Role-Based Access Control: The NIST Solution

Name: Hazen A. Weber
Date: October 8, 2003
Certification: GSEC
Version: 1.4b
Option: 1

Abstract

Today's competitive environment often times requires that data be secured and access to that data be limited to the "minimum necessary". Security models such as Mandatory Access Control and Discretionary Access Control have been the means by which to secure information and regulate access. But due to the inflexibility of these models, the rather new security concept of Role-Based Access Control (RBAC) as proposed by the National Institute of Standards and Technology (NIST) promises to become a more prominent security model in the near future. By decreasing rights administration efforts to role development and assignment, security and productivity can be both increased while greatly downplaying the "balance" effect of sacrificing one for the other.

Introduction: The Need

Wide spread use of the internet, lower technology costs, and a great need for data access and sharing in a competitive market has driven the development of new technologies and standards. Looking for a competitive edge, increased productivity, and security, both system vendors and implementers have been looking for the means to properly administer these rapidly expanding and costly infrastructures. More so now than ever, the downtime of users and the delay in account creation can mean losses in the thousands. With this growth, holes in security have generated media frenzy and have forced accrediting and governmental agencies to act by imposing requirements regarding security and privacy of information. From credit card transactions to patient health information, privacy is quickly becoming the centerpiece of a new wave in technological advance, not just in hardware but also in conceptual approaches to security. Concepts like "defense in depth" are becoming staples of the industry. Books on security are flying off of the shelf, system patches are released nearly each day, and companies like Symantec and McAfee are being constantly challenged by new and advanced virus threats. But it is the security concern within that often times goes unacknowledged. Account security in many organizations is loose at best, perpetuated by a high volume of requests and a security administration model grossly overpowered by the infrastructure it is forced to support. Role Based Access Control (RBAC) will allow for easier administration of today's large and complex corporate environments without sacrificing the need for securing data and access to it.

However in development of RBAC various vendors have developed their own model of RBAC. In order to bridge the gaps of difference the National Institute of Standards and Technology has worked towards developing and proposing their model of RBAC. This model by NIST is impressive in its offerings and is perhaps the mostly complete and well-documented version.

Security Models

In order to better understand the benefits of a Role-Based Access Control (RBAC) security administrative model, we must understand some of the current concepts being utilized. While there are many variations and ideas behind security administration, we are going to focus on three basic concepts: Mandatory Access Control, Discretionary Access Control and of course Role-Based Access Control. Please note, that while this paper explains many of the benefits of RBAC, a security administrator, analyst, or architect, must always take into consideration the needs and capabilities of their environment before ruling out any security model.

Mandatory Access Control (MAC):

Mandatory Access Control or MAC utilizes security provisions that are typically hard coded into an application or operating system. These provisions or rules apply to all objects, applications and various resources including the end-user that tries to access the data it is designed to protect. More so than operating systems, applications, especially military, governmental or occasionally specialized in-house developed applications, incorporate the MAC concept. This typically begins by classification of data, for example sensitive, secret and confidential, and next the classification of resources that will be making requests for data.

Let's view this in a simple manner as it might pertain to a family physicians office. Sensitive data may consist of patient names and appointment times so a secretary for a doctor could view this information and modify it as appointments were made, cancelled or changed. A nurse however, would not only be able to view the patient's appointment times, but would have the ability to view and modify "secret" information such as blood pressure or weight in the patient's record. This is because the nurse, as a requester of resources or data, is classified as both "sensitive" and "secret". The doctor, however, would have the ability to update and modify the "confidential" information within the record such as lab or test results because his classification is "confidential". It would also make sense that the doctor has "secret" level access so that pertinent information such as blood pressure could be viewed. However, since the doctor may not care to know which patient is next, it may not be necessary for him to be granted "sensitive" access to view patient appointment times. So MAC is very capable of enforcing "separation of duties" as may be needed.

This illustration explains the concept used by MAC in that access is granted based on the classification of the data and the permissions granted by the application to the classified resources trying to access the data. This type of access control is very secure in that it can be granular in design. Some implementations of MAC include a hierarchal structure, meaning that a user assigned "secret" has access to "secret" and "sensitive" data. A user assigned "confidential" could also have "secret" and "sensitive" data access in a hierarchal implementation. Another security benefit of this model is that the security rules are hard coded into the software so the chance for administrative error or social engineering is greatly reduced. Where MAC can fall short is in the development or modification of the rules within the application. Because it is hard coded programmers will need to review the coding of the application and make changes. This could be especially frustrating if the application is a turnkey solution from a vendor requiring vendor assistance for modification. MAC is best suited for specialty applications for a group of users with rather similar needs. As a rule it typically does not function well as a corporate wide authentication and accessibility security model.

As you can see, the inability for MAC to change in the age of consolidation, constant corporate mergers and co-op relations, makes it an administrative nightmare for general account administration in a dynamic and evolving environment.

Discretionary Access Control (DAC):

Discretionary Access Control (DAC) works both as a centralized security model and a distributed model. A centralized security model is when an administrator or team of administrators distributes access to data, applications and network devices. All requests for access changes need to be completed by this single department. In a large organization this can be very time consuming, especially if the administrators are off site or outsourced. A distributed model allows responsible and knowledgeable personnel to distribute access to data and applications. In large companies this may be a manager, supervisor, or team lead. In small organizations it may simply be the most computer savvy team member. The benefit of a distributed model is that delays can be avoided since the administration of accounts is dispersed.

For instance a manager of home loans for a bank may decide that records should be distributed between four individuals based on location within the country. Since DAC is the security model and it implementation is distributed the manager has the ability to assign access to his employees to data which he controls. With the DAC model the manager can grant one underling access to the "west coast records", a second to "east coast records" and so on and so forth. Each employee would be able to view "their" records granted to them by the manager, but would not be able to view the records of their peers. This inability to view data assigned to peers is not due to data classification as in the

MAC model, but simply because access was not granted to the employee by the manager.

Because DAC can be implemented in a distributed security model, it can greatly reduce account access change turnaround times by removing the “middle man”. Some network operating systems take into consideration this distributed DAC implementation and have created roles, such as Novell NetWare’s “Workgroup Manager” that can be granted the ability to modify access for accounts or even create accounts. Windows NT/2000 will also allow users to be associated with an “Administrators” group that has the ability to create users and grant access either on the workstation or server system. This concept of access control, even if used in a centralized security model does have the potential benefit of human reasoning. This allows the administrator to take into consideration circumstances and variables an application using the MAC model could never consider. This can be both positive and a potential threat.

While DAC would appear a reasonable solution for both large and small network environments, there are also some sizeable negatives to consider. Since access is distributed at the discretion of the data owner, there is the potential that uniformity of access for end-users with like job functions could be diminished. Consider that several individuals may be owners of the same data; would one know what access the other has granted to their resources? Now you see how the lack of understanding by the data owner could allow access greater than the *minimum necessary*. If explicit rights to data is not known by the data owners or administrators; then who can be sure that the access is not carried with the user as they move from job function to job function within the company. These issues could open the doors to costly and embarrassing repercussions.

What is Role-Based Access Control (RBAC):

In basic review of two standard security models discussed above we can see both benefits and weaknesses for each, dependent on the environment in which they are implemented. The military, which needs to maintain the confidentiality of certain information and greatly limit access to that information, depends more often than not on the hard coded security of the MAC model. Corporations however, may be more concerned about productivity, inter-organizational data sharing, and information workflow between different departments. Especially in smaller companies the DAC model is more suitable than MAC.

However, in recent years as information technology has become a main function in daily operations and as competition between industry leaders grows more intense, the risk of losing information to the competitor has become a growing concern. Should MAC then be implemented as it is for the military or can DAC be tightened down enough to offer the security needed? In the balance of security and productivity neither MAC nor DAC offer a solid solution. MAC can hinder productivity because of the rigid security that is not easily modified. DAC can be made to be very granular, however tracking of access and the micromanagement of access distribution will increase turnaround times, thus again hindering productivity.

Recognizing a need for a better security administration model the National Institute of Standards and Technology (NIST) began a project simply titled the "RBAC Project". While the use of roles have been in existence for over twenty years, primarily used in mainframe and UNIX environments, there lacked a standard model because each system used its own proprietary elements. The scope of this project was to design an access control model that would be standardized, scaleable, logical in design, non-system dependent, and would have positive economical ramifications upon implementation. In 1992 a model was introduced by David Ferraiolo and Rick Kuhn that attempted to meet the requirements of the scope and created a full-fledged RBAC solution. In order to understand the elements of the RBAC model it is helpful to understand the evolution of RBAC concepts.

RBAC Model Evolution

There are four main models regarding RBAC. Each has its own strengths with $RBAC_3$ being the most complete implementation, building on the other models capabilities.

$RBAC_0$:

$RBAC_0$ is the most simplistic in this evolutionary process consisting of least privileges and separation of duties. These are performed through permissions, however $RBAC_0$ does not contain a hierarchy, therefore the permissions were assigned directly to the user within a "role" or job function.

$RBAC_1$:

Based on $RBAC_0$, $RBAC_1$ introduces the use of hierarchies. This was developed to follow the natural distribution of responsibilities within an organization. Levels of responsibilities and corresponding job functions are usually layered as junior and senior roles. It made sense than that RBAC take into consideration these variations and create a layered security distribution method that better supports the needs of a large environment.

$RBAC_2$:

Constraints were introduced with $RBAC_2$ that can serve a number of functions within the RBAC environment. While not having hierarchies, constraints can serve as limiters enforcing the policy that only one individual can be assigned to a specific role. While you wouldn't want this to apply to most roles it could be useful to ensure that only one user has an administrative role granting full administrative rights to a system. Another purpose of constraints is to regulate access by ensuring that certain criteria are met. For instance, in order for a system analyst to gain the permissions given to the Senior Analyst role you may first require they have memberships to the Junior Analyst role. Finally constraints can serve the purpose of ensuring separation of duties. Thus a constraint can be enacted that states if a user is associated with the Accounts Payable role they can not be made a member of the Accounts Billable role.

RBAC$_3$:

RBAC$_3$ consists of both a hierarchal structure and constraints and is the model designed by NIST. Along with having the functionality listed in the above models, constraints can be used as regulatory means on the hierarchal structure. For example, a Junior Analyst role can be limited to only one Senior Analyst role. So while several Senior Analyst roles may exist, a user who is associated with a Junior Analyst role can only be a member of one Senior Analyst role. RBAC$_3$ is the most complex and detailed model of the Role-Based Access Control versions. However, the concepts of RBAC$_2$ and RBAC$_3$ can be integrated into RBAC$_0$ and RBAC$_1$ during the role engineering process, if the operating system or application will support it.

RBAC$_3$ Elements

RBAC$_3$ has five elements: *users*, *roles*, *permissions, operations,* and *objects* that facilitate the administration of access to data resource objects. Since RBAC$_3$ supports a hierarchal structure, each element of the model relates to other elements in order to create levels of permissions and constraints.

User:

User applies to any entity wishing to access a data resource or object. Unlike most discretionary access control models, the user will typically not have access to resources, but rather inherit access to resources through the role(s) they are associated with. Users are both employees and network mechanisms and entities that require access to a specific resource object.

Role:

A role is a package of permissions based on a job function within the organization. Users are assigned roles based on the responsibility of their position and the function in which they serve the organization. A user may have a single role associated with them or may have several depending on the needs of their position. It should be noted however that RBAC is a model that can be implemented at the network operating system level as well as within an application. This said, there could be additional permissions, operations, and objects within an application that are accessed based on roles within the application. These roles can be apart from the network operating system role that allowed execution of the application.

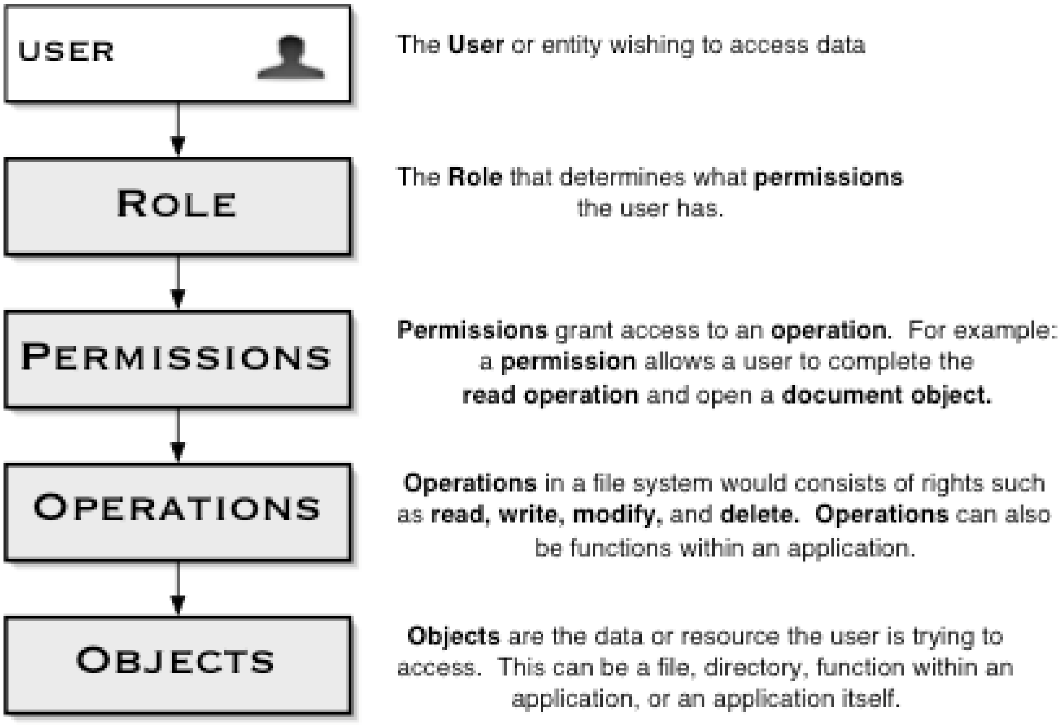
Permissions/Operations:

While Permissions and Operations are separate elements of the RBAC$_3$ model, they are so closely related describing one is nearly impossible without the other.

Permissions are assigned to a role and grant access to Operations. Operations are lower than Permissions in the $RBAC_3$ element hierarchy and usually are very specific functions. These functions can vary based on the system being accessed. In a Novell NetWare file system they would be file rights such as file scan, read, write, erase, create, modify, access control and supervisor. Operations could also be functions within a database such as insert, delete or append. An Operation could also consist of printing to a printer or accessing offline storage such as a tape drive.

Objects:
Objects are accessed through Operations that a user has Permission to access through the Role they are assigned. Objects are anything that contains information that needs to be accessed by a user or network device. In a Novell NetWare file system an Object would be a folder or file located on a NetWare volume. An Object can also be an application that the user wishes to launch. So ultimately an Object is that which the user is after whether it be a spreadsheet, database entry, application or a network device such as a printer.

The graphical representation below shows the hierarchy of the $RBAC_3$ elements and their relationship.


USER
The **User** or entity wishing to access data
ROLE
The **Role** that determines what **permissions** the user has.
PERMISSIONS
Permissions grant access to an **operation**. For example: a **permission** allows a user to complete the **read operation** and open a **document object.**
OPERATIONS
Operations in a file system would consists of rights such as **read, write, modify,** and **delete. Operations** can also be functions within an application.
OBJECTS
Objects are the data or resource the user is trying to access. This can be a file, directory, function within an application, or an application itself.

Graphic 1

RBAC: Basic Implementation Strategies

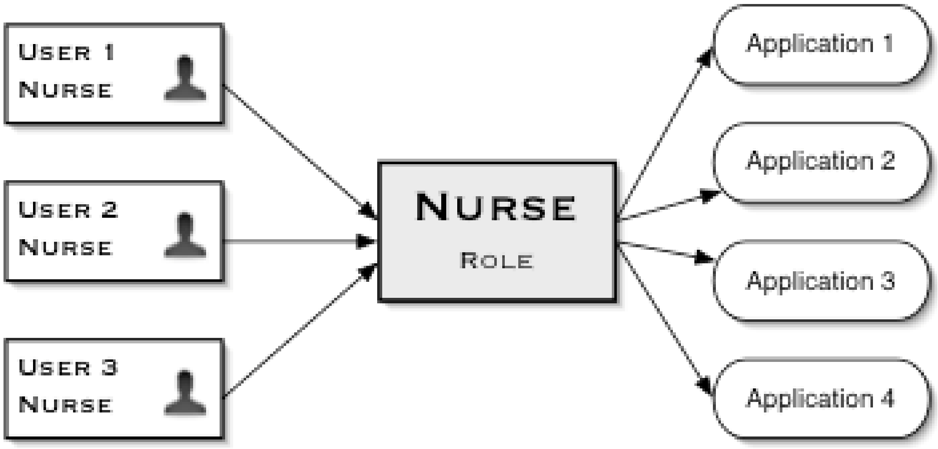
RBAC: Ease of Administration

Through *role engineering*, roles are created to reflect the various positions within the organizational policy, the personnel layout of your corporation. This process does require a great deal of research to ensure that the *minimum necessary* or the concept of *least privileged* is adhered to in the role design. Since one of the benefits of $RBAC_3$ is that discretion regarding rights access distribution is built into the role, much testing must be completed to ensure it is designed and implemented correctly. Once the role is tested and implemented the security administration department can reap the rewards in improved productivity and reduced user downtime when access changes are requested.

The graphic below shows a basic $RBAC_3$ concept in that a role is created with various permissions to objects associated to it, in this case access to applications. Multiple users are associated to the role, each receiving access to the four applications associated with the role. Rather than assigning individual applications to each user as typical in a DAC setting, only one relationship is required in this case, the user to a well-defined role.

Another benefit of $RBAC_3$ is if a user should leave, there is minimal work in ensuring that access to the system is denied. By disassociating the user from the role, they become unable to function even if their account should accidentally remain active. This is not a substitute for a solid termination policy and procedures, but offers a little added security in the result of a failure to properly terminate an account.

Should a user move from a department or job function to another, they are simply removed from their old role and associated with a new role as it corresponds to their new position. This removes the possibility of a user taking unneeded access with them from position to position and supports the concept of *least privilege* or *minimum necessary*.



Graphic 2

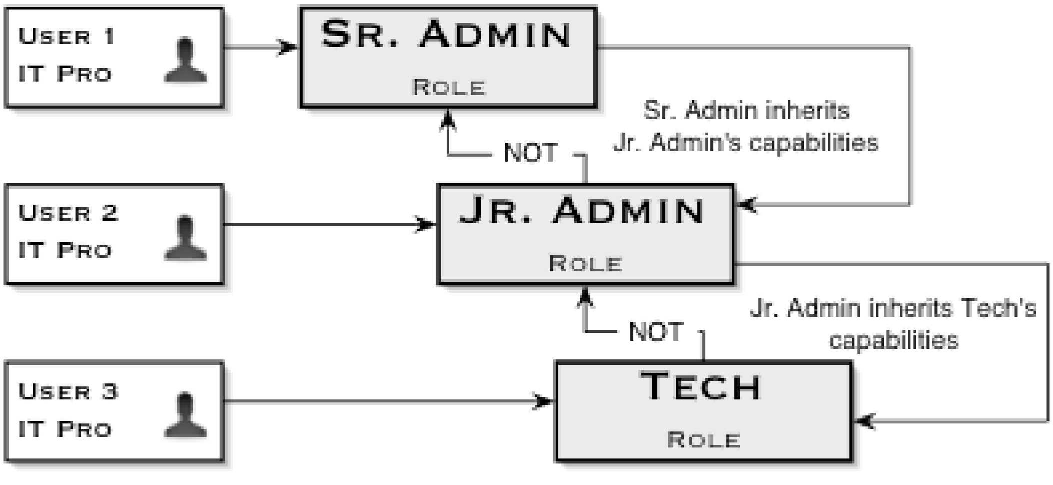
RBAC: Hierarchy and Rights Inheritance

RBAC$_3$ supports a hierarchal design, which can be used to ease administration by allowing rights to flow down to subordinate objects. In Novell NetWare an example would be rights that flow from an organizational unit down onto the users located below it. As rights flow down the hierarchal structure the users lower in the structure gain the accesses granted above them.

The other benefit of this comes into place regarding role design. This hierarchy can greatly reduce the number of roles created since rights can be combined by this rights flow or progression. Another benefit as shown in the graphic below is that multiple roles can be associated with each other to allow greater functionality for the end-user.

User 3 has been granted access to the "Tech" role, which gives him the functionality associated with that role as dictated by the *role engineering* process. The arrow that says "Not" connecting the "Tech" role to the "Jr. Admin" role is a constraint halting the user associated with the "Tech" role from being assigned the "Jr. Admin" role. If an administrator were to try and associate User 3 with the "Jr. Admin" role they would receive a message that this couldn't be completed as requested.

The same concept is carried out for User 2 and the "Jr. Admin" role in relationship with the "Sr. Admin" role. The "Sr. Admin" role is configured differently however from the other roles in that it is designed to inherit the access or capabilities assigned to the "Jr. Admin" and "Tech" roles. This access will be in addition to the access granted to the "Sr. Admin" role. So in essence by User 1 being associated with the "Sr. Admin" role he is associated with all three roles though only explicitly assigned to one.

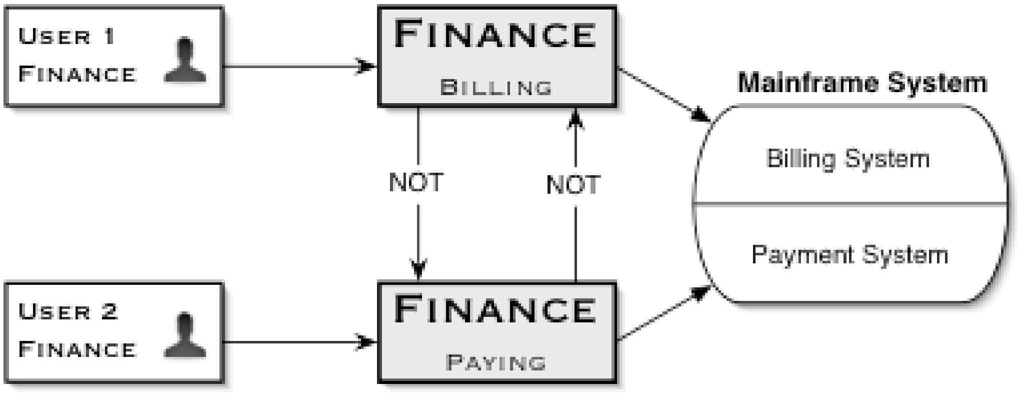


Graphic 3

RBAC: Separation of Duties

$RBAC_3$ can allow and enforce the separation of duties through the use of constraints, similar to those of Graphic 3. Separation of duties disallows a user with a certain job function to serve in another job function at the same time. This is very useful in enforcing regulations such as those required by the Health Insurance Portability and Accountability Act (HIPAA). It also makes good business sense, especially in areas such as finance departments where accounts receivable and accounts payable access to a single user could prove too much of a temptation.

Graphic 4 shows how constraints could be used in this situation to ensure that a single user doesn't have access to both portions of a finance system.



Graphic 4

User 1 has access to the Billing System residing on the Mainframe system through the "Finance Billing" role. User 2 likewise, has access to the Payment system on the same Mainframe through the "Finance Paying" role. The arrows stating "Not" represent constraints in place enforcing that a user associated with one role **cannot** be associated with other. These constraints enforce the separation of duties.

Advantages and Disadvantages of $RBAC_3$

We have briefly touched on both Mandatory and Discretionary Access Control and the advantages and disadvantages of each. MAC with its hard coded security lacks scalability and the ability to adapt. DAC offers scalability though it comes at the price of productivity and has security vulnerabilities in that the "discretion" of the administrator could be manipulated by false information or through social engineering.

We have also reviewed basic approaches to how $RBAC_3$ can be implemented into a network environment. The question remains were the original goals of the "RBAC Project" met: standardization, scalability, logical in construction, non-system dependent and positive economical ramifications?

Scalability:
Pros:

While DAC is a scaleable means of access control, the intelligence required for each modification of access, makes it time consuming and cumbersome in a large environment. The administrator to end-user ratio will need to remain static throughout infrastructure growth, which can lead to a large, and costly support staff and an increase in errors.

$RBAC_3$ is scaleable as well, provided that your organizational policy, that is your organizational layout, is strong and well documented. In an environment where this is the case, roles can be created by *role engineering* and modified only as needed. A benefit to this is that since access is granted in essence to groups of users, individual administration of accounts is greatly reduced if not eliminated. Since *role engineering* has developed the package of permissions for each user, the intelligence needed in DAC to distinguish appropriate rights for users is built in before implementation. As the organization grows more roles may be needed, however since $RBAC_3$ supports a hierarchal design allowing rights to flow down the tree and to rights to be constrained, this can be reduced some.

Cons:

Where $RBAC_3$ can create headaches is during an implementation where the organizational policy is poorly documented or not adhered to. $RBAC_3$ in these instances can become more of a hindrance than a blessing because the organizational policy or layout defines model. It is assumed that as the organization grows it will be in a logically constructed manner, thus your security model can adapt and adhere to the changing organizational policy that the model is based upon. Failure to follow a detailed organizational policy can introduce in essence "dead ends" that limit your models scalability. This will require some work to implement a redesign to get back on track, and/or introduces "work around" solutions such as the creation of additional unmodified roles and in the long run can contribute to the problem rather than rectify it. Adding additional unnecessary roles because of a poor organizational policy will increase the administrative workload and over time can increase the administrator to end-user ratio to resemble more of a DAC implementation. Thus the old adage, "Fail to plan, plan to fail" applies here, for without the appropriate pre-planning, the cost of implementation can grow exponentially, and the return of investment will be minimal to compound this added cost.

Security
Pros:

The fact that *role engineering* occurs as a precursor to implementation addresses some of the security vulnerabilities such as administrator error, false information, and social engineering that are inherit in the DAC security model.

In comparison with the security offered through the MAC security model, arguments have been made that RBAC is comparable. An argument by David Ferraiolo and Richard Kuhn in 1992 at the 15th National Computer Security Conference stated: "RBAC is in fact a form of mandatory access control, but it is not based on multilevel security requirements."

The argument that RBAC is a form of mandatory access control stems from the fact that MAC is dependent on data and user classification or labels, and RBAC uses RBAC roles as a form of classification. Through the use of hierarchies, rights inheritance, and constraints the argument could also be made that RBAC is a "multilevel" security model. However, RBAC does lack the hard coded security classifications that MAC offers, a large consideration for military security, and the primary implementer of the MAC security model. The limited use of MAC perhaps makes this a moot argument. However, $RBAC_3$ and its scalability along with its security could perhaps be a reasonable solution for state, county, or local government looking to secure a system and considering the MAC security model.

$RBAC_3$ also offers the ability through *role engineering*, hierarchy, and constraints to make roles as granular as needed to secure a system. This would of course increase the number of roles required and would increase the number of administrators needed.

In the end the security offered by $RBAC_3$ is impressive, but it is its security along with its scalability and ease of administration that makes it such a powerful security method.

Cons:

It is the practice at many organizations to deal with security issues only as they become known. This type of security practice opens the door for lax and loose security implementations that allow greater access than needed. In RBAC security administrators must have intimate knowledge of how permissions are being granted, why, and what operations are associated with those permissions and roles. It is a very hands-on practice, which while it is very proactive, is also expensive in requiring skilled and knowledgeable staff. Failure to have properly trained and competent staff can allow $RBAC_3$ to become as insecure as any other type of implementation. So if a company is only as good as its people, $RBAC_3$ security implementation is only as good as the staff you have supporting it.

Logical in Design

Pro:

$RBAC_3$ if anything is logical in design because it is based on the positions within the organization based on an organizational policy. It today's environment it is rather common to design network structures based on location. I have worked for a large hospital that divided its Novell NetWare Directory Services tree based on the location of over twenty campuses. This was logical from a data backup and replication strategy. In 1999 when I designed a NetWare NDS tree from the ground up for the government at the county level, I too followed a location-based design. With $RBAC_3$ a logical location based design at the highest level of the tree is a responsible approach, however, after this initial division, the remainder of the tree should follow a position or role based design. This is very logical as the equivalent positions on multiple campuses may use differing systems and applications, thus requiring separate roles.

This logical design will allow for easier administration and will allow for adaptation and inclusion of new entities should a merger or expansion of the corporation occur. When it comes to investigating misuse of resources or permissions, this logical hierarchal structure will help reduce confusion, which in turn may rectify misappropriation of rights situations from the onset. If the goal of the organization is to have a separation of duties, this logical design can assist with this greatly as well. Just by looking at a NDS tree with $RBAC_3$ in implemented into the design you will easily be able to see which user is associated to which role, what rights those roles offer and whether the separation of duties is present. With DAC this would involve looking at a number of users within a department as see their effective rights to a system or application. Rather than fixating on the user, which there could be dozens, you can focus on the role of which there should be two separating the duties.

Con:

Where $RBAC_3$ can be frustrating is during implementation if the infrastructure is and large pre-existing. In some organizations this may nearly require a duplication of existing server and infrastructure hardware to create a separate $RBAC_3$ network, and then migration to that network. This will tie in greatly to your cost and return on investment as address in the "Economical Ramifications" portion of this document.

The logical design goal of $RBAC_3$ also plays into the scalability as well. As stated before administrators will need to have intimate knowledge of the organization and will need the knowledge and skill to develop a hierarch tree structure and develop roles. Even more so, once the roles are developed they need to be placed appropriately into the tree. This requires a pre-planned implementation with skilled engineers and project managers along with intense conversation with the business side to understand their needs and organizational structure. In short, the logical design of RBAC$_3$ comes from intense research and design testing before implementation by skilled personnel.

Non-System Dependent and Standardization

The question of the advantages and disadvantages of a non-system dependent and standardized RBAC model is less of a priority than if the goal was accomplished. While proprietary systems such as Cisco's programming language for their routers and devices are beneficial, most IT professionals will agree that a cross-platform implementation of an enterprise wide security model should be anything but proprietary. Since it is very common for systems to interact with each other such as Novell NetWare and Windows NT, RBAC$_3$ needs to be able to function with it elements in each environment.

The document "The Economic Impact of Role-Based Access Control" prepared by RTI for the National Institute of Standards and Technology has a sampling of vendors that currently offer products using the RBAC$_3$ security model. This list is not a list of all vendors and does not include in-house developed applications (Gallaher, p.44).

- Access360, Inc.
- Adexa, Inc.
- BEA Systems, Inc.
- Cisco Systems, Inc.
- Entrust, Inc.
- Entrust Information Security Corp.
- International Business Machines Corp.
- Internet Security Systems, Inc.
- iPlanet E-Commerce Solutions
- Microsoft Corp.
- Network Associates, Inc.
- OpenNetwork Technologies, Inc.
- Oracle Corp.
- PGP Security, Inc.
- Protegrity, Inc.
- RSA Security, Inc.
- Secure Computing Corp.
- Siemens AG
- SETA Corp.
- Sun Microsystems, Inc.
- Sybase, Inc.
- Symantec Corp.
- Systor AG
- Tivoli Systems, Inc.
- Vignette Corp.
- Baltimore Technologies, Inc.
- BMC Software, Inc.
- Novell Corp.
- Radiant Logic, Inc.

RBAC$_3$ is non-system dependent, however in looking at documentation between various vendors the agreement on terms is lacking. Permissions may be titled privileges or rights; depending on which vendor you are dealing with. The concept of RBAC$_3$ is intact, but standardization on the terms and wording is currently not set.

Since NIST is proposing its concept of RBAC$_3$ as the Role-Based Access Control standard, it is premature to tell if they will be successful. In the past we can recall the Hayes and US-Robotics debate on the 56k modem standard later to be determined as v90. DVD players and drives have various formats, some of which existed in the battle for an early standard. The Zip Disk beat out the LS120 super floppy in the competition of disk storage allowing Zip Disk to appear as the standard to users and making LS120 disks nearly impossible to find.

In my experience, standardization has come only because of necessity, whether it is mass production or a company able to bully its standard over another by sheer volume. Until RBAC is implemented more widely and issues arise over cross platform integration it may be impossible for NIST to claim it has successfully developed the true standard. The fact that a standard does not exist may deter some from implementation though my personal opinion is that would be flawed thinking. If after thoroughly reviewing the benefits of MAC, DAC and RBAC, you find $RBAC_3$ a solid solution, cost effective, and are willing to plan well for its implementation, then why not proceed and reap the benefits. While we should not walk blindly into the future, capitalizing on the technology of today will put us in a great position to embrace the achievements of tomorrow.

Economical Ramifications

My original hope was to show how positive the economical ramifications of $RBAC_3$ would be as a clincher to my research. However, in my reading and applying my own personal experience of working in a company with 36,000+ system accounts, I had a hard time trying to come up with solid numbers. In the beginning of this paper I wrote:

> "Please note, that while this paper explains many of the benefits of RBAC, a security administrator, analyst, or architect, must always take into consideration the needs and capabilities of their environment before ruling out any security model."

That in essence is my argument for determining the economical ramifications of $RBAC_3$, it must be determined based on the variables of your environment.

In my past environment at a hospital, the cost of implementation would soar because a duplicate hardware configuration would be needed. Take into consideration the cost of the hardware, space for the hardware, then the cost of the implementation along with any licensing issues that may arise in having duplicate user accounts and you are easily pushing the one million dollar mark.

In contrast, at the county government during my redesign of the Novell NetWare NDS tree I could have implemented $RBAC_3$ without any additional hardware or space costs, and probably minimal administrative cost in the *role engineering* phases. Of course this was with the luxury of over six months preparation time, separate server hardware that was to replace our current systems, and the use of "VImport" a utility by Visio for rapid user account creation, on a network of 500 users.

So for me to argue the positive economical ramifications of $RBAC_3$ and offer a percentage of savings or return on investment each year would be illogical and irresponsible. Rather my recommendation would be to take into consideration the following issues before continuing forward with the implementation of $RBAC_3$.

- Consider the number of users that will be affected by the implementation?
- Will additional user or server licenses need to be purchased for compliance?
- Is down time involved, how much, and can it be afforded?
- Will this be a phased move on production systems, or will it be a parallel network that users will be phased over to.
- What server components will need to be purchased: hard disk, RAM, extra processor power, SAN or NAS solutions, backup hardware and media?
- Services contract costs, will they increase? By how much?
- If servers are purchased how will that affect other vendor licensing on that server for programs such as faxing, backup, etc.?
- Is space, power, or cooling and climate control an issue within your server room? Will additional hardware be needed to address those issues?
- How will this implementation effect your system replications, backups, or disaster recovery plan?
- What additional infrastructure hardware will be needed: wiring, routers, switches, hubs, fiber, and external connections such as frame-relay.
- What does your risk analysis or gap analyses reveal that you have missed in your design planning?
- Is the business side supportive? Is executive management on board?

These aren't nearly all the considerations that should be had, but are just a few questions that will contribute to your assessment of the economical ramifications. While the argument for security and scalability along with administrative ease are very strong and capable, the bottom line of cost is often times the "Rock of Gibraltar" that halts the progression of project implementation.

Conclusion

$RBAC_3$, when properly implemented following a well-defined organizational policy, can allow for a very scaleable, logical, and secure means of distributing access to file systems, applications, sub-systems or the like. Its ease in administration can negate some of the security vulnerabilities the DAC model is prone to while maintaining some of the same security characteristics of the well respected but limited in use MAC model. While standardization would be an overstatement for $RBAC_3$, even with the work of NIST, its concept is accepted and endorsed by several large information technology vendors allowing the implementer to have confidence in the technological concept and it's future. With any implementation cost will be factor that will need to be assessed upon the due diligence phase of the design. However, the benefits of $RBAC_3$ in the light of governmental regulations such as the Health Insurance Portability and Accountability Act (HIPAA), and accreditations offered by agencies, may be worth the peace of mind it brings.

References

Cross, Michael; Johnson, Norris L.; Piltzecker, Tony; Shimonski, Robert J.; Shinder, Debra Littlejohn. "Security+ Study Guide and DVD Training System" Rockland, MA. Syngress Publishing, Inc. 2002

"An Introduction to Role-Based Access Control"
NIST/ITL Bulletin, December 1995
URL: http://csrc.nist.gov/rbac/NIST-ITL-RBAC-bulletin.html (09/16/03)

Ferraiolo, David; Kuhn, Richard "Role Based Access Controls"
Reprinted from: 15th National Computer Security Conference 1992
URL: http://csrc.nist.gov/rbac/Role_Based_Access_Control-1992.html (09/16/03)

Gallaher, Michael P.; O'Connor, Alan C.; Kropp, Brian
"The Economic Impact of Role Based Access Control" (03/02)
URL: http://www.nist.gov/director/prog-ofc/report02-1.pdf (09/16/03)

Andress, Mandy. "Reach out and ID Someone: Access Control"
Information Security, April 2001
URL: http://infosecuritymag.techtarget.com/articles/april01/cover.shtml (09/16/03)

Ferraiolo, David F.; Sandhu, Ravi; Gavrila, Serban; Kuhn, D. Richard; Chandramouli, Ramswamy
"Proposed NIST Standard for Role-Based Access Control" (08/01)
URL: http://csrc.nist.gov/rbac/rbacSTD-ACM.pdf (09/16/03)

Secretariat: Information Technology Industry Council (ITI)
"Role Based Access Control: Draft 04/04/03" (04/04/03)
URL: http://csrc.nist.gov/rbac/rbac-std-ncits.pdf (09/16/03)

Chandramouli, R. "A Framework for Multiple Authorization Types in a Healthcare Application System" (17th Annual Computer Security Applications Conference – ACSAC, December 2001
URL: http://csrc.nist.gov/rbac/rmouli_healthcare.pdf

The document author created all graphics included in this document.