White Paper

The Challenges of Managing Privileged Access on Windows Desktops and Servers

By Darren Mar-Elia
Microsoft Group Policy MVP
& founder of gpoguy.com and sdmsoftware.com

February 2011

Darren Mar-Elia, MVP, is president and CTO of SDM Software, a Group Policy solutions company. He has over 20 years combined experience in information technology and software development. He was senior director of product engineering at Desktop Standard (acquired by Microsoft), and before that, served as CTO for Windows management solutions at Quest Software. He was also a director of distributed systems at Charles Schwab & Co. and helped guide that company's use of Microsoft Windows technologies. Darren has written or contributed to 12 books on Windows management topics, is a Microsoft MVP for Group Policy, and is a contributing editor for Windows IT Pro. He also created the popular gpoguy.com website for information and utilities related to Group Policy.

beyondtrust
privilege. made simple

www.beyondtrust.com

BeyondTrust
2173 Salk Avenue
Carlsbad, California 92008
Phone: +1 818 – 575 - 4000

Table of Contents

Overview

In this paper, I will talk about the goals and challenges of creating a privileged access management program for your Windows desktops and servers. Privileged access is a key issue these days, especially on desktops, for which an over-privileged user can be a weapon of destruction on your internal network if they inadvertently download and install malware.

But privileged access management can be equally critical on Windows servers to ensure that the right users have the right access to your production systems. In this paper, I'll lay out the issues and challenges around managing privileged access using the native Windows security model, especially across many desktops and servers. Then I'll talk about some different approaches that might help you in your efforts to manage access in your Windows environment.

What is Privileged Access Management?

It's important to define terms when it comes to privileged access management. Privileged access, as the name implies, is access to a system (desktop or server) that is above and beyond that which a "normal" user has access to. This definition, of course, is sufficiently vague that it requires additional context. Every organization may have a different definition of what is privileged, based on their business and security requirements.

But we can probably all agree that certain types of access (e.g. the ability to install software, create users or change system configuration) universally constitute privileged access. And the principles of Privileged Access Management are generally the same. Namely, you want to:

- Ensure that only those users who absolutely need access to a given set of privileges on desktops and servers have those privileges, and only for those systems for which they have a need.
- Ensure that privileged access is only used when it's needed (i.e. when the privileged operation needs to occur) and ideally, is only **granted** when it's needed and "un-granted" when it's no longer required.
- Centrally manage privileged access such that access can be granted and revoked quickly. This doesn't necessarily mean a single group managing access, but rather a single point of control or system for determining access, rather than many disparate systems depending upon the resource under management.
- Ensure that there is an audit trail for any privileged operation.

Of course, there will be differences between the types of privileged operations that are performed on Windows desktops vs. Windows servers. Desktop privileged access might include the ability to logon at the console (interactively) or the ability to install printers or install software. Server privileged access, on the other hand, might include the ability to shut down the server, change its network or disk configuration and stop and start key services.

Each of these different privilege sets may have different usage patterns as well. A desktop user probably needs to log onto their system interactively every day, and they may need to install software frequently as well. However, a server administrator may only need to log onto the server's console or restart its services on an infrequent basis. Therefore, the strategy you take when addressing privileged access management on each platform type may be different. We'll talk more about these differences later in the paper.

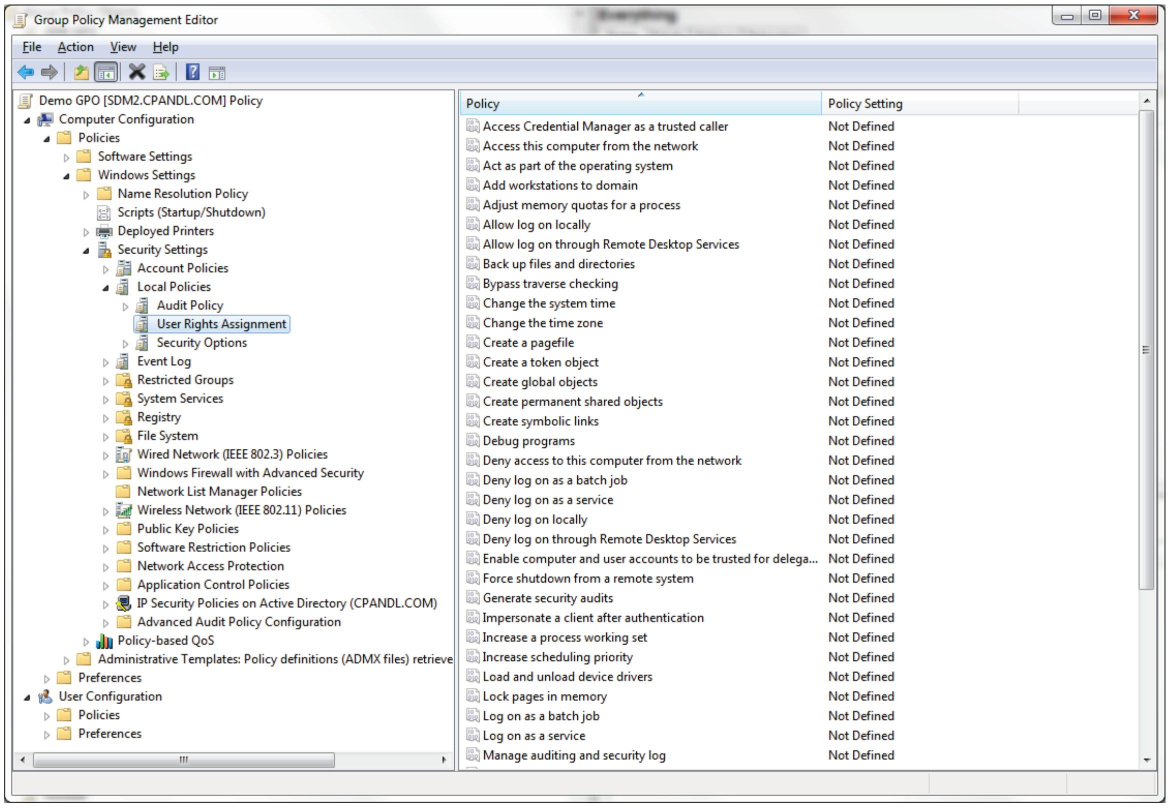
Privileged Access in Windows

The Windows platform provides some unique capabilities that also make it a challenge in the area of privileged access management. Ironically, one of Windows' biggest strengths over other platforms like Linux or UNIX is the granularity of control you get in the OS' security model. Indeed security is built into Windows as the deepest levels, and practically every resource and object in the OS is protected by some sort of access control list (ACL).

This results in unprecedented control over many aspects of a Windows system, but also unprecedented complexity when it comes to devising an enterprise-wide privileged access system. As an example, the following table shows just a few of the fundamental objects within Windows that have some kind of access control list associated with them.

Object	Purpose
System Services	Provide persistent processes on within Windows (managed via the Services Control Panel applet) but security controlled via the registry
Processes	Every process started by Windows has ACL associated with it
Files or Registry keys	Of course, the NTFS file system and the registry provide a robust permissioning system that lets you control who can read and write to files or keys
Event Logs	You can control who can view event logs and the OS has built-in controls over, for example, the security log.

In addition to ACLs, Windows has the concept of user rights or **privileges**—sets of built-in OS-specific primitives that grant access to certain tasks within the OS. As an example, you can see these user rights within the Group Policy Editor MMC snap-in within Windows, as shown in **Figure 1** below:


Group Policy Management Editor

File Action View Help

```
Demo GPO [SDM2.CPANDL.COM] Policy
  Computer Configuration
    Policies
      Software Settings
      Windows Settings
        Name Resolution Policy
        Scripts (Startup/Shutdown)
        Deployed Printers
        Security Settings
          Account Policies
          Local Policies
            Audit Policy
            User Rights Assignment
            Security Options
          Event Log
          Restricted Groups
          System Services
          Registry
          File System
          Wired Network (IEEE 802.3) Policies
          Windows Firewall with Advanced Security
          Network List Manager Policies
          Wireless Network (IEEE 802.11) Policies
          Public Key Policies
          Software Restriction Policies
          Network Access Protection
          Application Control Policies
          IP Security Policies on Active Directory (CPANDL.COM)
          Advanced Audit Policy Configuration
        Policy-based QoS
      Administrative Templates: Policy definitions (ADMX files) retrieve
    Preferences
  User Configuration
    Policies
    Preferences
```

Policy	Policy Setting
Access Credential Manager as a trusted caller	Not Defined
Access this computer from the network	Not Defined
Act as part of the operating system	Not Defined
Add workstations to domain	Not Defined
Adjust memory quotas for a process	Not Defined
Allow log on locally	Not Defined
Allow log on through Remote Desktop Services	Not Defined
Back up files and directories	Not Defined
Bypass traverse checking	Not Defined
Change the system time	Not Defined
Change the time zone	Not Defined
Create a pagefile	Not Defined
Create a token object	Not Defined
Create global objects	Not Defined
Create permanent shared objects	Not Defined
Create symbolic links	Not Defined
Debug programs	Not Defined
Deny access to this computer from the network	Not Defined
Deny log on as a batch job	Not Defined
Deny log on as a service	Not Defined
Deny log on locally	Not Defined
Deny log on through Remote Desktop Services	Not Defined
Enable computer and user accounts to be trusted for delega...	Not Defined
Force shutdown from a remote system	Not Defined
Generate security audits	Not Defined
Impersonate a client after authentication	Not Defined
Increase a process working set	Not Defined
Increase scheduling priority	Not Defined
Load and unload device drivers	Not Defined
Lock pages in memory	Not Defined
Log on as a batch job	Not Defined
Log on as a service	Not Defined
Manage auditing and security log	Not Defined

Figure 1: Viewing User Rights within Windows from Group Policy

As you can see from the figure, there are rights for everything from who can logon to the local console of a system or who can remote desktop to a system, to who can shut down a system to who can change the time zone on a system. Windows comes with a set of default rights configurations that vary based on the OS version (e.g., Windows 7 has a different set of rights and different set of built-in groups assigned to those rights than Windows XP). You can use Group Policy to manipulate these rights delegations (more on this later).

Bringing It All Together

So at the end of the day, Windows is an amalgam of ACLs and user rights (and in Windows Vista and Windows 7, something called "integrity levels" that further control which processes can talk to other processes). The process of managing privileged access on Windows, therefore, is the process of ensuring all of these ACLs and privileges are configured across all systems in an appropriate manner to meet the goals we talked about earlier.

If I want to delegate the ability for a group of Admins to be able to stop and start the spooler service, but do nothing else, on a group of 10 servers, I will likely have to configure a set of both ACLs on the services and privileges on the servers to manage that privilege. This scenario I've just described underscores the challenges of managing privileged access on Windows. "Privileges" are not well-defined things with black and white sets of ACLs or rights.

Privilege is often granted using a combination of these security constructs. In addition, some privileges are built into the OS while others must be defined, based on your definition of the privileged access. A simple example of this is that if you need to delegate the privilege to log onto a desktop and only modify the configuration of an installed application, this may end up being a combination of user rights and file system or registry permissions and it may be something you have to craft yourself and grant to a user or group on one or more workstations.

How Windows Access is Granted

We've talked about how Windows security model provides a very granular mechanism for delegating privileged access to OS objects or tasks. But how do you manage that privileged access in a typical Windows network? Well, access is typically granted to **security principals**. A security principal is a user, computer or security group within Windows. These can be defined locally (within the local Security Accounts Manager (SAM) database on a workstation or server) or within the Microsoft Active Directory (AD).

The latter is certainly the most common way of managing security principals across any decent-sized organization, as it centralized user, computer and group accounts into an LDAP-exposed directory service.

Best practice is generally to grant privileged access to groups, rather than individual users, since groups are easier to manage and result in smaller and more easily manageable ACLs. Active Directory supports three different kinds of groups, including Domain Local, Global and Universal Groups, that differ in their scope of usage. A deep discussion of the differences between these is beyond the scope of this paper but you can find plenty of information on group differences on the Microsoft TechNet site.

Privileged Access Using Windows Built-In Groups

As I alluded to above, there are a set of "built-in" security groups that come with Windows and Active Directory, that have hard-coded access to certain OS resources. These groups are typically privileged groups and their "powers" cannot be modified.

Or at the very least, they can't be reduced—they can only be added to. The table below gives an example of some the built-in groups available in Windows 7, and what privileges they impart:

Built-In Group Name	Privileges Provided
Administrators	The highest level of privilege on a Windows system. Members of this group can perform most any task on the system
Backup Operators	Members of this group may not have rights to files on the file system, but can still read those files for purposes of backing them up.
Event Log Readers	Members of this group can read all event logs on the system, even though they are not administrators on the system. This includes security logs
Network Configuration Operators	Members of this group have the ability to modify network stack configurations
Performance Log Users & Performance Monitor Users	Members of these groups can set up performance and trace logging and access performance monitor counters, respectively
Remote Desktop Users	Members of this group can remote desktop to this machine

As I mentioned, these built-in groups have hard-coded access, built into Windows that grants them access to the resources required to grant the various operations described above.

Of course, this is a fairly short list when you consider the full universe of privileged operations that you might want to perform on a Windows system. And this doesn't even include the various add-ons to Windows that you might want to control access to, such as IIS, SQL Server, Active Directory itself, etc.

Active Directory Built-In Groups

Active Directory comes with its own set of built-in groups, which like their Windows counterparts grant hard-coded access to certain operations within the directory. The **Domain Admins** group is an example of a group that grants "god-like" powers over Active Directory and is built into AD out-of-the-box.

The one thing you need to keep clear is that **privileged access to an AD object does not confer privileged access to the underlying resources**. What this means is, just because I have rights within AD to modify the computer account of a server or workstation, does not necessarily mean I have any special access to the underlying physical resource that is represented by that AD object. In fact, there is generally no connection whatsoever between the two sets of privileges.

I may have full ability, by virtue of my AD rights, to delete a workstation's AD object, but in fact not even be able to log into that workstation. This is an important distinction that should factor into your privileged access management plans. In reality, you probably need to have two different plans-one for managing privileged access to AD, and one for managing privileged access to the underlying resources (i.e. servers and workstations).

Using Built-In Groups for Privileged Access Management

You might be tempted to take advantage of some of the built-in groups within Windows for controlling privileged access. Indeed, in newer versions of Windows such as Windows 7 and Server 2008-R2, Microsoft has done a better job of creating more of these groups to allow for more granular privilege delegation.

But, by and large, these groups are still pretty coarse-grained. If you are trying to achieve "least privilege"—granting only those privileges needed for a user or administrator to do their job, against specific targets, for limited periods of time—then these built-in groups typically don't meet those requirements. They are broad groups that either grant too narrow a set of privileges (e.g. Network Configuration Operators) or too broad (e.g. Administrators or Server Operators).

So, while it may seem like a good idea to "punt" and give your users membership in these groups, it should only be done as a last resort rather than a strategy for managing privileged access.

Privileged Access and Group Policy

So far, we've talked about how Windows controls access to its resources (using ACLs and privileges), and how you can grant these accesses (using security principals locally or in AD). Now, we need a method for distributing and managing privileged access across many servers or workstations.

Enter Windows Group Policy! Group Policy is the technology built-into Windows for managing configuration—primarily security configuration. Group Policy is at its best when Active Directory is involved, since AD-based Group Policy gives you many more policy configuration options and a more robust targeting mechanism, as compared to having to configure local policy on every individual workstation or server.

So, Group Policy and Group Policy Objects (GPOs) in particular give you the distribution mechanism for privileged access management in Windows. Specifically:

1. **Privileged access is about:**
   Defining the groups that need elevated access, and modifying the security on the **resource** to enforce the privileges for each group

2. **AD provides the mechanism for #1**
   It lets you centrally define groups that correspond to tasks that need to be privileged (e.g. you could create new Print Server Admins or Web Site Recycler Admins that are specifically granted access to resources required to perform tasks for those Admins who are members of the groups)

3. **Group Policy is often the mechanism for #2**
   Provides centralized control of user rights and ACLs on groups of Windows desktops and servers

Policies for Controlling Privileged Access

Using Group Policy, there are basically three mechanisms for distributing privileged access.

- Users can be put in built-in groups, using Group Policy, that confer automatic access to certain things (less desirable since, as we mentioned, built-in groups can be overly coarse)
- Users can be added to ACLs on resources
- Users can be granted user rights (privileges) on a system

Within the Group Policy namespace, (i.e. if you're in the Group Policy Editor) the policies you will most often encounter for managing privileged access can be found **under Computer Configuration/Windows Settings/Security Settings.** As the name implies, this area contains security related policies, including the mechanisms for enforcing privileged access. More specifically, within this part of the GP namespace, the following areas address the three bulleted areas shown above.

Local Policies/User Rights Assignment lets you grant groups access to certain user rights
Restricted Groups lets you enforce membership on built-in groups
System Services lets you grant access to services (e.g., who can stop, start and reconfigure Windows services)

Registry & File System lets you control ACLs on files and registry keys
Note that you can see each of these areas within the GP namespace in Figure 1 earlier in this paper. Note that not all of these areas are available on the local GPO. You must be using AD-based GPOs to have access to all of these policy areas.
For more information on using each of these policy areas, check out my resource site at www.gpoguy.com or the Microsoft TechNet site.

Why Its Difficult Managing Privileged Access with Group Policy

So, at first blush, it would seem that managing privileged access with the tools in the box would be pretty straightforward. Unfortunately, that isn't the case. If you have a small network (e.g. 100 nodes) then it is probably possible to manage privileged access using Group Policy, but then I would argue you are probably too small care about the problem at that level.

However, anyone in a larger environment will quickly run into some of the limitations of both the Windows security model and Group Policy itself. I've already mentioned some of the reasons for this. Namely,

- The Windows security model, compared to other OS' like Linux, is almost too granular
    - Too many points of control
    - Forces administrators to "punt" and grant administrative access to ensure that things don't break

- Lack of a clear definition of what rights and permissions are required for which tasks
    - In an ideal world, what you want is to be able to say: "In order to perform task X, I need Y rights"
    - A lot of guessing and testing, or again, punting and putting users into built-in groups

- Group Policy could have been the ideal Privileged Access Management tool, but it has to work within the confines of the Windows Security Model
    - Too many combinations of policies, servers, workstations and tasks means that GPOs can easily become overwhelming and inflexible
    - Group Policy is best when used for simple delegation across large blocks of Windows machines
    - Doesn't handle subtle differences (e.g. Admin X can perform Y tasks on Servers 1-10, but not servers 11-20)

To underscore these points, I will give an example. Let's say that you want to control who can shut down groups of servers, without giving those users administrative control over the server. You can easily do that using Group Policy user rights assignment. So let's say you define a group in AD called "Shutdown Admins" and use Group Policy to apply that group to the "Shut down the System" user right on 1000 servers. Great! Now suppose you also want to grant some departmental Admins the ability to shutdown 50 of those 1000 servers. Sorry, no can do.

Because Group Policy has full control over that user right, any exceptions will be overwritten every time you try to add the departmental Admins group to those 50 server's user rights. The only way you can do it is create two GPOs that essentially overlap each other, and use "security group filtering" to make sure the 50 servers get the right groups assigned to that user right. Not a pretty sight when you try to scale that up to lots of different administrator roles across many different servers.

So, in the end, you are forced to find another way if you really want to get a handle on this privileged access management thing.

Finding another Way to Privileged Access Management

So, what's the solution? Well, I will tell you that it's not an easy problem to solve, which is why many shops do simply punt and make their desktop users administrators or their server users administrators on their Windows systems. But, as I've already mentioned, that creates business and security risks that will ultimately become unacceptable for most organizations. As a result, creative solutions are required.

And, as it turns out, you'll probably take different approaches for servers vs. desktops. This is because of the different usage models for each platform type. On the desktop, privileged access is about preventing or allowing the logged-on user to perform certain tasks. On the server, privileged access is about preventing, or allowing, administrators to perform infrequent, but highly privileged tasks.

On the desktop side, there is now a class of 3rd party products on the market, such as BeyondTrust's **PowerBroker for Desktops**, that provide a different way. Namely, these products start from the assumption that none of your users have privileged access to their desktops. You then create rules that elevate applications and tasks on a per-process basis, ensuring that only the needed rights and privileges are applied to that process to allow it to run successfully.

Think of it as elevating on demand and only when the process is executed. The user is and always remains a non-privileged user, but based on your rules; their applications and tasks can run as elevated users just for the duration needed. And, best of all, products like PowerBroker for Desktops use Group Policy as the rule distribution mechanism—letting you use familiar tools to create and target your rules, but leaving the complexities of the Windows security model and limitations within security Group Policy behind.
On the server side, the picture is less clear, but I can recommend some general approaches.

First off, the problems of trying to distribute management of the native Windows server security model are difficult enough that they should just be avoided. What do I mean by that? Essentially, the approach I would recommend is to come up with a list of tasks that your administrators need to perform against Windows servers. Once you have that set of tasks, you can take a "proxy"-based approach to allowing users to perform those tasks. Proxies are a class of products or an approach to management where the user themselves does not have native privileges on the underlying system.

However, the proxy application runs in the context of a user who does have full privileges on systems to perform privileged tasks. You can then define rules on the proxy to control which user groups have access to which tasks on which servers and at which times. From there, the proxy becomes your single point of control for all privileged access management of servers. You no longer have to worry about configuring the native security on 100s or 1000s of servers. You use the proxy as the “choke point” to ensure that you are doling out appropriate privileges only when needed.

That being said, this is not a trivial thing to do on Windows servers. Products like BeyondTrust’s PowerBroker Servers has taken this approach for UNIX and Linux for a while, but with those OS’, you only have a command-line interface to worry about. Windows is typically managed using a combination of GUI tools and command-line utilities, making a proxy solution more challenging, but not impossible.

At the moment, I haven’t seen a perfect solution for managing privileged access on Windows servers. But regardless of what approach you take, I do believe that the task-based approach is the right way to approach this problem. Instead of trying to organize your user populations by job, come up with a set of tasks that all administrators of servers need to perform. This list should be fairly well defined (e.g. stopping and starting services, installing software or patches, changing network configuration, modify security parameters, etc.) Once you have that list, focus on creating a privileged access management plan for delegating those tasks. I guarantee the list won’t be complete day one.

But as more users become subject to this approach, you will find more and more tasks that you can incorporate into the management plan. And whether you use Group Policy or a proxy-based approach, ultimately the problem will remain manageable and you’ll have more control over your Windows servers than if you had just “punted” and given your users too much privileged access in the first place.

About Darren Mar-Elia

Darren Mar-Elia, MVP, is president and CTO of SDM Software, a Group Policy solutions company. He has over 20 years combined experience in information technology and software development. He was senior director of product engineering at Desktop Standard (acquired by Microsoft), and before that, served as CTO for Windows management solutions at Quest Software. He was also a director of distributed systems at Charles Schwab & Co. and helped guide that company's use of Microsoft Windows technologies. Darren has written or contributed to 12 books on Windows management topics, is a Microsoft MVP for Group Policy, and is a contributing editor for Windows IT Pro. He also created the popular gpoguy.com website for information and utilities related to Group Policy.

About BeyondTrust

BeyondTrust is the global leader in privilege authorization management, access control and security solutions for virtualization and cloud computing environments. BeyondTrust empowers IT governance to strengthen security, improve productivity, drive compliance and reduce expense. The company's products eliminate the risk of intentional, accidental and indirect misuse of privileges on desktops and servers in heterogeneous IT systems.

With more than 25 years of global success, BeyondTrust is the pioneer of Privileged Identity Management (PIM) solutions for heterogeneous IT environments. More than half of the companies listed on the Dow Jones Industrial Average rely on BeyondTrust to secure their enterprises. Customers include eight of the world's 10 largest banks, seven of the world's 10 largest aerospace and defense firms, and six of the 10 largest U.S. pharmaceutical companies, as well as renowned universities.

The company is privately held, and headquartered in Los Angeles, California, with East Coast offices in the Greater Boston, Washington DC, as well as EMEA offices in London, UK.

Visit www.beyondtrust.com for more information.