Global Information Assurance Certification Paper

J.D. Edwards Security using RBAC

Scott Gordee
SANS GSEC Certification
Practical Assignment 14b

July 23, 2003

J.D. Edwards OneWorld Security Using RBAC

Scott Gordee

Abstract

OneWorld security is incredibly flexible, but can also become convoluted and difficult to manage if a security model isn't created and enforced in the infancy of its implementation. The main problem stems from the rule that a user may have only one group. All of the user's permissions must be applied to that one group. It is a management nightmare that we intended to remove from our dreams.

By using a variation of a Role-Based Access Control (RBAC) model and adhering to its principles, we were able to provide a quality security definition that met our business needs. The business needs were essentially to follow the CIA definition of security and be flexible enough to provide a granular solution. Confidentiality, integrity and availability requirements led to various security rules to prevent all user access unless specifically granted. Our logical security roles helped provide a granular solution by separating the users' duties into small groups. Even with challenges and obstacles the implementation has been successful.

J.D. Edwards OneWorld Security Using RBAC

As a highly configurable ERP solution, J.D. Edwards provides a lot of flexibility to its customers, and therefore complexity to its configuration. Security for an enterprise-wide project can become convoluted if a model isn't designed and enforced. Our team was recently chartered with this task of implementing security in J.D. Edwards OneWorld on an IBM I-Series.

The Problem

The first things we identified in our upcoming implementation were the goals and requirements of the users. Our customers required that we maintain the integrity of the data by allowing only the specified people to update information. They also stated that no user should be allowed to view anything unless it was specifically allowed. The opposite side of the same coin also implies that users should be able to do their jobs and have access to the necessary information. The final requirement was that the implementation had to be flexible so that we could put and take security as needed.

Our business needs flowed smoothly to the concept of CIA. Confidentiality. Integrity. Availability. The data needed to remain confidential and visible only to the specified people. It could only be updated by the specified people to help insure data integrity. The people also need to do their jobs and must have the applications available for them to use. The only piece missing is the flexibility.

J.D. Edwards can provide confidentiality, integrity and availability fairly easily due to the detailed nature of its security tool, Security Workbench. However, if you want a thorough implementation, it can become a management nightmare.

First I must describe some of the security that is available to professionals who implement J.D. Edwards OneWorld. I will describe the security surrounding users and groups, as well as tell you some of the issues we encountered. Finally, I will describe the model we used to meet our business needs.

Users

In most cases, the setup of users is like any other system. You have a unique identifier, a password, some settings for password expiration and failed logon attempts and so on. There are a few differences that need to be noted.

There are no description fields in the user profile. All external information is stored in a J.D. Edwards address book entry. The user profile is linked to a unique address book number, which stores the descriptive information. Security officers should learn what the address book has to offer them and the fields that

are used by the developers because the address book will be used extensively in the J.D. Edwards implementation.

Groups, or system roles, are vessels for your security definitions. You don't necessarily need to use them, but it makes user management easier. A user can have one and only one group. After you have seen everything involved in defining security for a group, you will see how large this restriction is.

System user ids and passwords are another important piece to a functioning profile. J.D. Edwards can run on various platforms. One such platform is the AS/400 or I-Series. A working J.D. Edwards user id needs to have an id on the AS/400. The J.D. Edwards id must pass the AS/400 id and password to the AS/400 when it logs in. This is, in essence, a version of unified logon. The user doesn't need to know the AS/400 id or password even exists because they will never need to enter it, but they won't be able to log in without it. It must be correctly linked and in synch with the id on the AS/400.

Even though you can link every J.D. Edwards user to one system id, I would recommend against it for three reasons. If you would need to change the system password for any reason, you would also need to change every J.D. Edwards profile. If the system id gets locked then every user won't be able to sign into J.D. Edwards. The third reason is that it helps troubleshooting problems on the system. The logs can identify a system id, but if every user has the same id it is difficult to differentiate between users.

System Roles

System Roles are loosely similar to groups for Active Directory. Permissions are applied to the group and the users are given groups. However, in this case the user to group relationship is not a many to many relationship. A user can be assigned only one System Role.

Security Workbench

A system role is comprised of many lines of security permissions and denials. A line of security can grant access to an application, allow an action to be taken, make a function visible or even prevent its use. Which security is applied is completely up to you, the customer, because the range of definitions is very broad and flexible. There are nine types of security that can be applied to a system role, user or *public. They are: Solution Explorer, Application, Action, Row, Column, Exit, Tab and External Calls security. They are all granted through a powerful tool called Security Workbench "Although your company may not need all of these different types of security, it is a good idea to be aware of the functionality that OneWorld offers." (Miller, p. 562) Let's look at what Security Workbench has to offer.

Solution Explorer Security

Solution Explorer Security controls some of J.D. Edwards' extras. With it, you can disallow users from going to the Internet within J.D.E or viewing Fast Path/Find It. It controls some of what can be done within Solution Explorer and the main menu.

Application Security

Perhaps the most important security definitions are applied with Application Security. Application Security determines whether a user can or cannot run a program. It's as simple as that. Some applications have many forms within them. You can individually secure those forms from being run as well. Of all the types of security in J.D. Edwards, this is the one I've used the most. It is also the most straightforward.

Action Security

The next most used type of security in J.D. Edwards is Action Security. As the name implies, you can secure six types of actions a user can do inside an application. They are: add, delete, copy, select or OK, change, and scroll to end. Four of these are directly related to the buttons at the top of an application. Add delete, copy and OK/select all have a button representing them on the screen and are self-explanatory. Change represents the user's ability to make a change in the form and save those changes by clicking OK. Scroll to end is simply the user's ability to use Ctrl-End to scroll to the end of the data in a form.

When action security is applied, the button will be grayed and unable to be selected. You can only secure the four buttons I mentioned. Other ones like About and Find are not able to be secured. Remember this tip about action security. When you want to allow the user to be able to click on an OK button, you must also give that user Change authority to that form or application. OK acts as a "save changes" button.

Row Security

Row security allows the administrator the ability to restrict or allow users to add, change, delete or view a range of rows in a table. Here's an example of how it works.

F0101 AddressType From='C' To='D' View=Y, Add=N, Change=N, Delete=N

This is the expression as it may appear in J.D. Edwards. Any table has rows and columns. Row security works with from and to values. I could restrict a user so that whenever they use an application that works with information from a

table (F0101 for example) they can only see rows where the field, AddressType, has values from C to D and everything in between. This also restricts those rows in the table to view only. It even overrides whatever security may be used on the applications using the table. So if the action security says that the user can change the data, the row security says it can't.

There are a few catches to row security though. Unlike any of the other security types in J.D. Edwards, a lot of row security can tax the system resources. It's best to use it sparingly unless you want your system folks breathing down your neck. Also, because the row security affects tables, any program that uses that table's data is affected. It can cause unwanted results and some things may be too secure for the users' tastes. Here's an example. Say we have two programs that access different parts of the same table. If row security is needed for only one program it will still affect the user's access in the program that doesn't need to be secured. The programs might not even update or display the same fields in the table, but since it is contained in one table, it is secured. To fix this, you would need to add more lines of row security to open the table up enough, but remember that would tax the system. It's a catch-22.

Column Security

Like row security, column security can be applied directly to the table. The exception is that column can also be applied to the application and form. Some also call it field security. A form might have many fields, and only some of them need to be managed by the user. You can secure each field to be updateable, grayed and unable to be updated, or not even visible. Even if you leave the action security so the user can update the form, you can still secure any field with column security to prevent them from changing the data. Also like row security is the column security's taxing of the system if it is applied in large numbers to the table itself. I haven't heard of any complaints if you apply it to the application or form.

Exit Security

In most applications, there are row or form exits. They are links to other applications or forms that the user can select. I view exit security, called Hyperexit security in J.D.E, as an example of security in layers. An exit can be secured so that the user cannot select it. The form that is accessed through the exit can also be secured through application security. This is because even if the button can be chosen, the application can't be run if it's not allowed. Exit security can be used to force a user to get to that application another way if possible, like through a menu item instead of an exit. So, the layers are in exit security and application security. If they can't run the program, they're secured. If they can't choose the exit, they can't run the program if it's not provided in other places.

Tab Security

A form might have many tabs on that form. They look like the tabs in an Excel workbook, just near the top of the form instead. Some of those tabs may contain sensitive data or be filled with unused fields. In these cases, you may want to remove them from the users view.

External Calls security

External calls are considered applications outside of the J.D. Edwards executable. However, they are standalone applications that can be accessed from menus within OneWorld. “An example of a standalone application that you probably want to restrict most of your users from using is the Universal Table Browser application.” (Miller, p. 580) The Universal Table Browser allows users to view any table they want and if you haven’t applied row or column security they will see everything. This will negate any security you’ve applied for confidentiality reasons.

The J.D. Edwards Security hierarchy

Security definitions can be applied to the system role, to the user, and to *public.
Since the security can be applied to all three places, we need a pecking order for the security that takes precedence.

Security applied to the user takes the most precedence in the hierarchy. “The absence of these values for a user profile causes OneWorld to defer to those values defined by a user’s group profile.” (Vidovic) The order is followed by the system role and ultimately *public. The security can be conflicting in the different levels because J.D. Edwards follows a search pattern to determine authority.

*Public could be considered a default system role. Its security is applied to every authenticated user in J.D. Edwards. We chose to allow nothing to be run unless we specifically granted it to a user. That can be done here. “It only takes the following single line of OneWorld application security to switch from the “include everything unless I exclude it” policy to one in which you “include nothing unless I include it”” (Miller, p587)

*PUBLIC *ALL RUN=’NO’

Our groups then required each application the user would need to perform their duties were granted back to them.

GROUP1 P0001 RUN=’YES’

This allows them to run P0001 and nothing else because the security is read and granted at the group level before it is even read at the *public level. If we were to apply a line at the user level that contradicts the group permission, the user level takes precedence

USER1 P0001 RUN='NO'

The user level is read first, so they can't run P0001 now. It is very important to understand the hierarchy with which J.D. Edwards grants access. I recommend that if you create a line in*public to restrict everybody from everything, you also create a user or group that is allowed to run everything. Companies have locked themselves out of important tools by not having that all-powerful user.

Job Roles versus System Roles (Groups)

Job Roles are another user attachment in J.D. Edwards. They can restrict users to seeing only portions of the menu and thus it limits them to only the tasks they can view.

Why would a person do all the work of defining security for each screen when a job role can restrict that person to see only the tasks they can use? Job roles are not what I would call security. They make the menus and tasks visible to the user and restrict that same user from seeing other menus and tasks. However, they do not actually restrict them from running the application. They will not necessarily provide confidentiality, keep the data integrity. They can help with availability by trimming the menu to only the necessary tasks, but it is window dressing. Do not be fooled. Job roles only give the impression of security

As an example, let's say we restrict the user from using menu item "Journal Entry" we'll say is application P0001 by not allowing the user to see the application in the menu structure via their job role. There may be another menu item called "Journal Maintenance" as application P0002. The user may open "Journal Maintenance", view the information, and click on a row exit called "Journal Entry". The exit, "Journal Entry", is application P0001, just accessible through another doorway. Our user has just accessed the very application we hoped to secure.

By using application security to restrict the user from running application P0001, we can prevent the user from running the program no matter what it is called, where it is on the menu, or how they try to get to it.

Preliminary Decisions

Our goals were to secure J.D. Edwards access to help prevent data corruption, unauthorized access and malicious damage. It was decided early in

the design phase that users would use individual user ids to access the system and that those ids would have only the access necessary for them to do their duties. This would force us to know exactly what each user could do when they were signed into J.D. Edwards. It also protected us from accidentally allowing users to perform a detrimental system task accidentally. Each application would be scrutinized and nothing would be missed. We restricted every application and action at the *public level with these commands.

```
*PUBLIC     *ALL          RUN='NO'
*PUBLIC     *ALL   ADD='NO', COPY='NO', CHANGE='NO',
                   DELETE='NO', OK/SELECT='NO', Scroll to End='NO'
```

Our plan was to grant the needed applications and actions back to the users through the group or user levels. We decided that it would be best to use the groups instead of maintaining each user's security individually. Because J.D. Edwards allows users to be given only one group, this presents a problem with providing any type of granularity. It almost forces the administrator to apply the different security to each user on an individual basis.

Unfortunately, a user's complete duties could entail hundreds of lines of security in order for them to actually perform those duties. In Windows Active Directory, you have multiple groups applied to a user. Each group has different permissions that are assumed by the users with that group. Imagine if you could only have one group and all the permissions a user needed to access had to be applied to that group. It would be an administrative nightmare.

You're incredibly lucky if you can pigeonhole your users into a small set of groups such as administrator, manager and user. With the size of our company, that would be quite impossible.

We really needed the ability to give multiple groups to a user so we could be flexible in what security is applied to the users and still keep it manageable. The key is found in the P00950 Security Workbench application and it's called a Copy tool. With it, you can copy the security definitions in one group to another. You can either use the Copy Applications form exit to copy an entire group, or the Copy action button to copy selected lines.

Logical RBAC to the Rescue

Role-Based Access Control (RBAC) is the concept of applying security based on the user's duties and as a whole versus in small pieces and a permission at a time. RBAC can save administration time on the part of the security officers, which can in turn save the company money.

Instead of having the users request security to a single application, they are given a large group of applications. The users are fit into roles based on their

duties. It is much faster than asking the user which applications they need. “Do you need this one? How about this one? And that one?” Role-Based Access Control is a time saving tool that can be used in many systems and devices. We felt it could be applied to our J.D. Edwards implementation.

Since the user community requested that we be flexible with our implementation, and since we could only apply one group to a user, we needed to do something different. We wanted to make something granular, yet role-based.

We decided to create a logical entity in J.D. Edwards called a Security Role. A security role is technically identical to a group. It has the same definitions and is in fact of the type *GROUP. The only difference is what you put in the security role versus what’s in the group. This is how we provide flexibility and granularity to a one-to-one user-to-group relationship.

A security role contains the security definitions for a set of tasks that one type of user would use. It is specific to a department and does not deviate from that department’s realm. Here is an example. In accounting, there may be different duties that the various users have as their responsibility. We would identify those different tasks and place them in different security roles. They are named Accounting Administrator, General Accountant, and Budget Analyst and so on. These security roles contain the security needed to do that set of accounting tasks, but nothing else. They are accounting specific.

Then there may be a Warehousing set of roles, such as Warehouse Manager, Warehouse Materials Handler and a Warehouse Inquiry role. They contain only warehousing duties. This continues for every set of tasks that will be done in J.D. Edwards. By doing this, we created granular roles in J.D. Edwards. Please note that these roles need to be defined by the people who will be performing these duties. Only they know what really is needed. Help them with their roles, but the decisions must be made by them.

Creating the Roles and Groups

Using the various types of security definitions described above like action, application and row security, we created each security role with a unique name in J.D. Edwards. I would suggest using a naming convention to separate groups from security roles, especially since they are of the same type in J.D. Edwards.

Once the roles are created, the business process designers would identify who among the employees was required to perform the tasks. This told us which users would receive which security roles. We created a grid with users on the side and security roles across the top. This identified the unique combinations of security roles that our user community would need. So the first user may be

allowed to have Accounting Administrator and Warehouse Inquiry. That's one unique combination.

When all of the unique combinations of security roles were identified, we used the copy tool to copy the various security roles into unique groups. The users were given the group that matched their collection of security roles. By using groups, we can change a user's security quickly without having to change a lot of the security definitions. It provides ease of management and saves us some time.

Maintenance

In order to maintain this model so that you can always make the assumptions that the groups are a collection of security roles, and that security roles are a specific set of tasks, you need to keep track of the security role to group relationships. Because J.D. Edwards doesn't recognize the difference between a security role and a group, it must be done externally. We chose to use an Access database, but I believe it could be maintained in a spreadsheet as long as your roles or groups don't exceed the number of columns allowed.

Once you are maintaining those connections, the next piece required for maintenance is easy. Any change in security must be applied to the security roles and then copied to the groups. If a change is made directly to a users group, then it should be changed in a security role that user has or a new security role needs to be made to reflect the addition or change. If you only make changes to the groups and not the security roles, the model breaks down. This is because the groups should get their security definitions from the security roles. If a group is accidentally deleted, the only way to create it again is by copying the security roles into the group. If you've made changes to the group alone, those changes would be lost when the group is recopied.

The Copy Problem

Like any model designed to augment a system beyond its original specifications, there exists a problem with the process. The copy problem pertains to action and row security mainly, but it can affect other types of security in J.D. Edwards.

This may vary with different versions of J.D. Edwards, but in OneWorld the issue will exist. Here is the crux of the problem. Let's say we have two security roles that we will copy into a group. Let's also say that both roles contain security for applications P0001 and P0002. Security role A has all action security for P0001, but only the select button for P0002. Security role B has the opposite: select only for P0001, and all action for P0002.

We copy security role A into the group. The group now has P0001 with all action security and P0002 with select only. Now we copy security role B. One might expect that the group would adopt the highest security available and have all action security for both applications. That is not the case.

What actually happens is the group keeps only the first entry put in the group. It does not overwrite, add or remove security for the same application and of the same type if it is already in the group. So even though our intent is to give all action security for both applications, only the first role copied will retain its values in the group. If we copied them in reverse order, security role B's security would be preventing security role A's security.

What can be done to prevent this? One fix is to remember which roles need to be copied first because they have the greater security. That only works if you don't have a situation like we described above. Another fix is to copy them and fix the problem in the group itself. This will work for one time only. If you had to remake the group, you would need to make the fixes each time you copy the group. The only true way to fix the problem is to remove the duplicate entries from both roles and create two new and very small roles. Then you can truly control whether the groups have the greater or lesser security because it fits in the model.

Conclusion

The role-based model I've proposed requires some discipline to work. Let's review. The security roles consist of a different scope of duties for each department. They are copied into a group, which can be given to multiple users. Each different combination of security roles creates a new group. Security is always changed in the security roles and then copied to the appropriate groups. Beware the problem that comes with the copy function. The options for securing an application are many, but the more detail required makes it more complex. Remember to keep track of the security role to group relationships outside of J.D. Edwards.

Our goal was to create a security model for J.D. Edwards OneWorld that can be flexible, granular and remain secure through confidentiality, integrity and availability. The security for J.D. Edwards can become convoluted quickly, but with a structured design and some discipline, the desired results can be achieved. Using Role-Based Access Controls, we are able to maintain security for almost four thousand users with less than two hundred roles. This security model has met our customers' needs and saved us some time and money in the process. I am certain that RBAC can do the same for you in this and other systems.

References

J. Miller, A. Jacot, J. Stern. J.D. Edwards OneWorld, the Complete Reference. Osborne/McGraw-Hill. Berkeley. 2001. Pgs 523-616.

A. Vidovic, et al. “J.D. Edwards OneWorld Implementation”. IBM Redbooks Abstract. Chapter 15: Security. http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg245195.html?Open

The Information Security Glossary. “Confidentiality, Availability and Integrity.” http://www.yourwindow.to/information-security/gl_confidentialityintegrityandavailabili.htm

Barka, Ezedin and Sandhu, Ravi. Framework for Role-Based Delegation Models. http://www.list.gmu.edu/confrnc/acsac/rbdmacsac00(org).pdf

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

D. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, R. Chandramouli, "A Proposed Standard for Role Based Access Control (PDF)," *ACM Transactions on Information and System Security* , vol. 4, no. 3 (August, 2001) - draft of a consensus standard for RBAC. http://csrc.nist.gov/rbac/rbacSTD-ACM.pdf