2001

A Fresh Look at Small-Granularity Role-Based Access Control

A.B Ruighaver
University of Melbourne, tobias@dis.unimelb.edu.au

A Fresh Look at Small-Granularity Role-Based Access Control

A.B. Ruighaver

Department of Information Systems
University of Melbourne, Melbourne, Australia
tobias@dis.unimelb.edu.au

Abstract

Small-granularity role based access control offers an effective solution to reduce the damage an intrusion can cause to your organization. We describe a new dynamic activation of roles, with automatic de-activation if the role is no longer used. This allows us to further decrease the granularity of roles, and use the roles as input to an anomaly based Intrusion Detection System. To show how easy it can be to add Role-Based Access Control to an existing platform, we briefly discuss a simple implementation for a WWW based Intranet.

Keywords

Access Control, Data Security, Distributed Systems

INTRODUCTION

More and more organizations are utilizing WWW technology to implement their corporate Information Systems. Current Intranets and Extranets are generally build using several HTTP based information servers, but may involve more complex JAVA and CORBA based distributed applications as well. The security implications of this new trend are almost never seriously considered. The push to leverage the latest technology to stay competitive is simply too strong, and the general believe is that a good firewall will provide adequate protection to prevent misuse of these systems from perpetrators from outside the organization.

In reality, the extensive use of WWW based technologies has created many new vulnerabilities [Ruighaver, 1996] and organizations are finding it more and more difficult to protect their corporate systems from Internet based attacks. In addition, it should be realized that often the more serious, and most costly, intrusions are not coming from outside the organization. Providing the same, easy to use, interface to all your internal information systems has made it easier for a disgruntled employee to wreak havoc. Where it used to be, that only a few of the employees were considered to be in vulnerable positions, now any employee potentially has the necessary skills and opportunities to try how far he/she can exploit the system.

With the realization that full prevention of intrusions is almost impossible and too costly, research in computer security is shifting from pure prevention to early detection of intrusions and/or ways to limit the damage/exposure resulting from such an intrusion. In this paper we discuss an implementation of role-based access control (RBAC), aimed at providing the security administrator with the necessary tools to achieve both these goals.

First we will discuss the concept of Role-Based Access Control, followed by a description of our extension of small-granularity role based access control with dynamic activation and automatic de-activation of roles. Then we explain the essentials of our anomaly based Intrusion Detection System and show how the combination of Intrusion Detection and small-granularity Role Based Access Control can help to reduce the exposure of a system to an unauthorized user masquerading as an authorized user. We conclude with a brief description of a prototype we developed for use in WWW based Intranets.

ROLE-BASED ACCESS CONTROL

Password based access control is the most common security service employed to protect the valuable information on our computer systems. A difficult to guess password [Spafford, 1992] is used to authenticate a user, after which all access of this user to objects such as files and applications is controlled with an access control list for that object. The standard access control service used by NT and Unix is called discretionary access control, as the owner of each object is responsible for assigning its access permissions. Mandatory access control, as described in the Trusted Computer Systems Evaluation Criteria (Department of Defense, 1985), is used in situations where security is more critical and assigns all permissions based on a security policy defined by the organization responsible for the system.

Role based Access Control (RBAC) [Ferraiolo and Kuhn, 1992] is often promoted as the next-generation access control, superceding the inadequate discretionary and mandatory access control [Sandhu et.al., 1994]. The limitations of these traditional access control services have often resulted in the implementation of additional security at the application level and RBAC is expected to consolidate the security needs in a single service again. This would improve the management of security, especially when RBAC is used as a non-discretionary access control.

The three basic elements of RBAC are roles, users, and permissions. In RBAC a user's access rights or permissions are based on the role that the individual user has in the organization (Barkley, Ferraiolo, and Radack, 1995). A role is assigned to a user based on his/hers authorities and competence to achieve tasks. For example, an accountant can issue a check, but only a manager can approve it. Organized in this way, RBAC can ensure that users naturally follow the security policy of an organization.

A role can be considered to be a collection of users with the same permissions. To better manage roles and permissions, [Sandhu et al., 1996] proposed the use of role hierarchies to provide a natural representation of an organization's structure. In figure 3, we give an example borrowed from [Sandhu et al., 1996]. At the lowest level we have the role of a staff member. Lecturer, tutor and lab supervisor are three different roles in a University department, each with different access permissions. All three, however, have also got the right to take up the role of a staff member with its accompanying access permissions. A course coordinator is a more powerful role, and a course coordinator can also get the role of a lecturer, a tutor or a lab supervisor.

In this example, a role relates clearly to a position within an organization and describes authorities and responsibilities to fulfil the functions of such a position. Although a user is usually considered to be a human being, a process such as a mail program, or a peripheral such as a printer, can also be treated as a user.

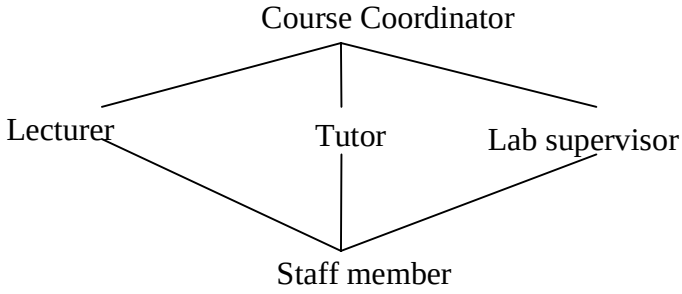

Figure 1: An example of role hierarchies

One benefit of using the role-based access control is that it can assist an employee to fulfil an organization's policy. For example, a tutor can not modify final marks but can agree with the students in a tutorial to move a it to a different time. A lecturer does not need to be able to change the time tabling of the tutorials, which is the responsibility of the course coordinator.

This simple example also shows the main problem of such large granularity RBAC hierarchies. The RBAC needs to have an additional mechanism to ensure that a tutor can only change the time of his/her own tutorials and a lecturer can only change marks in his/her own courses. A further discussion of solutions to this problem, however, is outside the scope of this paper.

Another issue in the design of a RBAC system is who is authorized to assign roles to users and to create new roles in an organization. In most cases, RBAC is an extension of mandatory access control and managed centrally [Ferraiolo, and Kuhn, 1992]. This makes it much easier to encourage users to abide by the security policy, as it is not necessary to make all users understand the policy. In a collaborative environment, however, it is inconvenient to use a centralized role-based access control [Jaeger, and Prakash, 1995] and it is suggested that RBAC in those cases should be able to behave as a discretionary access control. In other words, users should be able to at least grant access rights to their own objects and maybe also be allowed to create roles.

SMALL-GRANULARITY ROLE-BASED ACCESS CONTROL

The best way to prevent information misuse is to restrict access rights to the smallest subset, which is necessary to achieve a user's jobs in an organization [Holbein and Teufel, 1995]. Small-granularity RBAC applies this principle through a further decomposition of a user's role. Instead of assigning all the privileges of a lecturer to the lecturer role, we now establish the role of marker, the role of lecture-note author, and as many other roles as

we can distinguish. Using the same hierarchical role structure as before, the actual permissions of the general lecturer role are now limited, but a lecturer is allowed to take on the role of marker, etc, *when needed* (figure 2).

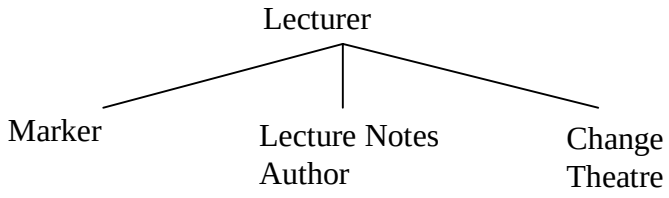

Figure 2: Sub roles of the lecturer role in figure 1.

This low level of granularity provides several important protections [National Computer Security Center, 1989]:

1 limits the effects of errors on the part of the user
2 limits the effects of incorrect code which implements the users functions
3 provides some protection against malicious users, in that damage that can be done is strictly contained to the privileges defined for the current role. This argument can be extended to malicious code, which is inserted into the user's functions.

This last point especially, depends on the ease with which a user can take on additional roles. In figure 3 we show how a RBAC system can allow the user to open multiple sessions, each with a different role. However, if a user keeps activating more roles, this user will end up activating all of the roles available to him. Hence, such a user will expose the system to unacceptably high risks.

The most secure solution would seem to be to restrict a user within a single role at any time. A user is not allowed to directly activate or deactivate roles, but only switch from one role to another one. To make this system workable, however, one would need to define the roles in such a way that there is no need to switch forward and backward between two roles too often. Hence, the result would be that the granularity of roles would increase again.

We propose, therefore, to allow the activation of multiple roles, but with a time limit on each role. Like in a password protected screen saver, non-activity in a specific role will result in an automatic de-activation of this role. Additional limits on the maximum time a role can be active, and constraints, on which roles can be active at the same time, are further mechanism that can be used to limit the damage an intrusion can produce.

To really, significantly, improve the protection that RBAC can offer, we need an additional mechanism to authenticate the necessity of the activation of a role [Holbein and Teufel, 1995]. However, preventing a user from accessing a role can be really annoying for that user and may lead to a significant increase in overhead for IT user support functions. Anecdotal evidence suggests that a similar increase has already occurred in organizations where the powerful but complex Windows NT access control lists are not carefully managed by a central IT department.

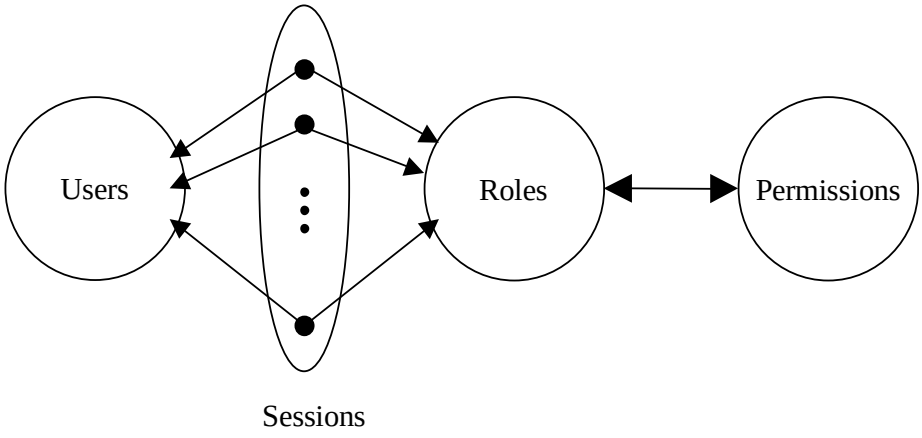

Figure 3: RBAC with multiple role assignments.

Even when an RBAC would temporarily prevent a user from accessing roles he does not need, this would only limit the initial damage an intruder can cause. A patient intruder (or Trojan Horse) can just wait until the compromised user account does get permission to access more powerful roles. Hence, we suggest that on its own

small granularity RBAC will offer only limited security and we propose that its applicability is improved when combined with a suitable Intrusion Detection System (IDS).

While just implementing small granularity RBAC will normally not prevent major damage, a proper modelling of roles in an organization can be used to slow down an attack and give an Intrusion Detection System more time to detect and react. To prevent a malicious user or program from doing a lot of damage in a short time, a small granularity RBAC should put a limit on the number of roles that any user is allowed to activate in a particular time frame. We believe that limiting the number of simultaneous role activations can often be more important than preventing access to particular roles. In many organizations there may be no real need to prevent a user from accessing any of the general-user roles; just let the user decide which roles he/she needs to access, but make sure you monitor role activation to detect abnormal behaviour.

Anomaly-based Intrusion Detection

An anomaly-based Intrusion Detection System (IDS) will, as the name implies, attempt to detect anomalies in data collected from one or more computers (or networks) on the assumption that those anomalies will represent an actual intrusion [Ruighaver, 1998]. An intrusion is generally defined as "the unauthorized use, misuse or abuse of a computer system, a definition that is too vague to be very useful when you need an objective measure of what an IDS is really capable of detecting.

For this paper, we will only consider such intrusions where an unauthorized user is masquerading as an authorized user. This can occur when you leave your computer without logging out, or when someone has watched you type in your password. Another common attack, which allows access by an unauthorized user, occurs when the program you run turns out to be a Trojan horse and executes a series of commands without you being aware.

In all these cases, the anomaly based IDS needs to be able to finger print the behavior of an authorized user, in such a way that the behavior of the unauthorized user is immediately recognized as anomalous. Our current IDS prototype uses a neural network based anomaly detection engine [Tan, 1995] and has been designed to achieve detection of a masquerading intruders as well as intrusive behavior of the user itself in a traditional Operating System environment (Unix and NT). As such, it uses the standard system logs to build up a profile of each user's behavior, and compares the current behavior with the user's profile of past behavior.

Even though the real-time aspect of the IDS has imposed some limitations, our prototype has shown that the resulting profile can provide a reasonable fingerprint of a user's behavior. However, depending on the application area of the IDS and the available audit data, the fingerprint may not necessarily be directly related to the intrusive behavior we would like to detect. Hence, a deviation of the behavior from this fingerprint does not automatically mean that there is an intrusion.

Although this IDS has proved to be useful when used by the system administrator as a behavioral monitor, the many "false" alarms make an automatic response infeasible. Ideally, we could cope with this limitation if it is possible to immediately re-authenticate the user, but that is normally not possible in a password based access control environment. Asking the user to re-enter his password would open the way for malicious programs aimed at password capture. So instead, the only acceptable first response to an alarm is to turn on full auditing for this user.

Another major problem occurs when we try to implement anomaly-based Intrusion detection in a WWW based environment where the point-and-click interface is prevalent. The user behavior in such an environment is just not complex enough to provide the IDS with a usable profile for each user. For anomaly based intrusion detection to be able to work we have to make the environment artificially more complex.

This is where small granularity RBAC can play a major role. One possible source of user behavior can be build into the role switching mechanism. When a user has multiple roles, it becomes difficult for a user to decide beforehand on the role he needs. We therefore decided on a role switching mechanism that only activates a role after the system detects a need for a new role. As there will in general be more than one role that can provide the necessary access privileges for the user, the system then provides the user with a list of available roles. The user has to select a role and, by providing this choice we have created an additional behavior that can be used by an anomaly based intrusion detection system.

A SIMPLE IMPLEMENTATION FOR INTRANETS

There are already several proposed implementations of the role-based access control for Intranets [Tari and Chan, 1997 and Barkley, 1995], but they do not have dynamic activation and automatic deactivation of roles. These RBAC systems are also complex designs aimed at distributed applications, and not suited for simple

Intranets, which often just consist of a few HTTP servers. Another well-known implementation [Kaijser, 1994] uses a proxy server between the browser and the WWW servers, a solution that in our view generates several new security risks. We therefore decided to implement our own simple prototype to demonstrate that the implementation of small granularity RBAC can be achieved on existing WWW servers with minimal effort.

In traditional operating systems, access rights are classified into read, write, or execute. Role-based access control is not limited to these traditional rights and can consider access rights in a different way, i.e a debit from an account can be an access right. On an Intranet the most elementary action is accessing a web page. Each page can contain different information or achieve special actions through forms or Java applets. As each action is started by requesting a web page, we can simply embed the relationship between roles and permissions into the web page by specifying which roles are allowed to access that page.

In our RBAC system, access permissions do not have to be kept secret, which allows us to use a simple implementation. According to the specification of Hypertext Markup Language (Raggett, Hors, and Jacobs, 1997), the purpose of the Meta tag is to provide information about the document rather than the document content. So, we added a Meta tag to specify the authorized roles. The WWW server now needs to check this Meta tag with the available roles for a user, before allowing the requested page to be returned to the user's browser.

After considering several alternatives for the management of the user's roles, we decided to start with a simple solution that stores the current roles, and their expiry times, in a cookie on the user's browser. Every time a user requests a page, the cookie is retrieved and the roles are checked. When the access is granted the expiry time is updated after which the cookie can be returned to the browser. It should be evident that both cookies and data transfers between the different components are protected by encryption.

To authenticate the user and to enable the switching of roles, we have used a Java applet. Before a user can start to access the Intranet, he/she needs to connect to a WWW page on the RBAC server. From there, the Java applet is downloaded to the user's browser and a permanent connection is established between the applet and the RBAC server. When a WWW server gets a request for a page and finds that no role is available that matches the authorized roles for that page, the WWW server contacts the RBAC server to report the need for a role switch. The RBAC server then sends the names of all authorized roles from this page to the Java applet, and allows the user to specify the new role. When the authentication of the role switch succeeds, the RBAC server updates the cookie in the user's browser, and tells the WWW server to try again.

A further description of the technical details of our implementation falls outside the scope of this paper. It should be realized, however, that any security service should be protected against direct attacks on the service. Correct implementation of a security service is just as important as the conceptual design of the security mechanisms used in the service.

CONCLUSION

Small-granularity role based access control offers an effective solution to reduce the damage an intrusion can cause to your organization. We described a new dynamic activation of roles, with automatic de-activation if the role is no longer used. This new mechanism allows us to further decrease the granularity of roles, which leads to an improved protection of the sensitive information on your systems.

Anomaly based Intrusion detection is a rapidly growing area in computer security, but its application relies on users showing behavioral patterns that can be used to provide a fingerprint. Unfortunately, the "point and click" user interface used in current applications is not a good provider of behavioral patterns. Hence, designing and building in patterns of behavior in an information system, without getting the user annoyed, is essential for a good anomaly based Intrusion Detection System.

We discussed a role switch mechanism for RBAC, which can be used to provide additional input for such an intrusion detection system. By providing additional choices for roles, which are all valid, we encourage the user to create a more complex behavior, making it more difficult for an unauthorized user to emulate the behavior of an authorized user.

RBAC and anomaly based IDS together form a powerful combination to protect your Intranet from intruders. To show how easy it can be to add these components to an existing platform, we described a simple implementation for a WWW based Intranet. This description was not meant to provide you with the full details of our RBAC-IDS architecture, as we believe the principles are more important than our choice of implementation.

REFERENCES

Barkley, J., Ferraiolo, D., and Radack, S. (1995) An Introduction to Role-based Access Control, Information Technology Laboratory Bulletins, National Institute of Standards and Technology, December 1995.

Barkley, J.F. (1995) Implementing Role Based Access Control using Object Technology, Proceedings of the first ACM Workshop on Role-based Access Control.

Champine, G.A., Geer, D.E., and Ruh, W.N. (1990) Project Athena as a Distributed Computer system, *IEEE Computer*, September, pp40-51.

Department of Defense (1985) *Trusted Computer Systems Evaluation Criteria*, DOD 5200.28-STD.

Dierks, T., and Allen, C. (1997) *The TLS Protocol Version 1.0*, the Internet Engineering Task Force (IETF) draft, November 12.

Ferraiolo, D., and Kuhn, R. (1992) Role-based Access Controls, *Proceedings of NIST-NCSC.*

Ferraiolo, D.F., Cugini, J.A., Kuhn, D.R. (1995) Role Based Access Control: Features and Motivations, *Proceedings of Annual Computer Security Applications Conference*, IEEE Computer Society Press.

Holbein, R., and Teufel, S. (1995) A Context Authentication Service for Role Based Access Control in Distributed Systems, *Proceedings of International*

Jaeger, T., and Prakash, A. (1995) Requirements of Role-based Access Control for Collaborative Systems, *Proceedings of the first ACM Workshop on Role-based Access Control*.

Kaijser, P., Parker, T., and Pinkas,D. (1994) SESAME: The solution to Security for Open Distributed Systems, Computer Communications 17 (7), pp501-518.

National Computer Security Center (1989) Guide to Understanding Trusted Facility Management, NCSC-TG-015, Library No. S-231, 429.

Raggett, D., Hors, A.L., and Jacobs, I. (1997) *HTML 4.0 Specification*, W3C Recommendation, December 18.

Ruighaver A.B. and Ahmad A. (1996) How Safe is Your Firewall: On the Security of Intranets, Proceedings of the Australasian Conference on Information Systems, Hobart, Australia, December 1996, pp. 601-608

Ruighaver, A.B., Thorne P.G. and Tan K. (1998) Evaluating a Real-time Anomaly-based Intrusion Detection System. Proceedings of the First Workshop on Recent advances in Intrusion Detection RAID98, Belgium, 1998.

Sandhu, R. (1996) Access Control: The Neglected Frontier, *Proceedings of the First Australian Conference on Information Security and Privacy*, June 24-26, pp219-227.

Spafford, E.H. (1992) OPUS: Preventing Weak Password Choices, Computers and security, 11, pp. 273-278.

Tan, K (1995) An application of Neural Networks To Unix System Security, *In Proceedings of the IEEE International Conference On Neural Networks*, November 1995.

ACKNOWLEDGEMENTS

I would like to thank Richard Wei for his contributions in implementing the prototype described in this paper.

COPYRIGHT