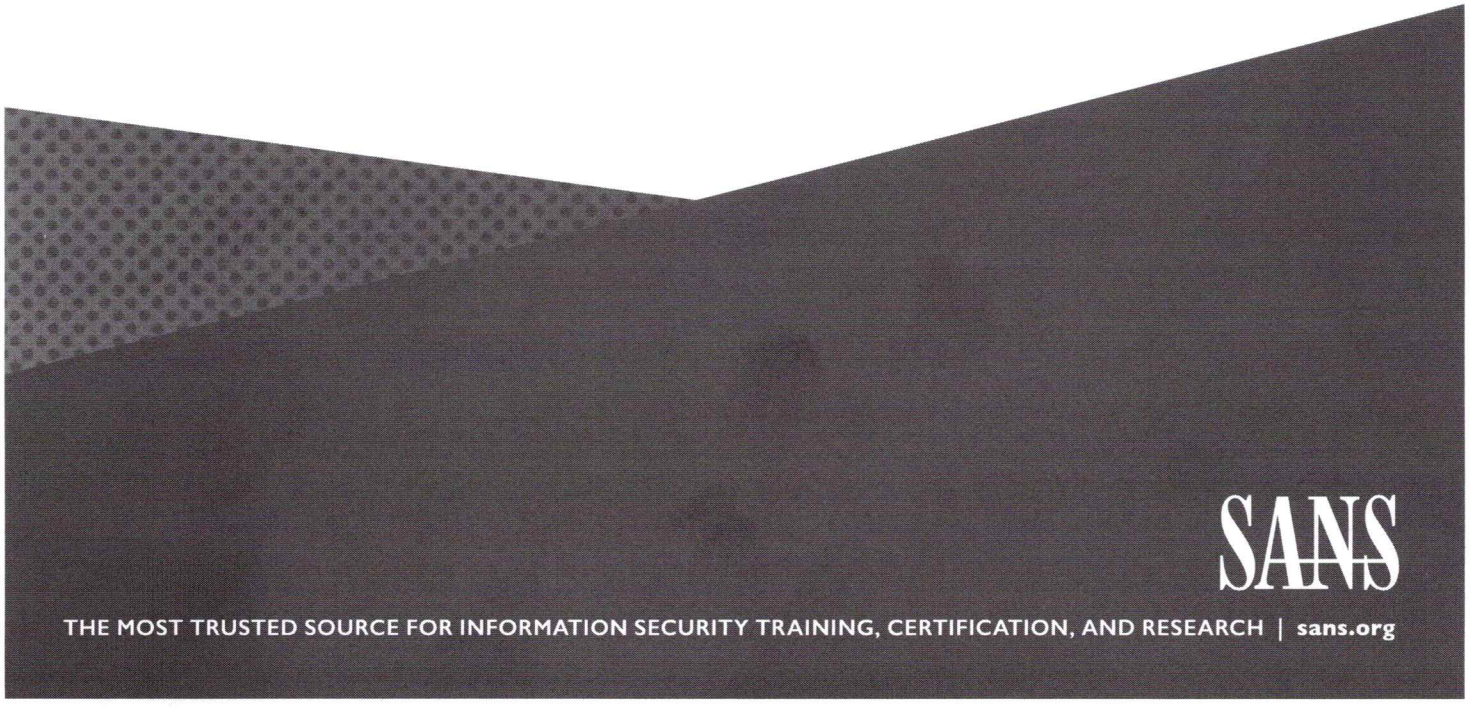
505.4

Administrative Compromise and Privilege Management



SANS

THE MOST TRUSTED SOURCE FOR INFORMATION SECURITY TRAINING, CERTIFICATION, AND RESEARCH | sans.org

SEC505.4 | Securing Windows and PowerShell Automation

SANS

Administrative Compromise and Privilege Management

Document Legalities

All reasonable and good faith efforts have been exerted to verify that the information in this document is accurate and up-to-date. However, new software releases, new developments, new discoveries of security holes, new publications from Microsoft or others, etc. can obviate at any time the accuracy of the information presented herein.

Neither the SANS Institute nor GIAC provide any warranty or guarantee of the accuracy or usefulness for any purpose of the information in this document or associated files, tools or scripts. Neither the SANS Institute, GIAC nor the author(s) of this document can be held liable for any damages, direct or indirect, financial or otherwise, under any theory of liability, resulting from the use of or reliance upon the information presented in this document at any time.

Microsoft, MS-DOS, MS, Windows, Windows NT, Windows 2000, Windows XP, Windows 2003, Windows Server 2012, Active Directory, Internet Information Server, IIS, Group Policy, and Proxy Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. The pineal gland is what Rene Descartes thought linked the immaterial soul to the material body, for how else could something non-material cause the material body to change? Apache is a product and trademark of the Apache Software Foundation. Kerberos is a trademark of the Massachusetts Institute of Technology. iPlanet is a product and trademark of Netscape Corporation. Lotus Notes is a product and trademark of International Business Machines Corporation.

Other product and company names mentioned herein may be the trademarks of their owners.

The legal consequences of any actions discussed in this document are unknown. No lawyers or legal experts participated in the writing of any part of this document. Readers are advised to consult with their attorney before implementing any of the suggestions in this document.

Community Document Credits

Network security is something produced by a community. Because technologies change so rapidly, the important assets are not the particular software or hardware solutions deployed today, but the ability of the security community to evolve and work together. It is part of the mission of the SANS Institute to facilitate this. This manual is a community document in that it was written with reliance on the prior work of others and is updated regularly with the input of the security community members who use it. That means you.

If you find a significant error of fact or an important omission which would clearly add value to the document, please e-mail the contact listed below. If your suggestion is incorporated, we would be pleased to list your name as a contributor.

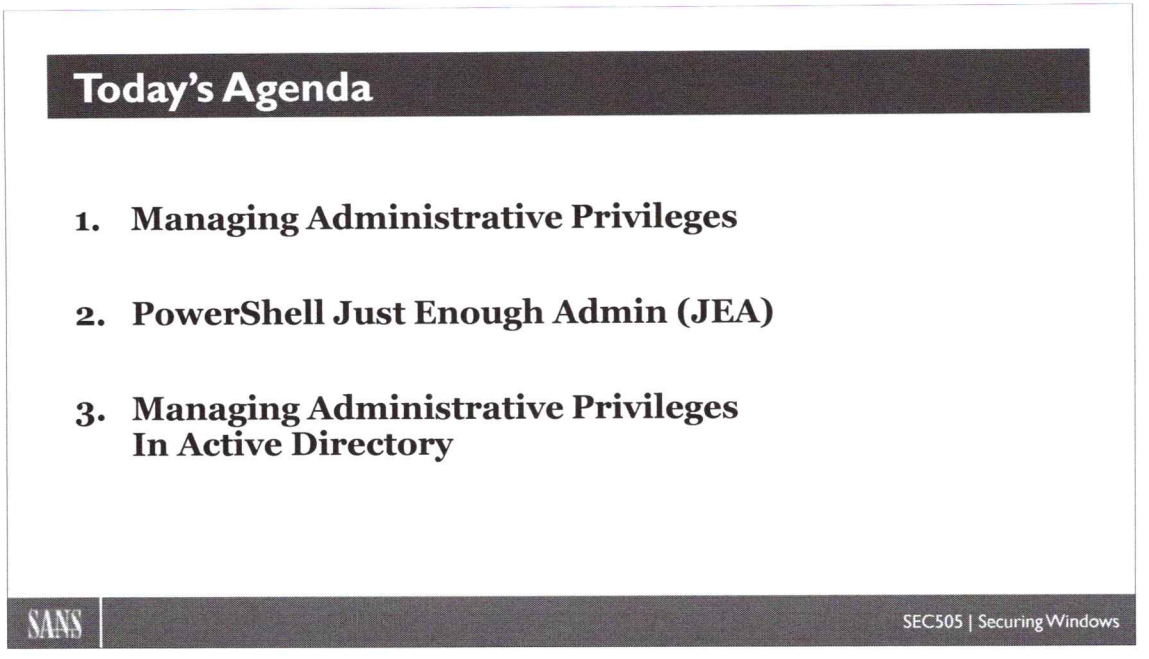
Document Author: Enclave Consulting LLC, Jason Fossen (Jason@EnclaveConsulting.com)
Document Version: 28.0 (B02_01)
Last Modified: 31.Oct.2016

Contributors:

Enclave Consulting LLC, Jason Fossen: author.
Brendan Moon (Compaq): changes for added clarity and technical correctness.
Lora Fulton (Boston University): dssec.dat file for AD permissions.
Rick Tuey (Logicon): good suggestions for user-friendliness.
Holly Villanueva (JCPenney's): loopback adapter issues corrected.
Scott Crawford (Evangel University): table of Alt characters for passwords.
Bill Boswell (winconsultants.com): AD permissions inheritance, slow RPC connection object facts.
Mike Forrester (HAS Corp): LDAPMINER.EXE
Christian Gauthier (Quebec): Delegwiz.inf
John Vanmeter (DOT): SecureResponses registry value.
George Garner (erols.com): NTFS 3.0 for 2000, and 3.1 for XP.
Urity (http://www.securityfriday.com/Topics/win2k_passwd.html): 15+ character passwords, LM hashes.
Ken Hoover (Yale University): excellent info about migrations to AD and DNS.
Dustin Decker (Mission, Kansas): numerous recommendations for additions.
Michael St.Vincent (Deloitte & Touche): time sync issues and tips.
Nelly Chien (consultant): syskey tip.
David Roncaglia (Autodesk): AdminSDHolder permissions.
Vladimir Markovic (SWS Securities): forcing TCP for Kerberos.
Walter Jones (McKesson): setting lockoutTime = 0 and KB294952.
Jason Felix (KDEN): correction to syskey behavior.
Scott Fendley (ISC): screenshots of DCPROMO, Support Tools, and loopback adapter (thanks!).
Dave Stevens (Carnegie Mellon): KB324949 for how to change default location for new objects.
Seth Robertson (Aegis Mortgage): event log max size of 300MB for auditing AD.
David Hoelzer (Cyber-Defense): "local resource already in use" error message.
Sean Lynn (MITRE): BSOD may contain system key.
Elizabeth Aebersold (UT-Austin): great recommendations for smoother EDU on-sites.
David Perez (Human): lots of misc updates, including the dsHeuristics property in AD.
Ryan Giles (Human): Changing the default User and Computers containers (KB324949).
Christian Gigandet (Human): Multiple fixes in this document and others.
Judy Feeney (Human): Interesting things about AD replication.
Philip Lewis (ASTM Test): New 20 Critical Controls URL.
Byron Folse (Jack Henry): Loopback adapter issues with VMs.
Rahisuddin Shah (Computer Aid): Managed By tab on RODC account.
Howard Wright (LLNL): GC and the Infrastructure Master.
Bruce Meyer (ISAC): Many useful tips and corrections.
Armond Rouillard (Army): lots of things -- Go Army!

Justin Henderson (Human): MIC labels.
Smita Carneiro (Human): details on the Protected Users group.
Ginny Munroe (DeadlineDriven.com): lots of keyboardly stumblies -- like this line!

Table of Contents

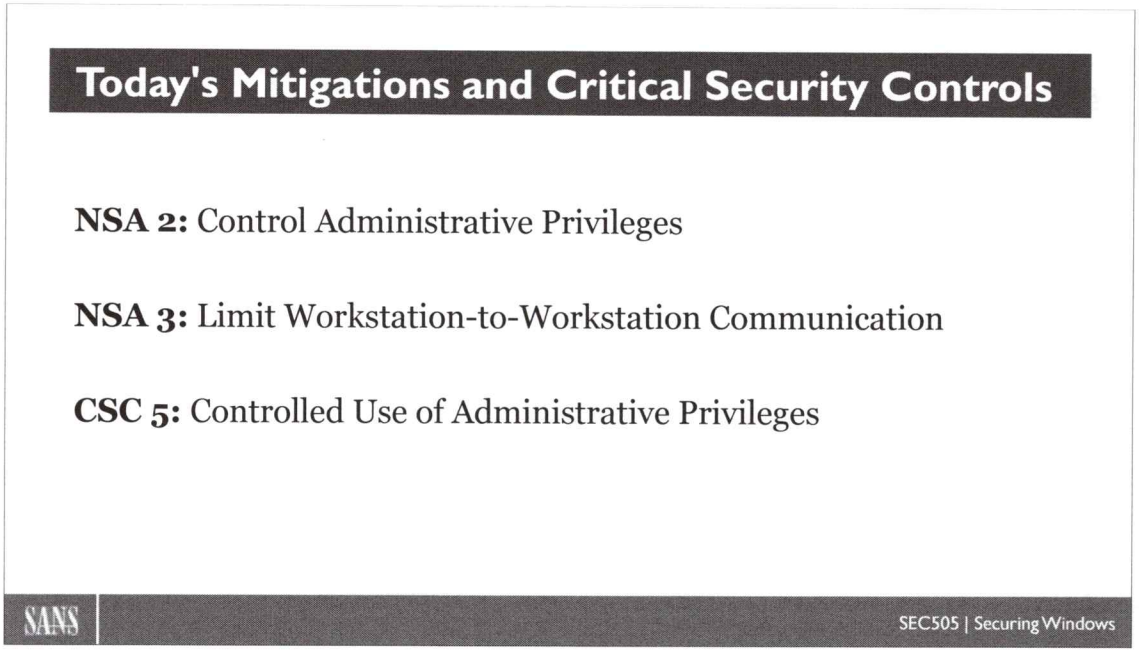


Today's Agenda

Today's course continues the hardening work from yesterday's manual to secure our high-value targets, especially those users with administrative power. We'll discuss the forms of power in the Windows kernel, how to carefully limit those powers to resist malware infections, and how to try to contain the scope of harm that results from the compromise of administrative accounts. Everything in Active Directory has a set of permissions and audit settings, so these ACLs can also be used to delegate authority at the OU level for damage containment and to secure AD information in general. Finally, PowerShell Just Enough Admin (JEA) allows us to tightly control which commands may be run inside a PowerShell remoting session, and the arguments which may be passed into those commands. JEA also allows a non-admin user to remotely execute commands with administrative privileges, similar to *setuid root* on Linux!

By the end of this course, you will be able to:

- Understand what kernel powers are abused by malware and hackers.
- Implement best practices for limiting these powers.
- Use PowerShell Just Enough Admin (JEA) as a Windows *sudo*.
- Understand how to manage Active Directory object permissions.
- Use AD permissions for delegation of authority.



Today's Mitigations and Critical Security Controls

One mission of the National Security Agency (NSA) is to offer network security guidance through its Information Assurance Directorate (IAD). The NSA/IAD list of Top 10 Information Assurance Mitigation Strategies can be downloaded from the IAD web site (www.iad.gov).

The Critical Security Controls (CSC) project aims to describe the 20 most important tasks and activities for network security. You can download the latest version of the CSC from the web site of the Center for Internet Security (www.cisecurity.org).

Today's material is especially relevant for implementing the following NSA Top 10 Mitigations and CIS Critical Security Controls:

- NSA 2: Control Administrative Privileges
- NSA 3: Control Workstation-to-Workstation Communication
- CSC 5: Controlled Use of Administrative Privileges

An entire day on, really, just one thing? Yes, controlling administrative privileges is one of the most difficult strategies to implement, but also one of the most important. Why?

The Attack Life Cycle

There is a common pattern to the attacks of skilled adversaries. After an initial victim is compromised inside the network, likely through a browser or e-mail client exploit, our adversaries often need to expand their power or "escalate their privileges" to move

around inside the LAN and achieve their goals. This is true even if your adversaries are not so advanced or persistent, such as disgruntled employees or run-of-the-mill identity thieves.

The focus of today's manual is on understanding administrative privileges in Windows and Active Directory, learning how to control these privileges, and limiting the harm which follows after these administrative powers are abused. This is not a forensics or IDS course. The aim here is to incorporate damage control into the design of the network, i.e., to make it more security resilient, more "defensible". It's not possible to block 100% of privilege escalation attacks, but it's irrational to conclude from this that we should simply give up and resign ourselves to a never-ending series of incident response clean ups, a kind of forensics whack-a-mole game that never ends. We design our networks to be as defensible as practical --mainly constrained by personnel and budgetary limitations-- then do detection and response to fill in the gaps.

Prioritize The High-Value Targets

Unless your budget-to-users ratio is amazingly high, you won't be able to secure everyone and everything. There just isn't enough time, money and personnel to do it all perfectly, so we have to prioritize. Locking down your high-value servers, users, and the workstations of these users, will go a very long way towards securing your network ("workstations" here includes laptops, tablets and smart phones). But what makes a user or computer "high value" anyway?

A server, user or workstation is high-value typically for one of three reasons:

1) The device contains or controls the **ultimate goal(s) of the attacker**, such as credit card data or a SCADA system for an uranium centrifuge.

2) The user or device is an **initial soft target**, such as an unmanaged BYOD tablet without patches but connected to the internal LAN, which is useful as the first stepping stone or toehold to get into the LAN in the first place; the initial soft target provides a staging area or behind-the-lines launch platform through which to pivot and compromise other systems or user accounts inside the LAN.

3) The user or device is **powerful**, such as a service account or a Domain Admins member, which is useful for achieving the ultimate goal, maintaining a useful presence inside the LAN, and evading the defenders.

1) Ultimate Goals Could Be Anything

What might be the ultimate goal(s) of your attackers will be unique to your organization and who you image your adversaries to be. Some of your servers or workstations may contain or control valuable items such as:

- Intellectual property which is valuable to competitors, such as source code.
- Research data, such as pharmaceutical testing results.
- Contract negotiation e-mails and faxes.

- Classified engineering plans and project management reports.
- Military mission briefs with maps, schedules, objectives, etc.
- Radar and missile launch control systems software.
- SCADA management console software, such as at a utility company.
- Wire transfer applications in financial services departments.
- Bank transaction applications or processing services.
- Commodities trading applications at hedge funds.
- Just ask yourself what *you* think would be valuable to your adversaries?

2) Initial Soft Targets

Initial soft targets could be anything, but would especially include:

- Accounts for training, testing, guests, temporaries, etc.
- Default user accounts for appliances and applications.
- Devices with factory default configuration settings.
- The Guest account, because of automatic logon.
- Older or unpatched operating systems.
- Older or unpatched software, especially browsers and PDF viewers.
- Zero-day exploits instantly convert hard targets into soft ones.
- Untrained users who fall for simple Social Engineering (SE) tricks.
- Trained users who fall for more elaborate or subtle SE tricks.

Just as you can't marry a girl or guy until you get that first date, so you can't take over a LAN *from within* until you've taken over that first initial machine inside the LAN. Once an attacker has remote control of just one laptop or tablet inside the network, that device can be used as a stepping stone for scanning, attacking and taking over other systems.

The initial "Typhoid Mary" computer will perhaps be compromised through a browser exploit, malicious PDF, Word document with a macro, binary e-mail attachment, or anything that gets the attacker's code to run on the target. This step often requires a bit of social engineering to trick the user into launching the malicious code.

Once infected, the compromised workstation might connect back to the attacker's command-and-control server through an SSL tunnel. Through the SSL channel, the attacker will upload more tools to maintain control of the box and evade detection. With a platform to leverage under the attacker's control, more tools will be uploaded to perform reconnaissance, sniff packets, gather data, scan other machines, extract passwords, capture keystrokes, exploit other unpatched machines which are "safe behind the firewall", and otherwise leapfrog from machine to machine on the way towards achieving the goals of the attacker.

Very often, the initial soft target is for a user or workstation which has no power. Restricted users and workstations are not very useful, so part of the attack lifecycle is usually to elevate the attackers' privileges to get more power. Power is the ability to

achieve the goals of the intrusion to begin with, so where is power concentrated in Windows Active Directory environments?

3) Power To Achieve Attackers' Goals

Powerful network devices and computers enforce security or provide access:

- Domain controllers (by far the most important).
- RADIUS servers.
- Proxy servers.
- Firewalls.
- IDS/IPS devices.
- VPN gateways.

Powerful users are typically members of groups like the following:

- Domain Admins.
- Enterprise Admins.
- Schema Admins.
- DnsAdmins.
- Cert Publishers
- Organizational Unit administrative accounts (OU Admins).
- Accounts for corporate executives and high-level managers.
- Accounts for wire transfers, online banking, purchasing, etc.
- Service accounts, e.g., the Exchange Site Services account.
- Accounts under which enterprise backup applications run.
- Any service account whose password is in the LSA Secrets.
- Accounts used for scheduled jobs that need elevated privileges.
- The local Administrator account on every machine.

And in the list of powerful devices above, we must include the workstations, laptops, tablets and phones of users who are members of any of these groups. If you own the laptop of the Domain Admin, you own the domain.

If you are a member of any of the above groups, especially Domain Admins, then whatever computer you are sitting at, and whatever computer you authenticate to over the network, becomes at that moment a high-value target too. We will see later that when you authenticate as a Domain Admin to another machine, you are giving a piece of yourself to that box and asking it to act as your agent, to impersonate you, to exercise your power for you to accomplish something. But if that computer has been previously compromised, what it's really doing is *waiting for you* to foolishly authenticate to it and loan it your power. The malware waiting on that computer will happily act as your agent too.

Example: Service Account In Administrators Group

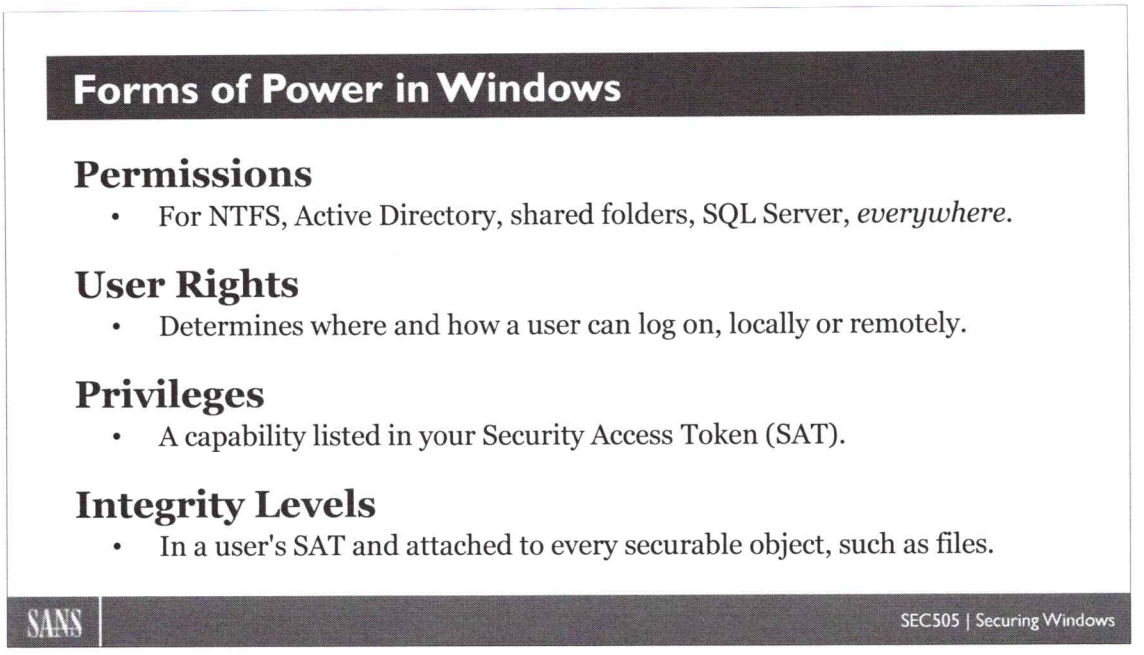
Imagine a background service which runs as a standard user account, instead of as Local System or Network Service. The password for that service's user account is typically stored in plaintext under a special set of registry keys called the "LSA Secrets", which refers to a component of the kernel called the Local Security Authority (LSA). The LSA Secrets are stored under HKLM\Security\Policy\Secrets. The permissions on these keys only allow access to Local System by default, but there are hacking tools and exploits which can be used to extract the data anyway.

Service accounts are very often members of the local Administrators group. How often do companies reset the passwords on service accounts? Sometimes only every few years when the service is upgraded to the new version, or sometimes never.

So, the problem is that if a soft target is compromised, like with a browser or PDF exploit after a bit of SE, it may have a service whose user account password is in the LSA Secrets, and this password is often identical across all/most of the other machines in LAN because they also have the service installed. (If this is a backup service or one used by the help desk, it's not unusual for nearly every workstation and server in the domain to have that service account.)

Now that the username and password have been compromised, the attackers use the compromised soft target as a platform for simply logging into (not really *hacking* into) any other desired machine in the LAN which also has that service.

Now, if you are unlikely to be targeted by such advanced or persistent adversaries, if your main concerns are run-of-the-mill spyware and malicious insiders, then you'll still need to do the brainstorming and design your defenses accordingly, even if your defenses won't need to be so expensive. Count yourself lucky! In this course we concentrate on the hard cases because defenses against *them* should cover the easy cases too.



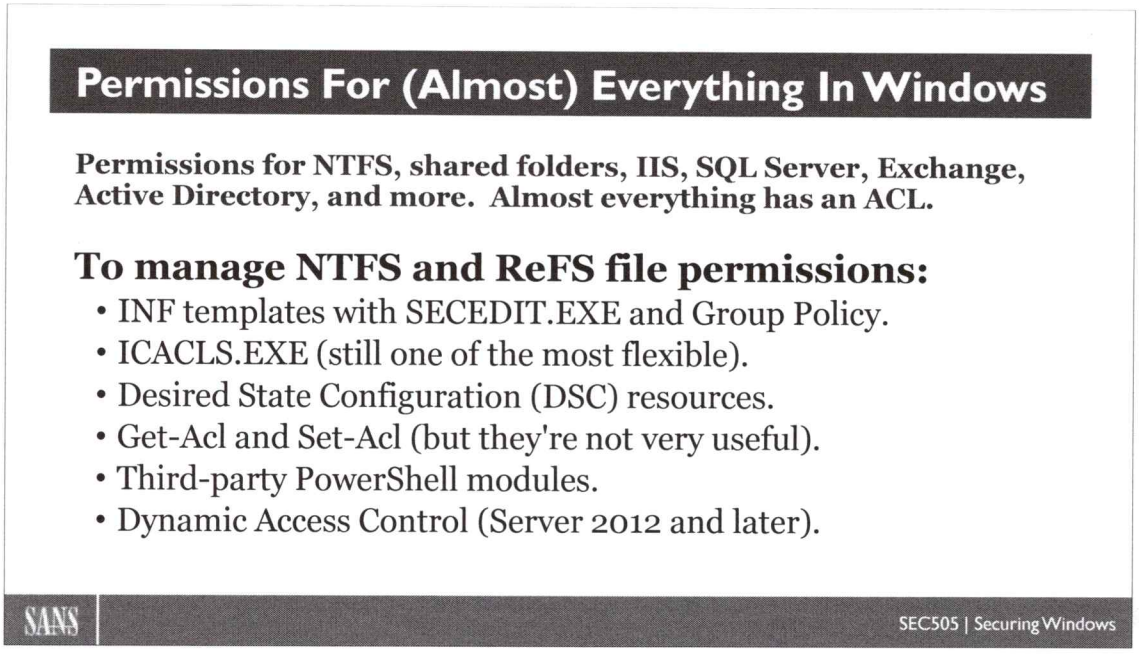
Forms of Power in Windows

The kernel of the Windows operating system uses various data structures and rules to decide who can do what. Every action occurs under the context of an identity or "principal", such as a user account, computer account, service identifier, or application identifier. This principal can be a member of various Windows groups or ASP.NET roles, and have various attributes in Active Directory which are being used as claims.

Through these group memberships and claims, a principal can be granted power. In Windows, virtually all forms of power are one or more combinations of the following:

- Permissions
- User Rights
- Privileges
- Integrity Levels

Let's discuss these forms of power in order to understand how malware and attackers abuse them, so that we may manage these powers more securely. There are more forms of power too, such as declared capabilities in Metro applications using the Windows Runtime (WinRT) API, but we need to start with foundational concepts first.



Permissions For (Almost) Everything In Windows

Almost everything in Windows has permissions. Permissions are also called "Access Control Lists (ACLs)". Unlike privileges, which express general capabilities not bound to any one particular thing, permissions are always assigned to particular objects. Permissions restrict how a "principal", e.g., a user or computer account, may interact with an object. Permissions are associated with the object, not with the principal. You have probably been managing permissions for years.

What types of objects have permissions in Windows? Here is a partial list:

- NTFS and ReFS files and folders.
- SMB shared folders.
- Active Directory organizational units, sites, users, printers, etc.
- SQL Server databases, tables, views, and rows.
- Exchange Server mailboxes.
- IIS virtual directories for HTTP and FTP.

But how can permissions be managed in an automated, scalable way?

INF Security Templates

An INF security template can include NTFS/ReFS permissions and audit settings, and may include nothing but NTFS/ReFS permissions, if desired. An easy way to (re)apply a complex set of NTFS/ReFS permissions is simply apply an INF security template with SECEDIT.EXE or Group Policy.

DSC Security Templates/Configurations

A resource for Desired State Configuration (DSC) can be designed to manage permissions of literally any type, not just NTFS permissions. The implementation details are hidden within the DSC resource module so that we only need to specify the principals and their permissions, usually as a simple string. Now, whether there is a resource module available for the product or object whose permissions you wish to manage, that is another question.

ICACLS.EXE

The best command-line tool for managing NTFS/ReFS permissions is still ICACLS.EXE. ICACLS can also manage ownership and integrity levels. A hardening script or DSC resource could use ICACLS to enforce permissions in a very flexible way. Unfortunately, this is an old text-oriented tool, not a PowerShell cmdlet (yet).

To see the command-line switches for ICACLS.EXE:

```
icalcs.exe /?
```

PowerShell

Unfortunately, even in WMF 5.0, PowerShell only has two inadequate cmdlets for managing NTFS/ReFS permissions: Get-Acl and Set-Acl. The cmdlets suffer from limitations which are annoying enough that you will likely prefer using ICACLS or an INF security template instead, if you are stuck with only using built-in tools.

Fortunately, there are free modules available for PowerShell which are much better for managing NTFS/ReFS permissions. Visit the PowerShell Gallery web site to find the latest versions (www.PowerShellGallery.com). One reason Microsoft has not developed a better set of cmdlets for permissions is that Microsoft expects most scripters to use DSC, but this is just not always the case.

It is possible to directly use the .NET Framework classes related to file system permissions; we do not have to use Get-Acl and Set-Acl. Here is an example.

To list the permissions on a folder in a human-readable format:

```
$folder = Get-Item -Path C:\Windows

$folder.GetAccessControl("Access").Access
```

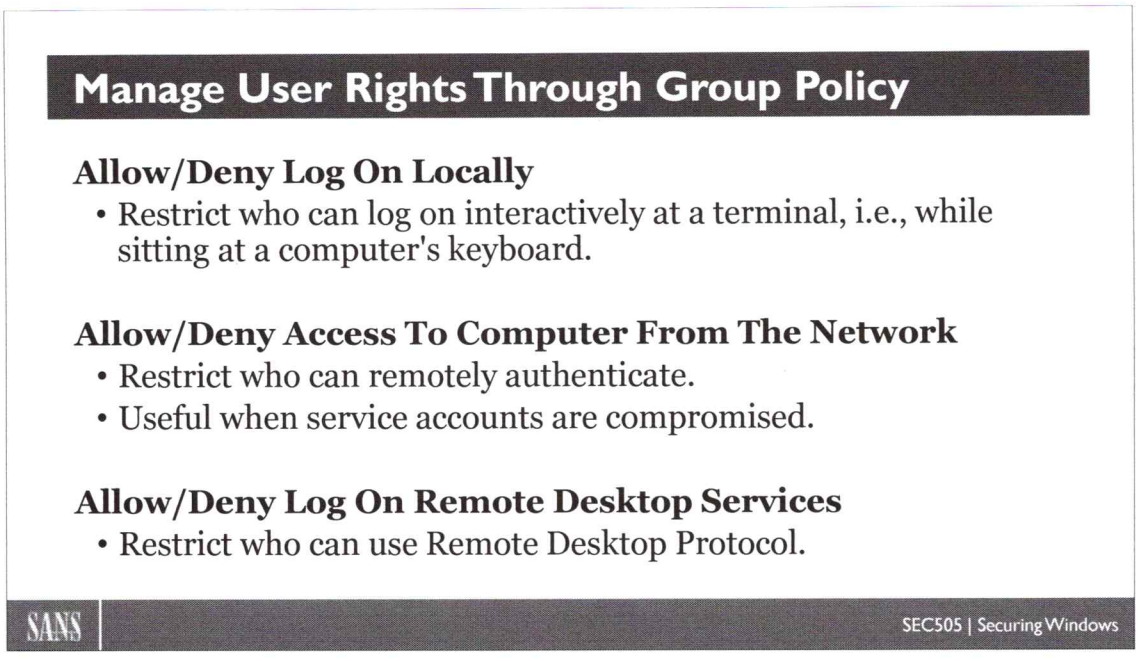
Getting the permissions is easy. Replacing a set of permissions entirely is somewhat easy. Scripting the editing of individual permissions is not so easy, and why you might be happier using DSC or a third-party module. (Microsoft, get on the ball here!)

For other products, such as IIS, Exchange and Azure, there may be several cmdlets for managing permissions, but this must researched on a case-by-case basis.

Dynamic Access Control (DAC)

Dynamic Access Control (DAC) was first introduced with Server 2012. It is an access control system based on the attributes of users, the attributes of the computers of these users, and the attributes of the resource the user is attempting to access, such as a file. The attributes of users and computers leveraged by DAC are called "claims", and DAC is a form of claims-based access control.

DAC is very flexible and scalable, including full PowerShell support, but it has complex requirements to set up and manage. DAC cannot be discussed here.



Manage User Rights Through Group Policy

You can manage user rights with Group Policy. A user right controls where a user is permitted to log on, either interactively or over the network. Unlike a privilege, user rights are not in a user's Security Access Token (SAT).

By regulating who can log on interactively at which machines, you lessen the odds of some machines being infected by negligent application use or USB flash drives; for example, at mission-critical computers you should only allow local logon to the bare minimum of trusted personnel.

By regulating who can log on over the network to a system, you reduce the odds of malware spreading over the network to it since some worms require successful authentication to a target before the target can be compromised; for example, if you have a thousand workstations in an organizational unit where it's known that the users there never need to log on over the network to each other's' workstations, then you might only permit over-the-network authentication to those workstations from Domain Admins, Help Desk, Backup Agents and other similar groups. One way Conficker can spread, for example, is by guessing passwords to log onto remote systems over SMB.

You have the affirmative "Allow *" user rights, which can be used to forbid logons by exclusion to anyone not granted the right explicitly, and then there are the corresponding negative "Deny *" rights, which can be used to define exceptions to a general Allow, since an assigned Deny right will override an Allow. Authenticated Users might be granted, for example, the "Allow Access To This Computer From The Network" right at all the servers in an OU, but then you could add a "Deny Access To This Computer From The Network" right at those machines for just the Contractors group, since they often

bring infected laptops into the LAN and perhaps there's no need for them to access any of the machines in that OU anyway.

More creative uses for rights are made scalable by Group Policy distribution. An OU of computers can be functionally isolated, for example, by giving just one non-administrative group the "Log on locally" and "Access this computer from the network" rights. In this case, only Domain Admins and that one group could access the resources on those machines either locally or remotely.

Note that IPSec computer authentication also requires that the remote computer account have the "Access this computer from the network" right too, hence, you can also limit which machines can access a server using IPSec and this GPO option.

Alternatively, all the computers in a site might be accessible to Authenticated Users, except for one untrusted group; that group would have the new "Deny access to this computer from the network" right assigned to it on all the computers in the site. Perhaps that group would be named Untrusted Users.

When a service runs under the context of a user account instead of Local System or Network Service, the password for that service's user account is stored in the registry in plaintext. Now, it is stored under a set of keys called the "LSA Secrets", and these keys have very restrictive permissions, but if malware/hackers can manage to execute commands as Local System or as an administrator, the password for the service account can still be extracted. This is a problem because that same service (with the same account and password) might be found on many/all computers in the LAN. But if that service account only needs to log on locally, then it's over-the-network logon rights can be denied, thus limiting the scope of harm caused from the first compromised box.

Security Settings > Local Policies > User Rights Assignment

Some groups are powerful because of the special user rights they have. But with the User Rights Assignment container under Local Policies, you can customize the distribution of user rights on machines any way you like. You can specify exactly which users and groups should have each right in each OU. Just as before, if some user/group has a certain right when your GPO does not include them, that right will be taken away from them the next time they log on.

The following is a complete list of the user rights, with the interesting ones are in bold:

- **Access this computer from the network**
- Allow log on locally
- **Allow log on through Remote Desktop Services**
- **Deny access to this computer from the network**
- Deny log on as a batch job
- Deny log on as a service
- **Deny log on locally**
- **Deny log on through Remote Desktop Services**

- Log on as a batch job
- Log on as a service

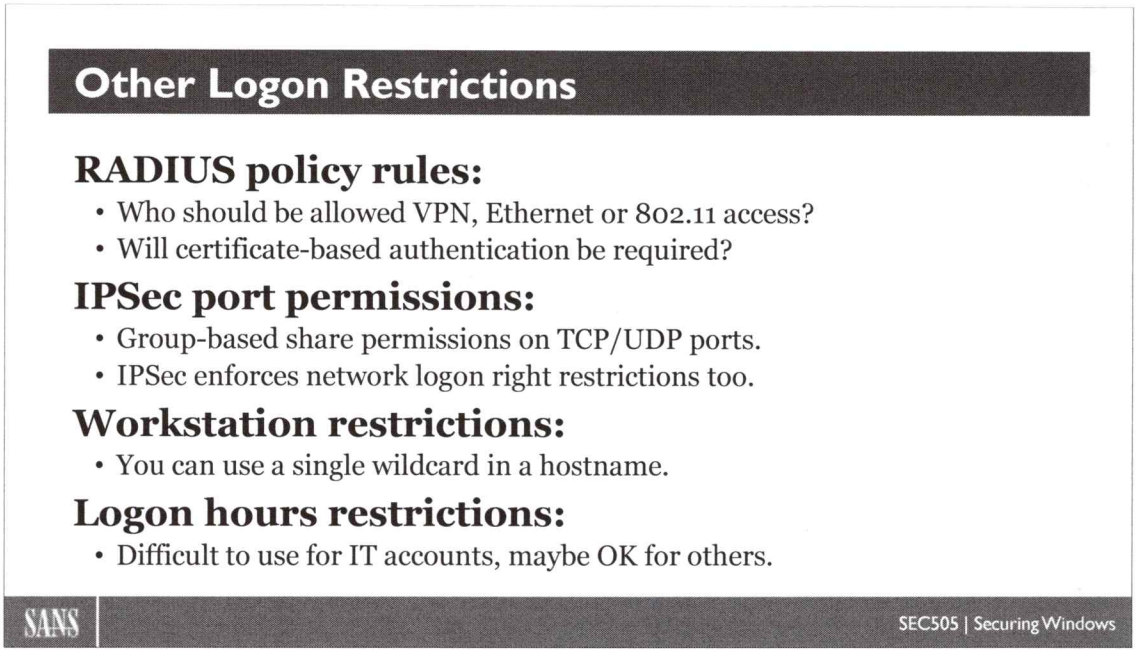
Note that the Administrators group must be granted the "Allow log on locally" right and cannot be assigned the "Deny log on locally" right.

Also, for a user to use Remote Desktop Protocol (RDP) to remote into another system, that user requires both the "Allow log on through Remote Desktop Services" and the "Access this computer from the network" logon rights, not just one or the other.

Logon User Rights for Local Accounts

In Windows 8.1, Server 2012-R2 and later operating systems, there are two new local well-known SID identities named "Local Account" and "Local Account and Member of the Administrators Group". These are not real groups whose membership you can manage, but they do represent all local user accounts and local user accounts who are in the Administrators group.

These new identities are very useful because they can be granted or denied logon rights. Even though the Administrators group itself must be allowed to log on locally, that doesn't mean that *local* user accounts in the Administrators group must be so allowed; hence, it's possible to only allow *global* user accounts in the Administrators group to log on locally or over the network.



Other Logon Restrictions

Besides formal user rights, there are other similar logon restrictions, some of which are integrated into the user rights configuration settings.

Remote Access and Wireless Policies

When security is paramount, it is best if all target accounts and groups are denied any form of remote access privileges whatsoever (dial-up, wireless or VPN). However, it is precisely the administrators who may wish to use remote access to manage the network, so keep in mind that it *is* possible to require a smart card or other multi-factor device for remote access authentication. In general, require the strongest available remote access encryption and authentication when high-value targets connect, but Enterprise Admins and Schema Admins accounts should always be denied remote access.

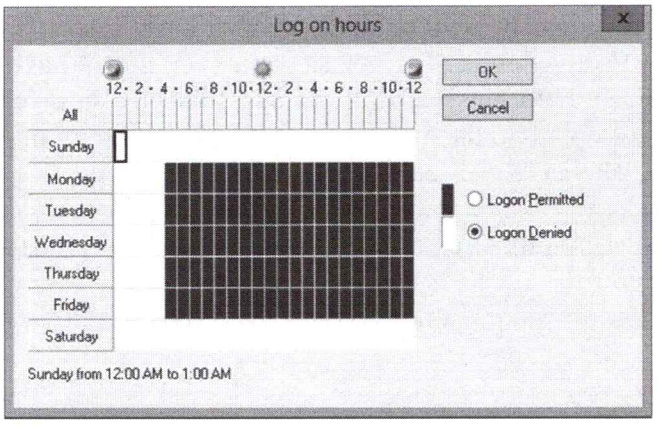
IPSec Port Permissions and Network Logon Restrictions

The workstations of target users should have host-based firewalls and require IPSec to restrict inbound access to only those TCP/UDP ports which are needed. But not only can IPSec as such be required to access these ports, you can also limit access to the ports based on the group memberships of the user connecting to the ports. Just as you can have share permissions to a folder or printer based on group memberships, so you can use these same group memberships to restrict access to TCP/UDP ports. But if a remote user does not have the "Allow access to the computer from the network" user right, the IPSec driver will enforce this restriction and deny all IPSec connections whatsoever. IPSec and the Windows Firewall are discussed elsewhere this week.

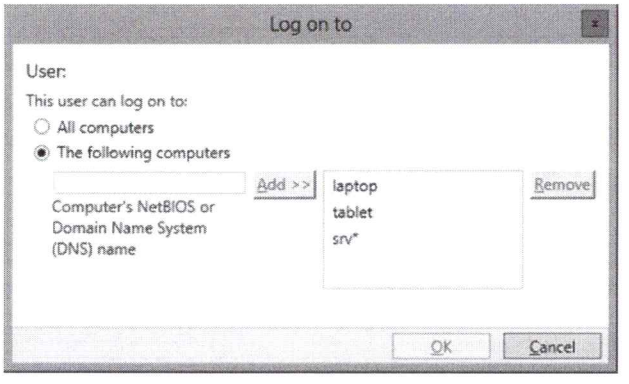
And don't forget that IPSec computer authentication also requires that the remote computer account have the "Access this computer from the network" right too, hence, you can also limit which machines can access a server using IPSec and this right.

Restrict Logon Hours and Workstations

When possible, restrict the days and hours when target users are permitted interactive logons, e.g., perhaps only during business hours, Monday through Friday. This won't be possible for network administrators, but for other targets, maybe so.



Also restrict as narrowly as possible the list of computers where this logon may occur, and note that a single wildcard asterisk may be used when defining workstation restrictions. These options are both configured on the Account area of the user's property sheet in AD.



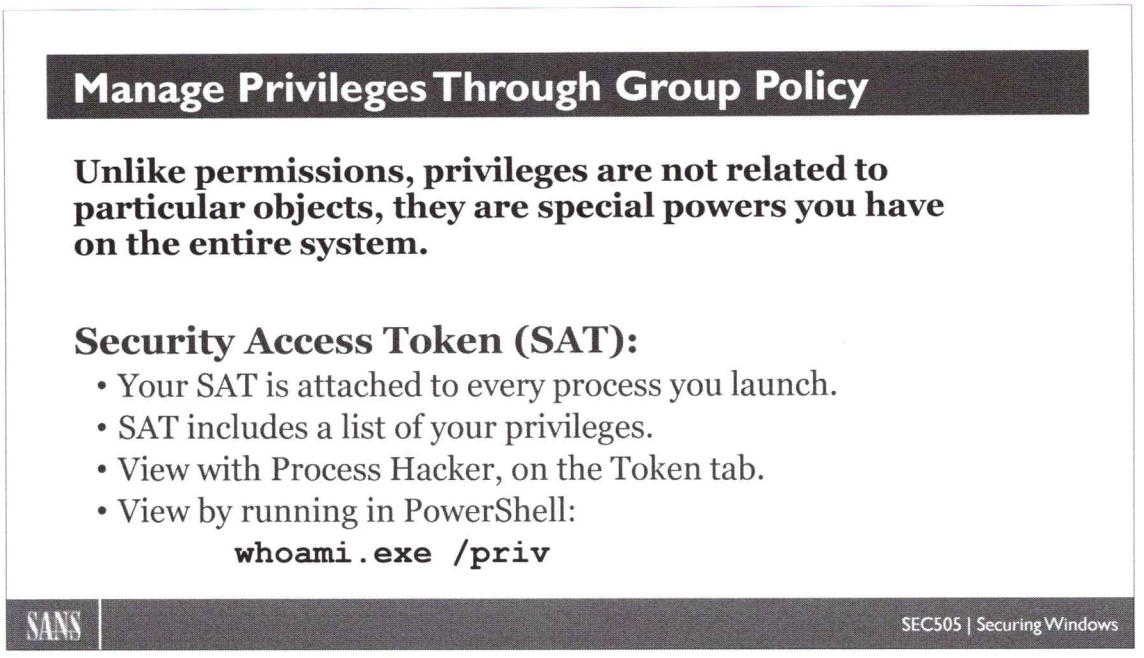
These restrictions may be impractical for OU Admins, but all Enterprise, Schema and Domain Admins should be restricted as tightly as possible. Indeed, for these most-targeted of accounts, even RDP logons should be disallowed if possible (Remote Desktop Services Profile area on property sheet) and certainly disable remote control (Remote Control area). Remember, Domain Admins are IT managers, they are not the people who actually implement and troubleshoot servers. Delegation of authority at the OU level is discussed elsewhere.

Require Smart Card Authentication

A "smart card" is the size and shape of a credit card, but it has a cryptographic CPU and some EEPROM memory embedded in it. There are also "smart tokens" which typically have a USB interface, but some are wireless or use a proprietary reader. Windows 8 and later also support using a TPM chip to implement a virtual smart card. Smart devices carry the user's public key certificate and private key. They enforce two-factor authentication because the user must 1) have the device and 2) know the PIN number to access it. If the device is lost or stolen, its digital certificate can be revoked.

Smart card authentication can be required on a per-user basis, hence, consider requiring a smart card for the interactive logon of target accounts (you do not have to put target accounts into their own OU for this). To force a user to have a smart card or token when logging onto their desktop, go to the Properties of that user account in AD > Account area > check the box labeled "Smart card is required for interactive logon".

PKI is discussed elsewhere this week and we'll see how to deploy smart cards.



Manage Privileges Through Group Policy

A user right controls where and how you may log on, while a privilege is a special power you might possess after you log on. Permissions are attached to particular objects, while privileges are not attached to the objects you access. Privileges are included in the Security Access Token (SAT) linked to your processes, while user rights and your NTFS permissions are not in your SAT.

To see what privileges the PowerShell or CMD shell process has:

```
whoami.exe /priv
```

In the Process Hacker tool, you can see the SAT associated with each process by double-clicking any process and examining the Token tab.

Group Policy Management

Privileges can be controlled through Group Policy. In a GPO they are located under Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > User Rights Assignment. Both rights and privileges are mixed here together unfortunately.

Some groups are powerful because of the special privileges they have, not just because of the permissions granted to those groups. The local Administrators group, for example, is especially powerful because of all the privileges granted to it by default. But you can customize the distribution of user rights on machines almost any way you wish. You can specify exactly which users and groups should have each right in each OU. And if there

is a privilege you don't want the Administrators group to possess on some of your machines, such as the Debug Programs privilege, this can be done through Group Policy.

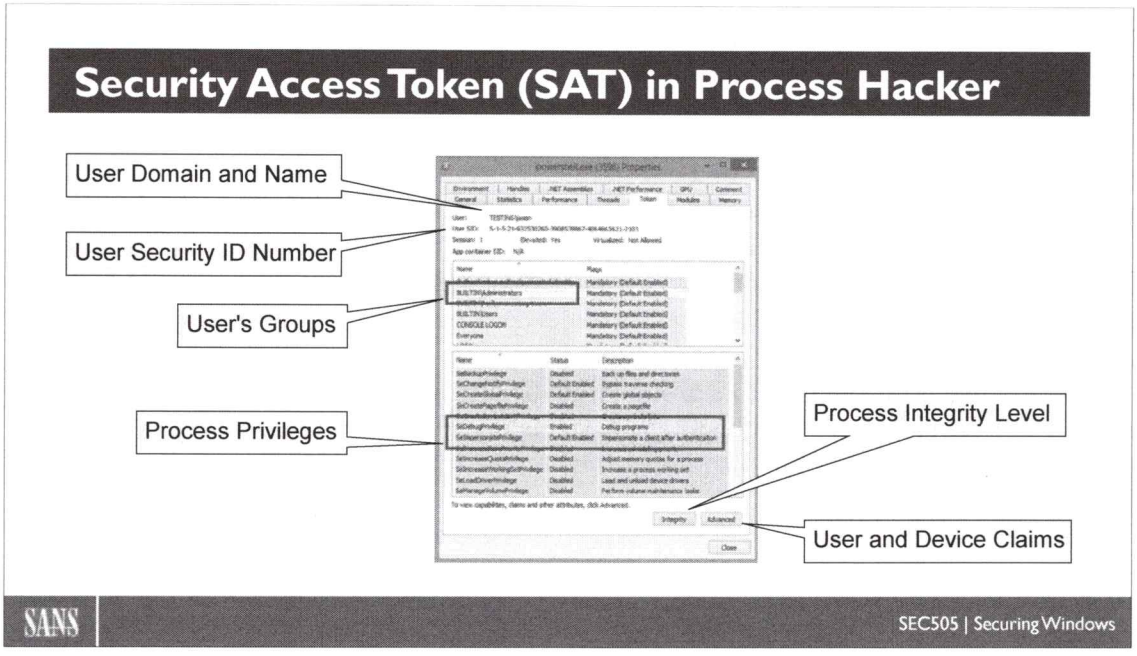
The following is a complete list of the available privileges with some interesting ones highlighted in bold (note that the user rights are not included):

- Access Credential Manager as a trusted caller
- **Act as part of the operating system**
- Add workstations to domain
- Adjust memory quotas for a process
- Back up files and directories
- Bypass traverse checking
- Change the system time
- Change the time zone
- Create a pagefile
- **Create a token object**
- Create global objects
- Create permanent shared objects
- Create symbolic links
- **Debug programs (this is the most dangerous privilege)**
- Enable computer and user accounts to be trusted for delegation
- Force shutdown from a remote system
- Generate security audits
- **Impersonate a client after authentication**
- Increase a process working set
- Increase scheduling priority
- **Load and unload device drivers**
- Lock pages in memory
- Log on as a batch job
- Log on as a service
- Manage auditing and security log
- **Modify an object label**
- Modify firmware environment values
- Perform volume maintenance tasks
- Profile single process
- Profile system performance
- Remove computer from docking station
- Replace a process level token
- **Restore files and directories**
- Shut down the system
- Synchronize directory service data
- **Take ownership of files or other objects**

How could these privileges be modified in a useful way?

For example, you might create a Restore Operators group and give it the "Restore files and directories right" and then remove this right from the Backup Operators group. This will separate powerful privileges into two mutually exclusive groups of non-administrative users. Domain Admins will still have both rights because they are typically the only ones who should be restoring DCs and other critical servers. Now, users who are somewhat less trusted can be put into the Backup Operators group. In the same way, each major OU could have its own separate pair of custom Restore/Backup Operators groups, and these groups would have their rights only on the machines in their own OUs.

Or consider that an IDS-Operators group might be created and given the "Manage auditing and security log" right on all the systems in a domain or OU. That group could also be put into the local Administrators group on the IDS sensors and management stations themselves.



Security Access Token (SAT) In Process Hacker

The slide above shows an example of a Security Access Token (SAT) as displayed by the free Process Hacker tool (http://processhacker.sourceforge.net).

In Process Hacker, double-click any process and examine the Token tab to see the SAT for that process.

To see the full SAT of the PowerShell or CMD shell process itself:

```
whoami.exe /all /fo list
```

On Your Computer

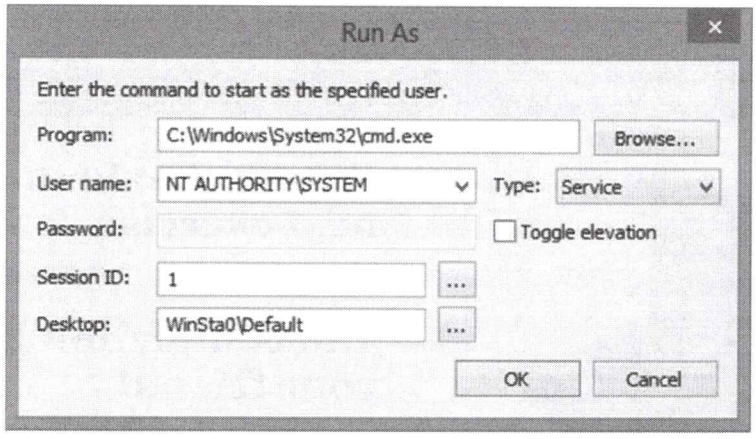
The free Process Hacker tool was installed by the SEC505 setup script at the beginning of the course. You can find the icon to launch Process Hacker on your desktop or by searching in the Start screen. If it is not installed, you can find the setup program for it in C:\SANS\Tools\ProcessHacker (or get it from http://processhacker.sourceforge.net).

Right-click the Process Hacker icon and run it as administrator.

In Process Hacker, double-click any running process in the list and examine the Token tab. This is the Security Access Token (SAT) for that process. The slide on the prior page identifies the different areas of information in the SAT. Close tab.

In Process Hacker, pull down the Hacker menu > Run As > enter the following, then OK:

- Program: C:\Windows\System32\cmd.exe
- User Name: NT AUTHORITY\SYSTEM (use pull-down menu)
- Type: Service
- Session ID: 2 (or 1 or 3, if a CMD shell does not pop up)
- Desktop: WinSta0\Default



Note: If Process Hacker appears to hang, terminate it with Task Manager, shut down the VM, and edit the settings of the VM so that it uses two CPUs (or two CPU cores). This seems to help on many machines. Ask the instructor if you cannot find the configuration settings for your VM.

This will launch a CMD shell running as the local System identity. Any command executed from within this CMD shell will execute with the SAT and privileges of System, i.e., the kernel of the operating system.

Confirm the identity and SAT of the new CMD process (hit spacebar to advance):

```
whoami.exe /all /fo list | more
```

Launch RegEdit as System in the CMD shell:

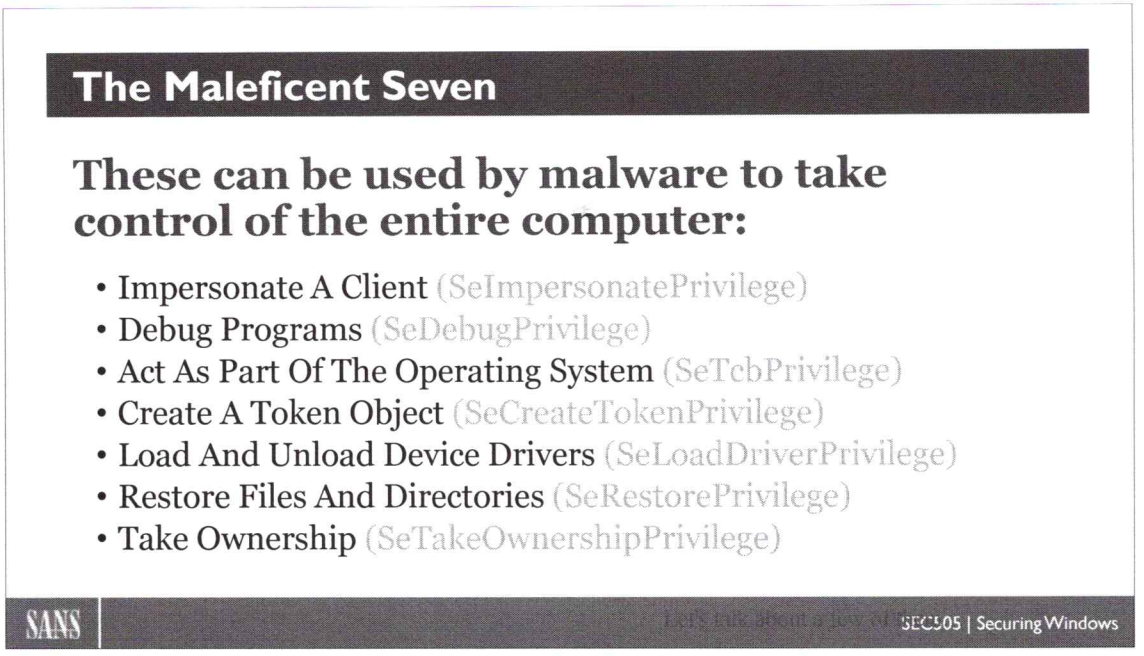
```
regedit.exe
```

Because RegEdit is running as System, you can access the local Security Accounts Manager (SAM) database in the registry; for example, in RegEdit you can browse to HKEY_LOCAL_MACHINE\SAM\SAM\Domains and see all the subkeys and values underneath. This is where local group memberships are stored as well as the password hashes of local user accounts.

In Process Hacker, find the regedit.exe process > double-click > see the Token tab to confirm that RegEdit is indeed running as System. Notice all the privileges listed in the SAT, including "Act as part of the operating system" (SeTcbPrivilege). Yikes! Close tab.

Close RegEdit.

Please leave the CMD shell running.



The Maleficent Seven

Of all the privileges, there are a few which can be used to take over the computer where one is sitting, or even perhaps to take over the entire domain if a Domain Admin gets infected with malware. Using Group Policy to limit who has these dangerous privileges is imperative. These are the maleficent seven privileges, with the display name shown first, followed by the internal name shown in parentheses:

- **Act As Part Of The Operating System (SeTcbPrivilege):** Malware could use this privilege to create a new logon session with a known username and password (such as the victim's account) but that session could be created with any arbitrary group memberships for the session's Security Access Token (SAT), hence, commands could be executed under the context of local Administrators, Domain Admins, Enterprise Admins, etc.

- **Create A Token Object (SeCreateTokenPrivilege):** Malware could use this privilege to execute a command under the context of a new fabricated Security Access Token (SAT) which can contain any arbitrary group memberships or other privileges. The SAT could be created with apparent memberships in local Administrators, Domain Admins, Enterprise Admins, etc.

- **Debug Programs (SeDebugPrivilege):** Malware could use this privilege to bypass the permissions on any running process, inject a malicious DLL into that process, and launch a new thread within the process to execute code from the injected DLL. This technique could be used to install a rootkit, open backdoor TCP ports, dump password hashes, execute commands as Local System, etc. This is the most dangerous privilege because it is so often used and abused.

- **Load And Unload Device Drivers (SeLoadDriverPrivilege):** Malware could use this privilege to install a Plug-and-Play device driver, then the driver, which is a binary with executable code like any other binary, could execute malicious commands under Local System context.

- **Restore Files And Directories (SeRestorePrivilege):** Malware could use this privilege to bypass NTFS permissions and replace any file, including operating system and application binaries, with the attacker's own modified files. When the files are later executed, perhaps by the operating system itself or by a more powerful user than the victim, the malware's code will also be run. (Incidentally, the corresponding "Backup files and directories" privilege, SeBackupPrivilege, can be used to bypass NTFS permissions to read any file too.)

- **Take Ownership (SeTakeOwnershipPrivilege):** Malware could use this to change the permissions on an object to its advantage and then read/modify/replace/delete that object. Objects at risk include files, folders, processes, threads, registry keys, printers, or anything else with an ACL. The only exception is when permissions are explicitly granted to the Owner Rights group, whose permissions take precedence over the do-anything default for object owners.

- **Impersonate A Client After Authentication (SeImpersonatePrivilege):** If a Domain Admin uses RDP to manage a server that's infected with malware which has this privilege, the malware can seize the Domain Admin's delegation Security Access Token (SAT) to execute commands on other remote machines with full Domain Admin powers. The attacks works not just with RDP, but with any local or over-the-network authentication which is considered an interactive logon by the operating system. Access to the Domain Admin's cached credentials or password hash is not necessary in this case.

Let's talk about just a few of these in more detail…

Primary Security Access Token (SAT):

- The true, original underlying identity of a process.
- `whoami.exe /all /fo list`

Impersonation Tokens:

- Used by network services to impersonate clients.
- Regular impersonation SAT (local resources only).
- Delegate impersonation SAT (local and remote).

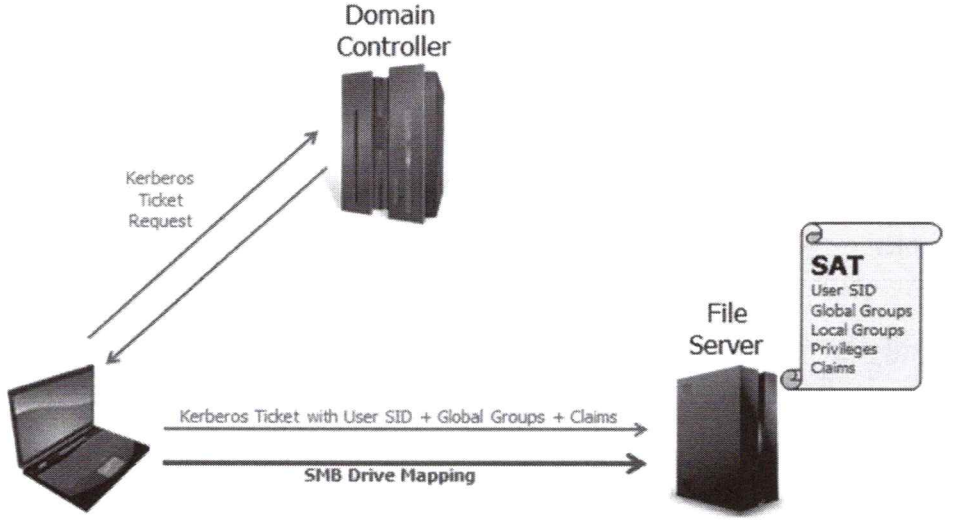
Token Stealing:

- Hijack delegation SATs for network authentications.

SEC505 | Securing Windows

Impersonate A Client Privilege

Every process has an associated Security Access Token (SAT) which represents the identity standing behind that process. A SAT includes the Security ID (SID) number of a user or computer account (or a well-known identity that's built into the operating system, such as Network Service), the SID numbers of the account's global and local group memberships, the privileges of the account on the local system, and the claims from that account's attributes in AD. The SAT of a process determines what that process can do.



Without an associated SAT, the operating system wouldn't know what permissions or privileges apply to the process. This SAT is called the "primary token" of the process, i.e., its true underlying identity.

The global group SIDs and claims information inside a SAT come from Active Directory, usually through the Kerberos protocol. The local group SIDs in the SAT come from the Security Accounts Manager (SAM) database on the local or remote computer where the user requested access. Similarly, the user's privileges in the SAT come from the local security policy store (the registry) on the local or remote computer where the user requested access. SATs are not transmitted over the network, they are constructed on-the-fly on each computer where the user authenticates. On a stand-alone computer, there is no information about global groups or claims in any SATs because stand-alones do not use Active Directory. The SAT itself is just a data structure in kernel memory space, but it is a kernel object carefully constructed and managed by the OS, not by the user.

Impersonation SAT

Some processes are network services, like the Server service for granting SMB access to shared folders. When a user authenticates over the network to a service, the operating system on the server usually constructs a SAT to represent the remote user and then gives that SAT to the listening service. This is how it works with SMB and the Server service. As the user sends requests and commands over the network to the server, the service uses the SAT representing the user to act on behalf of that user. The service acts as that user's agent or proxy. The SAT for the remote user is called an "impersonation token" because the service uses it to temporarily impersonate the user while carrying out the user's requests and commands.

Delegation SAT

But how far does the impersonation go? With a standard impersonation SAT, the network service can only use the SAT to access local resources on that server. The level of impersonation extends no further than the local file system, registry, database, or other resources directly controlled by the server. Most of the time, when you authenticate to a server over the network, a local-only impersonation SAT is created. With some tools, though, you can explicitly tell the remote server how far you want the impersonation to reach.

For example, to use PowerShell to authenticate to a remote machine for a WMI query, you can explicitly request normal (local-only) impersonation, i.e., no delegation:

```
Get-WmiObject Win32_OperatingSystem -Impersonation Impersonate
      -ComputerName controller.testing.local
```

But what if you are accessing a service which needs to impersonate you while accessing other computers on the network? By analogy, what if you authorized your attorney to not only read your bank statements sitting on your desk (local resources), but also to go from one bank to another and sign contracts *for* you (remote resources) as your representative? This is possible in Windows, it is by design. If the impersonation SAT created to

represent you on the server is instead a full "delegation token", that delegation SAT can be used not only to access local resources on the server, but also other resources on other servers over the network. For example, you might authenticate to an IIS web application which could use a delegation SAT to query and reconfigure other SQL Servers over the network as you.

For example, to use PowerShell to authenticate to a remote machine for a WMI query, you can explicitly request full delegate impersonation:

```
Get-WmiObject Win32_OperatingSystem -Impersonation Delegate
      -ComputerName controller.testing.local
```

Can just any process get an impersonation or delegation token and take on other users' identities? No, wearing another person's SAT like a mask requires a special privilege.

Impersonate A Client After Authentication

To get a handle to an impersonation SAT, the primary SAT of a process must have a special privilege named "Impersonate a client after authentication." This privilege is granted by default to identities like Local Service, Network Service, and, you guessed it, to the local Administrators group. In fact, you cannot take the privilege away from Administrators even if you try.

Token Stealing

If malware is running with the Impersonate A Client privilege, then that malware can execute commands with the identity of any impersonation SAT that happens to exist on the computer at that moment. To do so, the malware must have already fully taken over the machine (it's probably already running as Administrators or as Local System), so a local-only impersonation SAT wouldn't be very interesting to steal.

But a delegation SAT, on the other hand, could be very useful if it were owned by a Domain Admin. If malware on a computer could steal a delegation token of a high-value user, like a network administrator or the CEO of a corporation, then this could expand the power of that malware dramatically.

Importantly, keep in mind that stealing the SAT of another user on a computer is not a technique for initially compromising or getting into a machine, it is post-exploitation technique to raise the attacker's privileges. The attacker must use some other trick, like an infected e-mail attachment or a web page with malicious JavaScript, to gain initial control of a victim box.

For example, there is a free tool named Incognito which can be used to list available SATs and execute commands by hijacking them, assuming that one is already running as Local System (http://sourceforge.net/projects/incognito/). There are many other tools which can do the same thing, and the techniques used by Incognito are built into some malware.

> **Note:** Microsoft is aware of these tools and techniques, so a new patch or new operating system might block these tools and techniques temporarily (and maybe someday permanently, but don't hold your breath).

In the listing below, we can see some of the output of running Incognito in a CMD.EXE shell (don't be surprised if there is a pop-up error message, it's expected).

```
C:\> incognito.exe list_tokens -u

[*] Enumerating tokens
[*] Listing unique users found

Delegation Tokens Available
============================================
NT AUTHORITY\IUSR
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
SANS\Administrator
Window Manager\DWM-1
Window Manager\DWM-2
...
```

If Incognito is running as the local System identity, here is an example of how easy it is to steal the delegation token of SANS\Administrator (seen in the output above) in order to execute another command, specifically a command to add the Guest account to the local Administrators group:

```
incognito.exe execute SANS\Administrator "net.exe localgroup
administrators guest /add"
```

How does your delegation SAT get created on a machine? The most common way is by logging on interactively, namely, by sitting at the computer's keyboard and monitor. An interactive logon, as opposed to a network logon, creates a SAT with full delegation capabilities. Also, when you RDP into another computer, this is too is considered an interactive logon by the OS because of the desktop created for you. Many tools, such as the PowerShell example above, can log into a remote computer by specifying delegate impersonation, and some do it by default. The RUNAS.EXE tool creates a new delegation SAT for the process launched, and so does PSEXEC.EXE with the -U switch.

Hackers could install malware on a machine that just patiently waits, month after month, until the day when a Domain Admin logs on interactively at the machine (like with RDP), and then the malware springs into action, stealing the admin's delegation SAT to execute malicious commands on your domain controllers and other high-value servers. This event will be the first bad day in long string of bad days.

On Your Computer

This lab has multiple parts.

Incognito

Switch to the C:\Temp folder in the CMD shell running as System or as Administrator:

```
cd C:\Temp
```

> **Note:** If you have closed this CMD shell, please just right-click on the icon for CMD in the Start screen and select "Run as Administrator".

Use Incognito to list available Security Access Tokens (SATs) in memory:

```
incognito.exe list_tokens -u
```

> **Note:** In the following command, hit Enter twice if Incognito appears to hang.

Use Incognito to steal the delegation SAT of the Administrator to run a command:

```
incognito.exe execute -c TESTING\Administrator "net.exe
 localgroup administrators guest /add"
```

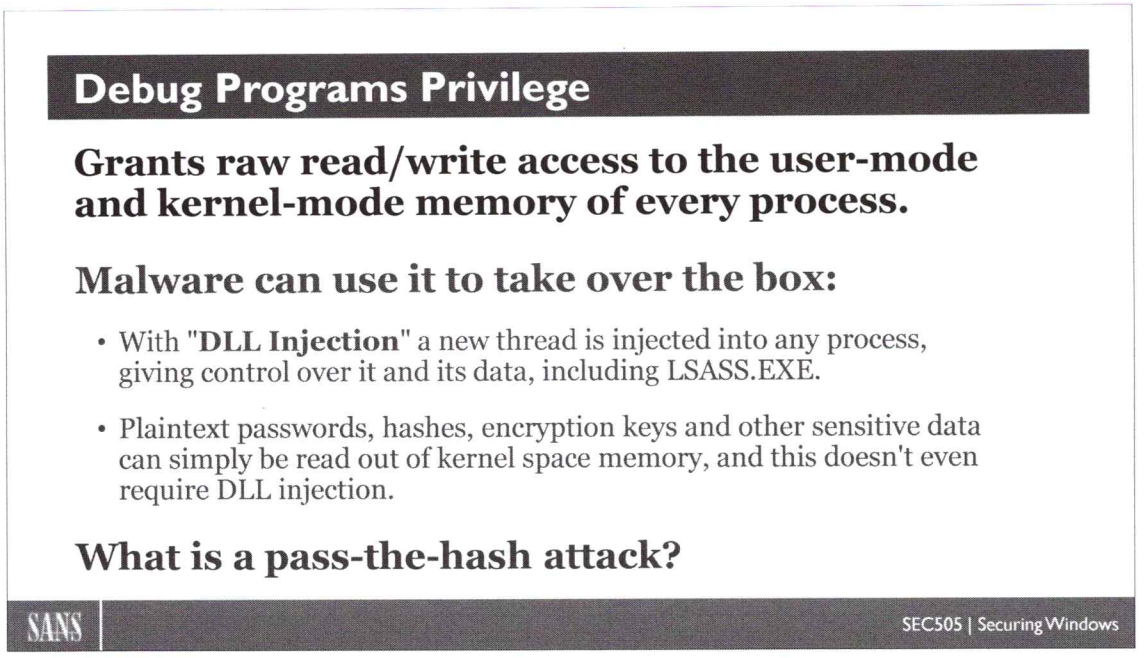
> **Note:** The above command assumes your domain is named "TESTING" and you are logged on as Administrator. If not, review the list of tokens from the prior commands and adjust. If Incognito appears to hang, hit Enter twice or Ctrl-C.

Confirm that Guest was added to the Administrators group:

```
net.exe localgroup administrators
```

Now remove the Guest account from the Administrators group:

```
net.exe localgroup administrators guest /delete
```



Debug Programs Privilege

A "debugger" is an application that allows one to examine and control a running program or operating system in order to troubleshoot or reverse engineer it. OllyDbg, for example, is a popular and free debugger for Windows (http://www.ollydbg.de). The problem is not users running debuggers, though, the problem is malware using this privilege to take over the box.

The *Debug Programs* privilege permits a user to attach a debugger to any process, even if that user did not launch the process being debugged. This is highly dangerous because a process might contain sensitive information in cleartext or the integrity of the process could be undermined. For example, a "DLL Injection" attack requires the *Debug Programs* privilege in order to work because this attack modifies a running process by injecting a new thread into the address space of the target process. The Cain tool (www.oxid.it) uses this trick to dump password hashes, for example, but it only works if Cain is running as someone with the *Debug Programs* privilege. Similarly, pass-the-hash tools require the *Debug* privilege to read password hashes out of kernel space memory.

Administrators Group

By default, only the local Administrators group has the *Debug Programs* privilege. Unfortunately, many regular users have been added to the local Administrators group on their workstations, and almost none of them (except for developers) actually need it. If a user's machine gets infected and the malware runs in a process with the user's SAT, such as their browser or e-mail program, the malware could use the *Debug Programs* privilege to intercept keystrokes, steal passwords, install a rootkit, open listening ports for a backdoor and configure the firewall to allow access to it, delete files,

etc. This privilege is widely abused by malware and hackers, and rarely used by anyone other than developers.

Hence, limit who has this right and audit changes to whom it has been granted. Using Group Policy, you should even test removing the privilege from the Administrators group on standard workstations (the exceptions might be for developers and security personnel) so that if you cannot get users out of the Administrators group at least you'll have stripped this privilege away from it. Most likely, your users won't even notice.

What Is A Pass-The-Hash Attack?

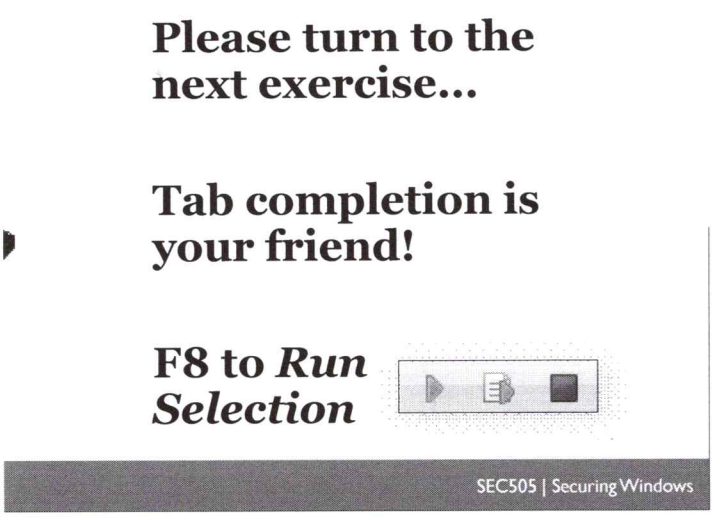
In a pass-the-hash attack, a hacker or piece of malware does not need a user's plaintext password in order to authenticate as that user. Simply being in possession of the hash of the user's password is sufficient for log on over the network or to launch a new process locally under the context of the user whose hash had been stolen.

Password hashes can be extracted from network traces with weaker protocols, like NTLMv1, but they can also be extracted directly from the hard drive or memory of compromised hosts. When a computer is compromised by an attacker or malware which has the Debug Programs privilege, such as under the context of Local System or a member of the Administrators group, it is possible to extract password hashes for local accounts from the SAM database, password hashes for interactive users with accounts in Active Directory, services running as a user account, User Account Control (UAC) and RUNAS.EXE authentications, and Remote Desktop Protocol (RDP) connections over the network when the user only "disconnects" the RDP session instead of fully logging off.

Once in possession of the hash, though, it can be used for local process launch and for network authentication to other systems. And while disabling NTLM and only using Kerberos will help, there are similar exploits for Kerberos (pass-the-ticket attacks).

Pass-the-hash attack tools do not require expert level hacking skills in order to run them. You can find YouTube videos and other tutorials demonstrating how to do it with free and popular tools like:

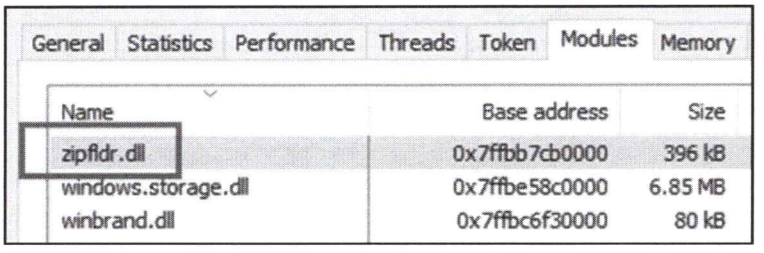
- Windows Credentials Editor (www.ampliasecurity.com)
- Metasploit (www.metasploit.com)
- Runhash (www.truesec.com)



On Your Computer

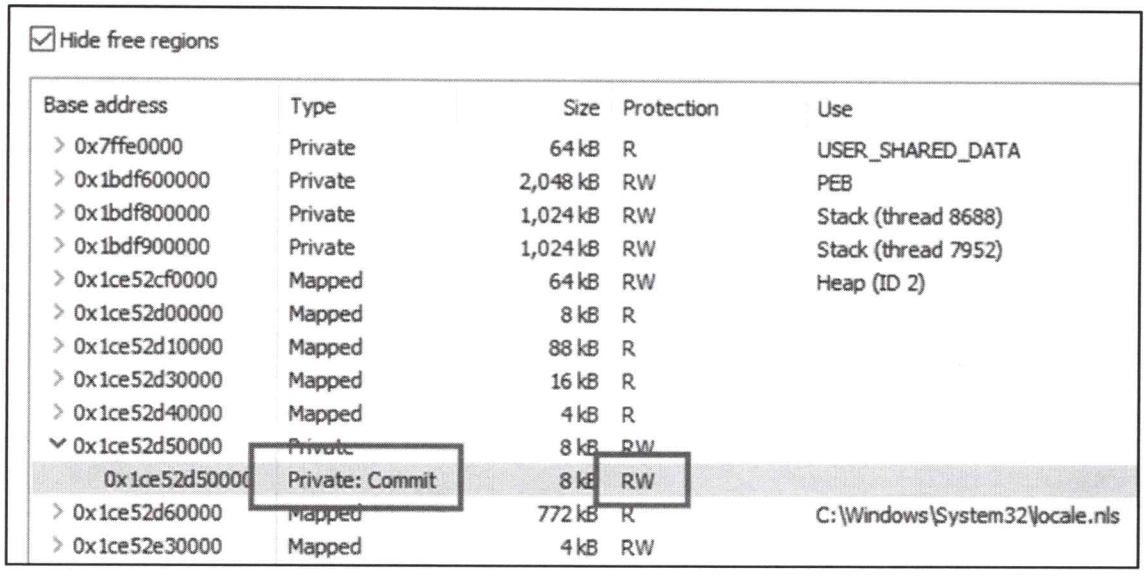
In Process Hacker, find your CMD.EXE process > right-click > Miscellaneous > Inject DLL > select C:\Windows\System32\zipfldr.dll > Open.

In Process Hacker, double-click your CMD.EXE process > Modules tab > scroll down the list of loaded modules to confirm that zipfldr.dll is now a part of the process (should be at the bottom of the list if you sort on the Name column).

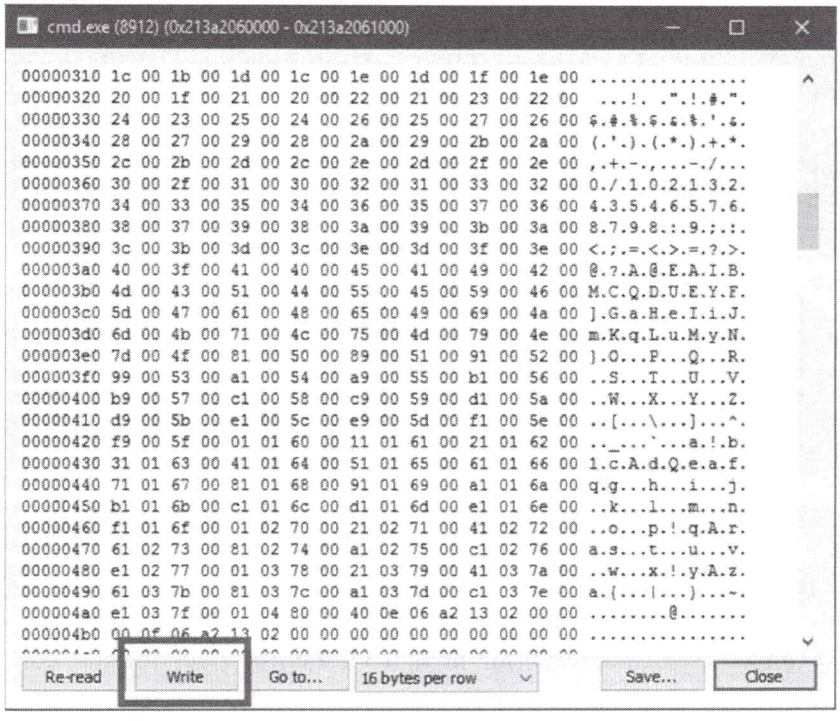


> **Note:** Imagine that the DLL we injected was malware. It could now execute code with the SAT of its host process, namely, as System! The malware could dump password hashes, steal encryption keys from memory, delete files, etc.

In Process Hacker, go to the Memory tab in the properties of your CMD.EXE process > expand down any area of memory with a type of "Private", look for a region that says "Private: Commit" and has a Protection of "RW", then double-click it. (If you get an error message, just try another region.)



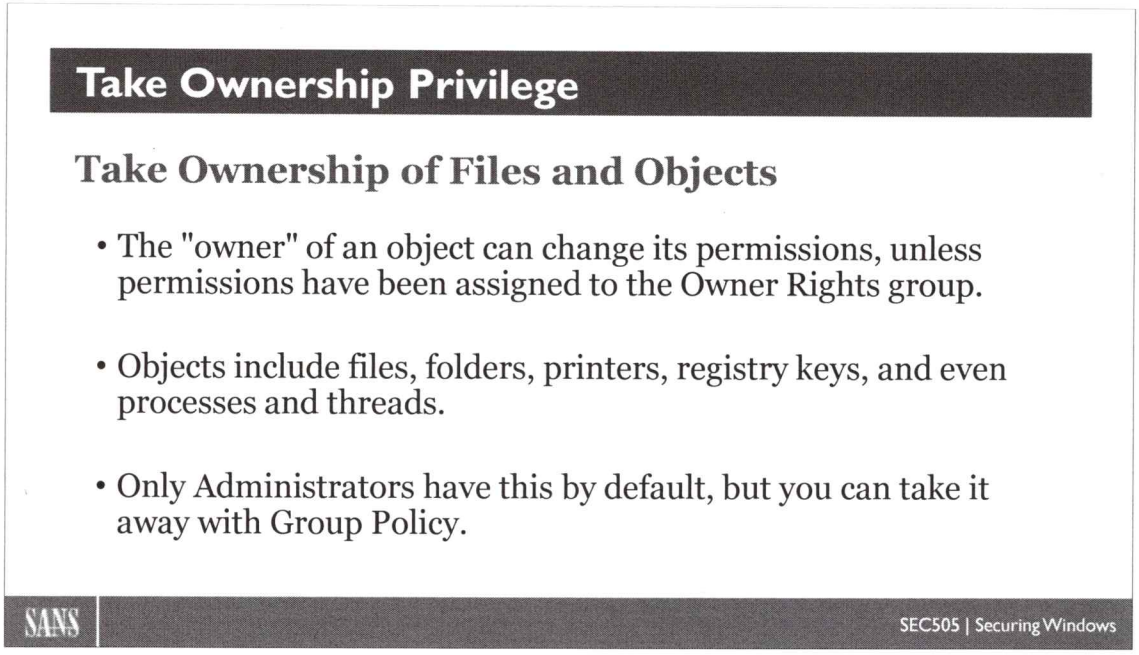
Notice that you can read, write and save any portion of the raw virtual memory of this process! Just click anywhere inside the hex window, hold down the "f" key to write bytes as hex characters, then click the Write button. When you're done, click Close.



If you click the Strings button in the upper right, then click OK, you can extract the textual strings from the virtual memory of the process. You may then use the Filter button at the bottom left to search these strings with regex patterns. Close.

In Process Hacker, right-click on your CMD.EXE process > Terminate Tree > click the Terminate button. (The "tree" includes all the indented child processes underneath the terminated process.)

Why was Process Hacker able to inject a DLL? How could it read or write the virtual memory address space of any other process? Process Hacker is running with administrative privileges, one of which is called "Debug Programs".



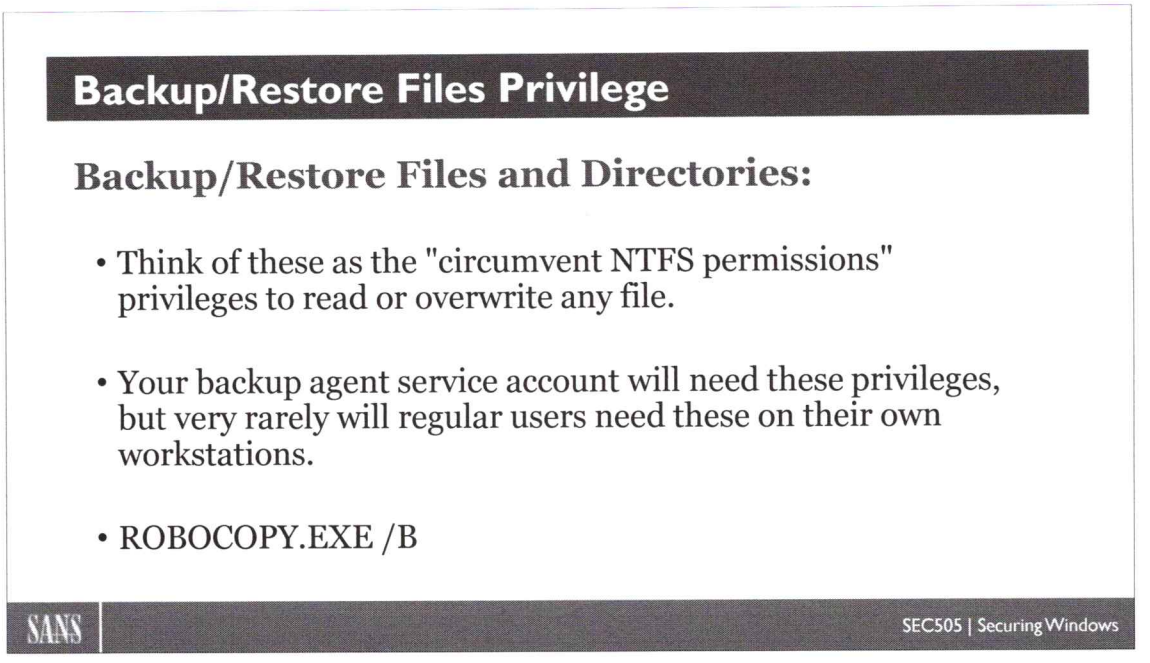
Take Ownership Privilege

Another dangerous privilege is *Take Ownership of Files Or Other Objects*. The owner of an object can change its permissions in any way desired. Objects that have owners include NTFS files and folders, Active Directory objects, printers, registry keys, processes and threads.

The only exception is when, on Vista/Server 2008 or later, an object also has permissions assigned to the built-in Owner Rights group, in which case the permissions assigned to Owner Rights take precedence over the default power of owners to do anything to the object.

Malware launched from the user's browser or e-mail client could use this privilege to plant trojans, copy or modify files, edit the Run key in the registry, and so on. In general, this privilege allows malware to get around restrictive permissions.

Using Group Policy, you should take away the Take Ownership privilege from the Administrators group and grant it instead to Domain Admins (and maybe the Help Desk group). If it's not possible to remove users from the Administrators group, at least you can remove this dangerous privilege.



Backup/Restore Files Privilege

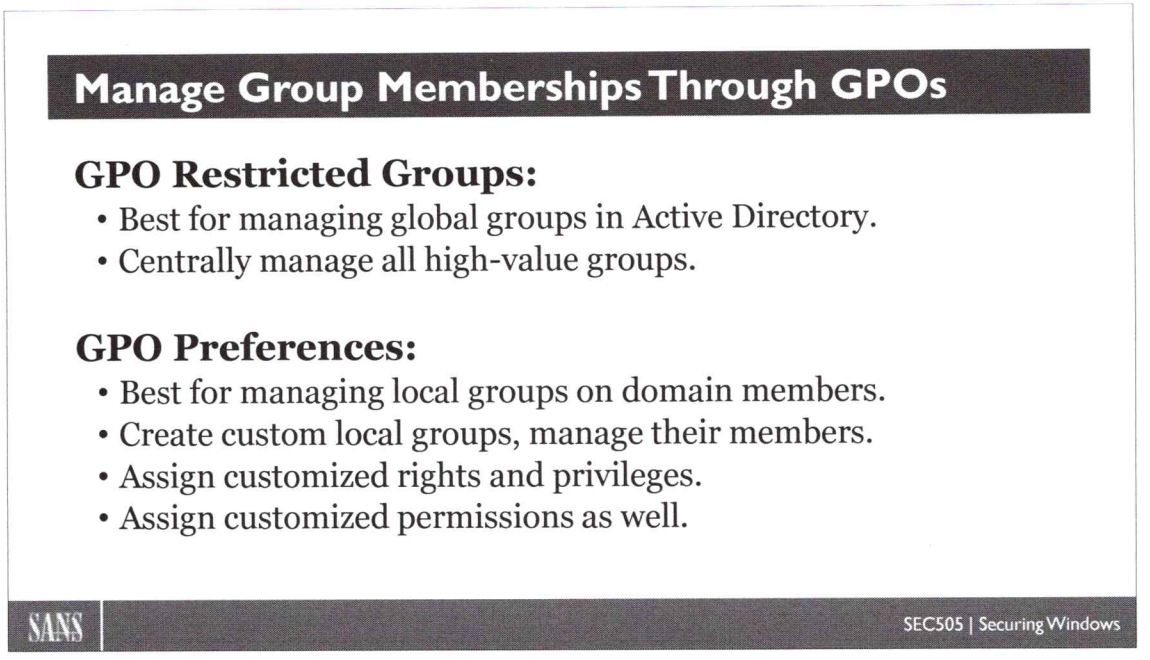
The seemingly innocuous *Backup Files And Directories* and *Restore Files And Directories* privileges are actually quite dangerous. Think of these as the *Ignore NTFS Permissions* privileges since one allows an infected process to read any file and the other allows an infected process to overwrite any file, including programs and OS binaries. A malware process with these privileges could read any unencrypted file on the infected machine and try to modify any binary file (but see the Windows Resource Protection service discussion later).

In an enterprise environment, user data is typically backed up and restored by a local or over-the-network agent, not by the users themselves. The backup agent software will run under the context of a service account, which may be a computer or user account itself. It is the backup service account which requires the *Backup Files And Directories* and *Restore Files And Directories* privileges, not the users themselves. Often, that service account is put into the local Administrators group too (which is bad for variety of reasons) but you can use Group Policy to separately grant that service account whatever permissions and privileges it needs (and no more).

Hence, use Group Policy to remove the *Backup Files And Directories* and *Restore Files And Directories* privileges from the local Administrators group on users' workstations, and then grant those privileges to a new custom group which contains your backup agent's service account (plus any additional NTFS permissions that group may require). This way, even if you can't remove users from their Administrators groups, this group won't have these dangerous privileges for malware to utilize.

Principle of Least Privileges

At this point I think you can see the pattern of recommendations here: remove privileges your users do not require. The good news is that the few we just discussed are by far the most important and dangerous, especially *Debug Programs*, so you don't have to spend weeks testing every single possible privilege.



Manage Group Memberships Through GPO

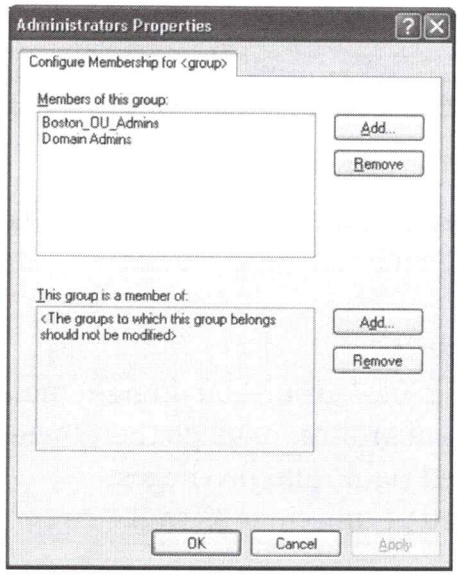
Active Directory permissions and Group Policy allow you to delegate authority very precisely. Group Policy can be used to manage the memberships of groups and the assignment of rights and privileges on all AD member computers. These abilities are extremely powerful. Power on a Windows network comes from being a member of groups because rights, privileges and permissions are assigned to groups.

Manage Global Groups

Group Policy can manage the membership of groups in AD. One place where these management features are located in a GPO are under Computer Configuration > Policies > Windows Settings > Security Settings > Restricted Groups. The Restricted Groups container permits you to define the exact membership of almost any group you wish, local or global, but we're going to use something else to manage local groups in a moment.

Keep in mind, though, that this Restricted Groups feature is not just for adding or appending more members to a group, it reconstitutes the group completely, i.e., removing the members that are already there and replacing the membership with that defined in the GPO.

For example, in the following screenshot the membership of the local Administrators group is being managed. The Administrators group's membership will be just Boston_OU_Admins and Domain Admins. If either group is missing from the Administrators group when the GPO with this setting is applied, they will be added automatically. If any *other* users or groups are members, they will be *removed.* An important exception, though, is the local Administrator account-- it can't be removed.



Try It Now!

To manage the membership of a group, open a GPO and go to Computer Configuration > Policies > Windows Settings > right-click on Restricted Groups > Add Group > enter or browse for the name of the group whose membership you wish to manage > OK > top Add button > enter or browse for the member > OK. Repeat as necessary until all desired members have been added.

Also, unless you run "gpupdate.exe /force", you'll have to wait until the next reboot or up to 16 hours before the group membership change takes effect. This is just how it is designed.

If you don't want to totally replace the membership of a group, then use the bottom half of the dialog box labeled "This group is a member of:". In this case, you would add the desired *global* group to the GPO, then add the target *local* group to the bottom "This group is a member of" list in the dialog box. See the difference? When you want to replace the membership of a local group with the GPO, you add the local group to the list of Restricted Groups, then you configure the top "Members of this group" list. When you want to append a global group to the current members of a local group, you add the global group to the Restricted Groups list in the GPO, then configure the bottom "This group is a member of" list with the name of the target local group.

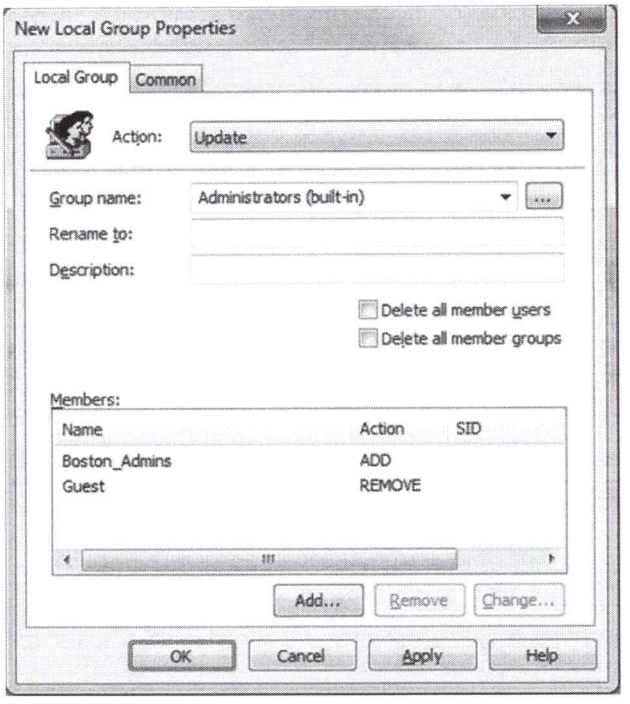
On the whole, however, it might be easier to manage Global and Universal groups through PowerShell scripts, perhaps as scheduled tasks, since the scripts can implement any decision-making logic desired.

But while it is possible to manage local groups with the Restricted Groups container in a GPO, there is actually a simpler way to do it with Group Policy.

Manage Local Groups

Another place in a GPO for managing local group memberships is located under Computer Configuration > Preferences > Control Panel Settings > Local Users and Groups. These GPO Preferences can be used to manage local user accounts and groups if you have Windows 7 or later (or Windows XP/2003/Vista with the necessary updates).

With GPO Preferences it's easy to specify whether a local group's membership should be wiped first or merely edited. You can also create or delete local groups this way.



Please note that sometimes this feature is finicky. You may need to add multiple rules to the GPO, some with the "Replace" action and others with the "Update" action, in order to achieve the end result you want across all the target versions of Windows in your environment.

Example Use: Emptying Administrators-Equivalent Local Groups
Microsoft recommends that the following local groups be kept empty whenever possible:

- Backup Operators
- Cryptographic Operators
- Hyper-V Administrators
- Network Configuration Operators
- Power Users
- Remote Desktop Users
- Replicator

Some of the above groups are essentially equivalent to the built-in Administrators group in terms of power to take over the computer and make malicious changes (such as Power Users), while others are too dangerous when misconfigured (such as Remote Desktop Users).

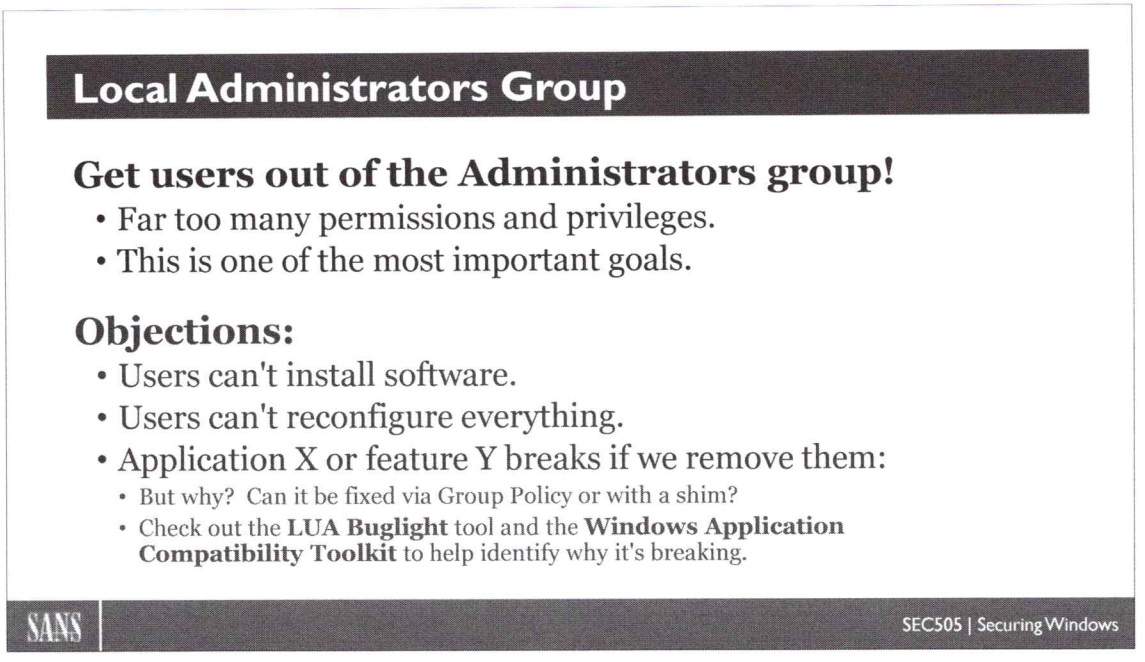
After enforcing an empty membership in the above groups, we can focus our attention on the Administrators group to keep its membership to the minimum.

Requirements
The target recipients of GPO Preferences (the managed clients) must have the following:

- Windows 7/Server 2008 or later (no other updates necessary).
- Windows Vista+SP1 or later SP, plus the Client Side Extensions (CSE).
- Windows Server 2003+SP1 or later SP, plus the Client Side Extensions (CSE) and, if the latest SP or IE is not installed, the XMLLite update too.
- Windows XP+SP2 or later SP, plus the Client Side Extensions (CSE) and, if the latest SP or IE is not installed, the XMLLite update too.

The Client Side Extensions (CSE) can be downloaded from Microsoft's main Group Policy page at http://technet.microsoft.com/windowsserver/grouppolicy/.

The XMLLite update is bundled into Internet Explorer 7.0 and later, with XP-SP3 and later, and with Server 2003-SP2 and later Service Packs. If necessary, XMLLite can be downloaded separately from http://go.microsoft.com/fwlink/?LinkId=111843.



Local Administrators Group

One of the most important defenses against targeted attacks and malware infections is to get users out of the local Administrators group on their computers.

When logged on with local Administrators group membership, a user is more likely to get infected and that infection is more likely to result in a total compromise of the machine (instead of just a crash or "merely" a hijacked HTTP session). The problem is that the Administrators group by default has write access to virtually the entire hard drive and registry, can seize ownership of any other file or key because of the *Take Ownership* privilege, and Administrators have the *Debug Programs* privilege which facilitates DLL Injection, pass-the-hash attacks, rootkit installation, and other nastiness.

Objections

One objection to this recommendation is that users won't be able to install software. *Good!* This is precisely what we don't want. The software users do require should be managed and installed centrally. The same infrastructure you've created to install patches and deploy MSI packages can also be used to install whatever legitimate software your users require. Centralized application control can also save you money if licenses are better managed.

Another objection is that users won't be able to reconfigure Windows and their applications as they desire. *Good!* When users have a choice between doing the easy thing and the right thing, they always choose the Big Easy. Again, we want centralized control over the configuration of their machines so that we can impose, against their will, the right choices. And for the configuration options that don't matter to security, like desktop color scheme, we can use Group Policy to grant that power without making users

full local Administrators. This objection is mainly a political or corporate culture issue, but after an outbreak you can often get management's support for being more strict.

A third objection is that users won't be able to run application X or use feature Y without being Administrators. But if this is true because of some registry or NTFS permissions, we might be able to use Group Policy to grant just those permissions. Or if this is true because a special driver needs to be loaded on the fly, then we might be able to use Group Policy to pre-load the driver. In general, virtually no application is explicitly designed to require Administrators membership, the application is just trying to do something dangerous and is being blocked by a missing permission or privilege, but these can be added through Group Policy and scripting, so the first question to ask is *Why* is the application or feature failing? (And there is a larger issue: if a user application requires Administrators membership, maybe the application is just badly designed, so perhaps it's better to upgrade or replace it anyway.)

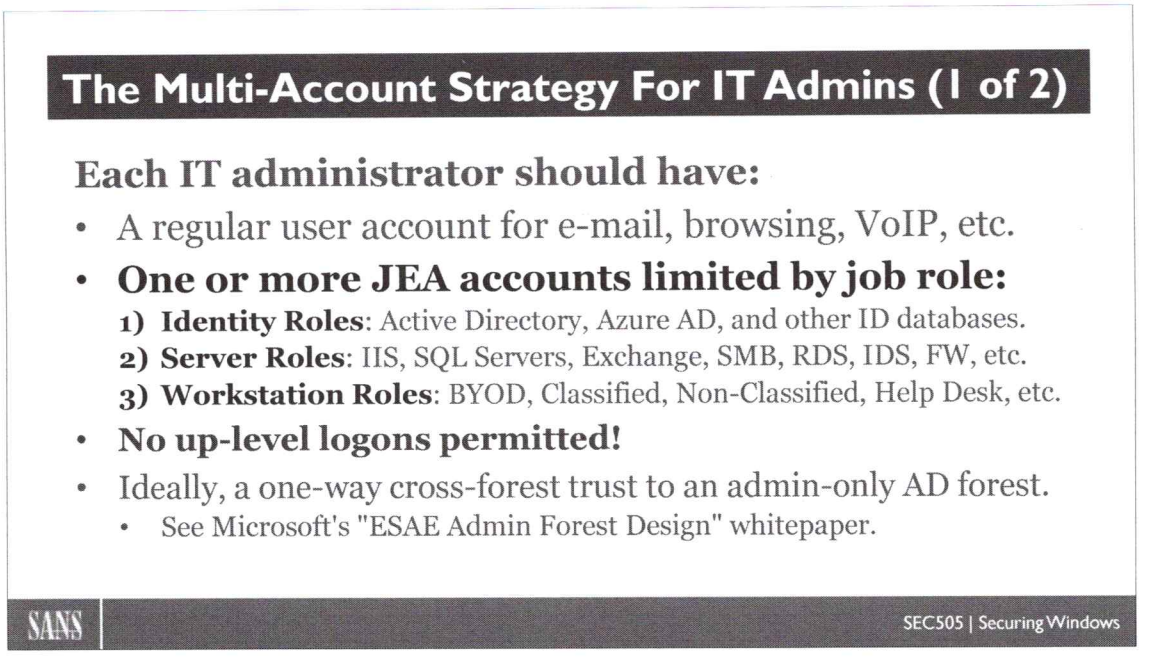
Analyze Application Failures

To help understand why an application or feature is breaking without Administrators membership, check out the LUA Buglight tool from Microsoft and its associated LUA blog (http://blogs.msdn.com/b/aaron_margosis/). LUA Buglight can help you to fix up applications to run on post-XP machines when the user is not a local Administrators member.

Also download Microsoft's Windows Application Compatibility Toolkit (ACT), which is specifically designed to help migrate applications to new platforms or new configurations of those platforms (do a search on the tool's name to get the URL to the latest version).

If you're really stumped, try using Process Monitor to log all file system and registry access during the time when the desired feature fails, which will help to pinpoint where the breakdown is occurring (http://technet.microsoft.com/en-us/sysinternals/).

If you're totally stumped, call the vendor. Other customers of the vendor will have run across these problems in the past too, so the vendor might have a ready answer for you.



The Multi-Account Strategy For IT Admins (1 of 2)

Everyone agrees that getting users out of the Administrators group is good for security, but what about IT personnel? Just because we work in IT doesn't mean we're immune to infections or hack attacks; in fact, just the opposite, precisely because we work in IT we are more likely to be targeted!

Administrative Users Should Have Two Or More User Accounts

Administrators should have two or more user accounts: one low-powered regular account for non-administrative activities, and one or more high-powered administrative accounts used only when necessary to perform IT duties, and only with the minimum power necessary to complete these duties (i.e., "just enough admin" accounts).

Admins should log onto their desktops with their regular accounts, then launch administrative tools under the context of their administrative accounts only as needed, preferably only at dedicated administrative workstations or VMs accessed via a thin-client protocol like Remote Desktop Protocol (RDP) or Citrix ICA.

Web Browsing and E-Mail

Most importantly, we don't want the applications through which we are most likely to be infected to run with local Administrators membership, such as the browser, e-mail program, instant messaging, document viewer, peer-to-peer client, etc. We also don't want these applications to run as Domain Admins either of course. Once a machine is infected, the malware can begin harvesting credentials from memory or keystrokes from the keyboard.

To avoid temptation, administrative user accounts should not have e-mail addresses or mailboxes on any e-mail servers. Outbound proxy servers should require authentication from users and block all Internet HTTP/FTP access by members of any administrative groups. If the proxy servers work anonymously, then the proxies should at least block all Internet access from the source IP addresses set aside for the admins' jump servers.

One or More Admin Accounts Per Job Role

Broadly speaking, administrative roles cover three major areas of responsibility with different levels of impact should an administrative account be compromised by hackers or malware. The three roles or levels of responsibility are:

1) **Identity Roles**: Responsible for Active Directory, Azure Active Directory, or other identity management systems such as LDAP and RADIUS servers.

2) **Server Roles**: Responsible for the various types of servers in the organization, such as public servers in the DMZ, private internal servers, cloud-hosted public or private servers, infrastructure devices, and other hardware- or software-based services upon which users rely but which they do not manage themselves.

3) **Workstation Roles**: Responsible for user endpoint devices like phones, tablets, laptops, PC workstations, Virtual Desktop Infrastructure (VDI) virtual machines, and the help desk technical support personnel for regular users.

Each admin should have zero, one or multiple administrative accounts at each role level, depending on the job duties of that administrator (not every admin will occupy roles at all levels).

Importantly, an admin account at any level is not permitted to log into or manage resources at any level *above* it. Workstations are more likely than servers to be infected with malware or under hacker control, and servers are less likely to be infected or remotely controlled than identity management systems (or the workstations used to manage them, hopefully). The goal is to help prevent the spread of harm from compromised admin credentials.

Up-level logons are to be prevented through user logon rights restrictions managed through Group Policy, INF templates or PowerShell DSC. Though it is possible for an administrative account to log on at systems at a lower level, it is better for each administrative role account to only log on and manage systems at the same role level.

Restrict Logon Rights

Consider having a naming standard for each admin's high-powered accounts, e.g., ending the username with "_a" or another pattern easy to recognize and filter. Carefully document the various high-powered groups in the domain at each role level; for example, the following groups tend to be very powerful:

- Account Operators

- Backup Operators
- Cryptographic Operators
- Domain Admins
- Domain Controllers
- Enterprise Admins
- Group Policy Creators Owners
- Print Operators
- Read-Only Domain Controllers
- Schema Admins
- Server Operators
- <your-domain>\Administrators (this is the *domain* local group)

Use Group Policy to deny local, batch and service logon rights to the privileged AD groups at down-level servers and workstations. You may need to customize service accounts and scheduled task accounts to achieve this.

Separate Admin Forest (Microsoft ESAE Design)

A set of Microsoft articles describes the Enhanced Security Administrative Environment (ESAE) forest model. In the ESAE model, a separate Active Directory forest exists for the sole purpose of managing and protecting administrative users, computers, groups and service accounts. The main production AD forest would have a one-way cross-forest trust to the administrative forest. The administrative forest would be tiny, consisting of just one domain and relatively small number of objects; only two domain controllers, which might be virtual machines, might be all this required, though this number depends on several factors.

This course (SEC505) used to have an entire manual dedicated to AD forest design, but this material was deleted to make space for new material. It was also deleted because 99% of attendees already had AD forests in production and it was generally not possible to redesign them. Please search Microsoft's web site for "ESAE forest" for further discussion of administrative forests and their implementation. If there is enough demand, SEC505 will include the AD design material again, but, please remember, everything added requires something else to be deleted to make space for it.

Run As Tricks

It is possible to log on as one administrative account and launch programs as a different administrative account. There are variety of ways to make this more convenient:

- Using the *Run As* feature and User Account Control (UAC), for example, you can right-click on any MMC console file and select "Run As..." to bring up a dialog box for the credentials of the administrative account under which you want the program to run.

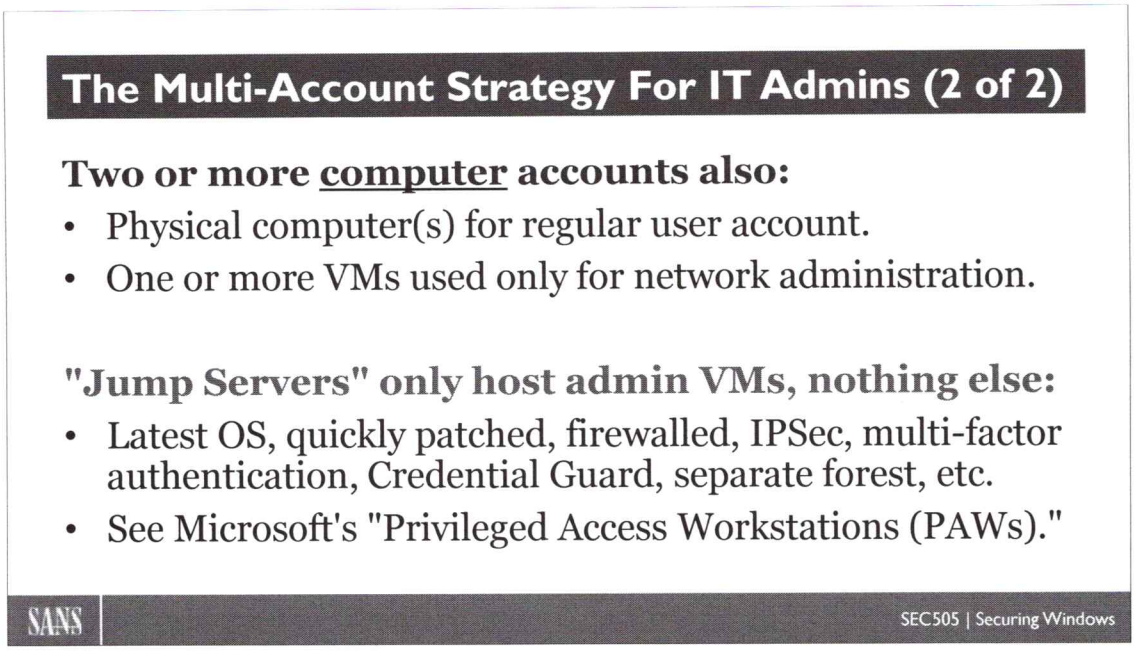
- You can add a shortcut to users' desktops or Start Menus which have the "Run As Administrator" option checked. Since the user is not a member of the local

Administrators group, the user will be prompted for a different username and password when they run the shortcut.

- You can also add a script to users' desktops or Start Menus which asks for a username and password for the application, then uses another tool to run the application like RUNAS.EXE, PSEXEC.EXE, CPAU.EXE, AUTOIT.EXE, AUT2EXE.EXE, EPAL.EXE, TQCRUNAS.EXE, etc. This has the advantage being able to use the other command-line switches of these tools (for good or ill).

- You could run the application only on a Citrix or Remote Desktop Services server, then prompt the user for the credentials when the application is accessed. The application appears to be running locally in that the thin client application does not show a full desktop at the Citrix/RDP server but only the running program and this program's cut/copy/paste and file system interaction is automatically redirected to the user's local computer. This strategy also provides greater control with the centralized hosting.

Hopefully, using these tricks will make having multiple administrative accounts convenient enough that the other administrators will not resist the strategy.



The Multi-Account Strategy For IT Admins (2 of 2)

Administrators should also have two or more *computer* accounts in Active Directory, not just multiple user accounts.

One computer account will be for the physical box used with the regular user account to surf the Internet, check e-mail, play games, manage documents, and to do all the other activities likely to result in malware infections.

The additional computer accounts should be for physical or virtual machines that are used only to perform IT administrative duties. Ideally, these computers would be physical devices connected to their own separate networking infrastructure, but this can be prohibitively expensive for smaller organizations, hence, using VMs instead is certainly acceptable (and certainly better than using just one physical workstation for everything).

Jump Servers

The dedicated admin VM could be run locally on the administrator's computer as a last resort, but really the VM should be run on a dedicated virtualization server. This virtualization server, called a "jump server", should host only the VMs of network administrators and nothing else. Admins will access their VMs via desktop thin-client protocols, like RDP or ICA, or via PowerShell remoting. The VMs will have all the tools the administrators require to get their jobs done.

Having one or more dedicated jump servers makes firewalling, IPSec, smart card usage, secure remote access, OS upgrades, application upgrades, patch management, and other hardening protections easier to manage and enforce. The aim is to keep the administrative VMs as clean and protected as possible.

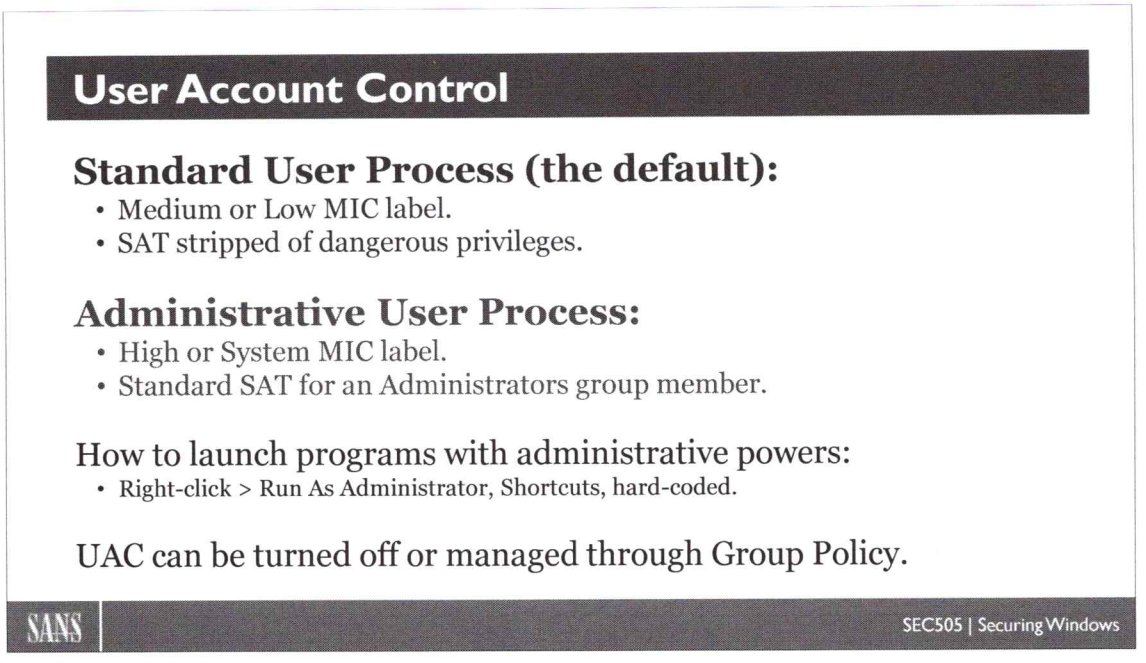
Per-Role VMs and/or Jump Servers

If an administrator has multiple admin accounts for different roles (Identity, Servers, Workstations), then that administrator should ideally have a separate VM for each role level. Ideally, there would be multiple jump servers, one for each role level, but this might be too expensive for small organizations, hence, these organizations will host multiple admin VMs at different role levels on one jump server.

SEC505 for Microsoft PAWs

These jump server admin VMs will be secured with all the techniques discussed in this course, such as with IPSec, firewall rules, Credential Guard, whole disk encryption, aggressive patch management, anti-exploit technologies, application whitelisting, VDI, and more.

Recently, Microsoft has dubbed such administrative jump servers and workstations "Privileged Access Workstations (PAWs)", but the idea has been around for more than a decade. Microsoft provides very nice guidance on the implementation of PAWs on the TechNet web site (just do an Internet search on the acronym). Most of the manuals in SEC505 are designed to help implement PAWs and secure jump servers.



User Account Control

Users who log on and run all of their programs as members of the local Administrators group endanger their computers because of malware and destructive mistakes. User Account Control (UAC) in Windows Vista and later allows users to conveniently install and run programs as low-privileged accounts and then temporarily raise privileges on an as-needed basis without logging-on-and-off or resorting to RUNAS.EXE in order to do so.

How UAC Works

Even when enabled, UAC does not apply to the built-in Administrator account by default. UAC does apply to all other accounts, even if those accounts have been added to the local Administrators group.

> **Note:** By default the built-in Administrator account cannot be used for interactive logons, but a registry modification makes it available in the graphical list of local accounts in the initial start screen. Launch REGEDIT.EXE as an Administrator and navigate to the following key: HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\UserList (if the SpecialAccounts and UserList keys do not exist, create them). In the UserList key, create a REG_DWORD value named "Administrator" and set it to 1. Afterwards, enable the Administrator account in Control Panel or with NET.EXE.

Whenever a user logs on (other than the built-in Administrator) the Security Access Token (SAT) of that user is stripped of most of its rights, its Mandatory Integrity Control (MIC) level is set to Medium, and, if the user is a member of the local Administrators group, the Administrators group's SID is added to user's SAT as a deny-only group (see

"WHOAMI.EXE /all"). A group's SID which has been marked as "deny-only" in a SAT cannot be used to grant a permission or right, it can only be used to deny access (for more information, search on "SE_GROUP_USE_FOR_DENY_ONLY" in the MSDN Library). Hence, even if a user is a member of the local Administrators group, that user acquires none of that group's rights or permissions when logging on.

If the user is a member of any of the following groups, these groups are marked as "Use for Deny Only" in the SAT when UAC modifies that token:

- Administrators
- Backup Operators
- Power Users
- Network Configuration Operators
- Cryptographic Operators
- Domain Admins
- Schema Admins
- Enterprise Admins
- Group Policy Creator Owners
- Domain Controllers
- Enterprise Read-Only Domain Controllers
- Account Operators
- Print Operators
- Server Operators
- RAS Servers
- Pre-Windows 2000 Compatible Access

UAC will also strip all privileges out of the SAT except for the following:

- Bypass traverse checking
- Shutdown the system
- Remove computer from docking station
- Increase a process working set
- Change the time zone

A process running with a SAT stripped of its higher privileges and with an MIC level of Medium or lower is said to be "running as a standard user". A process running with a SAT that includes the Administrators group's SID and other elevated rights with an MIC level of High or better is said to be "running as administrator". This nomenclature is a bit misleading since all users log on as "standard users", but just remember that a SAT is created for a process when that process is launched and that that SAT can be modified on-the-fly by the operating system for the sake of UAC and MIC.

> **Note:** When examining the privileges of a process with WHOAMI.EXE or Process Explorer, don't forget that a privilege labeled as "Disabled" can still be

enabled by the process as needed, but if a privilege is not listed at all, then that privilege cannot be enabled for that process.

Note: To read more about integrity labels, Google on "site:microsoft.com windows vista integrity mechanism technical reference", which was last seen at http://msdn2.microsoft.com/en-us/library/bb625964.aspx

If a standard user process attempts an action that requires administrative privileges, the action will usually fail; however, if that process is 32-bit, does not specify a requestedExecutionLevel in its application manifest (PE or .NET), is not running in kernel mode, is not impersonating a different user, and is failing because of an Access Denied error from an NTFS or registry permission on various items under %SystemRoot%, %ProgramFiles%, the SOFTWARE hive and some other locations (and not because of an MIC restriction), then the write access appears to be permitted, but it is only permitted to the "virtualized" folders and keys of the same names but not the same locations. These virtualized folders and keys were added by Microsoft for backwards compatibility and are located, respectively, under %LOCALAPPDATA%\VirtualStore\ and HKCU\Software\Classes\VirtualStore\. These virtualized folders and registry keys are per-user, hence, each user will have their own separate set of virtualized folders and keys that appear to be "the real ones", but only administrative processes can actually write to the real folders and keys. (And note that 64-bit processes by default cannot take advantage of this folder/key virtualization.)

If you right-click an executable or shortcut and select "Run As Administrator", then that process will run as an administrative user (if permitted). If the properties of a shortcut are modified to enable "Run With Different Credentials" (Shortcut tab > Advanced button) then that process will run just as though you had right-clicked it and selected "Run As Administrator" (which is quite different than the nearly useless but similar-looking feature in XP (search http://blogs.msdn.com/aaron_margosis/)). If the manifest in a Portable Executable (PE) binary or in a .NET assembly has the requestedExecutionLevel property set to "requireAdministrator" (do a strings search in the binary to see it), then that process will automatically launch as an administrative user (if permitted).

Note: The RUNAS.EXE /TrustLevel switch is for Software Restriction Policies, not UAC or MIC, but you can still use the /User switch to launch a program as the Administrator account (with MIC-High and UAC-Administrative privileges).

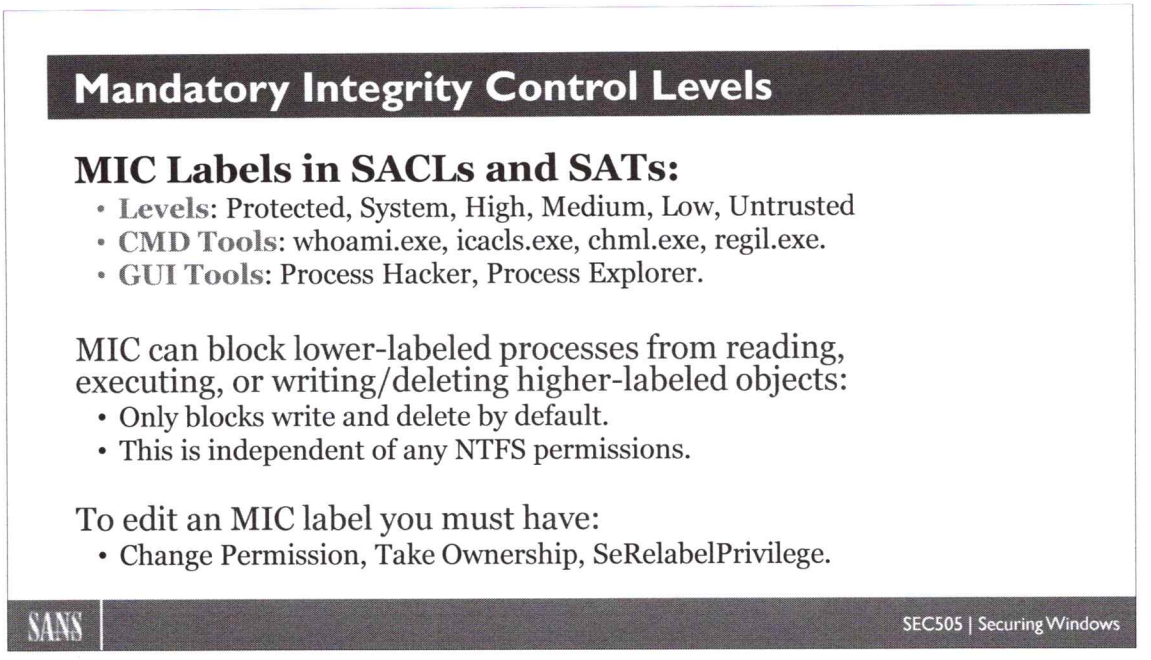
When is a process permitted to launch as an administrative user?

UAC Group Policy Options

Group Policy can be used to configure UAC. Inside a GPO, navigate to the following location: Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options. Here you will find the following UAC-related options:

- UAC: Admin Approval Mode for the Built-in Administrator Account.
- UAC: Behavior of the elevation prompt for admins in Admin Approval Mode.
- UAC: Behavior of the elevation prompt for standard users.
- UAC: Detect application installations and prompt for elevation.
- UAC: Only elevate executables that are signed and validated.
- UAC: Run all administrators in Admin Approval Mode.
- UAC: Control Switch to the secure desktop when prompting for elevation.
- UAC: Virtualize file and registry write failures to per-user locations.

The most important two options are for "UAC: Behavior of the elevation prompt…" (which are underlined). This determines how UAC allows the execution of commands that require administrative privileges. For administrator-level users, the behavior options are: 1) no prompt, just automatically elevate privileges, which gives the appearance that UAC is disabled, 2) Prompt for consent, to simply be alerted, and 3) prompt for credentials, which requires a passphrase or smart card. For standard users, the options are: 1) no prompt, and simply fail, or 2) prompt for credentials, which requires the passphrase or smart card of an administrative-level account.



Mandatory Integrity Control Levels

Another form of power in the Windows kernel revolves around the use of Mandatory Integrity Control (MIC) levels. MIC is a partial implementation of the Biba mandatory access control model for preserving data integrity, especially the integrity of operating system files, the registry, and data exchanged between visible applications on the desktop. MIC is enabled by default in Windows Vista and later. (MIC is also known as "Windows Integrity Control.")

MIC assigns a "label" to each securable object. Securable objects include folders, files, registry keys, shares, processes, threads, named pipes, services, IPC objects, Active Directory objects, and just about anything else that can have an access control list assigned to it, i.e., anything that can have permissions. The label is stored as a part of the object's System Access Control List (SACL) alongside the regular audit settings.

An MIC label is one of the following, from highest to lowest:

- Protected (SID: S-1-16-?, Hex: ?, SDDL: ?)
- System (SID: S-1-16-16384, Hex: 4000, SDDL: SI)
- High (SID: S-1-16-12288, Hex: 3000, SDDL: HI)
- Medium (SID: S-1-16-8192, Hex: 2000, SDDL: ME) - **Default Label**
- Low (SID: S-1-16-4096, Hex: 1000, SDDL: LW)
- Untrusted (SID: S-1-16-0, Hex: 0, SDDL:)

If an object lacks a MIC label, then that label is assumed to be Medium. Operating system files lack MIC labels, so they are handled as Medium-labeled files.

As you launch processes, your Security Access Token (SAT) is assigned to each process you launch. Your SAT includes the Security ID (SID) number of your user account, the SIDs of your groups, plus other information. In Windows Vista and later, your SAT also includes an integrity SID that identifies your MIC label. When your SAT is inherited by a process you launch, the label will be Medium if the process was launched as a standard user (the default), High if launched with administrative privileges, or Low if that process happens to be Internet Explorer or another program designed to run as Low. Most services run with the System label.

You can see these SAT labels with WHOAMI.EXE, Process Explorer, or Process Hacker. You can also launch processes with the Low MIC label with the PSEXEC.EXE tool from http://www.microsoft.com/sysinternals/.

```
C:\> whoami.exe /groups /fo list | select-string 'Mandatory'

Group Name: Mandatory Label\High Mandatory Level
```

When a process is launched, such as EXPLORER.EXE when you log on, that process will be assigned the Medium MIC label by default; but if that process will have a SAT that includes any of the following privileges, then that process will be assigned the High MIC label instead:

- Create a token object (SeCreateTokenPrivilege)
- Act as part of the operating system (SeTcbPrivilege)
- Debug programs (SeDebugPrivilege)
- Modify an object label (SeRelabelPrivilege)
- Take ownership of files or other objects (SeTakeOwnershipPrivilege)
- Load and unload device drivers (SeLoadDriverPrivilege)
- Back up files and directories (SeBackupPrivilege)
- Restore files and directories (SeRestorePrivilege)
- Impersonate a client after authentication (SeImpersonatePrivilege)

In general, if one process launches another, the child process inherits the MIC label of the parent process. But there are important exceptions. If the child process was "Run as administrator", then it'll get the High MIC label even if the parent process has a lower MIC label. If the child process was launched with RUNAS.EXE, the child process will get the Medium label, unless RUNAS.EXE specifies the local built-in Administrator account and it is still exempted from UAC (the default). If the executable file of the child process has a lower MIC label than the parent's, the child process does not inherit the parent's label, it will run with its own lower label, unless the parent process is running as High or better, in which case the child process will run High or better too. If the parent process triggers CONSENT.EXE for the sake of UAC when executing the child process, the child process will run as High if consent is granted to the elevation.

Integrity Control Tools

For viewing or editing MIC settings there are no GUI tools from Microsoft, but Microsoft does provide the built-in ICACLS.EXE for managing NTFS permissions and doing some basic MIC tasks. When using ICACLS.EXE, remember that the absence of a MIC label is interpreted as the Medium label by default; unfortunately, ICACLS.EXE does not show any MIC information if it has not been explicitly assigned.

However, much better MIC tools are the following, and they're all free:

- CHML.EXE and REGIL.EXE (http://www.minasi.com/apps)
- RUNASIL.EXE (http://blog.didierstevens.com/2010/12/01/runasil/)
- Process Hacker (http://processhacker.sourceforge.net)

Viewing and Changing Labels

To read the MIC label on an object, you only need Read permission. But to change a label, you need:

1) the NTFS Change Permission permission on that object,
2) the Take Ownership permission on that object, and
3) have the "Modify an object label" privilege (SeRelabelPrivilege) in your SAT.

By default, no one has the "Modify an object label" privilege, not even local Administrators, so if you want to experiment with MIC, grant yourself this privilege now, log off, and log back on.

When you change an MIC label, you can either raise or lower it, but you cannot raise it higher than the label of the process you are using to make the change, i.e., if you're running ICACLS.EXE in a CMD shell running with the High MIC label, you can at best upgrade another object's label to High, not to System. (Note: Scheduling a job task to run a script to change an object's label to System or Installer doesn't work, and neither does using a Group Policy-assigned startup or shutdown script to attempt the same.)

Integrity Levels vs. NTFS Permissions

MIC is enforced independently of, and prior to, the enforcement of permissions, such as NTFS and registry key permissions. For example, even if you have NTFS Full Control over a file, if you launch Notepad with the Medium MIC label and the file in question has the High label, then your Notepad will not be able to save changes to the file even though your account has Full Control; Notepad will just raise a not-too-useful error message when trying to save the changes: "Cannot create the file…" (the error message says nothing about MIC issues or how to resolve them).

No Read (NR) -- No Write (NW) -- No Execute (NX)

By default, MIC prevents a lower-labeled process from deleting or changing an object with a higher label. But MIC can also be used to prevent lower-labeled processes from reading or executing higher-labeled objects too.

Mark Minasi's free MIC tool, CHML.EXE, is better than Microsoft's ICACLS.EXE for managing MIC label options. For example, to assign the High MIC label (-i:h) to a file named "program.exe" and to prevent a lower-labeled process from reading (-nr), writing/deleting (-nw) or executing (-nx) it, you would run:

```
chml.exe program.exe -i:h -nr -nw -nx
```

Attempting to run, copy or delete the program from within a Medium-labeled CMD shell results in an error message even if you have NTFS Full Control. You can use all three options together, as in the example above, or you can forbid any one or any combination of actions, e.g., you might allow reading by lower-labeled processes, but deny execution or changes/deletion.

To set the label to High, allow Read access, but deny Write or Execute access:

```
chml.exe program.exe -i:h -nw -nx
```

If you assign the Untrusted MIC label to an executable, you won't be able to run that executable from Medium- or Low-labeled processes at all, but High-labeled or better processes will still be able to run it normally.

User Interface Privilege Isolation (UIPI)

User Interface Privilege Isolation (UIPI) prevents a process from sending Win32 GUI messages to other processes in the same session if the sending process has a lower MIC label than the MIC label of the receiving process; for example, a process with the Low MIC label cannot send Win32 messages to a Medium-labeled process. Microsoft added this to Vista and later to combat the threat from "shatter" attacks. And to help enforce session-0 isolation, Win32 messages cannot be sent across sessions by default either.

(Incidentally, if you are a developer, UIPI also prevents lower-labeled processes from performing a windows handle validation of a higher process, attach a thread hook to a higher process, monitor a higher process with a journal hook, or injecting DLLs into higher processes.)

In general, UIPI seeks to prevent almost all invasive interaction from a lower- to a higher-labeled process, while still trying to maintain backwards compatibility. Win32 messages between processes at the same level, or from a higher to a lower level, are still permitted as usual. However, UIPI does not prevent any process from accessing the desktop window, desktop heap read-only shared memory, global atom table, the visible surfaces of windows, or the clipboard. And don't forget that a user might have many processes running simultaneously with different MIC labels and different administrative

privileges, and all of these processes might be able to read/write some of the same files, registry values, LRPC interfaces, named pipes, etc. Hence, UIPI is not an airtight wall of separation.

MIC Lab Experiments

This lab simply walks us through using ICACLS.EXE and CHML.EXE to experiment with MIC labels. If we don't have time for this in seminar, you'll be able to reproduce it on your own using the following steps.

Ensure That User Account Control Is Enabled For Your Account

By default, the built-in Administrator account is immune to UAC and every process will launch with full privileges. The following lab won't work in this case. So either log on as any user other than the local or domain Administrator, or enable UAC for the Administrator account in Group Policy (GPO > Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options > set to Enabled the option named "User Account Control: Admin Approval Mode for the Built-in Administrator account").

View Integrity Labels of CMD Shells

1. Open a regular (non-elevated) CMD shell and run "title Medium CMD Shell". Notice that the title bar of that CMD shell now reads "Medium CMD Shell".

2. Open an elevated CMD shell and run "title High CMD Shell", and then run "color F4", which makes your background white and the foreground red.

3. In your Medium CMD Shell, run "whoami.exe /all /fo list". This more-or-less shows the Security Access Token (SAT) for this process. Notice that you have what appears to be a group membership near the bottom: "Group Name: Mandatory Label\Medium Mandatory Level". This labels your standard-user CMD shell as having the Medium MIC label.

4. In your High CMD Shell, run the same command: "whoami.exe /all /fo list". Notice that its MIC label is "Group Name: Mandatory Label\High Mandatory Level", or just High. Note also that you have a privilege near the bottom named "SeRelabelPrivilege", which allows you (or, rather, this CMD shell) to change the MIC label on an object.

Modify Integrity Labels of Files

1. In your High CMD Shell, go to the C:\Temp folder. (If it doesn't exist, create it now.)

2. In your High CMD Shell, run "echo HIGH > High.txt".

3. In your High CMD Shell, run "icacls.exe High.txt" to show the ACL on the High.txt file. Notice that no MIC label information is shown! If an object is unlabeled, what is its implied label? Medium.

4. In your High CMD Shell, run "icacls.exe High.txt /SetIntegrityLevel High" to change the MIC label on the High.txt file to High.

5. In your High CMD Shell, run "icacls.exe High.txt" and note that the ACL now says "Mandatory Label\High Mandatory Level:(NW)". The (NW) portion means that lower-labeled process cannot write or delete this file.

6. In your High CMD Shell, run "echo MEDIUM > Medium.txt".

7. In your High CMD Shell, run "icacls.exe Medium.txt /SetIntegrityLevel Medium".

8. In your High CMD Shell, run "icacls.exe Medium.txt". Note that you can see the MIC label even though it is Medium because it was explicitly set.

Test MIC Restrictions

1. In your **Medium** CMD Shell, go to the C:\Temp folder.

2. In your Medium CMD Shell, run "del High.txt". Note that access is denied even though you have NTFS delete permission on the file.

3. In your Medium CMD Shell, run "notepad.exe High.txt". Note that you can read the contents of the file. But now add some text and try to save the changes; you'll get a not-too-informative error message. Exit Notepad without saving.

4. In your **High** CMD Shell, run "notepad.exe Hight.txt", add some text, and save the changes. It worked because the Notepad process inherited the High MIC label of the CMD shell, hence, its label was equal-to-or-higher-than the label on the file being edited. Using this Notepad, you can also edit the Medium.txt file of course. Exit Notepad.

Using CHML.EXE

1. If you were given a CD-ROM by the instructor, copy the following file from the CD to your C:\Windows folder: CD:\Tools\chml\chml.exe. Or if you have Internet access, you can get this tool from Mark Minasi's website for free: http://www.minasi.com/apps/.

2. In your **High** CMD Shell, test that you can access the tool and view its available command-line switches by running "chml.exe /?". Make sure you're in the C:\Temp folder.

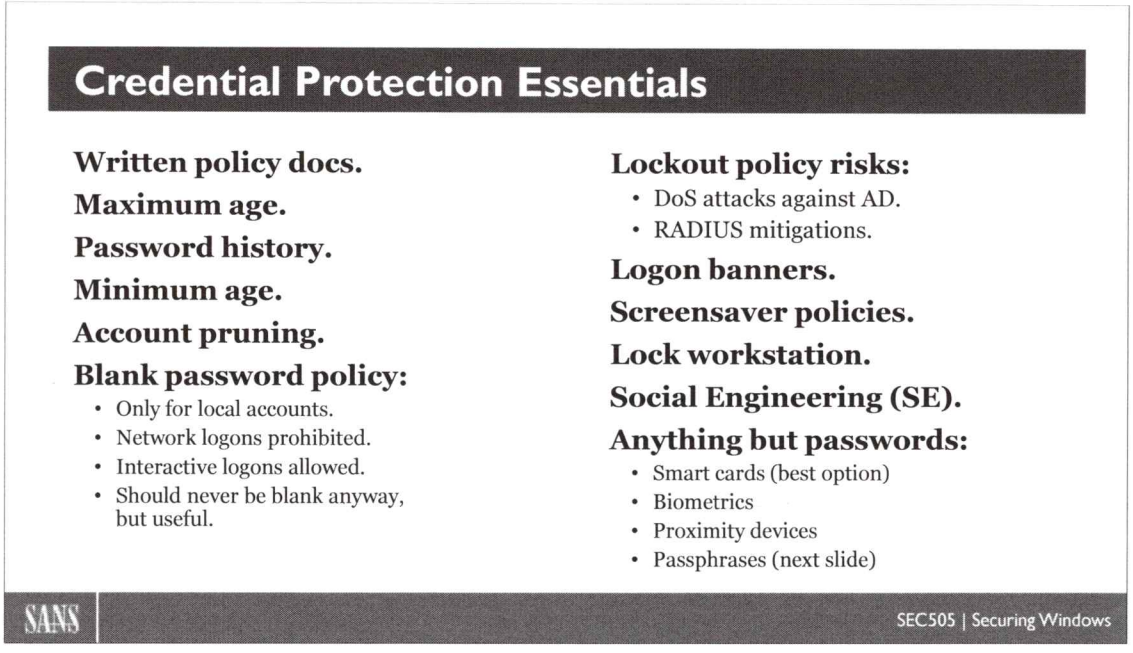
3. In your High CMD Shell, run "copy %systemroot%\notepad.exe High-Notepad.exe".

4. In your High CMD Shell, run "chml.exe High-Notepad.exe -i:h -nr -nw -nx", which will set the MIC label on High-Notepad.exe to High (-i:h) and deny read (-nr) and deny write/delete (-nw) and deny execute (-nx) to all other processes with lower MIC labels.

5. In your High CMD Shell, run "icacls.exe High-Notepad.exe". Notice that the MIC label for this file now says "High Mandatory Level:(NW,NR,NX)".

6. In your **Medium** CMD Shell, run "High-Notepad.exe". Note the error message.

7. In your Medium CMD Shell, run "icacls.exe High-Notepad.exe". Note the error message, you can't even read its properties!



Credential Protection Essentials

Your Security Access Token (SAT) will be filled with your groups and AD claims, but only after you log on. You must successfully authenticate first before you can do anything. Enforcing strong authentication requirements is critically important for securing Windows environments.

Passphrase and account lockout policies are set with Group Policy by default, but we'll see another way to manage these policies too. If you use Group Policy for this, you can only configure these options with a GPO linked at the domain level because all the DCs in a domain will only enforce one identical policy. This means there can be only one passphrase and lockout policy for the entire domain *if* you use Group Policy to manage these settings.

If you do configure passphrase or account lockout policies at the OU level, it only affects the *local* user accounts on the machines in that OU. The domain accounts in that OU are unaffected. So, yes, you can manage policy for local accounts, including the local Administrator and Guest accounts.

The following are recommendations for enforcing sound account policies.

Have A Written Passphrase Policy With Management Support

The following policy issues should all be laid out in detail in a written policy document. It should explain what the default settings should be, the rationale behind each setting, when exceptions can be made, when the policy must be more strict than the default, and, most importantly, the policy document must have management's stamp of authority.

Without management "buy-in" and official support, there will be endless complaints and resistance (often from management itself).

The SANS website has editable policy template files on a variety of topics that can be customized to your needs (http://www.sans.org/security-resources/policies/).

Account Policies > Password Policy > Maximum Password Age

Maximum Password Age determines how long a user can keep the same password. In a medium-security network, set this value to no more than 45 to 90 days. In a high-security network, no more than 30 days. But keep in mind that password length and complexity are far more important to enforce than a short password age. It is a misconception to believe that a short password age helps to prevent password cracking (unless your maximum password age is only one day). The purpose of this policy is to prevent a compromised password from being used *too long*, not to prevent its cracking. But a hacker with an administrative account only needs a few minutes (or seconds) to install trojans or back doors, so the focus should be on cracking prevention, not damage containment. You could safely allow users to keep their passwords for *months* as long as you enforce a minimum length of 20 or 25 characters, that is to say, as long as you require pass*phrases* instead.

Account Policies > Password Policy > Minimum Password Length

Minimum Password Length determines how few characters a password may have. Valid numbers are 0 to 14 in a GPO, even though Windows now supports passwords up to 127 characters in length. In a medium-security network, set this value to at least eight characters; in a high-security network, make it 14 characters and train users to use pass*phrases* instead. In actuality, though, you shouldn't enforce your password policy through GPOs, you should use a custom password filter or policy that requires at least 15 characters (more on this topic in the next section).

Account Policies > Password Policy > Password History

Password History specifies the number of prior passwords (up to 24) domain controllers should "remember" for each user account. When a user changes his or her password, the new password is compared against the list of that user's prior passwords. If the domain controller can "remember" that the new password has been used before, the user is forced to choose a different password. This policy is important because users will recycle their favorite passwords. Always set this option to 24 passwords remembered.

Account Policies > Password Policy > Minimum Password Age

The Minimum Password Age policy is used in conjunction with the Password History policy. The purpose of Minimum Password Age is to prevent users from cycling through enough password changes such that the domain controller will not "remember" their favorite password. Minimum Password Age compels a user to keep a new password for a period of time (up to 999 days). This prevents users from conveniently flushing out their password history list. Set this value to only one day on all networks. This prevents most cycling and there will be times when a user will have a valid reason for needing to change his or her password immediately.

Password Policy > Password Must Meet Complexity Requirements

This will be discussed in the next section. It ensures that the password has a variety of different types of characters. Keep in mind, though, that a long un-complex passphrase (20 characters or more) is vastly more difficult to crack than a short complex password.

Password Policy > Store Password Using Reversible Encryption

This will store another copy of the user's password with their account encrypted with a 3DES key derived from the master key of the DCs. This is required for IIS Digest authentication, RADIUS authentication, CHAP authentication (not MS-CHAP), and a few other scenarios. It is generally safe to enable this if necessary --assuming that DCs and their backups are physically secure-- but don't enable it if it will not be used. Keep in mind that "Advanced Digest" authentication in IIS 6.0 and later does not require this.

Limit Local Accounts With Blank Passwords To Console Logons Only

Windows XP and later has a security option named "Accounts: Limit Local Account Use of Blank Passwords to Console Logon Only". When enabled, if any local account on that machine has a blank password --including the local Administrator account-- then that account can only be used to log on to the machine while physically sitting at it, i.e., while logging in at the console interactively. This should always be enabled, but keep in mind that it only applies to local accounts.

Don't Forget About Local Administrator Accounts

An effective hacking tactic is to compromise a local administrative account on an undefended machine, then use that box to acquire domain credentials (perhaps by installing a sniffer or by extracting locally cached credentials). Hence, local accounts must be locked down as well. In a domain environment it's best to prevent the use of local account entirely. This should be done by assigning long passphrases to the local Administrator and Guest accounts, disabling Guest, renaming these accounts, and then deleting all others. Running a script is the best way to automate these changes and to document (in a secure location) the names and passphrases of the original Administrator accounts.

Disable The Guest Account

The Guest account is special, not because it is powerful, but because over-the-network users will be automatically logged on as Guest if 1) the Guest account is enabled, 2) the password for the Guest account is blank, and 3) the username supplied by the remote user does not exist in any accounts database to which the server has access, i.e., local, local domain or trusted domains.

The Everyone group includes the Guest account, but the Authenticated Users group does not include Guest, even if the Guest account has a password.

Windows XP and later has a security option named "Network Access: Sharing and Security Model for Local Accounts". When configured, this option will automatically demote to Guest status any remote user who authenticates to the machine with a local

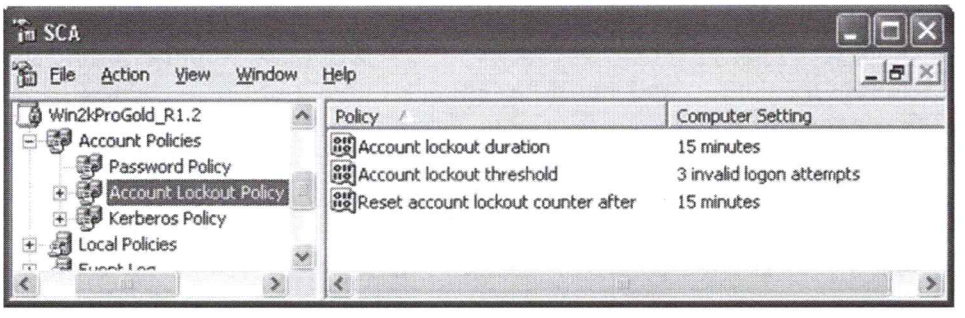
account on that machine. When users authenticate with their own domain accounts, the demotion does not occur. If you disable the Guest account and you enable this option, then no one will be able to log onto the machine with a local account (good thing).

Hence, disable the Guest account and assign it a long, random, non-blank passphrase; remove the Guest account from the Guests domain local group; remove the Guest account from the Domain Guests group; remove the Guest account from the Domain Users group (if it's there); if you've created an Untrusted Users group, add the Guest account to it; and, finally, enable the automatic demotion of local accounts to Guest through GPO on all computers.

Account Lockout Policy > Account Lockout Threshold/Duration/Reset

After a specifiable number of failed logon attempts, an account can be locked out. Be aware, there are known issues with lockouts triggering earlier than expected because of the way Windows sometimes uses both NTLM and Kerberos to log on (KB264678).

By default, the built-in global Administrator account in AD cannot be locked out by bad logon attempts, making it a prime target for brute-force guessing attacks. KB885119 describes how to use ADSI Edit to enable global Administrator account lockout, and, despite what the article implies, it still works on Server 2008-R2 and later. But be aware, with Server 2003 and later, the lockout applies to interactive logons while sitting at a domain controller too, which was not the behavior in Server 2000, hence, this feature is dangerous to enable.



Because malware or hackers could keep all accounts locked out indefinitely with continuous failed logons, your lockout policy is actually a liability. This is especially true if you are not using a RADIUS server with a lower lockout threshold than in AD and some of your exposed servers/devices do not use RADIUS, e.g., web servers, VPN gateways, wireless access points, dial-up servers, etc. When you add in the help desk costs and lost user productivity, it's better to not have a lockout policy at all.

Hence, either do not have a lockout policy at all for anyone (this is preferred), exempt all high-value target groups from lockout through custom lockout policy (second best), or at least keep the built-in Administrator account exempt from the lockout policy (third best). In an emergency, when all accounts are locked out, you'll still be able to log on as the global Administrator account and then unlock everyone else.

If an attacker can guess an administrator's passphrase after only a few million guesses, the problem is not the absence of a lockout policy, the problem is the short weak password or lack of smart cards. If a lockout policy is required, then configure a threshold of perhaps 50 failed authentications triggering only a 5-minute lockout; though, this will still not prevent malware or hackers from keeping accounts locked out indefinitely, they'll just have to fail to logon a little faster.

The local Administrator account should be assigned a long passphrase and disabled anyway, so the lockout policy for local accounts doesn't matter much. When laptops are stolen, hackers will try to reset local account passwords, they don't try logging on over-and-over again or cracking hashes. Whole drive encryption with a TPM plus UEFI Secure Boot are the best defenses in this case.

With a network- and host-based Intrusion Detection System (IDS) in place, you should receive alerts when being subjected to password-guessing attacks too.

Account Lockout Duration
When an account is locked out, it can either be locked out forever (set to zero) or for a specifiable number of minutes (up to 99,999). If the account is locked out forever, users will have to contact an administrator and request that the account be re-enabled.

Reset Account Lockout
Domain controllers keep a counter of bad logon attempts for each account. This counter can be reset to zero after a specifiable number of minutes (1 to 99,999). This is the maximum amount of time that can transpire between bad logons and yet still trigger account lockout when the threshold is reached. For example, if accounts are locked out after five bad logon attempts, and a user has already failed four times, the user should wait until the counter of bad logons is reset to zero before trying again. If the user is a hacker, he or she will not know what the lockout threshold is and will quickly cause the account to lock.

Display A Logon Banner With A Legal Notice

A logon banner doesn't deter determined hackers, but it may help you if you end up in a court of law trying to convict or sue people who have caused damage. The exact wording of the logon banner is important. Have legal personnel in your organization review its text before deployment. The following is a good banner to begin with. A logon banner is configured under Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options > Message Text For Users Attempting To Log On.

> This system is for the use of authorized users only and is not public. Individuals using this computer system without authority or in excess of their authority are subject to having all of their activities on this system monitored and recorded, including their keystrokes and mouseclicks. Anyone using this system expressly consents to such monitoring and is advised that if this monitoring reveals possible evidence of criminal activity, this evidence may be provided to law enforcement officials with the intent to prosecute.

Require Screensavers With Passwords

"Station hopping" is the trick of waiting until a logged-on user walks away from his or her machine, then jumping on to work with the victim's credentials. The lack of short-timeout and password-protected screensavers is also bad for maintaining non-repudiation in the office. You can require a particular screensaver, a password entered to unlock it, and a short timeout. Timeout should be between 10 and 50 minutes, depending on security requirements and office politics. This is configured with options found under User Configuration > Policies > Administrative Templates > Control Panel > Display.

Train Users To Lock Workstations (Or Use Proximity Devices)

Users don't like to log off because it takes time and it closes their applications. Some users are unaware that you can press Ctrl-Alt-Del while logged on and lock the desktop. One can also depress the Windows key plus "L" to lock the desktop. A locked workstation has no desktop, but the user's open files and applications are still running in the background. To retrieve their desktop, a user must enter their regular password. A locked workstation can also be unlocked by an administrator.

Users should still be encouraged to log off at the end of the day however.

Local Policies > Security Options> Smart Card Removal Behavior

If a user logs on with a smart card, then pulls the card out while still logged on, you can have Group Policy either forcibly log off the user or lock the desktop (just as though the user had hit Ctrl-Alt-Del > Lock Computer). Set it to "Lock Workstation".

Perform Regular Account Pruning

At least once per month, search your directory for accounts that tend to be soft or frequent targets. These include accounts whose passwords have not been changed in a long time (perhaps in twice the length of your maximum password age), that no one has used to log on with in that period, whose comments or descriptions give hints for their passwords, and any accounts named the following: test, demo, backup, admin, root, administrator, anonymous, backup, temp, secretary, user, repl, student, lab, job, public, ftp, batch, guest, *servicename*, *computername*, or *companyname*. A single PowerShell script could be scheduled to run each week that e-mails its search results for these accounts to you for review.

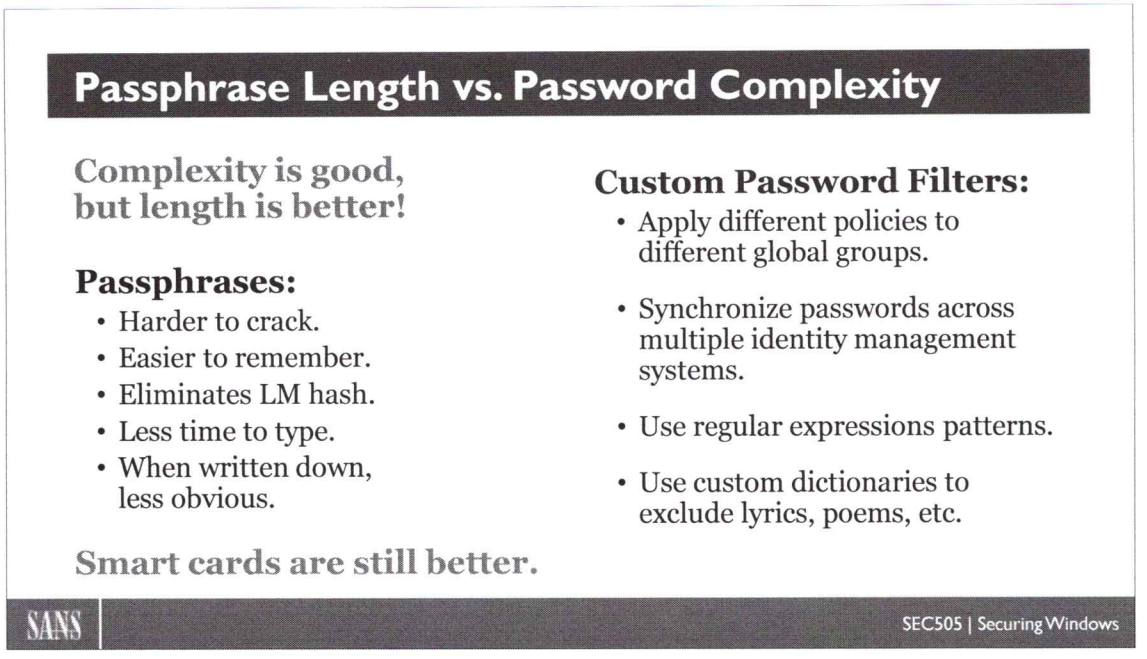
Educate Administrators And The Help Desk About Social Engineering

"Social Engineering (SE)" is using fraud and deception to trick well-meaning people into doing things bad for network security. SE is a widespread threat that is consistently ignored or underestimated because it's not technically sexy and because it doesn't fit management's image of what hacking is (i.e., it's not what you see in the movies). However, SE is and always will be the easiest way to penetrate a hardened network. Human beings will always be the weakest layer in the OSI model. See the Appendix on Social Engineering for more information, as well as inexpensive defenses against it...which will fail, but we have to at least try.

The Best Password Policy Would Be "No Passwords!"

Password authentication is the weakest authentication method in widespread use today. The best password policy is to not permit their use at all! At least for your high-value accounts and machines, try to require anything other than password authentication, e.g., smart cards, biometrics, two-factor tokens, long passphrases, etc.

But how to enforce a *passphrase* policy?



Passphrase Length vs. Password Complexity

The longer and more complex a password, the longer it takes an adversary to guess it by brute force. Built into Windows is a password complexity filter that enforces rules for strong passwords when users change them. The built-in password filter will require that changed passwords be at least six characters long, not contain any part of the user's full name, and contain at least three out of the four following categories of characters:

- Uppercase letters.
- Lowercase letters.
- Numbers.
- Non-alphanumeric symbols.

When a user account is first created, its password can be non-complex or blank. Existing account passwords can also remain non-complex or blank. However, once the password filter is enabled, when any password is *changed*, the password must meet the filter requirements.

Password complexity filtering is enabled at the domain level under Computer Configuration > Policies > Windows Settings > Security Settings > Account Policies > Password Policy > Passwords Must Meet Complexity Requirements.

When enabled on regular computers, it only affects local account passwords.

Passphrase Advantages

Windows supports passwords up to 127 characters, but Group Policy does not permit requiring a length longer than 14 characters. And yet the length of a password is much

more important than its complexity. A simple and effective password filter would require, for example, a 25-character or longer pass*phrase*. A passphrase is like a password, but it contains space characters so that a meaningful sentence can be formed.

Users actually *prefer* long but meaningful passphrases over short and random passwords. Consider, between the following two "passwords", which would a user rather memorize?

1. I love my kitty she's 4 years old [33 characters]

   -or-

2. ?cPe1704TKs<!# [14 characters]

Both passwords satisfy complexity requirements, but the passphrase would require billions more guesses in order to crack it by brute force. The passphrase is actually uncrackable in any reasonable length of time even though every word is in the dictionary.

If the attacker knew that user-friendly sentences were being used, then a somewhat-grammatically-correct-phrase-generator could be used instead, but where are these tools? Has anyone *ever* seen a cracker that generates quasi-meaningful "complex" passphrases longer than 25 characters? Also, even if such a tool cut the search time down by 75% (an optimistic assumption) most long passphrases could still not be cracked in any reasonable period of time.

For example, if a 30-character passphrase could only use lowercase letters (26 possible characters) and a 14-character password could use any standard valid character (72 possible characters), these are the possible combinations an unoptimized brute force search would face:

- 30-character letters-only passphrase: 2.8×10^{42} possible combinations.
- 14-character random password: 1.0×10^{26} possible combinations.

Also, if 30-character passphrases were required, you could safely allow users to keep them longer. Users are also much less likely to write down meaningful passphrases than random passwords. And, as an experiment, type the two passwords listed above into Notepad on your laptop. Which took less time to type? Which had more typos that had to be backspaced?

When creating a passphrase, start by imagining a visual scene with action in it that is funny, shocking, stunning, weird, or sexually charged, then create a short sentence that describes or relates to that scene in some way. The more outrageous the scene is, the easier the passphrase will be to remember and the more likely you'll use uncommon words to describe it. (This advice all comes straight from *The Memory Book*, by H. Lorayne and J. Lucas, which is a classic guide for improving one's memory skills.) Also, the worse your grammar be and the more misspelt wurds you use the better! (Bad spelling and poor grammar finally pay off! Always have at least one misspelled word.) If you know a foreign language or have an interest with a specialized vocabulary, then mix those words into the passphrase too (notice that this is the opposite of the

recommendation for passwords). Finally, consider appending a single blank space to the end of your passphrase as a touch of obscurity if you write it down or a keystroke logger captures it, and if you include the hat character (" ^ ") in your passphrase it might help to prevent the use of the passphrase in shell scripts.

The drawback to using passphrases is the lack of backwards-compatibility with older applications and operating systems. Also, many web-based applications are not coded to support long passphrases. On the other hand, the later might be an advantage: users tend to enter their current domain password whenever some third-party website or form requires a new password from them. If there are accounts which must have shorter passwords, an administrator can set them manually (if there are not too many).

Custom Filters

You can write your own custom password filter to replace the default PASSFILT.DLL. A custom DLL receives a cleartext copy of the proposed password before it is accepted, hence, it can be checked in any way desired: extended ASCII characters, against a list of weak passwords, against a policy based on the user's group memberships, anything you want (the DLL will be written in C++).

A number of commercial password filters also synchronize the user's password with an external accounts database such as Novell NDS, an LDAP server, a Unix passwd file, etc.

Once compiled, the DLL should be saved in the %SystemRoot%\System32 folder on all DCs. Next, add the name of the DLL file to the following registry value on all DCs with REGEDT32.EXE, and reboot (KB161990).

Hive:	HKEY_LOCAL_MACHINE
Key:	\System\CurrentControlSet\Control\Lsa
Value Name:	Notification Packages (notice the space character)
Value Type:	REG_MULTI_SZ
Value Data:	*<name of DLL, not including the ".dll" extension>*

Commercial, Freeware and Other Password Filters

The following is a partial list of the password complexity filters available. Note that your organization can hire a C++ developer to write a custom filter for you, and this will not be expensive as complexity filters are not very complex. If your organization has in-house C++ coders, they will know what to do just by browsing some MSDN articles.

- PASSFILT (http://www.microsoft.com) -- Built into Windows. The source code for a sample Microsoft filter can be obtained from the free Windows Platform SDK under \Samples\Winbase\Security\Winnt\Pwdfilt.

- Specops Password Policy (http://www.specopssoft.com) -- Commercial filter with support for regular expressions, dictionaries, Group Policy, and PowerShell.

- nFront Password Filter (http://www.nfrontsecurity.com) -- A simple to deploy, relatively inexpensive, and very capable commercial filter.

- Hitachi ID (http://hitachi-id.com/password-manager/) -- Provides password synchronization across non-Windows servers, user self-service password reset, complexity filters, audit password-related management, and more.

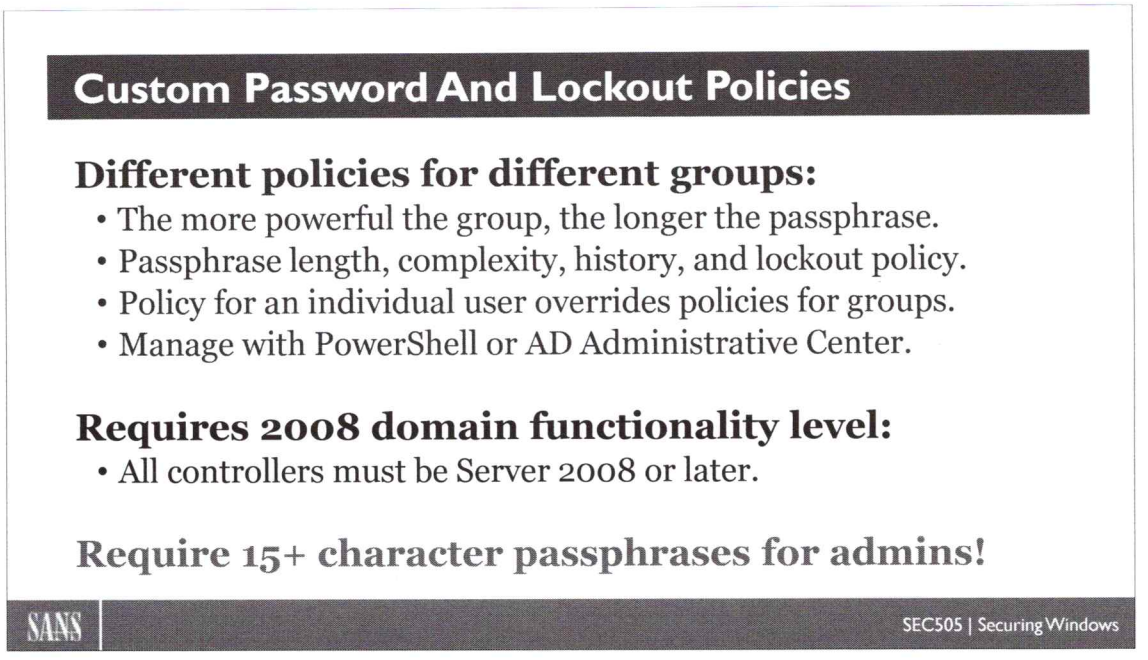
- Password Policy Enforcer (http://www.anixis.com) -- Full featured password management system, including the ability to set different policies for different individual users.

- PasswordFilter (http://www.denglernet.de) -- Source code available for free under GPL license, password complexity enforced, then synchronizes password with an LDAP server.

- Dictionary Password Filter (http://ebs-ebs.tripod.com/passwordfilter.html) -- Freeware filter, with source code, for doing standard complexity filtering and dictionary file look-up.

Trojan Horse Filters

Beware of Trojan Horse password filters. A custom PASSFILT.DLL can write passwords in cleartext to a file or transfer the passwords across the network. Take care to assign NTFS permissions to your filter DLL so that only Administrators and the System account can access it. Also set the same permissions on the registry key above.



Custom Password And Lockout Policies

A user is powerful based on his or her group memberships, not domain membership, yet oddly Microsoft has in the past only permitted password and lockout policies to only be applied to entire domains. What's more, you could only require a maximum character length of 14 characters in passphrases despite the importance of using 15+ characters for security. Starting with Windows Server 2008, however, you can apply a custom policy to each global group or user in Active Directory, and you can require more than 14 characters in passphrases.

Note: Domain functionality level must be at Server 2008 or better.

A custom password/lockout policy for a global group or user includes:

- Whether a reversibly encrypted copy of the user's password is kept in AD.
- Password history.
- Password complexity.
- Minimum password length.
- Minimum password age.
- Maximum password age.
- Lockout threshold.
- Lockout observation window of time.
- Lockout duration.

If a custom password/lockout policy is assigned to a particular user, this policy will override any such policies inherited from that user's group memberships which also have

custom policies applied to them. Custom password/lockout policies override those defined for the entire domain through Group Policy.

You cannot apply custom policies to OUs directly; they can only be applied to global users and global groups (and not to computer accounts).

If you already have a password filter DLL in place from Microsoft or a third-party developer, then these custom password/lockout policies can co-exist with the filter without problems.

Manage Custom Policies (Server 2012 and Later)

Custom password and lockout policies can be managed using the Active Directory Administrative Center (ADAC) on Server 2012 and later. ADAC can be launched from the Tools menu in Server Manager.

> ***Try It Now!***
>
> To manage custom password and lockout policies on Server 2012 and later, open Active Directory Administrative Center > click the Tree View tab in the upper left > expand your domain > System > Password Settings Container > double-click an existing policy or click New to create a new password settings object.

Third-Party Tools To Manage Custom Policies (Server 2008 and Later)

If you're not thrilled with the built-in tools for managing custom password policies, check out the following free tools instead:

- http://www.specopssoft.com/products/specops-password-policy
- http://www.joeware.net/freetools/tools/psomgr/index.htm
- http://www.powergui.org/entry.jspa?externalID=882&categoryID=46

Manage Custom Policies with PowerShell (Server 2008-R2 and Later)

Starting with Server 2008, it is possible to assign different custom password and lockout policies to different users and groups. With 2008-R2 and later, you can both query and reconfigure these fine-grained policies with PowerShell.

To display the default password and lockout policy for the domain of the currently logged-on user as shown in the Default Domain Policy GPO:

```
# C:\SANS\Day4-Admins\Custom_Password_Policies.ps1

get-addefaultdomainpasswordpolicy -current loggedonuser
```

To change the default password and lockout policy for the domain of the currently logged-on user as shown in the Default Domain Policy GPO:

```
$mydom = get-addomain -current loggedonuser
```

```
set-addefaultdomainpasswordpolicy -id $mydom -minpasswordlength 5
```

To list all your current fine-grained password policies in AD:

```
get-adfinegrainedpasswordpolicy -filter {(name -like "*")}
```

To create a new fine-grained password policy named "SalesGroupPwdPolicy" (parameters have been wrapped to new lines for easier reading):

```
new-adfinegrainedpasswordpolicy -name "SalesGroupPwdPolicy"
      -Precedence 700
      -LockoutThreshold 50
      -LockoutDuration "0.00:10:00"
      -LockoutObservationWindow "0.00:10:00"
      -MaxPasswordAge "90.00:00:00"
      -MinPasswordAge "1.00:00:00"
      -MinPasswordLength 17
      -PasswordHistoryCount 24
```

The -Precedence parameter above is an arbitrary number chosen to assign relative priorities to fine-grained password policies (ensure that no two policies have the same precedence number) when multiple policies would apply to the same user. The policy with the lower precedence number has a higher priority; for example, if PolicyBlue has a precedence of 300 and PolicyRed has a precedence of 950, and user Amy is subject to both policies, then PolicyBlue will be the effective policy for Amy.

The number format above is "*days.hours:minutes:seconds*", hence, "90.12:30:20" would be 90 days, 12 hours, 30 minutes and 20 seconds.

To modify the existing SalesGroupPwdPolicy object with a new MaxPasswordAge:

```
set-adfinegrainedpasswordpolicy -identity salesgrouppwdpolicy
      -maxpasswordage "120.00:00:00"
```

To add the Sales global group to the scope of the SalesGroupPwdPolicy object, as well as Susan, Jon, Aaron and Zach (separate multiple items with commas):

```
add-adfinegrainedpasswordpolicysubject -id SalesGroupPwdPolicy
    -subjects Sales,Susan,Jon,Aaron,Zach
```

To show the fine-grained password policy named "SalesGroupPwdPolicy":

```
get-adfinegrainedpasswordpolicy SalesGroupPwdPolicy
```

To show the subjects of the SalesGroupPwdPolicy object:

```
get-adfinegrainedpasswordpolicysubject -id SalesGroupPwdPolicy
```

To remove only Zach from the subjects list of the SalesGroupPwdPolicy object:

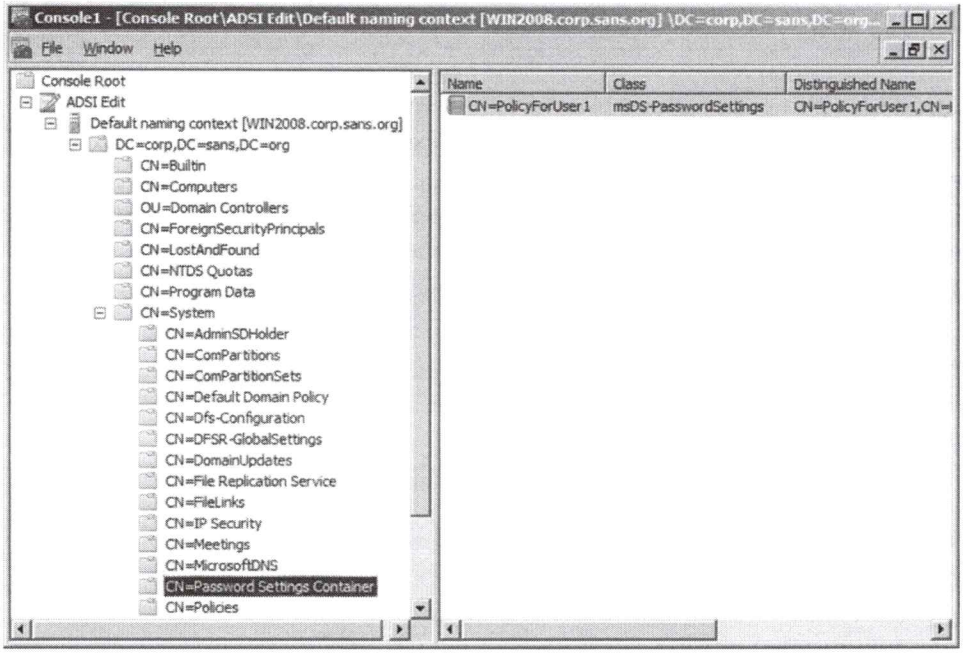
```
remove-adfinegrainedpasswordpolicysubject -id SalesGroupPwdPolicy
    -subjects Zach
```

To delete the SalesGroupPwdPolicy fine-grained policy completely:

```
remove-adfinegrainedpasswordpolicy -identity SalesGroupPwdPolicy
```

Active Directory Location of Settings

A custom password/lockout policy for a user or global group of users is actually an object of type msDS-PasswordSettings (a class in the schema). These objects must be created in the AD container named "DC=<*YourDomain*>,CN=System,CN=Password Settings Container". You can view this container with the ADSI Edit snap-in.



Only Domain Admins have the necessary permissions on the Password Settings objects under the System container to manage custom password/lockout policies. You can delegate this authority to others by granting the necessary read-write permissions on the Password Settings objects themselves, but you don't have to change any permissions on the target groups or users.

Import LDF File to Manage Settings (Server 2008 and Later)

Another way to add a custom password/lockout policy for a user or group is to edit an LDF file and import it into Active Directory with the built-in LDIFDE.EXE tool. The file would be formatted like the following, except that you would change the DN paths for your domain ("dn:" field) and the target user/group ("msDS-PSOAppliesTo:" field).

Note: If you received a CD-ROM from the instructor, an appropriate LDF file is located in the folder for this day of the course (Custom_Password_Policy.ldf). However, you must edit the DN paths for your domain ("dn:" field) and the target user/group ("msDS-PSOAppliesTo:" field) in the file first.

```
# Don't forget to change the DNs to match your domain!

dn: CN=Policy1,CN=Password Settings
Container,CN=System,DC=sans,DC=org
changetype: add
objectClass: msDS-PasswordSettings
msDS-MaximumPasswordAge:-38880000000000
msDS-MinimumPasswordAge:-864000000000
msDS-MinimumPasswordLength:15
msDS-PasswordHistoryLength:24
msDS-PasswordComplexityEnabled:FALSE
msDS-PasswordReversibleEncryptionEnabled:FALSE
msDS-LockoutObservationWindow:-3000000000
msDS-LockoutDuration:-3000000000
msDS-LockoutThreshold:5
msDS-PasswordSettingsPrecedence:10
msDS-PSOAppliesTo:CN=SomeGroup,CN=Users,DC=sans,DC=org
```

To import the above file into AD, run the following command in a privilege-elevated shell while logged on as a member of the Domain Admins group:

```
ldifde.exe -i -f filename.ldf
```

When you use LDIFDE.EXE to add a custom policy, you must enter the values of the time-related attributes in "I8" format, which are in intervals of 100 nanoseconds and must be entered here as a negative number. I know this seems weird and strange, but that's the way it is.

Hence, the following time-related attributes should have their total minutes, hours and days converted to a sum of 100ns units and specified as a negative number:

- msDS-MaximumPasswordAge
- msDS-MinimumPasswordAge
- msDS-LockoutObservationWindow
- msDS-LockoutDuration

To convert a number of minutes, hours or days to a negative number of 100ns units, use the following multiplication table:

To Get:	Multiply By:	Example:
Minutes	-600000000	20 minutes = -12000000000
Hours	-36000000000	5 hours = -180000000000
Days	-864000000000	30 days = -25920000000000

There are some constraints to keep in mind when setting these values:

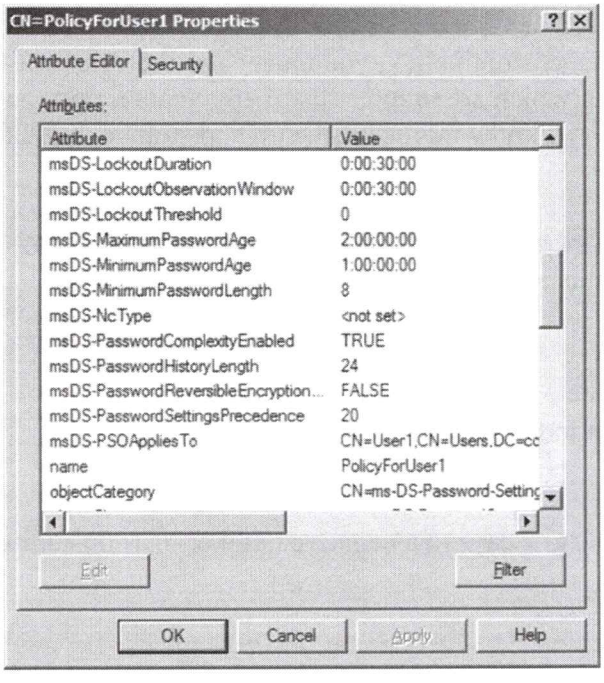
- The absolute value of msDS-MinimumPasswordAge must be smaller than or equal to the value of msDS-MaximumPasswordAge.
- The absolute value of msDS-LockoutObservationWindow cannot be smaller than the value of msDS-LockoutDuration.
- The value of msDS-MaximumPasswordAge cannot be zero.

To disable account lockout policies, set the msDS-LockoutThreshold attribute to zero.

For more information about LDIFDE.EXE, see KB237677, KB555636 and KB555637.

Editing Existing Custom Policies With ADSI Edit

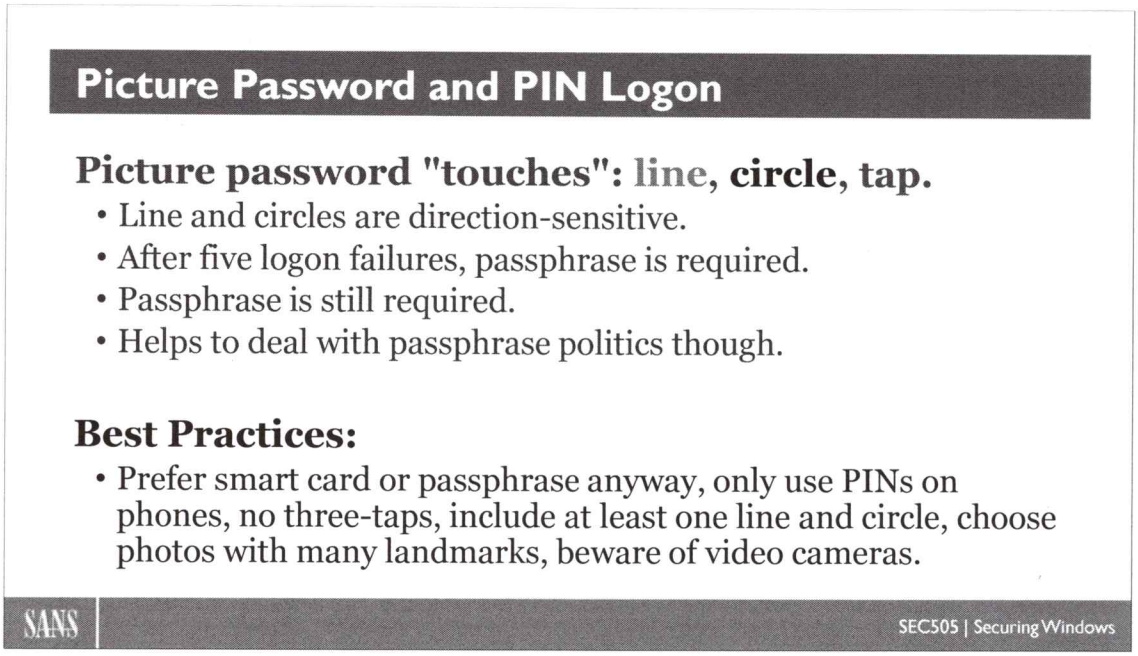
Once the msDS-PasswordSettings object is created using an LDF file, you can modify it with another LDF file, but it's generally easier to use the ADSI Edit snap-in. This is especially true when adding a few more DN paths to the msDS-PSOAppliesTo attribute, which contains one or more distinguished name paths to the users and/or groups to which the policy should apply (yes, you can add more than one).



Try It Now!

To edit a custom password/lockout policy with ADSI Edit, run MMC.EXE > File menu > Add/Remove Snap-In > double-click ADSI Edit > OK > right-click ADSI Edit > Connect To... > OK > expand the Default Naming Context container and navigate down to "CN=System,CN=Password Settings Container". You will see your custom password/lockout policy object(s) on the right-hand side. Right-click a policy object > Properties.

Notice that when using ADSI Edit you can specify the time-related attributes using "DD:HH:MM:SS" format instead of the large negative (I8) numbers.

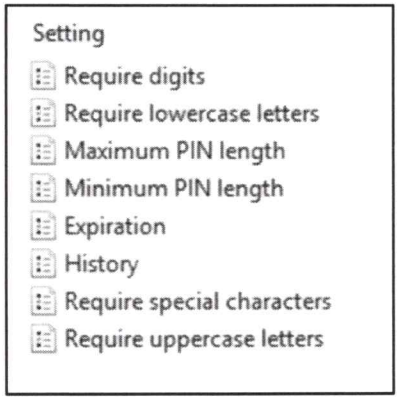


Picture Password and PIN Logon

Windows 8 introduced two new authentication options: Picture Password and PIN Logon.

"Picture Password" is the option to log on with a pattern of three touches on the wallpaper of the logon screen on a touch-sensitive device running Windows 8 or later. The three touch types are tap, circle, and line. The line and circle gestures are directionally sensitive. The wallpaper photograph is organized into a hidden grid of approximately 100x100 squares for the sake of registering the touches.

Windows 8 also introduced the option to log on with a 4-digit numeric PIN. On Windows 10 and later, the "PIN" may also include alphanumeric characters, and there are associated PIN complexity rules in Group Policy for it (GPO > User Configuration > Policies > Administrative Templates > Windows Components > Windows Hello for Business > PIN Complexity). This is separate from the domain password policies.



PINs may be between 4 and 127 characters in length.

Picture passwords and PIN logons are disabled with the following registry values:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\System]
"AllowDomainPicturePassword"=dword:00000000
"AllowDomainPINLogon"=dword:00000000
```

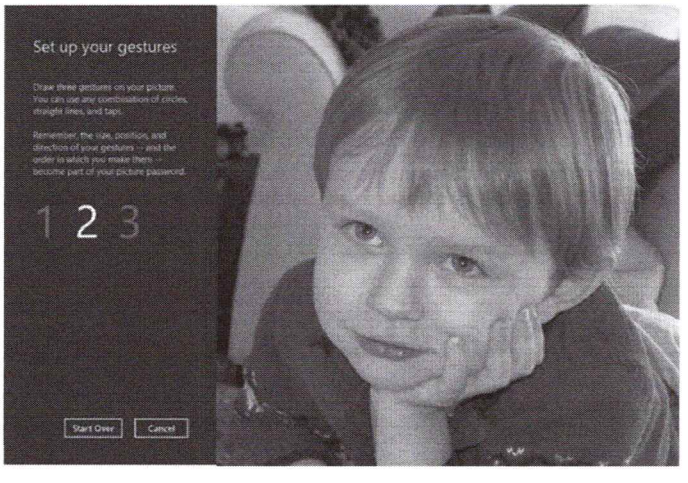
Importantly, a picture password or PIN is not a total replacement for a passphrase. A user will still have a passphrase and will still have to enter it under some circumstances, such as their first logon to a machine. A picture password or PIN can only be created after an initial successful logon with a passphrase.

Successful picture or PIN logon only unlocks a locally-cached and encrypted form of the user's passphrase from the Data Protection API (DPAPI) "Vault" for that user, similar to cached credentials for when domain controllers are not available.

> **Note:** To read more about the DPAPI vault, start with the following article because it is also important to know about a SID issue when troubleshooting: http://technet.microsoft.com/en-us/library/ee681624%28WS.10%29.aspx

If picture logon fails five times in a row, the user is not permitted to use picture logon again until after a successful logon is performed using another method, such as their passphrase. Unless a user calls the help desk, they will still need to know the passphrase.

Also, picture logon is only available for interactive logons, not for over-the-network or remote access authentications, hence, the user will still need to know their passphrase for these situations too.



Picture logon can help to make implementing a passphrase policy more politically acceptable. A user must still memorize their passphrase, but emphasize to users that the picture method can be used for interactive logons to their computers 99% of the time.

The passphrase will only be needed for network and remote access authentications where single sign-on is not possible.

On tablets and phones with no physical keyboard, enforcing a passphrase policy will be almost impossible because of the political opposition (would your manager or CEO tolerate this on his or her own tablet?). But a passphrase policy just might squeeze through after a few live demonstrations of picture logons first.

> **Note:** If you require a Ctrl-Alt-Del secure logon sequence, and a Windows phone or tablet lacks a keyboard, try pressing the Windows button and the power button simultaneously. Not all OEMs support this however.

Cryptanalysis and Security

Microsoft lays out the mathematics of PIN and picture gesture possibilities in the following two posts and makes a good argument that the average picture logon sequence has about the same number of possible combinations as the average 5- to 6-character password with *random* characters (which is pretty good, though not as good as a 15-character passphrase with a misspelled word in it):

- http://blogs.msdn.com/b/b8/archive/2011/12/16/signing-in-with-a-picture-password.aspx
- http://blogs.msdn.com/b/b8/archive/2011/12/19/optimizing-picture-password-security.aspx

Keep in mind that a picture password, PIN or biometric authentication merely unlocks access to a plaintext copy of the user's full passphrase. The user is still logging on with their normal passphrase, it's just that entering the correct picture password, PIN or biometric will trigger the operating system to enter the user's passphrase for him or her automatically. So where is the user's passphrase stored then? How is it encrypted?

The user's passphrase is secured with the Data Protection API (DPAPI) under the computer's account context, not as any specific user, since no user has logged on yet. DPAPI encrypts computer secrets and stores them in files found here:

> C:\Windows\System32\config\systemprofile\AppData\Local\Microsoft\Vault\4BF4C442-9B8A-41A0-B380-DD4A704DDB28

On Windows 10 and later, if the device has a TPM chip, the TPM assists in the encryption of the data. The details are unpublished however.

Unfortunately, if one's device lacks a TPM, any hacker or malware which can execute commands as Local System or with Administrators privileges can extract all computer-context DPAPI secrets in plaintext, including the plaintext passphrases of users who have picture passwords or PIN logons defined on that computer.

Moreover, because of the way computer-context DPAPI protection works, anyone with offline access to the file system on a device without a TPM can recover the passphrases of other users too. To try to prevent this, it is best to have whole disk encryption and UEFI Secure Boot enabled.

To be fair, though, if hackers or malware can execute elevated commands, or if the stolen drive is unencrypted, then the system was compromised anyway. The extra danger comes from *shared* computers. On a shared computer with multiple users employing picture passwords and PIN logons, a malware infection of *one* user reveals the passphrases of *all* the users sharing the computer.

PIN Best Practices

Best practices for PIN logons are relatively simple:

- Do not allow administrators or other high-value targets to use PIN logons.
- Enforce PIN complexity policies on Windows 10 and later devices.
- Implement whole drive encryption, preferably with UEFI Secure Boot and a TPM, to help protect any keys or ciphertext on the drive which are derived from the picture logon process. Normal system hardening is recommended also to help ward off malware which could compromise picture logon security, e.g., patch management, AV scanner, firewall, etc.
- Do not use PIN logons (or picture passwords) on shared devices.
- Train users to choose PINs which do not include any sequences from their phone numbers, the phone numbers of any family members or associates, Social Security numbers, birthdates, addresses, the number 1234, any one digit repeated four times, or any two digits repeated twice, such as 1212 or 6969.
- If possible, enforce the same PIN restrictions users are trained to follow, but your enforcement options will depend greatly on your choice of device and available infrastructure.
- Train users not to use the same PIN across multiple cards or devices.

Remember, if PIN logon fails five times in a row, the user is not permitted to use PIN logon again until after a successful logon with another method.

Picture Password Best Practices

Prefer a smart card over a passphrase, a passphrase over a complex password, and a complex password with more than 9 characters over a picture password. Hence, the following best practices are for when picture logons will be permitted either for political

reasons (as part of a deal to enforce a good passphrase policy) or when better options are not yet possible (such as smart cards).

- Leave the picture password option disabled on domain-joined computers, which is the default, except on touch-only devices when there is no other practical alternative. High-value target users should never use picture password.

- Implement whole drive encryption, preferably with UEFI Secure Boot and a TPM, to help protect any keys or ciphertext on the drive which are derived from the picture logon process. Normal system hardening is recommended also to help ward off malware which could compromise picture logon security, e.g., patch management, AV scanner, firewall, etc.

- Do not use picture passwords (or PIN logons) on shared devices.

- Remember, if picture logon fails five times in a row, the user is not permitted to use picture logon again until after a successful logon with a passphrase (PINs would not be allowed in this case), so enforce a good passphrase policy.

- Train users to never just use three taps as the picture logon sequence, which is what they will all prefer to do. The sequence should include at least one line and at least one circle. Try not to make every circle go in a clockwise direction. Try to make some circles at least as large as the palm of your hand. Try not to make every line go from left to right (or whatever would be the dominant choice in your culture). There are no Group Policy options to enforce "touch complexity."

- Train your users to understand that the choice of photograph is important. Choose a photograph which includes more than ten easily-selected landmarks spread across the photo. Users will tend to choose family photos and just tap eyes or noses. A better family photo might include multiple people, some wearing jewelry, some holding things in their hands, wearing hats, wearing distinctive clothing, and with distinct items visible in the background. In a photo like this there are many more easily-identifiable landmarks to choose from, which makes it more difficult for attackers to guess which points are being used.

- Train users to not use the same photo on their work computer as they do on any of their personal computers or phones. If the photos are the same, the touch pattern will likely be the same too.

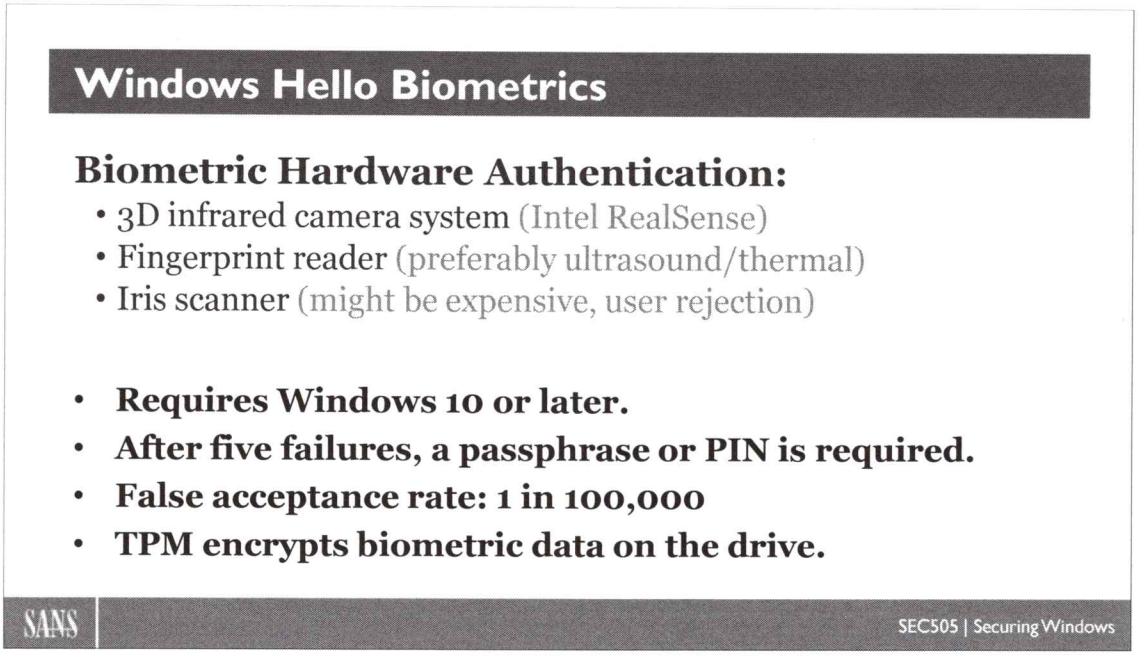
- If you choose the photograph for your users, choose a photo with dozens of landmarks, perhaps using a grid of funny icons scattered around the edges, and avoid including any human or animal faces. (Good luck getting management control over the photo, this will be *very* politically difficult.)

- Except in rare circumstances, analyzing finger smudges on the screen of a tablet is not a feasible way to reduce the number of touch attempts to less than five, but if

this is a concern, then choose at least one point on the photo near the edges where one often performs swiping. Don't train users to clean the screen, it doesn't improve "smudge security" (in fact, it makes it worse) and they'll never do it consistently anyway.

- Train users not to allow others to observe their picture logon touch sequence. For high-value targets, this includes training them to worry about video cameras when sitting in hostile environments. If a video camera or other observer is suspected, you can switch back to passphrase authentication at the time of logon and take care to cover the keyboard. If passphrase, PIN and picture logon are all configured on a computer at the same time, the user chooses which one to use at logon.

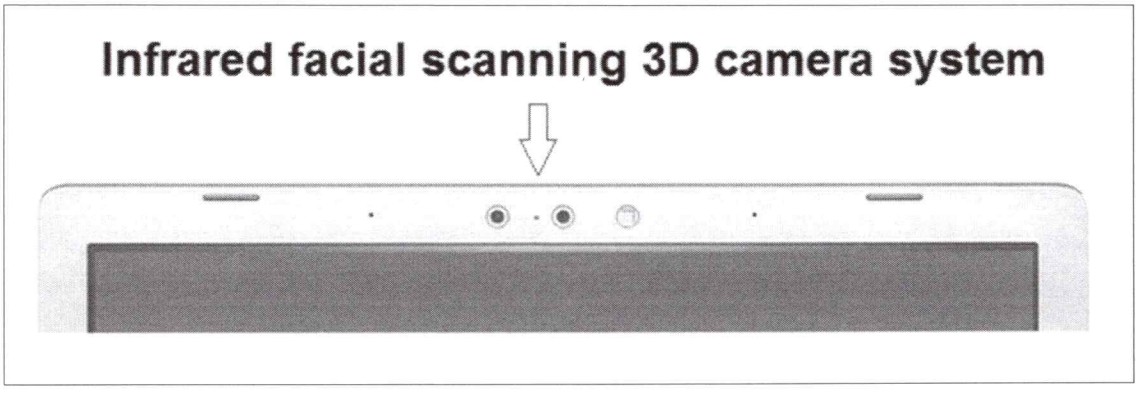
- Incidentally, assuming you will allow picture logons at all, using a line gesture is better than a circle because a line has two different points. Using a circle is more secure than a tap because the diameter of the circle and the direction of the circle are variables. The tap gesture is the least secure because it's just a single point on the grid. Hence, prefer lines over circles, circles over taps.



Windows Hello Biometrics

Windows 10 introduced a new biometric framework to allow authentication using an infrared facial scan, iris scan, thumbprint, or another third-party biometric module that uses that framework. Collectively, Microsoft refers to these authentication options as "Windows Hello".

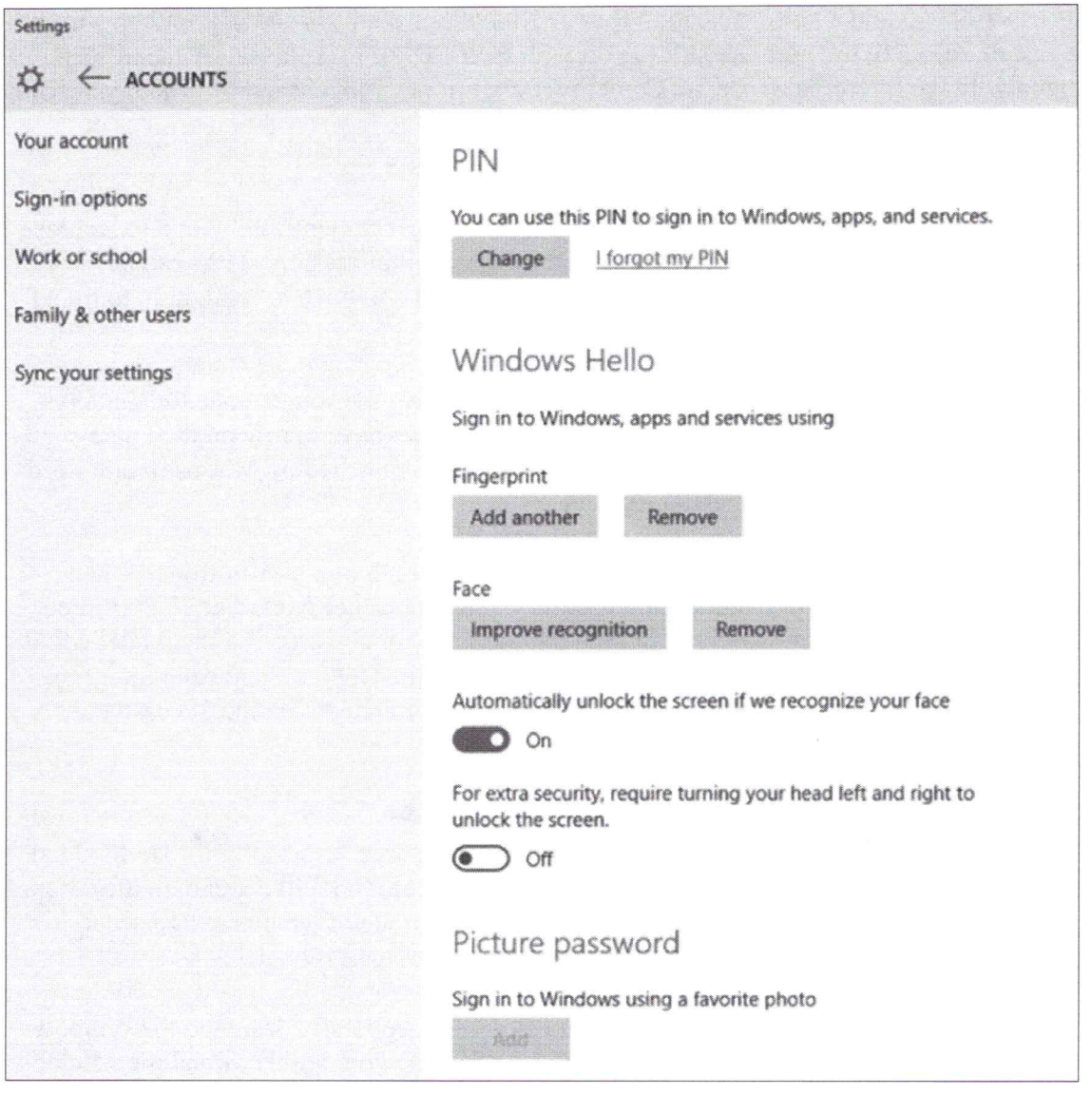
> **Note:** Windows Hello is not the same thing as Microsoft Passport, also first introduced with Windows 10, though they are related: after logging on to the desktop using Hello, a user can then use Passport credentials for single sign-on to FIDO-compatible web sites and services, such as with Edge or a universal app.



Biometric authentication requires special hardware. A three-dimensional facial geometry scan will require a multi-camera infrared system, such as the Intel RealSense 3D Camera,

built into the top edge of the laptop, tablet or phone. An iris scan will require an iris scanner, a thumbprint scan will require a reader, etc. Windows Hello is also compatible with ultrasound and/or thermal thumbprint readers. Ultrasound readers examine both the surface thumbprint and some of the underlying tissue and bone too. For iris scanners, be aware that the good quality scanners can be expensive, and users might reject iris scanning because they feel it is too invasive or dangerous to their retinas.

To configure Picture Password, PIN Logon or Hello on Windows 10 and later, open All Settings > Accounts > Sign-In Options.



Enabling one of these authentication options will launch a wizard to walk the user through the setup and enrollment process. Importantly, the user must perform this setup process on each device separately; the biometric or PIN data is not synced to any other device (or, at least, that's what Microsoft says). Hello is never used to authenticate over

the network, it is for local logon only. If the user's device has a Trusted Platform Module (TPM) chip, then the TPM encrypts the credentials data.

A user can enroll multiple fingers, iris scans, or facial scans on a device. These would not be used simultaneously to perform a logon, the various scans are recorded and can be used separately. Each scan is considered equally valid and sufficient for logon. A user might enroll multiple fingers and thumbs, for example, in case they get a finger cut, or a user might enroll two facial scans: one with glasses off, another with glasses on.

Microsoft claims that Hello facial geometry scans have a False Acceptance Rate (FAR) of 1 in 100,000 and a False Rejection Rate (FRR) of between 2% and 4%. When a user is falsely rejected, the user can still log on with their PIN or passphrase. A facial scan logon with the Intel RealSense 3D Camera requires about 1 to 3 seconds. The facial scan generates a vector-based map using about 60 points on the face. A thumbprint scan includes about 21 to 40 points, depending on the hardware and driver.

After five failed facial/iris/fingerprint scans, the user must authenticate with a passphrase or PIN. If the device has a TPM chip, then, after 32 PIN failures, the TPM can be configured to refuse any further attempts and the device will have to brought in to the IT department for a reset.

For facial scans and other biometric factors, we don't have the source code to Windows or its device drivers for the biometric input devices, so we have to rely on the False Acceptance Rate (FAR) and the False Rejection (FRR) published by Microsoft and third-party researchers trying to break the system.

We do know, however, that simply holding up a photograph of a user in front of their laptop using Hello and the Intel RealSense 3D Camera does not work, but a 3D-printed cast of the user's head might work, and the photographs for constructing the model could be extracted from social media. To combat this threat, the Hello system can require the user to turn their head and to blink; this could also be simulated with the 3D cast and some animatronics, but nothing will ever be perfect.

Windows Hello and Biometrics Best Practices

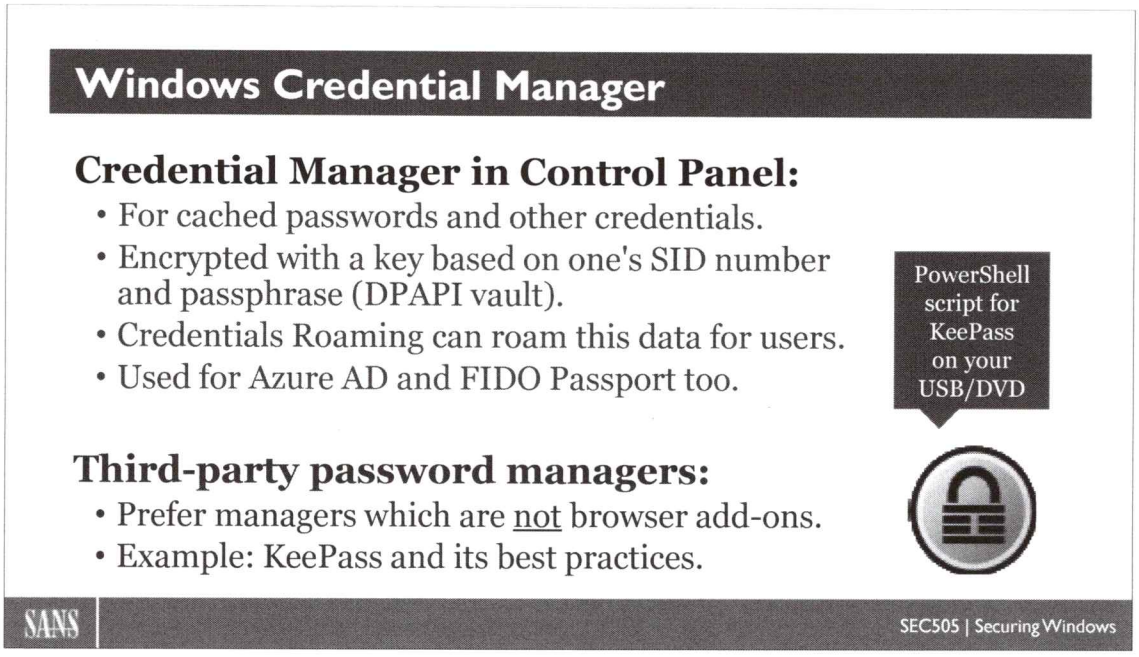
Biometric authentication requires special biometric hardware, especially for Hello facial and iris scans. Much of the (in)security of a biometric solution will depend on the details of the hardware, device drivers and operating system -- none of which are under our control. We do have control over which hardware to purchase though.

- Prefer facial and iris scanning camera systems specifically designed for Windows Hello security, not those that happen to be crudely compatible. Read the vendor's specifications to see if their model supports higher resolution, "liveness" detection, or other differentiators to reduce their FAR/FRR rates.

- Prefer ultrasonic and/or thermal thumbprint readers over the older surface-only readers, which are easier to spoof. Prefer higher resolution over lower.

- Prefer PCs, tablets and phones that include a Trusted Platform Module (TPM) chip and UEFI firmware that supports Secure Boot.
- Require users to turn their heads left and right during facial-scan Hello logons.
- Require an additional authentication factor, such as a PIN.

Again, take the above list of best practices with a grain of salt since a smart card is still preferred over picture password under any circumstances.



Credential Manager & Password Managers

Many applications have a "Cache Password" or similar checkbox when the application is used to authenticate to another service by prompting the user for credentials. If the application does not have its own caching feature, it is likely using the Windows Credential Manager. The Credential Manager applet in Control Panel shows what usernames, passwords, certificates and other credentials have been cached on behalf of the user to streamline that user's authentication in the future. This is often used for browser and universal app authentication to web applications, VPN clients, RDP, OneDrive cloud storage, mapping drive letters, FIDO Passport authentication in Windows 10 and later, etc.

If the Credentials Roaming feature is enabled through Group Policy, then a user's certificates, private keys, and optionally all the cached information shown in Credential Manager will follow that user from machine to machine (this is different than roaming user profiles). All these "roamed" items are encrypted with a key derived from the user's passphrase or smart card, then attached as attributes to the user's account in AD.

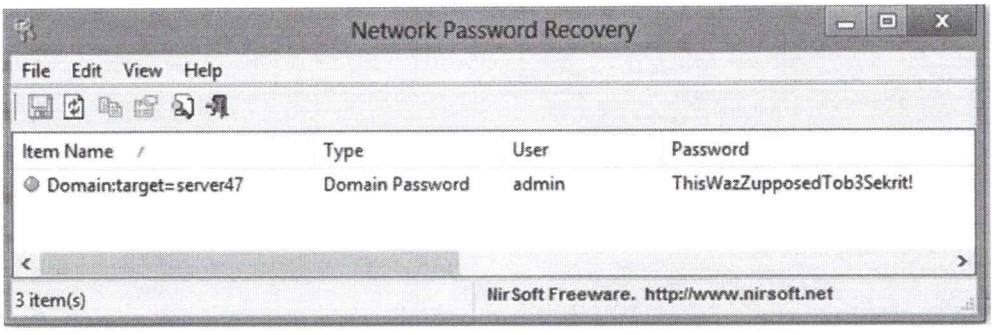
If one's local or domain account is linked to an account in Azure Active Directory for single sign-on to Microsoft's cloud services like OneDrive and Outlook.com, then Credential Manager items are also synced to one's other computers configured to use this feature.

Issues

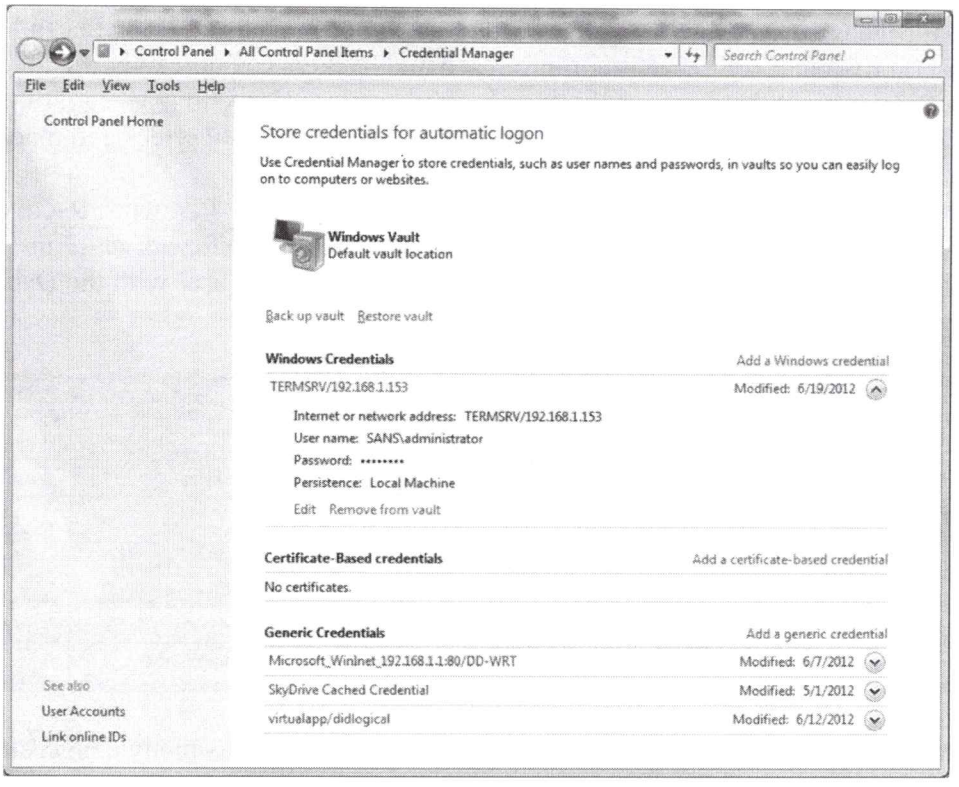
But there are two problems with the Credential Manager and all these roaming/synced credentials: 1) malware or hackers which can run commands with the privileges of Local System or the Administrators group will be able to extract in plaintext the items shown in

Credential Manager, and 2) when a user account is compromised and hackers can log on as that user, or when a computer is compromised and hackers can take over the logon session of anyone who logs on locally, then hackers can use Credential Manager the same way the intended user utilizes it, namely, to easily log on to other services and machines!

As an example, in the screenshot below of NirSoft's free Network Password Recovery tool (www.nirsoft.net) there is a cached username and passphrase for Server47 in Credential Manager. The tool was run on 64-bit Windows 8 as a user with the Debug Programs privilege.



Hence, if an administrative account is compromised and he/she has many dangerous passwords cached in Credential Manager, malware or hackers can use these cached credentials to further compromise the LAN or cloud-hosted web services. The roaming of credentials means that there are more opportunities (more computers) where the compromise can potentially occur.



There are various tools from Microsoft for Credential Manager too:

```
cmdkey.exe /list

vaultcmd.exe /list

net.exe use /savecred /?

control.exe /name Microsoft.CredentialManager

rundll32.exe keymgr.dll, KRShowKeyMgr
```

If you have many of these saved credentials saved and you would like to delete them all:

```
cmdkey.exe /list | select-string -pattern ':target=(.+)' |
foreach {$_.matches.groups[1].value} | foreach {cmdkey.exe
/delete:$_}
```

Disable Through Group Policy

The storage of passwords for Credential Manager can be turned off globally on a computer through Group Policy. The setting is located under GPO > Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options > Network access: Do not allow storage of passwords and credentials for network authentication.

However, if Credential Manager passwords cannot be saved, then they also cannot be saved for the sake of scheduled tasks which require a password on Server 2008, Vista and later (the Task Scheduler works differently on earlier operating systems). This prevents a scheduled task from running when the option named "Do not store password" is unchecked in that task's property sheet.

Third-Party Password Managers

If high-value users cannot use Credential Manager, won't this create an incentive to use the same, short password for as many systems and web services as possible, and to avoid ever changing this password? Yes, we have to fear unintended consequences like these, but there's an even bigger problem. High-value users like systems administrators often must keep track of dozens or hundreds of usernames, passwords, encryption keys and other secrets. Almost no one can keep all these in their heads and we don't want them kept in a plaintext spreadsheet or similar file (hackers look for them after breaking into administrators' workstations).

There are a variety of free and commercial password manager applications on the market. They all function as miniature encrypted databases with user-friendly GUIs to help automate the entering of username, passwords, keys, account numbers, commands, and other sensitive data. Some password managers can even encrypt and store other files.

However, don't use password managers which must be loaded as add-ons or extensions into browsers or PDF readers. It makes the password manager more likely to be compromised when the browser or reader is infected. Excluding these types of managers shortens the list, but if your favorite manager has a browser add-on as an option, simply don't use that option and then forcibly disable that option for the other high-value targets. If you can't disable the browser add-on feature, then that password manager cannot be used because soon most everyone will figure out how to turn it on.

Here are a few commercial and free password managers to consider then:

- KeePass (www.keepass.info) - Free
- Password Safe (passwordsafe.sourceforge.net) - Free
- Cyber-Ark (www.cyber-ark.com) - Commercial
- IronKey (www.ironkey.com) - Commercial
- Lieberman Software (www.liebsoft.com) - Commercial
- LastPass Pocket (www.lastpass.com) - Free and Commercial Versions
- Kaspersky Password Manager (www.kaspersky.com) - Commercial
- SplashID (www.splashdata.com) - Commercial
- RoboForm (www.roboform.com) - Commercial

Whatever you choose, just keep in mind that this little password database file is absolutely golden to attackers, and they will look for it because these types of utilities are so commonly used. You'll want to throw everything and the kitchen sink at this file to try to secure it.

Of course, the best way to secure a password to a company asset is to not have one at all. When possible, use smart card authentication to your organization's servers instead.

Best Practices

The following are some general best practices for password managers in general, followed by additional best practices for KeePass in particular (because KeePass is so popular, hackers will look for it):

- Remember, the primary way your adversaries will steal your password manager data is to infect the password manager application itself, and this will often require administrative privileges on the computer. For many reasons, focus on preventing malware infections and administrative compromises.

- Use dedicated and protected workstations, such as VMs on a jump server, to perform administration and to access password manager databases. Suspend or shut down these workstations when not in use, such as during weekends.

- Use anti-malware and anti-exploit HIPS tools, such as Microsoft EMET, to protect password manager applications from malicious code injection or modification. An endpoint HIPS can monitor and try to prevent undesired access to the password manager's process in memory from other processes.

- High-value user accounts should not be allowed to cache passwords in Credential Manager, to use Credential Roaming with passwords (private keys is fine), or to link their account to a Microsoft Account.

- If someone has two accounts (one privileged, the other regular) it might be permissible for the regular account to use Credential Manager, Credential Roaming or to link to a Microsoft Account, but it's preferable to avoid doing this. The reason is that an administrator might cheat by logging on with their regular account and then cache administrative passwords in Credential Manager for convenience. It might also be best to disallow it as a matter of policy simplicity.

- Do not use password managers which are loaded as add-ons or extensions into browsers. It makes the password manager more likely to be compromised when the browser is infected. It's best if the password manager is an entirely separate process.

- Do not use cloud-integrated password managers, except perhaps for online backup of locally-encrypted database files in archives which are themselves locally encrypted. Remember, if determined adversaries know that the crown jewels can be had by breaking into the cloud storage provider, which is online by definition, then this might be an easier path than directly breaking into your well-protected LAN.

- Some form of automatic password escrow must be established so that if a user is fired or becomes incapacitated, their password manager database will be accessible to authorized personnel. Some commercial password managers have this as a feature of their enterprise-class editions, but free managers can be used too with the right procedures and auditing in place.

- It is possible to use an on-screen keyboard instead of the physical keyboard to enter the master passphrase. However, if your adversaries are able to install a keystroke logger, they can also install a video screen capture driver too, which can be triggered to start recording when the password manager program is launched, which will show the mouse pointer as it is used to click-enter the passphrase. A kernel-mode driver may also intercept the keystrokes in memory, or even possibly extract the entire password database and decryption key from memory in plaintext. On-screen keyboards also create an incentive to use a short master password instead of a long passphrase because they are slow and inconvenient. We have to balance security with productivity, so the additional value of on-screen keyboards is debatable. This is a matter of an organization's preference, there is not a clear-cut correct answer.

Remember, when your adversaries break into your computer because you are an administrator, they will look for password manager database files. They will try to steal the database and hack into it. Every time you open your password manager you are putting the company at risk, but there's no way around it, you have to have a password manager to get your work done. It's a double-edged sword, just like single sign-on.

Best Practices for KeePass (www.keepass.info)

KeePass is a free and open source password manager with many security features, including the fact that it is not a web browser add-on. KeePass can be run from a USB flash drive, tries to thwart in-memory attacks, has many third-party extensions, can optionally be tied to a specific user account using DPAPI, and has some capabilities for administrators to enforce good passphrase security too. To automate the entering of data into web browsers without becoming part of the (infected) browser, KeePass has a feature similar to macro variables to enter keystrokes and paste in passwords from the clipboard.

Here are a few best practices that are somewhat unique to KeePass. This isn't an exhaustive list, it just contains some important items that are sometimes overlooked, and some recommendations are only for IT personnel, not for high-value users in general:

- KeePass can be installed and run entirely from an NTFS-formatted thumb drive. The thumb drive can be unplugged at the end of the day to sleep better at night (nothing like an air-gap firewall). A small USB hub with per-port power switches could be used instead of unplugging the thumb drive itself. NTFS permissions, auditing and BitLocker can all be used to help secure the thumb drive itself.

- Back up, monitor, and secure the KeePass configuration file (KeePass.config.xml) using whole disk encryption, NTFS auditing, and NTFS permissions. This file

will be located in C:\Users\<username>\AppData\Roaming\KeePass\ or in the same folder as the KeePass executable when using the portable version, like with a USB flash drive. This XML file usually contains the path to the secondary seed key and may be modified to execute malicious commands when KeePass runs.

- Use a combination of a master passphrase and a seed key file to generate the final key used to encrypt the database. The key file should be kept in a different folder than the KeePass database file or the KeePass program, and its name and folder should not indicate that it is being used as a KeePass key file. Even if a keystroke logger has captured the master passphrase, the key file would still also need to be stolen too in order to decrypt the database.

- Read and write access to the KeePass database file and key file should be restricted with both NTFS permissions and an MIC integrity level set to High. This means the KeePass program will have to be launched elevated every time (modify the shortcut). Do not turn off User Account Control (UAC) prompting. Running KeePass elevated helps to protect the KeePass process from other processes which are not running elevated, such as the browser.

- Require switching to the User Account Control (UAC) secure desktop before entering the decryption passphrase. To do so, enable the option named "Enter master key on secure desktop" (Tools menu > Options > Security tab).

- If you're not using it, disable the trigger system (Tools menu > Triggers).

- Under Tools menu > Options > Policy tab, uncheck at least the following boxes, if your workflows allow it: Plugins, Export, Export No Key Repeat, Print, Print No Key Repeat, and, Edit Triggers.

- Because laptops and tablets are more likely to be lost or stolen, on these devices in particular make sure to enable the option named "Lock workspace when the computer is about to be suspended" (Tools menu > Options > Security tab).

- Never use the "-pw" parameter when launching KeePass from the command line. The password you provide at the command line will be visible to any malware or hackers which can list current processes, and, if process tracking is enabled with the command line logging option, the password will appear in the System event log in plaintext too (event ID 4688).

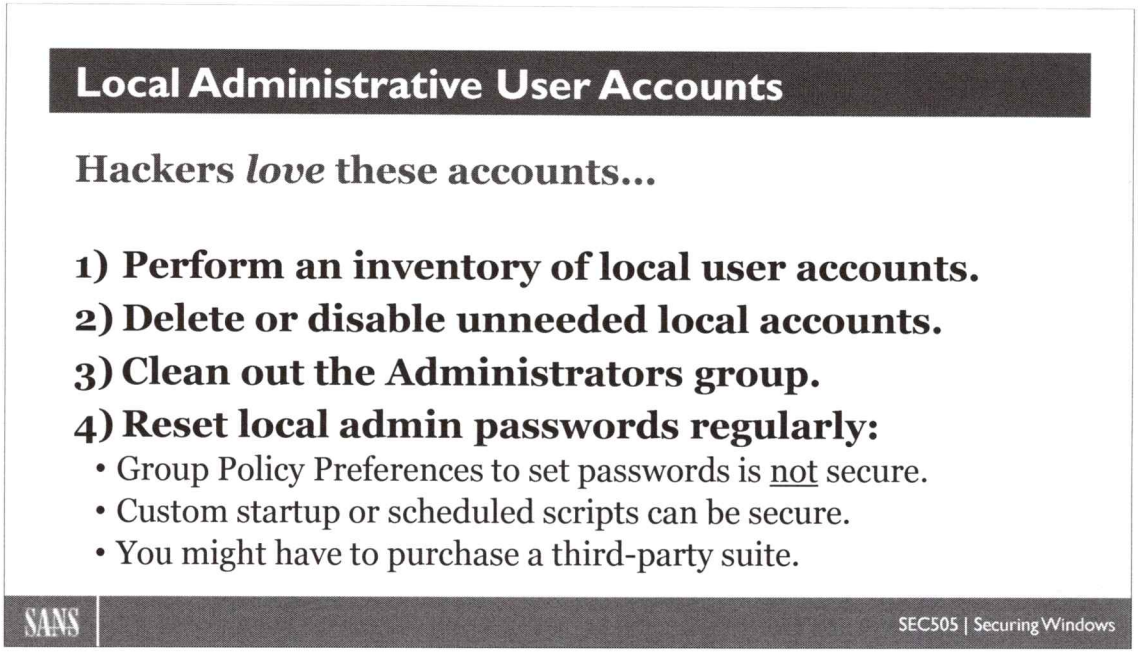
- Do not use the Windows account integration feature. This protects the KeePass database using the same DPAPI techniques as the Windows Credential Manager, which isn't bad, but we want separation from Windows integration in this case for practical reasons. Also, troubleshooting SID issues with the Data Protection API is not fun, and there is a non-zero chance of losing access to your data if there is a problem, e.g., when you must rebuild your workstation from scratch.

- If you choose to sync the KeePass database file over the network, use native Windows IPSec and/or SMB encryption, and confirm that the share and NTFS permissions are minimal on the SMB server. If you'd rather use SCP, SFTP or FTPS instead, there are KeePass extensions for these protocols too. Using a cloud provider can be secure, but it very much depends on the details of the cloud provider's security options and how they are configured; if in doubt, don't sync using third-party cloud providers.

- (Optional) Help hide the path to the key file by disabling the option named "Remember key sources" (Tools menu > Options > Advanced tab).

Note, to use CHML.EXE to change the MIC label and rules on the KeePass files:

```
chml.exe database-file-path.kdbx -i:h -nr -nw -nx

chml.exe key-file-path.gif -i:h -nr -nw -nx
```



Local Administrative Accounts for the Help Desk

Hackers *love* local administrative accounts created for the help desk and IT. Whether that local account is named "Administrator" or not, as long as it's in the local Administrators group it will have complete control of the machine. These local administrative accounts usually have the same username and password on every computer in the organization, and the password is rarely changed, if ever. Very often, the password is less than nine characters long and relatively easy to crack. Who needs a global account in Domain Admins when you can log on with administrative privileges everywhere you need to go already?

Group Policy Startup/Shutdown Scripts and Scheduled Jobs

As a reminder, recall that Group Policy can push out scripts which run at startup or shutdown using Local System privileges. Also, Group Policy can be used to manage scheduled jobs to run binaries and scripts on a recurring basis. These capabilities are important when performing mass inventories or command execution when solving problems related to local user accounts.

GPO-assigned scripts can add, delete or modify any local account, including resetting the password on any local account. In a GPO, these script options are located under Computer Configuration > Policies > Windows Settings > Scripts (Startup/Shutdown).

In a GPO, the scheduled job options are located under Computer Configuration > Preferences > Control Panel Settings > Scheduled Tasks. Recall from earlier that the target recipients of GPO Preferences (the managed clients) must have the following:

- Windows 7/Server 2008 or later (no other updates necessary).

- Windows Vista+SP1 or later SP, plus the Client Side Extensions (CSE).

- Windows Server 2003+SP1 or later SP, plus the Client Side Extensions (CSE) and, if the latest SP or IE is not installed, the XMLLite update too.

- Windows XP+SP2 or later SP, plus the Client Side Extensions (CSE) and, if the latest SP or IE is not installed, the XMLLite update too.

  The Client Side Extensions (CSE) can be downloaded from Microsoft's main Group Policy page at http://technet.microsoft.com/windowsserver/grouppolicy/.

  The XMLLite update is bundled into Internet Explorer 7.0 and later, with XP-SP3 and later, and with Server 2003-SP2 and later Service Packs. If necessary, XMLLite can be downloaded separately from http://go.microsoft.com/fwlink/?LinkId=111843.

Perform An Inventory

There are a variety of tools which can inventory the local user accounts and the members of the local Administrators group on remote computers. Some are part of expensive Enterprise Management System (EMS) products, others are just simple GUI tools or scripts. One way or another, though, we need to inventory all the local user accounts and all the members of the Administrators group on every computer in the organization.

This can be done with PowerShell, of course, and a script could be written to clean up the data before saving it to a comma-delimited file, XML file, spreadsheet, database, or however you prefer to format the inventory data so that you can use it later.

To query the list of local accounts on a remote computer named host.sans.org:

```
Get-WmiObject -Query "Select * From Win32_UserAccount Where
LocalAccount='True'" -Computer host.sans.org
```

To list the members of the local Administrators group on host.sans.org:

```
$LocalAdmins = Get-WmiObject -Query "Select * From Win32_Group
Where Name='Administrators' and LocalAccount='True'" -Computer
host.sans.org

$LocalAdmins.GetRelationships("Win32_GroupUser") | ft
PartComponent
```

Similar commands could be run from a GPO-assigned startup script, the output captured, and then the data saved over the network to one location, such as an SMB shared folder or a SQL Server database. Again, there are third-party products to handle this too.

Delete or Disable Unnecessary Local Accounts

If a local account exists on a computer, but it is no longer needed, then delete or disable it. Everyone knows this, but examining networks in real life proves that it is often not done.

On Vista and later, the built-in local Administrator account is disabled by default. There is also a Group Policy option to keep it disabled on Windows 2000 and later (GPO > Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options > Accounts: Administrator account status).

Using Group Policy, any local account can be created, deleted or disabled. In a GPO, these options are located under Computer Configuration > Preferences > Control Panel Settings > Local Users and Groups.

If more flexibility or customization is required, then a startup script or scheduled job can be pushed out through Group Policy too.

Remove Unnecessary Administrators Group Members

Using Group Policy preferences, startup scripts, scheduled jobs or remote command execution tools, if any user account found in the local Administrators group does not need to be there, remove it. This is why we need to perform an inventory first, to see who is actually a member of the Administrators group out there in the wild LAN.

With the inventory in hand, discuss with the developers and other IT personnel whether or not these accounts are needed, need to be in Administrators, or whether less-powerful privileges and permissions could be granted to satisfy their needs. This will be a long and political process, but it has to be done. The painfulness of this review process is part of the reason these admin accounts just accumulate over the years, which is partly the reason they are so attractive to hackers.

Block Network Logons of Local Accounts with Blank Passwords

No local account should have blank password, but just in case one is forgotten there is a Group Policy option to block all over-the-network authentications using any local account with a blank password on the target machine (GPO > Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options > Accounts: Limit local account use of blank passwords to console logon only).

Managing Local Passwords With GPO Preferences

For the local user accounts which are actually needed, their passwords can be managed through Group Policy, but there is a catch. These options are located under Computer Configuration > Preferences > Control Panel Settings > Local Users and Groups.

When you set a password using GPO Preferences, the password is encrypted with 256-bit AES and stored as part of the GPO in the SYSVOL shared folder on the domain controller, hence, the encrypted password is also downloaded by every Group Policy

client. However, the AES key is also a part of the GPO! The key is obscured, but researchers have figured out how the obfuscation scheme works. You can find free tools to decrypt the password on the Internet (do a search on "gpprefdecrypt.py" for example).

Hence, you'll need to carefully weigh the risks against the management conveniences. There may be a more secure method of managing the password other than GPO Preferences. At a minimum, if you do set a password with a GPO Preference, remove that option in the GPO after it's been processed by the clients. The password change will remain intact on the client computers even after this setting in the GPO has been deleted.

But there is a better way to manage local account passwords, namely, to use Group Policy to assign a scheduled job to manage these passwords.

Managing Local Passwords with GPO Scripts and Scheduled Jobs

When Group Policy runs a startup or shutdown script on managed computers, the script runs with Local System context and there is no password in the GPO or SYSVOL share. Nor does this method require an administrator to authenticate to the managed computer over the network, potentially exposing the administrator's Security Access Token (SAT) or password hash to any malware which already be present on the machine.

GPO-assigned startup scripts can inventory, delete and edit the properties of local accounts, but what about resetting their passwords?

It is not safe to push out a script which includes a password in plaintext inside it. The script can be read out of the SYSVOL share by anyone with a domain account, and the password might be sniffed out of packets as other computers process their GPOs.

It is safe, though, to have a GPO startup script generate a new random string in memory and reset a local user's password to use that random string. Because the string is random, each machine would have a different password.

But what if you want to reset a local user's password to a random string, then record the date, time, computer name, username and password at another server over the network to a central server? This can certainly be done. In fact, your course CD includes an entire solution in three scripts in CD:\Day4-Admins\UpdatePasswords\ (see the README file).

```
# Look in the SANS CD-ROM:\Day4-Admins\UpdatePasswords

.\Update-PasswordArchive.ps1 -cert PublicKeyCert.cer
     -LocalUserName Guest -PasswordArchivePath .\

# Open the new file with the long name in Notepad.
# Now import the following private key (the password
# is "password"):

.\Password-is-password.pfx
```

```
# With the private key, only you can decrypt the password file:

.\Recover-PasswordArchive.ps1
```

The Update-PasswordArchive.ps1 script uses a public key file of your choice (PublicKeyCert.cer) to encrypt the password, then saves that encrypted password as a file to a shared folder on a centralized server using SMB. Group Policy could assign the script to run as a scheduled job, perhaps every weekend or every night at 3AM. The password is never transmitted or saved to disk in plaintext. Only the holder the corresponding private key can decrypt the password archive files, and this private key can be on a smart card. For fault tolerance and scalability, use a Distributed File System (DFS) share on two or more SMB servers to store the password archive files. If the recovery private key is stored on a smart card, the card and PIN could be shared among the relevant IT staff on an as-needed basis only.

When someone in IT needs to authenticate using a local administrative account to a remote system (for example, to LAPTOP47), he or she could get the latest password from the shared folder server where the password archives are being stored, like this:

```
.\Recover-PasswordArchive.ps1 -PasswordArchivePath \\server\share
   -ComputerName LAPTOP47
```

The password never traverses the network in plaintext, it is decrypted in the memory of the PowerShell process of the administrator's computer and displayed in the shell.

Third-Party Management Tools

If you don't want to script a solution to local user password management yourself, then there are many third-party companies with products to sell. This will have the advantages of a graphical interface, technical support and a proven solution that works right from the start, but of course it won't be free. Not everyone who attends this course can afford the commercial solutions, so that's why the do-it-yourself options are discussed first (and it's a nice way to practice more PowerShell).

Do an Internet search on terms like "enterprise password reset local account" to find these vendors and products. Here are a few to get started:

- ManageEngine Password Manager Pro
  (http://www.manageengine.com/products/passwordmanagerpro/)

- Cyber-Ark Privileged Identity/Session Management
  (http://www.cyber-ark.com)

- Lieberman Software Password Manager
  (http://www.liebsoft.com)

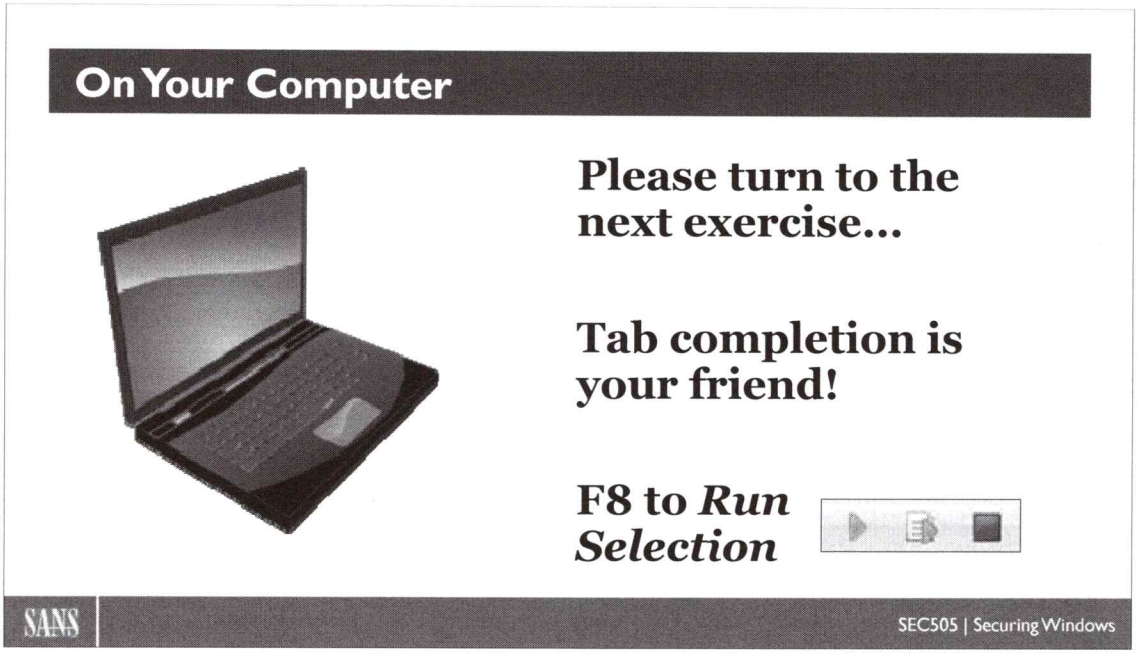
- Thycotic Secret Server
  (http://www.thycotic.com)

- PasswordCourier Password Management
  (http://www.courion.com/products/passwordcourier.html)

- NetWrix Privileged Account Manager
  (http://www.netwrix.com/privileged_identity_management.html)

- AutoCipher
  (http://www.autocipher.com)

Note: Beware of tools which require NTLM authentication in order to reset local account passwords, such as any tool or script using the "WinNT:" provider with the ADSI programming API. We want to use only Kerberos, so using such tools would just create another dependency on NTLM and make NTLM more difficult to phase out later on. If you are resetting passwords remotely with your own tools, you're better off performing remote command execution to do so.



On Your Computer

This exercise has two parts.

Inventory Local Group Membership

Switch to the C:\SANS\Day4-Admins directory in PowerShell:

```
cd C:\SANS\Day4-Admins
```

List the membership of the Administrators group on a local or remote computer (note that you can pass in an array of computer names instead of just one name):

```
.\Get-LocalGroupMembership.ps1 -LocalGroupName Administrators
  -ComputerName $env:ComputerName | Format-List *
```

Save the output of that query to an XML file for later processing:

```
.\Get-LocalGroupMembership.ps1 -LocalGroupName Administrators
  -ComputerName $env:ComputerName | Export-CliXml .\Inventory.xml
```

Or save the output to a comma-delimited text file to which more output can be appended:

```
.\Get-LocalGroupMembership.ps1 -LocalGroupName Administrators
  -ComputerName $env:ComputerName -CommaSeparatedOutput |
  Out-File -Append -FilePath .\Inventory.csv
```

Note: In the example above, we are querying only one computer, but in real life we would more likely query an array of computer names instead of just one, hence, the importance of appending all the output to one CSV text file.

Update Local User Account Password

Switch to the C:\SANS\Day4-Admins\UpdatePasswords directory in PowerShell:

```
cd C:\SANS\Day4-Admins\UpdatePasswords
```

Note: A .pfx file contains a certificate and private key, protected by a password. The .pfx file name extension is associated with a graphical wizard for importing the certificate and key into the current user's certificate store. Accept all defaults.

Import a certificate and private key for testing (the import password is "password"):

```
.\Password-is-password.pfx
```

Reset the password on the Guest account with a randomly-generated password:

```
.\Update-PasswordArchive.ps1 -LocalUserName Guest
  -CertificateFilePath .\PublicKeyCert.cer -PasswordArchivePath .
```

Examine the encrypted password archive file that was just created (notice the "."):

```
dir

explorer.exe .
```

Decrypt the password of the Guest account with your private key (notice the "."):

```
.\Recover-PasswordArchive.ps1 -PasswordArchivePath .
  -ComputerName $env:COMPUTERNAME -UserName Guest
```

Directly pipe this password into the clipboard (notice the "." again for the path):

```
.\Recover-PasswordArchive.ps1 -PasswordArchivePath .
  -ComputerName $env:COMPUTERNAME -UserName Guest | Select-Object
  -ExpandProperty Password | clip.exe
```

Instead of piping into the clipboard, we could pipe the password into another script or tool that requires the password for some reason, e.g., copy files or open an RDP session.

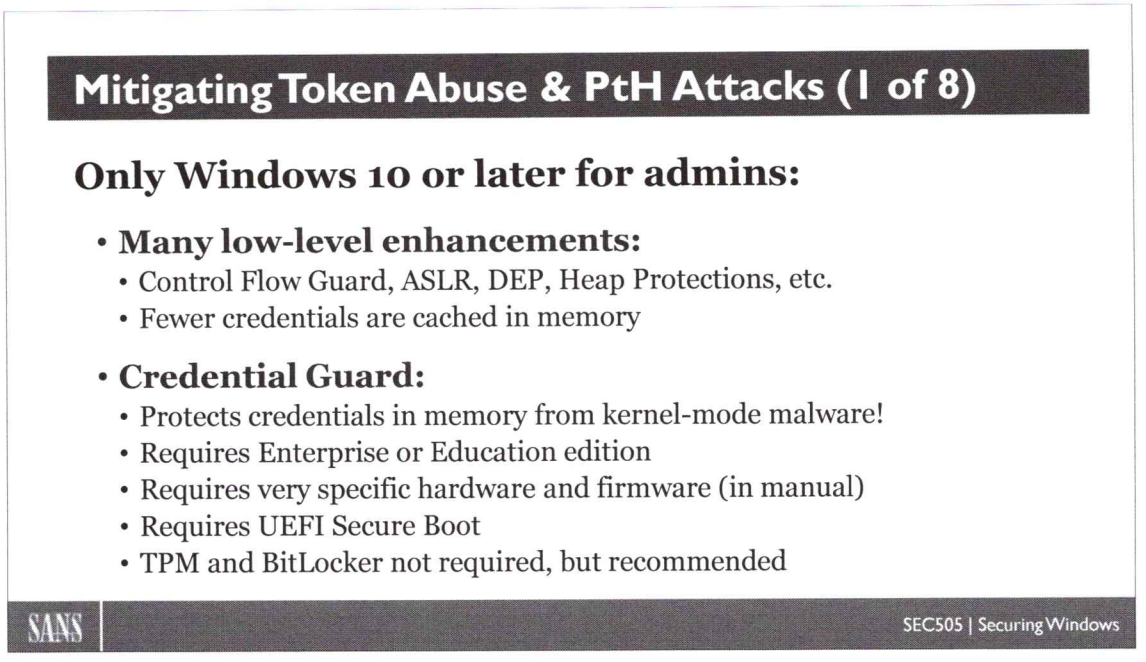
Remember, too, that the scripts, certificates and encrypted files can all be run from and saved to shared folders on the network. Without the private key, the password file cannot be decrypted even if your adversaries steal a copy of the public key certificate.

Finished Already?

We don't have time to do this in seminar, but, in real life, we could use Group Policy to create a weekly scheduled job on each machine to run the Update-PasswordArchive.ps1 script from a shared folder and to save the encrypted output to the same or another shared folder. Do this in your VM if you want to try!

To read more about how the full solution would be implemented in real life, including tips for security, fault tolerance and scalability:

```
notepad .\README.txt
```



Mitigating Token Abuse & Pass-The-Hash Attacks

There is no universal patch against SAT stealing or pass-the-hash attacks. Once malware is running in kernel mode as a part of the operating system, there is nothing it cannot do (these attacks are just the beginning). Hence, the line in the silicon which must be absolutely defended is to prevent malware or hackers from executing code with kernel-mode privileges in the first place. They often achieve this by compromising users or network services that have administrative privileges like Debug Programs and Impersonate A Client. Once your adversaries are running kernel-mode code on a computer, that computer is lost, and it's possible they may leapfrog to other machines as well if administrative credentials could be stolen.

Only Windows 10 and Later for Administrators

The workstations of administrators should run Windows 10 or later. Windows 8.1, Server 2012-R2 and later can limit access to the passwords and hashes the kernel stores in memory; for example, the LanManager and Digest hashes, and the RDP password, can be purged from memory. These operating systems also limit access to the memory address space of lsass.exe (where these hashes are cached) from lower-privileged processes. Other kernel defenses come from the redesign of the user-mode heap manager too. In Windows 10 and later, most OS binaries are compiled with support for Control Flow Guard (CFG), a technology related to ASLR and DEP for combating exploits.

To ensure that some of these memory protections are enabled, please see KB2973351 and KB2975625. Originally enabled by default, Microsoft turned them off, requiring a registry value to be set (DisableRestrictedAdmin) to enable these protections, then turned the protections back on again by default in Windows 10 and later. Some of these protections can also be applied to Windows 7 too (see the KB articles). However, this

particular change will require careful testing, there are known issues, especially on Windows 7. Nonetheless, there are other credential protections in Windows 8.1 and later that do not depend on this registry value.

In general, of course, network administrators should always have the latest version of whatever operating system they prefer to use, whether that is Windows, Linux or Mac OS. Because of the complexities involved, new kernel-level security enhancements are usually obtainable only by upgrading to the latest OS version and patches.

Credential Guard

Windows 10 and later supports "Device Guard", which is a catch-all marketing term that includes several technologies, some of which existed in Windows 8.1 too. A related technology is "Credential Guard", which is not the same thing as Device Guard, but Microsoft's documentation often does not make this clear. Credential Guard does not require Device Guard, nor vice versa, but they both have similar hardware, firmware and OS requirements. Importantly, you can enable Credential Guard without enabling all of Device Guard, so please don't let Microsoft's lengthy Device Guard documentation scare you away. And, in order to make things as confusing as possible, Microsoft also refers to "Virtualization-Based Security (VBS)" and "Virtual Secure Mode (VSM)" and "Hyper-V-Based Security" as meta-catch-all terms that apparently includes both Device Guard and Credential Guard (but not Hyper-V Containers, which is something else again).

This section will only talk about Credential Guard. Credential Guard protects credentials and other secrets in memory from kernel-mode malware. Credential Guard relies on hardware, firmware and hypervisor features to secure these secrets. It is not just an OS configuration setting. Without the correct hardware, Credential Guard will not work. Only security features enabled in the hardware, firmware and hypervisor can prevent malware in the OS kernel from stealing these secrets, not Windows itself or alone. This is because the hardware, firmware and hypervisor are *below* the OS kernel in the computer's architecture.

Credential Guard Hardware and Firmware Requirements

To enable Credential Guard, your hardware device requires:

- UEFI 2.3.1 Errata B or higher firmware (2.4 or later preferred).
- UEFI Secure Boot enabled.
- 64-bit (x64) CPU and chipset.
- Motherboard chipset compatibility with Input/Output Memory Management Unit (IOMMU) use, such as in many of the Intel vPro-branded motherboards.
- Intel VT-d support in the CPU for IOMMU (or AMD-Vi IOMMU).
- Intel VT-x support in the CPU for type-1 hypervisors (or AMD RVI).

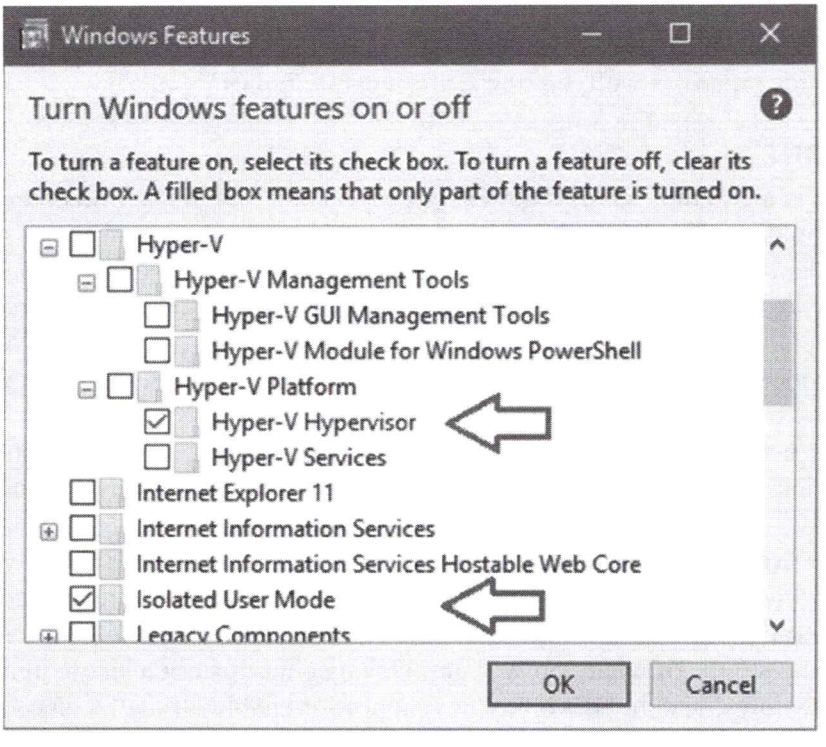
- Intel EPT support for Second Level Address Translation (or AMD RVI).

It is not required, but highly recommended, to also have a TPM chip in the motherboard to help encrypt secrets on the drive and to implement a virtual smart card. It is also highly recommended to use whole disk encryption, such as BitLocker, preferably integrated with the TPM too. Again, a TPM and BitLocker are not required for Credential Guard, but highly recommended, especially if other Device Guard features will be enabled.

Credential Guard Software and Registry Requirements
To enable Credential Guard, your OS requires or must be:

- Windows 10 or later, Enterprise or Education editions only, not Pro.

- Two Windows features must be enabled. They are named "Hyper-V Hypervisor" and "Isolated User Mode", and they can be seen in Control Panel in the Programs and Features applet when you click on the link for "Turn Windows features on or off".



- Three registry values must be configured too. This can be done with REGEDIT, a PowerShell script, Group Policy, or any other method. Here are the three values:

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\DeviceGuard]
"EnableVirtualizationBasedSecurity"=dword:00000001
"RequirePlatformSecurityFeatures"=dword:00000002
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa]
"LsaCfgFlags"=dword:00000001
```

In a GPO with the necessary ADMX template loaded, such as on a Server 2016 or later domain controller, the path to the Credential Guard GPO settings are located here: Computer Configuration > Administrative Templates > System > Device Guard. In this container you will find a setting named "Turn On Virtualization Based Security", and this includes a checkbox to enable Credential Guard. There is another checkbox there too named "Enable Virtualization Based Protection of Code Integrity", BUT DO NOT TOUCH IT!

> **Warning!** Do NOT check the box in the GPO for "Enable Virtualization Based Protection of Code Integrity"! This is for Device Guard and can render your PC permanently unbootable. You must consult Microsoft's documentation on Device Guard and perform adequate lab testing first. **You have been warned!**

After meeting the hardware, firmware and OS requirements, and setting the three registry values listed above, reboot the system and Credential Guard will be enabled. There is nothing else to see and no user training required. It's highly recommended to also use whole disk encryption, deploy smart cards (physical or virtual TPM smart cards), and to configure Device Guard as well, but these are separate topics.

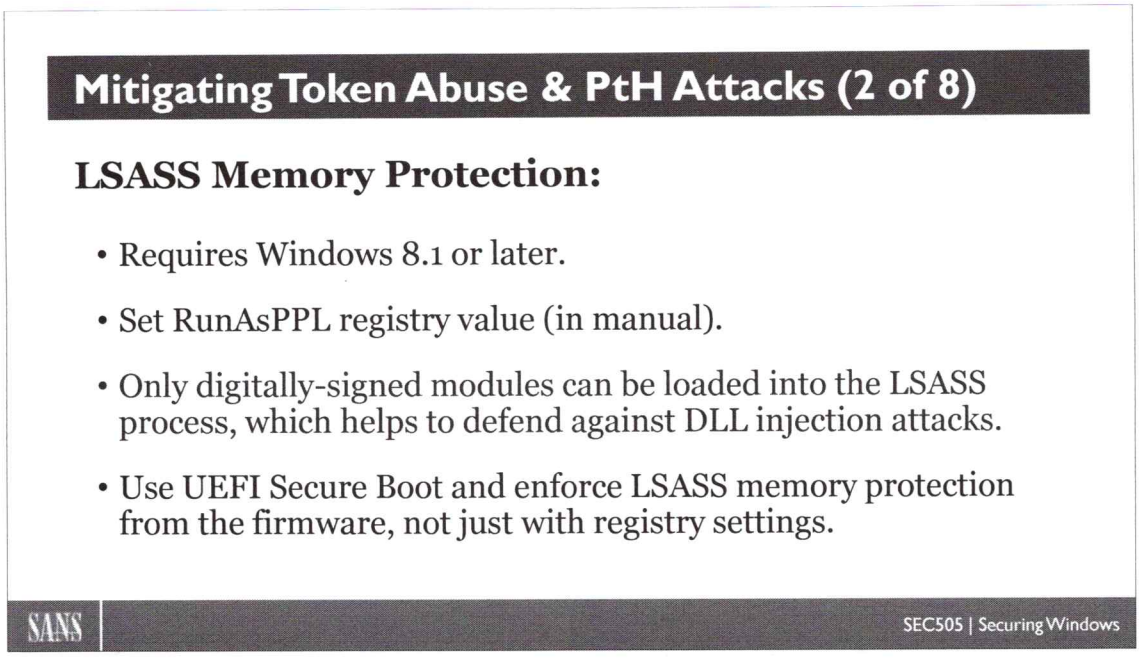
Device Guard

Device Guard is a complex set of features related to code integrity enforcement, both in memory and on disk. The hardware, firmware and OS requirements are very similar to those for Credential Guard above, but the configuration settings and testing necessary are far more complex. The configuration and testing necessary for the various code integrity policies and catalog files cannot even be summarized here, you must consult Microsoft's latest on-line documentation. Device Guard could literally be a one-day course by itself.

There is a real risk of misconfiguring Device Guard and turning a computer into an unbootable brick. Please be careful! Do not play around with any Device Guard settings.

It is likely that you will not configure Device Guard yourself. It is more likely that you will deploy Device Guard by purchasing new machines that come from the factory pre-configured with Device Guard enabled. Please consult your favorite OEM vendor, and then confirm the details of what you will get: Device Guard is not a single item, it's a collection of features, and an OEM may not support or enable all the Device Guard features you want by default.

Nonetheless, a Windows computer with UEFI Secure Boot, Device Guard, Credential Guard, TPM virtual smart card, and whole disk encryption represents a vastly more difficult target for hackers than prior Microsoft operating systems. When combined with the other security technologies discussed in this course, such as AppLocker and Windows Firewall IPSec, it is possible to significantly reduce your APT infection rate.



Local Security Authority (LSA) Memory Protection

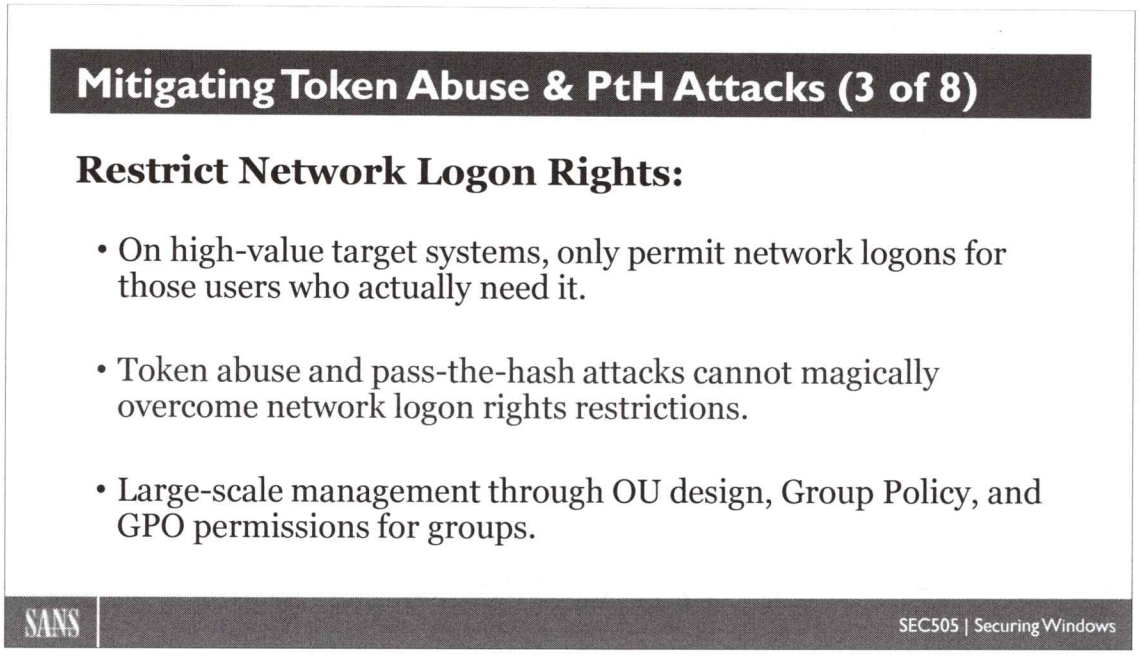
On Windows 8.1 and later, set the following registry value to enable LSA memory protections such that only modules digitally signed by Microsoft can be loaded into the LSASS.EXE process:

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa]
"RunAsPPL"=dword:00000001
```

DLL injection into the LSASS.EXE process is a very common post-exploitation technique, such as for dumping password hashes or installing a rootkit, and enabling this protection will make the injection more difficult.

This must be carefully tested first, though, because your organization may be loading benevolent modules for the LSA which happen to not be signed by Microsoft; for example, you may have an older smart card driver or a Host Intrusion Prevention (HIPS) agent which loads modules into kernel for the sake of the LSA.

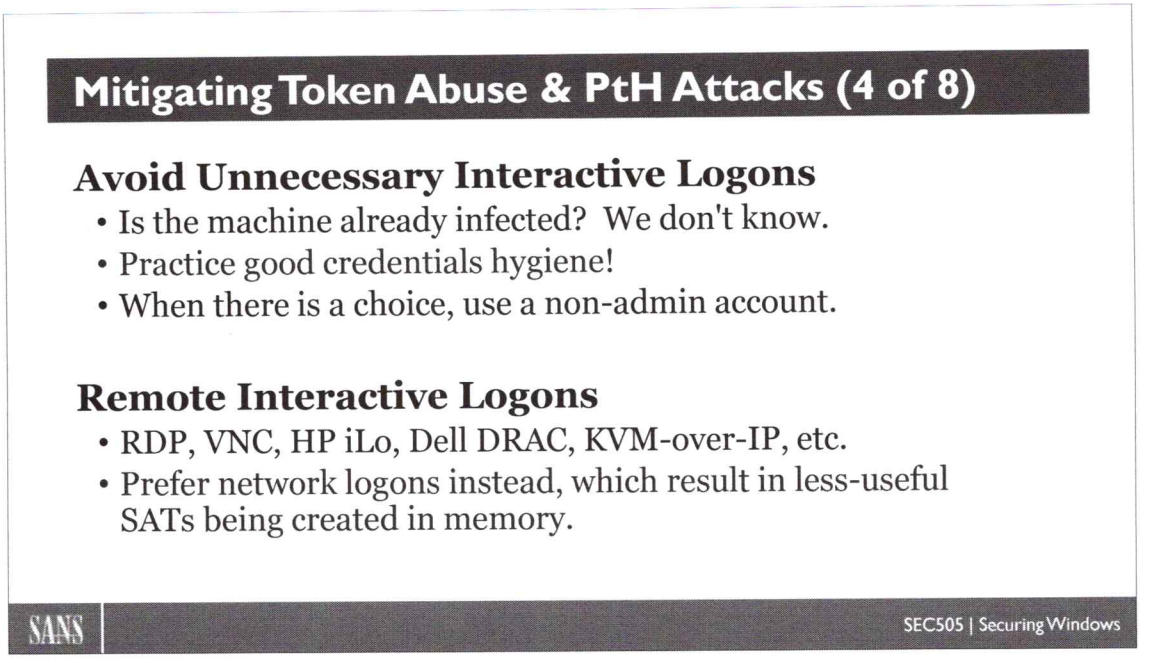
On Windows 8.1 and later, LSA memory protection is enabled by default and cannot be turned off without first compromising the machine (by which time it doesn't matter anymore). Post-exploitation, though, LSASS memory protection can be turned off. Unless, that is, you have Secure Boot enabled and LSASS protection enabled through the UEFI firmware. With UEFI Secure Boot, the protection cannot be disabled by editing the registry, it can only be disabled through modification of the UEFI firmware, which would require local access to the computer, not just a kernel-level compromise.



Restrict Network Logon Rights

But the compromise of one computer does not necessarily mean that all the other computers in the domain must fall like dominos. It depends on the details of the compromise. If a workstation has been taken over and now our adversaries have a copy of a Domain Admins user account password hash or delegation SAT, then the entire domain is almost certainly lost (and probably soon the entire forest too).

Remember, from the point of view of a remote target server, a pass-the-hash attack is not an attack, it's simply a network logon. The same is true when a stolen SAT is used to do a network logon to a remote server. So if a user named Amy does not have any network logon rights to Server47, neither her delegation SAT nor her stolen hash can be used to log on over the network to that server. You can't log on over the network if you do not have the "Access this computer from the network" right. (There might be other exploits possible, but that's another issue.) Hence, use Group Policy to restrict this network logon right. This is one of the few partial mitigations to SAT stealing and pass-the-hash attacks. Far more important, of course, is to try to prevent the malicious code from running with kernel privileges to begin with, but when the inevitable compromises does occur someday, at least we can try to contain the harm to a subset of our machines.



Avoid Interactive Logons With Domain Accounts

If an authentication results in an interactive logon on a computer (event log ID 4624 with logon type 2), then a delegation SAT and password hash will become available to any kernel-mode malware running on that computer. So avoid interactive logons with administrative domain accounts when it's not really necessary, especially with Domain Admin accounts. What about local administrative accounts? These are less dangerous, but that discussion is coming up in a later section.

A network logon, on the other hand (event log ID 4624 with logon type 3), results in only a standard impersonation token and no password hash in the memory of the target. When in doubt, query the security event log on the target to see what type of authentication is being used; you have a PowerShell script on the USB/CD for this: Get-SuccessfulLogon.ps1.

A "real" interactive logon occurs when your fingers are on the physical keyboard or touch screen of a computer to log on locally into its desktop. This is called a *console* interactive logon. In the old days, this was about the only kind of interactive logon, but not anymore.

Avoid Logging On with RDP, VNC, iLo, iDRAC or KVM-Over-IP

If a high-value user connects to a machine with Remote Desktop Protocol (RDP) or VNC, this is considered an interactive logon too. If you log on to the console of a remote machine with Hewlett-Packard iLo, Dell iDRAC, a KVM-over-IP switch or similar technologies, then these are all interactive logons also. I know this is bad news, but avoid using RDP, VNC, iLO, iDRAC and KVM-over-IP when you have other alternatives.

> **Note:** Not all use of iLo or iDRAC is dangerous, just interactive console logons, and not because of problems with iLo or iDRAC, the issue is in Windows.

As a matter of habit, only use RDP/VNC/iLo/iDRAC/KVM as a last resort. Prefer to use standard remote administration tools instead, such as MMC console snap-ins and PowerShell remoting, and then fall back to RDP/VNC/iLo/iDRAC/KVM only as needed.

> **Note:** An exception might be RDP in restricted admin mode (discussed next).

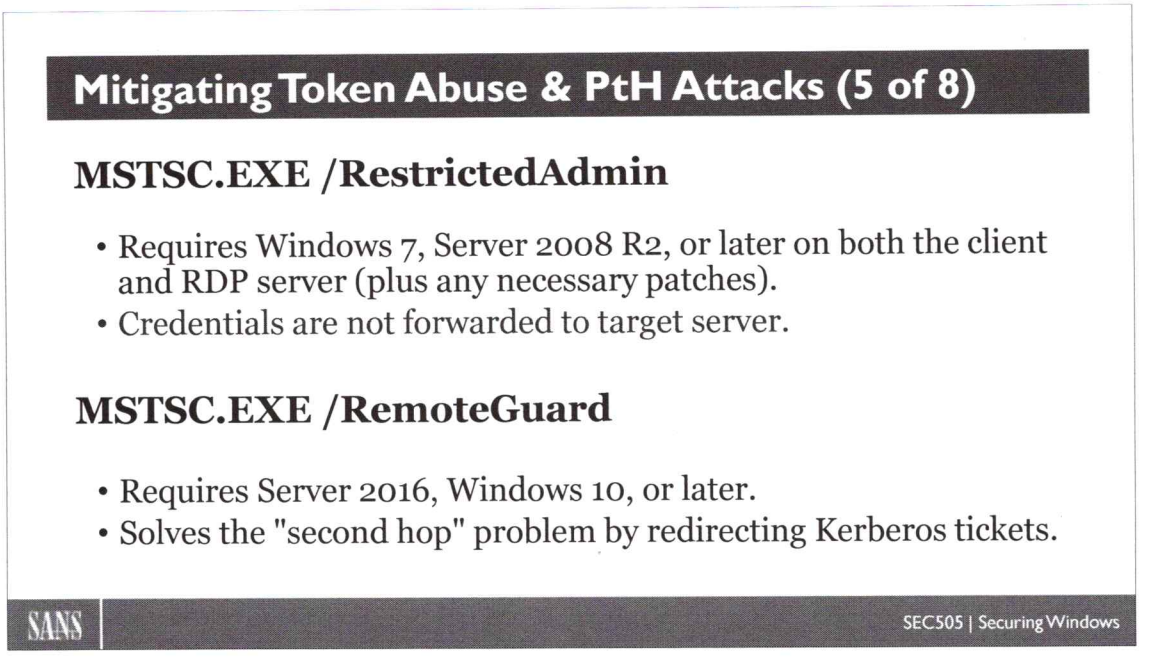
Remember too that startup scripts can be pushed out through Group Policy to run under Local System context, which allows administrative commands to be run without exposing an admin's delegation SAT or password hash.

Group Policy and SCHTASKS.EXE can also be used to create scheduled tasks which run as Local Service, Network Service or Local System, and these will not expose any user hashes or delegation SATs either. Nor will the over-the-network authentication of SCHTASKS.EXE to configure these jobs.

For WMI remote command execution, both PowerShell and WMIC.EXE use network logons by default, not interactive logons, and there is the -Impersonation parameter if you wish to ensure that delegate impersonation will not be used. The popular PSEXEC.EXE tool with the -S switch will use a network authentication and then execute a command as Local System, which is good, but beware of the -U switch, which sends the password in plaintext and results in an interactive logon. (PSEXEC.EXE can also use single sign-on for a network authentication, which is good for avoiding a delegation SAT and a password hash being exposed, but it's still better to use the -S switch and run the command as Local System. We'll discuss this more later on in the section on service accounts.)

> **Note:** WMIC.EXE has been deprecated in favor of PowerShell; it will still be installed by default for years, but there will be no future enhancements to the tool.

When you must use RDP or the other tools, you should log on, preferably with a local account, get the work done, and then log off completely (don't just disconnect and leave the session running). We will discuss local accounts for the help desk and IT in a later section.



RDP Remote Credential Guard

When using the Remote Desktop Protocol (RDP) to connect to the desktop of a remote computer, if that remote computer is already compromised, then there is a risk of the user's credentials being stolen from memory at the remote computer. If using RDP is unavoidable, what can be done to reduce the risk of credential theft?

RDP Restricted Administration Mode

On Windows 8.1, Server 2012 R2, and later operating systems, the RDP thin client can be launched with a special command-line switch:

```
mstsc.exe /RestrictedAdmin
```

This switch is also available for Windows 7, Windows 8, Server 2008 R2 and Server 2012 if the appropriate patches are installed and the necessary registry values set (see KB2984972 and KB2973501 for the patch details).

In this mode, the RDP thin client will not forward a copy of the user's credentials to the remote computer (which is the default with CredSSP authentication), hence, if the remote computer has been compromised by hackers or malware, these particular credentials will not be available to steal for pass-the-hash attacks.

Also, this feature requires that both the RDP client and the target computer must be running one of the supported operating systems mentioned above. This is not just a client-side feature of the mstsc.exe tool itself.

If you wish to force the use of restricted administration mode with RDP on Windows 8.1 and later clients, there is a GPO option for this (GPO > Computer Configuration > Policies > Administrative Templates > System > Credentials Delegation). In this same GPO container you will find other options to control CredSSP credentials sharing, but be careful not to shoot yourself in the foot: at the end of the day, you still have to be able to manage the network even if there are pass-the-hash and SAT abuse risks.

For both restricted admin mode and remote credential guard (below), the following registry value must be set:

Key: HKLM\System\CurrentControlSet\Control\Lsa
Value: DisableRestrictedAdmin
Value Type: DWORD
Value Data: 0

Second Hop Problems

Note that after connecting with RDP to a remote computer in restricted administration mode, when you attempt to connect from that remote computer to a third machine, you will be prompted for credentials. If the remote computer has been compromised, when you enter your credentials again, it's another opportunity for those credentials to be stolen. Hence, when practical, avoid providing your credentials manually after connecting to remote computers with RDP in restricted administration mode; instead, it would be slightly better to directly connect to each target instead of "hopping" from one machine to the next.

Also, please see KB2973351 and KB2975625 for further complications, because Microsoft has changed what is manageable when using restricted admin mode.

What can be done to make these kinds of "second hop problems" less annoying?

RDP Remote Credential Guard

With Server 2016, Windows 10 version 1607, and later operating systems, there is a new command-line switch for the mstsc.exe tool to help make RDP more secure and less annoying for "second hop" issues:

```
mstsc.exe /RemoteGuard
```

To use the /RemoteGuard command-line switch, the requirements are:

- Both client and RDP server must be running Windows 10 version 1607, Server 2016, or later.
- Neither client nor server can be a stand-alone.
- Neither client nor server can be joined to Azure Active Directory, they must be joined to an on-premises Active Directory domain (or mutually-trusted domains).

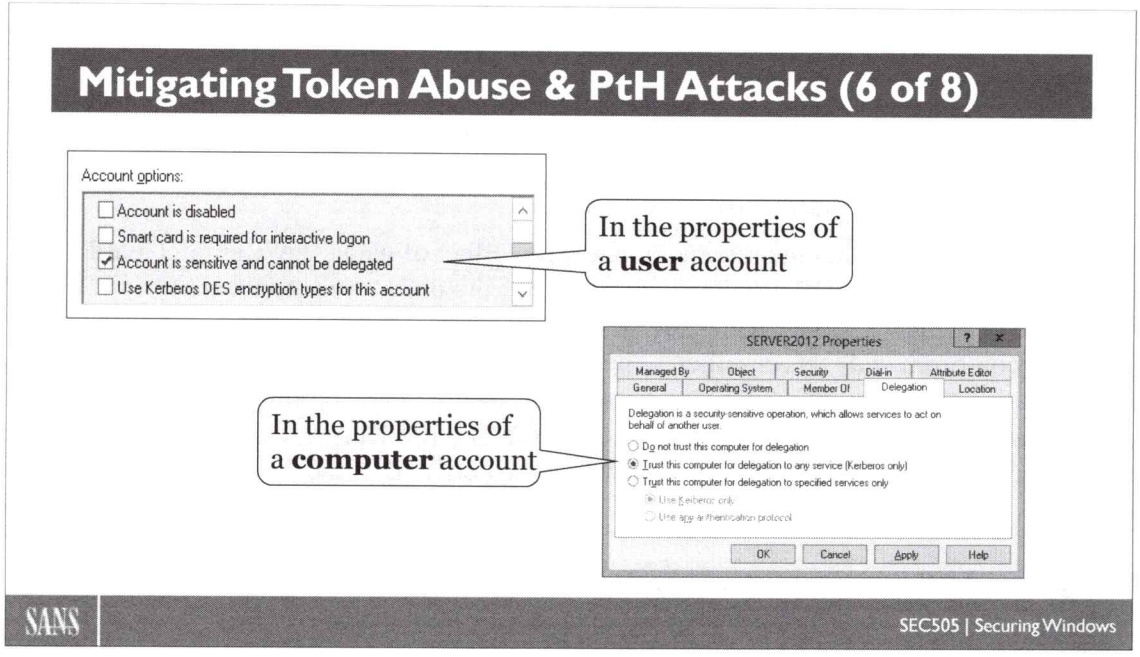
- The Remote Desktop Gateway for RDS cannot be used.
- The client cannot provide alternative credentials; single sign-on is mandatory.
- The client must authenticate with Kerberos, not NTLM.

When the above requirements are met, what is the effect of using the /RemoteGuard switch? Just like with RDP restricted admin mode, the user's credentials are not forwarded to the RDP server, hence, these credentials are not held in memory at the (possibly compromised) server either. The difference is that the user's Kerberos authentication to the target server is redirected back to the user's computer for authentication. This means that 1) the user's own computer Kerberos credentials are not being used, as when using the /RestrictedAdmin switch, and 2) the "second hop" problem is solved.

The "second hop" problem is solved because, if the user attempts to connect to a third machine from the RDP server, the Kerberos authentication necessary for this third "hop" is transparently redirected back to the user's computer where the authentication succeeds without the user being prompted for any credentials (it's single sign-on) and none of the user's credentials are exposed to either the RDP server or the third machine being accessed.

This is a major victory for both convenience and security, at least as far as in-memory credentials go, but please don't forget that Security Access Tokens (SATs) are still being created in the memory of all three computers involved (client, RDP server, and third machine) and these SATs may themselves still be abused.

If you wish to enable remote credential guard, or to make it mandatory, or to fall back to RDP restricted administration mode when remote credential guard is not available, then there is an GPO option for this (GPO > Computer Configuration > Policies > Administrative Templates > System > Credentials Delegation, then see the option named "Restrict delegation of credentials to remote servers").

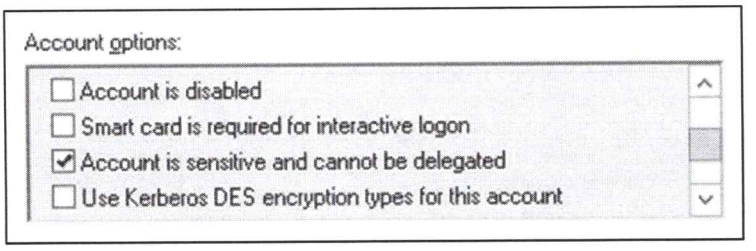


Account Is Sensitive And Cannot Be Delegated

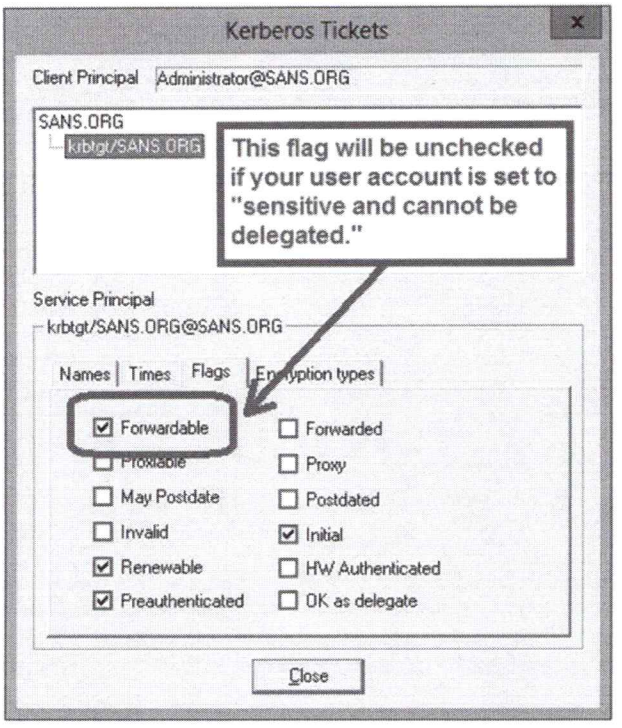
When a user performs a network logon to a remote server (not interactive), usually that server will not create a delegation SAT for the user. But if a server has been marked in AD as trusted for delegation, then the server can attempt to create a delegation SAT anyway and request a copy of the user's Kerberos ticket-granting ticket (TGT). This special type of server trust is configured in the properties of a computer account in AD on the Delegation tab.

Even if there are servers which are trusted for delegation, if you never want to loan any server your Kerberos TGT, you can disable support for this in your user account. To do

this go to the properties of your user account in AD, Account tab, and check the box named "Account is sensitive and cannot be delegated".



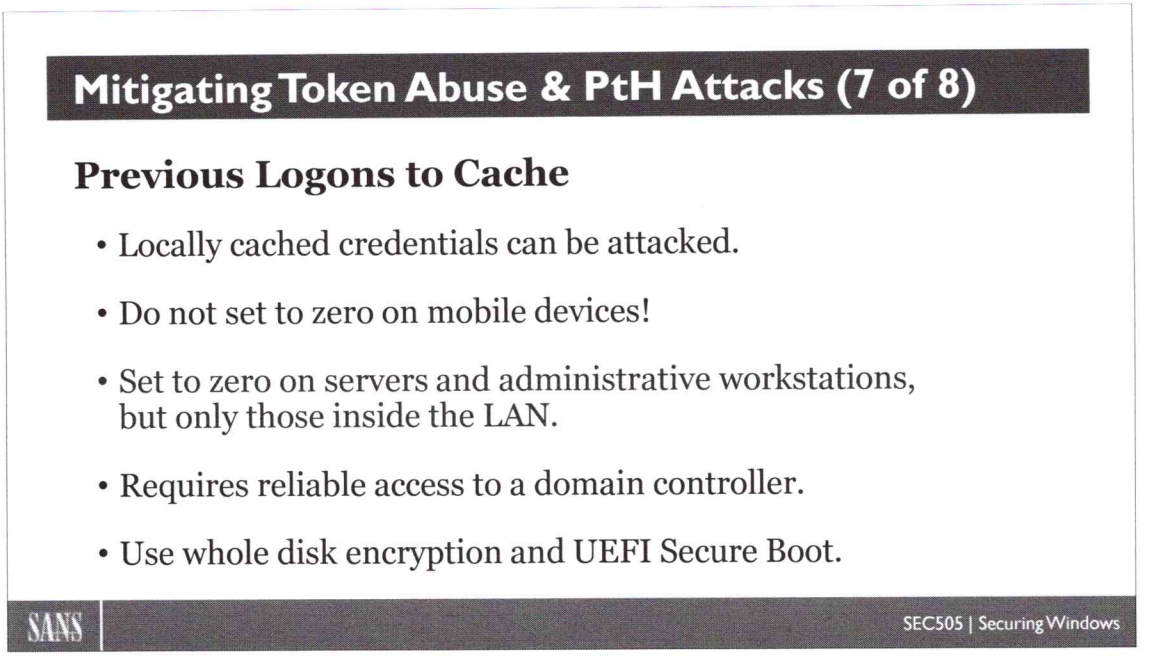
The effect of this can be seen with the KERBTRAY.EXE tool, which shows one's Kerberos tickets (it's not installed by default, it's a download from Microsoft's web site). In KERBTRAY, if your user account is marked as "sensitive and cannot be delegated", then the Forwardable flag will be turned off in your ticket-granting ticket (krbtgt/*domain* in the screenshot). This means your computer will not loan this TGT out to other systems even if they are trusted for delegation.



Note: The "OK as delegate" flag in a Kerberos service ticket is what you would have in a ticket to a server which has been trusted for delegation. This is how your computer would know that the server is trusted, i.e., from this flag being set by your domain controller in your Kerberos ticket to that server.

Unfortunately, some servers really do need your TGT in order to function correctly, even if it is rare. In these cases, though, administrators will typically have two user accounts

anyway: an administrative account (with "cannot be delegated" checked for security) and a regular account (with "cannot be delegated" unchecked for compatibility if necessary).

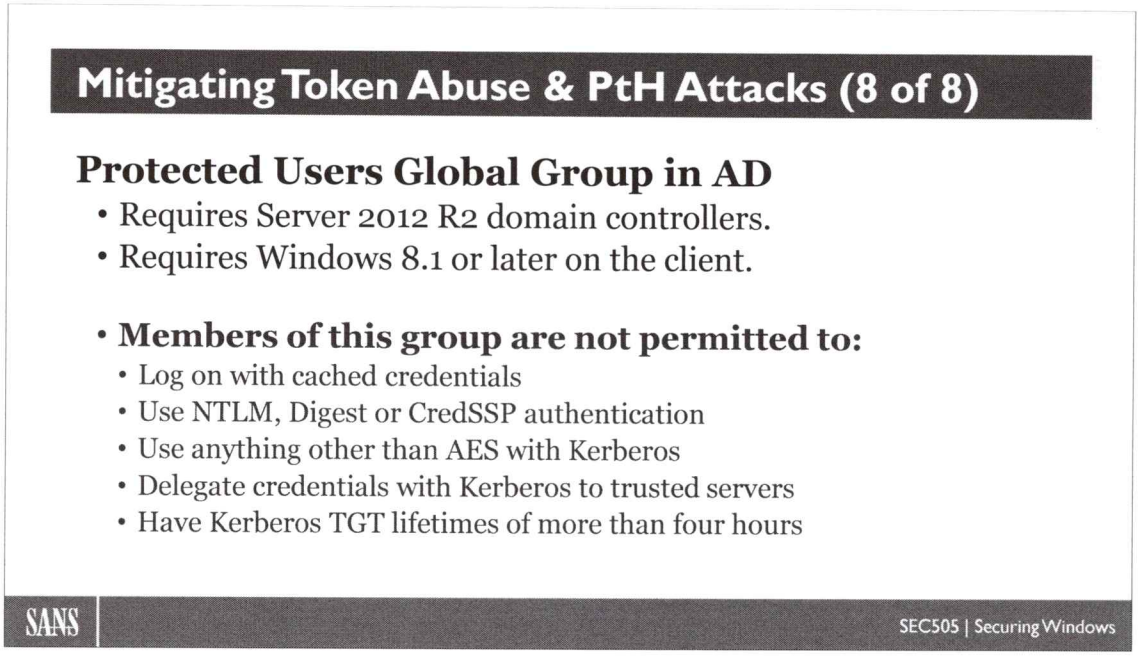


Previous Logons To Cache

Interactive logons also leave behind locally cached credentials, so it's best to set the count of locally cached credentials to either zero or one. On mobile devices it will have to be set to at least one because they will often not have access to any domain controllers while on the road. On internal workstations, when it is likely a user will log on again soon, it can be set to zero or one. On internal servers, though, sometimes the only interactive logons are the RDP sessions of administrators, so on these machines it is best for security to set it to zero (and then have multiple, reliable domain controllers).

The GPO setting for cached credentials is located under Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options > Interactive logons: Number of previous logons to cache (in case a domain controller is not available).

To protect these on-drive cached credentials, it's also best to use whole disk encryption and UEFI Secure Boot.



Protected Users Global Group in AD

On Server 2012 R2 and later domain controllers a global group exists named "Protected Users". The group is intended for administrators and other high-value IT target users (but not for computer or service accounts).

Members of this group have special restrictions applied to them:

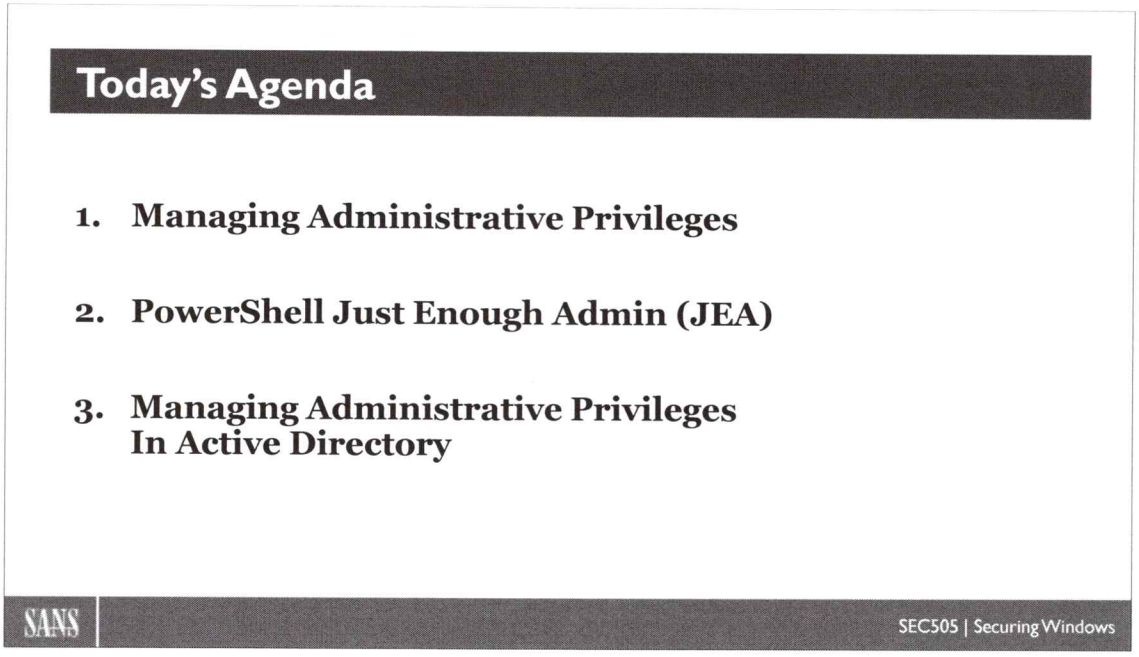
- Members cannot log on with locally cached credentials.
- Members cannot use NTLM, Digest or CredSSP authentication because the passwords and/or hashes for these protocols are not cached in memory (or available to be stolen from memory either).
- Only Kerberos authentication is permitted, and it must use AES encryption, not RC4 or DES. Other security policies can force 256-bit AES instead of 128-bit.
- Members cannot use Kerberos delegation of identity, constrained or otherwise.
- The Kerberos Ticket Granting Tickets (TGTs) of members expire after only four hours, forcing a fresh authentication again.

Membership in the Protected Users group is intended to be used along with Authentication Policy Silos to further control Kerberos authentications in Server 2012 R2 and later domains.

These restrictions require that at least the PDC Operations Master controller in the domain be running Server 2012 R2 or later, and the member user's workstation must be running Windows 8.1 or later. However, for full integration it's best if all controllers in the domain run Server 2012 R2 or later, not just the PDC Operations Master. The forest functionality level does not have to be Server 2012 R2 or later, just the intended domain.

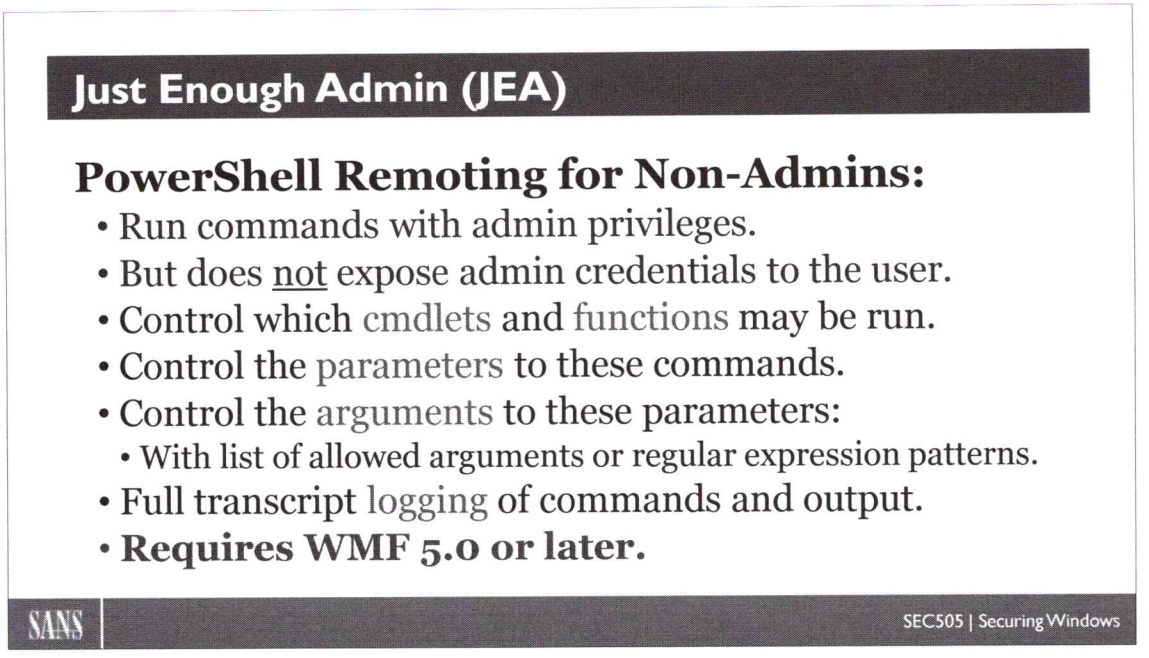
Summary

In general, just remember that when you log onto a computer *interactively* at the console or over the network, you are potentially exposing your delegation SAT and your passphrase hash to your adversaries. A standard hacking technique is to deliberately cause errors on a computer in the hope of tricking an unsuspecting Domain Admin into logging into that machine at the console or with RDP in order to investigate. Then they have you. It's a Catch-22 and difficult to deal with, but that's the reality today.



Today's Agenda

Linux has *su* and *sudo* to allow non-administrative users to execute commands with administrative privileges. The Windows version of *sudo* is PowerShell remoting enhanced with a set of features called "Just Enough Admin" (JEA). A server with a JEA remoting endpoint permits very tight control over which cmdlets and functions may be executed, as well as the parameters and arguments to these commands.



Just Enough Admin (JEA)

When too many users have too much unnecessary power, it's a recipe for disaster. Eventually, someone will be infected with malware, have their credentials stolen by hackers, or become disgruntled.

Just Enough Admin (JEA) is the Principle of Least Privilege as applied to administrators and IT personnel. JEA means we should only grant the minimum power necessary to administrators and IT personnel for them to get their legitimate work done, nothing more. Furthermore, whenever they do exercise their limited powers, these actions should be logged in detail. Later, when the inevitable compromise occurs, the hope is that this strategy will greatly limit the damage and will aid in a forensics response and recovery.

JEA is not just a general security principle, it is also the name of a built-in feature of PowerShell. Using PowerShell JEA, we can:

- Allow non-administrative IT personnel to use PowerShell remoting.
- Block all commands by default within a remoting session.
- Allow only the cmdlets, functions, aliases, and language features desired.
- Restrict which parameters may be used with allowed cmdlets and functions.
- Restrict which arguments may be passed into allowed parameters.
- Run administrative commands without granting administrative privileges.
- Log a detailed transcript of all commands and their output.

And don't forget that graphical tools can also use PowerShell remoting connections under the hood. JEA admins do not have to only run scripts and commands in a shell. If they want to use nice, intuitive, graphical applications to get their work done, this is entirely

possible, as long those applications are designed to use PowerShell remoting under the hood.

Requirements

To use JEA, the target computer where JEA will restrict commands requires:

- Windows Management Framework (WMF) 5.0 or later.
- PowerShell remoting enabled and accessible over the network.

The client computer connecting outbound to the JEA-controlled target only requires PowerShell version 2.0 or later because JEA is enforced at the target server, not at the client. The client only requires the remoting cmdlets, such as Enter-PSSession.

You do not need to use Desired State Configuration (DSC) to implement JEA, but leveraging DSC will make it much easier to deploy and monitor JEA in a large environment. JEA and DSC are often discussed together, but DSC is not a requirement for using JEA.

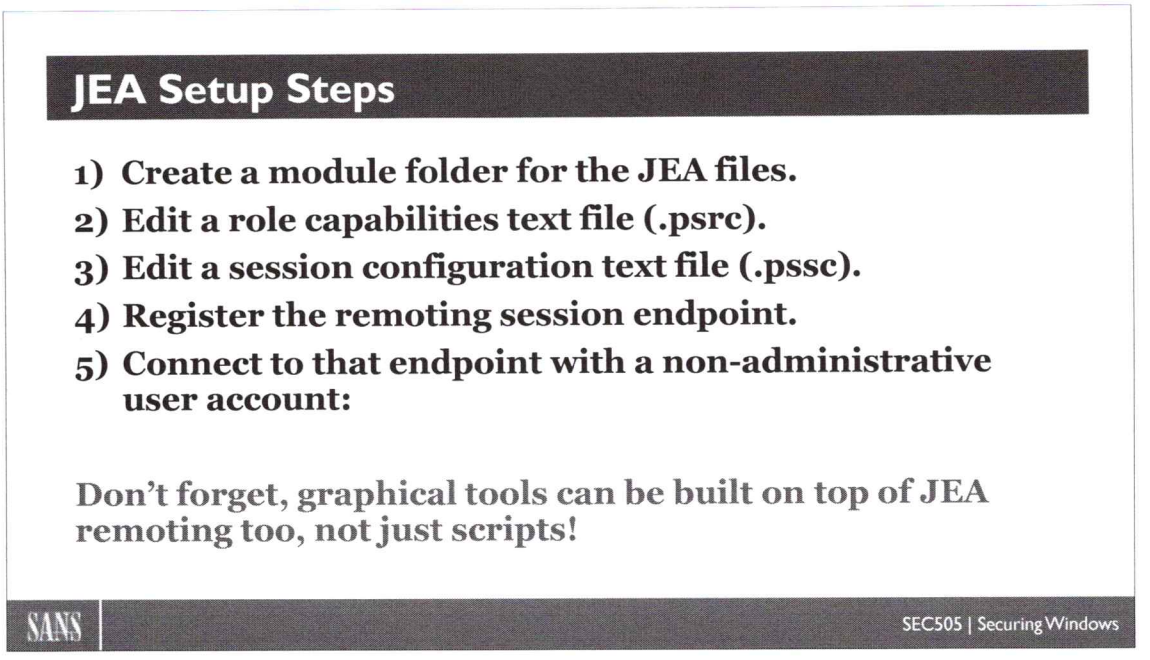
Reference

Just for reference, here are the different file types associated with JEA:

File or Folder Type	Description
.pssc	Session configuration file to define a remoting endpoint.
.psrc	Role capabilities file to list allowed commands per group.
.psd1	Data file, such as for a module manifest.

Here are the PowerShell cmdlets related to JEA:

Cmdlet Name	Description
Get-PSSessionCapability	List per-user endpoint restrictions.
Get-PSSessionConfiguration	Lists available session endpoints.
New-ModuleManifest	Creates a .psd1 file.
New-PSRoleCapabilityFile	Creates a .psrc file.
New-PSSessionConfigurationFile	Creates a .pssc file.
Register-PSSessionConfiguration	Creates a named session endpoint.
Test-PSSessionConfigurationFile	Validate the contents of a .pssc file.
Unregister-PSSessionConfiguration	Remove a named session endpoint.



JEA Setup Steps

There are several steps in the setup and use of Just Enough Admin (JEA):

1) Create a module folder to hold the JEA files.
2) Create and edit a role capabilities text file (.psrc).
3) Create and edit a session configuration text file (.pssc).
4) Register the remoting endpoint with the session file (.pssc).
5) Connect to that endpoint as a JEA-restricted user.

In the next few pages, let's discuss each step.

Requirements

These steps assume that the target computer has at least WMF 5.0 installed, PowerShell remoting is enabled, and the remoting ports are accessible (TCP 5985 and 5986).

The requirement for WMF 5.0 or later might seem tough, but remember that JEA is mainly intended to be used on servers, and upgrading PowerShell on servers is much easier than upgrading on thousands of endpoints.

Remoting is enabled by default on Windows Server 2012 and later server operating systems. Remoting is not enabled by default (yet) on any client operating systems, such as Windows 10, but remoting can be enabled on clients through Group Policy.

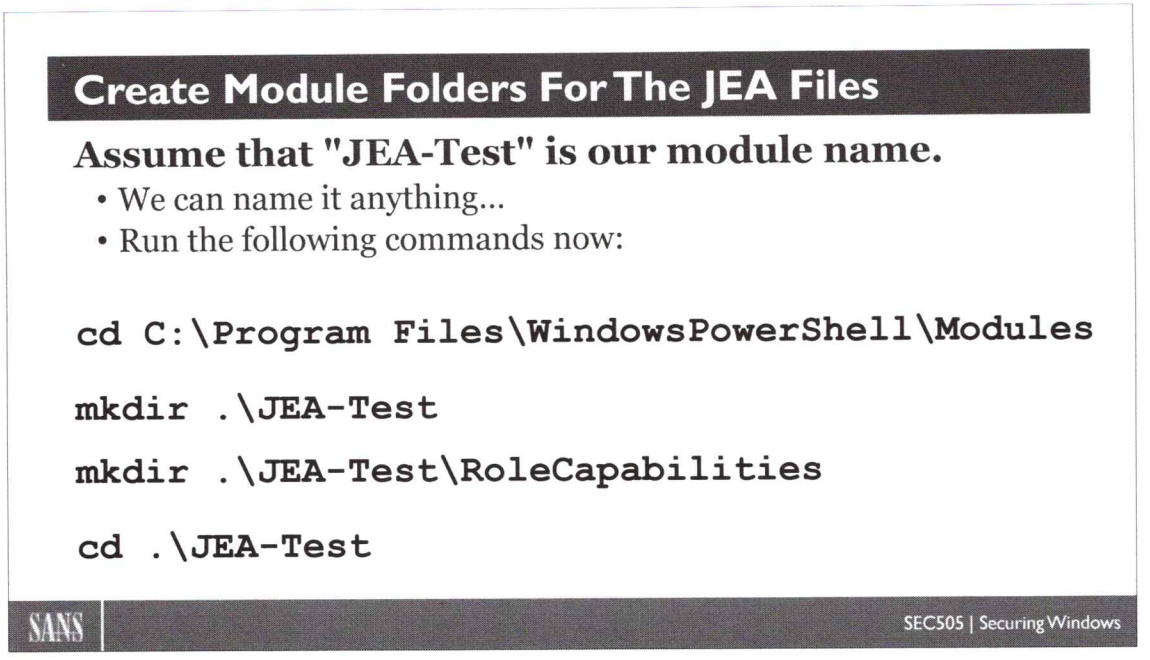
If there is any doubt whether remoting is enabled, run this command:

```
Enable-PSRemoting -Force
```

PowerShell remoting uses TCP port 5985 when using its native encryption, and TCP port 5986 when using SSL/TLS. It's highly recommended that access to these ports be limited by source IP address and IPSec using the Windows Firewall.

Graphical Tools Can Use JEA Too!

This course emphasizes the use of PowerShell in the console by running scripts, but please don't forget that graphical tools can use PowerShell and WinRM remoting in the background too. A nice, friendly GUI tool could be written in C# or C++ which invokes PowerShell in the background. A non-technical user could be using JEA and not even know it. Graphical tools could be written by your organization, but most of your JEA-integrated GUI tools will probably be third-party commercial or FOSS products that you install like any other app.



Create Module Folders For The JEA Files

It's best if the files related to JEA are stored in a new PowerShell module folder. Using a module will help with version control, permissions, and management of JEA through Desired State Configuration (DSC). Using DSC to manage JEA is not required though.

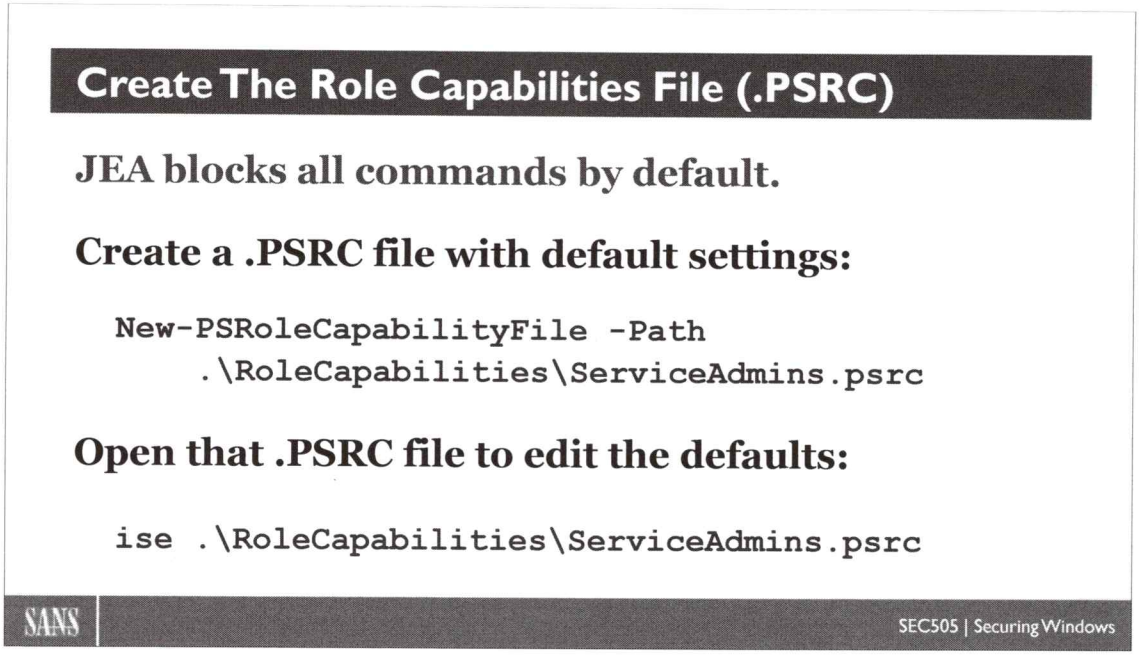
The module folder can be named anything. Just make sure the name will not conflict with any other JEA-related modules from Microsoft or others. For this course, we will use "JEA-Test" as the name of our module folder. Hence, create these two folders:

- C:\Program Files\WindowsPowerShell\Modules**JEA-Test**
- C:\Program Files\WindowsPowerShell\Modules**JEA-Test\RoleCapabilities**

The above folders can be created with these commands:

```
cd C:\Program Files\WindowsPowerShell\Modules

mkdir .\JEA-Test

mkdir .\JEA-Test\RoleCapabilities

cd .\JEA-Test
```

When deploying JEA across an enterprise, these folders can be created using Desired State Configuration (DSC), Group Policy, PowerShell remoting, a third-party configuration management system, a custom build script, or any other method of creating folders.



Create The Role Capabilities File (.PSRC)

A PowerShell Role Capabilities file (PSRC) is a text file with a ".psrc" file name extension. The PSRC file describes which aliases, functions, cmdlets, providers, assemblies and modules are accessible to the JEA-restricted administrator when he or she remotes into the target computer.

The PSRC file blocks every command by default. Any command which the administrator requires must be added to the PSRC file. It is not possible to allow all commands by default and then only block a few unwanted commands.

The name of the PSRC file is somewhat important. It should be named after the role or group whose actions should be restricted by JEA. So, if you have a global group named "ServiceAdmins" and you want its members to be restricted by JEA, you might name this PSRC file "ServiceAdmins.psrc" as a reminder. This isn't require, just recommended.

The PSRC file must be placed in the \RoleCapabilities subfolder of the module folder. Hence, if the module is named "JEA-Test", place the PSRC file in here:

- C:\Program Files\WindowsPowerShell\Modules\JEA-Test\RoleCapabilities

Let's imagine we do have a ServiceAdmins global group. We need to create a PSRC file for that group. The easiest way to do this is with the New-PSRoleCapabilityFile cmdlet.

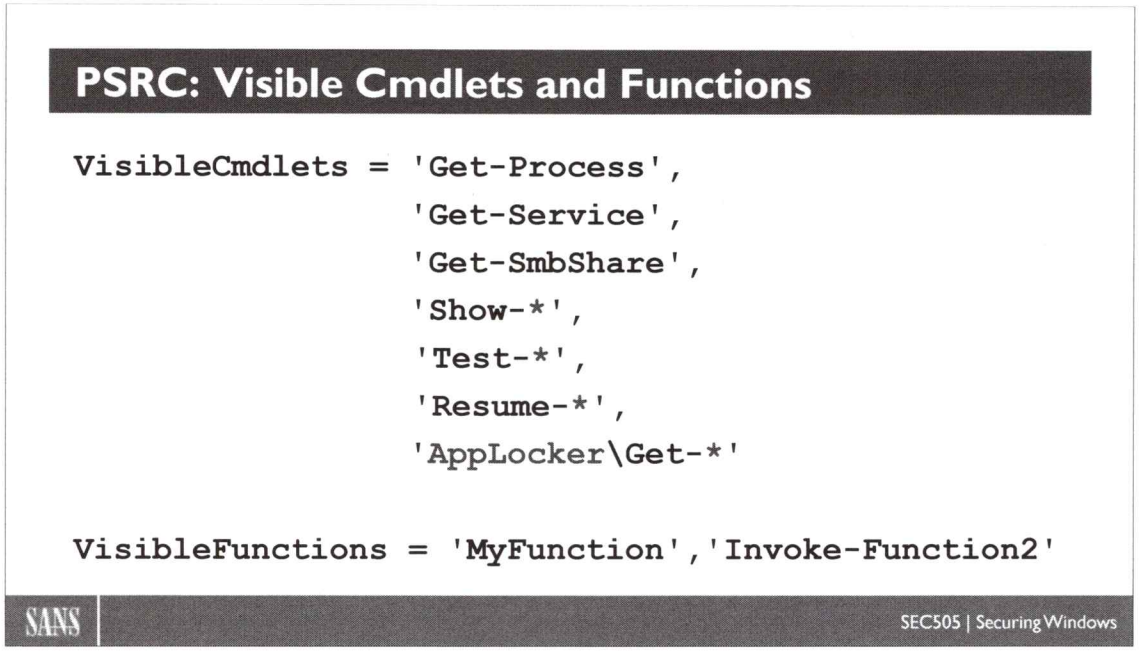
When you use the New-PSRoleCapabilityFile cmdlet to create a new PSRC file, that file will automatically be filled with comments and a few default settings to help you get started. Afterwards, the PSRC file can be edited with any text editor, including ISE.

To create a new PSRC file named "ServiceAdmins.psrc" for the JEA-Test module:

```
cd "C:\Program Files\WindowsPowerShell\Modules\JEA-Test"

New-PSRoleCapabilityFile -Path
   .\RoleCapabilities\ServiceAdmins.psrc
```

To view and edit the ServiceAdmins.psrc file afterwards:

```
ise .\RoleCapabilities\ServiceAdmins.psrc
```



PSRC: Visible Cmdlets and Functions

Our task is to edit and save the PSRC file for later use. If a command, alias, function, or other PowerShell item is not explicitly allowed in the PSRC file, it will be invisible and blocked for any user that remotes into a JEA-controlled machine using this PSRC file. It is not possible to allow all commands by default and then only block a few unwanted commands.

Here are the variables in a PSRC file for defining what JEA users can see or access:

- ModulesToImport
- VisibleAliases
- VisibleCmdlets
- VisibleFunctions
- VisibleExternalCommands
- VisibleProviders
- ScriptsToProcess
- AliasDefinitions
- FunctionDefinitions
- VariableDefinitions
- EnvironmentVariables
- TypesToProcess
- FormatsToProcess
- AssembliesToLoad

The comments and example data for the above variables in the PSRC file are fairly self-explanatory. Microsoft has done a good job in getting us started, but some of these PSRC variables need more discussion.

VisibleCmdlets and VisibleFunctions

The VisibleCmdlets and VisibleFunctions settings control which cmdlets and functions may be seen or executed by the JEA user. These are perhaps the most important PSRC settings. Each setting is a comma-delimited list. Here are examples of legal syntax for each item (remember, each item is separated by a comma from the next).

Each cmdlet or function can be listed explicitly:

```
VisibleCmdlets = 'Get-Process','Get-Service','Get-SmbShare'

VisibleFunctions = 'MyFunction','Invoke-Function2'
```

A wildcard may be used to include multiple cmdlets or functions:

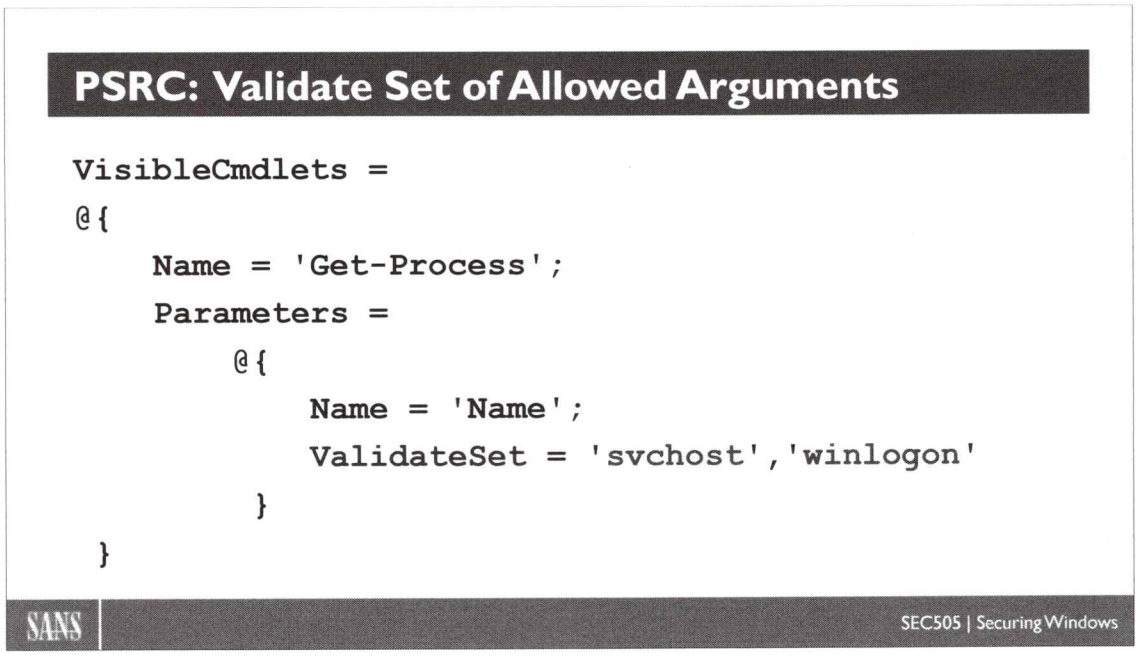
```
VisibleCmdlets = 'Show-*','Test-*','Resume-*'
```

The wildcard may be combined with a module name:

```
VisibleCmdlets = 'AppLocker\Get-*','PKI\*'
```

The list of visible cmdlets or functions may span multiple lines in the PSRC file, which makes reading and editing these lists much easier. Here is an example:

```
VisibleCmdlets = 'Get-Process',
                 'Get-Service',
                 'Get-SmbShare',
                 'Show-*','Test-*','Resume-*',
                 'AppLocker\Get-*',
                 'Microsoft.PowerShell.Management\*'
```



PSRC: Validate Set of Allowed Arguments

A particular cmdlet or function can be allowed, but then limited with regard to 1) which of its parameters may be invoked and 2) which arguments may be passed into these parameters. The arguments can be limited to a set of explicit values (ValidSet) or limited by whether the argument matches a regular expression pattern (ValidatePattern).

To only allow the -Name parameter with the Get-Process cmdlet, and only allow "svchost" or "winlogon" to be passed in as an argument to the -Name parameter:

```
VisibleCmdlets =
@{
    Name = 'Get-Process';
    Parameters =
        @{
            Name = 'Name';
            ValidateSet = 'svchost','winlogon'
         }
 }
```

It's easy to get lost in the complex syntax. The VisibleCmdlets setting can be given a list of items separated by commas. In this case, there is no comma because the VisibleCmdlets list only has one item, and that item is a hashtable. So, a comma-delimited list may have an entire hashtable as one of its items, but if this is the only item on the list, there is no comma.

When a hashtable has two pairings inside of it, each pairing is separated from the next by a semicolon, hence, it has the following syntax:

```
@{ key1 = value1 ; key2 = value2 }
```

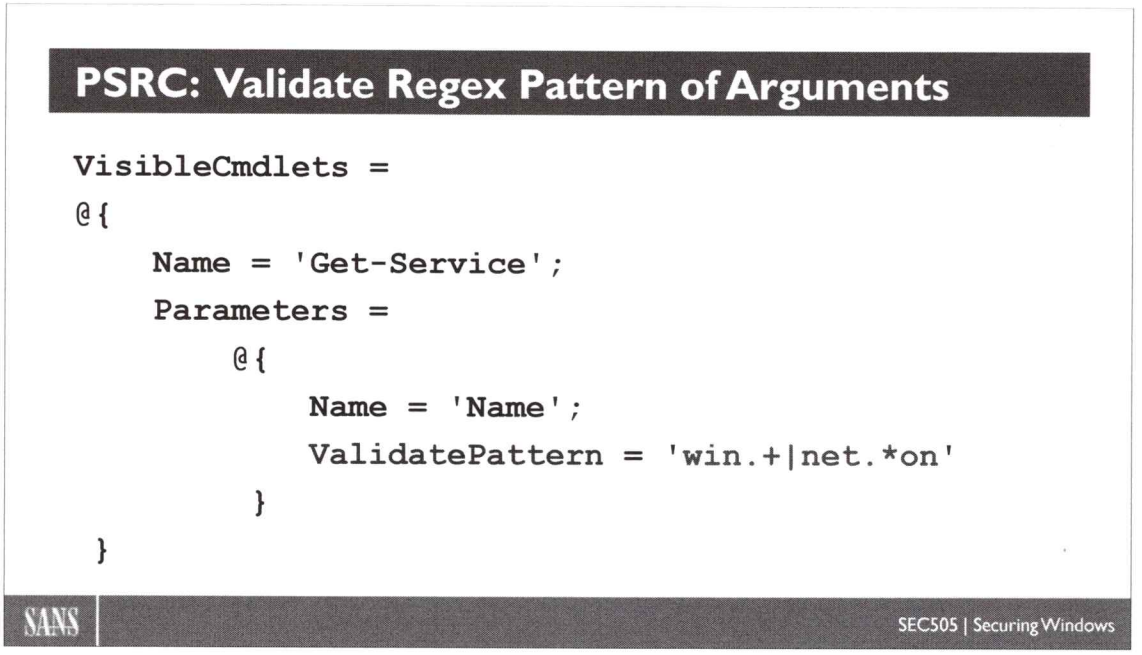
Each item in a hashtable is a pairing. A pairing is made with an equal sign ("="). Hence, from the PSRC example above, this is the first pairing in the hashtable:

```
Name = 'Get-Process'
```

And this is the second pairing in the hashtable:

```
Parameters = @{ Name = 'Name'; ValidateSet = 'svchost', 'winlogon' }
```

Notice that the second pairing contains another hashtable! The Parameters key is paired with a value that is itself a whole hashtable. So, the VisibleCmdlet list has just one item, which is a hashtable, and the value of that hashtable contains a second hashtable. The second hashtable refers to the name of the parameter for Get-Process, the "-Name" parameter, and its two valid arguments: svchost or winlogon. So, when you see "Name='Name'" in that second hashtable, it is referring to the name of the parameter, which just happens to be the -Name parameter (just to make it as confusing as possible).



PSRC: Validate Regex Pattern of Arguments

Let's look at another example, but this time using a regular expression pattern to limit what arguments may be used. In this example, only the Get-Service cmdlet is permitted, the only parameter allowed to Get-Service is the -Name parameter, and any argument given to the -Name parameter must match a regular expression ('win.+|net.*on').

> **Note:** The regular expression pattern used with ValidatePattern is not case sensitive. There are many tutorials on the Internet for writing regex patterns.

To only allow the -Name parameter with the Get-Service cmdlet, and only allow arguments to the -Name parameter that match a particular regex pattern (win.+|net.*on):

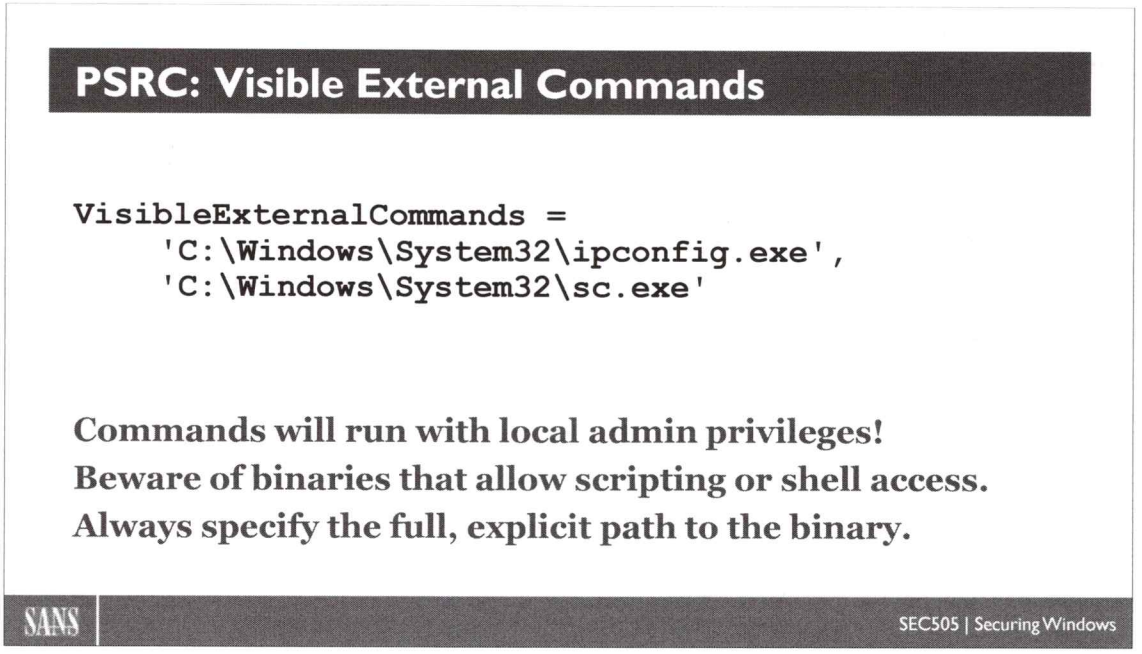
```
VisibleCmdlets =
@{
    Name = 'Get-Service';
    Parameters =
        @{
            Name = 'Name';
            ValidatePattern = 'win.+|net.*on'
        }
 }
```

The only difference between this example and the prior one is that it uses the ValidatePattern keyword instead of the ValidateSet keyword. But otherwise the hashtable syntax is the same.

But what if we want to allow both Get-Process and Get-Service? What if we wanted to combine many of the examples above? Remember, the VisibleCmdlets setting is actually a comma-delimited list, so we separate each item with a comma, like this:

```
VisibleCmdlets =
'Get-SmbShare',
'Show-*','Test-*','Resume-*',
'AppLocker\Get-*',
'Microsoft.PowerShell.Management\*',
@{
    Name = 'Get-Process';
    Parameters =
        @{
            Name = 'Name';
            ValidateSet = 'svchost','winlogon'
         }
 },
@{
    Name = 'Get-Service';
    Parameters =
        @{
            Name = 'Name';
            ValidatePattern = 'win.+|net.*on'
         }
 }
```

Note that when a cmdlet or function has limited parameters, it does not mean that one of the allowed parameters *must* be used every time the cmdlet or function is executed; rather, if an optional parameter is used, that parameter has to be on the allowed list. In the example above, it would be permitted to run Get-Service with no parameters or arguments at all, but if any parameter is used, it could only be the -Name parameter.



PSRC: Visible External Commands

Not every command is a PowerShell cmdlet or function. An administrator might need to run legacy binaries or scripts written in other languages. When allowing external commands, sometimes called "native" commands, be sure to include the full path.

```
VisibleExternalCommands = 'C:\Windows\System32\ipconfig.exe',
                          'C:\Windows\System32\sc.exe'
```

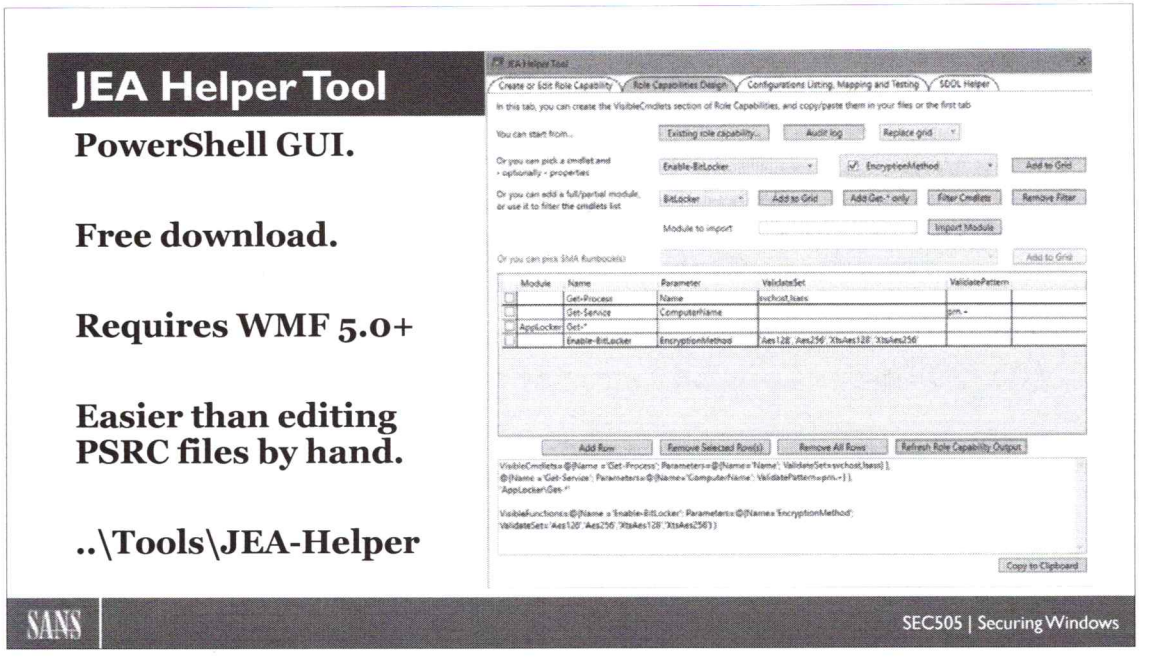
Be careful of allowing any binary executable which has its own scripting capabilities or which indirectly may allow further commands to be executed outside of that binary.

ScriptsToProcess

The ScriptsToProcess setting can be used to execute one or more PowerShell scripts automatically after a user connects to the endpoint. This is very handy for "JEA logon scripts", but also a bit dangerous. Make sure to specify the full path to the script and then protect that script against malicious modification.

ModulesToImport

A best practice for security is to explicitly control which modules are imported. When you don't use the ModulesToImport settings, modules are supposed to be imported automatically on an as-needed basis, but it works more reliably if any necessary modules are imported explicitly. The downside to explicitly controlling the import of modules is that you now have to know which modules are necessary and then import them. To maximize security, or to ensure application version compatibility, you can also import a module by version and GUID number, but beware of complexities this may create related to patch management and handling upgrades.



JEA Helper Tool

Instead of editing JEA configuration files by hand, you can use a free, graphical tool to do most of the work for you. The JEA Helper Tool is a free download from Microsoft. The quickest way to find it is to just search on "jea helper tool download", which will take you to the correct microsoft.com download page.

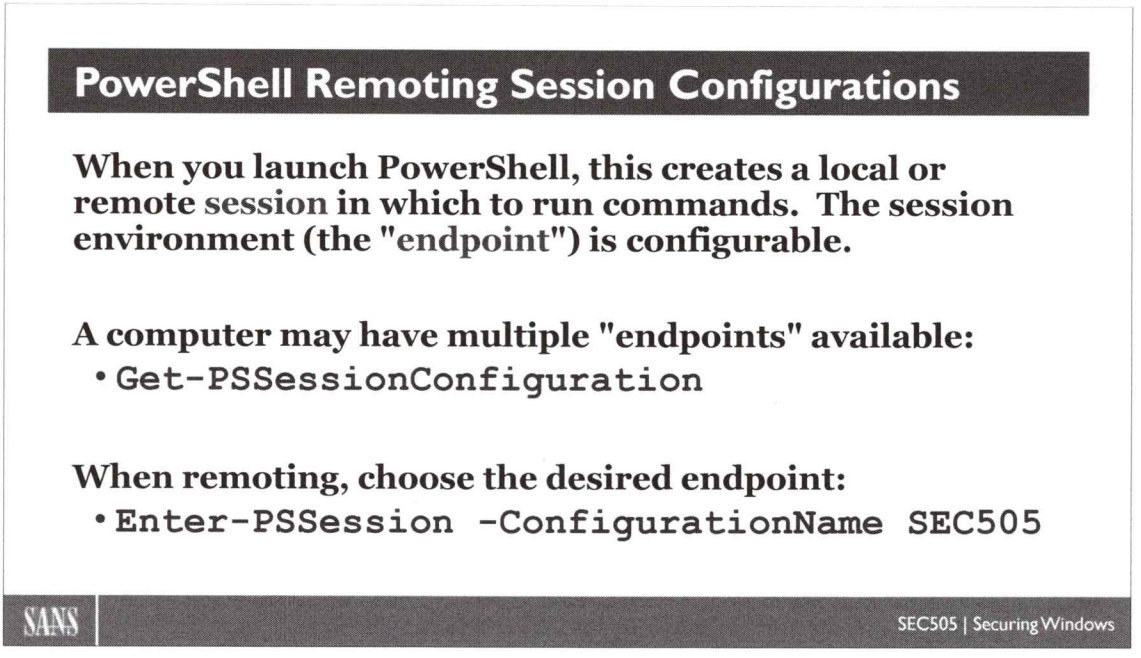
The JEA Helper Tool is written in PowerShell. It uses XAML to create its graphical interface. Peruse the source code if you want to see how it's done.

In the ISE editor, open a new PowerShell tab (File menu > New PowerShell Tab) and run the JEAHelperTool.ps1 script. Do it in a new PowerShell tab because the tool seizes control of the PowerShell instance which launches it, and the tool displays information in the command shell too, so it's best to have a separate shell just for the tool.

The JEA Helper Tool requires WMF 5.0 or later.

The JEA Helper Tool can help with the following:

- Create and edit PSRC capabilities files.
- Create and edit PSSC session configuration files.
- Easily switch between different session configurations during testing.
- For a given user or group, display resultant allowed commands.
- Manage SDDL permissions strings, such as for multi-factor authentication.



PowerShell Remoting Session Configurations

A PowerShell "session" is an initial launch of PowerShell by a user (or another identity known to Windows) on a local or remote computer. Using PowerShell remoting to connect to another computer creates a new session in the memory of that remote computer.

When you disconnect from a remote computer, the session is normally destroyed, but it's possible to disconnect and keep the session alive in memory, which is called a "disconnected session", and you could then later connect back to that same session instead of creating a new one. This is similar to how you can disconnect from a Remote Desktop Protocol (RDP) session on a remote computer, then reconnect to that same desktop again later; but if you log off from an RDP session, that session goes away, and when you reauthenticate to that same machine later, you get a whole new RDP session. It's similar for PowerShell remoting sessions.

A session includes any child instances of PowerShell that stem from the initial launch, such as when one script runs another script. The initial PowerShell launch is authenticated using either implicit single sign-on or with explicit credentials provided, such as with the Get-Credential cmdlet. The initial or "parent" PowerShell launch creates the top-most scope for finding variables, functions, drives and other internal objects that your executable code might look for and use.

Importantly, the details of how a PowerShell session operates is not written in stone. Like everything else, a PowerShell session is a type of object, so it has properties, and Microsoft permits us to modify some of these properties. This changes how the session behaves while it is alive. JEA is built on top the customizability of PowerShell sessions.

Session Configurations ("Endpoints")

A set of session configuration settings can be given a name and saved for repeated use. A named and saved set of session options is called a "session configuration", which is a bit long to say, so it's also called a "session endpoint" or just "endpoint."

To see what session configurations (endpoints) are saved and available on your computer:

```
Get-PSSessionConfiguration
```

To see all the details of the session configurations (endpoints) defined on your computer:

```
Get-PSSessionConfiguration | Format-List *
```

Notice that every endpoint has a name and a set of permissions to define which groups are permitted to use that session configuration. This is important: session configurations have permissions. Not just anyone can use or connect to any endpoint he or she wishes.

The default session configuration is named "Microsoft.PowerShell". This is what you get by default when you launch PowerShell locally on your own computer or when you remote into another machine. If you launch the 32-bit (x86) version of PowerShell for some reason, you will use the "Microsoft.PowerShell32" configuration instead.

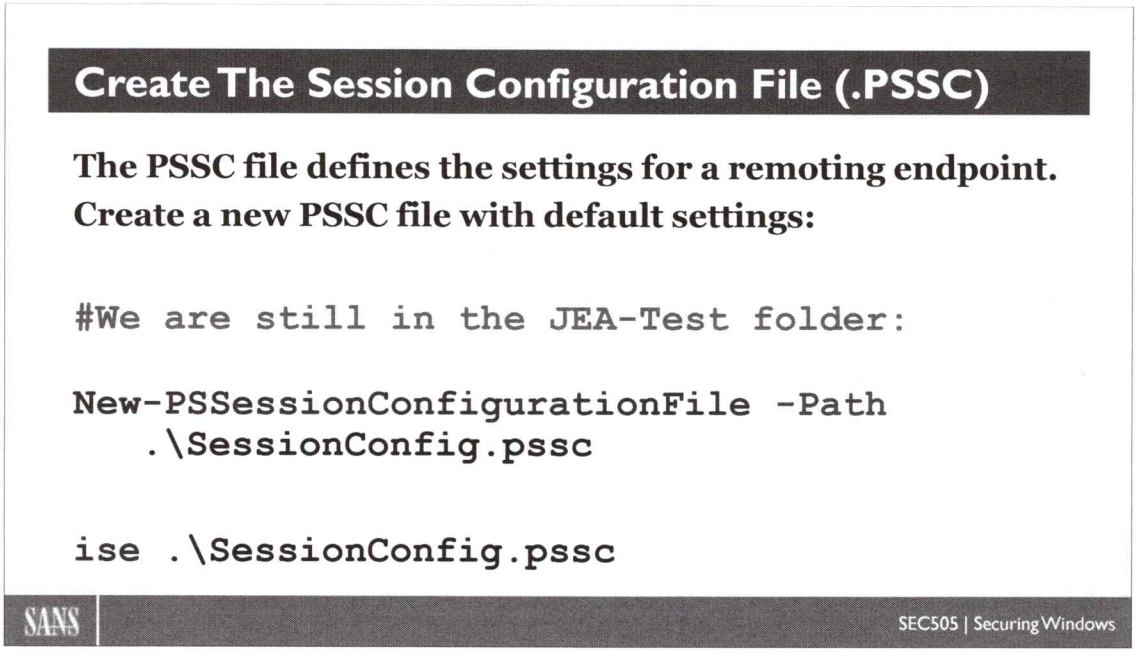
When remoting into another box, the Microsoft.PowerShell endpoint is used by default, but the -ConfigurationName parameter can be used to choose a different endpoint:

```
# The following remoting command:

Enter-PSSession -ComputerName Server47



# Uses the same endpoint (session configuration) as this command:

Enter-PSSession -ComputerName Server47 -ConfigurationName
  Microsoft.PowerShell
```

So, if there were an endpoint named "SEC505", you could remote to that endpoint:

```
Enter-PSSession -ComputerName Server47 -ConfigurationName SEC505
```

But how do we define our own custom endpoints for the sake of JEA?



Create The Session Configuration File (.PSSC)

A PowerShell Session Configuration (PSSC) file is a text file that ends with the ".pssc" file name extension. The PSSC file defines endpoint configuration settings which can override the default settings.

A new endpoint is created by inventing a name for that endpoint and giving the path to a PSSC file. Note that the name of the endpoint is not always the same as the base name of the file, e.g., the endpoint name might be "foo" while the PSSC file might be "bar.pssc". When remoting into a computer, the name of that custom endpoint (not the name of the PSSC file) can be given as an argument so that the custom PSSC settings that came from the file will be used for that session instead of the default.

How do PSRC and PSSC files relate to each other? Most importantly, a PSSC file will define which PSRC file to use for JEA. Said another way without all the acronyms, we can make a text file that says how a remoting endpoint should be customized (the .pssc file), and this customization can include a list of not-blocked cmdlets and functions from another file (the .psrc file).

PowerShell *sudo*

But it gets even better. A remoting endpoint defined by a PSSC file can grant remoting access to a group of users who are not members of the local Administrators group on that computer, but still allow them to execute JEA-permitted commands with administrative privileges! It's like RUNAS.EXE as the Administrator account, but only for the cmdlets and functions allowed by the PSRC file (all other commands are blocked by default), and you never have to share any administrative passwords with the remoting users, the

elevation of privileges all happens automatically and transparently. It's like *sudo* on Linux, but with more fine-grained control.

Finally, every command executed in this JEA session, and the output of all these commands, can be logged to text files for continuous monitoring and forensic analysis.

When you create a new PSSC file, the name of the file can be anything, as long as it ends with ".pssc", and it can be created in any folder, since it will later be copied into the correct module folder for JEA anyway.

To create a new PSSC file (the path and file name do not matter):

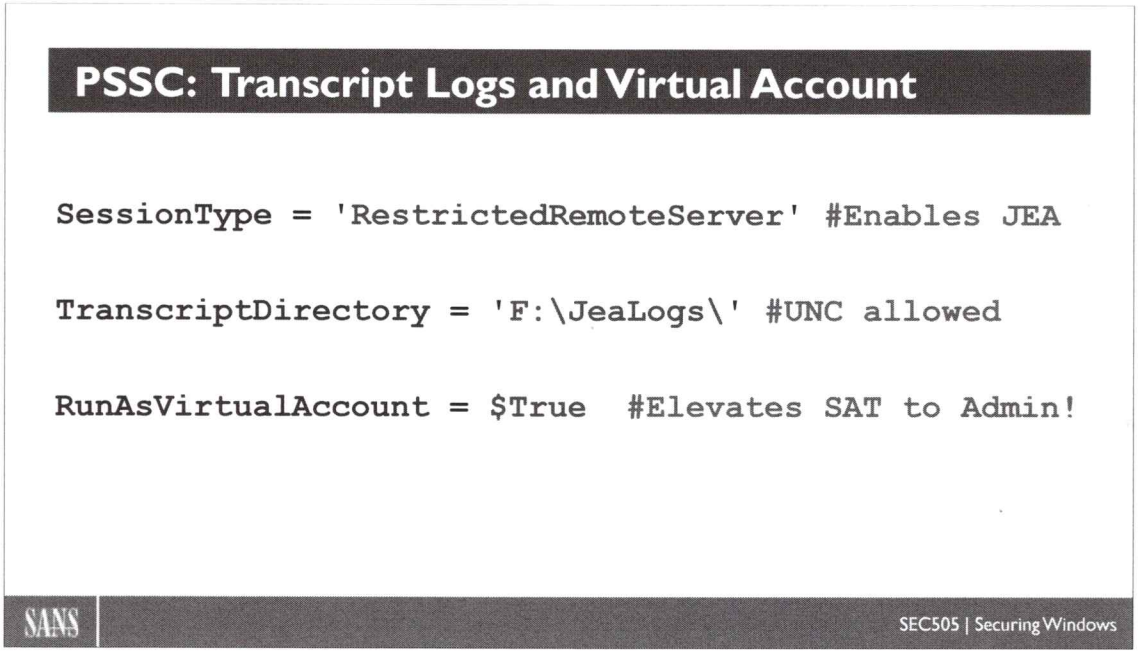
```
New-PSSessionConfigurationFile -Path .\SessionConfig.pssc
```

When you use the New-PSSessionConfigurationFile cmdlet to create a new PSSC file, that file will be automatically filled with comments and a few default settings to help you get started. You can also use the graphical JEA Helper tool to manage PSSC files.

You can edit the new PSSC file with any text editor, including ISE:

```
ise .\SessionConfig.pssc
```

The comments and variable names in the PSSC file make editing sometimes easy to figure out, but not always. Let's look at some of the more important but obscure PSSC settings.



PSSC: Transcript Logs and Virtual Account

There are four settings in the PSSC file that are mandatory for JEA:

- **SessionType**: Must be set to "RestrictedRemoteServer" to enable JEA.
- **TranscriptDirectory**: Set this to a local or UNC folder where full command-and-output transcription logs will be saved for every session using this PSSC endpoint. The NTFS permissions on this folder should not allow any access to any users whose activities are being regulated through JEA.
- **RunAsVirtualAccount**: Must be set to $True to enable JEA. This allows a non-administrative user to execute JEA-permitted commands with administrative privileges. The commands run with a Security Access Token (SAT) that includes membership in the Administrators group as an auto-generated user SID, but no new user account is actually created (it creates a virtual account, not a real one).
- **RoleDefinitions**: Must be set to a hashtable where each item defines a group and the PSRC file which restricts that group's commands (discussed next).

Here are examples of the four settings (above) inside a PSSC file:

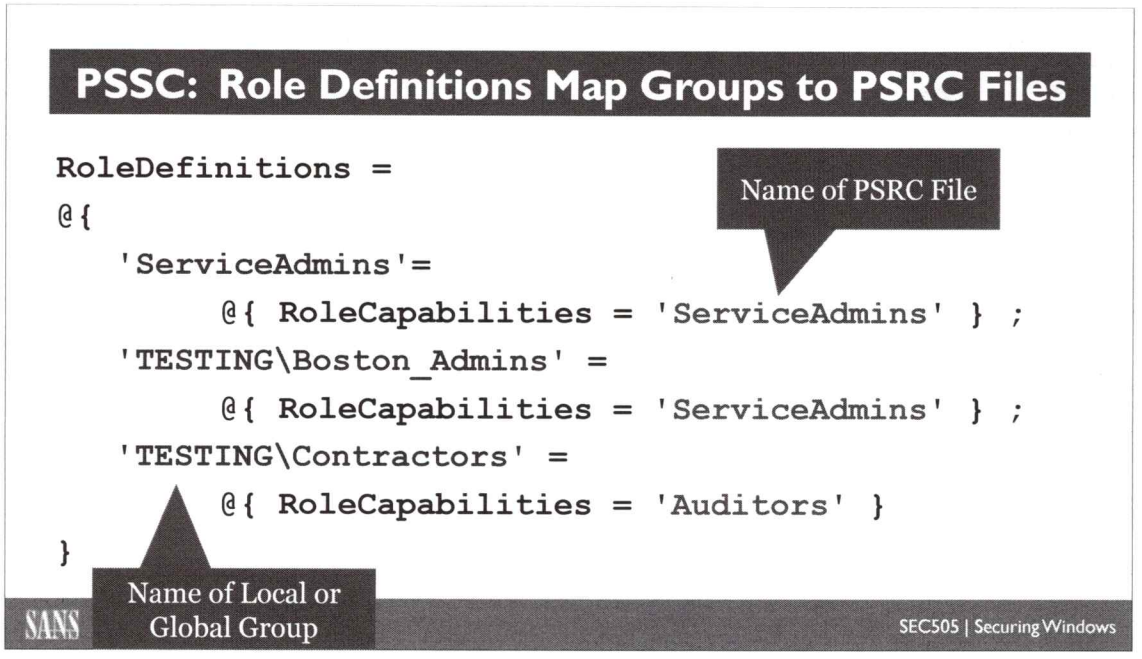
```
SessionType = 'RestrictedRemoteServer'

TranscriptDirectory = 'F:\JeaLogs\'
```

```
RunAsVirtualAccount = $True

RoleDefinitions = @{
    'ServiceAdmins' = @{ RoleCapabilities = 'ServiceAdmins' }
}
```

RoleDefinitions is set to a hashtable of the form "@{ key = value }", where the key is a string to identify a group whose members 1) are permitted to connect to this remoting endpoint and 2) who are restricted to only running the allowed commands indicated by the value in this hashtable. In your organization, you would create a new group for the sake of JEA and choose a name for it. Whatever name you choose, that name goes here. If it is a local group on the server where the JEA endpoint is being accessed, the string is just the simple name of the group, e.g., "ServiceAdmins", and if it is a global group from Active Directory, then it's the name of the group preceded by the NetBIOS name of the domain and a backslash, e.g., "TESTING\ServiceAdmins", where TESTING is the name of the domain.

When a new endpoint is registered using this PSSC file, the permissions on the endpoint will be automatically set to match the group(s) listed for RoleDefinitions. You can override these permissions if you wish, but it's not required.

The value part of the RoleDefinitions hashtable is itself another hashtable: the key is "RoleCapabilities" and the value will be the name of the PSRC file you wish to use, but minus the ".psrc" file name extension on that PSRC file; for example, if the PSRC file is named "ServiceAdmins.psrc", then just "ServiceAdmins" would be the value for the RoleCapabilities key in the hashtable (as seen above).



PSSC: Role Definitions Map Groups to PSRC Files

When someone remotes into a machine using an endpoint created with the PSSC settings in the example code above, if that user is a member of the ServiceAdmins local group on that machine, he or she will be allowed to connect, and that user's commands will be limited to only those defined in the PSRC file named ServiceAdmins.psrc. The name of the group (ServiceAdmins) matching the name of the file (ServiceAdmins.psrc) is not required, they do not have to match at all, but hopefully by choosing matching names it will be easier to understand and manage.

You may have multiple groups in RoleDefinitions. Each group may have its own separate PSRC file, or all the groups listed may share just one PSRC file, or there may be any combination of group-to-PSRC file mappings desired. And, again, the name of the group does not have to match the name of the PSRC file.

Here is an example with multiple group-to-PSRC file mappings (notice the semicolon separating each pairing):

```
RoleDefinitions = @{
   'ServiceAdmins'     = @{ RoleCapabilities = 'ServiceAdmins' } ;
   'TESTING\HelpDesk'  = @{ RoleCapabilities = 'UserAssist' } ;
   'TESTING\Contractors' = @{ RoleCapabilities = 'Auditors' } ;
   'TESTING\Managers'  = @{ RoleCapabilities = 'Auditors' }
}
```

But which PSRC file is used? Where are these PSRC files on the drive?

Recall that earlier we created these two folders for a module named "JEA-Test":

- C:\Program Files\WindowsPowerShell\Modules\JEA-Test
- C:\Program Files\WindowsPowerShell\Modules\JEA-Test\RoleCapabilities

When you're done editing the PSSC file, save it with any name you wish, but give it a ".pssc" file name extension, e.g., SessionConfig.pssc. Then copy the PSSC file into the following module folder, assuming you chose "JEA-Test" as your module name:

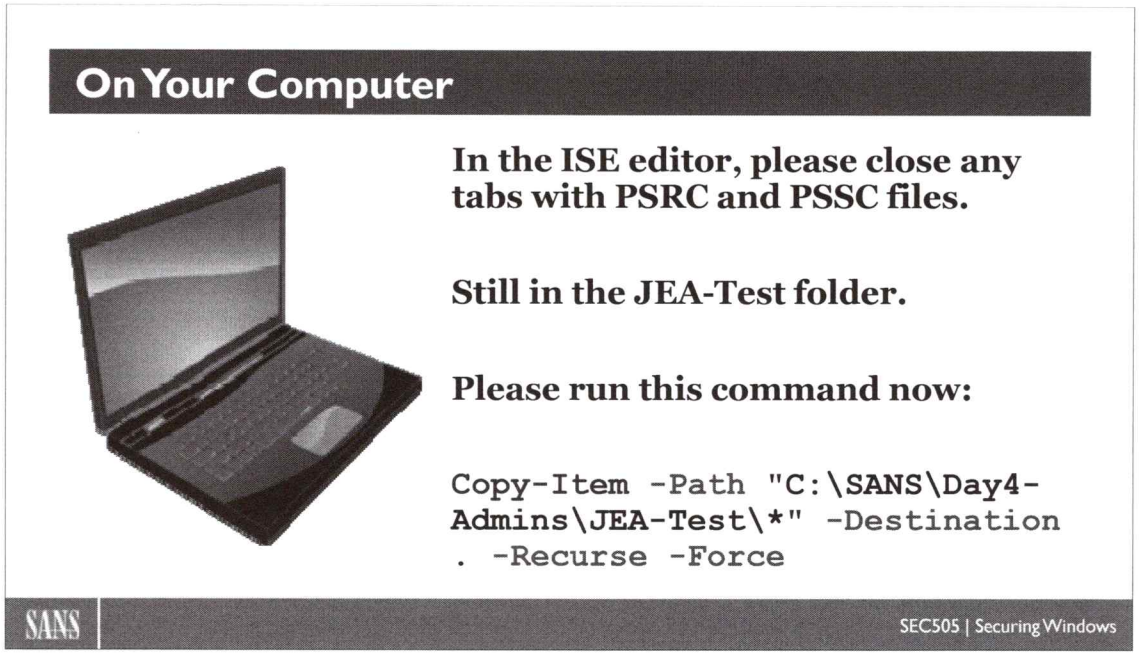
- C:\Program Files\WindowsPowerShell\Modules\JEA-Test

Then copy your PSRC file(s) into the RoleCapabilities subdirectory for that module:

- C:\Program Files\WindowsPowerShell\Modules\JEA-Test\RoleCapabilities

The names of your PSRC file(s) must match the value(s) paired with the RoleCapabilities key for each item in the RoleDefinitions hashtable in your PSSC file. Hence, using the example code above, you would have ServiceAdmins.psrc and Auditors.psrc both placed inside the \JEA-Test\RoleCapabilities folder.

> **Note:** Strictly speaking, the PSSC file does not have to be in any particular folder, but for version control (and sanity) it's recommended that you place the PSSC file into a new module folder created just for JEA.



On Your Computer

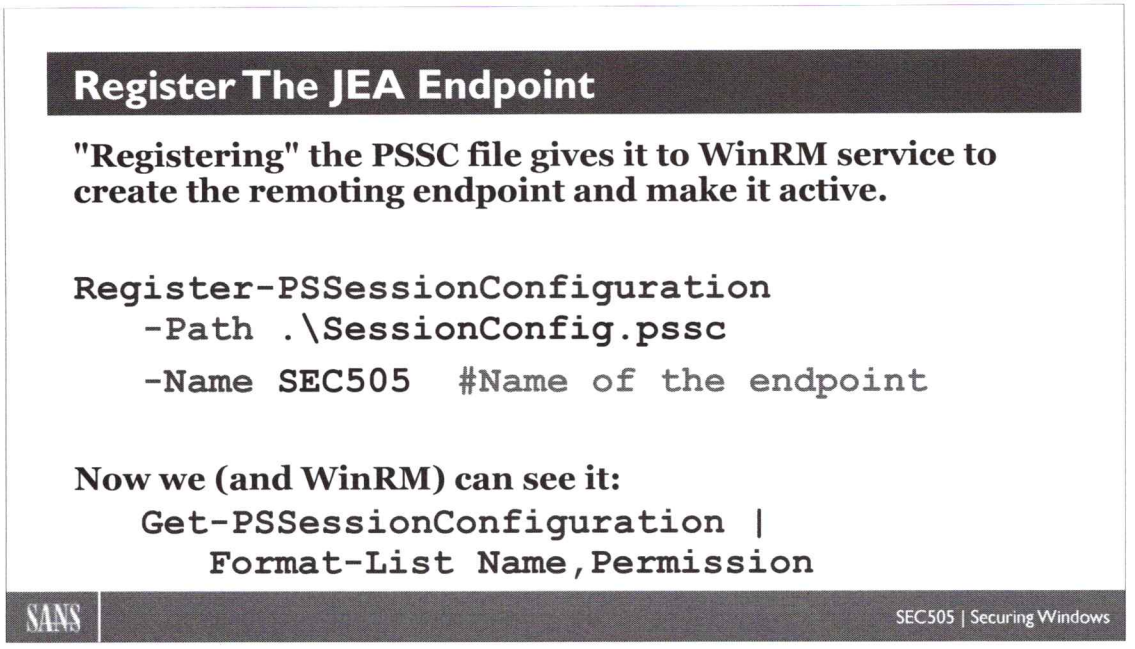
In the PowerShell ISE graphical editor, close any open tabs with PSRC or PSSC files.

In PowerShell, confirm that you are still in the .\JEA-Test folder.

Please run the following command to copy some pre-configured PSRC and PSSC files:

Note: The destination folder is ".", which the present directory (JEA-Test).

```
Copy-Item -Path "C:\SANS\Day4-Admins\JEA-Test\*"
   -Destination . -Recurse -Force
```



Register The JEA Endpoint

Once we have the correct module folder structure, a PSSC file, and one or more PSRC files, we can create the JEA remoting endpoint that uses the settings from these files.

Create Groups and Test Users

Our new PSSC file specifies one or more groups whose members are permitted to remote in and whose commands will be controlled by the PSRC file. Users in these groups should not be members of any other high-powered groups, such as Domain Admins or local Administrators, and these users must be global users, not local user accounts.

For the sake of testing, we will need to create or confirm:

- All the groups named in the PSSC file.
- Global user accounts created just for testing (not local accounts).
- Add the test users to the PSSC groups such that there is at least one user for each unique combination of group-and-PSRC role.

Before registering a new PSSC endpoint, ensure that any groups mentioned in that PSSC file actually exist. You cannot create an endpoint whose permissions use group SIDs if those groups do not exist yet. If you suspect there are permissions problems with the session configuration, these permissions can be viewed and even edited.

To view the permissions on an endpoint named "SEC505":

```
Set-PSSessionConfiguration -Name SEC505 -ShowSecurityDescriptorUI
```

Register The JEA Endpoint

Recall that an endpoint is officially known as a "session configuration", and you can see the names of the available session configurations (endpoints) on your computer like this:

```
Get-PSSessionConfiguration | Select Name
```

The above command shows the names of registered endpoints. When an endpoint is "registered", that means it exists, it is potentially accessible to remoting users (if they have the necessary permissions), and it can be listed with Get-PsSessionConfiguration. When an endpoint is unregistered, it is deleted from the computer, inaccessible, and no longer visible. A new endpoint is registered by using a PSSC file. When unregistered, an endpoint is deleted, but the associated PSSC file is not deleted. The PSSC can be used to register new endpoints again.

When an endpoint is registered using a PSSC file, an endpoint name must be chosen. The name does not have to match the name of the PSSC file in any way; it is arbitrary. Multiple endpoints can be registered using the same PSSC file as long as a different endpoint name is used each time. JEA users must know the name of the endpoint; they'll use it when connecting with the Enter-PSSession cmdlet.

To register a new PSSC session configuration using "SEC505" as the endpoint name:

```
Register-PSSessionConfiguration -Path .\SessionConfig.pssc
   -Name SEC505
```

You should now be able to see "SEC505" as the name of an available endpoint:

```
Get-PSSessionConfiguration | Select Name
```

It's usually unnecessary, but, if the endpoint doesn't work as expected, restart WinRM:

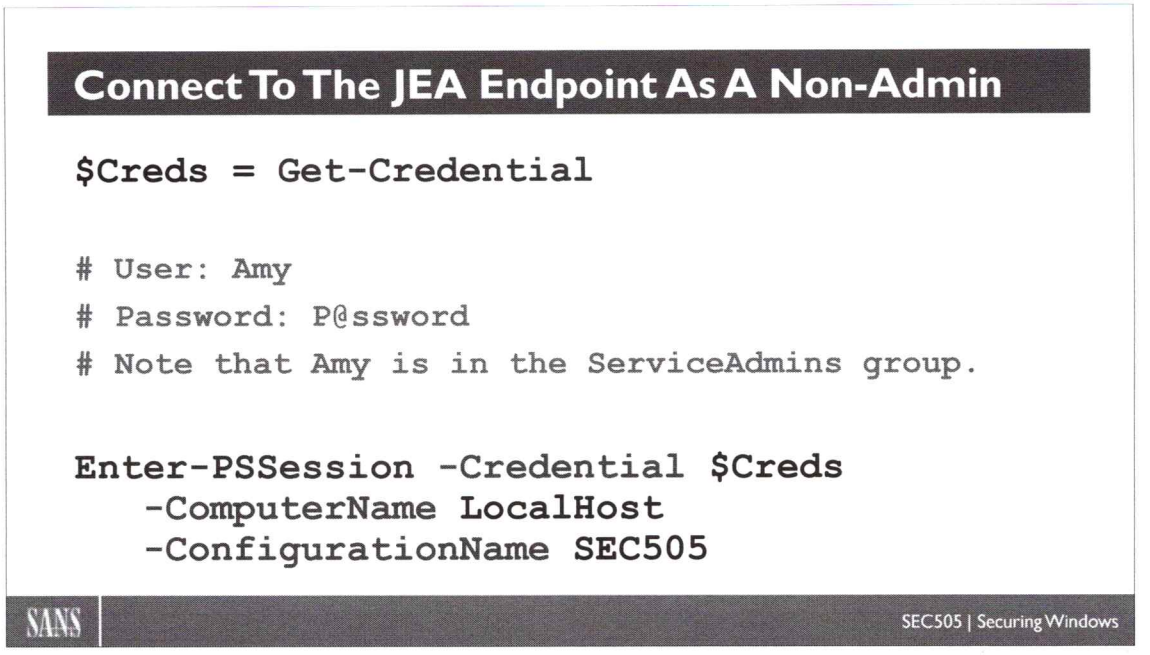
```
Restart-Service -Name WinRM        #Usually not necessary.
```

To remove a session configuration (endpoint), just unregister it:

```
Unregister-PSSessionConfiguration -Name SEC505
```

Don't worry, unregistering the endpoint will not delete your PSSC or PSRC files; nor will it delete any of the folders you created for this JEA module.

> **Note:** After modifying a PSRC file it is not necessary to re-register the session endpoint. The PSRC file is read every time a user remotes into the machine.



Connect To The JEA Endpoint As A Non-Admin

With the session configuration endpoint created for JEA, it's time to test each unique combination of PSSC group and PSRC restricted role. We will need to connect with a test user account for each group-to-PSRC file combination.

For this lab, we'll use an account (Amy) that was created by the SEC505 setup script.

UserName: Amy
Password: P@ssword

Get Amy's credentials as a credentials object:

```
$creds = Get-Credential
```

Now remote into the target system, specifying the registered name of the endpoint (SEC505) and using the explicit credentials of the test user (Amy):

```
Enter-PSSession -ComputerName localhost -ConfigurationName sec505
   -Credential $creds
```

Once you are in the session, try to run commands which should be allowed and other commands which should be blocked (commands are blocked by default). Don't forget to try external commands (like ipconfig.exe and netstat.exe) as well as cmdlets whose arguments are limited by a validation set or regular expression pattern (like Get-Service).

If you have configured a transcription logging directory in the PSSC file, such as C:\Temp\JEA-Logging\, then check out that directory after running some commands.

If the Get-Command cmdlet is permitted, run it to see a list of allowed cmdlets; blocked commands will be suppressed from the listing. Similarly, in PowerShell ISE, pull down the View menu, select "Show Command Add-On", and the Commands tab will only show the commands permitted by JEA for this session.

In particular, if the PSRC file you are testing has the following function defined:

```
FunctionDefinitions =  @{
    Name = 'Get-PSSenderInfo';
    ScriptBlock = { $PSSenderInfo }
}
```

Then, run that Get-PSSenderInfo function while remoting in as the JEA user:

```
Get-PSSenderInfo
```

The output of this function shows both the user Amy (the ConnectedUser property) as well as the virtual user account created on-the-fly to represent that user's elevated privileges (the RunAsUser property). This is not a real user account; it will no longer exist in memory after the JEA user logs off.

While in a JEA session, PowerShell runs in a special mode of operation called "NoLanguage" which restricts what can be done in that PowerShell session. To read about what's unavailable in NoLanguage mode, see:

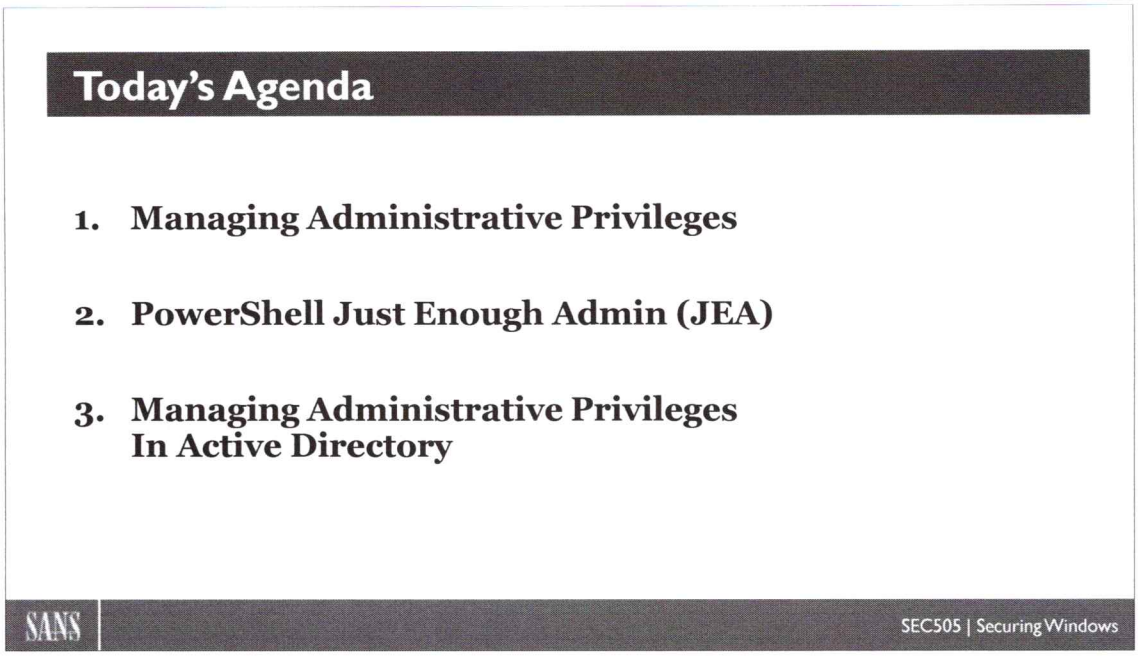
```
get-help about_Language_Modes
```

JEA Configuration Auditing

As the JEA administrator, you might wonder how to audit the capabilities permitted to each group defined in the PSSC and PSRC files. The files themselves can be examined, of course, but a handy cmdlet is Get-PSSessionCapability. With this cmdlet you can specify an endpoint name and a user name, and the cmdlet will output all the commands that user is permitted to execute with that endpoint. The output is similar in concept to what that user would see if she or he were to run Get-Command from within a session that allowed that command.

To see what commands a user (Amy) can run within a particular session (sec505):

```
Get-PSSessionCapability -ConfigurationName sec505
   -Username TESTING\Amy
```

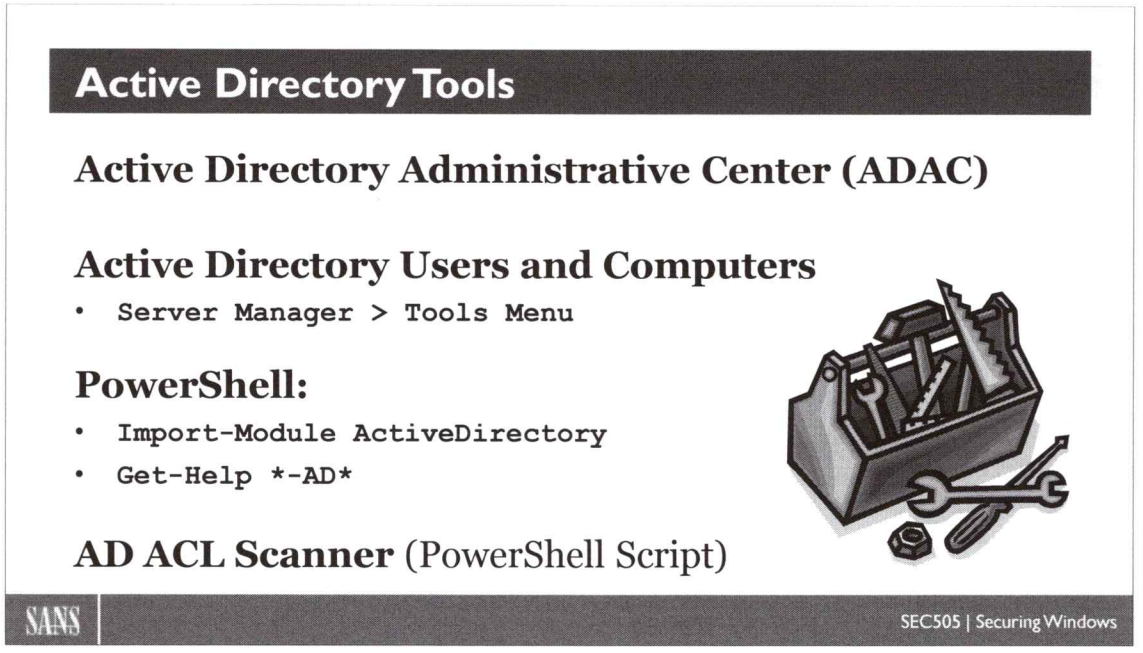


Today's Agenda

Virtually every object in Windows has permissions. An object's set of permissions is its Discretionary Access Control List (DACL), while an object's set of audit settings for the sake of writing to the event logs is that object's System Access Control List (SACL).

We are most familiar with NTFS and share permissions, but there are also permissions on registry keys, processes, threads, tables in SQL Server, mailboxes in Exchange, etc. And there are permissions on the objects in Active Directory.

Active Directory permissions can be used for delegation of IT authority and to limit the harm from compromise. Domain controllers are some of the *highest* high-value targets in our networks, so understanding AD permissions and auditing is very important.



Active Directory Tools

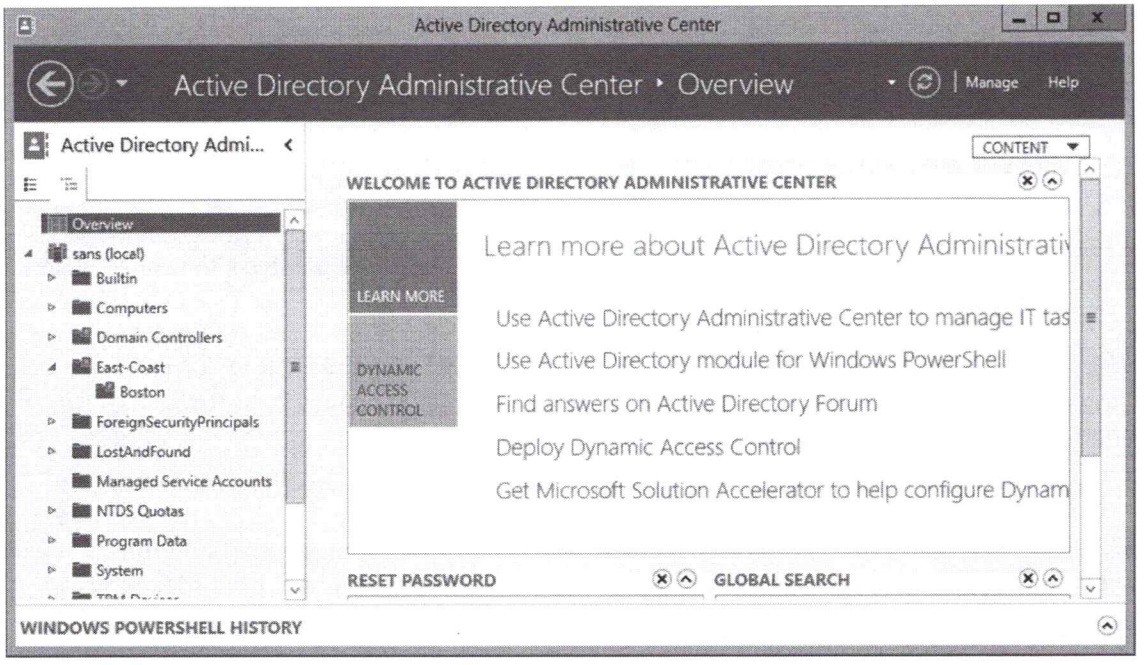
There are a variety of graphical and command-line tools for managing Active Directory. Most of these tools will be installed by default after you run Server Manager to install Active Directory. Others are a part of various Microsoft Resource Kits, downloaded separately from Microsoft's website or obtained from various third-party websites. Hence, just expect that you'll need to build your own personal collection of binaries and scripts to keep in your AD toolbox. If you can't find one of the tools below, just Google the filename.

Microsoft Touch-Oriented AD Tools:

The following are graphical, touch-oriented "modern" management tools:

- Server Manager (in Server 2012 and later)
- Active Directory Administrative Center (in Server 2012 and later)

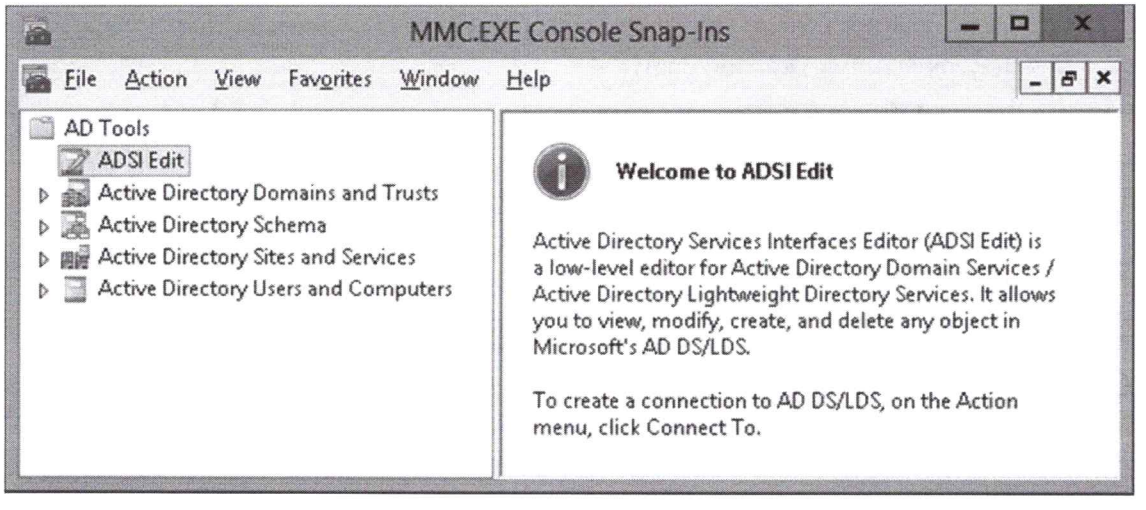
Modern management tools also tend to be GUI wrappers around PowerShell commands and scripts, but this is not required.



Microsoft Desktop AD Tools:

Applications which use the traditional Windows API (WinAPI) are called "desktop" applications. Most desktop GUI management tools are snap-ins for the Microsoft Management Console (MMC.EXE):

- Active Directory Users and Computers
- Active Directory Domains and Trusts
- Active Directory Sites and Services
- Active Directory Schema
- Group Policy Management (GPMC)
- ADSI Edit



If the Schema Manager snap-in is missing when you try to add it, you will need to register its DLL (schmmgmt.dll) with the operating system using REGSVR32.EXE. Follow the instructions in the next *Try It Now!* exercise to do so.

> ***Try It Now!***
>
> In an elevated command shell, execute "`regsvr32.exe schmmgmt.dll`" > click OK in the dialog box that appears. Now execute "`mmc.exe`" > File menu > Add/Remove Snap-In > Add > Active Directory Schema. Add all of the above snap-in tools to your console and save it to your desktop as "AD Tools".

Note too that in Windows Server 2008 and later, you can stop and restart Active Directory Domain Services just like any other service. This is useful, for example, for patching and performing defragmentation of the AD database without the necessity of rebooting into Directory Services Restore Mode (a type of Safe Mode).

LDP.EXE is a query and edit utility for AD which allows explicit control over many LDAP parameters; it can be very useful for troubleshooting and hacking (KB224543).

> **Note:** DCPROMO.EXE was the wizard to convert older servers into domain controllers, but this functionality was moved into Sever Manager in 2012.

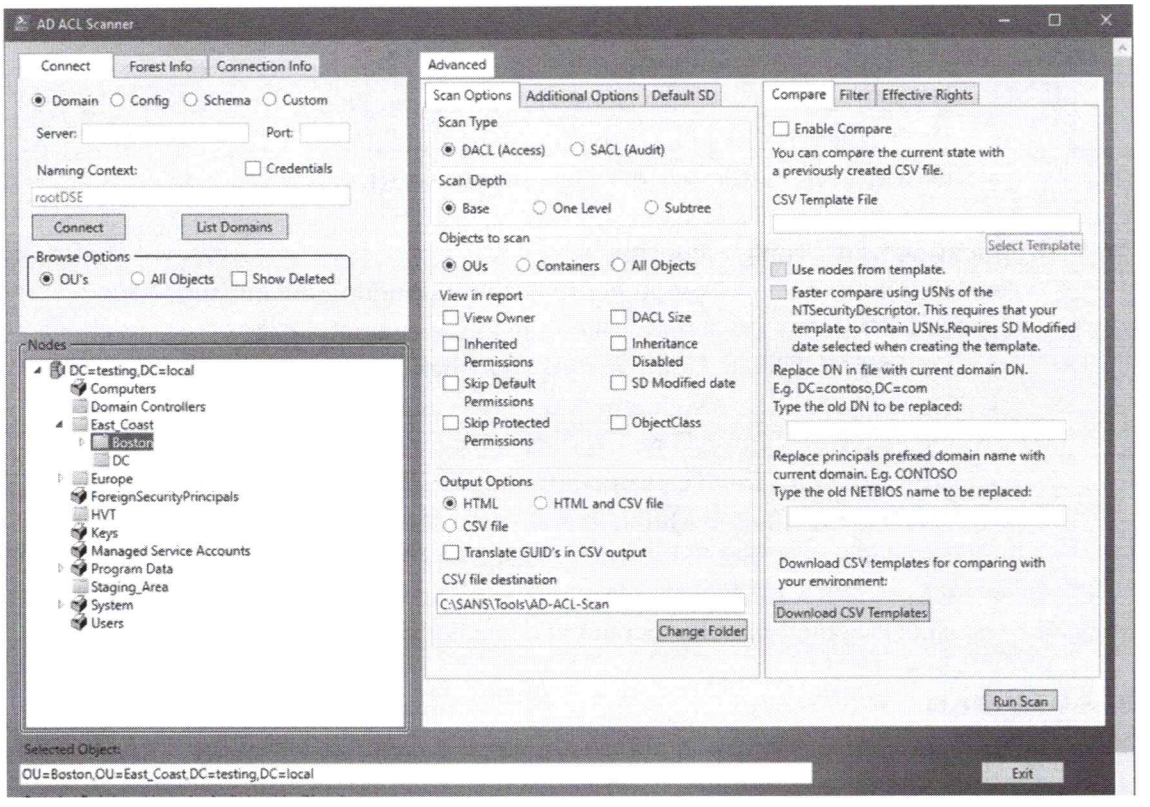
AD PowerShell Cmdlets

Active Directory can be managed entirely with PowerShell 2.0 or later. To see your Active Directory cmdlets, open PowerShell elevated and run:

```
import-module activedirectory

get-help *-ad*
```

PowerShell AD ACL Scanner

A graphical application for viewing, comparing, exporting, auditing and searching AD permissions and audit settings is AD ACL Scanner (https://adaclscan.codeplex.com). This GUI app is written 100% in PowerShell. ACL data can be exported to HTML or CSV files for easy comparison and analysis.



Remote Server Administration Tools (RSAT)

A package of tools can be downloaded from Microsoft for free to be installed on Windows Vista or later for the sake of administering Active Directory and Server 2008 or later (KB941314). This package is called the "Remote Server Administration Tools (RSAT)" and it comes in both 32-bit and 64-bit versions. Installing RSAT is also how to install the Group Policy Management Console (GPMC) on Vista+SP1 or later.

Microsoft's AD Command-Line EXE Tools:

Permissions and Trusts

- ACLDIAG.EXE -- Displays permissions on AD objects.
- DSACLS.EXE -- Manage ACLs on AD objects (good for scripts!).
- DSREVOKE.EXE -- View or delete AD permissions.
- ENUMPROP.EXE -- LDAP and permissions browser.
- SDCHECK.EXE -- Checks the Security Descriptor of AD objects.
- NETDOM.EXE -- Manage trusts and computer domain memberships.
- NLTEST.EXE -- Manage NetLogon service and DCs, updated for AD.

Import/Export and Bulk Management

- CSVDE.EXE -- Import/export AD data to a comma-delimited text file.
- LDIFDE.EXE -- Import/export bulk AD data to a text file (KB237677).
- MOVETREE.EXE -- Move objects between domains.
- DSADD.EXE -- Add objects to AD.

- DSGET.EXE -- Display properties of selected AD objects.
- DSMOD.EXE -- Modify objects in AD.
- DSMOVE.EXE -- Move or rename objects in AD.
- DSQUERY.EXE -- Find types of objects in AD.
- DSRM.EXE -- Remove or delete objects in AD.

Replication and Troubleshooting

- DCDIAG.EXE -- Comprehensive troubleshooting diagnostics.
- DNSCMD.EXE -- Troubleshooting and managing DNS.
- DSASTAT.EXE -- Compares AD data between multiple servers.
- DSMGMT.EXE -- Maintain AD store, manage FSMO servers, clean metadata, perform AD recovery tasks, delegate roles, and more.
- NETDIAG.EXE -- Tests operation of IP, DNS, Trusts, Kerberos, etc.
- NTDSUTIL.EXE -- Maintain AD store, manage FSMO servers, clean metadata, perform AD recovery tasks, and more.
- REPADMIN.EXE -- Monitor and manage replication links and the KCC. This is something like a command-line version of REPLMON.EXE.

DSACLS.EXE

DSACLS.EXE is a Support Tool that is the command-line equivalent of the Security tab on AD objects. An advantage of DSACLS is that it can edit permissions on an item without overwriting the item's other permissions. The utility can also control inheritance and reset permissions back to the defaults defined in the Schema.

Display Permissions, Owner and Audit Settings (/A)

To show the permissions on the Guest account, as well as its owner and audit settings, you would execute the following (without /A, only permissions shown):

```
Dsacls.exe "cn=Guest,cn=Users,dc=domain,dc=domain" /A
```

Restore Default Permissions from Schema (/S and /T)

When a new object is created in AD it can inherit permissions from its container, but that object's initial explicit permissions come from its class definition in the schema. The schema not only defines an object's structure, but also that object's default permissions. DSACLS.EXE can reset an object's explicit permissions back to their "factory defaults" when mistakes have been made customizing its ACL (/S). You can do the same for an entire OU and everything inside it (/S /T).

The ability to restore explicit permissions to their schema defaults is *extremely* useful. For example, if you accidentally mangle the permissions on a few thousand user accounts (or any type of object) and you wanted to fix it before anyone else notices, try this:

1. Create a new, temporary OU.
2. Move the objects into the new OU.
3. Use DSACLS.EXE to restore all those objects to their default permissions.

4. Move the objects back into their old OU.
5. Delete the temporary OU.

If the permissions were mangled on the OU itself, just edit its ACL by hand to match the ACL for the organizationalUnit class. Be wary of using DSACLS.EXE on the old OU itself because you'll likely reset the default permissions on everything *in* the OU too (if you use the /T switch). If that's what you want, fine, but the steps above assume there are *other* things in the OU besides the mangled user accounts.

Another use is when you wish to change the default permissions on all objects of a certain class. You'll modify the ACL on the class in the schema, which will affect all new objects based on it, then you'll use DSACLS.EXE to reset all the existing objects of that type to their (new) schema defaults.

Let's look at some command-line examples. If you wanted to reset the default explicit permissions on a user account named "Bob Terwilliger":

```
dsacls.exe "cn=Bob Terwilliger,ou=Boston,ou=East Coast,dc=usa,dc=sans,dc=org" /S
```

If you wanted to reset the default permissions on *all the objects in* the Boston OU, the command-line would look like this (careful of the /T switch!):

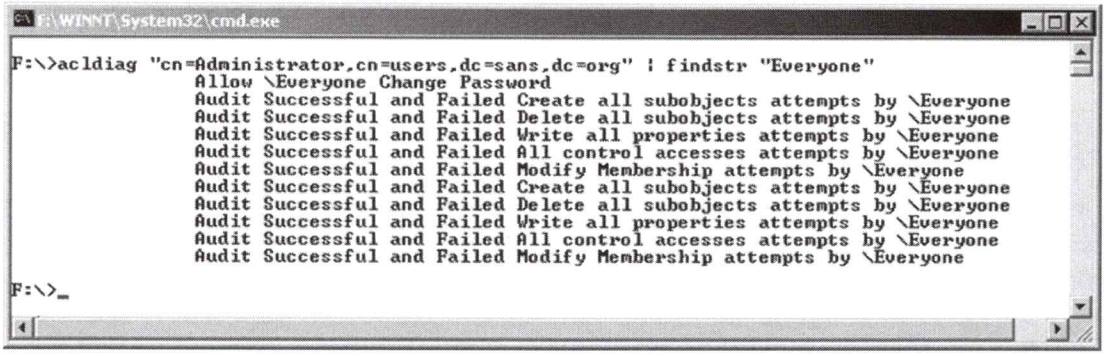
```
dsacls.exe "ou=Boston,ou=East Coast,dc=usa,dc=sans,dc=org" /S /T
```

ACLDIAG.EXE

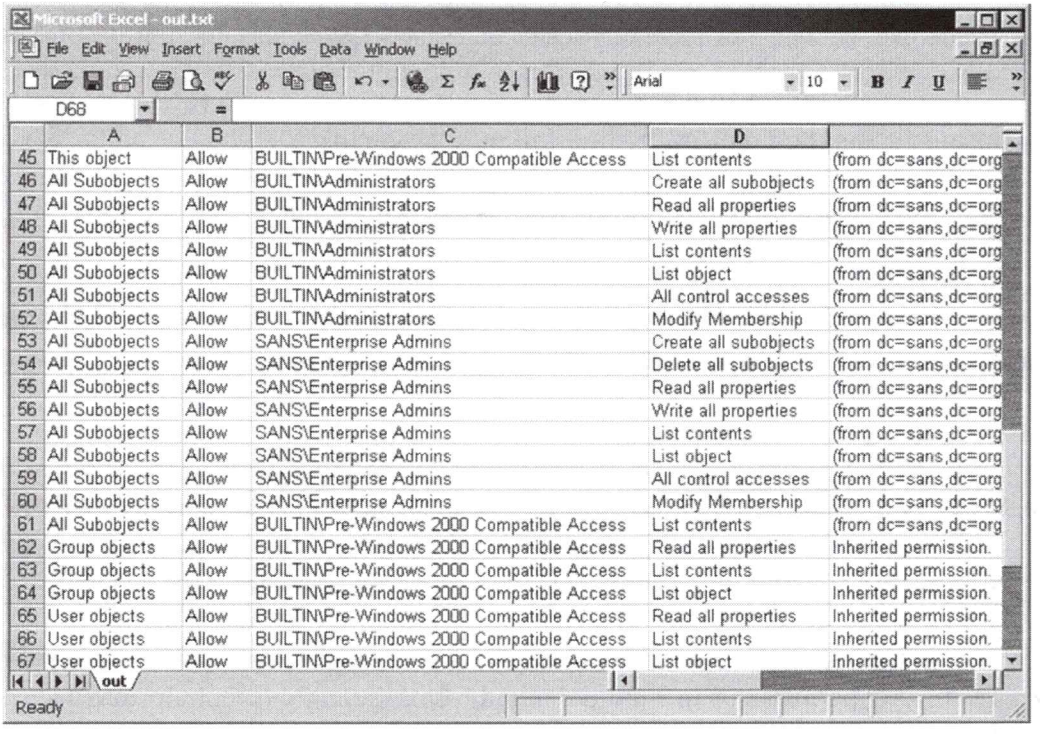
ACLDIAG.EXE is a Support Tool from the Windows Server CD-ROM for viewing AD permissions and audit settings. It cannot be used to access remote systems.

ACLDIAG.EXE can do the following:

- Display the final, cumulative, effective permissions and audit settings a user or group has to an item in AD in a user-friendly format.
- Compare AD permissions against the default permissions from the schema.
- Fix permissions set with the Delegation of Control Wizard using a built-in template (this amounts to simply reapplying the template from the command line).
- Dump AD permissions in a tab-delimited format suitable for import into a spreadsheet or database.



The following screenshot shows a tab-delimited ACL produced by ACLDIAG.EXE imported into a spreadsheet for analysis. This capability is very important for conducting automated audits of AD access control lists. A script could dump all ACLs into a database, compare them against a similar dump from three months ago, then list all the differences. Having such a comparison baseline is also useful for troubleshooting.



SDCHECK.EXE

SDCHECK.EXE, also one of the Support Tools, displays ACL information in a very detailed format. It cannot be used to change permissions or audit settings. A useful aspect of the tool is that it shows both inherited and explicit permissions, and gives the parent container from which an inherited permission was inherited.

DSREVOKE.EXE

DSREVOKE is a separate download from Microsoft. It recursively displays or deletes AD permissions for a specified user or group at your specified OU and its sub-OUs. However, the tool cannot remove permissions set on non-OU containers or other non-OU objects unless those permissions were inherited from a parent OU. The tool also cannot remove permissions set within the Schema or Configuration naming contexts.

Microsoft's Migration Tools:

- ADPREP.EXE -- This tool must be run on the Schema FSMO master (/forestprep) and all Infrastructure FSMO masters (/domainprep) in all domains prior to upgrading AD. The tool is found in the \sources\adprep folder of the Windows Server installation DVD.

- Active Directory Migration Tool (ADMT) -- The ADMT is the primary tool for migrating your current Windows domain(s) to a more recent version. This is also the best tool for consolidating multiple domains into one. Make sure to use the latest version, it is updated periodically.

- CLONEPRINCIPAL Scripts -- These VBScript scripts use COM+ components to create mirror user, group and computer accounts in another domain (in another forest) which possess the same SIDs as the originals; this is useful for creating a parallel forest while preserving the original upgraded forest as fall-back position; the scripts and DLLs include Clonepr.dll, Clone-gg.vbs, Clone-ggu.vbs, Clone-lg.vbs, Clone-pr.vbs, Sidhist.vbs, ADsSecurity.dll and ADsError.dll.

JoeWare Tools (www.JoeWare.net)

A popular website for Windows tools is http://www.joeware.net/freetools/, including a collection of free tools for Active Directory. The tools are regularly updated and the documentation is nice. These tools can be run from within either CMD.EXE or PowerShell. Some of the more interesting tools include:

- ADFIND.EXE -- Better than Microsoft's own DSGET or DSQUERY, it's a flexible search tool that can also show deleted items and server-side query performance statistics. Very nice for automating security audits.

- ADMOD.EXE -- Better than Microsoft's own DSMOD or DSRM, a tool to modify, create, delete *or undelete* objects in AD.

- ATSN.EXE -- Matches IP addresses to their AD sites, or lack thereof.

- OLDCMP.EXE -- A safe way to list or delete old computer or user accounts.

- SECDATA.EXE -- Dump the important properties of user and computer objects to a properly-formatted CSV file which can be easily parsed or imported to Excel.

- UNLOCK.EXE -- Flexible way of displaying or unlocking locked accounts.

ADRESTORE.EXE

It's been acquired by Microsoft now, but one of the many mind-bogglingly cool tools from Sysinternals is the free ADRESTORE.EXE utility for restoring deleted AD objects in Windows Server 2003 and later. Microsoft even has a KB article on how to use it: KB840001. For the time being at least, you can still get the Sysinternals tools from http://www.microsoft.com/sysinternals/.

ADSI Scripting Support

Active Directory can be managed entirely through scripts. The scripts can be written in VBScript, PowerShell, JScript, Perl, or any other language, as long as the script can use the ADSI interface.

The Active Directory Services Interface (ADSI) exposes an easy-to-use interface for scripting the management of AD. Despite its name, ADSI is a generic interface for any vendor's directory services, Microsoft's or otherwise.

PowerShell also provides access to the *System.DirectoryServices* classes in the .NET Framework for managing Active Directory. There are also free AD cmdlets from third-party companies, such as Quest Software (now Dell), which make the use of these classes easier. PowerShell is available in Windows Server 2008 and later by default, but it can be installed on Windows XP/2003/Vista too (www.microsoft.com/powershell/).

What Is LDAP?

The main protocol used to query and edit AD is the Lightweight Directory Access Protocol (see RFC 2251 and KB221606 for more information about LDAP). LDAP is an industry-standard protocol for accessing many types of directory databases besides AD. LDAP is built into Internet Explorer, My Network Places "Entire Directory", AD management snap-ins, etc. Microsoft also has a generic LDAP client named LDP.EXE.

LDAP servers listen on ports TCP 389 and 636 by default. The Global Catalog service listening on ports TCP 3268 and 3269 is actually an LDAP server. LDAP over SSL on TCP ports 636 and 3269 requires the installation of a digital certificate on the domain controller (KB247078).

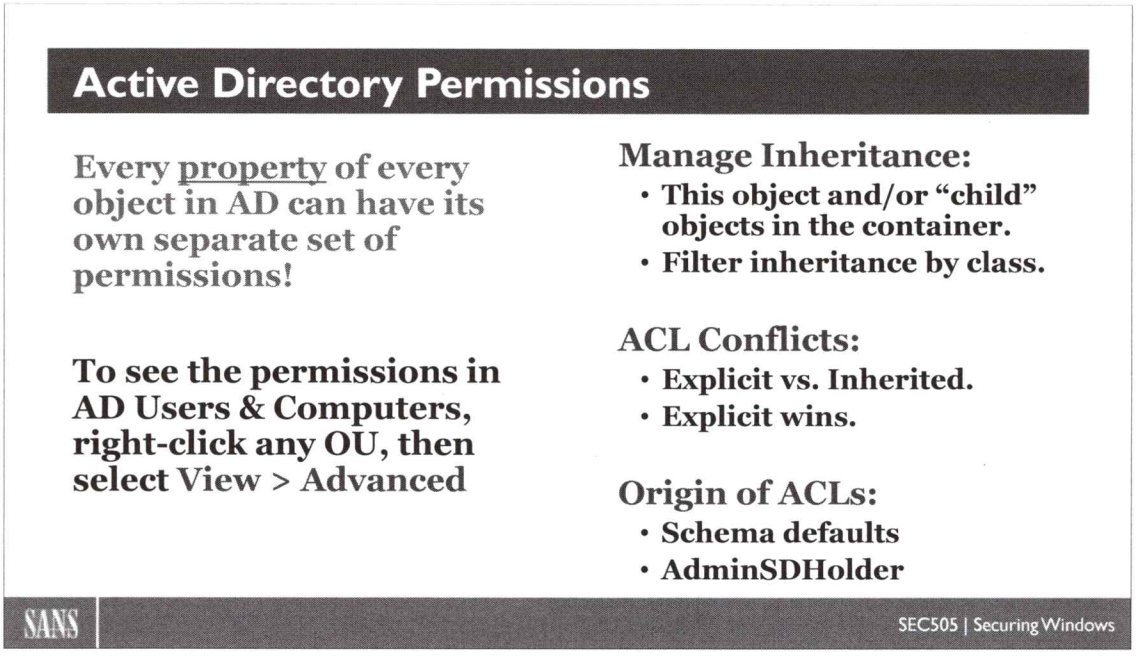
Secure LDAP Administration Traffic

Windows Server 2003 and later signs and encrypts its LDAP management traffic by default, and virtually all the AD administration tools that come with Windows 2003 and later will use secure LDAP when connecting to other domain controllers, even if those controllers are running Windows 2000. However, for Windows 2000 domain controllers to support secure LDAP they must have Service Pack 3 or later installed (KB325465). The administration tools from the Windows Server 2003 or later DVD (installed with ADMINPAK.MSI or the Remote Server Administration Tools (RSAT)) should be installed on any machine from which you plan to manage Active Directory. This will significantly increase the integrity and privacy of your administration work.

Essential Port Numbers

There are ports to definitely block at the perimeter firewall, including LDAP.

Summary of Windows Port Numbers to Block at the Firewall		
Port Number	**Protocol**	**Description**
3268	TCP	Global Catalog with LDAP
3269	TCP	Global Catalog with LDAP and SSL encryption
544	TCP	Kerberos KSHELL
464	TCP and UDP	Kerberos Passwords
88	TCP and UDP	Kerberos Secure Authentication
636	TCP	LDAP SSL
389	TCP and UDP	Lightweight Directory Access Protocol (LDAP)
137	UDP	NetBIOS query requests
138	UDP	NetBIOS query responses
139	TCP	NetBIOS Session (for SMB or CIFS)
135	TCP	RPC mapper
445	TCP	SMB without NetBIOS (CIFS)
3389	TCP	Remote Desktop Services
42	TCP	WINS replication



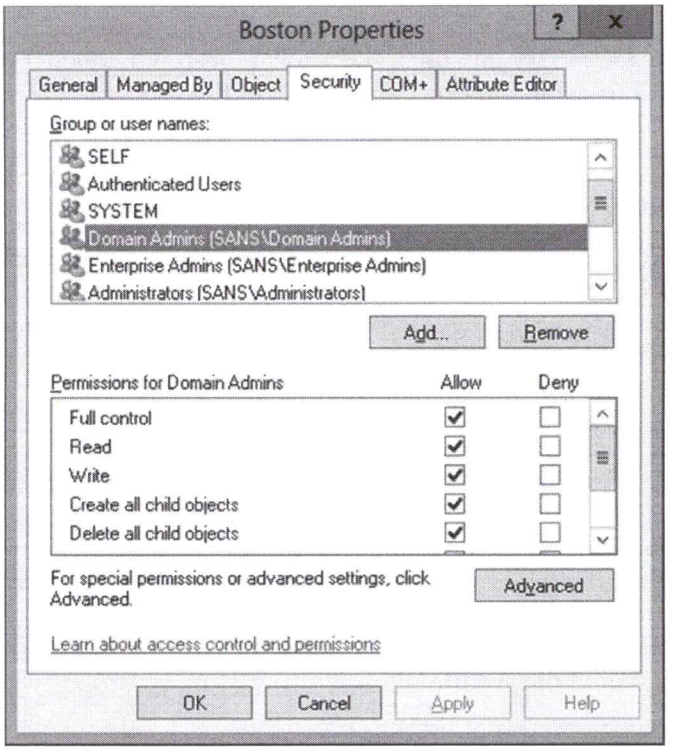
Active Directory Permissions

Every *property* of every object in the Active Directory database can have *its own separate set of permissions*! In addition, there are object permissions, container permissions, control of inherited permissions from containers, access auditing, and more. These features make it possible to delegate authority and to regulate AD access very precisely.

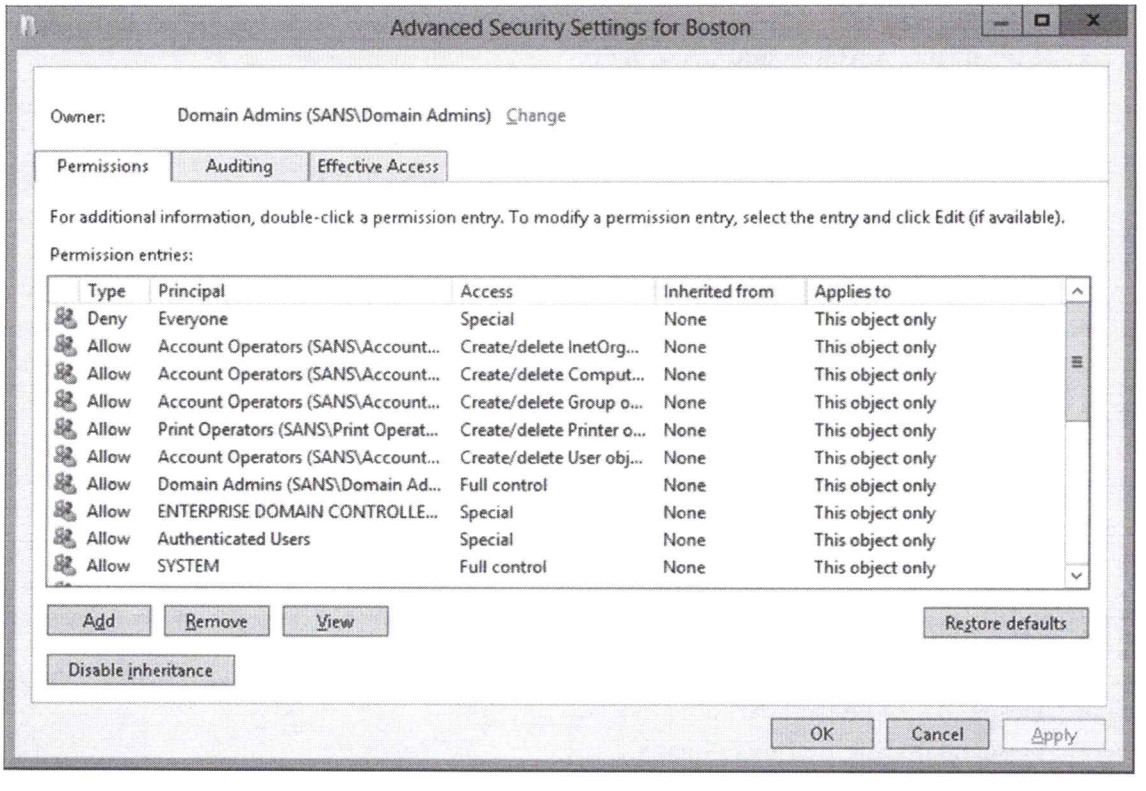
To See The Security Tab: Right-Click > View > Advanced Features

In order to see the Security tab on AD objects, you must enable the viewing of Advanced Features. To do this, open Server Manager > Tools menu > Active Directory Users and Computers > right-click on any container > View > Advanced Features.

To see the permissions and audit settings on a container or object, right-click the item > Properties > Security tab. The following screenshot shows the Security tab.



Clicking on the Advanced button will show the full set of permissions and audit settings for the item, as well as its owner. Only a summary of these settings is shown on the prior Security tab. The following screenshot shows the Advanced property sheet.



If you select a permission entry and click the View button, you open a property sheet defining the permission. Notice there are two tabs in the property sheet: one for permissions for the object as a whole, another for the permissions for the properties of that object (see below). The Name at the top shows for which user/group the permission applies; click Change to select another.

Inheritance of DACL and SACL ACEs

Access control entries (ACEs) can be inherited from parent containers. This is true for both discretionary access control lists (DACLs) and system access control lists (SACLs), i.e., it is true for both permissions and audit settings.

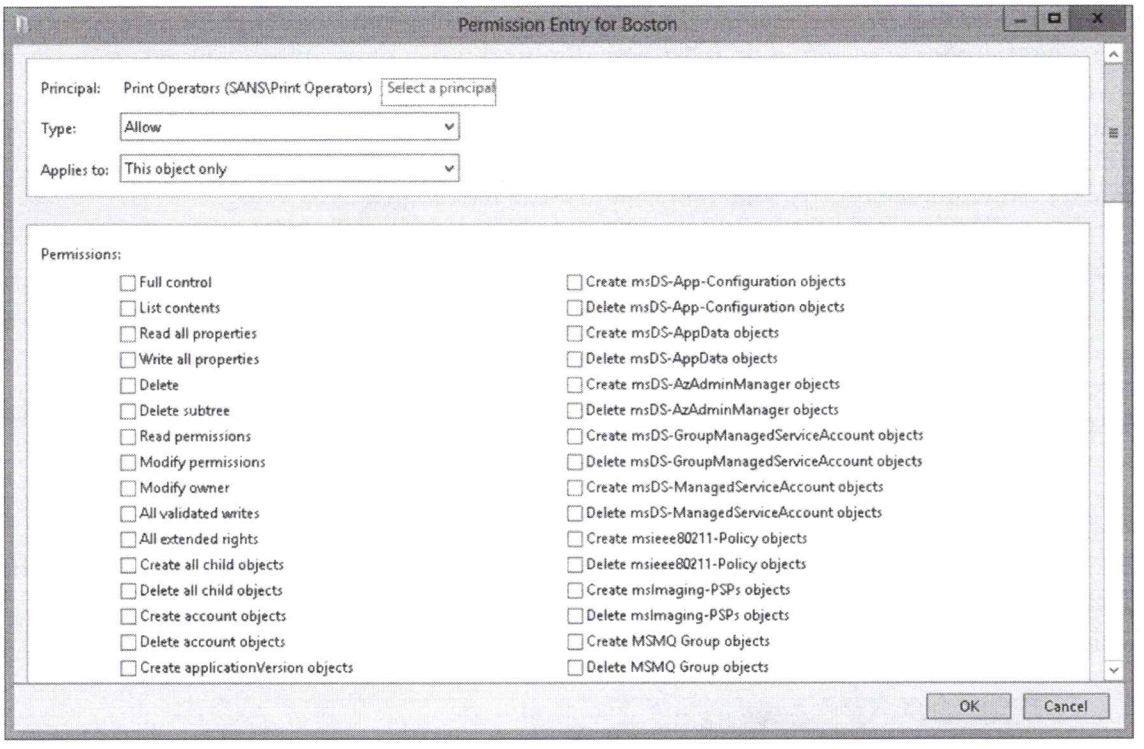
An inherited permission or audit setting is displayed with a slightly grayed-out key icon or a checkbox with a gray background.

A property of the object/container as a whole is the "protection flag", shown as a checkbox labeled "Allow inheritable permissions from parent to propagate to this object". Note that you can disable/enable ACE inheritance for both DACLs and SACLs separately on the Permissions tab and the Auditing tab of the Advanced property sheet.

When you select View/Edit on an ACE, the Apply Onto pull-down menu controls the propagation of the permission to subobjects and subcontainers which are inheriting (KB178170). The Apply Onto setting can be configured separately for the object ACL and the properties ACL. The items on the Apply Onto list include items such as:

- **This object only**.
- **This object and descendant objects**.
- **Descendant objects only**.
- Descendant aCSResourceLimits objects.
- Descendant certificationAuthority objects.
- Descendant Computer objects.
- Descendant Connection objects.
- Descendant Contact objects.
- Descendant Group objects.
- Descendant groupPolicyContainer objects.
- Descendant IntelliMirror Group objects.
- Descendant IntelliMirror Service objects.
- Descendant MSMQ Configuration objects.
- Descendant Organizational Unit objects.
- Descendant Printer objects.
- Descendant Shared Folder objects.
- Descendant Site objects.
- Descendant Site Link objects.
- Descendant Site Link Bridge objects.
- Descendant Site Settings objects.

- Descendant Site Container objects.
- Descendant Subnet objects.
- Descendant Subnets Container objects.
- Descendant Trusted Domain objects.
- Descendant User objects.



Notice the checkbox at the very bottom labeled "Apply these permissions to objects and/or containers within this container only". This box is grayed out if you select Apply Onto This Object Only. But if these permissions are to be inherited by descendant objects in this container, this checkbox controls whether the permissions are inherited by descendant objects in the subcontainers as well (and the sub-sub-containers, and the sub-sub-sub-containers, and so on). This checkbox is equivalent to the "propagate permissions down only one level deep" found in other ACL editing tools.

Conflicts Between Inherited and Explicit Permissions

Note that a permission set explicitly on an object/container for a particular user/group will override a conflicting permission on that object/container inherited from higher in AD. Permissions assigned to a user/group are inherited, unless that user/group has been specifically assigned a different permission on the item inheriting. Explicit permissions always override inherited permissions; they are not merged or compared (KB233419).

For example, if user Bob has No Access on an OU, and a printer object in that OU is inheriting, but the printer has been explicitly assigned Full Control to Bob, then the inherited permission for Bob is ignored and Bob will indeed have Full Control of the

printer. It is not that Bob's Full Control permission somehow wins out over the No Access, rather, the No Access permission is simply not processed for Bob. The same would have been true if Bob had Write as the inherited permission and Read as the explicit one: Bob's final permission would be Read, not Write.

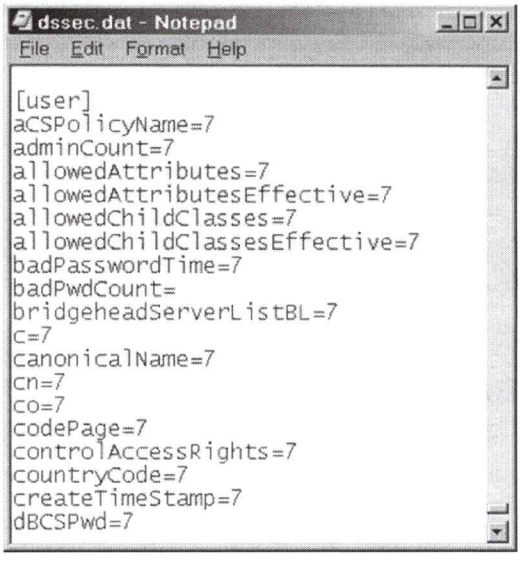
Object Ownership

An object's default owner is either Domain Admins or the Administrators group on the domain controller, depending on what type of object it is. This can be changed on the Owner tab. The owner of an object, just as with NTFS, can change the permissions on that object at will and possesses the permissions assigned to CREATOR OWNER as well. Users do not own their own user accounts by default. Properties do not have owners, only objects like user accounts and OUs have owners.

DSSEC.DAT

While every AD property can have its own access control list, not every property is visible in the GUI interface when assigning permissions. Many objects have over 100 properties, so it would be cumbersome to show them all in the graphical ACL editors. Also, many of these properties are only modified by AD system processes and administrators should not normally tamper with them.

Exactly which properties are visible to snap-ins is determined by a text file named DSSEC.DAT located in \%SystemRoot%\System32\. This can be edited with Notepad to reveal any property of any object in snap-ins so that permissions can be assigned to it with those GUI tools. The alternative is to use a command-line ACL editor.



Each type of object (like "[User]") is in square brackets; the properties of that object type are listed below it. Simply change the "7" after the property name to zero or blank (see "badPwdCount=" in the screenshot above) then close and re-open your snap-in, such as the "AD Users and Computers" snap-in. You will see that property in the GUI ACL editor. If a property is not listed, you can add it manually. The change takes effect

immediately, but you still must close and reopen the tool. Strictly speaking, 7 means that the property is not shown in the GUI on the computer running the snap-in, 6 means that the "Read" property will be shown, 5 means that the "Write" property will be shown, and 0 or blank means both properties will be shown.

> **Note:** Each new permission adds about 70 bytes to the AD database (KB197054).

Where Do The Default Permissions Come From?

The ACL on an object is determined by four factors:

- The default permissions for that object's class in the Schema.
- The permissions explicitly assigned to that object manually.
- The permissions that object has inherited from its parent container(s).
- The permissions on the AdminSDHolder container.

Every 60 minutes the PDC Emulator will compare the permissions on the users in the Administrators, Domain Admins, Schema Admins and Enterprise Admins groups against the permissions on the \System\AdminSDHolder container in AD. If they differ, the ACLs on the users will be changed to match the ACL on the AdminSDHolder container (KB232199, KB318180). Be aware, though, that once a user has been added to one of these privileged groups, then, even if that user is later removed from these groups, the user's permissions can still get overwritten automatically (see KB817433 concerning the adminCount attribute).

Whatever permissions and audit settings are set in a class in the Schema are the default explicit permissions set on any new object created based on that class. Hence, if you change the permissions in a Schema class, then create a new object based on that class, that new object will have the custom permissions you added to the class. However, any objects of that same type which already existed before you modify the ACL will not have their permissions dynamically reset to match the new ACL on that class. Note that it is not the permissions on the class object itself which are copied, but the permissions defined in the defaultSecurityDescriptor property of the class which are copied to the new instance of the class. Search for "defaultSecurityDescriptor" on Microsoft's website for more information about the syntax of the SDDL strings found there (KB297947).

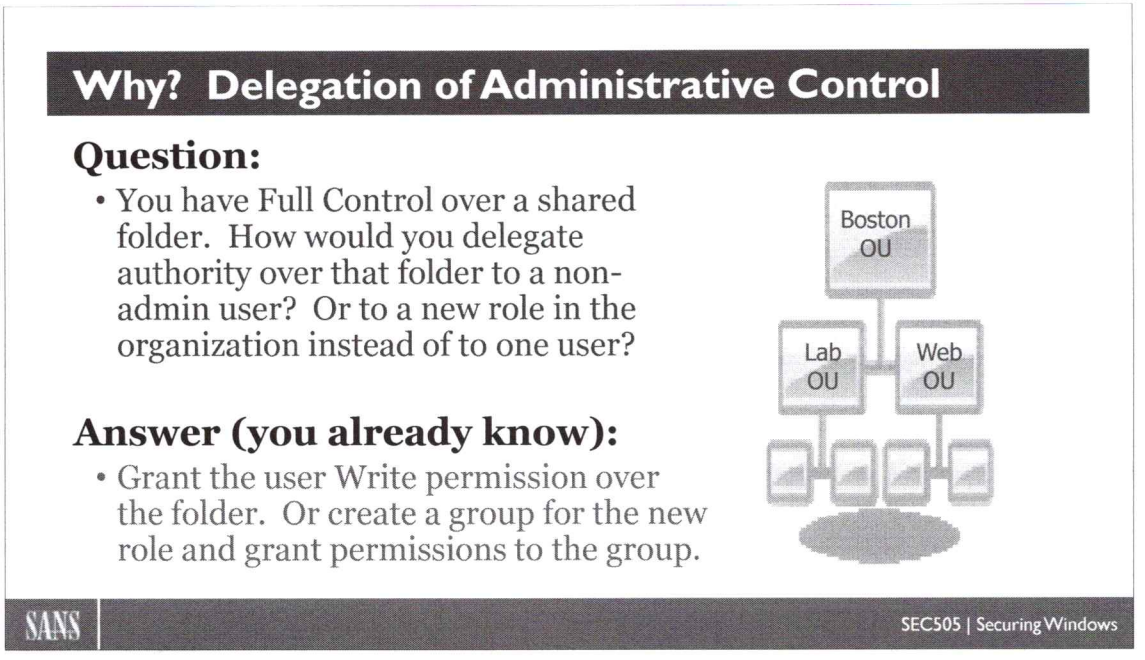
> **Warning!** Modifications to the Schema can have far-reaching and unintended consequences. Test all proposed changes in a lab first. Fortunately, you can always go back and re-edit the ACL on a class, so this change is more forgiving than other Schema modifications, but note that this still will not dynamically change the ACLs of the objects which had been created based on it.

As an example, you might create a new class called "sensitiveUser" based on the "user" class. This new class would be no different, except that its default DACL would be more strict, and its SACL (for auditing) would audit all failed access and all successful changes. Sensitive user accounts --like those of managers, OU administrators, HR

personnel, etc.-- would be created based on the sensitiveUser class instead of the default user class. A custom MMC console with a script would be used to create the accounts.

Another example would be to modify the ACL on the user class so that users could not modify their own personal information, such as phone numbers (KB292304). But any user account which had already been created would have the old default permissions.

Fortunately, you can use DSACLS.EXE to search-and-reset default Schema permissions on objects in selected domains and OUs.



Why? Delegation of Administrative Control

In NT 4.0, it was difficult to give a user any administrative powers without making him or her a full administrator for the entire domain. In Active Directory, on the other hand, it is possible to delegate administrative power very precisely. This is true even in a single-domain environment.

Here are a few examples of how power can be delegated in AD:

- A non-administrative user could have full control over all user and computer accounts in an OU, but no other OUs. "Non-administrative" means that that user is not a member of Enterprise Admins, Domain Admins or the local Administrators group on any domain controller.

- A help desk global group could be given the power to reset anyone's password in the domain except for the sensitive user accounts, such as those of administrators and service accounts.

- Alternatively, each OU could have its own separate help desk group, and each group could only reset passwords for users in their assigned OU.

- A non-administrative user or group could be given the authority to join a single particular computer to the domain or have the blanket authority to add any number of computers to the domain. This can be restricted on a per-OU basis too.

- The receptionists global group could be given the power to change a single property of all user accounts (e.g., home phone number) but not be able to change anything else.

- A global group could be given the Backup Folders and Files user right on all computers in an OU, but not have this right anywhere else. This group would not necessarily have to have the Restore Folders and Files right either.

Analogy: How Would You Delegate Authority Over A Shared Folder?

Consider, if you were a Domain Admin and you wanted to give a regular user the ability to manage the contents of a shared folder, how would you do it? Simple! You simply give that user Full Control over the NTFS folder and the share. Now that non-administrative user can add, delete, rename, move, and copy files in the share. All other users can only read the contents of the folder.

Instead of a shared folder, apply the same thought process to an OU. In this case, it is not files in a folder but user and computer accounts in an OU that the non-administrative person will have power over. The only thing that's strange here is that a user account is both an object in AD (hence, it has permissions) and something that another person uses to log onto the domain.

And because each property of a user account can have its own separate ACL, you can delegate authority over each of these properties.

In abstract, then, "delegation" has three parts:

1. The *object* --user, group, computer, shared folder, OU, domain, whatever-- that someone will be given full or limited power over (power is always power "over" *something*). This object must have permissions or some kind of access control mechanism that can be modified on a per user or per group basis.

2. The person who already has the power to change the permissions on that object. This is almost always an administrator, but includes anyone with the equivalent of the Change Permissions permission on the object. This person is the one who "grants" authority to another over the object.

3. The person who "receives" the authority. This is the person who will have additional permissions on the thing over which they will have authority. The exercise of this authority is made possible through these permissions. People without the authority lack the permissions on the object to modify it anyway.

Delegation assumes that the person who grants the authority 1) has the legal/political right to do so, 2) the person receiving the authority has the legal/political right to exercise it through powers over the delegated object, 3) there is a mechanism in place to prevent others from exercising those same powers when they have not been given these legal/political rights. These "legal/political rights" are relative to your organization (e.g.,

commercial corporation, military base, government office) and may be enforced through threat of employment termination, fines, three years in Leavenworth prison, etc. Hence, delegation of authority in AD is more than just modification of AD permissions, but it is these permissions we are concerned with here.

Best Practices for Delegating Authority

The following is a list of best practices for delegation of authority, with examples to follow in the next few slides:

- Write a policy document which describes exactly how you want to delegate authority, the names of the groups to which authority will be granted, the exact permissions these groups will have, and keep a written log of all changes. AD permissions can become confusing very quickly, hence, write a plan first, stick to it, and document when you deviate from it.

- Focus delegation on Organizational Units. Try to delegate authority as high as possible in the OU structure. Design your OUs around how you plan to delegate authority, then create sub-OUs as necessary for Group Policy needs.

- Try to avoid delegating authority over sub-OUs in a way which contradicts the authority assigned at the parent level. In short, avoid creating "delegation orphans" in an OU structure where descendant OUs are not under the control of those groups which control its parent(s).

- In general, it is better to create new OUs and assign custom permissions to them than to modify the permissions on the built-in OUs or the domain container itself. If there is ever a problem with the ACLs you configure on your custom OU, you can usually move its contents to another OU or reset its ACLs to their schema defaults without worrying too much about crashing AD. When modifying permissions on built-in OUs or at the domain level, see if you can achieve your ends by adding new Allow permissions instead of adding Deny permissions or changing the factory-default access control entries from Microsoft.

- Assign permissions to groups when you delegate, not individual users, and grant the least power necessary to let delegates get their work done, but no more. This is just best practice for assigning any kind of permission.

- Don't forget that AD supports auditing as well as permissions. Diligent auditing will not prevent misuse, but it can detect it, limit its further action, and document what damage has been done so that it may be repaired. In a politically-charged organization, auditing allows peaceful coexistence through the policy of "trust but verify". (Auditing is discussed in an up-coming section.)

- Use Group Policy to restrict access to powerful snap-ins. For example, create a domain-wide Policy that blocks access to all administrative snap-ins to all users

except members of the various OU and Domain Admins groups. But don't forget that other tools are easily available which are not snap-ins.

- Create custom MMC consoles for delegates that only show the OUs or objects over which they have control, and add "Tasks" to the console to help them use their authority safely. This should also help reduce their support phone calls to you.

- Be cognizant of the political ripples your delegation choices will make. Political battles among IT staff are made *more* intense by Active Directory, not less.

- Consider investigating third-party tools for managing AD permissions if you'd rather not edit the raw AD permissions.

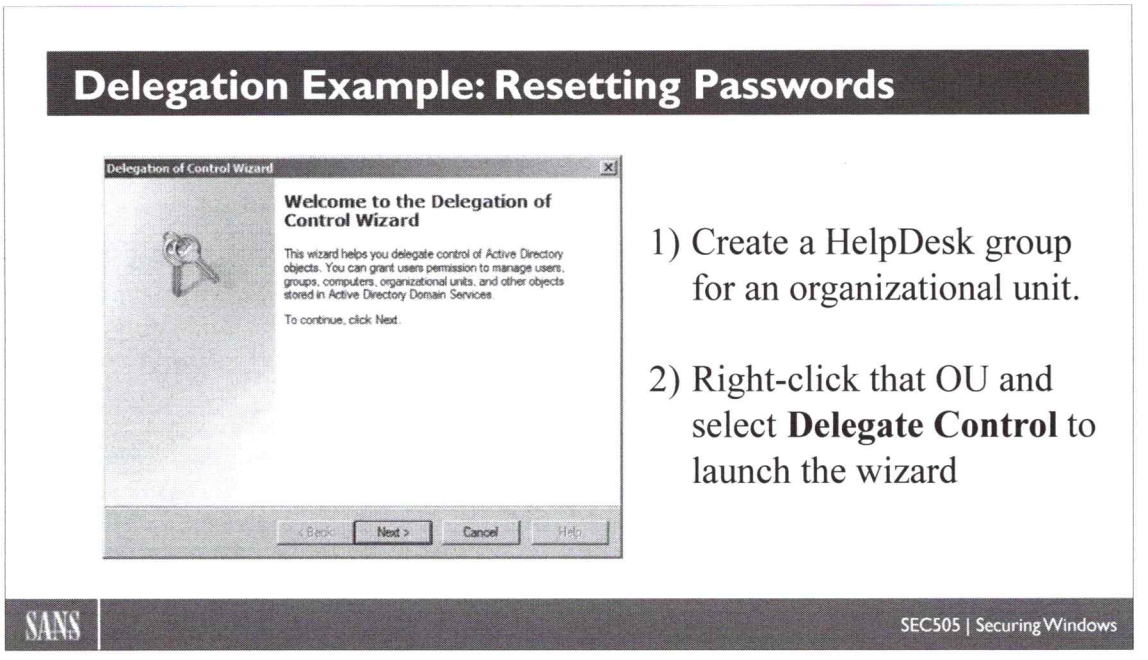
Third-Party Tools
Vendors have jumped on the delegation bandwagon by providing tools that simplify and automate the delegation process:

- Quest Software through Dell (http://software.dell.com/platforms/active-directory/)
- FastLane ActiveRoles (http://www.quest.com/fastlane/activeroles/)
- NetIQ Directory Security Administrator (http://www.netiq.com/products/dsa/)

Example: FastLane ActiveRoles
FastLane ActiveRoles (http://www.quest.com/fastlane/activeroles/) is a commercial product specifically designed to help manage delegation of authority. It uses the native permissions in AD for delegation, so you are not forever bound to their product and you don't have to deploy special FastLane applications to those who will be granted authority.

Create custom "roles" or use the built-in categories in the product, then assign users or groups to these roles. Next, choose the OU or domain where you want that role to have special authority. Role definitions can be downloaded from their website or you can define custom ones yourself.

Importantly, you can produce detailed reports on sets of AD permissions, and you can use the verify feature to check that the roles set by the product are still in effect. There are also web-based tools for help desk staff. The ActiveRoles tool itself is an MMC snap-in.



Example: Resetting Passwords

To assist in the delegation of administrative powers, Windows Server includes a Delegation of Control Wizard. It can be found on the context menus of containers in the "AD Users and Computers" and "AD Sites and Services" snap-ins.

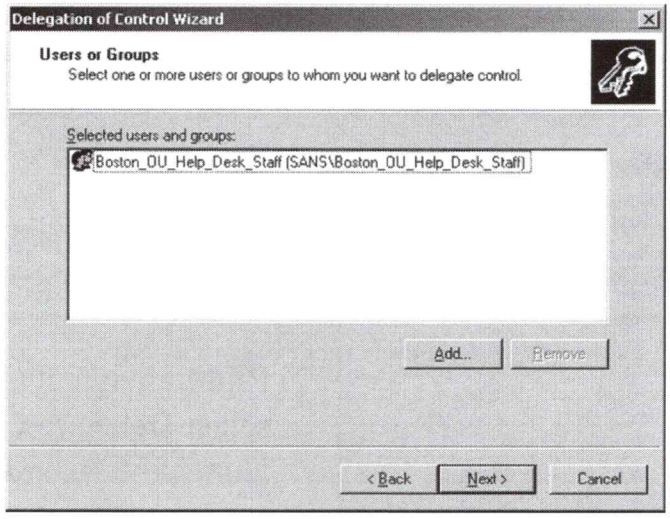
The Wizard can simplify the process of delegating many common tasks, such as resetting passwords and the ability to add users to groups. The end result of using the Wizard is a new set of permissions on the relevant AD items.

Let's use the Delegation of Control Wizard to give a global group the ability to reset passwords on any user account in a single OU. Create an Organizational Unit named "Boston" and a security global group named "Boston_Help_Desk".

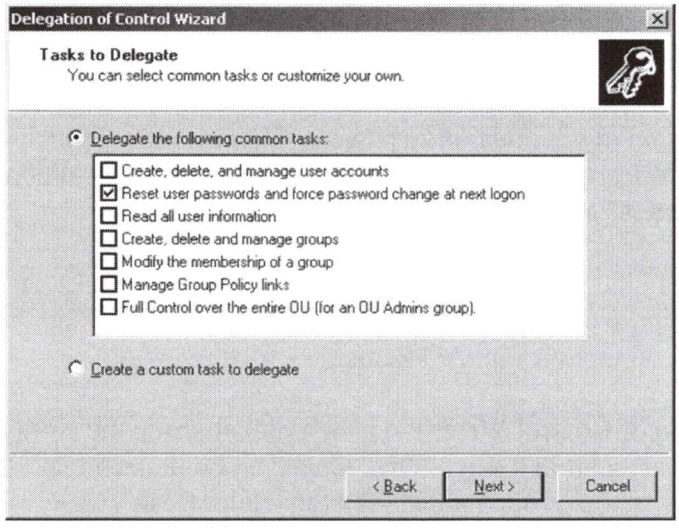
Note, too, that the checkboxes which appear in the Delegation Wizard can be edited to include any custom task you wish. The DELEGWIZ.INF file determines what checkboxes appear in the Wizard and what changes they make (KB308404). The format of the file is not too obscure, and, once made, the file can be shared with other administrators to simplify their work.

Try It Now!

To delegate password-reset authority over the Boston OU > right-click the Boston OU > Delegate Control > Next > Add > select the group you wish to grant authority to, e.g., Boston_OU_Help_Desk_Staff > Add > OK > Next.



Check the box labeled "Reset password" > Next > Finish. That's it! (Note: the checkboxes in the screenshot below will look different from yours because the picture came from a machine with a custom DELEGWIZ.INF file.



Limitations of the Delegation of Control Wizard

The Delegation Wizard tool suffers from some important limitations:

- It only appends additional permissions to an item's ACL. It cannot remove prior existing permissions or de-delegate authority already granted (see DSREVOKE.EXE).
- It cannot be used to specify Deny permissions to limit authority.
- It cannot configure audit settings.
- It cannot delegate control over an individual object separately from its OU.

- It cannot be used on the Builtin container (or the LostAndFound container).

- And, if there are problems during the delegation process, the Wizard will not help you to repair faulty permissions you have already set or otherwise assist in troubleshooting, e.g., it cannot compare permissions on an object against that object's default permissions from the schema.

However, all of these limitations can be overcome when you manage the raw AD permissions yourself. Let's run through a variety of delegation tasks together.

Don't Use The Wizard, Just Edit AD Permissions Directly

IT staff are not the only ones who will need to edit Active Directory. Because AD is intended to be a general-purpose directory database, many groups within your organization may need to manage their portion of it. For example, if AD were used as the Human Resources database, then HR personnel would need to be able to edit and have exclusive access to such data as salaries, disciplinary histories, 401K plan information, Social Security numbers, health insurance plan beneficiaries, etc., all of which would be stored in AD as properties of user accounts.

Example: Editing Selected Properties

You can delegate read/write access over the individual properties of objects and have these permissions be inherited from OU containers. This gives us very fine-grained control over these properties and who can read or modify them.

This is important for a variety purposes, but it's especially important for Dynamic Access Control (DAC) in Windows Server 2012 and later. We normally manage permissions and audit settings through a user's group memberships, but with DAC it is possible to use the attributes of user and computer accounts in AD for making authorization and logging decisions, such as for Data Loss Prevention (DLP). For example, a DAC policy might allow a user to access files in a shared folder only if 1) the user's two-letter country code in Active Directory is set to "US" or "CA" for United States or Canada, 2) the user's department attribute in AD is set to "Human Resources", and 3) the user is sitting at a computer whose operating system version is 6.2 or later. Because some of the AD attributes of users and computers can be used for access control decisions, delegation of control over these selected attributes is important too.

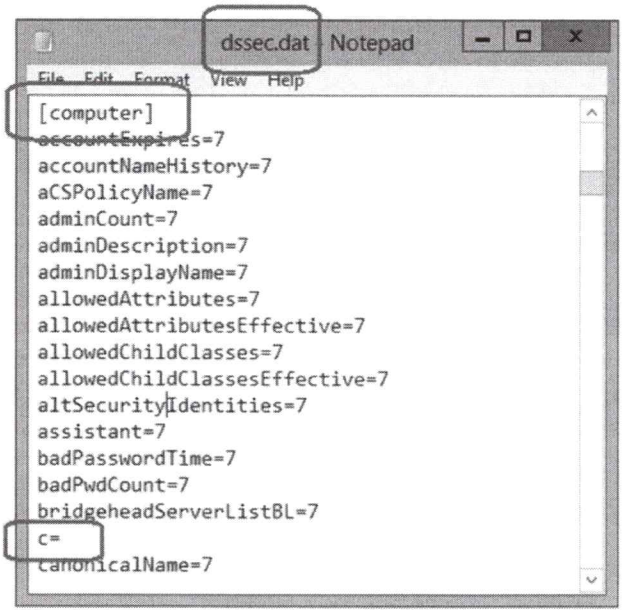
As an example, let's delegate control over the department attribute and the two-letter country/region abbreviation attribute (internally named "c") for both users and computers in a domain. We'll imagine that these are attributes used for the sake of Dynamic Access Control policies.

> **Note:** If you ran the SEC505 setup script for this course, the Human_Resources group should have been created for you already.

Create a global security group named "Human_Resources", if it doesn't exist, then give that group write permission on the attributes named "Department" and "Country/Region

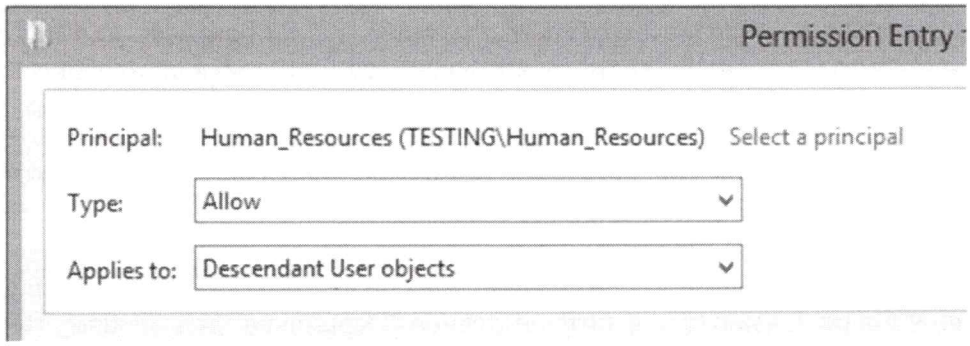
Abbreviation" at the domain level to be inherited only by descendent User objects and descendent Computer objects. These permissions will be inherited throughout the domain. Alternatively, you could assign these permissions on just a selected OU, but for this example it will be for the entire domain.

But wait! The "Country/Region Abbreviation" attribute is not visible by default (nor is "c"). Microsoft hides this attribute in the graphical interface when editing permissions. The internal name of that attribute is just the letter "c", so we will need to edit the %SystemRoot%\System32\dssec.dat file, as discussed above, in order to show this attribute when using the graphical AD management tools.



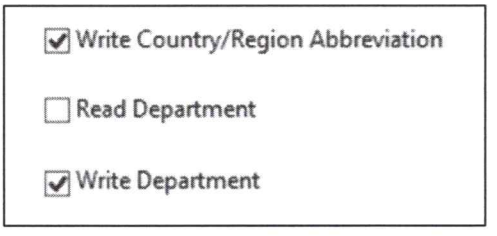
Try It Now!

In PowerShell, run "notepad.exe $env:systemroot\system32\dssec.dat" to open the dssec.dat file in Notepad. Find the section named "[computer]", then delete the number 7 on the line which reads "c=7" so that the line only has "c=" afterwards. Then find the section named "[user]" and make the same change to "c=". Close Notepad, save changes, then close any graphical tools you have open for editing AD permissions. When you open a graphical tool for editing AD permissions again, you'll see that the "c" attribute is displayed as "Country/Region Abbreviation" in the graphical interface.



Now we are ready to modify the attribute permissions. The permissions you will be granting to the Human_Resources group are the following:

- Location: the domain container at the very top
- Principal: Human_Resources
- Type: Allow
- Applies To: Descendent Computer objects (and for Descendent User too)
- Write Department
- Write Country/Region Abbreviation (for User objects only)
- Write c (for Computer objects only)



Try It Now!

To grant write permission to the Human_Resources group, right-click on your domain name in "AD Users and Computers" > Properties > Security > Advanced > Add > select Human_Resources as the principal > select "Descendent Computer objects" from the list of Applies To options > scroll all the way down to the bottom and click the "Clear All" button > check the boxes next to "Write department" and "Write c" in the properties list > OK > Apply. Now, repeat the exact same procedure, but for "Descendent User objects" instead, but notice that instead of showing the true internal name of the "c" attribute, the graphical interface shows the "Country/Region Abbreviation", so grant Write access to that display name instead.

Why does Microsoft hide some attributes so that we have to edit the dssec.dat file? Why are some attributes shown with alternative display names instead of their true internal names? Microsoft is trying to be helpful, but sometimes the help can get in the way, so we have to know how to work around the "help" to accomplish what we need.

Example: Prevent Edits To Users and Computers In One OU Only

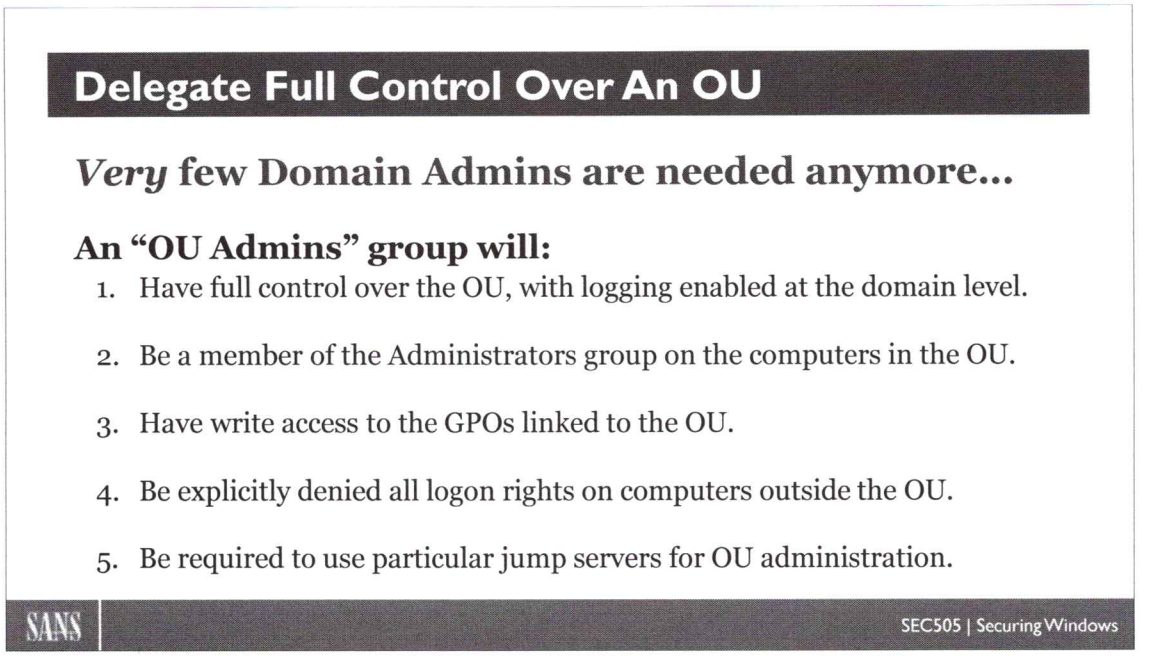
Imagine that we do not want the Human_Resources group to change any department or country attributes on any users or computers in a particular OU even though they should be allowed to make such changes anywhere else in the domain. We could prevent the OU from inheriting any permissions at all, but this is overkill. Instead, we will simply deny write access on the permissions above to Human_Resources on the OU. These deny permissions will be inherited by the user and computer objects in the OU, hence, the Human_Resources group will have both Allow:Write and Deny:Write permissions, but because neither permission has been explicitly assigned, the deny permission will override the allow permission.

Next, we will use the same process as before, except that these permissions will be set to Deny and we will be doing it on an OU instead of at the domain level.

Try It Now!

To prevent the Human_Resources group from modifying any users or computers in an OU, right-click on that OU > Properties > Security tab > Advanced > Add > select Human_Resources as the principal > set Type to Deny > select "Descendent Computer objects" from the Applies To list > click the "Clear All" button at the bottom of the properties list > check the boxes for "Write department" and "Write c" > OK > Apply. Then do the same for "Descendent User objects" and their "Write Department" and "Write Country/Region Abbreviation".

Keep in mind, too, that these permissions will always be enforced against Human_Resources. It doesn't matter whether they use a script, snap-in, touch-oriented tablet app, or third-party tool.



Delegate Full Control Over An OU

It is possible to create an "OU Admins" global group and give it administrative power over all the users and computers in an OU. This group can create, delete, and modify all the user accounts, computer accounts, and groups in their OU. This OU Admins global group can also be automatically added to the local Administrators group on every computer in that OU through Group Policy, just as the Domain Admins group was automatically added.

What an OU Admin loses is the power to change domain-wide settings, such as password/lockout policies, Kerberos policies, and trust links. But that doesn't mean they can't politically have a say in how these things are determined (once again, the authority vs. power distinction) and it's not like these settings are reconfigured on a daily basis anyway. Politically, there should be a committee which jointly agrees on domain-wide settings, then the CTO/CISO will actually authorize and make the change.

Delegation Steps: Giving Power To An OU Admins Group

To delegate authority over an OU, perform the following as a Domain Admin:

1. Create a new global security group that will be granted power over the OU. Place that group in the OU it will control, or, if the Domain Admins wish to reserve control over the OU administrators, then the group could be created in a separate OU over which only the Domain Admins have control. For example, if the name of the OU is "Boston", then create a "Boston_OU_Admins" group in the Boston OU or in another OU over which the Domain Admins retain control.

2. Give the OU Admins group Full Control to the OU and all descendants.

3. Create a GPO, link it to the target OU, and grant Full Control over the GPO to the OU Admins group for that OU.

4. Use the GPO to add the OU Admins global group to the local Administrators group on all the servers and workstations in the target OU.

Issues

When creating an OU Admins group and delegating authority to it, there are a few issues to keep in mind during your planning:

- Where will the OU Admins group itself be located in AD? If you want the current members of that group to have control over it, then place the group in the OU over which the group itself has authority. If you want centralized control over its membership, then place the group in a different OU controlled by the central IT staff. Keep in mind, however, that if OU Admins control Group Policy in their OU, they can add whoever they like to the local Administrators group on computers in their OU. Hence, it will likely be the case that either 1) the OU Admins group controls both their Group Policy and its own group membership, or 2) the OU Admins group will control neither their Group Policy nor their own membership.

- The same question is valid for the user accounts in the OU Admins group. Where are they in AD? Who has control over them? If the OU Admins group will be controlled by a central IT authority, then perhaps central IT would like to have exclusive control over the user accounts in it too.

- Do you want to allow the OU Admins group to manage the Group Policy Objects linked to its OU? Group Policy is somewhat complex, so not all OU Admins groups will be able to manage it. Because of the importance of Group Policy for security, moreover, central IT might want to retain control over Group Policy itself. This will become a sore political topic, so watch out.

Delegate IT Authority With OUs

IT administrative authority should be delegated at the OU level whenever possible. Don't just dump every IT staff member into the Domain Admins group. If an OU Admin gets infected with malware, compromised by hackers, or deliberately/accidentally harms the organization, the scope of the damage will hopefully be limited to just the OU.

Have Separate Organizational Units For High-Value Targets

The importance of high-value accounts and groups warrants the creation of one or more special OUs just for them. Separate OUs assist in the correct application of Group Policy security options and AD permissions. Because there are relatively few high-value target users or groups, having separate OUs will hopefully simplify the auditing of what's really important without adding too many new OUs.

These special organizational unit(s) should be given innocuous names and perhaps buried a few layers deep in the AD hierarchy. Do not name the OU "High-Value-Targets."

Note that some built-in groups in AD, such as the global Administrators group (not the local one), cannot be moved to another OU.

Delegate Authority To New "OU Admins" Global Groups

The Domain Admins group is added to the local Administrators group automatically when a computer is joined to the domain. This is one reason why the Domain Admins group is so powerful.

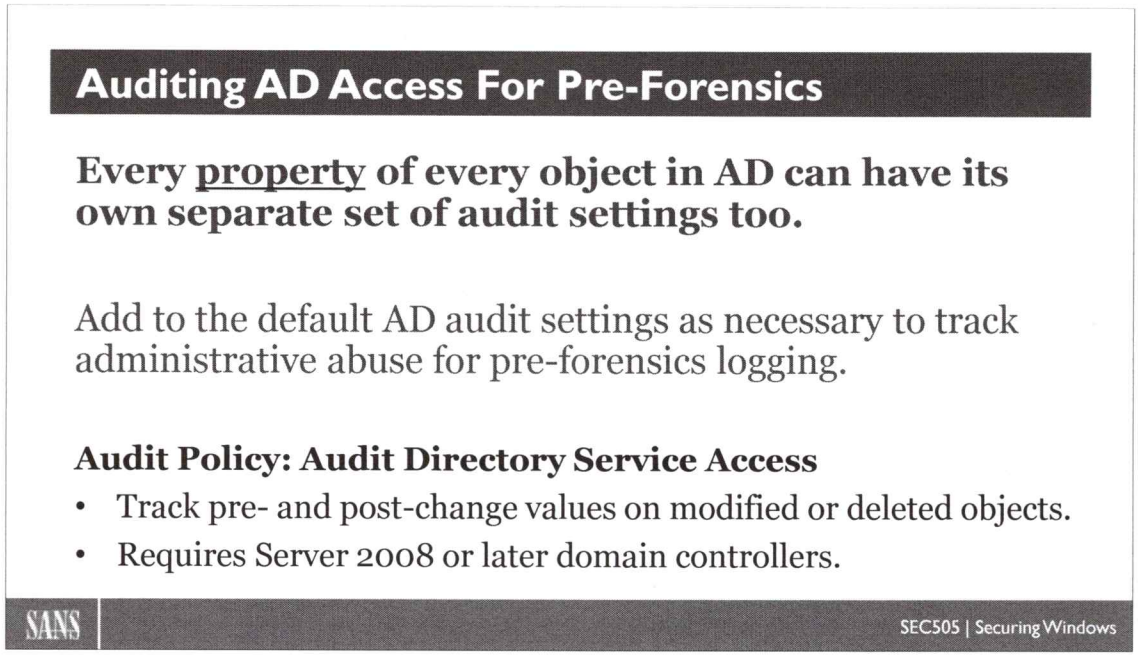
But if you create your own custom "OU Admins" groups for each major OU over which you wish to delegate authority, you can add the appropriate OU Admins group to the local Administrators group on each computer in that OU. Hence, on those computers in the OU, both the Domain Admins and OU Admins groups will have full administrative power. An OU Admins group can also be granted control over the GPOs linked to their OU.

If you are an OU Admin, you have all the power you need to install software, fix problems, do backups and otherwise manage their machines in your OU. What power would you lack to get your work done? You are just as powerful as a Domain Admin, but your power is limited in scope to the OUs you manage.

Later, if an OU Admin account is compromised, the scope of the damage will hopefully be limited to just that one OU. This is the main security benefit. OUs are like pressure doors in a submarine which contain the harm of a leak in a compartment to that one compartment. When hackers or malware steals the password hash or Security Access Token of an OU Admin account, the harm will hopefully be contained to just that one OU. Outside of that OU, the administrator is just a regular user because his or her account is 1) not a member of any administrative groups on machines outside of the local OU and 2) has not been granted any special permissions in Active Directory or anywhere else outside of the local OU. Remember, there is no cure-all patch for lost administrative passwords, lost service account or scheduled job passwords, token abuse or pass-the-hash attacks, the best we can do is damage containment.

Audit All Access To The Target Organizational Units

The OUs for the high-value targets should have extensive auditing enabled to keep track of nearly all successful/failed access to them. Because there are relatively few such targets, this will hopefully not generate an excessive amount of event log data, but what is AD auditing for if for not tracking changes to high-value targets? If all the accounts are in special OUs, then configuring the audit settings should be easier and more consistent.



Auditing AD Access for Pre-Forensics

A phenomenal amount of data can be logged when auditing access to AD. The same level of detail and flexibility available when configuring AD permissions is also available for auditing. Every property of every object could be separately audited.

Audit data appear as items in the Event Viewer Security log on each DC separately, hence, custom scripts or add-on tools are required to remotely extract, centrally consolidate, and intelligently analyze this log data.

AD auditing must be enabled before any data appear in Event Viewer. It is disabled by default prior to Windows Server 2003 R2. After it is enabled, the system access control lists (SACLs) on objects and containers in AD can be customized to fine-tune exactly which actions will be logged.

How To Enable AD SACL Auditing

AD auditing on domain controllers is enabled through Group Policy. Keep in mind that this is auditing as determined by the System Access Control Lists (SACLs) on the individual AD objects and containers configured for auditing, it is not a general auditing policy which is simply switched on, such as logon/logoff auditing.

> ***Try It Now!***
>
> To enable AD SACL auditing, open the "Group Policy Management Console" MMC snap-in > right-click on the Default Domain Controllers Policy > Edit > Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Audit Policy > double-click "Audit directory service access" > check all three boxes, including Success and Failure > OK > close Group Policy Management Editor window.

Customizing AD Object and Container SACLs

Unlike NTFS SACLs, which must all be configured by hand, Active Directory installs with an extensive set of default SACLs. The default audit settings for an object are --just like default permissions-- defined by the schema and the container into which the object is placed. Access to the Schema itself by the Everyone group is almost 100% audited.

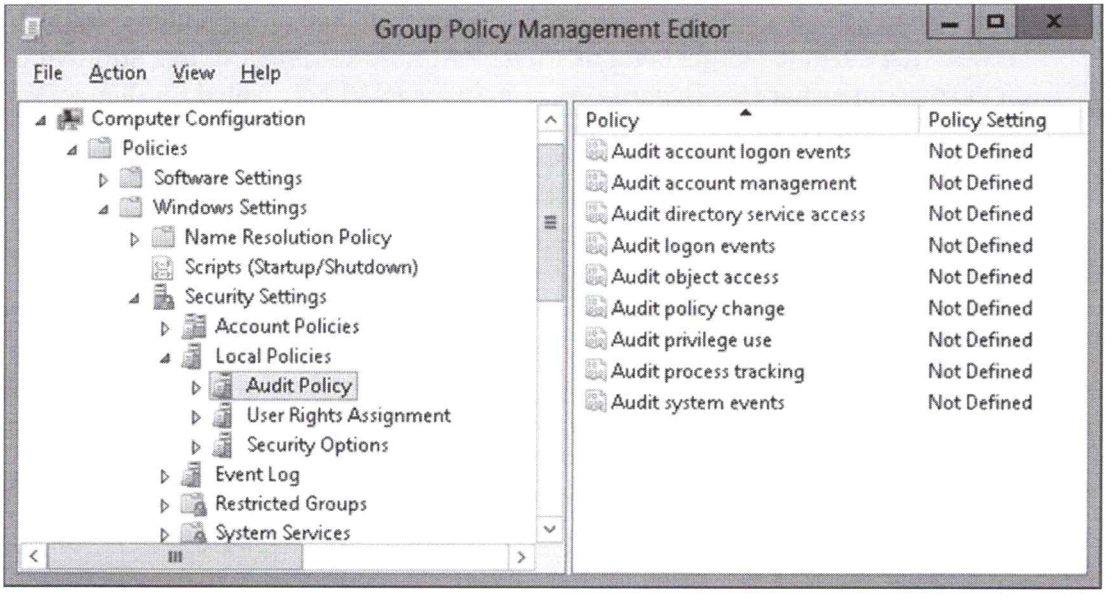
For most object types the default SACL tracks all modifications to the object, but no read/list access to it. Hence, simply enabling "Audit Directory Service Access" will immediately cause audit data to be generated for almost all AD changes without the administrator being required to configure any AD SACLs by hand first.

However, to modify an AD SACL, simply edit the ACL just as you would for AD DACL permissions. Don't forget to "View > Advanced Settings" to see the Security tab!

> ***Try It Now!***
>
> To modify the audit settings on an AD object, right-click the object > Properties > Security tab > Advanced button > Auditing tab > select Add, Remove or View/Edit as needed.

How To Enable Generic AD Auditing

However, AD access can be *indirectly* audited through other audit policies found in a Group Policy Object. If direct auditing occurs in virtue of SACLs on particular AD objects, indirect auditing occurs when certain types of actions occur on the domain controller, such as adding a user to a group or changing password policy. These actions result in AD modifications, hence, they also fall under the rubric of AD auditing.



Here are the different audit policies directly relevant to AD auditing:

- **Audit Account Logon Events** (Success, Failure)-- This tracks authentication requests processed by the domain controllers even when the access is not to the domain controller itself. These authentication requests are sent by users' machines when a user logs on locally at those desktops, and by servers when a client logs on remotely to those servers, e.g., when a user maps a drive letter to a server's shared folder. Think of DCs as providing a service for the sake of other machines on the network (checking usernames and passwords) and this category logs whenever that service is provided. When this policy is enabled on the workstations and servers themselves, then it applies only to the local accounts on those machines, hence, it only applies to authenticated access to those machines. Strictly speaking, this audit policy logs information at the machine where the account in use is physically stored, i.e., either in the global AD database on a DC or in the local SAM database on a member server.

- **Audit Account Management** (Success, Failure)-- This monitors user and group tasks such as account creation, deletion, modification, and group membership changes. Note that only the bare fact that an account or group has been modified will be logged through this policy, not the detailed data made available with "Audit Directory Service Access".

- **Audit Directory Service Access** (Success, Failure)-- This is required to begin logging access to AD objects as defined on those objects' individual SACLs. By analogy, NTFS SACLs also do not cause data to be written to the Event Log unless the "Audit Object Access" policy is enabled.

- **Audit Logon Events** (Success, Failure)-- This tracks interactive and over-the-network logons to the target computer itself. Strictly speaking, it logs information on the machine where the Security Access Token (SAT) is created for the local or remote user that is performing the logon, but the SIDs for that SAT do not all have to come from the target computer itself: the global SIDs will come from a DC and the local SIDs will come from the target machine, but the target machine is still the location where the SAT is created.

- **Audit Object Access** (Success, Failure)-- This is required to begin logging access to NTFS folders and files, registry keys, and shared printers. It is not the case that enabling this category will cause all filesystem, registry and printer access to be logged. Rather, enabling the category makes it possible to have the audit ACLs (the System Access Control Lists) on those objects not do nothing. It takes two changes to audit access to a file, for example; this category must be enabled and that particular file must be configured to audit some kind of access to it.

- **Audit Policy Change** (Success, Failure)-- Tracks changes to the audit policies themselves, and changes to privilege assignments.

- **Audit Privilege Use** (Not Defined)-- Monitors the exercise of the various privileges on the machine, e.g., take ownership, change system time, etc. Enable this policy on an as-needed basis only to avoid filling the logs.

- **Audit Process Tracking** (Not Defined)-- This is rarely enabled, and usually only by programmers who are debugging their own code. This category tracks program execution, process loading and unloading, filesystem handle creation and release, indirect object access, and other low-level OS behaviors. Enabling this category will cause a vast amount of extra log data and will slow the system down considerably.

- **Audit System Events** (Success, Failure)-- Tracks system startup, shutdown, and other system-wide events. This also records clearing of the System and Security logs.

Directory Service Change Auditing

Prior to Windows Server 2008, auditing in AD could tell you who made a change to which attribute, but the event logs wouldn't tell you what the attribute was both before and after the change, hence, you knew which attribute was changed, you knew what the new (current) value is, but you didn't know what the prior setting was. These leaves an important gap in our auditing and forensics work, and makes it more difficult to reverse-out malicious or accidental changes to the directory.

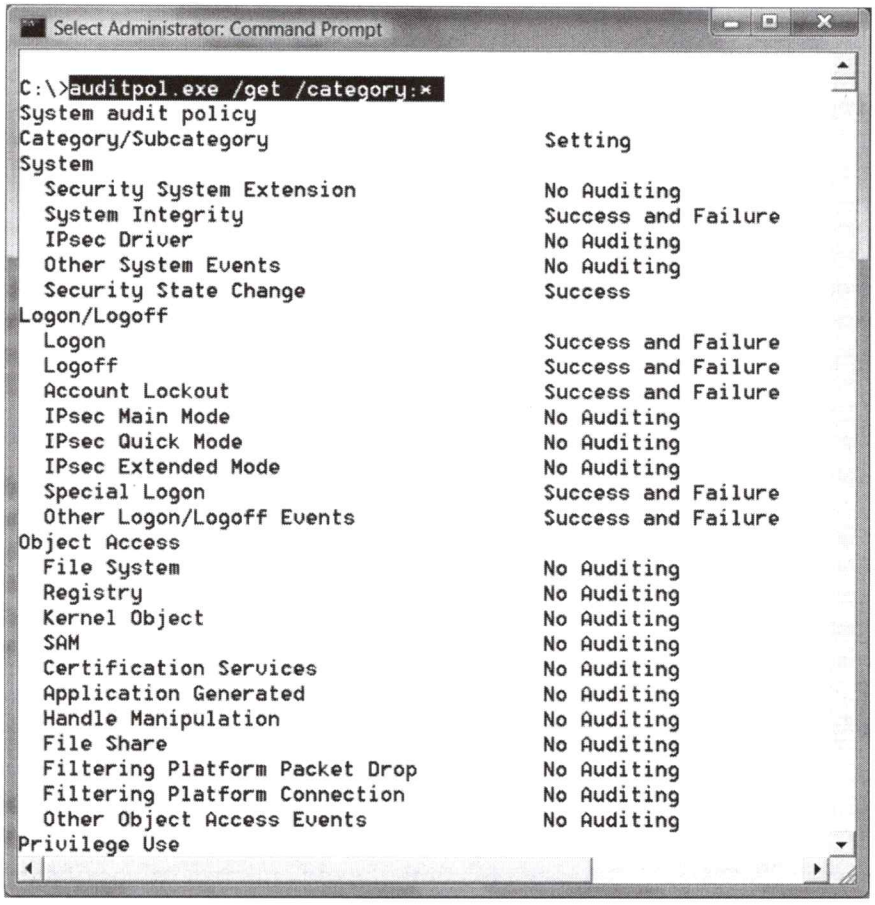
Starting with Server 2008, though, you can log both the old and new values of a changed attribute, as well as track more precisely move, create and undelete operations. These changes appear in the Security event log with ID numbers 5136 (modify), 5137 (create), 5138 (undelete) and 5139 (move).

In a security template or Group Policy, enabling the "Audit Directory Service Access" audit policy category will enable change auditing on Windows Server 2008 and later.

AUDITPOL.EXE

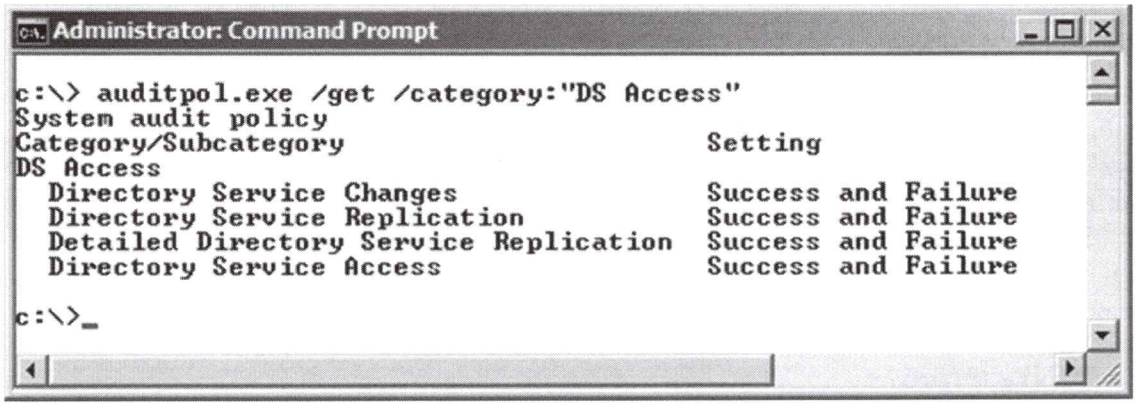
Starting with Windows Vista, the various audit categories have been broken into subcategories which can be enabled/disabled independently of each other. You cannot manage these subcategories individually through Group Policy except by pushing out a script which runs AUDITPOL.EXE. This binary is the only way in both Vista and Server 2008 to view and manage these subcategories separately from each other. However, when you enable the "Audit Directory Service Access" audit category in Server 2008 and later, all the subcategories under it are enabled by default too.

> ***Try It Now!***
>
> To see your current audit policies, including the subcategories, open a command shell with elevated privileges and run: auditpol.exe /get /category:*



On a Server 2008 and later domain controller, to see just the AD-related audit subcategories, run the following command:

```
auditpol.exe /get /category:"DS Access"
```



The audit subcategory which captures pre- and post-change attribute values is named "Directory Service Changes". This, and all the other subcategories, are enabled when the "DS Access" parent category is enabled. In a security template or Group Policy, this parent audit category is named "Audit Directory Service Access."

Auditing Best Practices

The following is a list of best practices for auditing AD:

- To save disk space and optimize performance, only collect the audit data which will be useful to you. At a minimum, audit the Account Logon Events, Account Management, and Policy Change categories (successful and failed). In medium- to high-security environments, also audit the Directory Service Access category (successful and failed). With Windows Serve 2008 and later, if you need to keep the Security log small, you can selectively disable the "Directory Service Changes" audit subcategory with AUDITPOL.EXE or a GPO, but keep this enabled otherwise.

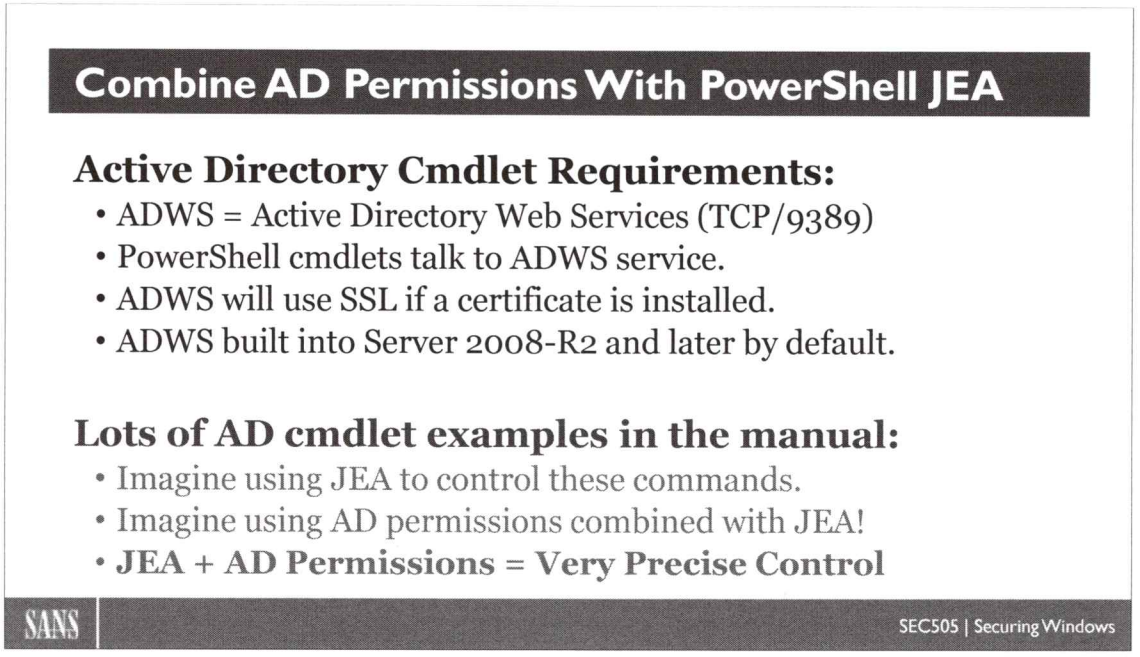
- On controllers prior to Server 2008, increase the size of the security log to no larger than 300MB; this low limit is due to a known bug which can cause events to be dropped if the log grows larger than 300MB (KB183097). On Windows Server 2008 and later, there is no such artificial limit, so change the size of the log to 2GB or larger.

- Set the retention method on the security log to "Overwrite events by days", and only overwrite events after they have been saved twice during normal backup. For example, if you make a full back up every seven days, set the security log to only overwrite events older than 15 days. If you back up the logs each night, then only overwrite events older than three days.

- If you enable "Audit Directory Service Access" and you customize the SACLs on AD objects, ensure to at least audit all failed and successful attempts to modify any of the following by the Everyone group (these are high-value targets):

  Enterprise Admins
  Schema Admins
  Domain Admins
  Group Policy Creator Owners
  Pre-Windows 2000 Compatible Access
  Cert Publishers
  Domain Controllers
  DnsAdmins
  All user/computer accounts which are members of any of the above.
  Any other users, groups or computers which are high-value targets.

- Consider investing in a host-based IDS which is capable of scanning for AD changes. The IDS should ideally be capable of searching for user-definable events in the logs in near real-time. Barring a full IDS, consider writing your own scripts to do the same.



Combine AD Permissions With PowerShell JEA

PowerShell 2.0 and later supports using cmdlets for managing Active Directory after you import the AD module (see below). PowerShell 5.0 and later supports Just Enough Admin (JEA). JEA can be combined with AD permissions for very precise control! Imagine safely granting auditors, help desk personnel, managers, contractors and others just the cmdlets they need, allowing only the parameters required, limiting arguments using regular expression patterns, and backing up these safeguards with AD permissions! This is a very nice solution, and don't forget that graphical tools can leverage all this as well, your less-technical users never have to open a command shell.

Requirements for Active Directory Cmdlets

First, you must run PowerShell 2.0 or later from a Windows 7 or Server 2008-R2 or later computer. You must run the AD cmdlets from this machine.

Second, at least one domain controller in the domain must be running one or the other of the following services:

- Active Directory Web Service (ADWS)
- Active Directory Management Gateway Service (ADMGS)

These services both do approximately the same thing, but the two products are for different versions of Windows Server:

- ADWS: Built into Server 2008-R2 or later domain controllers by default.
- ADMGS: Manually installed on Server 2003/2003-R2/2008 domain controllers.

Simply installing Active Directory on Server 2008-R2 or later to turn it into a domain controller will install and enable ADWS. Nothing else must be done.

For Server 2003/2003-R2/2008, you'll have to download and install ADMGS manually (the download URL has changed a few times already, please Google on "site:microsoft.com management gateway service adws").

The ADWS/ADMGS web service listens on TCP/9389 by default and will use SSL if a digital certificate is installed from a trusted Certification Authority (CA) and that certificate is marked as valid for server authentication.

Installing The Active Directory Module

The AD PowerShell module is available by default on domain controllers running 2008-R2 or later, but it must be installed manually on non-controllers running Windows 7 or later.

To install the module on Windows Server 2008-R2 or later, open the Server Manager console > right-click Features > Add Features > Remote Server Administration Tools > Role Administration Tools > AD DS Tools > check the box for "Active Directory PowerShell Snap-In" > Install. You may wish to install all of the RSAT tools while you're at it.

To confirm installation from the command line on Windows Server 2008-R2 or later, run "servermanagercmd.exe -q" and note that "Active Directory PowerShell Snap-In [RSAT-AD-PowerShell]" is marked as installed with an X. To install from the command line, run "servermanagercmd.exe -i RSAT-AD-PowerShell".

To install the module on Windows 7 or later, download and install the latest version of the Remote Server Administration Tools (RSAT) from Microsoft's web site, then go to Control Panel > Programs and Features > Turn Windows Features On or Off > Remote Server Administration Tools.

Load The Active Directory Module Into PowerShell

Installing the AD snap-in makes the AD module available on the computer, but that does not mean it is loaded into every instance of PowerShell you launch. To install the AD module into your current PowerShell process, or to load the module in any script which will require it, run the following:

```
# C:\SANS\Day4-Admins\ActiveDirectory\Active_Directory.ps1

Import-Module ActiveDirectory
```

> **Note:** From the Administrative Tools folder of the Start Menu, if you launch "Active Directory PowerShell", then this shell imports the module automatically; see the properties of that shortcut to see how it's done.

To see all your currently-loaded modules, run the following, then look for the ActiveDirectory module:

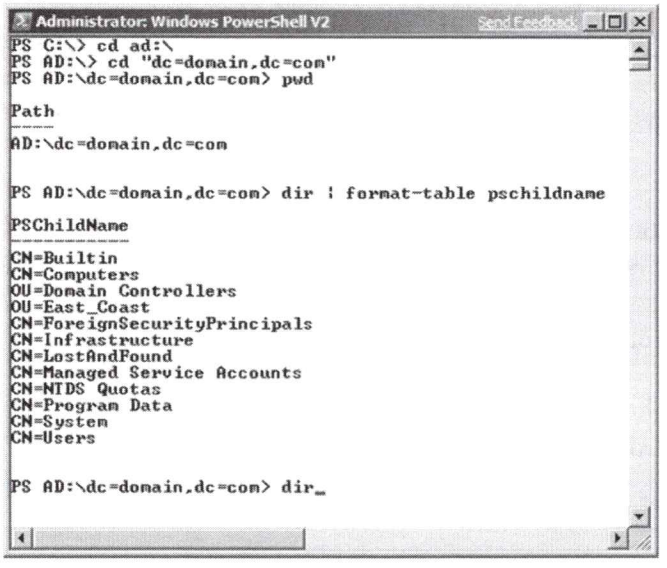
```
Get-Command -Module ActiveDirectory
```

And of course you can search these new cmdlets for help information (*verb*-**AD***noun*):

```
Get-Help *-AD*
```

Browse The Active Directory Provider

If your computer is a domain member, then, after loading the AD module, a new provider drive named "ad:" is automatically created for you. You can change location into it and list its contents just like you can list files in a folder. Note that the AD paths are in X.500 distinguished name format, e.g., if the DNS name of your domain is "testing.local", then the distinguished name of your domain is "dc=testing,dc=local".



If you wish to connect to a domain controller in a different domain or forest, you'll need to use the new-psdrive cmdlet to map a drive name (other than "ad:" of course) to an appropriate controller. If you wish to connect to the global catalog port of that controller, use the -globalcatalog switch. If you need to manually authenticate, use the -credential switch. The -root switch specifies which container in AD is targeted, with the RootDSE container being the default if left blank. If you specify a DNS domain name instead of a specific controller for the -server switch, then PowerShell will identify and connect to a controller in that domain automatically using SRV record queries in DNS.

To map a drive named "OtherAD:" to the global catalog port on a domain controller with an IP address of 10.4.4.1 with a username of "testing\tim":

```
New-PSDrive -PSProvider ActiveDirectory -Server 10.4.4.1
```

```
    -GlobalCatalog -Root "" -Credential "testing\tim"
    -Name OtherAD
```

If you will execute many commands in a row which interact with an Active Directory domain, it's best if you switch to an AD drive in PowerShell first. Changing to a drive mapped to the appropriate AD location will result in a 75% performance increase when many commands against AD are executed in succession. The AD-related cmdlets, if executed while the present working location in PowerShell is outside of an AD drive, will open and close a new connection with every command, but, if executed while in an AD drive, will reuse the single connection maintained by the provider drive itself, which is much more efficient.

If you wish to connect to a special domain controller, such the Schema Master or PDC Emulator, then use cmdlets like get-addomaincontroller, get-adforest or get-addomain to obtain these server names in your forest (see the help for these cmdlets).

Manage User Accounts

When creating objects in AD, if you are in an AD drive location at the time of creation and you don't specify a different container, then your current location (pwd) determines where the new object will be created in AD. If outside of an AD drive, then the default location for that type of object will be used; if there isn't a default, you'll need to provide a path to the desired container, e.g., "ou=boston,ou=east_coast,dc=testing,dc=local".

When creating user accounts, the "SamAccountName" property is the old backwards-compatible username field, which can be different than the display name. The SamAccountName is the familiar username you enter after hitting Ctrl-Alt-Del.

Also, note that a password must be converted into a so-called *secure string* before it can be assigned to an account. This is done with the built-in CovertTo-SecureString cmdlet, which outputs an object which contains the obfuscated data, not a plain string.

To create a new user account with a password assigned to it:

```
$pw = convertto-securestring "Pa55wurD" -asplaintext -force

new-aduser -name "Justin McCarthy" -samaccountname "Justin"
   -accountpassword $pw -enabled $true
```

To reset the password on an existing user account by being prompted for the new password at the command line:

```
function Reset-Password ($UserName)
{
    $pw = read-host -assecurestring -prompt "Enter New Password"

    set-adaccountpassword $UserName -reset -newpassword $pw
}
```

```
Reset-Password -UserName Justin
```

Note: In many AD cmdlets, "-Identity" is the default parameter.

To change some attributes of an existing user account:

```
set-aduser Justin -Description "Engineering"
   -EmailAddress "justin@sans.org" -SmartcardLogonRequired $true
```

To disable, enable or unlock a user account:

```
disable-adaccount "cn=Justin,ou=boston,dc=testing,dc=local"

enable-adaccount "Justin"

unlock-adaccount Justin
```

To unlock all user accounts whose name matches "*admin*":

```
get-aduser -filter {name -like '*admin*'} | unlock-adaccount
```

To set the expiration date on a user account to a specific date and time, to expire an account in 3 days, to expire an account in 12 hours, or to clear any expiration settings from an account, again using a variety of equally-valid naming formats:

```
set-adaccountexpiration Justin -datetime "12/25/2016 6:00 AM"

set-adaccountexpiration Justin" -timespan "3"

set-adaccountexpiration Justin -timespan "12:00"

clear-adaccountexpiration Justin
```

To delete a user account with a username, i.e., a SamAccountName:

```
remove-aduser -identity Justin
```

Manage Computer Accounts

Computer accounts are managed in a very similar way to managing user accounts.

To create a new computer account in the current AD drive location:

```
new-adcomputer -samaccountname SERVER38 -name SERVER38
```

To create a computer account and place it in a particular AD location:

```
new-adcomputer -samaccountname SERVER44 -name SERVER44
   -path "ou=boston,ou=east_coast,dc=testing,dc=local"
```

To modify a property of an existing computer account:

```
set-adcomputer SERVER44 -OperatingSystem "Server 2016 Standard"
```

To delete an existing computer account with no confirmation prompt:

```
remove-adcomputer SERVER44 -confirm:$false
```

Manage Groups

Universal, global and domain local groups can all be managed through the AD cmdlets. When you create a group, it is a security group by default, not distribution.

To create a global group in the current AD drive location:

```
new-adgroup "Sales" -groupscope global
```

To create a global group in a specified AD container:

```
new-adgroup "Sales" -groupscope global
   -path "ou=boston,ou=east_coast,dc=testing,dc=local"
```

To add one or more members to an existing global group (separate multiple members with commas):

```
add-adgroupmember "Sales" -member Justin,Administrator
```

To get the current members of a group:

```
$members = get-adgroupmember "Sales"

get-adgroupmember "ou=boston,ou=east_coast,dc=testing,dc=local"
```

To remove a member from a group:

```
remove-adgroupmember "Sales" -member Justin
```

To list all the groups to which a user is an immediate member, but not the additional groups of which a user is a member via nested grouping:

```
get-adprincipalgroupmembership Justin
```

To delete a group, but not its members:

```
remove-adgroup "Sales"

remove-adgroup "cn=Sales,ou=dallas,dc=testing,dc=local"
```

Forest and Domain Objects

To obtain an object representing an AD forest using the name of the root domain of the forest, the forest membership of the computer running your PowerShell commands, or the forest membership of the user account with which you logged on:

```
get-adforest "testing.local"

get-adforest -current localcomputer

get-adforest -current loggedonuser
```

To obtain an object representing an AD domain using the name of the domain, the domain membership of the computer running your PowerShell commands, or the domain membership of the user account with which you logged on:

```
get-addomain "testing.local"

get-addomain -current localcomputer

get-addomain -current loggedonuser
```

Search-ADAccount Cmdlet

There are various ways to query AD information, including not just users and groups, but also whole forests and domains. This is important because the output of a search is often piped into other commands which process that output.

The search-adaccount cmdlet is specifically for searching for user and computer accounts on the basis of password expiration, account enabled status, whether the password can be changed, and logon history.

To search for accounts that are disabled, whose passwords have already expired or whose passwords are set to never expire:

```
$results = search-adaccount -accountdisabled

search-adaccount -passwordexpired

search-adaccount -passwordneverexpires
```

To search for accounts which will expire within 7 days:

```
search-adaccount -accountexpiring -timespan "7"
```

To search for accounts that have not logged on in over 180 days:

```
search-adaccount -accountinactive -timespan "180"
```

Get-ADObject, Get-ADUser and Get-ADComputer

The all-purpose search tool, which can be used to look for any type of object using either PowerShell or LDAP query syntax, is get-adobject. This cmdlet deserves a longer discussion of its parameters and how to use them.

Here is an example of how to use the get-adobject cmdlet, more specifically, how to get an array of objects which represent computer accounts in a particular domain or OU:

```
$results = get-adobject -filter {(objectclass -eq "computer")}
   -searchbase "dc=testing,dc=local"
```

The -filter parameter is for specifying the criteria and boolean operators with which to search AD. The search criteria can include schema class and property values, and multiple tests can be combined together using boolean operators (-and, -or, -not). When a class name or property is evaluated, the evaluations can use any of the standard PowerShell comparison operators (-eq, -like, -lt, -match, -gt, and so on).

The -searchbase parameter specifies where the search should begin in AD. The location is always expressed in distinguished name X.500 format; for example, to search the entire domain, specify "dc=testing,dc=local", or to search just one OU in that domain, specify "ou=boston,ou=east_coast,dc=testing,dc=local". By default, all searches will include any subcontainers nested underneath the specified searchbase, but this can be changed with the -searchscope parameter. If you do not specify a searchbase, the default is the domain container of which the computer running the command is a member, or, if you are in an AD drive when you run the command, the present location of the AD drive you are in.

To obtain an array of all user accounts in the local domain:

```
$results = get-adobject -filter { (objectclass -eq "user")
   -and (objectcategory -eq "person") }
```

But why did we have to include "objectcategory" in the criteria when searching for users? Because computer accounts are also indirectly derived from their user class in the schema, hence, we needed to exclude the computer accounts from the results. To simplify this, just perform your searches for user accounts with the get-aduser cmdlet and the '(objectclass -eq "user") -and (objectcategory -eq "person")' portion of the search filter is more-or-less added for you automatically. In addition to get-aduser, you also have get-adcomputer and get-adgroup for the same purpose, and all of these support the -filter and -searchbase parameters.

To find all users whose name begins with "R" in the Sales organizational unit:

```
get-adobject -filter { (name -like "r*")
   -and (objectclass -eq "user")
   -and (objectcategory -eq "person") }
   -searchbase "dc=testing,dc=local"
```

Or to perform that same search for "R*" users, but with the get-aduser cmdlet instead so that we don't have to explicitly include the objectclass and objectcategory criteria:

```
get-aduser -filter {name -like "r*"}
   -searchbase "dc=testing,dc=local"
```

What criteria can be used in the filter? Remember that the ADSI Edit snap-in lets you see object properties, and so does the Attribute Editor tab in the properties of an object in the AD Users and Computers snap-in, but this discovery of properties can be from within PowerShell too of course.

To examine the properties of the Administrator account:

```
$me = get-aduser Administrator -properties *

$me | get-member

$me | format-list *
```

Once you can see the properties of the types of objects you're interested in, you can start to construct filters to select just the objects you want. Notice that if you do not specify "-properties *" when you get an object, you will not get all the properties of the object, you'll only get a small set of essential properties.

To get all users in the domain who are listed as living in Dallas:

```
get-aduser -filter { City -eq "Dallas" }
```

To find the user accounts in the Boston OU who have suffered more than ten bad password logon attempts since the last successful logon with that account:

```
get-aduser -filter { badpwdcount -gt 10 }
   -searchbase "ou=boston,ou=east_coast,dc=testing,dc=local"
```

To find all users whose LastLogonTimestamp field indicates a logon between 1.Jun.2016 and 30.Aug.2016 (but remember that this field is only accurate within +/- 14 days):

```
$begin = get-date "June 1, 2016"

$end   = get-date "August 30, 2016"

get-aduser -filter { (lastlogontimestamp -gt $begin) -and
```

```
                    (lastlogontimestamp -lt $end)
                }
```

To find all organizational units created within the last 30 days:

```
$30daysago = $(get-date) - $(new-timespan -days 30)

get-adobject -filter { (objectclass -eq
   "organizationalunit") -and (whenCreated -gt $30daysago) }
```

To find users with names like "*admin*" in the Boston OU whose bad password count is greater than 100, likely indicating password-guessing attacks:

```
get-adobject -searchbase "ou=Boston,dc=testing,dc=local"
             -filter {
                (objectclass -eq "user") -and
                (objectcategory -eq "person") -and
                (name -like "*admin*") -and
                (badpwdcount -gt 100 )
              }
```

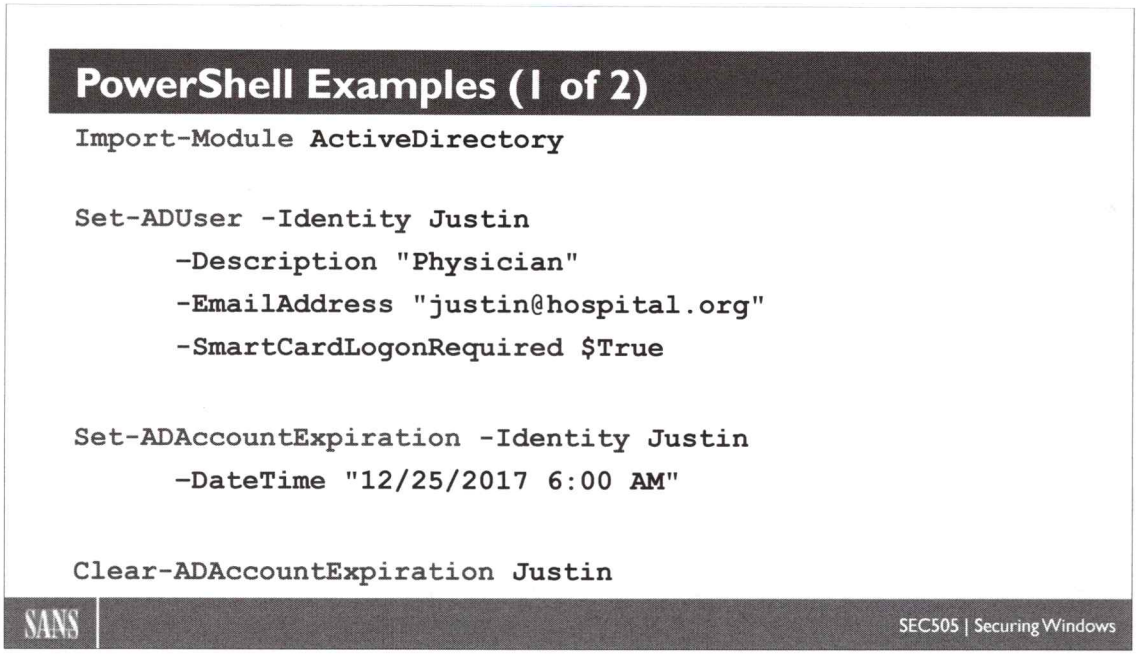
To find all users whose passwords never expire and who are not required to logon with a smart card when logging on interactively at the console (see the Account tab of the user):

```
get-aduser -filter { (PasswordNeverExpires -eq $true)
   -and (SmartCardLogonRequired -eq $false) }
```

To find all users whose workstation logon restrictions are not blank (blank is the default) and which have logged on at least once at some computer somewhere:

```
get-aduser -filter { (LogonWorkstations -like "*")
   -and (logonCount -gt 0) }
```

OK, from all these examples you should be able to start constructing your own filters!



PowerShell Examples (1 of 2)

Installing the AD snap-in makes the AD module available on the computer, but that does not mean it is loaded into every instance of PowerShell you launch.

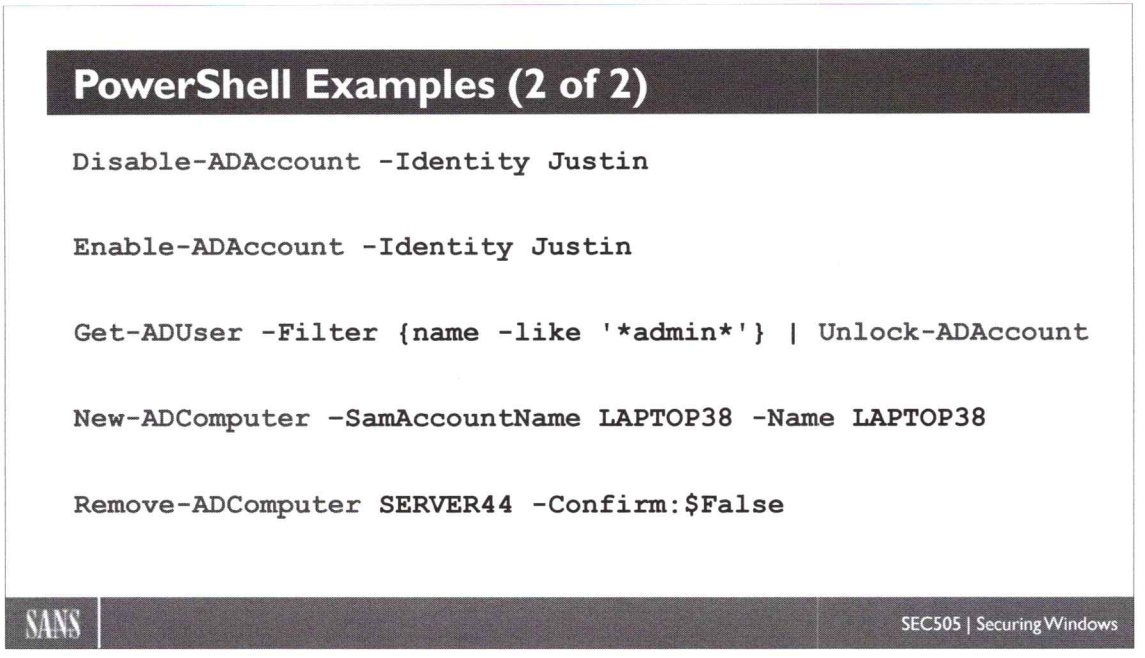
To install the AD module into your current PowerShell process, or to load the module in any script which will require it, run the following:

```
# C:\SANS\Day4-Admins\ActiveDirectory\Active_Directory.ps1

Import-Module ActiveDirectory
```

Manually importing the module this way is not supposed to be necessary, it's supposed to be loaded on-the-fly automatically, but it seems to work better when it's loaded explicitly (and it doesn't hurt anything either).

The Set-AdUser cmdlet can modify existing user accounts. This cmdlet has many parameters, not just the ones shown in the slide.

Set-AdAccountExpiration can set an automatic self-expiration date and time. This might be used to auto-expire the accounts of temporaries and contractors working on a six-month project.



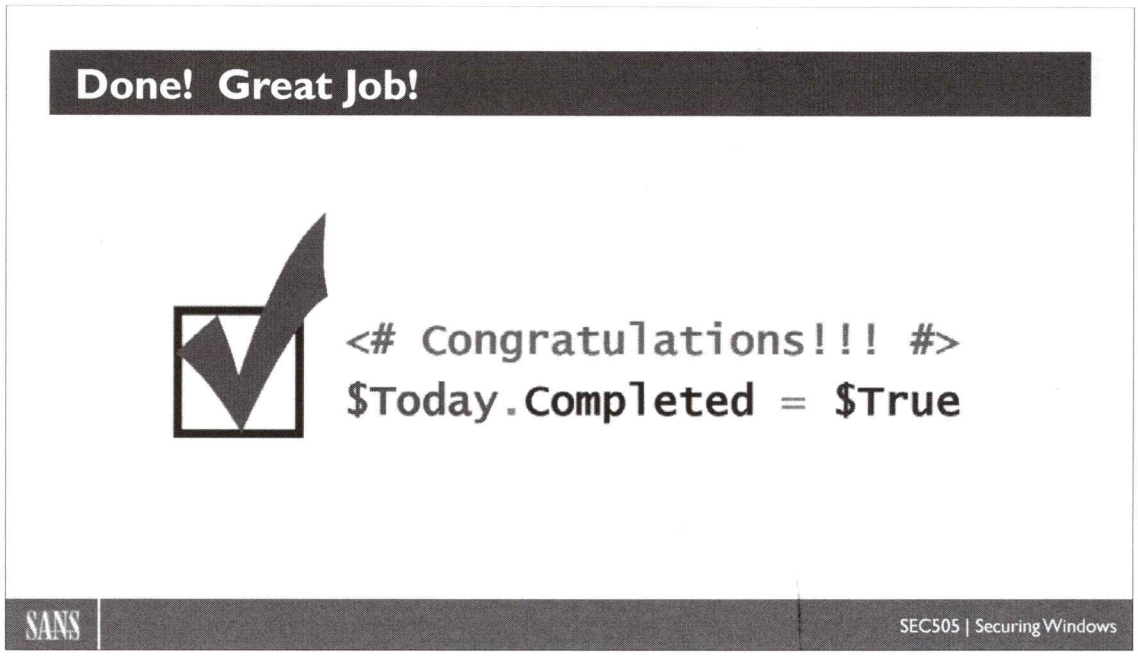
PowerShell Examples (2 of 2)

The Disable-AdAccount and Enable-AdAccount cmdlets are fairly self-explanatory just be reading their names. Imagine the help desk group having access to these cmdlets, their parameters and arguments controlled by JEA. Active Directory permissions could also be used --alone or combined with JEA-- to restrict which users could be enabled/disabled.

Get-AdUser and other cmdlets support a -filter parameter to perform searches using strings with wildcards. Notice that the output of this command is being piped into Unlock-AdAccount to unlock all the users who match the filter. Almost all AD cmdlets support the piping of their output into further commands.

New-AdComputer requires both a Security Accounts Manager (SAM) account name and also a normal common name. The SAM was the domain database on Windows NT, and for the sake of backwards compatibility and third-party application compatibility, user and computer accounts still require these old NetBIOS-compatible names.

When you delete a computer account, by default you will get a pop-up dialog box asking you to confirm this potentially-damaging action. If you do not want to be prompted, if you want to remove the computer account without any further confirmation, then use "-confirm:$false" with caution.



Congratulations!

You have finished the course!

Thank you for attending this seminar, and please complete the evaluation form. Your feedback plays an important role in the development of future courses and the editing of this current one.

Thank You!