Realtime publishers

The Essentials Series: Enterprise Identity and Access Management

Authorization

sponsored by

QUEST SOFTWARE®

by Richard Siddaway

Copyright Statement

Authorization

Authorization is the process of granting users appropriate access to resources once they have established their identity through authentication. In a Windows environment, access permissions should be assigned to groups rather than individuals and should be based on roles and established policy. The first decision, though, is what data needs to be protected. Unfortunately, in non-Windows environments, controlling authorization can be more difficult, as these systems don't benefit from the role-based management paradigm of Active Directory (AD).

What Needs to Be Protected?

The starting point for this discussion is the file system. The data is accessed through shares and can be protected by permissions on the shares and the file system itself. One concept that should be avoided is the "open file system" where all data is available to all users. There may be large proportions of data that should be available to many, if not all users, but even that level of access should be controlled. It would be very surprising if an organization did not have some data that was deemed to be confidential and that should have a restricted access policy.

It is not just file system data that needs protecting. The identity-related data stored in AD must be protected, as should the data held in databases and applications. AD has the ability to delegate control of subsets of the directory to groups of users. This can reduce the number of administrators with access to the whole directory. Delegation in this way is a very time-consuming activity that requires a lot of work to get right and maintain. If a large amount of delegation is required, it would be more efficient to obtain a tool that can manage the delegation of permissions within AD.

Database systems have their own methods of granting permissions to work with the data. These methods may be based on AD groups, which can simplify administration—especially if the database system supports the concept of roles. Assign the users to a group and assign the group to the role.

Non-Windows–based systems will also have data that needs to be protected. Even though the methods of granting permissions may be different, the end result is the same. Only those users who are authorized to do so may access the data.

Anonymous access to data (that is, access without authentication) should be discouraged. If an organization does not know who is accessing its data, then it is vulnerable.

Authorization should not be restricted to the standard modes of access—reading, writing, modifying, and so on. Most users have access to email in the corporate environment. Consider the use of Rights Management systems to further control what users can do with confidential data. If they can read a file, they can email it out of the organization. Using AD-based Rights Management leverages your existing investment and extends the level of protection afforded to the data. Once the data that needs protecting has been identified, appropriate permissions can be assigned to allow access to the data. These permissions should not be assigned to individuals; full use should be made of AD groups. And, where possible, those groups should be leveraged beyond AD to provide authorization for non-Windows systems, data, and applications.

Groups

AD supplies three security group types, as Table 1 shows.

Group type	Membership	Permissions
Domain local	Across trusted domains	Within the domain
Global	Within the domain	Across trusted domains
Universal	Across the forest	Across trusted domains

Table 1: AD security groups.

The stated best practice is that user accounts are put into global groups. Global groups are made members of domain local groups, which are granted permissions. Universal groups are used to hold global groups across multiple domains in the forest and are put into the appropriate domain local groups.

In many cases, this practice is not followed. What often happens is that groups are created without planning. This can lead to

- Multiple groups being granted access to the same data
- Conflicting permissions due to a user's membership in multiple groups
- Groups being created because the administrators are not aware of pre-existing groups

Plan and document the groups to avoid these situations. When designing a strategy for using groups, consider adopting the concept of role-based access. Complicating matters even more is the fact that, for non-Windows systems, the AD-based group infrastructure cannot natively be applied to provide unified authorization across platform boundaries.

Role-Based Access

Windows controls access to resources by using Discretionary Access Control (DAC). When using DAC, the resource owner, usually the administrator determines who can have access and sets the appropriate permissions.

Role-based access works by defining a role to cover the job needs of a group of users. The job needs can be defined, for instance, as the ability to

- Access particular sets of data—set appropriate file permissions
- Run a suite of programs
- Access particular tables in a database

Role-based access is already present to a certain degree in the Windows environment. Windows has a number of built-in groups that have predefined permissions and abilities. SQL Server and Exchange, for instance, both define roles to which users can be assigned. SQL Server controls the roles internally, whereas Exchange uses AD-based groups.

SQL Server requires authentication at the server (or instance) level. Authorization then determines to which roles a user has been assigned. There are built-in roles that are defined at the server level and others separately at the database level. Server-level roles include:

- dbCreator—Allows the creation of databases
- Security Administrator—Administers security for the server
- SysAdmin—Enables complete control of the server and databases

These are all administrative roles. At the database level, there are administrative roles and roles that enable access to the data, such as

- Db_datareader
- Db_datawriter

Putting a user into multiple roles delivers the required level of access. Using AD groups for authentication and authorization (via the SQL Server roles) will simplify administration.

All Windows server machines have built-in groups that provide access to perform a specific set of tasks. This functionality is duplicated in AD with a number of built-in groups that can be used as the basis of role-based access. This works for some administrative roles, but if a more fine-grained delegation model is required, it will be more efficient to adopt a toolset that can control the delegation via groups.

Role-based access for data access is not available in Windows. The Authorization Manger console allows administrators to control access to functionality within applications using roles. The roles must be created in the application by the developers.

Alternatively, role-based access must be simulated by defining the required access and designing a group structure to meet that need. The user accounts can be put in the groups, and the groups given the appropriate permissions. Ideally, the group membership should be established as a user account is created. This is a perfect example of the need to automate user provisioning to ensure the required steps are performed in a consistent and repeatable manner. The automation tool should also be able to cope with users being transferred between roles.

Within the Windows environment, we have the ability to use groups, or built-in roles, to control authorization. This leads to the ideal situation of a single sign-on (SSO) environment. One set of credentials is entered and the user gains access to all of their data and applications. This is fine for Windows-based systems but what about non-Windows systems?

Heterogeneous Environments

Authentication in a heterogeneous environment has already been examined. Authorization also needs to be considered. This can be required in a number of directions (consider a Unix system for discussion, though it could be another system):

- Unix-based account requires access to resources on the Unix system
- Unix-based account requires access to resources on Windows machines
- AD-based account requires access to resources on the Unix machine

These requirements will become very complicated to administer when there are multiple non-Windows machines that all have their own authorization mechanisms. A better solution is to consolidate the authorization information in a single source.

In addition, some environments do not natively provide the ability to delegate access based on groups, roles, or policy. Consequently, without additional tools, administrative access is an all-or-nothing proposition.

Consolidation of Authorization Systems

AD has already been considered for use as the authentication mechanism for the enterprise. If this concept is extended to authorization, the organization's investment in AD is leveraged to maximum advantage.

The tools that enable authentication to span non-Windows systems also enable authorization using AD groups. This approach has a number of benefits:

- Simplified administration with associated cost savings
- Single view of the enterprise
- Enhanced reporting for compliance

There are a number of ways to establish AD as the authorization data store within the enterprise. This should be done using an established tool rather than attempting to perform this task manually. The tool can also be extended to control access to certain applications as well.

Utilizing AD in this way has additional benefits in terms of the ability to extend system management beyond Windows. A prime example is the ability to create and apply Group Policy Objects (GPOs) onto non-Windows machines.

Summary

Identity and access management in the enterprise consists of administering authentication and authorization across multiple platforms. As we have explored, there are a significant number of challenges for the administrator to overcome, and many ways for the administrator to do so.