBM Almaden Research Center

Abstract. We study attribute-based access control for resource sharing in collaborative work environments. The goal of our work is to encourage sharing within an organization by striking a balance between usability and security. Inspired by the great success of a number of collaboration-based Web 2.0 systems, such as Wikipedia and Del.icio.us, we propose a novel attribute-based access control framework that acquires information on users' attributes from the collaborative efforts of all users in a system, instead of from a small number of trusted agents. Intuitively, if several users say that someone has a certain attribute, our system believes that the latter indeed has the attribute. In order to allow users to specify and maintain the attributes of each other, we employ the mechanism of people-tagging, where users can tag each other with the terms they want, and tags from different users are combined and viewable by all users in the system. In this article, we describe the system framework of our solution, propose a language to specify access control policies, and design an example-based policy specification method that is friendly to ordinary users. We have implemented a prototype of our solution based on a real-world and large-scale people-tagging system in IBM. Experiments have been performed on the data collected by the system.

1 Introduction

Computer-supported collaborative work environment is gaining popularity in enterprises. Popular systems that support collaborative work environment include IBM's Lotus Connection and Microsoft's SharePoint Server. Such collaboration systems improve the efficiency of enterprises by building connections between employees, encouraging communications, and facilitating employees with different expertise to collaborate on multidisciplinary tasks.

Resource sharing is one of the most common activities in collaborative work environments. In most cases, the goal of resource sharing in collaborative work environments is to offer help or seek collaboration. For example, a senior engineer may want to share her proposal on a database project with her colleagues, so as to get feedback from people with expertise on the topic of her proposal. For another example, a product team may want to ask their colleagues who have experiences on product XYZ to download and test the alpha version of an add-on for XYZ developed by the team.

Resource sharing is oftentimes selective and thus requires access control. Traditional access control systems focus on limiting access so as to prevent sensitive information

[⋆] Part of this work was done when the author was an intern at IBM Almaden Research Center.

from leaking to unauthorized users. However, the importance of enabling and encouraging access is being increasingly realized. As stated in a report [5] from the Jason Program Office, "we also need new information security constructs so that the full value of the information can be realized by delivering it to the broadest set of users consistent with its prudent protection". A desired access control system should be able to maximize the benefit of resource sharing by granting access to all people who can potentially make good use of the resource, while maintaining the overall risk at an acceptable level.

Resources shared in collaborative work environments are normally not sensitive with respect to employees of the organization. However, this does not mean that we should simply allow everyone to access everything. In most organizations, the amount of shared resources is overabundant for any single user, and granting access to users who are not interested in the resource will not bring in additional benefit. Furthermore, even though the risk of allowing any single employee to access a shared resource is low, the aggregated risk is large when too many employees have access. Therefore, it is better to grant access rights in collaborative work environments based on potential interests and needs, even though every employee has sufficient security clearance for every resource. Similar practice is also found in real-world multi-level security (MLS) systems. In practice, even though MLS allows granting the access right of a piece of sensitive information to all users with the right security clearance, it is rarely the case that the right is indeed given to all such users. The access right is usually granted only to those users who need the information, so as to be compliant with the principle of need-to-know. But the existing approach for enforcing need-to-know based on compartments in MSL is too rigid for most resource sharing activities in enterprises.

To perform selective resource sharing in collaborative work environments, attribute-based access control (ABAC) is a natural choice, as resource owners usually seek help or collaboration from colleagues with certain expertise or interests. One of the most fundamental problems in ABAC is how to determine whether a user has a certain attribute. In existing work, users' attributes are certified by a limited number of trusted agents, such as certificate authorities and users trusted by the resource owner through certificate chains. Certificate authorities certify users' attributes using digital signatures, which can usually result in strong security assurance. However, in practice, the types of attribute that can be certified by certificate authorities are very limited. For instance, how can a user acquire a certificate to prove that she has expertise in database? Furthermore, maintaining certificate authorities in an organization is expensive, as dedicated human resource is usually required. Another common approach on attribute certification is to let the resource owner to determine which users have the required attributes. The resource owner may also issue certification statements to specify the users she trusts on determining certain attributes. And the trusted users may issue their own certification statements as well. A user is considered to have a certain attribute if there is a chain of certification statements on the attribute from the resource owner to the user. This approach does not have limit on the types of attributes that can be certified. However, it limits the scope of authorized users to those who have direct or indirect connections with the resource owner via certification statements. In large organizations with multiple sites, such a limitation may disqualify a large number of users who have the required attributes from accessing a shared resource and thus reduces the benefit of sharing. Furthermore, key management in such systems can be complicated. Finally, requiring resource owners to specify and maintain certification statements is a burden that makes resource sharing more difficult for ordinary users.

In this article, we explore the idea of identifying users' attributes through the collaborative efforts of all users in an organization. In our framework, any user can say that another user has a certain attribute. The opinions of all users are combined and shared with everyone in the organization. This is similar to reputation systems, such as the one on eBay, except that we allow users to evaluate others with a variety of attribute names rather than numerical scores. Intuitively, if several users say that someone has a certain attribute, our system believes that the latter indeed has the attribute and will make access control decisions based on such information. Our approach essentially replaces a small number of trusted agents in traditional solutions with the collaborative efforts of all users in the system to determine and maintain information on users' attributes. Even though any single user's opinion may not be reliable, the aggregated opinion of many users usually is.

Information acquisition and maintenance through user collaboration is one of the most important concepts in Web 2.0. The great success of Web 2.0 systems, such as Wikipedia and Del.icio.us, demonstrates that many people are indeed willing to honestly share their knowledge with others. No one is responsible to contribute a lot, but the small pieces of contribution made by everyone can be combined into a set of complete and up-to-date information. Every user in the community, including those who do not contribute at all, can take advantages of the combined set of information.

To our knowledge, our work is the first attempt to perform attribute-based access control using the collaborative efforts of all users in a system. Different from most existing literature on access control which focuses mainly on security, our goal is to encourage resource sharing by striking a balance between usability and security. Since our approach relies on general users rather than trusted agents to maintain information, not all the information used to make access control decisions is trustworthy. In particular, our approach may be vulnerable to the collusion of malicious users. But unlike reputation systems on internet, in enterprise environments, an employee can have only one user account. Hence, a single person cannot create multiple accounts for collusion purpose. Also, most employees of an organization should be honest, which makes it difficult to find colluding partners. Finally, a number of mechanisms may be employed to enhance the security of our approach. Since resource sharing in collaborative work environment does not involve highly sensitive materials, the security provided by our approach should be sufficient. Detailed discussion on security will be given in Section 4.

Our access control framework is for ordinary users rather than for security administrators only. Usability is thus an important factor in our system design. To allow users to easily specify the attributes of each other, we employ the mechanism of tagging. Tagging has gained popularity as a lightweight and flexible approach to classifying and retrieving information. It has been used to manage bookmarks (Del.icio.us), images (Flickr), and products (Amazon.com). Tagging has also been applied to describe people. For example, Fringe Contacts (or Fringe for short) [3] is a reference system designed to augment employee profiles with tagging in IBM. In Fringe, people are allowed to tag each other with terms they consider appropriate, and the tags one received from others are viewable on his/her employee profile. As of May 14, 2008, 53844 IBM employees have been tagged with a total of 170137 tags in Fringe. According to the data collected by Fringe, tags applied to a user usually describe the user's attributes, such as her affiliations, expertise, and the projects she has been involved in. The initial goal of people-tagging in Fringe is to help users to organize their connections and facilitate expertise search within IBM. For instance, if Alice has been tagged with "java" by many colleagues, she probably is an expert in Java. If Bob searches "java" in Fringe, all the users who have been tagged with "java" (including Alice) will be returned and ranked by the number of times they have been tagged with "java". In this article, we propose access control for resource sharing as another application that may be supported by people-tagging systems. Tagging systems have relatively low maintenance cost and most existing employee profiling systems can be easily modified to support people-tagging. Our solution can thus be applied in most enterprise environments.

The rest of this article is organized as follows. We will describe our access control system in Section 2, and details of access control policy specification will be given in Section 3. After that, we will discuss the security of our access control system in Section 4. We will present a user-friendly example-based policy specification method in Section 5. Implementation and experimental results will be discussed in Section 6. Finally, we will discuss related work in Section 7 and conclude in Section 8.

2 Access Control with People-Tagging

In this section, we describe our access control solution for resource sharing in collaborative work environments. The goal of our system is to allow a user to easily share a resource with other users in the same organization who have certain attributes. We consider common enterprise settings, where there is a one-to-one mapping between employees and user accounts. Our solution consists of a people-tagging system and a number of host servers.

In the people-tagging system, every user has a profile and users can tag each other with the terms they want. A *tag instance* is represented as a tuple $\langle u_1, u_2, t\rangle$, where u_1 and u_2 are users and t is a text term. The tuple $\langle u_1, u_2, t\rangle$ indicates that u_1 tags u_2 with the term t, where u_1 is called the *tagger* and u_2 is called the *receiver*. For example, the instance $\langle Alice, Bob, \text{“}java\text{”}\rangle$ indicates that Alice tags Bob with the term "java". Note that a user cannot tag another user with the same term more than once, e.g. Alice cannot tag Bob with "java" twice. But a user may be tagged with the same term by multiple users, e.g. Bob may be tagged "java" by five different colleagues. Tags applied to a user are combined and viewable on her profile.

Users may place the resources they would like to share on a host server, which is accessible by other users in the same organization. A host servers is responsible to control access to the resources on it.

Next, we present the outline of our solution. Assume that Alice would like to share a resource r with other users. Our solution consists of the following steps.

1. Alice uploads r to a host server s.
2. Alice specifies access control policy p_r for r on s. The policy p_r is based on tags.
3. Bob requests to access r on s.

4. The server s queries Bob's tags from the people-tagging system.
5. The server s evaluates Bob's tags against the access control policy p_r. If p_r is satisfied, Bob's request to access r is granted; otherwise, Bob's request is declined.

We will describe the specification and evaluation of access control policies in Section 3. The advantages of our solution are summarized as follows:

– Our solution takes advantages of the collaborative efforts of all users in a people-tagging system to determine users' attributes. It has been shown that, in most cases, the combined set of tags one received from others adequately describe oneself and tags are updated more frequently than other sources of personal information, such as homepages [3]. We can thus make appropriate access control decisions based on the comprehensive and up-to-date attribute information of users'.
– Our solution is easy to use by ordinary users. People-tagging systems already exist in some organizations. In people-tagging systems, applying tags to another user is as easy as typing a number of words on the user's profile. Resource owners do not need to issue and manage certificate statements before sharing. They can even completely rely on tags applied by their colleagues if they want, so that they do not need to do anything besides posting the resource and specifying an access control policy. This makes resource sharing a very easy task.
– The cost of maintaining a collaborative people-tagging system is small. Information on the people-tagging system is maintained by all users in the organization. No one is responsible for a large amount of work, and dedicated human resource is not necessary.
– Unlike solutions based on trusted certificate authorities, there is no limit on the types of attributes that are supported by our access control system. Any attributes may be used in access control policies as long as there are users having tags corresponding to such attributes.
– Unlike solutions based on certificate chains, in our system, all the users with the appropriate tags throughout the organization may be authorized to access a shared resource, even if they do not have connections with the resource owner. This maximizes the scope of sharing, which can potentially lead to more benefits and opportunities. Furthermore, we do not need to deal with complicated key management schemes for certificate chain management.

Relying on collaborative efforts to determine attribute information also has disadvantages. Our access control system may be vulnerable to the collusion of malicious users, who may apply inappropriate tags to each other so as to satisfy access control policies. Detailed discussion on the security of our system will be given in Section 4.

3 Access Control Policy Specification and Evaluation

We have described the outline of our solution. In this section, we describe the specification and evaluation of access control policy in detail.

An access control policy in our system is given as a tuple of six components

$$\langle \{e_1, \ldots, e_n\}, f, a, \langle L_b, L_w \rangle, k, \theta \rangle$$

where $\{e_1, \ldots, e_n\}$ is a set of expressions on attribute requirements, f is a tag filter, a is an option for approximate matching, and $\langle L_b, L_w \rangle$ is a pair of blacklist and whitelist.

Parameters k and θ together state the satisfaction condition of the access control policy: parameter k states the minimum number of attribute expressions that must be satisfied, and parameter θ states the desired number of qualified users. In the following, we discuss each component in an access control policy in detail.

Attribute Expressions. An access control policy contains a non-empty set of attribute expressions $\{e_1, \ldots, e_n\}$. An attribute expression e_i is in the following form

$$T_1 \ \wedge \ T_2 \ \wedge \ \ldots \ \wedge \ T_m$$

where T_i $(i \in [1, m])$ is an *atomic term* and $\wedge$ is the the conjunction operator. The expression is satisfied by a user who satisfies every atomic term T_i $(i \in [1, m])$.

An atomic term takes the form of $\langle t(n) \rangle$, where t is a text term (called the *attribute requirement* of the atomic term) and $n \geq 0$ is an integer (called the *quantity requirement* of the atomic term). An atomic term $\langle t(n) \rangle$ is satisfied by a user who has been tagged with t by at least n people. For example, the expression $\langle database(2) \rangle \wedge \langle security(3) \rangle$ is satisfied by any user who has been tagged with "database" at least twice and "security" at least three times. Note that a tagger may tag Alice with both "database" and "security". We do not require the taggers for different atomic terms in an expression to be different. For instance, if Alice is tagged with both "database" and "security" by Bob and Carl, and is tagged with "security" by Doris, she satisfies the expression $\langle database(2) \rangle \wedge \langle security(3) \rangle$.

Tag Filter. The resource owner may choose one of the following three tag filters to determine which tags in the people-tagging system will be considered during the evaluation of the access control policy. For convenience, we name the resource owner Alice.

- *Self*: Only consider the tags applied by Alice herself.
  When the tag filter *self* is used, tags applied by other users do not count and thus Alice has complete control on who may access her resource. This is essentially discretionary access control (DAC) and Alice has to manage the access control list in the form of tags by herself.
- *Friends*: Only consider the tags applied by Alice or by those users who have been tagged by Alice.
  In most cases, a user only tags the people he/she knows. When the tag filter *friends* is used, the tags applied by the users Alice knows are taken into account. This filter limits the sharing scope to those people who are not too "far away" from Alice in real world.
- *Aggregated*: All the tags in the people-system will be considered.
  This is the default selection. The tag filter *aggregate* takes full advantages of collaborative efforts of all users in the system, while the other two filters give Alice full or partial control on the sharing scope of her resource.

Approximate Matching Option. In tagging systems, users have the freedom to choose the words they like when tagging others. Different users may use different words to mean the same thing. A common practice is to use abbreviations instead of complete words as tags. For example, "sna" is short for "social network analysis" and both terms have been used as tags in Fringe. For another example, "de" is short for "distinguished engineer" and both are used in Fringe as well. Not considering synonyms in the evaluation of attribute expressions could make certain qualified users unable to access a shared resource.

Besides synonyms, certain words are closely related to each other. For instance, some users have been tagged with "db2" but not "database" in Fringe. Since DB2 is a database product of IBM, we say that "db2" and "database" are closely related words. If a resource is to be shared with users having expertise in database, then those users being tagged with "db2" should be granted access as well.

When the approximate matching option is selected, synonyms and closely related words are considered to be equivalent with each other during the evaluation of attribute expressions. For example, a user being tagged with "sna" by two users satisfies the atomic term $\langle$social-network-anaylsis$(2)\rangle$, while a user being tagged with "database" once and "db2" twice satisfies the atomic term $\langle$database$(3)\rangle$, if she receives the tags "database" and "db2" from three different users. Note the quantity requirement in $\langle$database$(3)\rangle$ is on the *number of people* who have the opinion that the user is related to database, and hence, a tagger applying both "database" and "db2" to the user is counted as one.

Synonyms and closely related words can be automatically discovered by the people-tagging system. We adopt the approach introduced in [10] to find such information in tags. The high-level idea is that if two words often appear in the tags applied to the same user, then they may be related to each other; if two words are close to each other syntactically, they may be related as well. The approach in [10] combines both statistical approach and syntactic similarity to find synonyms and closely related words in tags. Experimental results on the data collected by Fringe showed that the approach is effective [10].

In the prototype of our access control system, the resource owner can see the list of words that are considered to be related to any word she is using in an attribute expression. The resource owner may modify the list of related words, if she thinks the list found by the system is not correct. This allows the resource owner to have control on the evaluation of her access control policy when approximate matching is enabled.

When the approximate matching option is not selected, exact string matching will be used in the evaluation of attribute expressions. For example, "sna" will be considered different from "social network analysis" in that case.

Blacklist and Whitelist. A blacklist and a whitelist may be used for discretionary access control purpose. The two lists are empty by default. If the resource owner would like to prevent a specific user (such as an internal competitor) from accessing her resource, she may add the user to the blacklist; on the contrary, if the resource owner would like to grant access to a specific user, she may add the user to the whitelist. Users appearing in the blacklist (or the whitelist) are declined (or granted) access to the resource regardless of they meet the satisfaction condition of the access control policy or not. The two lists allow the resource owner to fine-tune her access control policy.

Number of Satisfied Expressions. A qualified user must satisfy at least k attribute expressions in the access control policy. The default value of k is one, which indicates that a user just needs to satisfy any one of the attribute expressions in the policy. In the default case, the attribute expressions in the policy can be viewed as being connected by disjunction operators.

Desired Number of Qualified Users. The parameter θ specifies the desired number of users the resource is shared with. It can take a value of ∞, (x, spec), or (x, req),

Input: Resource owner u_o, requestor u_r, and an access control policy p
Output: "Yes" or "No"

```
If $u_r$ is in the blacklist of $p$, return "No";
If $u_r$ is in the whitelist of $p$, return "Yes";
Retrieve the set of tags $T_r$ of $u_r$ from the people-tagging system;
Filter the tags in $T_r$ using the tag filter of $p$;
$counter = 0,\ sum = 0$;
For every attribute expression $e_i$ in $p$
    If $T_r$ satisfies $e_i$
        $score = 0$;
        For every atomic term $< t(n) >$
            $N(t)$ is the number of different people who tagged $u_r$ with $t$ in $T_r$;
            $score = score + \log N(t)$;
        $counter = counter + 1,\ sum = sum + score$;
If $counter < k$, return "No";
If $\theta == \infty$, return "Yes";
If $score$ is one of the $x$ highest among all users with respect to $p$, return "Yes";
Return "No";
```

Fig. 1. Outline of the algorithm on determining whether a user satisfy an access control policy. Logarithm is used in the formula "$score = score + \log N(t)$" to reduce the impact when a user has been tagged with t by a very large number of people.

where $x > 0$ is an integer. The default value of θ is ∞, which places no limit on the number of qualified users. When $\theta = (x, \text{spec})$, a relevant score is computed for every user who satisfies at least k of the attribute expressions in the access control policy. After that, all the users are ranked based on their relevant scores, and the top x users with the highest scores, at the time when the policy is specified, will be granted access to the shared resource. The case when $\theta = (x, \text{req})$ is similar, except that access will be granted if the requestor is among the top x users with the highest relevant scores, at the time when the access request is made. The algorithm on relevant score computation is given in Figure 1. Intuitively, if a user has been tagged with a term by many other users, then he/she must be well-known for the corresponding attribute; the better known a user is with regards to the attributes in an attribute expression, the higher score he/she has. This allows a resource owner to share only with the highly qualified users in the organization. For example, a resource owner may want to share her patent disclosure on content protection only with the thirty most well-known experts on cryptography in the organization.

The outline of the algorithm that checks whether a user satisfies an access control policy is given in Figure 1.

Next, we study the computational complexity of the algorithm on determining policy satisfaction. We show that the running time of the algorithm is linear when $\theta = \infty$. The discovery of synonyms and closely related words only needs to be performed once in a while and the results can be stored by the people-tagging system and/or host servers. We do not consider the complexity of the discovery algorithm here.

For convenience, we call the resource owner Alice and the user who requests to access the resource Bob. First, filtering Bob's tags using a tag filter takes time linear in the sum of the number of tags Alice applied to others and the number of tags Bob has received. Second, hashtables may be used to store users' tagging information so that we can retrieve how many times Bob has been tagged with a certain term in constant time. This indicates that checking whether Bob satisfies an attribute expression takes time linear in the size of the expression. Hence, determining whether Bob satisfies at least k attribute expressions takes time linear in the size of the access control policy, which indicates that the satisfaction problem takes linear time when θ is ∞. When θ is not ∞, we will need to compute the relevant scores of all users and sort them. When $\theta = (x, \text{spec})$, the computation of relevant scores and sorting just need to be performed once, and we may store the top x users in the whitelist at policy specification time. When $\theta = (x, \text{req})$, score computation and sorting need to be performed at request time. In our prototype of the system, we avoid sorting the qualified users upon every access request by storing the xth highest score among the qualified users as a threshold score after a sorting. Upon each request, instead of computing the top x users, we simply compare the requestor's score with the threshold score for access decision. Our prototype recomputes the threshold score once in a while. As users receive new tags over time, the top x users with regards to a policy may change as well. But dramatic changes is unlikely in a short period of time (say, within a couple of days). Our prototype trades some accuracy for better performance.

4 The Security of Collaboration-Based Access Control

In our access control system, the attribute information used to make access control decisions comes from general users instead of trusted agents. Our system is designed for enterprise settings, where every person can have only one user account and most people are honest. Even though no single user is fully trustworthy, the security of our approach lies in the fact that we do not rely on any single source of information but the aggregated information from multiple sources. Assume that resource r is guarded by an access control policy with a single attribute expression $\langle t(n) \rangle$. The quantity requirement n in the atomic term $\langle t(n) \rangle$ states that a user has to be tagged by at least n other users with t so as to satisfy the atomic term. The quantity requirement n in the atomic term has a similar spirit as separation of duty, which requires more than one users to involve in a sensitive task so as to prevent frauds. If a malicious user Eve without attribute t wants to access r, he has to find n other users to collude with him and tag him (inappropriately) with t. Note that Eve cannot have more than one user account in his organization. Since most employees of an organization are honest, it is very difficult for Eve to form a colluding group with a large number of users. Therefore, it is practically infeasible for Eve to bypass the access control policy when n is large.

However, many users with attribute t may not have been tagged many times. A large quantity requirement n may make the access control policy too strict by preventing many qualified users from accessing the shared resource. The resource owner needs to find a balance between security and the benefits of sharing. More qualified users having access to the shared resource may lead to more benefits, but this may require a small n, which makes the success of collusion more likely. In the following, we discuss how to detect collusion among malicious users in a people-tagging system so as to enhance the security of our access control system.

To circumvent an access control policy, a malicious user Eve, who does not have the required attributes, has to ask his colluding partners to tag him with the corresponding tags that do no match his background in real world. In people-tagging systems such as Fringe, all the tags applied to a user are viewable on his/her employee profile. Hence, Eve may not want the inappropriate tags to remain on his profile for too long, or his colleagues who know him personally may see those tags and question him about that. For example, Eve's manager may ask him why he has received several tags on a project that he is not supposed to be working on. In this case, Eve may need to remove the inappropriate tags soon after accessing the targeted resource so that others do not see those tags. However, playing such a trick several times will result in many short-living tags on Eve's profile, which is abnormal as a user's attributes do not change frequently (according to the data collected by Fringe, tags are rarely removed after applied). Therefore, if the people-tagging system detects that a user has much more short-living tags than average, it may alert system administrators about the abnormality. Furthermore, lots of research has been done on detecting inappropriate reviews in reputation systems. Some of those approaches could be applied to detect inappropriate tags in people-tagging systems. Detailed discussion on this is beyond the scope of this paper.

In general, an access control policy in our system may be vulnerable to collusion among malicious users when its quality requirements on tags are small. In contrast, the security of existing ABAC approaches that rely on trusted agents may be broken when a trusted agent is compromised. In many practical scenarios, compromising (or colluding with) multiple general users might not be easier than compromising a single trusted agent, especially when some agents trusted by the resource owner are virtually general users in the system. Therefore, relying on the collaborative efforts of general users does not necessarily result in less secure system than counting on trusted agents in practice.

Finally, the primary goal of our system is to encourage resource sharing. Tradeoff has to be made between convenience and security. The vulnerability to collusion seems to be an inevitable cost of relying on the collaborative efforts of all users instead of a few trusted agents to make access control decisions. Our system is thus most suitable for the sharing of resources that are not very sensitive with respect to employees in the same organization, where occasional security breaches is acceptable. As we have pointed out in Section 1, resource sharing activities in collaborative work environments normally do not involve highly confidential materials with respect to employees. With the advantages of user collaboration, our system is an excellent complement to existing access control schemes in collaborative work environments.

5 Example-Based Access Control Policy Specification

In Section 3, we have introduced the formal specification of access control policies in our system. However, in collaborative work environments, resource owners, who are responsible to specify access control policies, are ordinary users who may not have any expertise in formal policy specification. For many users, the formal specification introduced in Section 3 may not be easy to use. To enhance the usability of our system, we propose an example-based access control policy specification scheme to help resource owners with policy specification.

Intuitively, with example-based policy specification, a resource owner may give examples on users who should have access to the shared resource, instead of explicitly specifying what attributes are required. For example, a resource owner may say that the resource should be shared with users similar to Bob and Carl. Our system will then automatically extract important attributes from the example users and grant access to other users with the extracted attributes. The resource owner can also monitor which extracted attributes should be used in the access control policy. For many users, giving examples is more natural than formal specification. Our example-based specification scheme is also helpful when resource owners are having difficulties listing the required attributes. Oftentimes, people may not know exactly what they want, but they normally have examples in mind. Attributes extracted from examples may provide hints to the resource owners on what attributes should be included in the access control policies, which makes policy specification easier and less error-prone. In the rest of this section, we describe our example-based access control policy specification scheme in detail.

For convenience, we assume that the resource owner is Alice. Intuitively, for each example user Alice provides, we would like to find out the attributes that are important and special with respect to the example user. The results for all the example users will be combined to form a list of the most important attributes with regards to the set of examples Alice gives. Alice can then select which important attributes she would like to include in the access control policy, or an access control policy can be automatically created from the returned attributes using a default template. Our approach consists of the following steps:

1. Alice gives a set U_e of k $(k > 1)$ example users, where $U_e = \{u_1, \ldots, u_k\}$.
2. For each $u_i \in U_e$, we sort the words appearing in u_i's tags in descending order based on their *importance scores*.

   The importance score is a statistical measure used to evaluate how important a word is to u_i. The importance score of word w is computed as

   $$S(w, u_i) = N(w, u_i) \times \log \frac{|U|}{|\{u : w \in T(u)\}|}$$

   where $N(w, u_i)$ is the number of users who have tagged u_i with w, $|U|$ is the total number of users in the people-tagging system, and $|\{u : w \in T(u)\}|$ is the number of users who have been tagged with w. The computation of importance scores is analogous to that of the well-known TF-IDF weight (Term Frequency-Inverse Document Frequency) for keyword extraction of text documents. The importance score increases proportionally to the number of times u_i has been tagged with w but is offset by the frequency of w in other users' tags. Intuitively, if u_i has been tagged with w many times, it is likely that w describes an important property of u_i's. However, there may exist words that are very common among the tags of all users and such words are not so special to u_i and should thus be discounted. For example, more than three thousand Fringe users have been tagged with "work", so "work" is not very special to anyone.

   Finally, synonyms and closely related words in u_i's tags may be grouped together before the computation of importance scores. This avoids extracting several words with the same meaning from u_i's tags.

3. We sort the words appearing in any example user's tags in descending order based on their *group-importance scores* and return the top x $(x \geq 1)$ words in the sorted list to Alice.
   The group-importance score of word w with respect to U_e is computed as
   $$S_g(w, U_e) = \Sigma_{u_i \in U_e} S(w, u_i) \times |\{u_j : u_j \in U_e \wedge w \in T(u_j)\}|$$
   where $S(w, u_i)$ is the importance score of word w with respect to user u_i and $|\{u_j : u_j \in U_e \wedge w \in T(u_j)\}|$ is the number of example users in U_e who have been tagged with w. Note that $S(w, u_i) = 0$ if u_i is not tagged with w. A bonus $|\{u_j : u_j \in U_e \wedge w \in T(u_j)\}|$ is applied to favor common tags among users in U_e.
4. Alice may modify the list of words returned by the system, specify quantity requirements for each selected word, and construct attribute expressions.
   By default, an attribute expression will be constructed by connecting the selected words using conjunction operators and the default value of quantity requirement is one. And the default values of parameters k and θ are one and ∞, respectively.

The most expensive steps in the above algorithm are the computation of importance scores of users' tags and sorting those scores. The computational complexity of computing the importance scores of a user's tags is similar to extracting keywords from a text article, which can be done very efficiently with appropriate data structures. Our implementation and experiments showed that computational cost is not an issue for the example-based policy specification approach introduced in this section.

6 Implementation and Experimental Results

We have implemented a proof-of-concept prototype of our access control system for Fringe. Our prototype allows users to specify access control policies using either the language introduced in Section 3 or the example-based approach in Section 5. When a user specifies an access control policy with our prototype, she can see the set of users who are allowed access by the current draft of the policy anytime during the specification process, and she may make modification accordingly. Such an interactive approach helps users to design policies that are neither too strict nor too permissive.

We have performed experiments on our example-based policy specification approach using the data collected by Fringe, which contains 53844 users and 170137 tags. The goal of our experiments are two-folded:

First, we would like to evaluate the effectiveness of our approach in determining what are the important attributes that are likely to be used in the targeted access control policy, given a set U_e of examples on qualified users. In particular, we would like to compare the performance of our approach with a naive example-based approach that sorts words appeared in example users' tags based on the value of $\Sigma_{u_i \in U_e} N(w, u_i) \times |\{u_j : u_j \in U_e \wedge w \in T(u_j)\}|$, where $N(w, u_i)$ is the number of times u_i has been tagged with w and $|\{u_j : u_j \in U_e \wedge w \in T(u_j)\}|$ is the number of users in U_e who have been tagged with w. In other words, the naive approach replaces the importance score $S(w, u_i)$ with number of occurrences $N(w, u_i)$ in the formula. For simplicity, we call the example-based policy specification approach introduced in the last section EBPS, and the naive example-based approach introduced in this paragraph NAIVE.

Second, we would like to see how different factors may affect the performance of EBPS. The factors we considered in the experiments are the quantity and quality of examples, and the complexity of targeted access control policies. First, the more example users are given, the easier it is to identify their common attributes. The quality of the example users is important as well. Assume that w is a desired attribute the resource owner would like to see. Then, the quality of an example user u with respect to w is measured by the number of times u has been tagged with w. Intuitively, the more times the example users have been tagged with w, the more likely EBPS is able to identify w as a desired attribute. Second, the complexity of an access control policy is measured by the number of atomic terms in an attribute expression. Intuitively, the more attributes an attribute expression contains, the harder it is to correctly identify all of them from the examples, especially when the number of attributes returned by EBPS is fixed to a small value.

Test Method. We design a test method that allows us to evaluate the effectiveness of EBPS without the need of going over the returned attributes manually. The high-level idea of our test method is as follows. First, we generate an access control policy with a set of attributes. After that, we select a number of (2 or 3) users who satisfy the access control policy as example users. We then ask EBPS (or NAIVE) to guess what are the attributes used in the access control policy based on the example users. For each parameter setting, we run the test over different access control policies and different example users, and then we compute the average passing rates of both EBPS and NAIVE.

The detail of each step in a test is given in below:

1. We select a set A of attributes and then generate an access control policy p, whose only attribute expression consists of attributes in A.

   In practice, the number of attributes in an expression is usually small. We test the cases where $|A| = 1$ and $|A| = 2$. Other parameters of p, such as k and θ, take default values. We disable approximate matching so as make our test results independent of the performance of the algorithm for finding synonyms in [10].

   In our experiments, the 1000 most popular words in Fringe tags were used to generate testing policies. When $|A| = 1$, we enumerated all the 1000 access control policies, each of which contain one of the 1000 most popular words. When $|A| = 2$, we paired the 1000 words to create policies with two attributes, and we reduced the number of pairs by pairing any single word with at most 20 other words.
2. We compute the set U_s of Fringe users who satisfy p.
3. We select a set U_e of m users as examples, where $U_e \subseteq U_s$. We input U_e to EBPS (or NAIVE).

   In practice, a resource owner will not give too many examples. We test the cases where $|U_e| = 2$ and $|U_e| = 3$. There are also quality requirements on the users in U_e with respect to attributes in A. We test two cases where every user in U_e must have been tagged with every attribute in A at least 2 and 4 times, respectively. For each access control policy generated in Step 1, all possible sets of example users were tested in our experiments.
4. EBPS (or NAIVE) returns a set A_o of n attributes as the most important attributes with respect to U_e.

The more attributes we return to the resource owner, the more likely that all the desired attributes are included in the result. However, in practice, n cannot be too large, or the result will become distracting as it contains many attributes that are not needed. We test cases where $n \in \{2, 4, 6, 8\}$.

5. We verify whether $A \subseteq A_o$. If the answer is yes, EBPS (or NAIVE) passes the test case; otherwise, it fails the test case.

Experimental Results. Our experimental results are given in Figure 2. By comparing the results in Table (a) with those in Table (b) (similarly, compare Table (c) with Table (d)), we can see that EBPS had a higher average passing rate than NAIVE in every test setting, especially when the number n of returned attributes is small. This indicates that, given two or three examples on qualified users, EBPS is more effective in giving high rankings to those attributes that are actually used in the targeted access control policies.

EBPS performed very well when the testing access control policies contain only one attribute. More specifically, from Table (a) in Figure 2, EBPS had average passing rates over 0.9 in all four columns, even when $n = 2$. This indicates that for more than 90% of the time, the desired attribute was among the top two attributes that were returned by EBPS. When the testing policies contain two attributes, it becomes more difficult to identify both desired attributes correctly from the example users. From Table (c), EBPS had average passing rates between 0.4 and 0.6, when $n = 2$. This indicates that for around half of the time, the two attributes actually used in a testing policy were exactly the top two attributes returned by EBPS. When $n = 4$, EBPS had average passing rates between 0.74 and 0.95. In practice, such performance should allow resource owner to find the desired attributes to be used in their policies quickly most of the time.

n	E2 + Q2	E2 + Q4	E3 + Q2	E3 + Q4
2	0.9260	0.9196	0.9597	0.9636
4	0.9784	0.9864	0.9884	0.9972
6	0.9908	0.9958	0.9980	1.0000
8	0.9952	0.9983	0.9998	1.0000

(a) Passing rates of EBPS, when each testing policy contains one attribute

n	E2 + Q2	E2 + Q4	E3 + Q2	E3 + Q4
2	0.6254	0.7518	0.6761	0.8269
4	0.8376	0.9218	0.8810	0.9525
6	0.9196	0.9803	0.9573	0.9910
8	0.9662	0.9975	0.9850	0.9997

(b) Passing rates of NAIVE, when each testing policy contains one attribute

n	E2 + Q2	E2 + Q4	E3 + Q2	E3 + Q4
2	0.4050	0.5456	0.5212	0.6397
4	0.7421	0.8836	0.8462	0.9481
6	0.8742	0.9639	0.9521	1.0000
8	0.9414	0.9906	0.9872	1.0000

(c) Passing rates of EBPS, when each testing policy contains two attributes

n	E2 + Q2	E2 + Q4	E3 + Q2	E3 + Q4
2	0.2782	0.4003	0.3416	0.5127
4	0.6681	0.7480	0.7307	0.8110
6	0.8186	0.8996	0.8703	0.9509
8	0.9081	0.9824	0.9513	0.9894

(d) Passing rates of NAIVE, when each testing policy contains two attributes

Fig. 2. Experimental results of EBPS and NAIVE. The numbers in the first column of each table are the numbers of attributes returned by the corresponding approach. The names of the columns represent the values of test parameters, where "Ex+Qy" states that the number of example users is x and the minimum number of times an example user has been tagged with the attributes in the corresponding access control policy is y.

By comparing the values in different columns of Table (a) or Table (c) in Figure 2, we can find that the more example users are given and/or the higher quality the example users have, the higher passing rates EBPS has. In particular, the quantity and quality of example users are more important when testing policies contain two attributes than when they contain only one. For example, according to Table (c), there are significant differences on the passing rates between columns "E2 + Q2" and "E4 + Q4"; but the differences between the same pair of columns are not so impressing in Table (a).

As to the performance on running time, a test containing more than 19,000 test cases could be completed in about 6 seconds on a workstation with a 2.20GHz Intel Core 2 Duo CPU and 3GB of main memory. In other words, EBPS can return an answer for each test case in less than 0.3 millisecond on average, which is clearly fast enough. The 6-second does not include the time of loading the Fringe data from hard-disk to data structures in main memory at the very beginning of the test, which is a one-time effort and takes about 3 seconds.

To sum up, our experimental results on real-word data demonstrate that EBPS is both effective and efficient.

7 Related Work

Tagging in collaborative environments has attracted significant amount of interests in the research community [3,4,12,8,10,9]. To our knowledge, the notion of people-tagging was first introduced in [3], where Farrell and Lau introduced the first people-tagging system, Fringe Contacts. The initial goal of people-tagging is to help users in enterprise environments to organize their contacts and search for experts in different fields. In a subsequent article, Farrell et al. [4] reported their findings from user survey and interviews, and the results demonstrated the effectiveness and a variety of advantages of people-tagging in Fringe.

Recently, researchers began to explore applications on people-tagging. Razavi and Iverson [8] studied using people-tagging to perform information sharing. In their approach, a resource owner may apply tags to others and say that only those who have been tagged with a certain term (say, "friends") by herself are allowed to access her certain information. Tags applied by users other than the resource owner do not count in the evaluation of access control policies for that owner's resources. In this case, their scheme does not make use of the collaborative efforts of different users. In their scheme, a resource owner still has to select the right people and maintain the selected sets of people (through tagging) by herself. They essentially implemented discretionary access control (DAC) with tagging mechanism. Wang and Jin [9] proposed to use people-tagging to selectively distribute messages. Their system infers people's interests based on their tags in a people-tagging system, and sends messages to those people who are likely to be interested in the topic of the messages. Similar to our work, their approach takes advantages of the collaborative efforts of all users in the system to acquire and maintain user information. Neither [8] nor [9] proposed a formal language on tag-based policy specification as the one in Section 3 of this article. They did not propose example-based policy specification on people-tagging either.

A lot of work has been done on systems that enable individual users to sharing their resources over the web easily and securely. In one of the recent work [7], Mannan and van Oorschot proposed a scheme to enable users to selectively share their information with the help of Instant Messaging (IM) networks. Discretionary access control (DAC) is employed in their system, as a user has to manually select the people to share with. And the scope of sharing is limited to one's IM contacts.

Finally, our work is related to trust management. A large number of languages and frameworks have been proposed for trust management [2,1,6,11]. But most of the languages and frameworks do not support quantity requirements on the aggregation of certificates. In [11], West et al. proposed a quantitative trust management system, which combines elements from trust management and reputation management. Different from our system, their system does not support the aggregation of collaborative efforts of general users. Authorized users still have to be connected to the resource owner via certificate statements in their system. To our knowledge, none of the existing trust management literature has proposed a user-friendly policy specification method that is similar to our example-based approach.

8 Conclusion

We have proposed a novel access control system for resource sharing in collaborative work environments in enterprises. Different from existing work in literature, our system utilizes the collaborative efforts of all users in the system instead of a small number of trusted agents to identify users' attributes. Our system is easy to use, has no limit on supported attributes, provides comprehensive and up-to-date attribute information, and has low maintenance cost. We have designed a formal language as well as a user-friendly example-based scheme for access control policy specification. We have also built a prototype of our system and performed experiments on Fringe data.

References

1. Blaze, M., Feigenbaum, J., Ioannidis, J., Keromytis, A.D.: The KeyNote trust-management system, version 2. IETF RFC 2704 (September 1999)
2. Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized trust management. In: Proceedings of the 1996 IEEE Symposium on Security and Privacy, pp. 164–173. IEEE Computer Society Press, Los Alamitos (1996)
3. Farrell, S., Lau, T.: Fringe contacts: People-tagging for the enterprise. In: WWW 2006: Collaborative Web Tagging Workshop, Edinburgh, Scotland (2006)
4. Farrell, S., Lau, T., Nusser, S., Wilcox, E., Muller, M.: Socially augmenting employee profiles with people-tagging. In: Proceedings of the ACM Symposium on User Interface Software and Technology (UIST), pp. 91–100. ACM Press, New York (2007)
5. Jason Program Office. Horizontal Integration: Broader Access Models for Realizing Information Dominance. The MITRE Corporation (December 2004)
6. Li, N., Mitchell, J.C., Winsborough, W.H.: Design of a role-based trust management framework. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy, pp. 114–130. IEEE Computer Society Press, Los Alamitos (2002)
7. Mannan, M., van Oorschot, P.C.: Privacy-enhanced sharing of personal content on the web. In: WWW 2008: Proceeding of the 17th international conference on World Wide Web, pp. 487–496. ACM Press, New York (2008)

8. Najafian Razavi, M., Iverson, L.: Supporting selective information sharing with people-tagging. In: ACM Conference on Human Factors in Computing Systems (CHI) (Work-in-Progress), pp. 3423–3428. ACM Press, New York (2008)
9. Wang, Q., Jin, H.: Selective message distribution with people-tagging in user-collaborative environments. In: ACM Conference on Human Factors in Computing Systems (CHI) (Work-in-Progress), pp. 3423–3428. ACM Press, New York (2009)
10. Wang, Q., Jin, H., Nusser, S.: Automatic categorization of tags in collaborative environments. In: Proceedings of the International Conference on Collaborative Computing (Cllaborate-Com), ICST (2008)
11. West, A.G., Aviv, A.J., Chang, J., Prabhu, V.S., Blaze, M., Kannan, S., Lee, I., Smith, J.M., Sokolsky, O.: Quantm: a quantitative trust management system. In: EUROSEC 2009: Proceedings of the Second European Workshop on System Security, pp. 28–35. ACM Press, New York (2009)
12. Xu, Z., Fu, Y., Mao, J., Su, D.: Towards the semantic web: Collaborative tag suggestions. In: WWW 2006: Collaborative Web Tagging Workshop, Edinburgh, Scotland (2006)