# Using Active Directory’s AdminCount Attribute to Find Privileged Accounts

blog.netwrix.com/2022/09/30/admincount_attribute

Active Directory accounts with elevated privileges pose a serious security risk: They are a top target for attackers because they provide administrative access to systems and data, and they can also be misused by their owners, either deliberately or accidentally. Therefore, it’s critical for IT teams to keep close track of accounts with elevated permissions.

Handpicked related content:

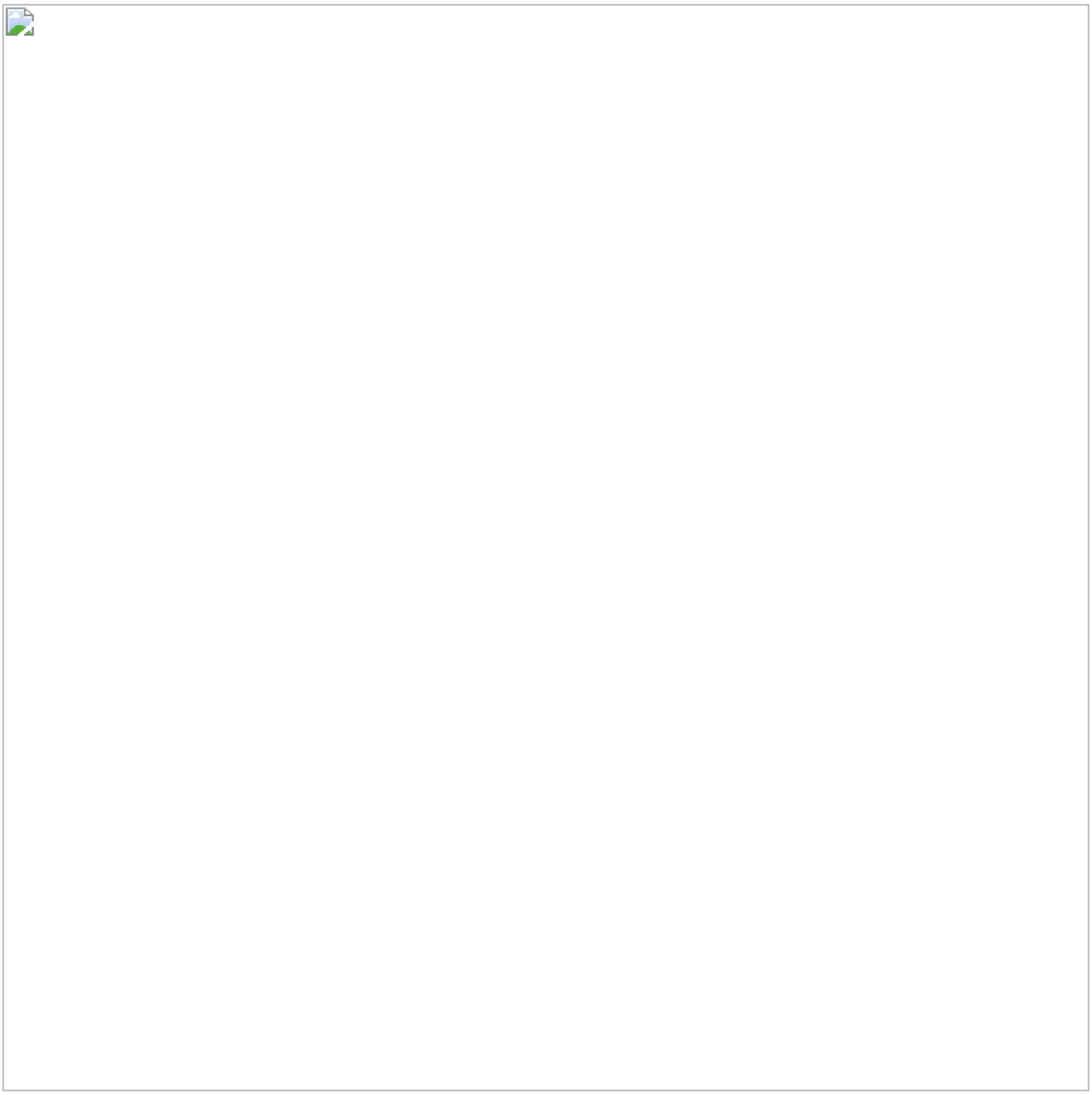
[Free Guide] Privileged Access Management Best Practices

One common strategy is to monitor the value of the Active Directory AdminCount attribute. All AD user, group and computer objects have this attribute. By default, it has the value “<NOT SET>”. But when the object is added (directly or transitively) to certain protected groups, the value is updated to “1”. As a result, checking this attribute seems like a good method for identifying objects with administrative privileges.

However, it’s not really that simple. Let’s take a deeper look at how the Active Directory AdminCount attribute works and explore its limitations for tracking privileged users.

## AdminCount and Protected Objects

The following table lists Active Directory’s default protected object sets, including the groups that may induce an update of the AdminCount attribute on its members:

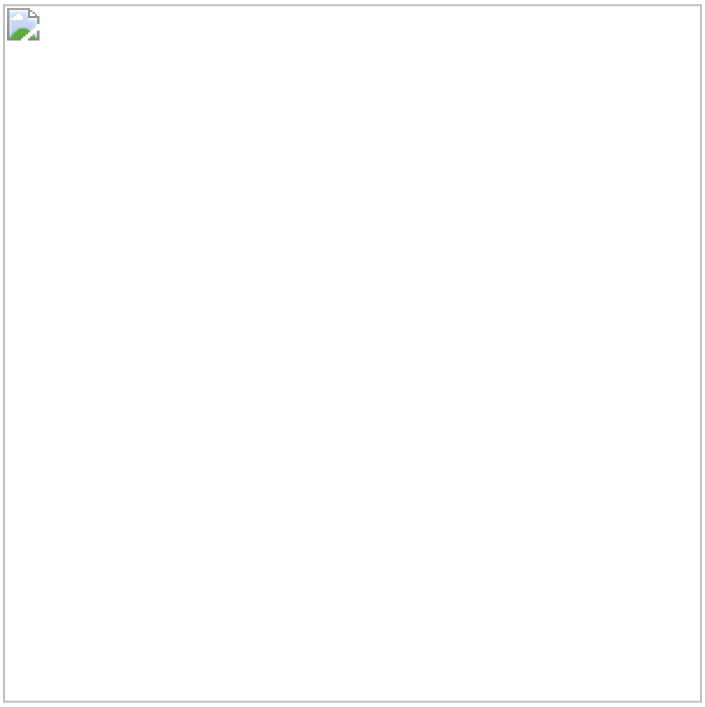


You may also have noticed that I said “may induce an update of the AdminCount attribute.” That’s because there are a number of variables that influence membership in the protected object set. Let’s start with the superscript annotations:

1. The default protected object set varies across the different Active Directory functional levels. This results in differences in the groups that may produce an update of the AdminCount attribute on their members.

2. The default protected object set includes four groups that can be manually excluded from protection: Account Operators, Backup Operators, Print Operators and Server This behavior was introduced in a hotfix for Windows Server 2000 and Windows Server 2003 and has persisted in subsequent releases (which may help to explain why the process itself is a little complicated).The mechanism that controls this behavior is located in the dsHeuristics attribute of the CN=Directory Service,CN=Windows NT,CN=Services,<FOREST ROOT DN> object. The value of this attribute is a Unicode string in which each character represents a single forest-wide configuration setting. The relevant character to

this discussion is the 16th character, which, if it exists, represents the “dwAdminSDExMask”. Its value is presented as a hexadecimal character, but it’s a little easier to understand the behavior if we look at its binary equivalent. Each bit in the binary representation signifies a specific group. If the bit associated with any specific group is not 0, that group will be excluded from the protected object set.



3. The default protected object set includes two user objects: Administrator and krbtgt. These objects are explicitly protected, but they won’t induce AdminCount updates on other objects because you can’t make a user, group or computer a member of a user object.

4. The default protected object set includes two groups that do not confer their protected status on their members: Domain Controllers and Read-Only Domain Controllers. While the DCPROMO command will add the associated computer object to the appropriate domain controller group when promoting a host as a domain controller, it won’t result in the object being added to the protected object set.

5. The default protected object set’s members are identified using their objectSID. While many of the default protected object set’s members are configured with systemFlag values that prevent them from being renamed or moved, Active Directory relies on the well-known objectSID values of these objects to ensure that they are consistently identified correctly.

In short, there are a number of behaviors that complicate the process of understanding which groups in any particular domain will cause an AdminCount update on members.

## AdminSDHolder and SDPROP

Now let’s talk about the mechanism that controls the AdminCount attribute’s behavior.

### The AdminSDHolder Object

Each AD object has a security descriptor that contains information about the object’s ownership, its primary group, the users and groups that are allowed or denied permission to access the object (the Discretionary Access Control List [DACL]), and the auditable events that will generate a record in the security event log (the System Access Control List [SACL]). They also contain control bits that can modify the security descriptor's behavior.

To help secure objects known to possess elevated administrative privileges, Active Directory applies a strict security descriptor called the Authoritative Security Descriptor to every member of a domain's protected object set. The Authoritative Security Descriptor is defined in the AdminSDHolder object located in the System container of every Active Directory domain's default naming context (e.g., CN=AdminSDHolder,CN=System, <DOMAIN DN>).

The Authoritative Security Descriptor is designed to secure protected objects by:

- **Limiting the object's DACL to a restricted set of Access Control Entries (ACEs).** These ACEs constrain the ability to modify the object to the NT AUTHORITYSystem account and members of the Administrators, Domain Admins and Enterprise Admins groups.
- **Enabling the object's SE_DACL_PROTECTED security descriptor control bit.** This disables security inheritance and prevents the protected object's DACL from being modified by inheritable ACEs possessed by any of the protected object's parent objects.
- **Restricting ownership of the protected object to the Domain Admins group.** This limits the possibility of a non-privileged account taking ownership of a protected object, which would grant the non-privileged account modify permissions on the protected object and allow the non-privileged account to grant itself full control of the protected object.

To be effective, this behavior requires that Active Directory can ensure that the security descriptor of each member of the protected object set matches the Authoritative Security Descriptor, , and will continue to match it. It also requires that Active Directory can ensure that the Authoritative Security Descriptor is applied to objects when they become a member (either directly or transitively) of a member of the default protected object set.

## The Security Descriptor Propagator (SDPROP) Task

Active Directory addresses these needs with a task called the Security Descriptor Propagator (SDPROP). This task is executed by the Local Security Authority Subsystem Service on domain controllers that own the PDC Emulator FSMO role every 60 minutes, by default. This period serves to limit the length of time that a modification of a highly privileged object's security descriptor, whether malicious or accidental, might persist, while acknowledging the relatively computationally expensive nature of SDPROP execution.

(The default SDPROP execution period can be overridden at the domain level by adding the AdminSDProtectFrequency entry to the HKEY_LOCAL_MACHINESYSTEMCurrentControlSetServicesNTDSParameters registry key on the domain controller that owns the domain's PDC Emulator FSMO role.

AdminSDProtectFrequency will accept values ranging from 60 to 7200 seconds. However, overriding the default execution period is not recommended. Increasing the execution frequency can be expected to result in a substantial increase in CPU utilization on the PDC Emulator, and decreasing the execution frequency lengthens the time during which a modification to the security descriptor of a protected object can persist.)

When SDPROP is executed, it identifies the domain's default protected object set and then recursively moves through the membership tree of each of those objects to identify the complete membership of the protected object set. SDPROP compares the security descriptor of each object to the Authoritative Security Descriptor; if they do not match, the object's security descriptor is replaced with the Authoritative Security Descriptor and the value of the object's AdminCount attribute is set to 1.

The fact that a dedicated task was created specifically to perform these checks may seem unnecessary, but Active Directory's security descriptor propagation isn't really set up to address this behavior. While updates to an object's security descriptor or its distinguished name results in Active Directory propagating any associated inheritable Access Control List changes almost immediately (unless the distinguished name change results in the object's new parent being the domain's Deleted Objects container), group membership changes don't initiate that process. This means that adding an object to a protected group won't trigger the process and can't be used to apply the Authoritative Security Descriptor to the objects affected by the group membership update.

## Limitations of AdminCount

Let's review the key limitations of the AdminCount attribute and the misunderstandings they can potentially create.

### SDPROP executes on a schedule.

As mentioned above, by default the SDPROP task executes only once an hour. Consequently, it can take up to an hour for an account that has been added to a protected group to be identified by SDPROP as a member of the protected object set. This means that if an object becomes a member of a protected object (either directly or transitively) and that membership is removed before SDPROP executes, the object will not be identified as a protected object and the value of the object's AdminCount attribute will remain unchanged.

### SDPROP updates the AdminCount attribute when it modifies an object's security descriptor.

What's the big deal here? Well, SDPROP doesn't pay any attention to the value of an object's AdminCount attribute. If SDPROP updates a protected object's security descriptor, it will set the value of the AdminCount attribute to 1. If the protected object's security descriptor matches the Authoritative Security Descriptor, SDPROP will leave the protected object’s AdminCount attribute unchanged, regardless of its value. Consequently, changing or removing the value of a protected object’s AdminCount attribute can effectively hide protected objects from simple reporting scans.

### SDPROP only looks at active members of the protected object set.

SDPROP scans start with the members of the default protected object set and iterates through their memberships. However, highly privileged objects that are not members of the default protected object set are ignored by SDPROP and the value of their AdminCount attribute will remain <NOT SET>. As a result, the AdminCount attribute can’t be relied upon to identify all of the highly privileged objects in a domain.

Moreover, once a protected object is removed from the groups it inherited its protected status from, it will subsequently be ignored by SDPROP while continuing to look exactly like a protected object. This is because Active Directory lacks a mechanism to undo the changes made by SDPROP when an object became a member of the protected object set; its AdminCount attribute remains set to 1 (or whatever its value was before its group membership changed). More importantly, the object’s security descriptor will also remain unchanged and will continue to block security inheritance from the object’s parents.

### SDPROP doesn’t distinguish between group types.

There are two types of Active Directory groups: security groups and distribution groups. A security group can pass privileges on to its members, while a distribution group cannot. While it may seem odd to protect objects that are transitive members of a default protected object through a distribution group (and therefore aren’t inheriting any privileges from the default protected object), a group’s category can be changed. Since changing a group’s type changes its ability to confer privilege on its members, SDPROP ignores group categories entirely to prevent this behavior from being abused.

## Summary

At the end of the day, the AdminCount attribute is just a flag. In order to understand what that flag can tell you, you need to also understand what it isn’t able to tell you.

It might seem that you can get a list of objects that are being protected by SDPROP by running the following command:

```
Get-ADObject -LDAPFilter "(adminCount=1)"
```

However, what this command actually tells you is which objects have an AdminCount attribute with a value of 1.

The only reliable way to accurately identify protected objects is to do exactly what SDPROP does: Identify the domain’s default protected object set and identify the complete membership of each of those objects.

## How can Netwrix help?

Secure your Active Directory from end to end with the Netwrix Active Directory security solution. It will enable you to:

- Uncover security risks in Active Directory and prioritize your mitigation efforts.
- Harden security configurations across your IT infrastructure.
- Promptly detect and contain even advanced threats, such as DCSync and Golden Ticket attacks.
- Respond to known threats instantly with automated response options.
- Minimize business disruptions with fast Active Directory recovery.

## FAQ

**What is the AdminCount attribute in Active Directory?**

The AdminCount attribute shows that an object's ACLs was modified to a more secure setting by the system because it belonged to one of the administrative groups.

**When I remove a user from a protected group, why is AdminCount not changed to 0 or "not set"?**

Early in the development of Windows 2000, a poll of users revealed that they preferred destroying a user account once its high-privilege privileges were withdrawn, since the account may have constructed explicit backdoors prior to having its rights removed. Since it is believed that the account will be terminated or disabled, the DC does not remove the AdminCount attribute.

Kevin Joyce
Senior Technical Product Manager at Netwrix. Kevin is passionate about cyber-security and holds a Bachelor of Science degree in Digital Forensics from Bloomsburg University of Pennsylvania.