White Paper

STEALTHbits TECHNOLOGIES

Top 10 Ways to Identify and Detect Privileged Users

www.stealthbits.com | 201-447-9300

Identify threats. Secure data. Reduce risk.

STEALTHbits TECHNOLOGIES

Table of Contents

Privileged users are the penultimate goal of cyberattacks. Once attackers have privileged access, it's only a small step to the information they want to steal. Cybercriminals leverage tools such as malware and phishing scams to gain a foothold within your organization, looking for ways to access and utilize credentials. In "wash, rinse, repeat" fashion, attackers patiently claw and scrape their way from first gaining access to a low-level local account all the way up to getting the highest privileged accounts in the system. The whole focus is on gaining access to as much sensitive—and valuable—data as possible.

And the biggest prizes are your privileged accounts in Active Directory (AD).

After someone has privileged access to AD, there is little on your network that they can't access. Nearly every piece of data, every system, and every application relies on AD. A systematic approach to both understanding and governing who has privileged access to AD becomes vital.

The task involves identifying who has access, as well as detecting when new users obtain privileged access. These questions might sound simple, but as you'll see, some methods that attackers use to attain privileged access to AD are part of a multistep process. It's in the details of this process that you can identify where your risks lie, in the form of users, groups, user rights, and privileges. You can also gain an understanding of which changes and actions (both within and outside of AD) need to be monitored if you are to maintain a vigilant stance over your security.

The goal of this paper is to educate you on all the methods a user can utilize to gain privileged access. This paper will empower you to identify who has privileged access and will tell you how to detect changes that give someone access using any of the methods mentioned.

We've built a Top 10 list of these methods, ranging from the downright obvious to *"How in the world did you come up with that?"* Each method outlines a way in which an attacker can obtain privileged access to AD. Some strategies, such as being added to the Domain Admins group, are direct; others, such as attaining Local Admin access over a domain controller, are indirect. All are stepping stones towards eventually gaining access within AD. Let's look at each and see where you need to focus your efforts to both maintain the current state of security and monitor for changes to that state.

STEALTHbits: Principals of Privilege

The challenge with AD security is that it's a constantly moving target. Memberships change, permissions are assigned, configurations are modified—and IT has time to audit only once a year. So AD security becomes a one-off, infrequent task when the auditors show up and is forgotten the rest of the time.

But attackers won't wait until audit time to spring their dastardly plans. You need a way to understand the state of your AD while keeping a watchful eye on daily changes.

Although you can accomplish most privileged user management with native tools, the reality is that no organization without a dedicated solution ever stays on top of these tasks. Countless control audits of AD-based environments show that organizations without the tools to automate the monitoring and reporting of privileged access within their AD-centric environments always end up having embarrassing numbers of people with unsanctioned authority.

STEALTHbits' AD solutions automate the laborious tasks associated with constantly ensuring AD security. Tools to identify, audit, enforce, recover, and report privileged access within AD streamline the process of efficiently gaining visibility and intelligently responding.

Look for insights from STEALTHbits throughout this paper.

1. Built-in privileged groups such as Domain Admins

Built-in privileged groups are an obvious place to start. Consider any group that provides privileged access within AD (e.g., Administrators, Domain Admins, Enterprise Admins, and Schema Admins). Also think of groups that provide privileged access on endpoints (i.e., any groups that are members of the Local Administrators group on a given endpoint).

You have a few ways to identify members of these groups. The one that should come to mind first is the use of Active Directory Users & Computers. By simply revealing the properties of a given group and looking at the Members tab, you can see which users are privileged by means of their membership.

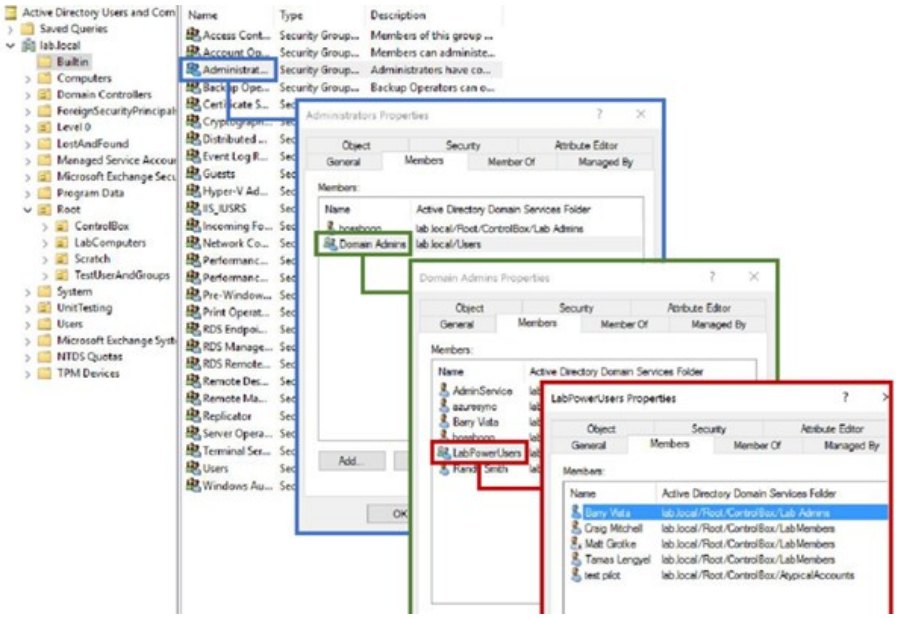
Because there aren't that many groups, you can use this approach manually on occasion. A bit faster method can be to use PowerShell, via the *Get-ADGroupMember* cmdlet.

Keep in mind that periodic reviews of group members provide insight only into the state of the group at that moment. For example, should someone be added to the Domain Admins group and then removed shortly thereafter, your review of the group weeks later won't reveal that interim state.

2. Nested groups within privileged groups

Although equally obvious, groups that are nested within privileged groups are often overlooked. A group that's a member of the Domain Admins group might have a name that looks right, but you still need to check out exactly who is in that group. Following the nesting trail to identify each and every member is important. After all, these *are* memberships that give users rights to do just about *anything* they want in AD.

Take the following example of the Administrators group within AD. By traversing the nested members, you'll take several steps to uncover the complete list of every group member.



A group thought to have only a few members can quickly grow to tens or hundreds of users—especially when groups are repurposed without checking to see to which other groups they belong. These factors make group nesting, in many organizations, more of a problem than a benefit.

Numerous PowerShell scripts are available online, but most focus only on the immediate members of the queried group. The next example uses a special LDAP filer (shown in red) to perform a deep membership search of the specified group ($groupDn).

```
param([string]$groupDn )
$s = new-object system.directoryservices.directorysearcher
$s.searchroot = new-object system.directoryservices.directoryentry
$s.filter = "(&(memberOf:1.2.840.113556.1.4.1941:=$groupDn))"
$s.propertiestoload.add("name")
$s.propertiestoload.add("objectclass")
$r = $s.FindAll()
foreach ($e in $r)
{Write-Host
$e.Properties.objectclass[$e.Properties.objectclass.Count-1]:
$e.properties.name
```

When run, the output of this script looks like the detail for the Administrators group:

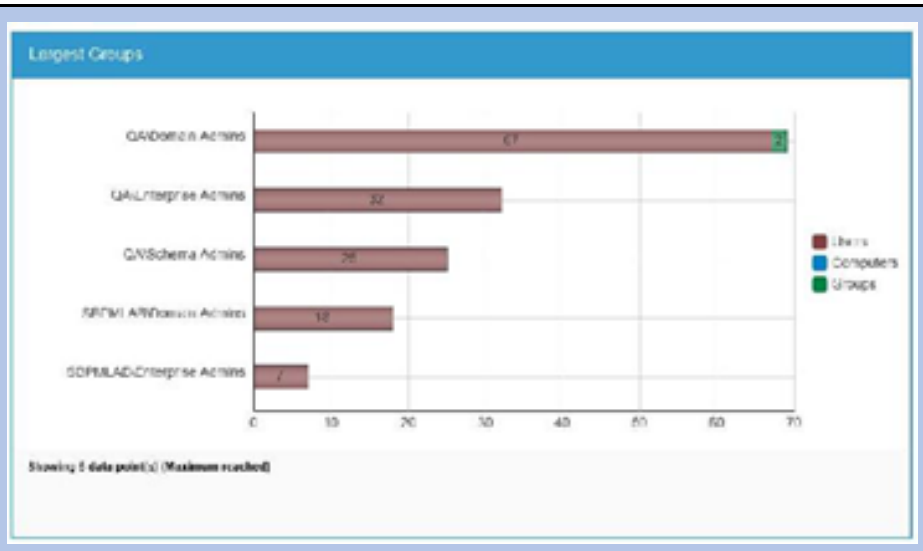
```
.\GetNestedMembers.ps1 -groupDn "CN=Administrators,CN=Builtin,DC=lab,DC=local"
group : Domain Admins
user : bosshogg
user : azuresync
user : Barry Vista
user : Randy Smith
user : AdminService
```

You'd need to run this script on a per-group basis.

STEALTHbits Insight: Watching the Watchers

All the manual methods report only the current state of Domain Admins (or any other elevated group) at the time the task is performed. What's needed is fast and simplified visibility—potentially across multiple domains—into the current state of membership of your privileged built-in groups, including the use of nesting.

For example, this figure shows how many members of Domain Admins are groups, as well as the total number of members, to provide IT with valuable insight around privileged access within AD. (The answer is 2, represented by the green section in the upper right).



3. Organizational unit permissions

When you think of administrative permissions, you probably think of built-in groups. But with AD's ability to delegate, users have the potential for elevated permissions to an organizational unit (OU) and all the objects residing within it. The simple act of granting the Everyone group Full Control permissions at the root (yes, it does happen) is the equivalent of adding someone to the Domain Admins group, and can lead to catastrophic breaches. Even someone granted these permissions at a lower level in the tree can be as harmful to the organization.

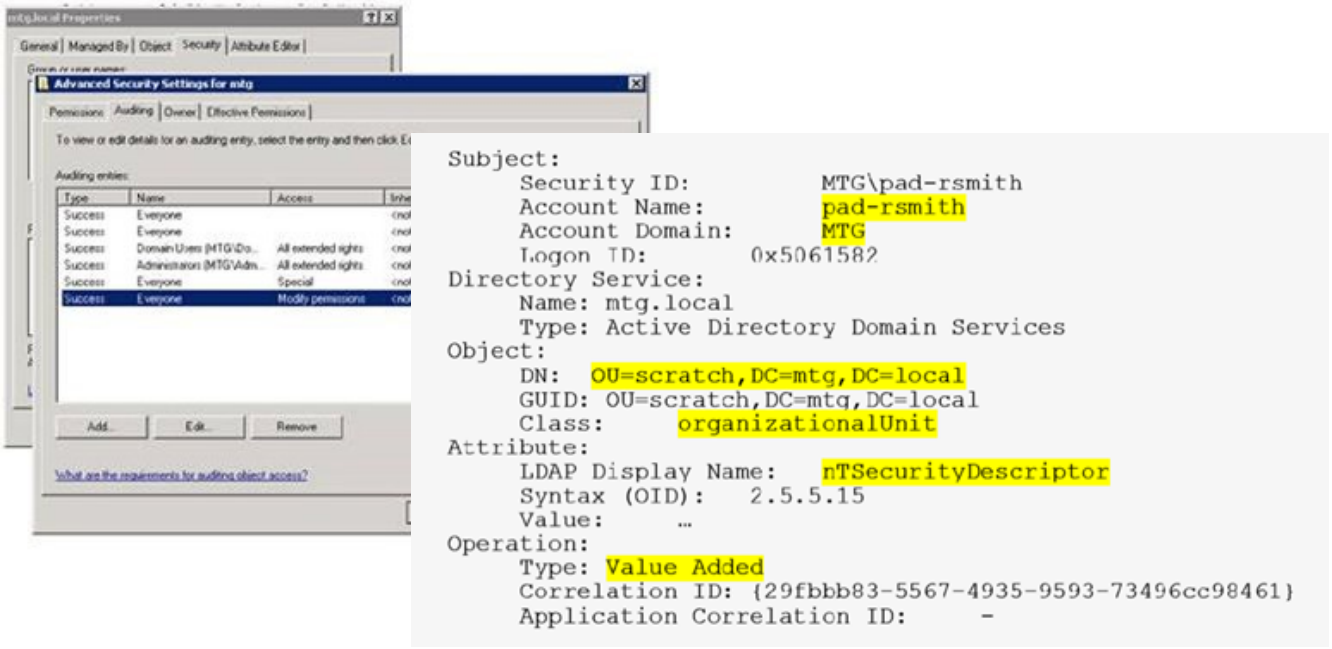
In some cases, the problem isn't even the default permission sets. Specific permissions can be granted that, when exercised, provide an ability to compromise accounts under a user's control, enabling access to critical resources and data.

In most cases, finding inappropriate permissions is like finding a needle in a haystack. There are a myriad of permissions for any given object. The reason is two-fold. First, AD assigns non-inherited permissions to every created object by default. This behavior is defined in the schema and happens automatically. Second is the issue of inheritance. Like NTFS security, permissions in parent OUs propagate down to child OUs and leaf objects.

This propagating of permissions down an OU path to every object within it further complicates the situation, as the challenge becomes both finding inappropriate permissions and traversing back up an OU path to identify their source.

The work necessary to identify the state of privileged OU permissions on your own requires navigating every OU one-by-one within Active Directory Users & Computers, looking for explicit permissions by going to the OU's properties, selecting the Security tab, and pressing the Advanced button to see all the permissions in detail. Should you see any Special permissions assigned, you'll need to edit them to see the individual assigned special permissions. These tasks involve hours of work but are necessary to ensure appropriate permissions.

Because you don't have time to do this task with any kind of reasonable frequency, you should track permission changes within AD to see the ongoing state of OU permissions. You can do so by enabling the Audit Directory Service Changes audit policy within the Default Domain Controllers Policy, and then adding an audit entry at the root of the domain to include the auditing of the successful modifying of permissions (as the next figure shows).



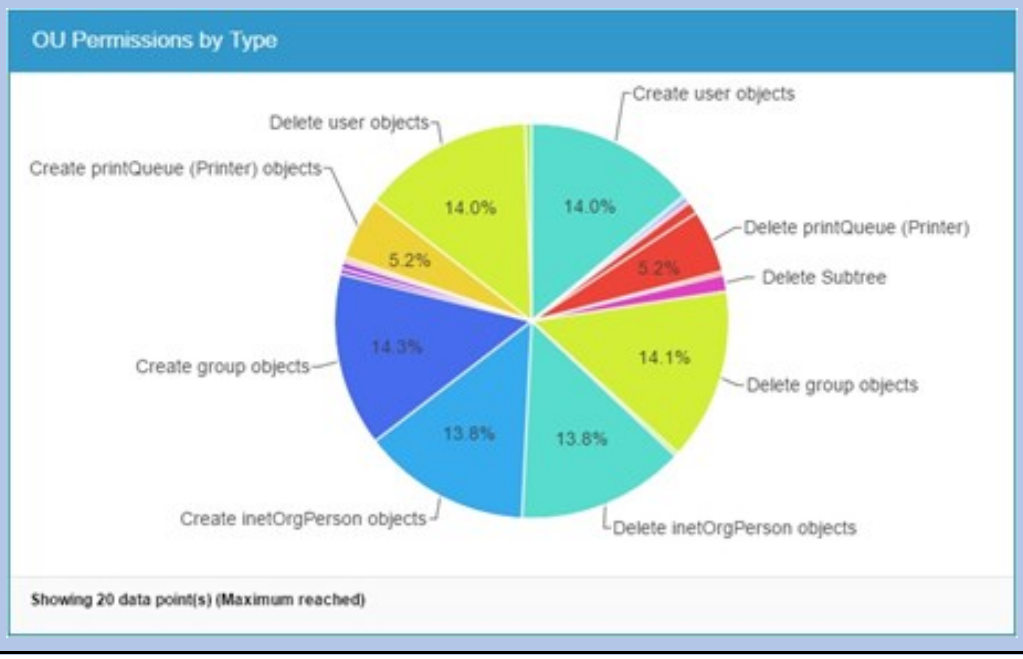
The audit results in the security log (also shown in the figure) identify a change to the ntSecurityDiscriptor (i.e., a change to the permissions), as well as details about which object is being modified, who is making the change, and more.

Although the audit entry doesn't spell out which permissions are being assigned, it does indicate to IT that the permissions for a specific OU need to be reviewed.

STEALTHbits Insight: Bringing Permissions to the Forefront

Permissions reviews are necessary, but they remain inefficient and, in some cases, ineffective. Many types of permission assignments can cause problems. For example, breaking inheritance from a parent OU can lock out those who are given rights to manage objects throughout the domain. Or, permissions to change only the passwords of users with access to sensitive data can provide an attacker or insider the ability to compromise an account.

STEALTHbits prides itself on its ability not just to automate the assessment of domain, OU, and object permissions in AD, but to also give IT teams' visibility into which permissions are assigned, as well as the capability to drill down quickly into questionable permissions to see exactly what's happening.



4. Admin equivalent rights on domain controllers

Thus far, this paper has covered only those places to look for users who have been granted direct privileged access within AD. But attackers don't expect to just email in some malware-laden attachment and immediately get control over a Domain Admin-level user account. Their path is one of patience, in which they seek out every account over which they can gain control, and then see how they can utilize each one to gain even more privilege. One such method is to compromise an account that has permissions within the operating system such that the account has full rights to the system.

While seemingly an indirect approach, if an account with Local Admin equivalence on a domain controller (DC) is compromised, the attacker has an unlimited ability to begin hacking AD from the outside. A number of Admin-equivalent user rights give an attacker full access to a DC:

Act as part of the OS	**Impersonate a client**
SeTcbPrivilege	SeImpersonatePrivilege
Back up files and directories	**Manage auditing**
SeBackupPrivilege	SeAuditPrivilege
Restore files and directories	**Replace a process level token**
SeRestorePrivilege	SeAssignPrimaryTokenPrivilege
Create a token object	**Take ownership**
SeCreateTokenPrivilege	SeTakeOwnershipPrivilege
Debug programs	
SeDebugPrivilege	

These rights are granted within Group Policy under the *User Rights Assignment* section of *Computer Configuration*. The rights are often granted to service accounts that can be compromised through tools such as *mimikatz*, which searches the memory of a compromised endpoint for cleartext passwords, password hashes, and Kerberos tickets to be used or hacked to gain access to the credentials associated with each.

How can you identify which accounts have these rights?

First, it's highly unlikely that two DCs have different user rights assigned. So you can simply look at the *Default Domain Controllers* policy and review the assigned user rights.

You can also use the *Get-AccountsWithUserRight* PowerShell script to see the listing of accounts:

```
(Get-AccountsWithUserRight -Right SeServiceLogonRight ).account
NT VIRTUAL MACHINE\Virtual Machines
IIS APPPOOL\.NET v4.5 Classic
IIS APPPOOL\DefaultAppPool
IIS APPPOOL\.NET v4.5
NT SERVICE\SQLTELEMETRY$SUPERCHARGER
NT SERVICE\MSSQL$SUPERCHARGER
NT SERVICE\SQLAgent$SUPERCHARGER
NT SERVICE\ALL SERVICES
RFSH\SQLServer2005SQLBrowserUser$RFSH
NT AUTHORITY\NETWORK SERVICE
```

You'll need to run this script per DC and per Right, so building out a larger script that comprehensively assesses each DC makes sense. At a minimum, run the script on one DC. You can download the script at ***bit.ly/2neFFdE***.

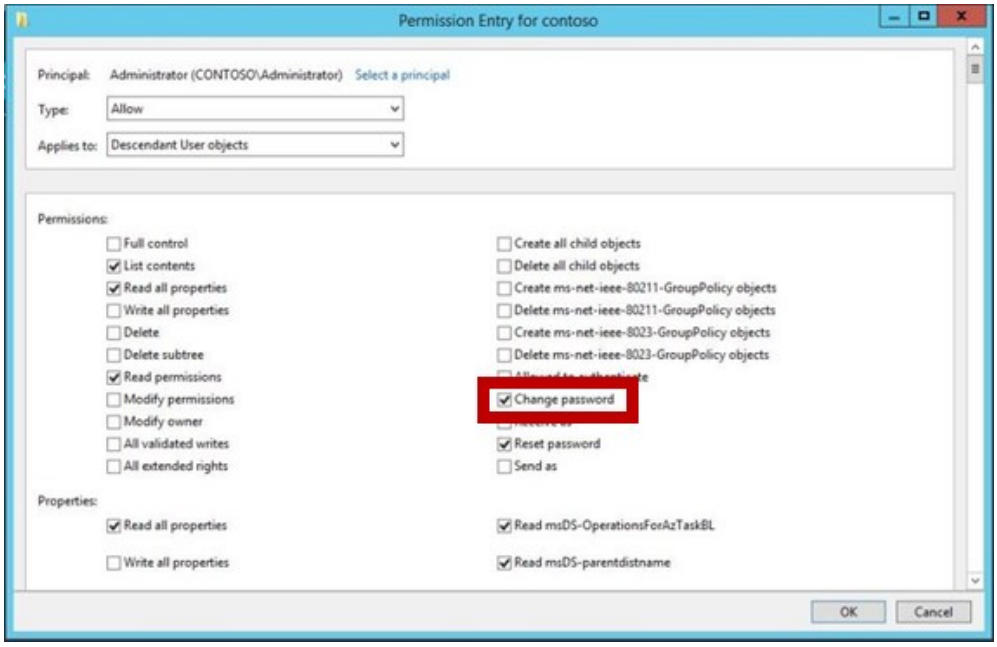
STEALTHbits Insight: User Rights and Groups

PowerShell scripts that enumerate accounts with specific user rights list only user and group accounts and don't provide any visibility into group detail, such as group members and nested groups. STEALTHbits automated reports surface this detail on group memberships as well as nested groups for users with rights to logon to domain controllers.

5. Users with password reset authority over other users

As previously mentioned, one capability granted by full control over objects is simply to reset the password of an account. While looking for those who have control over entire OUs and the objects that they contain, it's also important to look for those with the specific ability to reset the passwords of other users.

Don't confuse the *password change* permission with the *password reset* permission. The *password change* permission is granted to every user and enables them to change their own password. The *password reset* permission is the one you need to worry about. This specific permission (shown in the figure) is assigned within AD—usually at the OU level, but possibly at the user object level.



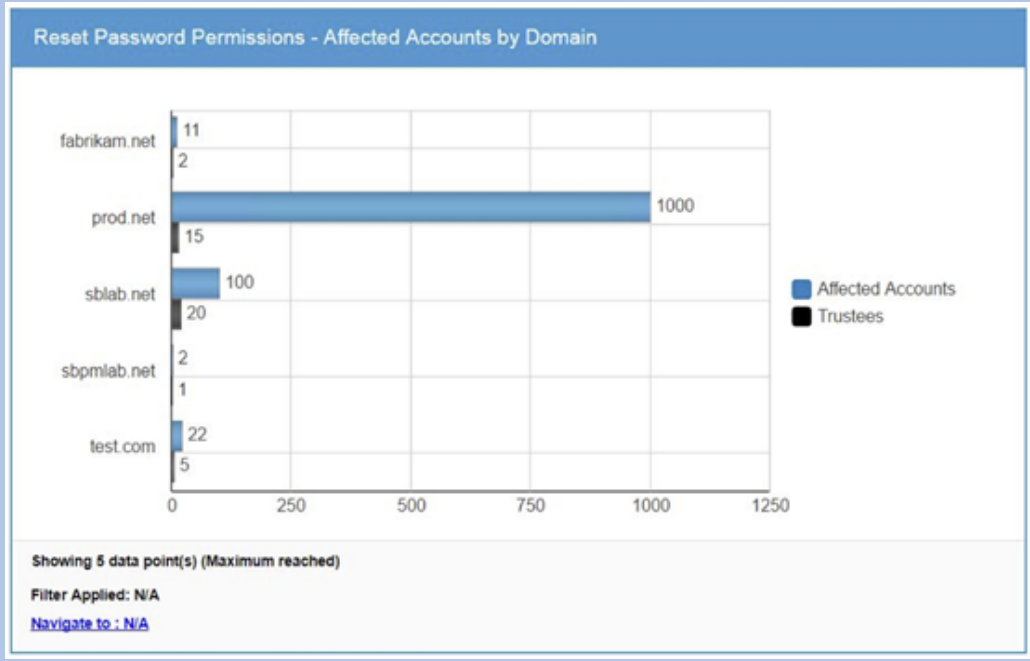
When granted to one user over another user's account (whether directly or via inheritance), this permission gives someone the ability to modify another user's password. The good news is that you already know how to find this permission assignment: The task involves the same work as finding privileges at the OU level. However, this time you might need to scour every user object to ensure that no malicious privileges are lying in wait. For example, some low-level bogus user account might have rights to change the CEO's password as an "insurance policy" for an IT contractor that thinks they might be getting canned soon.

Despite the ability to automate much of the searching with PowerShell (as with OU permissions), you need to audit and monitor for any time this permission is changed. Otherwise, you'll be stuck doing periodic, stateful audits that show you only the permissions at the time you run the script or perform the manual check.

STEALTHbits Insight: Monitoring for Password Privileges

The act of resetting a user's password with the intent of commandeering that account for malicious purposes is an attack that can easily slip under the radar. Unless the actions performed by the compromised account are under scrutiny, the only indicator an organization might have is the password reset act itself or—more proactively—the existence of the permission to do so.

As with any assigned permissions that elevate a user to a place of privilege, the ability to reset a user's password is one that needs constant monitoring. A periodic check of the state of AD is simply inadequate. STEALTHbits' suite of AD security products include the ability to actively keep watch on the state of reset password permissions. As shown in this figure, visibility into the state of password privileges is achieved; for example, 1,000 users in the prod.net domain can have their password reset by just 15 individuals.



This constant visibility serves as the basis for monitoring, reporting, and alerting to changes in, and use of, password reset privileges.

6. Users with knowledge of any privileged service accounts

Lorem Privileged service accounts (e.g., those used for Exchange, SQL Server, and backups) usually have some level of elevated—if not Local Administrative—privileges on the systems on which those accounts are used.

So it stands to reason that if someone has knowledge of a privileged account's credentials, that service account can be used maliciously. This possibility becomes an issue particularly on your DCs, as you don't want anyone other than the people who administer all of AD to have Admin access to a DC.

This situation can be difficult to quantify. Think about it: How many of your IT staff know the password to the backup service account? Many people find that question difficult to answer with any certainty.

Start by assuming that the IT team that is responsible for administering AD, or the applications that use the service accounts, know the passwords. Even then, appropriate usage of these accounts can easily be identified by auditing the logon of each service account, ensuring that you see only service startups (and not, say, an interactive logon at a DC's console).

STEALTHbits Insight: Auditing Service Accounts on DCs

You can use a number of events in a DC's Security log when auditing the logon of a privileged service account. Each event provides a piece of the overall "is a service account being improperly used" puzzle, but you need to correlate the events to make sense of it all. In total, you need to know when a privileged account is used and whether the logon type is service-related.

Events 4768 (A Kerberos authentication ticket (TGT) was requested) and 4672 (Special privileges assigned to new logon) can tell you when a privileged account is used during authentication (4768 also identifies where the account is used). But neither event provides the logon type. Both need to be correlated with event 4624 (An account was successfully logged on), which includes the logon type.

Although the data from event ID 4624 looks a little different across Windows Server 2008, 2012, and 2016, the values you care about are Account Name and Logon Type. Any type not equal to a value of 5 (which denotes a service startup) is an issue.

But before you do any of this, you must first know which accounts are your service accounts. STEALTHbits' extensive domain auditing capabilities include the ability to identify each privileged service account in the domain, as shown in this figure.

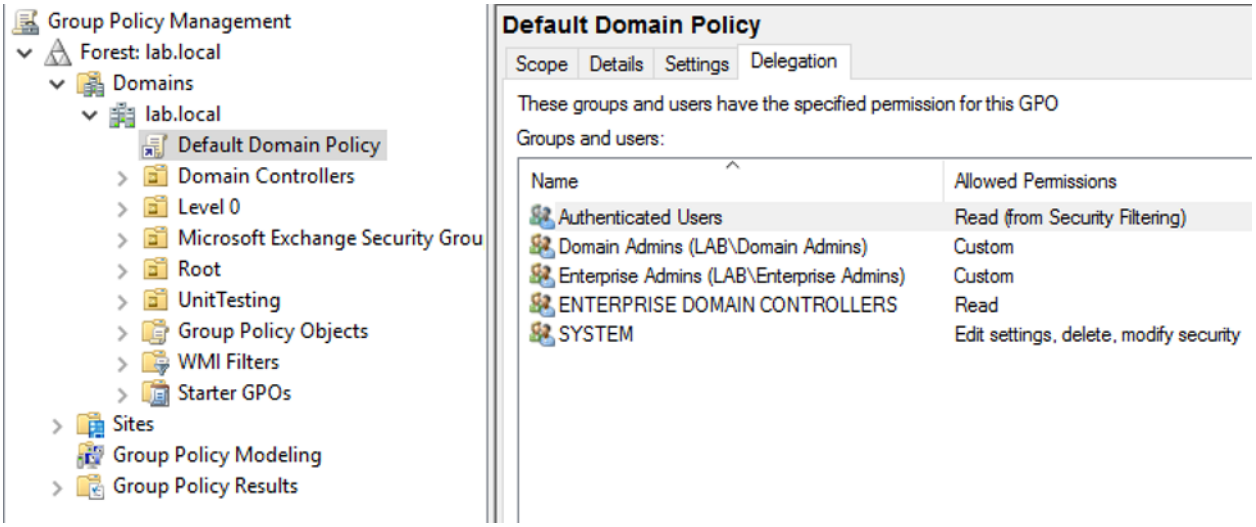
Service Account Details

Domain	User	NT Account	Service Principal Name	Service Class	Instance Name	Port
SBPMLAB	SBPMLAB\krbtgt	SBPMLAB\krbtgt	kadmin/changepw	kadmin	changepw	
SBITSINC	SBITSINC\krbtgt	SBITSINC\krbtgt	kadmin/changepw	kadmin	changepw	
SBITSINC	SBITSINC\RTCCompon	SBITSINC\RTCCompon	http/SBNJOCS05.sbitsin	http	SBNJOCS05.sbitsinc.co	
SBITSINC	SBITSINC\CWAService	SBITSINC\CWAService	http/im.sbitsinc.com	http	im.sbitsinc.com	
SBITSINC	SBITSINC\RTCService	SBITSINC\RTCService	sip/SBNJOCS05.sbitsinc	sip	SBNJOCS05.sbitsinc.co	
SBITSINC	CRMAPService	SBITSINC\CRMAPServi	HTTP/crm	HTTP	crm	

7. Users with write access to GPOs that are applied to DCs or servers running applications with domain privileged access

This issue builds on method #4 (Admin equivalent rights on domain controllers). Because these user rights are granted via a group policy, you also need to be wary of anyone with Write access to any group policy that is applied to DCs or servers with applications that have domain privileged access.

Begin with the permissions on all GPOs that are linked to either the Domain root or the Domain Controllers OU. As shown in this figure, you can review the Delegation tab to quickly see which users and groups have this capability.



In addition, identify any servers that run applications with privileged domain access. Perform the same permissions review on any policies that are linked to each parent OU of the server objects, to ensure that you have a comprehensive understanding of every user that can modify these policies.

8. User accounts with access to any AD management solutions

Many organizations use third-party solutions to simplify and enforce management delegation or to improve the productivity of management tasks. The use of these solutions creates two methods by which an attacker can achieve privileged access.

The first method is the use of either a service or proxy account (most AD management solutions use one) that is granted privileged access to all or a subset of AD to facilitate the solution's management ability. In some cases, these accounts are granted elevated privileges by means of some of the other methods discussed in this paper, such as membership in a built-in privileged group or OU-based permissions.

The second method is the use of any account with permissions within the management solution. Here, a typically low-level user might be granted the ability to perform tasks (e.g., reset another user's password) over a subset of accounts within AD. Depending on the level of delegation, gaining control over an account like this one is just as good as being a Domain Admin.

As with the other methods in this paper, to gain understanding of who has access, you need to determine whether any such applications are in use (keeping in mind that they might not be running on DCs). After you have a list of these applications, you need to identify any service or proxy accounts that have privileged access. You can use the same Directory Service auditing discussed previously to monitor what these accounts are doing.

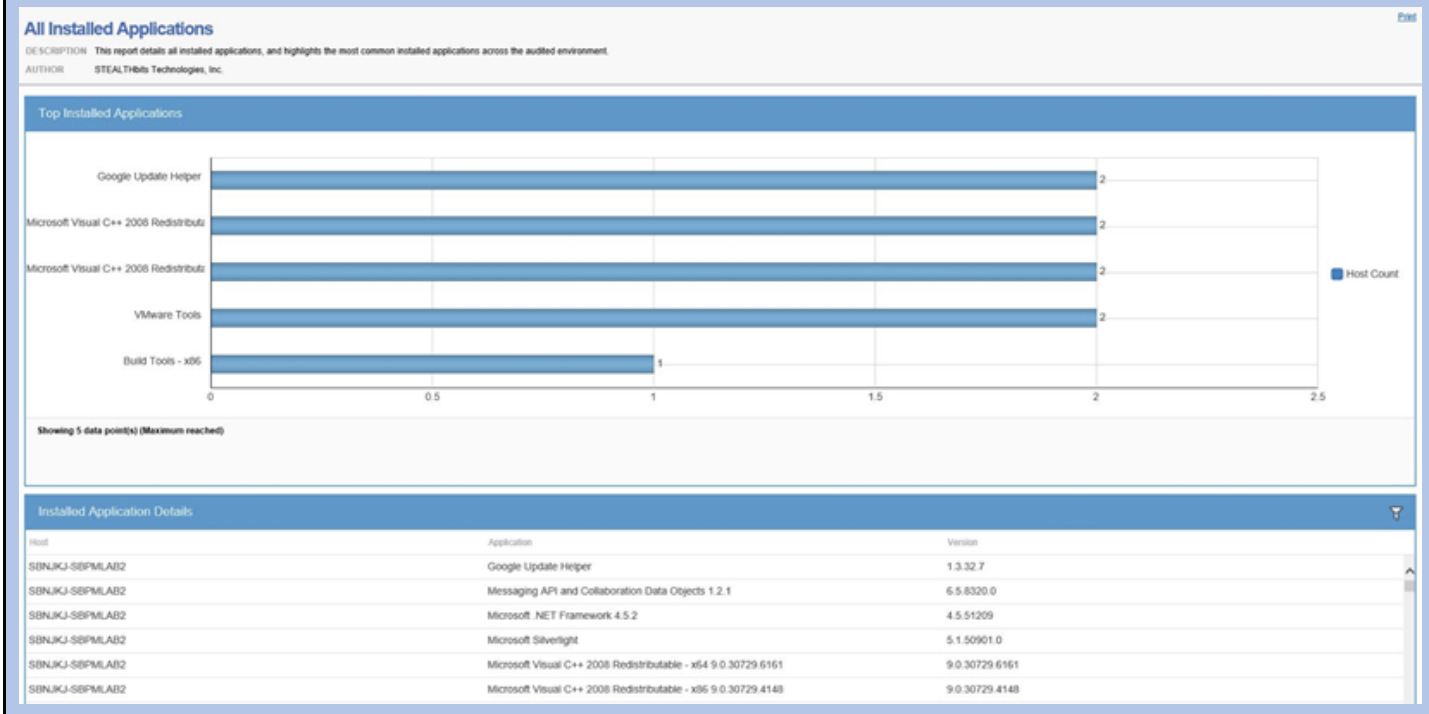
Because the use of service or proxy accounts obfuscates who specifically is making changes, see if the solution itself has any kind of audit trail that you can use to monitor for inappropriate behavior. If auditing is part of the solution, see whether alerts can be set up within the solution or can be piped out to a SIEM solution from which alerts can be configured.

STEALTHbits Insight: Application Awareness

Because the use of management solutions (or any other application with privileged access to AD) is something you need to know about, it's critical to have an up-to-date understanding of which applications are installed on a given server. Beyond the potential risk to AD, unsanctioned applications can bring unpatched vulnerabilities and can act as an entry point and foothold for external attackers.

Inventory generation can be scripted via PowerShell by obtaining the WMI information—specifically the Win32_Product class, which represents installed applications. However, this approach comes with the same issues mentioned previously: It returns a static result and needs to be run frequently. On top of that, only applications that are installed via Windows Installer show up in the list.

STEALTHbits audits systems across your network, looking for and providing visibility (among other data points) into installed applications. As shown in this figure, details about installed applications can be seen both on a per-application (at the top of the figure) and per-system (at the bottom of the figure) basis. These details provide insight into where risk to AD and the network as a whole might exist.



9. Virtualization infrastructure admins

A fair portion of this paper discusses the ways a user can gain privileged access over DCs or servers, which they can then try to use to gain access to AD. One additional method that needs to be covered occurs when running DCs or member servers within a virtual infrastructure.

Anyone who manages the virtual environment that hosts DCs or member servers has the equivalent of administrative access to a physical machine. For example, if you're running Hyper-V, members of the Hyper-V Administrators local group have Admin-level access on the guest operating system. Similarly, in VMware environments, the root user on an ESXi system has the same level of access.

You might wonder why this discussion points out both DCs and servers. The reason? The use of methods such as access to service accounts with AD privileged access or the presence of AD management solutions, which exist on member servers—also present the same potential privilege risk on virtual servers.

You need to identify which accounts have privileged access to your virtual infrastructure, either by enumerating Local Admin groups on a given DC or server, or by looking for privileged access within the virtual infrastructure itself.

10. Credential artifacts

When attackers gain entry into your network, artifacts are left behind in the memory of the servers and workstations that can be exploited to authenticate the attacker as a privileged account. When a user logs on to a system, pieces of credential information often remain after the user logs off. This information can include cleartext passwords, password hashes, NTLM hashes, and Kerberos tickets.

Tools such as Mimikatz locate credential details that can be used in a number of ways. For example, cleartext passwords can simply be reused, and hashes can be passed as part of an authentication request, using additional hacking tools to gain access to other systems.

Identifying which machines have privileged credential artifacts on them is nearly impossible. Therefore, your focus needs to be on determining where privileged users are logging on.

As previously mentioned, you can monitor event ID 4627 *(Special privileges assigned to new logon)* on Windows 10 and Windows Server 2016 systems for groups that are known to be privileged. On DCs, monitor event ID 4769 *(A Kerberos service ticket was requested)*, looking for users who are known to be privileged.

After you have an idea of how privileged accounts are being used, start by ensuring WDigest settings prevent cleartext passwords from being stored in memory. After that, the best protection is to establish multiple levels of privileged accounts. This way, an account that has Domain Admin privileges remains secure, whereas an account with Local Admin privileges is used to address an end-user issue.

Keeping a Tight Grip on Privileged Users

The privileged user is a much sought after level of access by cyberattackers. Whether with a Local Admin account on an endpoint or one with Domain Admin privileges, attackers take each account they can get their hands on and use them as a stepping stone to find and purpose accounts with even greater access.

Understanding exactly who your privileged users are, where and how privileged access can be granted, and when changes occur that affect access is of the utmost importance. By regularly assessing the current state of assignments while also monitoring any changes to those assignments, you put your organization in a constant and vigilant state of security.

Keep in mind, doing so is a manual process that, even with scripting, needs to be addressed frequently. Therefore, the assessing and detecting of privileged users is an obvious avenue for a third-party solution that automates some or all tasks that are covered in this paper. Without one, it's safe to say that no organization can stay on top of every method of privileged access.

Whether done manually or with a solution, the process of continually monitoring the risk that exists in the form of privileged users is crucial to maintaining security. By doing so, you gain visibility into the known paths that attackers take, while also gaining better control over the configuration of your AD security.

About Randy Franklin Smith

Randy Franklin Smith is an internationally recognized expert on the security and control of Windows and AD security. Randy publishes www.UltimateWindowsSecurity.com and wrote The Windows Server 2008 Security Log Revealed—the only book devoted to the Windows security log. Randy is the creator of LOGbinder software, which makes cryptic application logs understandable and available to log-management and SIEM solutions. As a Certified Information Systems Auditor, Randy performs security reviews for clients ranging from small, privately held firms to Fortune 500 companies, national, and international organizations. Randy is also a Microsoft Security Most Valuable Professional.

Disclaimer AND COPYRIGHT

About STEALTHbits Technologies, Inc.

STEALTHbits Technologies is a cybersecurity software company focused on protecting an organization's credentials and data. By removing inappropriate data access, enforcing security policy, and detecting advanced threats, we reduce security risk, fulfill compliance requirements and decrease operations expense.

Identify threats. Secure data. Reduce risk.

Learn More

Attend a Demo - http://www.stealthbits.com/events

Browse the Resource Library - http://www.stealthbits.com/resources

Ask us a Question - http://go.stealthbits.com/contact-us

Request a Free Trial - http://www.stealthbits.com/free-trial

Visit the Official STEALTHbits Blog - http://blog.stealthbits.com

STEALTHbits Technologies, Inc.
200 Central Avenue
Hawthorne, NJ 07506
P: 1.201.447.9300 | F: 1.201.447.1818
sales@stealthbits.com | support@stealthbits.com
www.stealthbits.com