Hiding Active Directory Objects and Attributes

https://www.itprotoday.com/active-directory/hiding-active-directory-objects-and-attributes

Use normal permissions to grant or remove access to AD data

Active Directory (AD) allows delegated administration of users, groups, or computers, according to any security principal. But default AD permissions can complicate the task of making specific data visible to only those users who need to see it. AD data-hiding options can be based on typical AD permissions, a special AD permission feature called List Mode, or a little-known option called the confidentiality bit.

Part 1 of this multi-part series, "Hiding Data in Active Directory," explains the challenge of hiding data in AD and goes into details such as AD's default security settings and property set definitions. In this article, I'll describe how to use normal permissions to hide objects and attributes. Future articles will cover enabling List Mode in the AD forest and adjusting the default security of objects in AD.

Using Normal Permissions

You can use the normal capabilities of AD permissions to ensure that users can view and access only their entitled objects and attributes. The principle is straightforward: If a user isn't authorized to view an object or attribute, AD won't display the information to the user.

To view the objects in an organizational unit (OU), a user must have at least the List Contents permission on the OU. As I described in detail in the first part of this series, various security principals, including Authenticated Users, are granted the read permission on any newly created OU. These principals are also granted the List Contents permission on an OU, although AD doesn't enforce this permission by default. The default read permission is a combination of permissions, including List Contents, as Figure 1 shows.

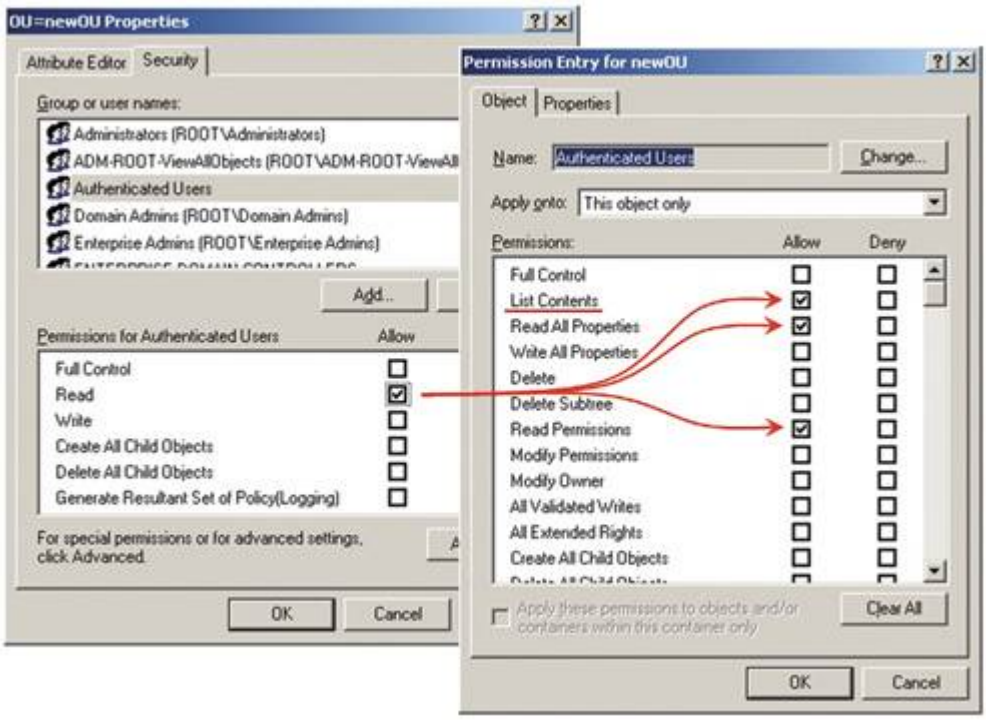

Figure 1: Set of permissions granted the read permission

By default, the List Contents permission grants sufficient rights to list all objects in an OU. This doesn't mean that a user can automatically view the attributes of all objects in an OU; to do this, the user requires read permissions on the respective attributes of the objects within the OU. But the administrator has no control over which *objects* are displayed. You can change this behavior by enabling the List Mode access in AD, which I describe in the next section.

If the List Contents permission for Authenticated Users is removed from an OU, then a query for all user objects in the AD forest won't return the objects in that OU. The same is true when using the Windows object picker to list objects. The object picker is typically used to add users to groups or to select a security principal when granting permissions to resources. By default, the object picker displays all objects in the containers to which the current user has List Contents permission (and which are appropriate for the task the user is performing).

Removing the List Contents permission for Authenticated Users from an OU effectively hides all the objects in that OU for all users in AD, unless the users are members of other groups that are specifically granted the appropriate

permissions. Explicitly denying the List Contents permission to a group has the same effect. But to keep things simple and comprehensible for any systems administrator, it's better to work with positive permissions (allow) instead of negative permissions (deny); more about that in a bit.

You can easily remove the permission for a single OU via the Microsoft Management Console (MMC) *Active Directory Users and Computers* snap-in or the ADSI Edit GUI, by using the Security tab of an object's property. To do so from the command line, you can use the Dsacls utility that comes with the OS. (Dsacls is part of the Windows Server support tools. See " Dsacls Syntax" for more information.)

This command is unable to remove single permissions directly. To do so, you must first remove all permissions for the respective security principal and then assign new ones. (Note that although most companies are still happily using Dsacls, Windows Server 2008 R2 and later provide new AD cmdlets for Windows PowerShell, as I discuss in the sidebar "Managing Active Directory Permissions via Windows PowerShell." These cmdlets offer new options for managing the security of AD objects.)

Removing all permissions for a security principal from the command line shouldn't be taken lightly, especially if you don't know exactly which permissions the security principal has on the object. You should first generate a report of the ACLs of an object so that you can more easily reset them if needed. A couple of tools are available to do this; all of them have their advantages and disadvantages. The Microsoft Dsrevoke tool achieves good results by allowing you to report on the ACLs for a specific security principal. However, the tool can't do so for well-known security principals such as Authenticated Users or for built-in ones such as Administrators. Furthermore, this tool is supported only on Windows Server 2003 and earlier. Dsacls always list the permissions for all security principals, as Figure 2 shows, yet that's better than not listing any permissions at all.

Figure 2: Partial result of Dsacls listing ACLs on an OU

Removing Permissions

To remove the List Contents permissions for Authenticated Users on a new OU, use the following command:

```
dsacls  /R
```

In our example, this command would look like this:

```
dsacls "OU=newOU,DC=root,DC=net" /R "Authenticated Users"
```

Afterwards, you must reset all the default object permissions (i.e., read permissions, Read All Properties, List Object) except for the one that you removed. To do so, use the following command:

```
dsacls  /G :RCRPLO
```

In our example, the command would look like this:

```
dsacls "OU=newOU,DC=root,DC=net" /G "Authenticated Users":RCRPLO
```

To combine both commands, you'd typically create a batch file and use variables. If you want to run the commands via a single command-line statement, you can use the && command:

```
set DN="OU=newOU,DC=root,DC=net"&& set SP="Authenticated Users"

&& dsacls %DN% /R %SP%&& DSACLS %DN% /G %SP%:RCRPLO
```

(Note that any character preceding the && command is part of the previous command. This command is especially tricky when used after setting a variable via *set*, when a trailing space is added to the variable.)

The outcome of these changes is that, if the requesting user doesn't have or is denied read access to an attribute, AD doesn't return any data that's stored within this attribute. The GUI that comes with AD either displays an empty field in

the MMC *Active Directory Users and Computers* snap-in or displays for the attribute value when using ADSI Edit, as Figure 3 shows.

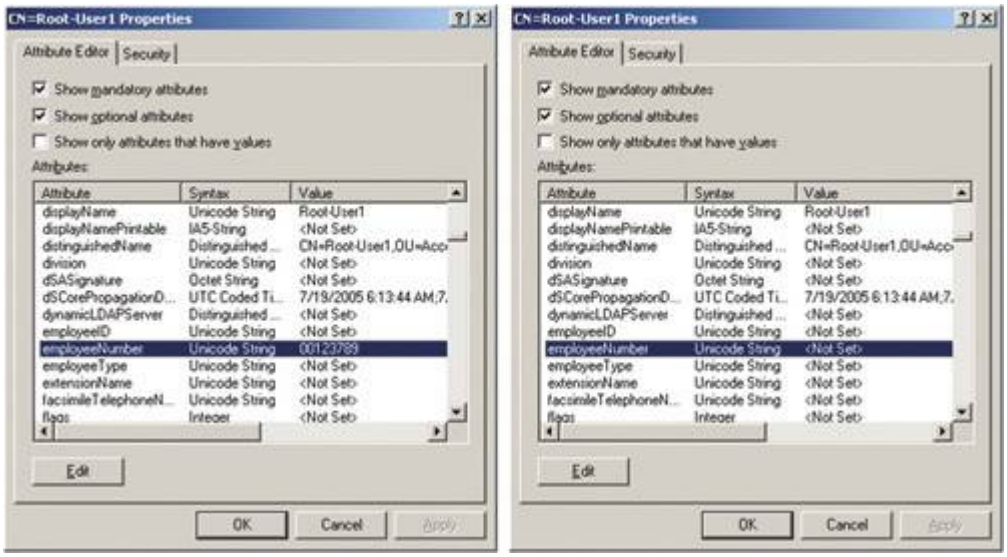

Figure 3: Content of attributes with read access (left) and without read access (right), using ADSI Edit

The special challenge of controlling read access at the attribute level in AD is a consequence of the grouping of attributes in permission property sets and the vast amount of explicit default permissions that are assigned to new objects in AD.

What if, for example, you want to restrict access to the homePhone attribute to members of the HR department? Looking back at " Hiding Data in Active Directory," you can see that the homePhone attribute is one of the 47 attributes that belong to the Personal Information property set and that Authenticated Users are granted read permissions to this property set for any new user object in AD. Removing the read permission from a single attribute isn't easy if the attribute is a member of a property set that has been granted read permissions on the object. Instead, the read permission for the whole property set needs to be removed and replaced with a separate read permission for all properties *except* the one that isn't required. To remove the permission for Authenticated Users to read a user's homePhone attribute, you'd need to replace the read permission for the Personal Information property set with 46 separate read permissions for all the other attributes in the property set. Another option is to adjust the attributes that belong to a property set, but for now let's assume that you don't want to adjust the property set.

Even though you want to generally avoid using deny permissions, in this scenario working with a deny read permission on the homePhone attribute of the respective user objects for a security group that contains all users except the HR department is the efficient option. Let's call this group non-HR-users. However, creating such a group and keeping it up-to-date is a challenge on its own. Another challenge is that you can't simply set the required deny permission at the OU level or the domain level and allow the respective user objects to inherit it. As described previously, the explicit allow permissions that are granted to Authenticated Users directly on the user object will override the inherited permission from the parent objects. So you must add an explicit deny read permission for the non-HR-users group to all user objects in the AD forest.

More Dsacls

Presuming that you don't want to set the explicit deny permission to all objects by clicking through the GUI for hours, you can achieve the same task fairly easily by using the Dsacls command-line tool. Because you must set explicit permissions on each user object to achieve your goal, you need to run separate Dsacls commands -- each with the distinguished name (DN) of the user object for which you need to set the deny permission. There are various ways to retrieve the list of user object DNs: You can use the Dsquery command-line tool that comes with Windows 2003; you can also use this tool to query a Windows 2000 domain controller (DC), if you still have one. (You can find more information about using Dsquery on the Microsoft website.)

To set the permission to deny read access of the homePhone attribute on a single user object, you can use this command:

```
dsacls  /D :RP;homePhone
```

For our example, the command would look like this:

```
dsacls "CN=Doe\, John,OU=newOU,DC=root,DC=net" /D root\

non-HR-users:RP;homePhone
```

For multiple objects, you must first create a list of DNs and save them to a file. You can use Dsquery:

```
dsquery user  > queryresult.txt
```

For our example, the command would look like this:

```
dsquery user OU=newOU,DC=root,DC=net > queryresult.txt
```

Next, after ensuring that the query results meet your expectations, you must perform a *for* loop to execute the Dsacls command against all objects in the file:

```
for /f "delims=" %I in (queryresult.txt) do DSACLS "%~I" /D root\

non-HR-users:RP;homePhone
```

This command sets the desired permission for all objects that are listed in the queryresult.txt file.

In summary, keep the following in mind when using the default AD permissions to hide objects and attributes:

- You must understand the default and current permissions that are granted to an object and its parent container. If a user isn't granted the List Contents permission on a container object, then the objects in the container aren't visible.
- You must understand how attributes are granted permissions by property sets. An attribute isn't visible if a user doesn't have or is denied read access to it.
- If permission for an attribute must be denied and that attribute is also part of a property set that's explicitly granted the permission, then the deny permission for the attribute must be explicitly set for all objects on which it's supposed to take effect.

Know the Possibilities

Using the normal AD permissions to hide data in AD is certainly possible. Two more fairly straightforward options for hiding data are available: List Object mode and changing the default security descriptor of objects in AD. I'll describe these methods in the next articles in this series, before I finish off with a few more advanced topics: handling built-in property sets, handling AD attribute permissions with the confidentiality bit, and with the filtered attribute set (FAS).