Realtime publishers

Protecting Critical Data by Managing the Active Directory Identity Lifecycle

Darren Mar-Elia

sponsored by

NetIQ™

Copyright Statement

Chapter 2: Managing the Identity Lifecycle

Chapter 1 discussed how Active Directory (AD) has become a key player in many organizations' identity landscape—controlling access to not only Windows desktops but also an increasing number of systems, applications, and sensitive corporate data. The chapter also talked about the importance of managing the identity lifecycle, as defined by creating, updating, auditing, and removing identities within both AD and related identity systems. This chapter digs into that identity lifecycle, identifying downsides of not getting a handle on your AD and drilling into each of the four phases of the identity lifecycle.

The Challenges and Importance of Managing the Identity Lifecycle

Managing identity is ultimately about managing access to your corporate resources. Users authenticate to resources with their identity, then use the properties of that identity (for example, group membership) to get authorized to resources. In a typical midsize-to-large organization, you might find the following sources of identity

- AD
- Other directory services
- HR systems
- Databases
- Custom LOB (LOB) applications
- Third-party software-as-a-service (SaaS) Web applications
- Local system accounts on Windows, Linux, and/or Unix

All of these identity stores present challenges. Each one requires its own provisioning event (and de-provisioning as well) into what are usually disparate data stores: directories, databases, flat files, or in some cases, proprietary formats. Each one has its own set of authorization mechanisms and unique ways of granting access. AD Windows uses security groups, databases like Oracle use custom roles built-in to the database, and other LOB applications use yet different mechanisms. More recently, SaaS applications are becoming more prevalent, which means you're now required to provision access to both internal and external applications.

It's also important to not blur the lines between authentication and authorization. Some products—I'll use my previous example of Oracle databases—are able to integrate into AD for authentication (for example, through Kerberos) but still keep their own authorization mechanisms that don't directly leverage AD ones such as security groups. This kind of mixed integration may or may not help your provisioning processes.

This mix of identity stores increases the complexity around ensuring that the right users are provisioned into your environment, and de-provisioned when the time comes. But it also increases the importance of having lifecycle management in place because it becomes a lot easier to "lose track" of identities if they are not all knitted together using a common framework. I've seen many an organization that had far more identities stored in a system than they had users. When asked why that was, the response was usually something like, "Oh, those are old users who are no longer here." I can remember personally being at a job for a number of years, then going back to do some work for them 5 years later, only to find 10 year-old Unix accounts that I had the first time I was still there floating around their systems. That kind of poor identity management is a recipe for unauthorized access, failed audits, or both.

Reducing the Problem—Reducing the Number of Identity Stores

If you're in one of those organizations that has a wide variety of identity stores, you already know that you have your work cut out for you in terms of managing all of those identities in a cohesive framework. But there's another point to consider here. One option is to work to reduce the number of identity stores by finding common identity systems that you can use to collapse other standalone systems into. For example, as I mentioned in Chapter 1, AD is increasingly becoming that common identity system for more systems and applications.

There are third-party and built-in products that let you use AD as the primary authentication mechanism for Linux/Unix and Mac. These solutions typically leverage the Pluggable Authentication Module (PAM) architecture within these operating systems (OSs) to allow AD to act as a Kerberos authentication realm for these systems, almost the same way that Windows systems do. In fact, with many of these mechanisms, you can "join" Linux/Unix or Mac computers to AD just as you would Windows desktops or servers. And instead of logging into Unix or Linux systems using a local account, you can now use an AD account to authenticate your users and ultimately authorize them to Unix resources using AD groups. Further, any applications running on those non-Windows boxes that leverage PAM to authenticate and authorize users to local accounts may now be able to leverage AD integration to support AD account authentication and authorization. Again, this situation will be application dependent, but it means you may get some AD integration "for free" once you integrate the base OS.

In addition, many third-party applications and application platforms support authentication and authorization using AD in some form, including packaged apps such as SAP and Java Web application servers from Oracle and IBM. Even Oracle databases support authentication and authorization integration into AD using a variety of integration methods from straight Kerberos to integration into Oracle's own LDAP directory service.

The Advantages of an Identity Management System

Regardless of the method, there are obvious advantages in trying to reduce the number of identity stores that you have to fold into your identity management lifecycle. If AD can be that "point of consolidation" for many of your business applications and systems, you can focus on provisioning and de-provisioning tasks into AD—and maybe a few other key identity stores that can't be collapsed (for example, an HR system). Thus, the task of de-provisioning a user from most of your internal and external systems and applications can become a simple matter of disabling a user account in AD and just a few other places. Figure 2.1 gives a pictorial example of what this might look like.

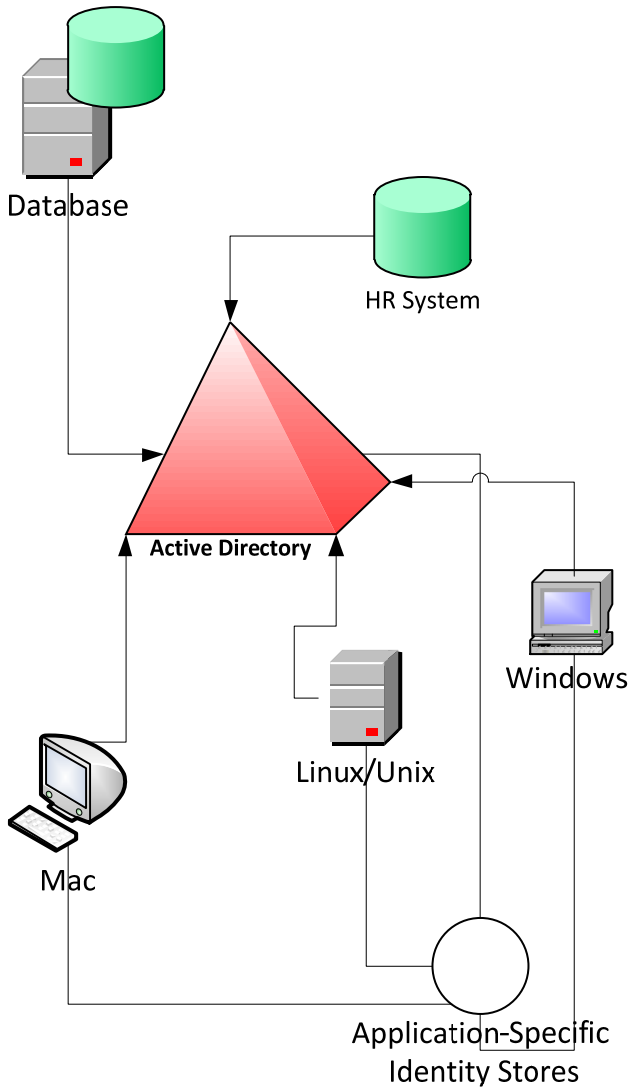

Figure 2.1: Viewing an identity system with multiple stores and clients.

Figure 2.1 shows a system that is centered on AD to provide authentication and authorization for Windows, Linux/Unix, Mac, and database systems—but also one that contains other application-specific identity stores that users of these systems must also authenticate to. In addition, the green cylinder represents a corporate HR system, which is typically the "system of record" for employees and contractors that enter an organization.

It's not uncommon for identity to "begin" within an organization within the HR system, which then pushes that identity out to other required identity stores, such as AD or application-specific identity stores. It usually is the job of a formal provisioning/de-provisioning system to perform these kinds of updates on the various connected systems—keeping identities in-sync and removing them from all connected systems when the user leaves the organization. Such solutions might have their own directory service where all this data from the connected systems is aggregated—often referred to as a meta-directory. Or the solution may simply keep track of the mappings between connected systems, mapping key fields in one system to another. For example, in Figure 2.2, the employee ID is mapped out of an HR system to corresponding key fields in the various identity stores within an organization.

	HR System	Active Directory	Application-Specific Identity Store	Meta-Directory
Key Field	Employee ID	employeeID attribute on user object	Employee ID field in user store	employeeID field on meta-directory user object

Figure 2.2: Viewing field mappings of connected identity stores.

The goal of such a mapping is to find a field that identifies a user uniquely within all identity stores. That ensures that John Smith within AD is the same John Smith that was just hired by HR and the same John Smith that has access to the application-specific application by virtue of the identities that it stores. And, when John Smith leaves the organization, his user id is disabled from the HR system, AD, and the application-specific store in one operation.

Building a Solid Foundation to Reduce Risk

So far, all of what we've talked about in this chapter has been around the challenges of managing reasonably complex identity systems and how you can take steps—like reducing the number of identity stores you have in place—to help reduce the challenges and complexity. But let's shift gears and talk about why it's actually important to have a system in place for managing the identity lifecycle—the real, on-the-ground reasons for building a system that keeps track of who exists in your organization and what they have access to.

Data Loss

Without a doubt, one of the scariest risks that most businesses that deal with private information—such as customer data—face is the inadvertent loss of that information. Any reasonably large organization today faces any number of possible avenues for data loss: from employees walking out with customer lists on USB keys to an executive getting their unencrypted laptop with sensitive financial information stolen from an airport lounge. Some of these scenarios are preventable with the proper tools and procedures in place, but by far one of the worst mistakes you can make is to lose data by virtue of someone having the wrong level of access or having access to systems that should have been removed long ago. These scenarios are bad because they are all preventable with a cohesive identity plan in place that focuses on ensuring good processes and good automation are in place every time a user enters, changes jobs, or leaves the organization. Data loss is especially bad in today's Internet world because a company's online reputation, and their ability to keep or lose their customer's data, can have a direct and immediate impact on their bottom line and the level of trust that they have with their customers.

How does having a good identity management plan in place help data loss? Simple—if you have good control over the users that are able to authenticate to your systems and good processes in place for granting access only to the data they need to do their jobs, the chances that the wrong data falls into the wrong hands is greatly reduced.

The most troubling part of this challenge is that the threat landscape is rapidly changing. A recent report on data breaches commissioned by Verizon[1] shows that, while external attacks against organizations are still a major vector for data loss, increasingly internal threats from both inadvertent mistakes due to poor system security as well as organized internal fraud efforts are a major concern. In fact, in a nod towards having good identity controls in place, a vast majority of internal threats (~85%) were perpetrated by "normal" users without privileged access—that is privileged administrators and users were a small percentage of the internal threat source. Thus, having good controls in place around access to data and systems can have a demonstrable impact on preventing data loss by internal users bent on malicious activities. These controls are only possible when you have a system that can provision and identify the right users with the right levels of access, such as when you have a provisioning system in place that identifies a user's authorization levels based on their business function.

[1] "Verizon 2011 Data Breach Investigations Report" at http://www.verizonbusiness.com/resources/reports/rp_data-breach-investigations-report-2011_en_xg.pdf

Note

Even the best provisioning system in the world can't prevent a user who is rightly authorized to sensitive data from abusing that data, but it can help ensure that only those users who need access to that data are allowed such access.

Regulatory Risks

Along with the risk of data loss comes the attendant risks for organizations that are subject to governmental regulations. Regulations such as the Sarbanes-Oxley Act, Health Insurance Portability and Accountability Act (HIPAA), Payment Card Industry (PCI), and others have varying degrees of explicit prescription when it comes to methods for protecting customer and/or non-public data. All of these regulations mandate that such data must be protected, with stiff fines and possible criminal ramifications for gross failure to protect such data. If your organization is subject to such regulations and you don't have a solid plan in place to address the variety of risks around data breaches, you are asking for trouble. In addition to other controls, putting a solid identity management plan in place can help you get better control over who is accessing your systems. And, going back to Chapter 1 and the identity lifecycle graphic, if you are able to provision, update, audit, and de-provision user accounts with a certainty that you are touching all critical systems, your ability to talk to and show auditors your compliance with regulations becomes much more straightforward.

Business Impacts

In addition to data loss and compliance risks, good identity management results in better system availability—and fewer business impacts. How? Identity management systems control access to not only business data and applications but also systems. Putting a "least privilege" plan for system authorization in place through your identity management system such that only those administrators have access to server resources for which they are responsible will go a long way towards preventing unwanted server outages. What you want to avoid at all costs is an administrator with privileges far greater than their role making a change to the wrong server at the wrong time and bringing down your main business system.

As an example, I've heard countless anecdotes of administrators with "Domain Admins" access of their AD—which is essentially unfettered access to read and write most objects in AD—accidentally deleting a critical application service account, or inadvertently moving objects from one organizational unit (OU) to the next, causing different Group Policies to apply to OUs and subsequently changing their behavior. Either of these examples might be enough to cause a major outage just because someone "punted" when it came time to provision a user account and granted that user far more rights than they needed (or were equipped to handle).

Group Policy is another area that is ripe for having a good system in place for controlling who can access and make changes. Group Policy changes can have a vast impact over an organization. I once helped someone who had made a policy change to security policies that essentially prevented ALL of their users from being able to log into their PCs. And that change was caused by a single click of a policy—a change that took less than a minute to make but that was done by someone who was just "testing" something and should not have been able to commit changes to production GPOs. These kinds of changes benefit from ensuring that the delegation model around Group Policy is tightly controlled, which usually involves ensuring that the right users are in the right AD groups that have the ability to modify GPOs—as Figure 2.3 shows.

User Provisioning vs. Resource Permissioning

The bottom line with all of these access issues is that having a good identity management system in place, with a standard process for provisioning and updating user accounts with their proper groups and other authorizations, helps ensure that the right users have access to the right resources. Keep in mind that the actual permissioning of resources to do the actual granting of resources is not covered as part of the user provisioning process, but it is an important prerequisite for being able to properly leverage your identity provisioning process.

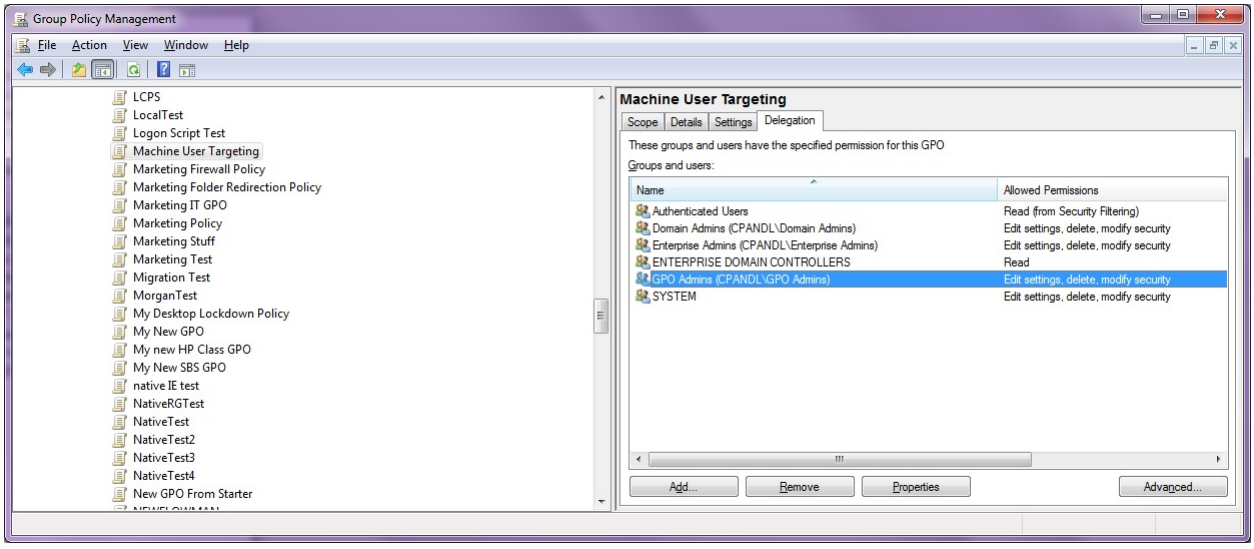

Figure 2.3: Controlling writes to GPOs using AD security groups.

Now that we've talked about the importance and challenges around building a solid identity management system and the risks of not doing so (or not doing so well), let's shift gears and start to put the pieces in place for creating a top-notch system for managing the identity lifecycle—starting with the provisioning and de-provisioning processes.

Automating Provisioning and De-Provisioning

The first step in the identity lifecycle chart from Chapter 1, which Figure 2.4 shows, is creation of identity-provisioning of accounts.

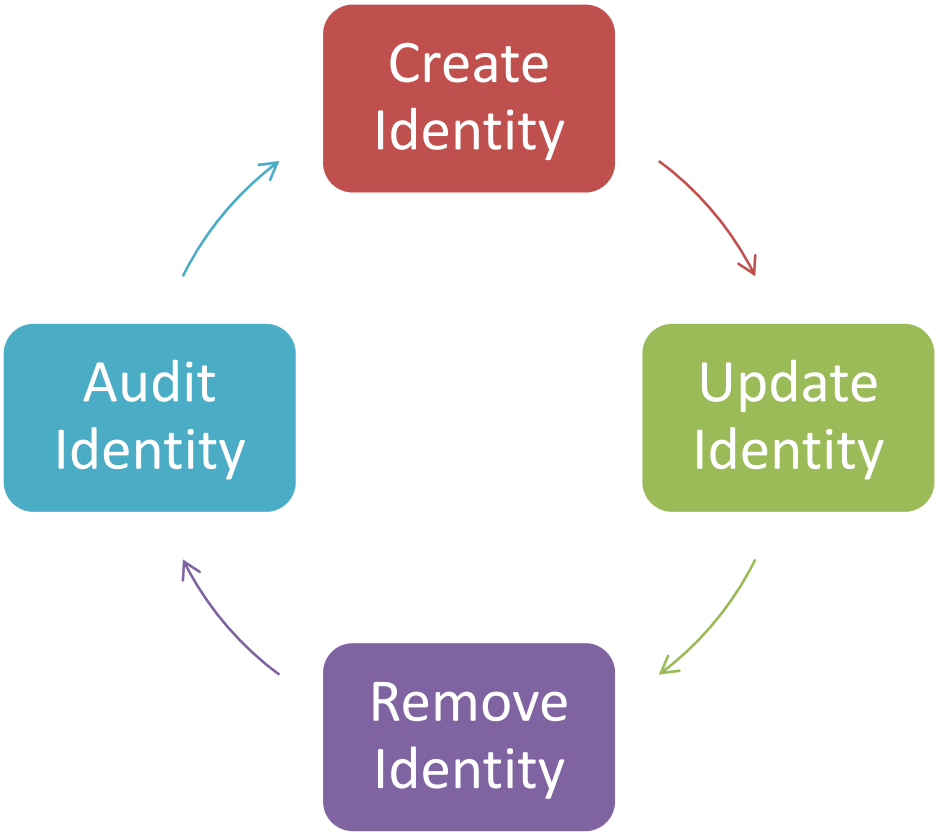

Figure 2.4: The identity lifecycle.

Creation of identities, in a perfect world, should be fully automated—or at least not require much in the way of user intervention. When a new employee is hired and a record is created in your HR system—or whatever the system-of-record is for your organization—that process should trigger a set of automated workflows that generate identities in all connected systems through your provisioning solution. The flowchart in Figure 2.5 shows an example of how a typical automated provisioning workflow might proceed.

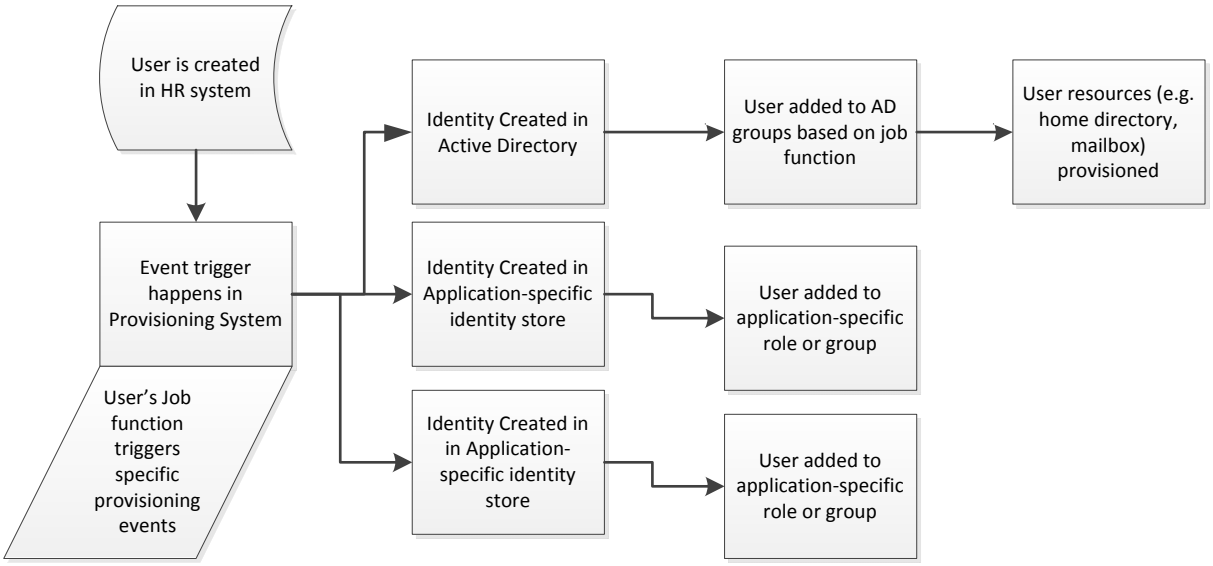

Figure 2.5: An example of an automated provisioning workflow.

In the workflow depicted in Figure 2.5, a new employee joins the company and gets a record created in the HR system, usually with some unique identifier—like an employee ID. The provisioning system detects that a new record was added to HR, which triggers a set of workflows in the connected identity systems, including AD and any other integrated identity stores. Ideally, you can define the user's job function within the provisioning system and doing so triggers a specific set of operations within the connected systems.

For example, if you are hiring a new salesperson, somehow that information is captured within the HR system or after it gets into the provisioning system. That information is forwarded along to the workflows that happen in each identity store, to ensure that, for example, the user is placed in all the correct security groups to allow access to sales-related data and potentially applications. Similarly, other connected identity stores—for example either an internal or SaaS-based CRM system, might also put the sales user into the appropriate role for accessing CRM data. Finally, any resource the user needs, such as being placed on the correct mail server or having a home directory set up on a file server, is performed out of the provisioning workflow.

All of these steps could very well happen without any human intervention, although approval-based workflow along the path could do things like verify that the sales user's manager really wants them to have access to the company price list on a particular server share that is governed by an AD group membership.

Tying Identity into External Systems

So far I've sort of glossed over the fact that my described identity management system starts all-new provisioning events with an event in an HR system. But let's look at that in more detail. Most HR systems are based on a relational database of some kind. Thus, it becomes relatively easy for a provisioning system to set up a standard SQL query that can detect new records and use that event to trigger the provisioning cycle. Another approach is to use an event from a service desk product (for example, BMC Remedy or Microsoft Service Desk Manager), such as a request for a new user, to trigger provisioning events. Whichever the source—HR system or service desk request—the key is that your provisioning system knows about all-new identity requests and can incorporate them into a standard process for creating user accounts in your identity stores.

Another common approach is tying your provisioning solution into some kind of meta-directory, such as Microsoft Forefront Identity Manager (FIM). Products like FIM contain a repository that synchronizes all identity stores and contains a central view of identity across the organization. Often, the meta-directory has ties to the HR system, AD, and other identity stores. When an event occurs in the HR system, the meta-directory synchronizes that new identity with all other stores that it knows about, using a set of workflows and rules that have been previously set up to inform it as to what groups or roles the user needs, where they belong in the AD hierarchy, and so on. In that scenario, customers might have additional provisioning systems that take over where the meta-directory leaves off—performing additional provisioning of resources for a particular application or systems or the meta-directory product itself may provide these functions.

In other scenarios I've seen, the meta-directory is an LDAP-based directory that contains the whole universe of identity data within an organization, and that meta-directory, while not being responsible for the provisioning events themselves, serves as the company's "white pages" for people information—the place where applications can go to look up information about employees, contractors, and so on. In this example, the meta-directory is not "authoritative" for any of the people data—that comes from the connected systems such as AD or the HR system or applications-specific repositories.

How you end up integrating your external data sources into the provisioning process is going to depend upon your own requirements. There is no right or wrong answer about how this is done, but I like to keep in mind a few guiding principles when deciding on an architectural approach for identity management and provisioning solutions:

- Be clear on which system is authoritative for which attributes. The last thing you want is multiple identity stores that each think they are authoritative for things like user name, password, email address, and so on. This will result in conflicts and inconsistent data, depending upon which identity stores are being queried.
- Try to keep synchronization of identity stores to a minimum. In some cases, it's required, but whenever you introduce synchronization of data, you introduce complexity and points of failure. The fewer authoritative sources you have for identity, the better
- For LDAP-enabled applications, try to have a single source of truth—be it a meta-directory or AD. That way, you can build standard authentication and authorization practices and guidelines for your developers and ensure that the LDAP "service" is reliable and performs well. If you have multiple LDAP repositories that developers build against, it is much harder to guarantee that they will get the results they expect and the quality of service they need.

User Entitlement to Resources

One of the key steps of the provisioning process that I mentioned in Figure 2.5 is the entitling of users to resources. Within the realm of AD, this typically means adding users to AD security groups. In non-AD based systems, it could also mean adding users to database roles or application-specific roles or groups. I'll talk more about group management in a bit, but the bottom line here is that a good provisioning process incorporates entitlement to resources as a function of the user's job. Of course, this presumes that you have set up your resources and groups such that they map reasonably closely to your users' job functions. For instance, from my earlier example of the sales user coming into an organization, it is likely that there will be price lists, customer data, and documents related to the sales process that you want to ensure this new user has access to. But that access requires that the file share (see Figure 2.6) or Microsoft SharePoint site that contains that data has been permissioned with a security group (for example, Sales Users) to control access to that data. Or at least, it probably should be if that sales data is not destined for anyone in the organization to access.

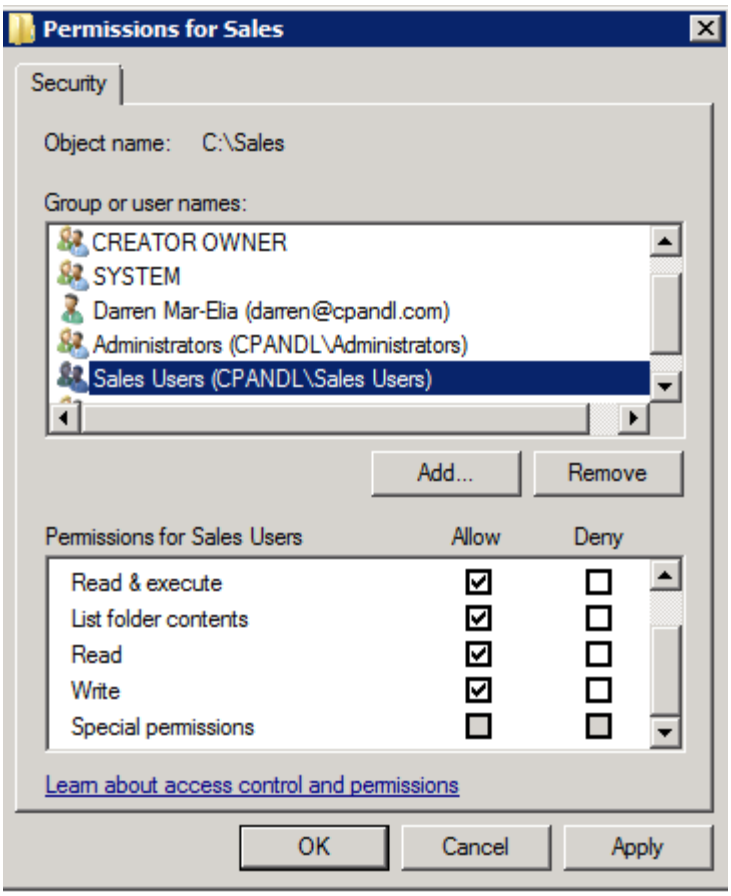

Figure 2.6: Setting permissions on a Sales share for the Sales Users group.

I definitely recommend that you have all of your resources properly permissioned before you undertake a provisioning implementation. All of the automated provisioning in the world won't help you if your resources are not actually protected the way you need them to be, and adding users to groups (or removing them from groups) doesn't actually help protect your data. Although a discussion of the cleanup of groups is not within the scope of this book, in the next section, I will talk about good strategies for managing group memberships in AD going forward.

Managing AD Group Memberships in an Automated Way

I've already talked about the entitling of users to resources as part of the provisioning process. Of course, this also holds true for de-provisioning—where the user is automatically removed from all the groups they are a member of when their account is deleted. But many organizations also need a good process for managing changes in group membership in an ongoing fashion—this relates to the second part of the identity lifecycle: keeping identities updated as time goes on to ensure that the identity continues to match the user's job function. From an AD perspective, there are two basic types of groups—security groups for use in permissioning resources and distribution groups for use in email systems like Microsoft Exchange.

Both types of groups benefit from some kind of automated management. The benefits of making the changes to group memberships automated are that you reduce the chances that users end up in groups that they don't belong in—thereby exposing the underlying resources that are permissioned by those groups to unnecessary risks of access. Automated systems for managing group membership are often incorporated into the provisioning platform, but regardless of whether they are or not, they have a number of characteristics that should be considered as you look to implement group management solutions:

- The ability to automate group membership changes based on external criteria—for example, a user's group memberships may need to change if their user object moves OUs or if an attribute on the user object changes. For example, a change in the "Department" attribute from "Sales" to "Marketing" might warrant an automated change in group membership.
- The ability to create profiles or job families for a given type of user. For example, it would be useful during the provisioning process to simply tell the system that the user being created is a sales user, which then triggers a set of related group membership changes that are pre-determined based on what a user needs to do in their job. This works equally well whether you're provisioning a new sales user or a new systems administrator, and is perhaps equally important when the groups you're provisioning the user into grant access to privileged operations.
- The ability to support approval-based workflows for adding of users to certain privileged groups. Regardless of whether the group grants access to privileged company data or privileged systems, you're probably going to want someone in a position of authority to approve any additions to these groups, even if it's part of an automated provisioning process. Having this kind of approval-based workflow in place typically requires the "owner" of the group, or the person responsible for the group's membership, to approve the addition of the new or changed user before the operation actually happens. This can slow down the provisioning process but ensures that only those users with a "need to know" are placed in those sensitive groups.
- Any group management solution should equally support the automatic removal of users groups if certain conditions apply. For example, if a user is disabled or moved to a special "To Be Removed" OU, this may warrant automatic removal from all groups that could grant access to that user. Even though disabling a user account in AD prevents that user from being able to use the account (and its related groups), removing the user from all of their groups guarantees that that user account doesn't show up on any audit reports.
- Attestation of group membership is an important feedback mechanism for any group management and provisioning solution. Attestation is the process by which the "owner" of a particular group periodically reviews a group's membership list to ensure that the users in that list still need access to the resources conveyed by that group. This attestation can and should be automated so that group owners are tracked and they receive periodic reminders that require them to purposefully attest to group memberships that they own. This is especially important for groups that convey privileged access or access to sensitive resources.

Keep in mind that all of these characteristics of automated group management can apply equally well to both AD security AND distribution groups. Sometimes distribution groups used for sensitive email communications are just as important to control as security groups used to control access to company data.

In addition, automated group management need not be limited to AD groups. Any groups or roles used by other identity stores should ideally fold into this process because it's not likely that AD and AD groups will not be your only mechanism of authorization within your environment. Remember that identity lifecycle management is about covering all of the different sources for authentication and authorization and ensuring that they fit into the identity management lifecycle. Your automated group management solution should have the ability to touch external groups and roles leveraging the same criteria listed earlier.

Leveraging Self-Service in Identity Lifecycle Management

As another part of the update phase of the identity lifecycle, anything you can do within your identity lifecycle management to reduce the points of contact between users and systems administrators will add to the reliability of the data that makes its way into the identity systems. Tasks such as password reset or account unlock requests and requests for group or distribution list membership changes can easily be offloaded to end users with the proper controls and workflows in place. Consistently, password resets of AD accounts rank high on the list of areas where Help desks spend their time. And yet, in this day and age, there is little reason not to offer a self-service password reset capability to your end users.

It's not just a question of time spent by the Help desk. Many organizations have outsourced their Help desk functions to low-cost locations, and therefore don't immediately see the monetary value in a self-service password reset system. However, when you count in the time lost by the end user trying to get into their account (keep in mind how much the CEO of a reasonably large corporation makes per hour and factor that into your costs when he/she has to wait 20 minutes to have the Help desk reset their password) as well the security issues of relying on someone calling a Help desk and "saying" they are the CEO, then the costs of a such a system are easily justified. Modern self-service password management solutions (see Figure 2.7) integrate into AD and other identity stores seamlessly and help synchronize password changes across all of those stores. They also provide security by allowing a user to choose "secret questions" that uniquely identify them to the system and ensures the Help desk can't be spoofed into giving away someone's password.

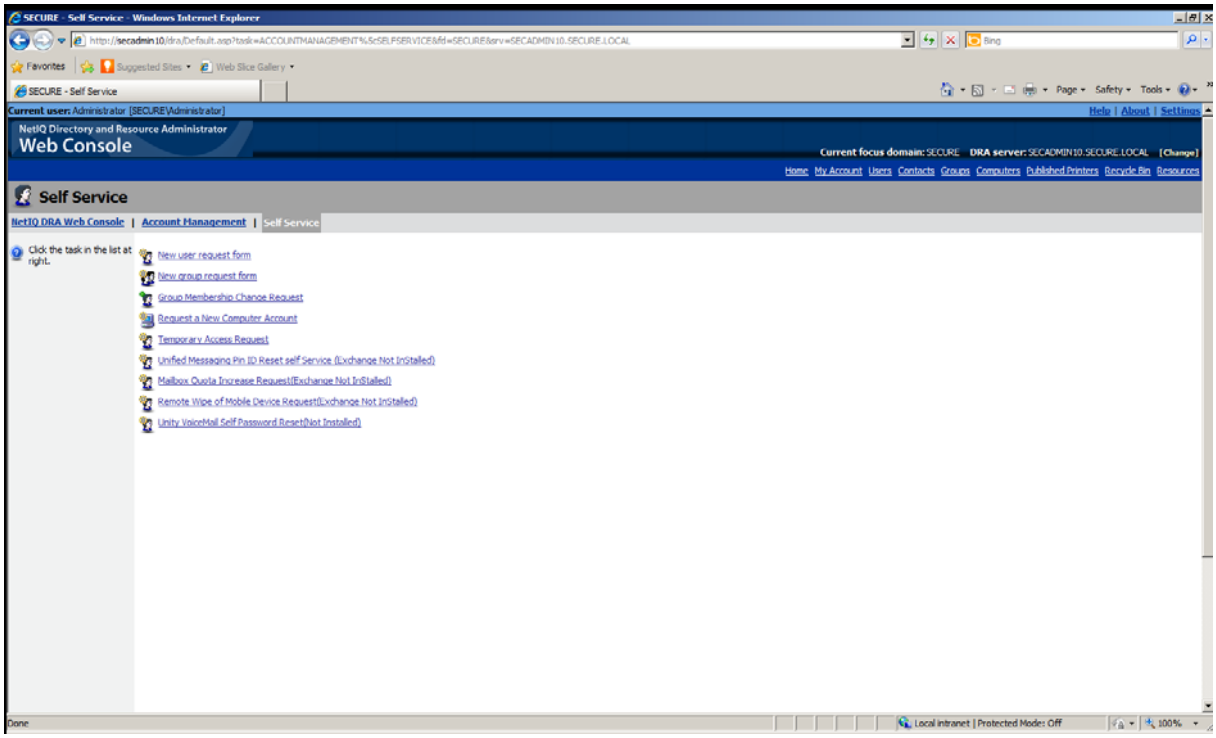

Figure 2.7: Viewing a commercial self-service password reset solution.

The best part about implementing such a solution is that you free up your Help desk to handle more pressing (and complicated) requests, while maintaining the security of your accounts.

Self-service group management is similarly liberating for both users and Help desk staff. Most group membership requests by users are fairly non-controversial, in that they are requests for users to be put into non-privileged groups that grant them access to some innocuous resource or email distribution list (for example, the company softball team d-list). These types of requests hardly require the user filing out a lengthy IT request form in their trouble ticketing system, providing justification, and so on and then requiring a senior administrator to perform the change in membership. Rather, a user should be able to go to a Web form, request membership in a group, and have the self-service tool handle their request right then and there. Typically, behind the scenes, the self-service tool uses a privileged "service account" that has native rights over all group's membership in the source identity stores and can make the change on behalf of the user—thus the user's own account does not need to be granted rights over the groups to make the changes.

For requests that involve changes to sensitive group or role memberships, self-service solutions typically allow for an approval-based workflow that asks the group's owner or responsible party to approve the group change, just as I described in the provisioning solution earlier. A group owner can approve or disapprove the change and then the process completes and the user is added to the group—or sent an explanation as to why they were not added.

Self-service applications not only reduce costs for users and Help desks but also help to keep your group management processes clean by ensuring that changes to groups or roles that grant access to sensitive company resources always go through a process that respects security permissions on those groups, respects a group owner's right to reject a group change, and are audited such that all group membership changes and their requests are tracked and recorded for future reporting and compliance audits. Speaking of audits, let's end the chapter by talking about the value of auditing changes during the provisioning process.

The Auditing Process

A key part of the identity lifecycle I presented in Figure 2.4 is the feedback loop that auditing provides. All of the three previous phases—creation, update, and removal of identities—require a good audit trail so that you and folks that plan to audit you can see the who, what, where, when, and why of each provisioning or de-provisioning event. Most provisioning solutions provide their own internal audit trail so that operations that are specific to the provisioning and de-provisioning processes, as well automated object or group membership changes that happen during the life of the identity, are audited within the context of that solution.

In other words, as long as you are using the product's interfaces to initiate provisioning, de-provisioning, or change events, these events are captured in the product's audit log. These logs are meant to provide a supplement of richer information with respect to changes to your identity system as compared with the native audit logs that exist in the various systems that you're managing. For example, a provisioning solution might audit the fact that a user was moved from the Sales to the Marketing OU by administrator Joe Jones, and it might also record that an automated rule triggered by that move caused Joe to be moved into the Marketing Users group and removed from the Sales Users group. However, it wouldn't replace the fact that the OS itself is also likely logging the resulting change within AD, as Figure 2.8 shows.

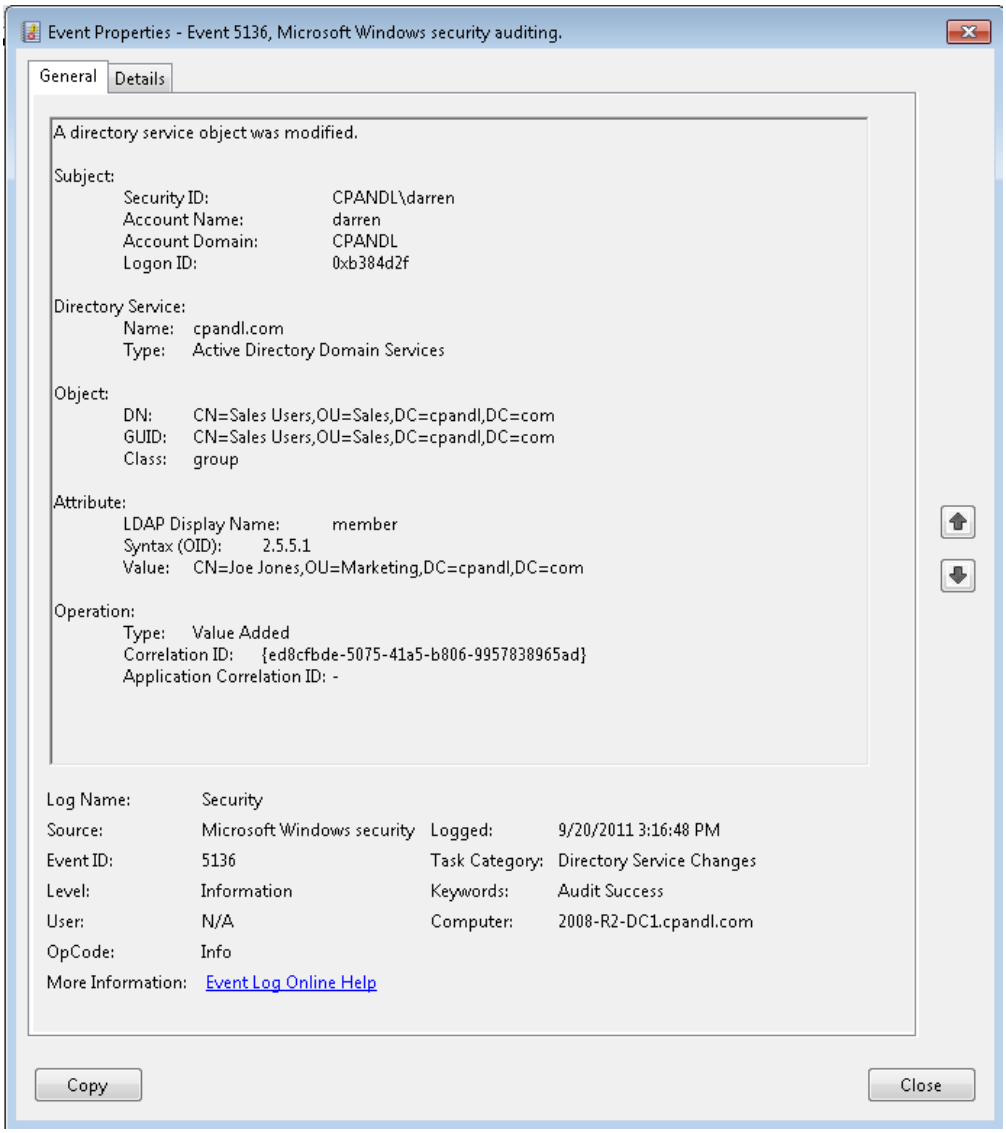

Figure 2.8: Viewing a group membership change in the native Windows security log.

The one aspect of native OS audit logs, such as the Windows security log, is that they are considered non-repudiated. That is, they cannot be altered under normal circumstances to hide an operation that a user does not want to have auditors or systems administrators see. This is a critical feature within a provisioning system, and one that you will likely want to be aware of, in case auditors ask about how secure your provisioning system's audit capabilities are. There are third-party solutions on the market, outside of the native Windows event logs, that support this non-repudiation feature. Figure 2.9 gives an example of one of these.

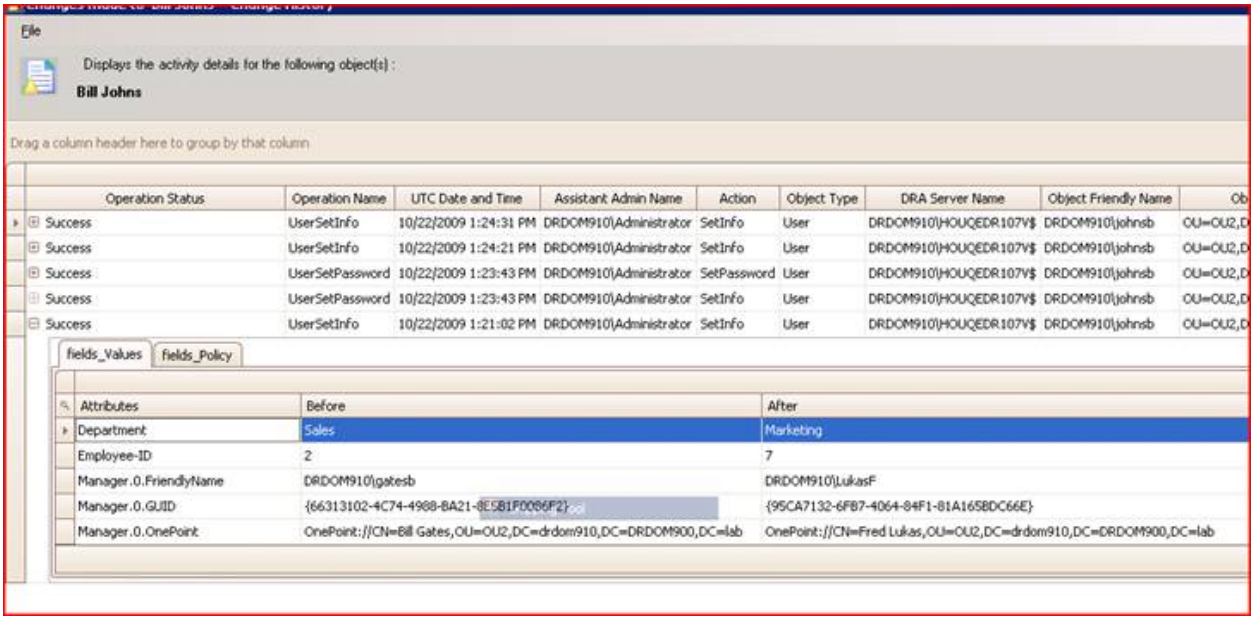

Figure 2.9: Viewing a third-party solution that supports non-repudiated audit logs.

Regardless of the mechanism that you use for tracking changes to your identity systems, it's important to ensure that each step of the identity lifecycle has an associated audit capability that records who made the change, what the value of the change was before and after the change (if relevant), and, ideally, records why the change was made (for example, as the result of an automated provisioning event out of the HR system or based on a self-service request from a user). This auditing is your safety valve to ensure that all identity lifecycle processes are working as expected. You can also use this information to compare with native OS and application security logs to ensure that there are no changes to authentication- or authorization-related data that is occurring outside of your official systems and processes. Auditors generally like to see that you not only have good processes in place for performing changes within the normal boundaries but also have mechanisms in place that ensure that you know when changes happen outside those boundaries.

Summary

This chapter talked about the importance and challenges of managing identities across the organization, especially in organizations with multiple identity stores and lots of sensitive data and privileged systems. The risks associated with poor identity management far outweigh the costs of implementing good provisioning and de-provisioning. And to that end, reducing the number of identity stores that you have in your environment as much as possible and implementing a provisioning and de-provisioning solution that covers those remaining identities and integrates those workflows into your HR system (or wherever identity originates in your organization) can go a long way towards reducing some of the risks associated with users who have inappropriate access. Automation is a key part of this story, with automated provisioning, updating, and de-provisioning of user accounts and group and role memberships being critical to any good identity lifecycle management system. Self-service can also help reduce mistakes and save time with respect to password and group/distribution list management and should be integrated into your identity provisioning solution. Finally, auditing, both through the provisioning solution and native OSs and applications, provides the necessary feedback loop to ensure that your provisioning and de-provisioning solutions are working and that users are not going around the system.