Realtime publishers

Protecting Critical Data by Managing the Active Directory Identity Lifecycle

Darren Mar-Elia

sponsored by

NetIQ™

Copyright Statement

Chapter 3: Securing Active Directory

So far, we've discussed the increasing importance of Active Directory (AD) within the identity management life cycle (Chapter 1) for most organizations, and we've talked about some of the finer points of that life cycle (Chapter 2). This chapter is dedicated to the point that, if AD is going to be a key part of your identity strategy, you have to protect it. You have to ensure that the data within AD is sacrosanct and that only those with a business reason to access AD information are granted that access.

This chapter provides a practical guide to protecting your AD-based identity data. It is not part of the life cycle I spoke about in earlier chapters, but it is an important part of ensuring that any identity system you implement that leverages AD is protected such that it is able to do its job of authenticating and authorizing the right people to the right resources. All the great identity provisioning processes in the world won't help you if your AD is a free-for-all that anyone can fiddle with to their heart's content. This chapter will dive into the AD security model and provide techniques and best practices for securing the data that resides in AD.

The Challenges of Securing AD

We wouldn't have to dedicate a whole chapter to managing AD's security model if the model was straightforward and simple. But the nature of a hierarchical directory service that serves many purposes (for example, application directory, authentication directory, desktop management directory, etc.) is such that the security model can be a handful. More importantly, if you don't take a proactive approach to managing the security of your AD, it can quickly get out of control. Take, for example, the simple task of delegating management of user accounts in AD. Because of the granular nature of the AD security model, managing user accounts, while seemingly a simple task, could break down into a dizzying array of permissions that must be delegated:

- Permission to create user objects
- Permission to delete user objects
- Permission to move user objects
- Permissions on properties on the user object (may break down into sensitive, such as department, manager, group memberships, etc.) versus non-sensitive (telephone number, office address properties)
- Permissions to reset the user's password or unlock their account
- Permissions to control who can change a user's permissions

This list is not comprehensive, but it underscores the possible complexity of managing delegation on just this one task. Take into consideration that each of these tasks (or at least groups of them) might be delegated to sub-groups of administrators, and that these permission sets might vary based on what OU a set of users are in. Throw into the mix that permissions can be inherited from parent objects in AD to their children (for example, from the Marketing OU to the Users OU under Marketing), and things can really get gummed up if you're not careful.

Not only is the complexity of AD's security model challenging (more on this later), it requires discipline to establish a good delegation model and keep it that way over time, as the one-off requests and unusual business needs drive you to make compromises in that model. At the risk of being overly dramatic, "That way madness lies." Over the years, I've seen so many AD delegation plans get mired in over-complexity and intricacy so as to be completely unusable over time. If our ultimate goal is to protect the data in AD that is critical to your organization's authentication and authorization mechanisms, then it's important to get and keep a handle on AD security. We'll talk about how to do this later in this chapter, but now let's dive into more detail about how AD's security model works.

Understanding the AD Security Model

Fundamentally, understanding the AD security model is about understanding how AD is structured. AD, not unlike a relational database, contains a schema that defines the available classes of objects and their associated attributes. A user object in AD is an instantiation of the schema class "user." That user object, as per the schema, contains a set of attributes such as first name, last name, department, manager, phone number, etc. In addition, each object in AD has a security descriptor associated with it. This security descriptor defines the permissions on that object (see Figure 3.1), which shows an example of a user object's permission set, or Access Control List (ACL), as viewed from Active Directory Users & Computers.

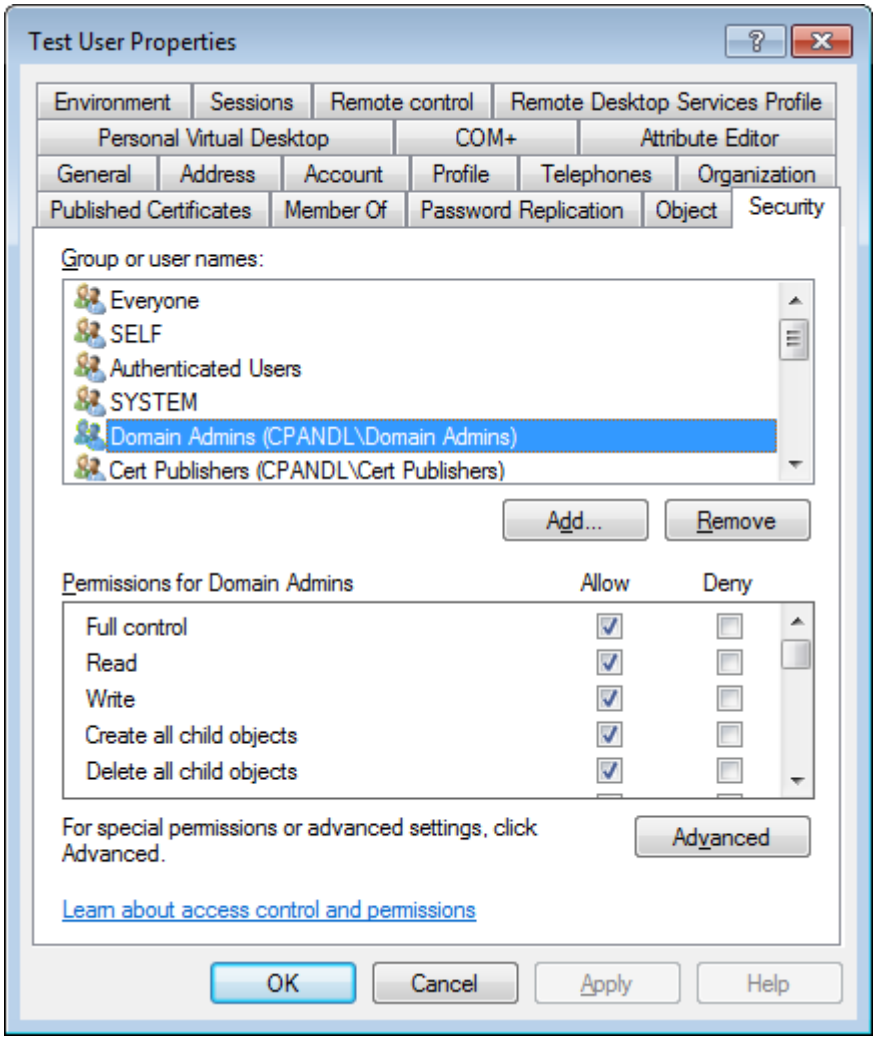

Figure 3.1: Viewing the ACL of a user object.

The ACL is composed of a set of security principals (usually users or groups) that have rights over that object, and the rights, or permissions associated with that security principal. In the screenshot that Figure 3.1 shows, you see that the Domain Admins group has Full Control (that is, all possible permissions) over the user object called "Test User." In addition, a particular permission can be either an "Allow" or "Deny" permission. The default, Allow, grants that permission to the security principal. But if Deny is selected, then that permission is explicitly denied to that security principal. In fact, if an object inherits permissions from its parent (for example, an OU) and there are clashing allow and deny Access Control Entries (ACEs) for a given permission, then typically the Deny will win (but not always—there are additional complexities there that we won't go into at the moment).

Standard vs. Extended Rights

As if things weren't already complicated, not every object class in AD has the same set of permissions associated with it. This feature is great, in fact, because it means that permissions can be tailored to the type of object involved. For example, having a "trigger replication" permission associated with an AD naming context object allows you to delegate who can force replication between two domain controllers. But trigger replication has no relevance to a user object. In fact, every object has a set of "standard rights" associated with it. These include familiar ones such as:

- Read
- Write
- List
- Create
- Delete
- Read and Write Properties

Beyond the standard rights, a schema class can have extended rights associated with it, like my example of the replication permission associated with the naming context class of object. Another more familiar example of extended rights is the permissions found on a computer object. You'll note that a computer has permissions such as "Read and Write host name attributes," as Figure 3.2 shows, which are specific to the computer class of object.

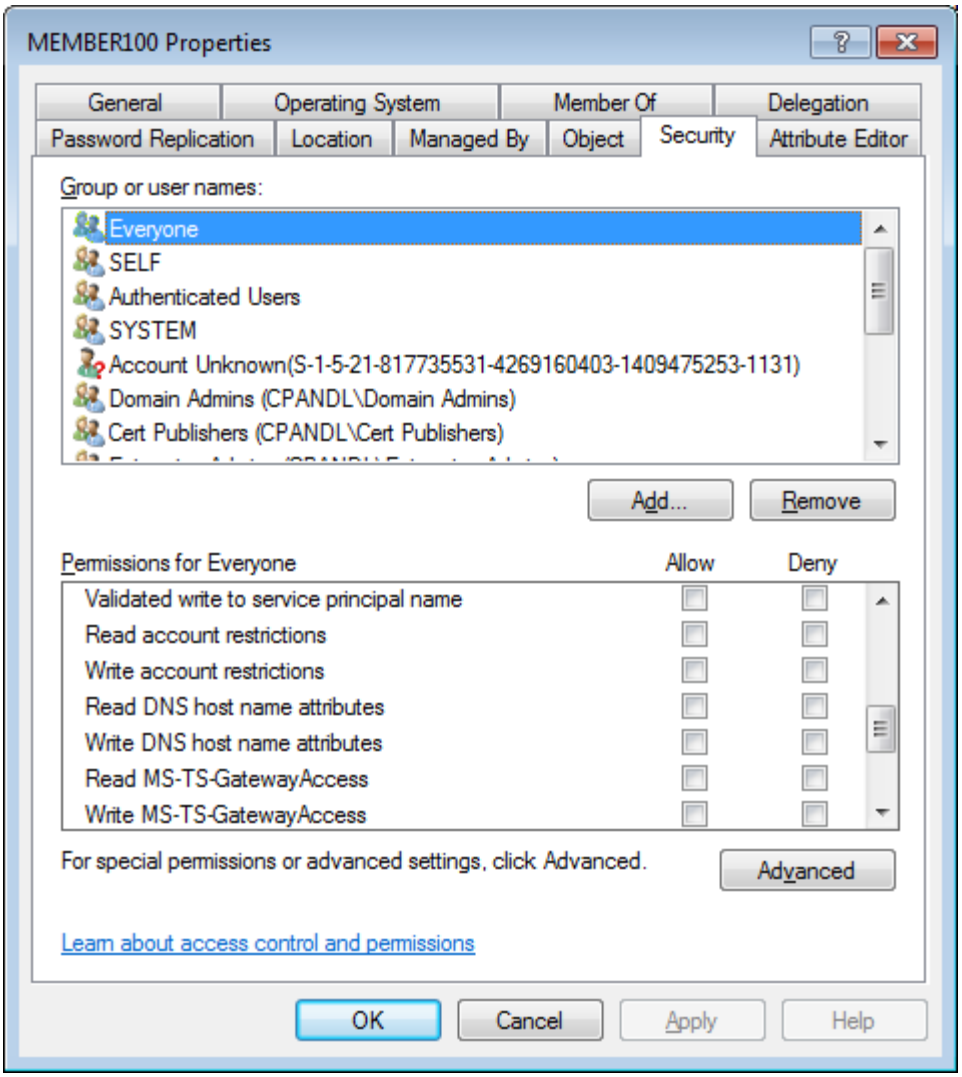

Figure 3.2: Viewing extended rights on a computer object.

This extensibility within the AD security model provides an opportunity to create a very rich and granular delegation of tasks for your administrators and users. If you extend the AD schema with a new class of object, it can have its own set of extended rights that control delegation specific to that object type. It just means that you also have to be aware of the various differences across the object classes you want to delegate.

Understanding Security Inheritance

Another aspect that makes AD's security model challenging is this notion that permissions can inherit through the AD hierarchy. A permission set at the top of a domain can trickle all the way down through nested OUs to objects at the very lowest levels of the domain hierarchy.

Of course, you can control this inheritance, both from the top down as well as the bottom up. For example, let's say you're setting permissions for user objects within an OU hierarchy composed of a top-level "Marketing" OU and two sub-OUs called "Users-East" and "Users-West." You want to take advantage of inheritance to set permissions for all user objects at the Marketing OU level and have that trickle down to all user objects in both OUs. You can do so by creating the new ACE within the Marketing OU's ACL (using AD Users & Computers as an example), and then, after setting the permissions, having it apply to all "Descendant User Objects," as Figure 3.3 shows.

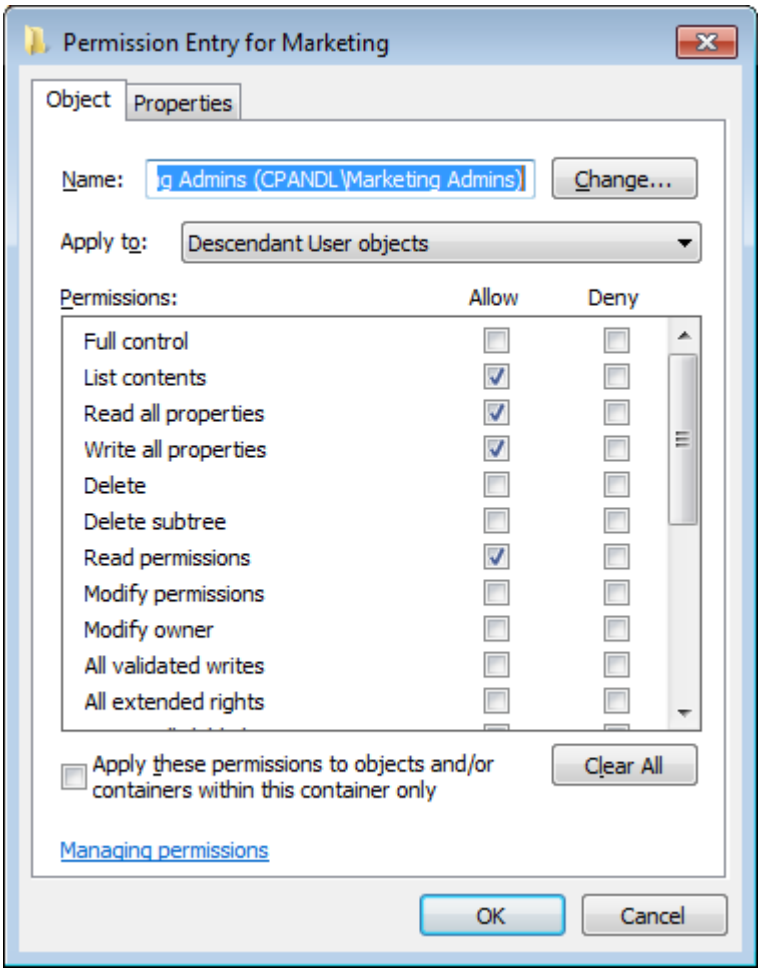

Figure 3.3: Setting a permission to apply to all user objects.

Likewise, as an administrator of, say, the Users-East OU, I might decide that the permissions that have been placed upon my user objects at the higher level do not apply to me. If I have sufficient permissions, I can essentially "turn-off" the inheritance that comes down to me from the Marketing OU. I do so by simply clearing the check box within the Advanced section of the ACL editor in AD Users & Computers that breaks that inheritance chain (see Figure 3.4).

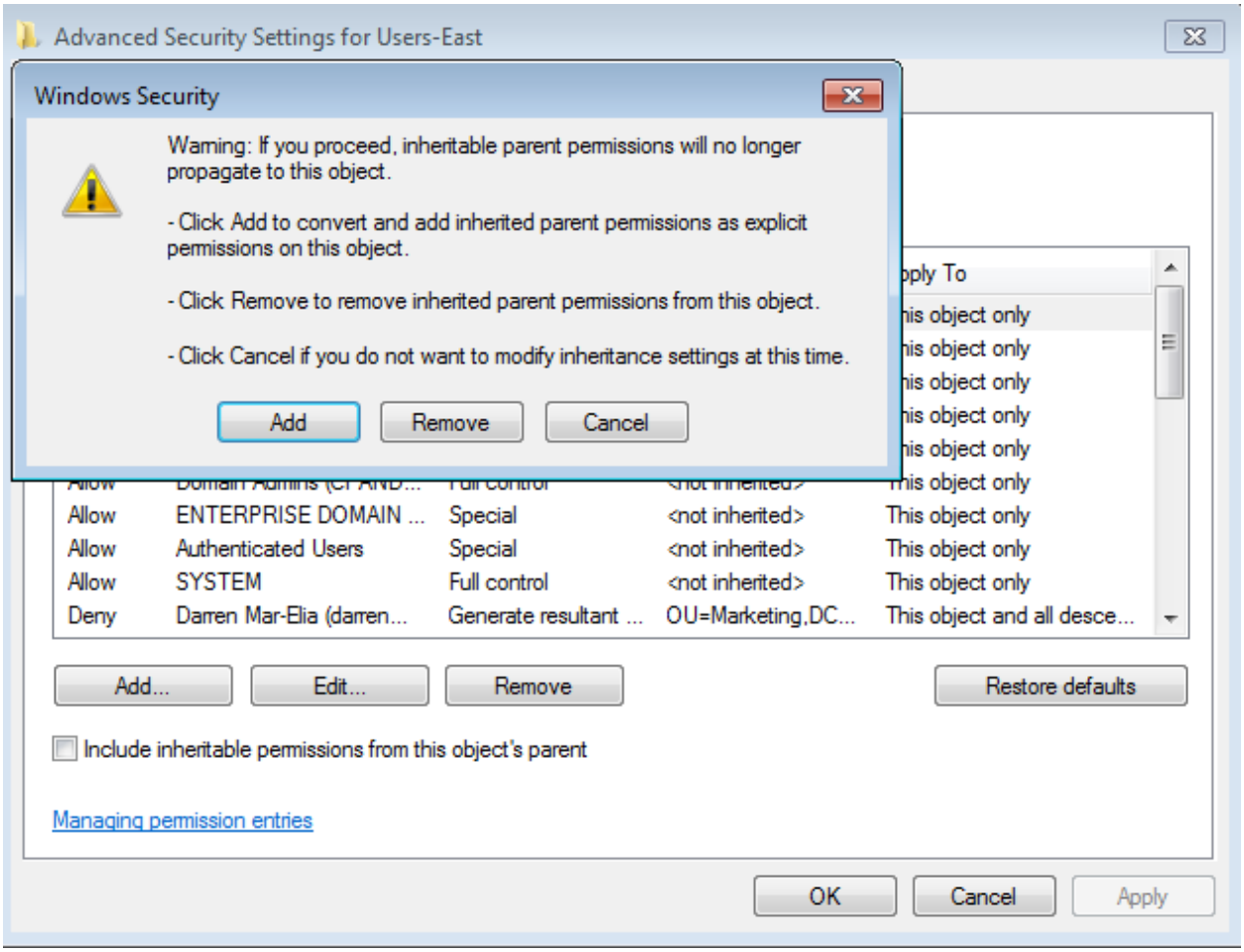

Figure 3.4: Breaking inheritance of permissions on a child OU.

Note

Not anyone can do this; I have to have sufficient privileges on my Users-East OU to allow me to break this inheritance—not something that I would normally be granted as an OU administrator, but it illustrates the possible complexities of AD inheritance model (or lack thereof in some cases).

What Is Delegation?

Delegation, in the context of AD, is the process by which you grant permissions to AD objects to allow users and groups to perform specific tasks against AD objects. Delegation implies (at least to me) some kind of orderly plan for giving the right users the right permissions on the right AD objects, across your AD structure. That's not always the way it ends up of course, but that is the goal. An example of delegation is creating a group called Help Desk Admins that all of the folks that man your Help desk belong to, and then granting that group the ability to reset user passwords on all user objects within your AD domain. This gives these users the ability to handle one of the primary tasks of their job (assuming you don't have a self-service password reset solution in place, of course!). Another common delegation example is granting desktop administrators the ability to join computers to the domain. This is a delegated permission on computer objects, typically applied to OUs where computer objects might reside.

Creating a delegation model for your AD is probably one of the most important planning tasks you can perform, especially when it involves delegating access to sensitive data held within AD (for example, group memberships granting access to sensitive resources). It doesn't matter whether you've just rolled out AD or are in the process of migrating your 10-year-old AD to a new domain structure. In either case, it's never too late to plan and create a delegation model that protects critical objects within AD and delegates access appropriately.

Some Common Delegations Related to Identity

There is a lot of "stuff" in AD that can and should be secured, but not all of it relates to your identity system. The following list highlights areas to start with in terms of protecting your identity data within AD:

- **User properties:** Attributes on users may or may not be sensitive and require delegation. But there are certain attributes, such as the user's Job Title, Department, and Manager that are often managed by the HR department (or, if AD is integrated into an HR system, those attributes might be managed through that system). In that case, you would want to prevent all users from being able to modify these attributes on their own.
- **Group memberships:** Groups can be used to control access to a variety of resources, from public file servers to sensitive database data. As a result, controlling who has permissions to change group memberships is probably one of the first steps you should take in your AD delegation tasks. This translates into who is allowed to write to the Members attribute on AD group objects. A user that has this permission can modify group memberships and a user that doesn't have this permission can't.

- **OU moves**: Although moving objects such as users between OUs may seem fairly benign from an identity management perspective, such a move can often have downstream effects. Some organizations have automated processes associated with OU membership that could change things, such as a user's group memberships or what Group Policy settings they process. These changes could, in turn, inadvertently grant the user access to resources that were unintended. As a result, OU moves of user objects should be tightly controlled.

In-the-Box Delegation Tools

There is some help available in the box with respect to delegation. This comes in the form of the AD Users & Computers Delegation of Control Wizard. The DoC Wizard is available whenever you right-click a container object (for example, domain or OU) within AD Users & Computers. It essentially templatizes a set of standard tasks that you might want to delegate within AD, as Figure 3.5 shows.

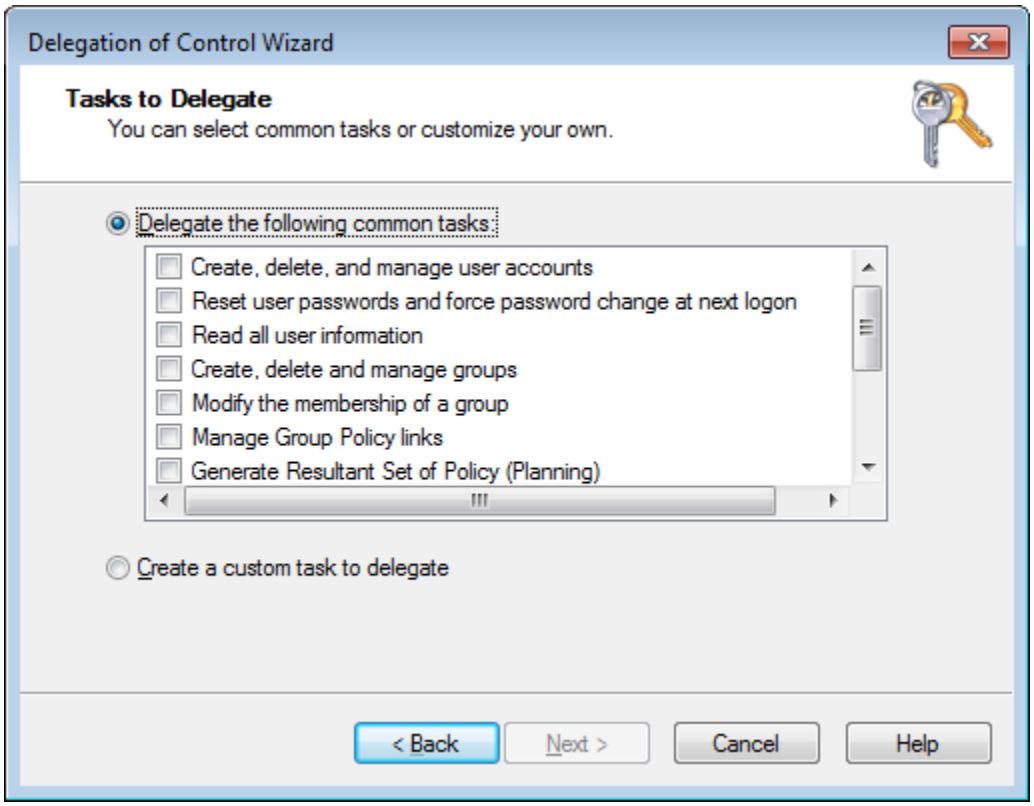

Figure 3.5: Using the Delegation of Control Wizard.

It also allows you to create custom delegation tasks by essentially picking object classes and choosing what rights you need on those classes. The result of the DoC wizard is that it stamps a set of permissions on the container you're focused on, with the permissions that you've requested

The thing I like about the DoC Wizard is that it makes simple delegation tasks easy. The things I don't like about the DoC Wizard are more numerous:

- It only supports a small set of delegation tasks (though you can extend it).
- It is a moment-in-time delegation. In other words, no state of the delegation you just performed is kept—the permissions are simply stamped on the AD objects you're focused on, and you can't modify that delegation through the wizard after the fact—you would need to go in and manually edit the permissions on the ACL directly.
- The DoC Wizard gives you no birds-eye view of delegation across your entire AD. Because it's ultimately just stamping privileges, there is no way to "keep track" of what delegations you've done without looking at the ACLs of each AD object of interest.

Best Practices for AD Delegation

There are a number of best practices that have emerged over the years that are worth considering as you determine how to create your delegation model to protect your AD data. A common approach is to take a sort of "role-based" approach to delegating tasks within AD. This involves listing out the tasks that you expect people to perform within AD (for example, user administration, joining computers to the domain, modifying group memberships). This list should be fairly long, as you'll want to really flesh out all those things you expect people to do against your AD, especially when it comes to modifying sensitive objects.

The next step in the process is identifying those groups of users that need to perform each task or role. By performing this mapping between groups that need access to AD and those types of access that you want to support, you are effectively creating a role-based delegation model that can be made real through permissions within AD. Table 3.1 gives an example of what this might look like.

AD Group Name	Tasks Permitted
Help Desk Admins	Manage user accounts, reset user passwords, manage non-sensitive user data
Desktop Admins	Join computers to domain, move computer accounts, create OUs, link GPOs
Server Admins	Join computers to domain, move computer accounts, create OUs, create/modify GPOs, link GPOs, modify computer group memberships
HR Admins	Modify sensitive user data
Resource Admins	Modify user group memberships

Table 3.1: Viewing a proposed role-based delegation model.

The examples in Table 3.1 are not meant to be comprehensive, but instead give you a sense of how you can design an AD delegation model that breaks security of AD into tasks or roles and then assigns those tasks to user groups that perform those tasks. This approach can greatly simplify management of AD security and take it from reactive granting of permissions to proactive managing of access to AD data.

> **Note**
>
> There are several third-party products on the market that greatly aid in managing AD security by allowing you to persist the creation of roles or tasks as objects (either in AD or in an external database). You need only modify the role definition, and permissions throughout AD are updated to reflect those changes. So, using Table 3.1 as an example, if you wanted to grant the Desktop Admins the ability to "modify user group memberships," you need only change the role definition for "Desktop Admins" and these third-party products will propagate the underlying permission changes to all relevant AD objects where the role is currently applied. This makes consistently applying permissions (and removing or adding permissions) much simpler over time.

Considerations for Removing Read Access

Out of the box, AD grants the Authenticated Users group, which includes all users defined within a domain, the ability to at least read most all data within AD. This is usually fine if you're not storing sensitive information within AD. But if you are, you may need to consider removing read access for some user groups to some areas of your AD domain. This usually amounts to removing the "Authenticated Users" Read ACE from the ACL on objects within your AD domain. Another option, less desirable, might be to set a Deny ACE, as I mentioned earlier, for a particular group on a particular set of objects. For example, in Figure 3.6, I'm setting a deny read ACE for the Unprivileged Users group on the HR OU.

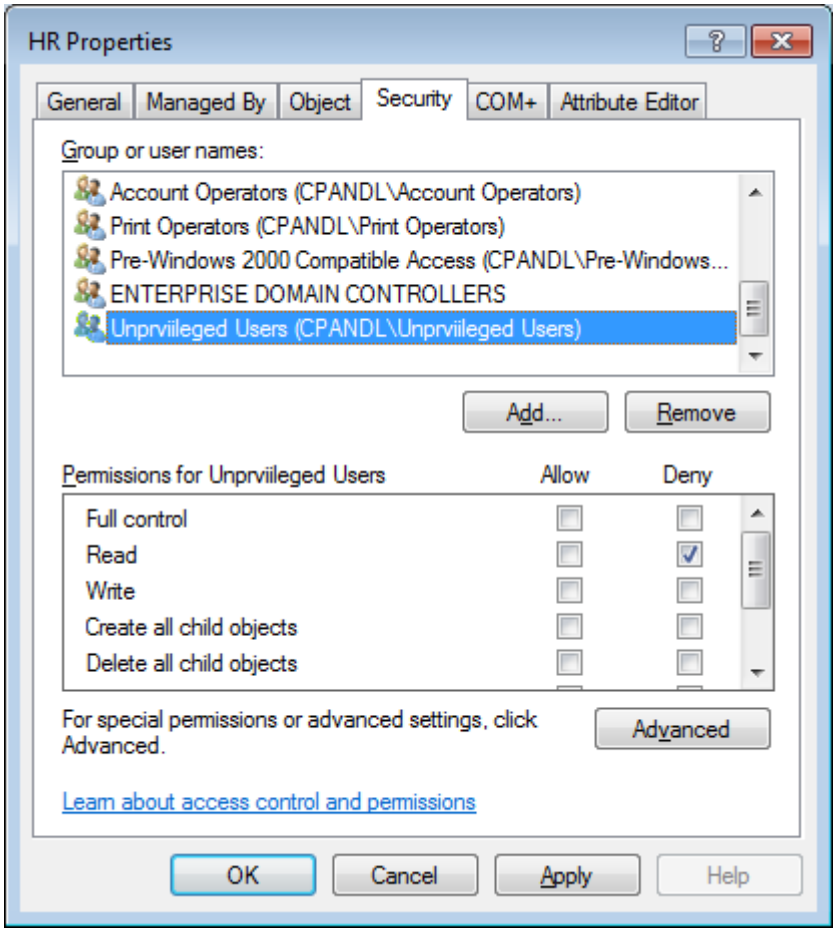

Figure 3.6: Setting a deny read ACE on an OU.

If you get to that point, it's important to proceed cautiously. Some tools and applications may not behave well when they are searching AD and stumble upon objects that they cannot read. Indeed, some of Microsoft's own tools don't behave well in these circumstances. The bottom line is that when you start denying read access to objects within AD, make sure you test all directory-enabled applications and tools that access AD to ensure that the way you are protecting your objects does not break their function.

Least Privilege and AD Delegation

I introduced the concept of least privilege in earlier chapters as a way of assigning access to resources such that only those rights that are needed for a user to do their job are granted. This principle is at least as important when protecting AD objects and their attributes as it is when protecting sensitive file shares or databases. Granting many administrative groups wide access to be able to modify AD objects is generally not going to lead to anything good.

Using the approach I outlined in Table 3.1, you can ensure that everyone from regular users to administrators only have the permissions they need to do their jobs. I'll talk more about this in the next section, but a perfect example of least privilege with respect to AD delegation is the use (or misuse) of the Domain Admins group. Often, administrators will grant access to this group because "someone needs to perform an AD change" and it's easier to grant this access than to figure out what exact rights are needed to perform the particular change. This is the "path of least resistance" approach and will not serve you well if you are relying on AD as a critical identity store.

Instead, take each new request to perform a particular task in AD to add to your role-based delegation. Microsoft has, over the years, released great guidance on what rights are required to perform which changes to AD. These are outlined really nicely in the following presentation: http://technet.microsoft.com/en-us/bb899735, which was written for Server 2003 but still applies to a large degree in newer OS versions.

AD Delegation vs. Resource Delegation

In this chapter, I've talked primarily about delegation and security access control of AD objects. I contrast this with discussions I've had in earlier chapters around using AD (and primarily security groups) to control access to resources such as file servers, printers, applications, and databases. It must be stated that delegating access to AD objects does not automatically translate into granting access to the underlying resources represented by those objects. For example, an administrator may be able to modify properties on a computer object in AD that represents a file server, but that modify ability grants no underlying access to that physical server itself. That is, even though the administrator can change the AD server object's OU, modify its group membership, or even delete the computer account from the domain, he/she may not even be able to log into the server—that is controlled by resource delegation.

For example, access to a server's resources is typically granted by being a member of a group that has permissions on that server. Regardless of whether we're talking about administrators being able to log on and change the configuration of the server or users being able to access files on the server's share, those are all separate sets of entitlements as compared to the permissions on the AD object that represents the server. In the case of granting access to that underlying resource, there are a number of mechanisms that can be used, which we'll discuss here.

The Role of Group Policy in Delegation

Group Policy, the configuration management feature in Windows that is closely tied to AD, is an excellent mechanism for handling delegation of server and desktop resources. Unfortunately, Group Policy is not a mechanism for handling delegation and access control of AD objects themselves. Group Policy's primary target is Windows desktops and servers and their security configuration—basically resource delegation. Group Policy can and often is the primary mechanism that shops use to make the connection between AD groups and what those groups can do on Windows resources. As an example, Figure 3.7 shows a Windows Group Policy Object (GPO) that grants the Server Admins group the ability to remote desktop into an OU containing servers, by virtue of making them a member of those servers' local Remote Desktop Users group.

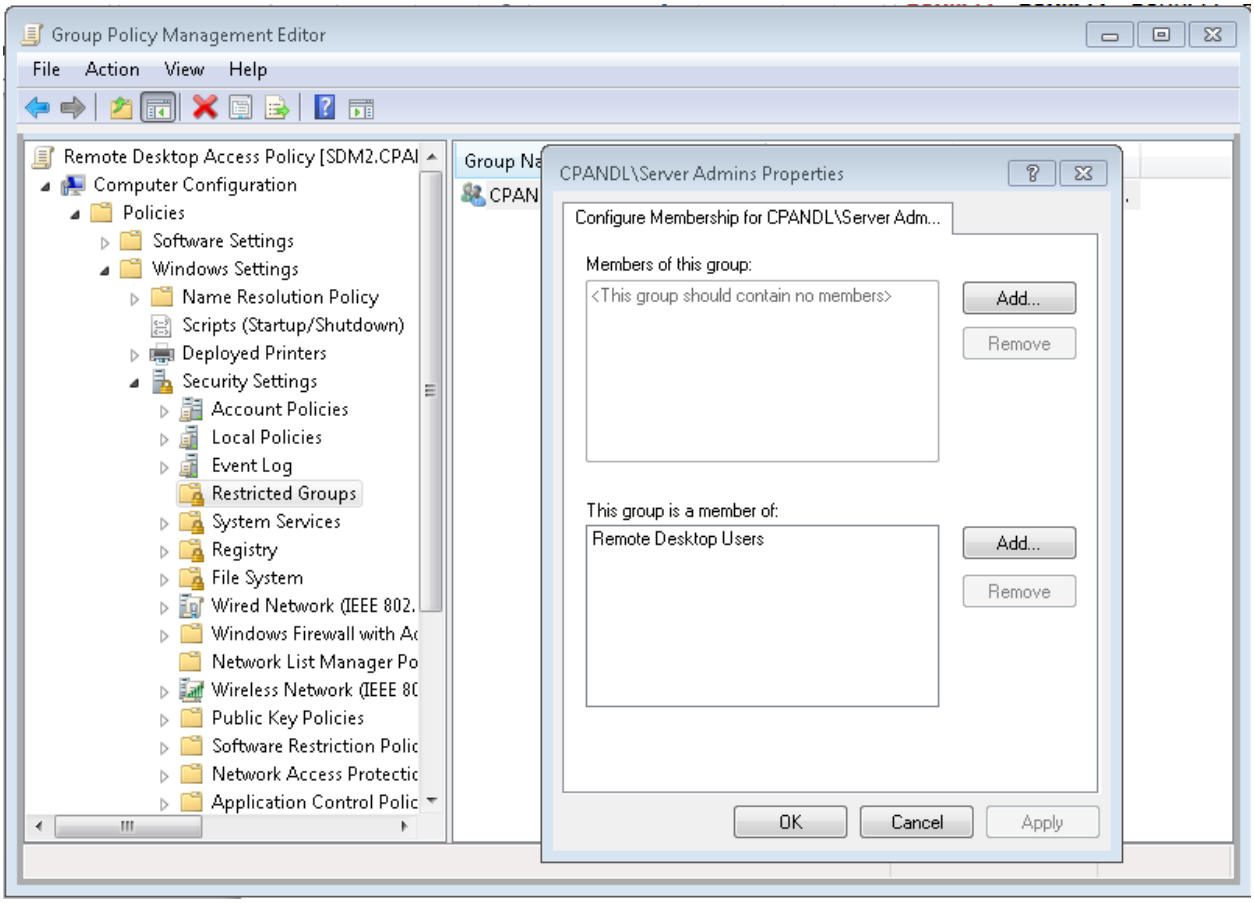

Figure 3.7: Using Group Policy to set resource delegation.

Group Policy is also really good at "hardening" servers, or ensuring that their overall security configuration is locked down to prevent access from users that should not be able to access them. Microsoft provides a good starting point to this hardening in the form of the Security Compliance Manager tool (SCM), available at http://www.microsoft.com/download/en/details.aspx?displayLang=en&id=16776.

The SCM tool lets you view and edit Microsoft baseline security configurations for a variety of hardening scenarios and for a variety of Microsoft products (for example, Windows desktops, servers, Microsoft Office) and lets you deploy these configurations as GPOs in your live environment.

Delegating GPOs

Another thing to note about Group Policy: it's just as important to protect access to your GPOs as it is to protect your AD. GPOs can grant elevated access to server and desktop resources, so letting administrators tamper with them is a big problem. You can delegate two main areas within Group Policy. You can delegate who can create and edit GPOs and you can delegate who can link existing GPOs. The latter is actually an AD delegation task because linking of GPOs is controlled on AD containers such as the domain or OUs by controlling who can write to the gpLink attribute on that domain or OU (see Figure 3.8).

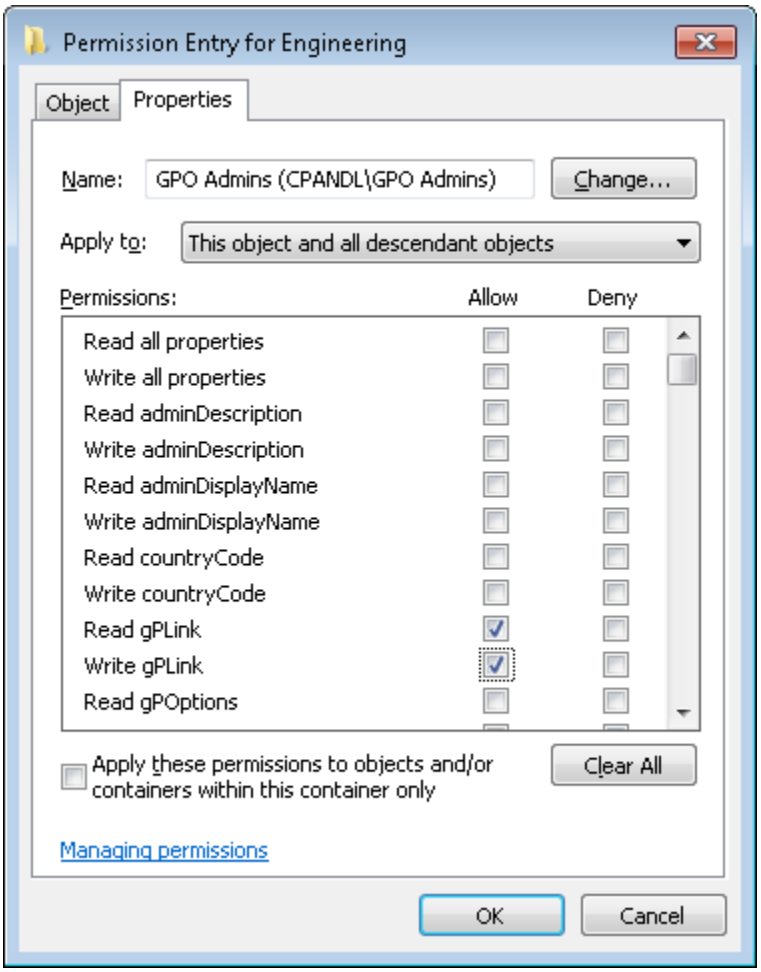

Figure 3.8: Controlling who can link GPOs to an OU.

The former, controlling who can create and edit GPOs, is controlled through setting GPO permissions—best done using the GPMC console that Microsoft provides. If you select the Group Policy Objects container, and choose the Delegation tab, you can control who can edit a given GPO. If you right-click any given GPO, you can set Delegation on that individual GPO.

Controlling Access to Domain Controllers—Protecting AD

You might be tempted to look at domain controllers the way you look at any other Windows server resource within your environment. However, AD domain controllers are special. Because domain controllers hold the AD database, and each domain controller is capable of reading and writing changes to the AD database, they should be secured differently. By default, anyone who is in the Domain Admins group can log into a domain controller. In addition, such users have full administrative access of servers in the domain and thus over domain controllers.

Thus, if an administrator is logged into a domain controller and is an administrator on that server, there isn't much they cannot do to that domain controller and therefore AD, which you might try to protect against using other mechanisms, such as delegation of AD objects. This is especially true if the admin user also has physical access. Given this fact, it is super-critical that only a small handful of users have the ability to log into a domain controller—and this includes not only folks who might need to log onto a domain controller to manage AD but also those who are performing OS maintenance tasks unrelated to AD. In both cases, I recommend keeping the list to a very small number of administrators who have knowledge about AD.

The bottom line here is that AD servers—domain controllers—should be treated different than other servers. And until Microsoft provides more granular delegation over who can do what when logged into a domain controller, you will want to limit the number of administrators who can access these special servers directly.

Summary

In this chapter, we talked about the importance of protecting the data in AD itself. We went over the security model that AD employs, which can be complex, talked about a role-based approach to AD delegation, and covered examples of the kinds of tasks, such as group membership changes, that are critical to your identity management systems. We also talked about the differences between AD delegation and resource delegation—discussing the role of Group Policy within that story. And finally, we talked about the importance of protecting your AD domain controllers, as physical access to them for large numbers of administrators puts all of the work you've done around delegation at risk.

Download Additional Books from Realtime Nexus!

Realtime Nexus—The Digital Library provides world-class expert resources that IT professionals depend on to learn about the newest technologies. If you found this book to be informative, we encourage you to download more of our industry-leading technology books and video guides at Realtime Nexus. Please visit http://nexus.realtimepublishers.com.