Some Usability Considerations in Access Control Systems

[Position Paper]

Elisa Bertino
CERIAS and Dep. of CS
Purdue University
West Lafayette, IN, USA
bertino@cs.purdue.edu

Seraphin Calo
IBM
T.J. Watson Research Center
Hawthorne, NY, USA
scalo@us.ibm.com

Hong Chen
CERIAS and Dep. of CS
Purdue University
West Lafayette, IN, USA
chen131@cs.purdue.edu

Ninghui Li
CERIAS and Dep. of CS
Purdue University
West Lafayette, IN, USA
ninghui@cs.purdue.edu

Tiancheng Li
CERIAS and Dep. of CS
Purdue University
West Lafayette, IN, USA
li83@cs.purdue.edu

Jorge Lobo
IBM
T.J. Watson Research Center
Hawthorne, NY, USA
lobo@us.ibm.com

Ian Molloy
CERIAS and Dep. of CS
Purdue University
West Lafayette, IN, USA
imolloy@cs.purdue.edu

Qihua Wang
CERIAS and Dep. of CS
Purdue University
West Lafayette, IN, USA
wangq@cs.purdue.edu

ABSTRACT

Role-based access control is one the most popular models adopted in commercial security and identity management products. However creating and maintaining such systems have been proven to be not an easy task. In this paper we review several issues that affect the usability of RBAC systems and discuss the advantages and limitations of role mining, a popular topic in the research community, as mechanism to simplify usability.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Access Control*; D.2.2 [**Design Tools and Techniques**]: User Interfaces

General Terms

Security, Management

Keywords

RBAC, role engineering, role mining

1. INTRODUCTION

Grouping users is a methodology that emerges naturally in access control management. The idea is that instead of assigning directly privileges to users or user accounts an administrator creates groups of users that may need similar privileges, assigns the privileges to the groups and group membership will determine which privileges get assigned to the user. Thus, with a single assignment of a privilege to a group or a revocation of the privilege from the group all members of the group are affected. This concept forms the backbone behind the Role-based Access Control Model (RBAC), formalized first in [3] and recently standardized by ANCI/INCITS [1]. The basic RBAC model is depicted in the figure 1. Groups in RBAC are called roles. The arrows depict many-to-many relationships. The RBAC model is more than just grouping of users into roles. When a user wants to access a resource the user needs to create a session and activate a role that let them access the resource through the session. In RBAC the administrator can impose constraints on the roles that can simultaneously become active in the same session. This allows RBAC systems to model semantic constraints such separation of duty or Chine walls [9] which are fairly common requirements in access control. However in this article we will ignore anything related to session management and we will limit our discussion to the management of roles (i.e. groups of users with the same

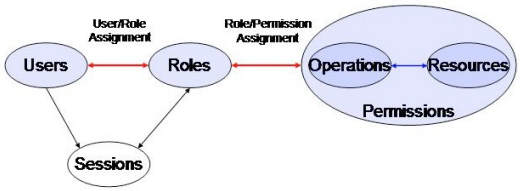

Figure 1: Role-based Access Control (RBAC) Model.

Symposium on Usable Privacy and Security (SOUPS) 2008, July 23–25, 2008, Pittsburgh, PA USA
.

access permissions).

Role-based access control (RBAC) is widely used in enterprise security management and enterprise identity management products. Take one of the popular management products, IBM Tivoli Identity Manager (ITIM), for example. ITIM allows centralized management of user accounts on a variety of systems and applications. In ITIM, accounts cannot be assigned to users directly; they must be provisioned to roles and roles are assigned to users. In a mid size enterprise with a few thousands employees we can easy find hundreds of roles and resources, many times roles matching the organizational structure of the enterprise. In large enterprises we find thousands of roles and resources. IBM marketing research has shown that RBAC systems "are creating both a valid Return On Investment (ROI) and driving better control over the assets of an organization" [2]. Attracted by strong ROI, more and more companies are driven to migrate to RBAC. However, for most companies, creating an RBAC configuration from scratch is not easy. According to a study by NIST [4], building an RBAC system is the costliest part of migrating to an RBAC implementation. Any improvement on methodology that can reduce the cost of RBAC system creation will further improve the ROI of RBAC and will accelerate RBAC's adoption in practice.

There are two general approaches to construct an RBAC system: the *top-down* approach and the *bottom-up* approach. In the top-down approach, people perform a detailed analysis of business processes and derive roles from such analysis. Since such a top-down analysis is human-intensive, it is believed to be slow and expensive. To overcome the drawback of top-down approaches, researchers have proposed to use data mining techniques to discover roles from existing system configuration data. Such a bottom-up approach is called *role mining*. Role mining can potentially accelerate RBAC system construction to a great extent, and it has raised significant interests in the research community [5, 8, 11, 10, 12].

However, the practical value of role mining is a controversial topic. The major argument against role mining is that *existing role mining approaches cannot discover roles with real-world meanings*. Roles that are discovered by existing role mining approaches are no more than a set of permissions and it is unclear whether such roles correspond to any real-world concepts, such as a job position or a work location. Without semantic meanings, such roles may be hard to use and maintain in practice. In addition, role mining in its current conception does not help in the day to day management of RBAC systems. System managers need to add, remove and modify users, roles and resources on regular basis, but role mining techniques have been developed assuming a static environment.

While some may believe that the top-down approach is more desirable despite of higher cost, because it produces higher quality results, the configuration data we gathered from ITIM shows that this is not always the case. These pieces of configuration data reveal the role-permission assignment of a number of real-world organizations that use ITIM as identity management solution. Some of these configurations are poorly designed. In an extreme case, a company created one role (and occasionally two roles) for each of the 486 permissions in the system, which results in 489 roles in total. With such an almost one-to-one correspondence between roles and permissions, the company can hardly enjoy the advantages of RBAC. Some other configurations are unnecessarily complicated due to redundancies. Some roles and permission assignments can be removed from the configuration without affecting the privileges of anyone.

There are several reasons that top-down approaches may sometimes fail to produce "good" RBAC systems in practice. First, building an RBAC system is challenging. It is common for people to adopt some trivial design, such as blindly creating one role for each job position of the organization regardless whether these job positions share the same set of permissions. Second, some system designers have been deeply influenced by Discretionary Access Control (DAC). When they are asked to construct an RBAC system, they tend to do it in a DAC manner, such as creating a role for each permission, thinking that the most flexibility is provided in that way. This is also affected by the fact that RBAC systems evolve all the time. Administrators address requests for access rights on demand and it is easier to create a new role each time since there are no mechanisms to visualize the current state of the system to find out if and how the new request may fit into the current RBAC system state. Third, many organizations do not have expertise in designing RBAC systems. They do not know what is a "good" RBAC system. Even though some companies, such as Eurekify, offer consulting and technical services on role management, such services are costly, and some companies consider their internal structure confidential and are reluctant to reveal this information to a third party. In fact, *there is no standard or accepted metrics to evaluate the goodness of an RBAC system with respect management and usability.*

In general, the data we acquired demonstrates that some organizations are having difficulties in designing an efficient and easy-to-manage RBAC systems by themselves using top-down approaches. Automatic tools that can help with RBAC system design have great commercial values, as more and more companies are considering RBAC implementation.

We believe that effective role mining tools will provide valuable help to role engineering and is complementary to the top-down approach of role engineering. For companies that do not have sufficient RBAC expertise, tools provide inexpensive help (comparing to hiring external consultants) and avoid having to leak sensitive organization information to outsiders.

To design a tool that can help, we need to make good use of all the information that is available. Those companies that are trying to migrate to RBAC have an access control system that is running fine for their business. Access control information can be gathered from this system and use data mining techniques, i.e. role mining, to help create an RBAC system from current access control configurations. However, as we have pointed out earlier, existing role mining approaches fail to discover roles with real-world meaning, which severely impairs the practical values of role mining.

Given the great success of data mining techniques in discovering meaningful information in areas such as marketing, forecasting and economics, it is reasonable to believe that they can be applied to discovering meaningful roles. We believe that there are two main reasons why existing role mining approaches fail to discover roles with semantic meanings. First, researchers have yet to find the right data mining techniques for role mining. Techniques that have been applied, including permission clustering and find-

ing frequent permission sets, focus on grouping permissions. For role mining, one needs to look at grouping permissions and users at the same time. Second, existing role mining problem definitions use only user-permission assignment information. Since usernames and permission names are both symbols without meanings, this limits one's ability to identify meaningful roles. Third, there is no formal definition on the meaning of roles to guide the design of role mining algorithms. Roles discovered by existing role mining approaches are sets of permissions. Without a formal notion of role semantics, an algorithm cannot distinguish more meaningful roles from less meaningful ones.

In general, existing work has built a foundation on role mining by studying a simplified model of the problem. However, role mining is more than discovering frequent itemsets from user-permission associations. A practical role mining approach must be able to justify the meaning of the roles it discovered. To achieve this, a formal notion of role semantics is required and we need to introduce more information that is widely available in practice into the role mining model. Designing a practical role mining approach is challenging but rewarding.

Finally, we need to reiterate that RBAC systems are not static systems. Once an RBAC system is built and put into use, we will need to maintain it. Overtime, an RBAC system is updated to meet the changes on access needs in an organization. For example, new employees and new applications (which bring in new permissions) will be added into the system, and existing employees may leave or change positions. When the initial RBAC configuration becomes bulky and inefficient after being used for a while, as a result of many updates, we may reconsider its structure. How to handle access control updates and how to improve an existing RBAC system without performing a complete reconstruction are important research topics that will improve usability of RBAC implementations. The study of role mining is one tool. Role engineering, which consists of both the construction and the maintenance of RBAC systems, is a rich area with a lot of practical and interesting problems for researchers to explore.

In the next section we briefly review how role mining techniques can be used to increase the usability of RBAC systems. In Section 3 we discuss the limitations of using only mining techniques and point other approaches that can be used to improve usability to complement role mining. Some final remarks on the usability of RBAC can be found in Section 4.

2. A ROADMAP FOR ROLE MINING

The general area of using data mining techniques for role engineering is a very rich area. Many challenging and practical problems exist. We now give a roadmap describing these research problems. To come up with the roadmap, we examine two dimensions. The first dimension is what kinds of data are available for mining, and the second dimension is what problems one aims to solve using data mining techniques.

We first look at the data dimension. At the bare minimum, one would have *user permission* information, that is, the set of users, the set of permissions, and the binary user-permission relation. In some cases, one also has *user attribute* information, e.g., a user' job title, the department and location a user is in. Often times, one also has *permission parameter* information, which is similar to user attribute information, but is for permissions. For example, a number of permissions may be about the same enterprise information management application. Or a permission may entail accounts on machines in one domain. In some systems, one may have *permission update* information, i.e., from logs that record how the access control state has evolved in the past. For example, a log entry may record, at a certain time in the past, a user was assigned a number of permissions soon after the user was revoked certain permissions. This piece of information would be useful for role mining because it may reflect a job position change event. Finally, one may have *permission usage* information. For example, one may have logs showing which permissions are used and at what time.

Now we look at the problem dimension. The first problem that naturally comes up is to mine an RBAC state (i.e., roles, role hierarchy, role-permission assignments, and user-role assignments) while optimizing some complexity measure. The second problem is to mine roles with good *semantic meanings*, i.e., roles that correspond to real-world concept units, e.g. a role for lecturers in the CS department. A similar problem is to construct *parameterized roles* that correspond to categories of concepts. For example, we may create a role for lecturers with the course name as a parameter. Finally, an access control configuration may contain *noise or outliers*. For example, one may find that a permission or a role has been assigned to all but one users in the same department. It would be good if such outliers can be discovered and reported to the administrator for investigation to discover and correct potential authorization errors.

By combining the data dimension and the problem dimension, we have a picture on what problems can be solved (or partially solved) with different data availability. A summary of the discussion below is given in Table 1.

With user permission information only. With such limited information, the only problem that could be satisfactorily solved is creating an RBAC system that is equivalent with the input user-permission relation while having minimum system complexity. In the literature, people have studied optimization on number of roles and on number of edges (user-role assignment and permission-role assignment). In [7], we introduced the notion of *weighted structural complexity* as a more general system complexity measure. The measure is a linear combination of numbers of roles, assignments and role hierarchies. Furthermore, it is possible to identify potential outliers by discovering association rules, such as 99% of the users who have permission p_1 also have permission p_2. However, without additional information, such as user-attribute information, both false positive and false negative ratios could be high.

With also user-attribute information. User-attribute information is a valuable plus for mining roles with semantic meanings. Intuitively, members of a role that corresponds to a real-world concept should share some attributes. Also, as mentioned earlier, user-attribute information can help better identify outliers, because in addition to comparing permissions, we can now compare attributes.

With also permission-parameter information. In many situations, permissions may be parameterized. E.g., instructor of a course, advisor of a student, permission about a database, permission about a file, permission about a directory, etc. This information enables the discovery of parameterized roles, especially when combined with user-attribute

	Low Complexity	*Good Semantics*	*Parameterized Roles*	*Least Privilege*	*Detect Outliers*
User Permission Only	✓	`Limited`			`Limited`
With User-Attribute	✓	✓			✓
With Permission-Parameter	✓	✓	✓		✓
With Update Log	✓	✓			✓
With Usage Log	✓	✓		✓	✓

Table 1: Summary of potential role engineering problems with different information availability. "✓" indicates that the corresponding problem is worth studying and a good solution to the problem is possible; "`Limited`" indicates that a solution could be provided for the problem, but the solution may be limited without more information; an empty cell indicates that the provided information is insufficient to study a problem.

information. Using parameterized roles could greatly reduce the number of roles in a system.

With also permission update information. Permission update information can help create roles with semantic meanings. For example, those permissions that often change together are probably associated with the same real-world concept. Update information also provides evidence on legacy permissions, i.e., permissions that should have been removed earlier.

With also permission-usage information. Permission usage data may be helpful for finding roles as well. For instance, permissions that are used together are likely to be associated with the same role. Also, for systems that require role activation, it may be desirable to group commonly-used permissions and rarely-used permissions into separate roles so as to enforce least privilege while minimizing the number of roles one has to activate for daily tasks. Finally, usage information also contributes to the detection of legacy permission assignments and erroneous permission assignments. For example, if a permission has never been used or has not been used for a long time by a user, then the permission assignment may be unnecessary.

3. MANAGING EVOLVING AND LEGACY RBAC SYSTEMS

The problems discussed above are mostly related to RBAC system creation. Another important task in role engineering is RBAC system maintenance. While the techniques for generating an RBAC state is useful for migrating to RBAC, it may be limited when the goal is to improve an already existing RBAC state. In an already existing and evolving RBAC systems one needs to deal with unnecessary user-to-role assignments, role-to-permission assignment and the proliferation of roles.

There are two types of sources for the unnecessary assignments. Since creating an RBAC system either manually using top-down techniques or automatically using bottom-up mining techniques is an empirical process the information is imprecise and redundant roles can be created. The opposite effect may also happens, that is, roles might not be created during the initial design but they might be needed latter. Having to deal with users with insufficient permission could be costly. Hence sometimes permissive design with excessive roles and roles assignments are done generating excess of assignments. The second source of unnecessary assignments may come from changes in the organization. Job positions change, task requirements change and projects finish. All these changes leave behind legacy assignments.

Role proliferate happens for similar reasons. New jobs are created or new projects are started and without the appropriate tools it is easier for the administrator to create new roles that fit the new requirements rather than looking for existing roles that might fit the assigments. Also employees leave and projects are completed and roles may be left behind without users.

Given a messy RBAC state resulted from a long time of usage, the administrator is unlikely to completely reconfigure the RBAC system running a role mining tool. It is thus useful to develop techniques that do two things: (1) Given an RBAC state, come up with an optimization that updates the RBAC state in some "localized way". (2) Given an RBAC state and a update request (e.g., changing a user's permission from one set to another), come up with a suggested update to the RBAC system so that the accumulated results of multiple updates will not lead to a messy state that is difficult to "understand" and manage.

All these management tool suggestions can be considered static tools, i.e. given a state of an RBAC system tools are ran to improve the state. However, simplifications can also be driven by permission usage. We can keep logs on how users use their permission and develop tools in which decisions and recommendations can be justified by usage. We can remove permissions from permission-to-role assignments if the permissions have been never exercised by user in that role or merge roles whose permissions are often used in tandem. Logs can also be used to clean-up roles that have been not activated for a long time.

We are not aware of any RBAC system that supports dynamic management tools.

4. FINAL REMARKS

The simplicity of the group and role concept makes it pervasive in almost any access or identity management system, but volume, i.e. the presence of hundreds or thousands of roles and resources, makes the management of role or group-based systems very hard. Thus, designing domain independent usability techniques for the management of roles and groups for access control will have high practical impact.

Perhaps the mere simplicity of the concept has misled to believe that groups and roles are easy to manage and very little effort has been put into the topic. Good visualization techniques are badly needed. Recently IBM has developed a graph-based interface for ITIM. A partial view of such interface is shown in Figure 2. This is an ECLIPSE plugin. The left panel of the window shows the organization structure of the enterprise and can be used to define the scope of the RBAC rules. An administrator can travel this structure to find RBAC associations. The graph on the right shows users towards the left, roles in the middle and permissions

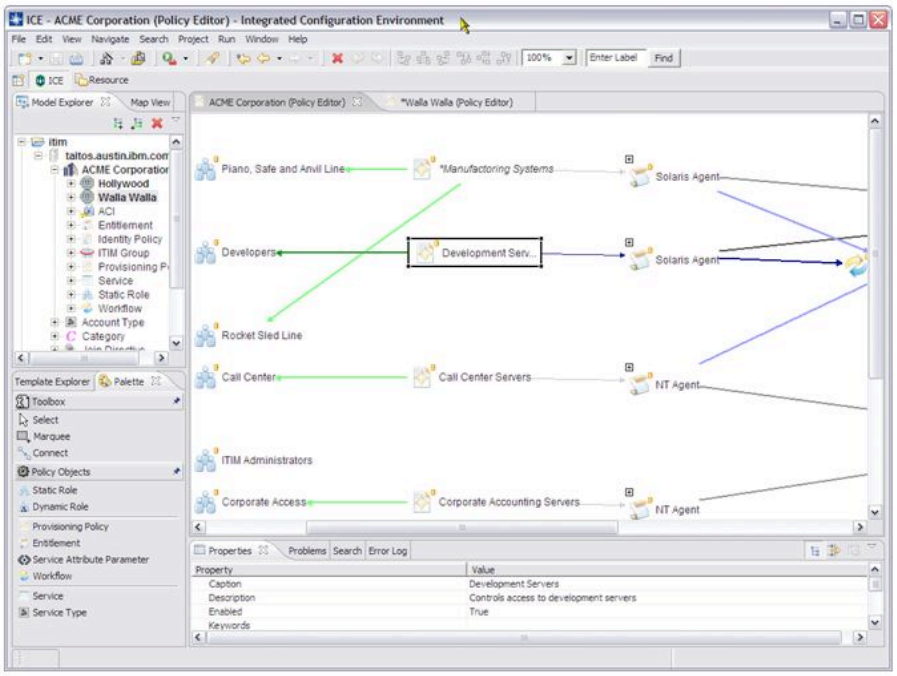

Figure 2: Graphical Configuration Editor for ITIM.

on the right of the organization selected on the left. This interface complements the original ITIM menu based interface where information is shown in different panels in list form through a web interface. The graphical interface still has its limitation because of the limited real-state available on the screen to show a large graph. Better graph rendering techniques specialized for the application (i.e. access control) or other types of visualizations that are able to handle the large volume of roles are needed.

Most likely we will need a combination of top-down and bottom-up tools, static and dynamic but the research community needs to come up with metrics that let us evaluate the quality of RBAC system designs. Current metrics only measure structural complexity and it is not obvious that simpler structures will lead to better management. New metrics should be developed and most include how easy is to management a system over it life time.

5. ACKNOWLEDGMENTS

This work has been partially supported by the IBM OCR project “Privacy and Security Policy Management”.

6. REFERENCES

[1] ANSI/INCITS. Information technology - role based access control, 2004.

[2] A. Bucker, A. Camp, R. Cohen, D. Edwards, C. Penman, and T. Santana. *Identity Management Design Guide with IBM Tivoli Identity Manager*. IBM, 2st edition, 2003.

[3] D. Ferraiolo and R. Kuhn. Role-based access controls. In *Proceedings of 15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.

[4] M. P. Gallaher, A. C. O’Connor, and B. Kropp. The economic impact of role-based access control, March 2002.

[5] M. Kuhlmann, D. Shohat, and G. Schimpf. Role mining - revealing business roles for security administration using data mining technology. In *SACMAT*, pages 179–186. ACM, 2003.

[6] V. Lotz and B. M. Thuraisingham, editors. *SACMAT 2007, 12th ACM Symposium on Access Control Models and Technologies, Sophia Antipolis, France, June 20-22, 2007, Proceedings*. ACM, 2007.

[7] I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. B. Calo, and J. Lobo. Mining roles with semantic meanings. In I. Ray and N. Li, editors, *SACMAT*, pages 21–30. ACM, 2008.

[8] J. Schlegelmilch and U. Steffens. Role mining with orca. In E. Ferrari and G.-J. Ahn, editors, *SACMAT*, pages 168–176. ACM, 2005.

[9] R. Simon and M. E. Zurko. Separation of duty in role-based environments. In *CSFW*, pages 183–194. IEEE Computer Society, 1997.

[10] J. Vaidya, V. Atluri, and Q. Guo. The role mining problem: finding a minimal descriptive set of roles. In Lotz and Thuraisingham [6], pages 175–184.

[11] J. Vaidya, V. Atluri, and J. Warner. Roleminer: mining roles using subset enumeration. In A. Juels, R. N. Wright, and S. D. C. di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 144–153. ACM, 2006.

[12] D. Zhang, K. Ramamohanarao, and T. Ebringer. Role engineering using graph optimisation. In Lotz and Thuraisingham [6], pages 139–144.