Health Information ARCHITECTURE

Role-Based Access Control (RBAC) Newsletter

Volume 32 – October 2006

Inside this issue:

VHA/IHS RBAC TF Chair
Robert O'Hara, MD
Robert.Ohara@med.va.gov

VHA Deputy Chief Architect
RBAC Project Manager
Steve Wagner
Steve.Wagner@med.va.gov

VHA Security Architect
RBAC Architect
Mike Davis, CISSP
Mike.Davis@med.va.gov

VHA Security Architect
RBAC Architect
Ed Coyne, PhD
Ed.Coyne@med.va.gov

VHA Security Analyst
Sepideh Khosravifar
Sepideh.Khosravifar@va.gov

RBAC Project Lead
Suzanne Webb
Suzanne.Gonzales-Webb@va.gov

Abstract Review, Excerpts - A Reference Monitor for Workflow Systems with Constrained Task Execution; Specifying and Enforcing Constraints in Role Based Access Control

Jason Crampton, Information Security Group
Royal Holloway, University of London

[Continued from previous RBAC Newsletter issue]

ENFORCING CONSTRAINTS IN A ROLE-BASED SYSTEM

A request is an interaction by a user with a system that potentially results in a change to the configuration or state of the system. In a role-based context, such requests include: a request to invoke permission; a request to activate a role or to establish a session; attempts to assign a user or permission to a role; and changes to the role hierarchy.

The Enforcement Context of a Constraint

Constraints can be enforced by either the configuration or the state, or the state history of a system. These are usually referred to as static, dynamic and historical constraints, respectively.

A brief description of a *typical* role constraint requires that no user be assigned to both r1 and r2 (Where r1 and r2 are roles). When this is a static constraint, then it is satisfied if for all users. When describing a dynamic role constraint however, the constraint is satisfied if (for all users) when no user has activated both r1 and r2, such as in the previous example of the ward clerk and the charge nurse (Note that it is possible in this case for a user to be assigned to both roles, but the user will not be permitted to activate both in the same session.). It should be noted that it is difficult to interpret <the dynamic role> as a historical constraint. Should it mean that once the user has been assigned to r1 (say), then the user can never be assigned to r2. Alternatively, should it mean that once the user has activated r1, then u can never activate r2? These interpretations of enforcement context could be classified as static historical and dynamic historical, respectively. To the author's knowledge, no such distinction exists in the literature. A detailed discussion of these constraints and whether we need additional historical interpretations of them is beyond the scope of this paper. Dr. Crampton believes that the dynamic historical interpretation is likely to be more meaningful in practice, and is the interpretation he adopts in this paper. It is easily seen that the enforcement of a static authorization constraint implies that the corresponding dynamic constraint is enforced.

Observations

Separation of duty is not a complicated concept!

Research papers on separation of duty in computer systems regularly describe constraints that are defined in terms of users and in terms of roles. It is not immediately obvious when these constraints are to be specified.

Upcoming Meetings

- **November HL7 Educational Summit**
  November 7-9, 2006
  Lynnwood, WA
- **INCITS Workshop on Cyber Security SC 27**
  Oct 31 - Nov 1, 2006
  Singapore
- **ONC (ONCHIT)**
  American Health Information Community Meeting
  October 7, 2006
  Washington, DC
- **OASIS Adoption Forum**
  October 28-29, 2006
  London, United Kingdom
- **World Innovation & Technology Conference**
  November 1-3, 2006
  Washington, DC
- **HL7 Educational Summit**
  November 7-9, 2006
  Lynnwood, WA
- **ASTM Committees E31**
  November 12-14, 2006
  Atlanta, GA

~

Separation of duty requirements articulate circumstances that would lead to the violation of business rules. A violation could be regarded as the occurrence of a set of events that contravene the business rules. In other words, to specify a separation of duty constraint we merely need to specify one-by-one the "undesirable" set or sets of events (per permission or role as the case may be), and to enforce a constraint we must ensure that all the events specified in that set cannot occur.

There are certain separation of duty constraints that could be hard-wired into a healthcare application – such as no user can be assigned as both a ward clerk and a charge nurse. It is the users of an application who will be required to specify separation of duty constraints.

User Permission Constraint

The work of Sandhu on transaction control expressions [1] assumed it was possible to define a life cycle for transient objects. For example: In the case of an RN receiving narcotics from the inpatient pharmacy, the life cycle (for the narcotic transfer) begins when the RN receives the narcotics and ends when it is logged into the floor. That is, certain permissions are necessarily only invoked a certain number of times. Hence, once permission ceases to become meaningful, it can be deleted from any blacklist it belongs to.

The Constraint Monitor

Dr. Crampton assumes the existence of a constraint monitor that is responsible for enforcing authorization constraints. Each constrained request could potentially cause a violation of an authorization constraint. Hence each instance of a constrained request is passed to the constraint monitor. The constraint monitor checks whether granting the request would violate an authorization constraint and takes appropriate action. Note that it is possible to enforce a static constraint by considering the configuration of the system. In other words, we only require blacklists for historical authorization constraints. Similarly, it is possible to enforce a dynamic constraint by considering the state of the system. For example, suppose a user attempts to start a session by activating two roles r1 and r2, which form a dynamic separation of duty constraint. Then the security monitor would prevent the session from starting.

Hence we envisage that there could be several different classes of constraint monitors derived from some abstract constraint monitor class. These could include a role hierarchy monitor class, a user-role assignment monitor class, a permission-role assignment monitor class, a session monitor class and monitor classes for specific types. Each monitor will maintain a list of authorization constraints that are relevant to that monitor.

Blacklist: Enforcing historical constraints

In order to enforce a historical authorization constraint it may be necessary to deny certain requests.

One possible approach to enforcement is to employ a historical record [1, 2, 3] of all previous invocations of permissions and to consider this record whenever a request to invoke a permission is made. The Chinese Wall model[1] defines the history matrix for precisely this purpose. It is our belief that approaches of this nature will not scale well to large-scale applications. We employ a different approach to the enforcement of historical constraints by dynamically changing the requests that can succeed. Hence, in order to enforce a constraint, we create a blacklist – a dynamic access control structure that contains constrained requests. When a constrained request occurs the relevant blacklist is consulted. If the request belongs to the blacklist, the request is denied; otherwise, the request is referred to the role-based reference monitor.

EVALUATION AND RELATED WORK

Specification. The specification scheme outlined in Dr. Crampton's paper has its basis in our set-based approach to conflict of interest policies [4]. It is similar to the model developed by Jaeger and Tidswell [5] which uses set predicates to define separation of duty constraints. The scheme Dr. Crampton proposes is considerably simpler syntactically than their scheme because he makes no attempt to define the conditions that must be met for the constraint to be satisfied. Indeed, with the exception of the work of Simon and Zurko [3], most previous specification schemes for separation of duty have followed this approach and used some fragment of first-order logic as the specification language. Crampton believes that it is well understood when a separation of duty constraint is violated and that including the conditions that would cause a violation simply increases the number of predicates and functions required to specify the constraints.

There are certain separation of duty constraints that can be specified in RCL 2000[2] [2] and in Jaeger and Tidswell's model that we cannot specify using Crampton's approach. Broadly speaking these are constraints where it is not sufficient to iterate over the constraint set for each element of the scope. It is not possible to express the following separation of duty requirement: given two users u and v and two roles r and s, we do not want either u or v to be assigned to both roles or u to be assigned one role and v assigned to the other. Such a constraint is potentially useful in preventing collusion between groups of users. A little thought shows that such a constraint defines the following constrained sets:

{(u, r), (u, s)}, {(v, r), (v, s)}, {(u, r), (v, s)} and {(u, s), (v, r)}.

In other words, we could introduce a fourth parameter in our definition of constraint which determines the "direct product" semantics of the constraint: that is, whether we should iterate over both the scope and the constraint set or simply over the constraint set (as we currently do). (The Jaeger and Tidswell scheme includes eight different ways of iterating through the various components in a constraint.)

[1] The Chinese Wall model is a combination of free choice and mandatory control. It can be combined with DAC policies.

[2] RCL 2000 (For Role-Based Constrains Language 2000, pronounced *Rickle* 2000) is the specification language for role-based authorization constraints [2].

~

RBAC Newsletter

ATTN: Suzanne Webb
RBAC Project Lead
10260 Campus Point MS-B1E
La Jolla, CA 92121

Or e-mail:

Suzanne.Gonzales-Webb@va.gov

Dr. Crampton has also made an assumption that he did not need to consider order-dependent historical constraints. However, if it proved that such an assumption were not valid, it should be possible to interpret the constraint set as a constraint list. When an object is instantiated for which such a constraint is defined, we enter all elements of the list except the head of the list (the next event that should be executed) into the appropriate blacklist(s). Once that event occurs, the next element in the list is removed from the blacklist(s). Of course, this scheme is only applicable to linear workflows; if the workflow branches, we need to model a partial order. It is not immediately obvious how such workflow constraints could be specified within his proposed scheme.

Enforcement. Dynamically modifying access control structures in order to reflect previous access requests or execution paths has received attention in several recent research papers. Edjlali et al. proposed a dynamic approach to controlling the access rights of mobile code in order to enforce requirements of the following form: if an application has accessed a file on the local host system, then the application cannot open a socket [6]. More recently Abadi and Fournet have proposed an alternative to the "stack walk" semantics for virtual machines using the intersection of access rights that have been available to each process [7]. Crampton has used these ideas as a starting point to develop the idea of a blacklist, which dynamically limits the permissions available to users (and possibly roles). The concept of a blacklist also employs the concept of negative permissions, which have received little attention since their introduction to the role-based access control literature [8].

The enforcement model Dr. Crampton defines in this paper can only enforce constraints in which the constraint set has no more than two elements. We do this because it is not possible to implement blacklists with a simple semantics otherwise. To see this, consider the constraint (U, {p1, p2, p3}, h) and assume that none of the three permissions have been invoked by user u. Once u invokes the permission p1, say, which of the remaining two permissions should be entered in the blacklist? Clearly, we could enter both p2 and p3 into the blacklist, but this would be too restrictive.

The alternative is to keep a historical record of all access requests and to enter either p2 or p3 at some point in the future. Crampton prefers not to keep a historical record because he believes that such an approach will have unacceptable performance overheads. Therefore, he chooses to impose this upper bound on the cardinality of the constraint set in historical authorization constraints. It should be noted that most existing approaches to separation of duty only consider constraint sets with precisely two elements, the exceptions being the RCL 2000 specification language [9] and the work of Simon and Zurko [3].

SUMMARY AND FUTURE WORK

Dr. Crampton has developed a simple set-based specification scheme for authorization constraints in role-based access control systems. He has also suggested an enforcement model for a restricted subset of this scheme. To the author's knowledge, this is the first attempt at defining a specification and

~

RBAC Newsletter

ATTN: Suzanne Webb
RBAC Project Lead
10260 Campus Point MS-B1E
La Jolla, CA 92121

Or e-mail:

Suzanne.Gonzales-Webb@va.gov

enforcement model for authorization constraints since the work by Simon and Zurko [3]. Crampton believes that his specification is easier to understand than their scheme and that this enforcement model, which does not rely on maintaining a historical record of all previous system activity, is likely to have lower performance overheads, particularly for large-scale applications. There are several interesting directions for future work, some of which have been alluded to in the body of this paper. Crampton would like to investigate whether constraint sets with an arbitrary number of elements can be enforced in a simple way. Dr. Crampton would also like to find an intuitive scheme for defining different combinations of elements in the constraint scope and the constraint set in order to increase the range of constraints that his scheme can support. The most ambitious goal is to develop abstract Java classes, constraint and monitor, that implement our constraint specification and enforcement schemes. The ultimate objective being, to develop a generic middleware authorization constraint engine that can be instantiated by application developers and systems administrators to support enterprise-wide heterogeneous constraint authorization policies.

RBAC96

Dr. Crampton developed the material in this paper in the context of RBAC96, the most widely known role-based access control model [6].

Use of this article was by permission of the author, Jason Crampton.

REFERENCES

[1] Sandhu, R. Transaction control expressions for separation of duties. In Proceedings of 4th Aerospace Computer Security Conference (Orlando, Florida, 1988), pp. 282–286.

[2] Brewer, D., and Nash, M. The Chinese Wall security policy. In Proceedings of 1989 IEEE Symposium on Security and Privacy (Oakland, California, 1989), IEEE Computer Society Press, pp. 206–214.

[3] Simon, R., and Zurko, M. Separation of duty in role-based environments. In Proceedings of 10th IEEE Computer Security Foundations Workshop (Rockport, Massachusetts, 1997), pp. 183–194.

[4] Crampton, J., and Loizou, G. Structural complexity of conflict of interest policies. Tech. Rep. BBKCS-00-07, Birkbeck College, University of London, 2000.

[5] Jaeger, T., and Tidswell, J. Practical safety in flexible access control models. ACM transactions on Information and System Security 4, 2 (2001), 158–190.

[6] Edjlali, G., Acharya, A., and Chaudhary, V. History-based access control for mobile code. In Proceedings of Fifth ACM Conference on Computer and Communications Security (1998), pp. 38–48.

[7] Abadi, M., and Fournet, C. Access control based on execution history. In Proceedings of 10th Annual Network and Distributed System Security Symposium (2003). To appear.

[8] Sandhu, R., Coyne, E., Feinstein, H., and Youman, C. Role-based access control models. IEEE Computer 29, 2 (1996), 38–47.

[9] Ahn, G.-J., and Sandhu, R. Role-based authorization constraints specification. ACM Transactions on Information and System Security 3, 4 (2000), 207–226.

~

RBAC Newsletter

ATTN: Suzanne Webb
RBAC Project Lead
10260 Campus Point MS-B1E
La Jolla, CA 92121

Or e-mail:

Suzanne.Gonzales-Webb@va.gov

Health Information ARCHITECTURE

Role-Based Access Control (RBAC) Newsletter

RBAC Taskforce – Update

The next RBAC Taskforce meeting call will be held October 18th, 2006 - Wednesday at 1300CT / 1100PST / 1200MT / 1400EST; a meeting update has been sent out to the current participants. If you would like to be a part of the Task Force please contact Suzanne Gonzales-Webb for more information, thank you.

The RBAC Taskforce will continue discussions surrounding the definition of constraints on current Permission Catalog and Roles, as well as an update on the initiation of RBAC incorporation into the VA re-engineering projects. Current Task Force Members will be contacted with additional materials in preparation for the meeting.

~

Role-Based Access Control is critically important to the security aspects of the VA and other healthcare organizations. There is a growing management and security demand for RBAC to be implemented in healthcare systems.

RBAC grants rights and permissions to roles rather than individual users. Users then acquire the rights and permissions by being assigned to appropriate roles. By grouping individuals with other individuals who have similar access rights, RBAC can provide significant security management efficiencies.

The latest RBAC Documentation additions and prior RBAC Newsletters can be found on the RBAC Website.

~

The RBAC Newsletter will now be published quarterly instead of monthly. Please be on the lookout for the next issue due January 2007!

~

RBAC Newsletter

ATTN: Suzanne Webb
RBAC Project Lead
10260 Campus Point MS-B1E
La Jolla, CA 92121

Or e-mail:

Suzanne.Gonzales-Webb@va.gov