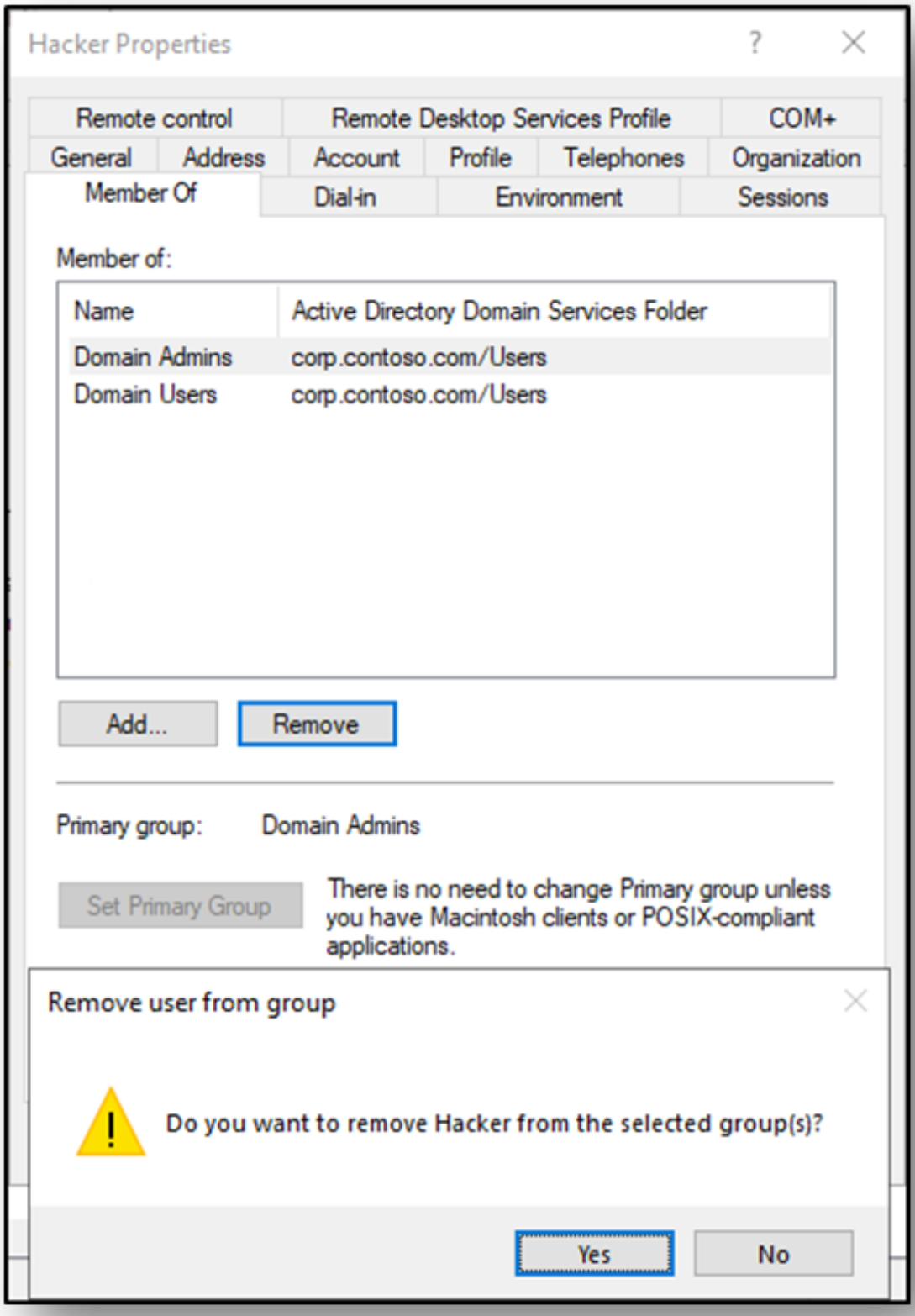
Primary Group Behavior, Reporting and Exploitation

hub.trimarcsecurity.com/post/primary-group-behavior-reporting-and-exploitation

Brandon Colley May 23, 2023



May 23, 2023

Updated: May 25, 2023

Introduction

If you've administered Active Directory (AD) for any significant time, chances are you've come across the *primaryGroupID* attribute. Originally developed as a method for AD to support POSIX-compliant applications, the attribute has been better known by a different name: An Attack Vector. Commonly referenced in a DCShadow attack, an adversary can set a user's *primaryGroupID* to 512 (Domain Admins) and effectively become a member of that group.

One of the post important concepts to grasp here is this: AD group membership is the **combination** of "members" in a group as well as any user with the primary group set.

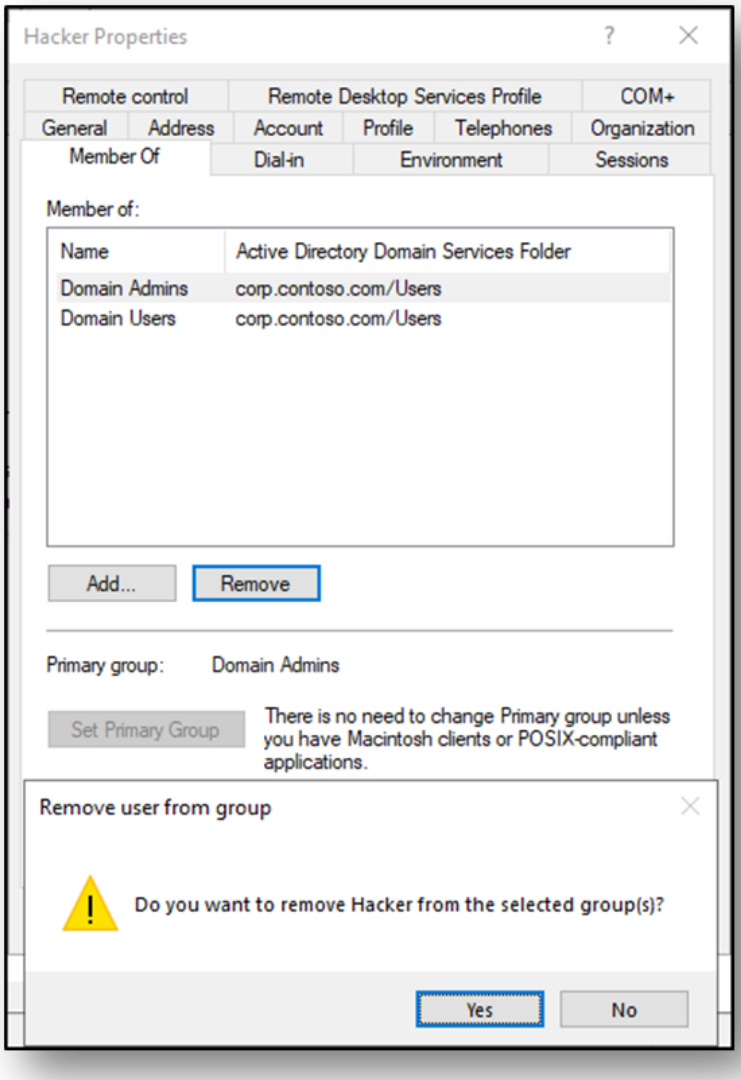
Deep down, the two attributes are **separate,** AD just attempts to combine the two for convenience. However, this convenience is not universal and various AD tools report on group membership differently. This inconsistency in reporting can cause issues with monitoring & alerting on group changes and group enumeration. (This is not a new revelation; we can date this conversation all the way back to 2005.)

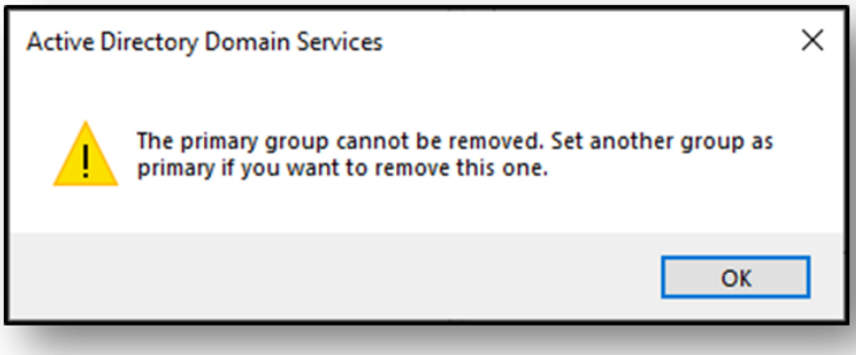
Finally, since the *primaryGroupID* is an independent attribute, it is possible for an attacker to modify the Discretionary Access Control List (DACL), effectively hiding group membership from all users in the forest, even Domain Admins.

Exploitation

When I first investigated primary groups, I assumed that an attacker could use this attribute to sneakily maintain the privileges associated with the group without appearing in the group membership. This turned out to only be partially true.

When attempting to remove group membership for a primary group using Active Directory Users and Computers (ADUC), the Administrator is met with this message:

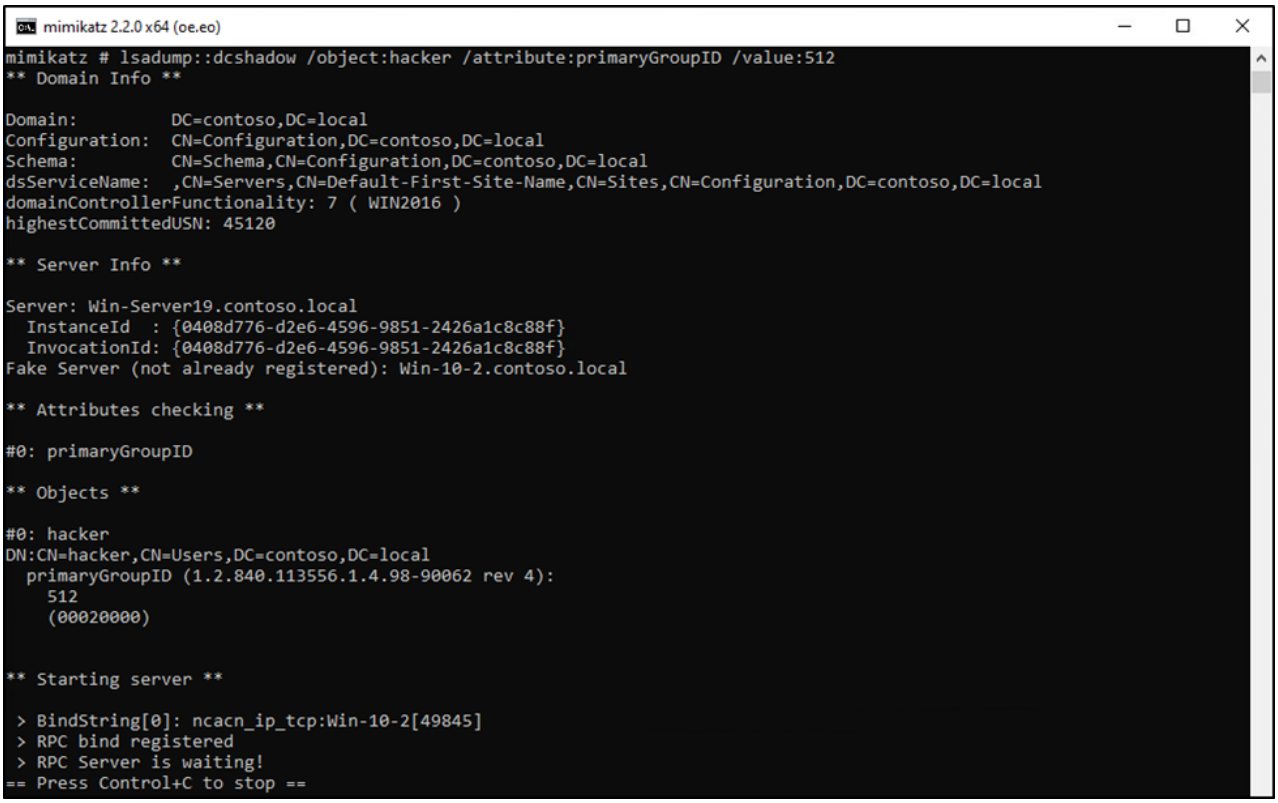




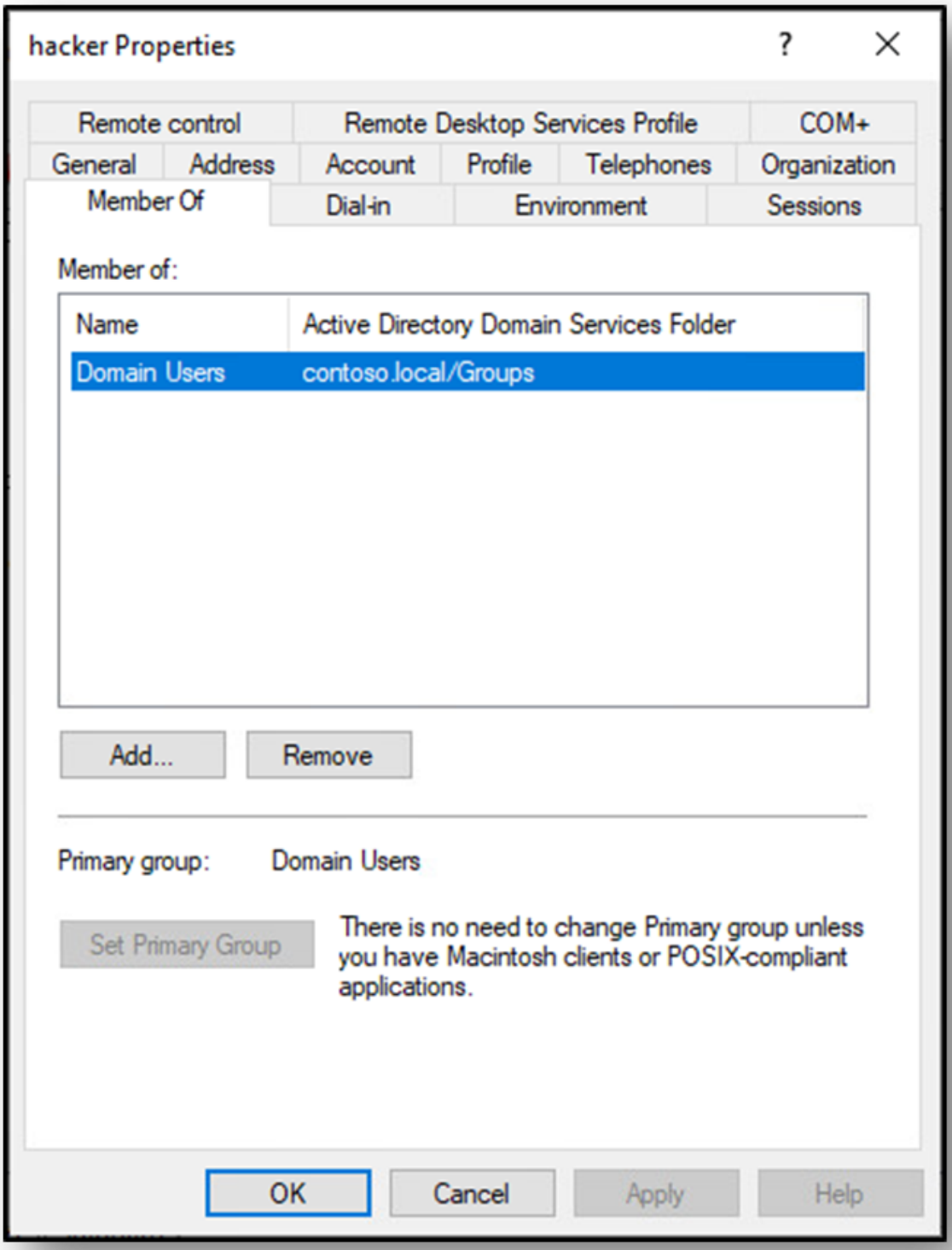
Even when using the PowerShell cmdlet Remove-ADGroupMember you get a similar error message:

```
Remove-ADGroupMember : The user cannot be removed from a group because the group is currently the
user's primary group
```

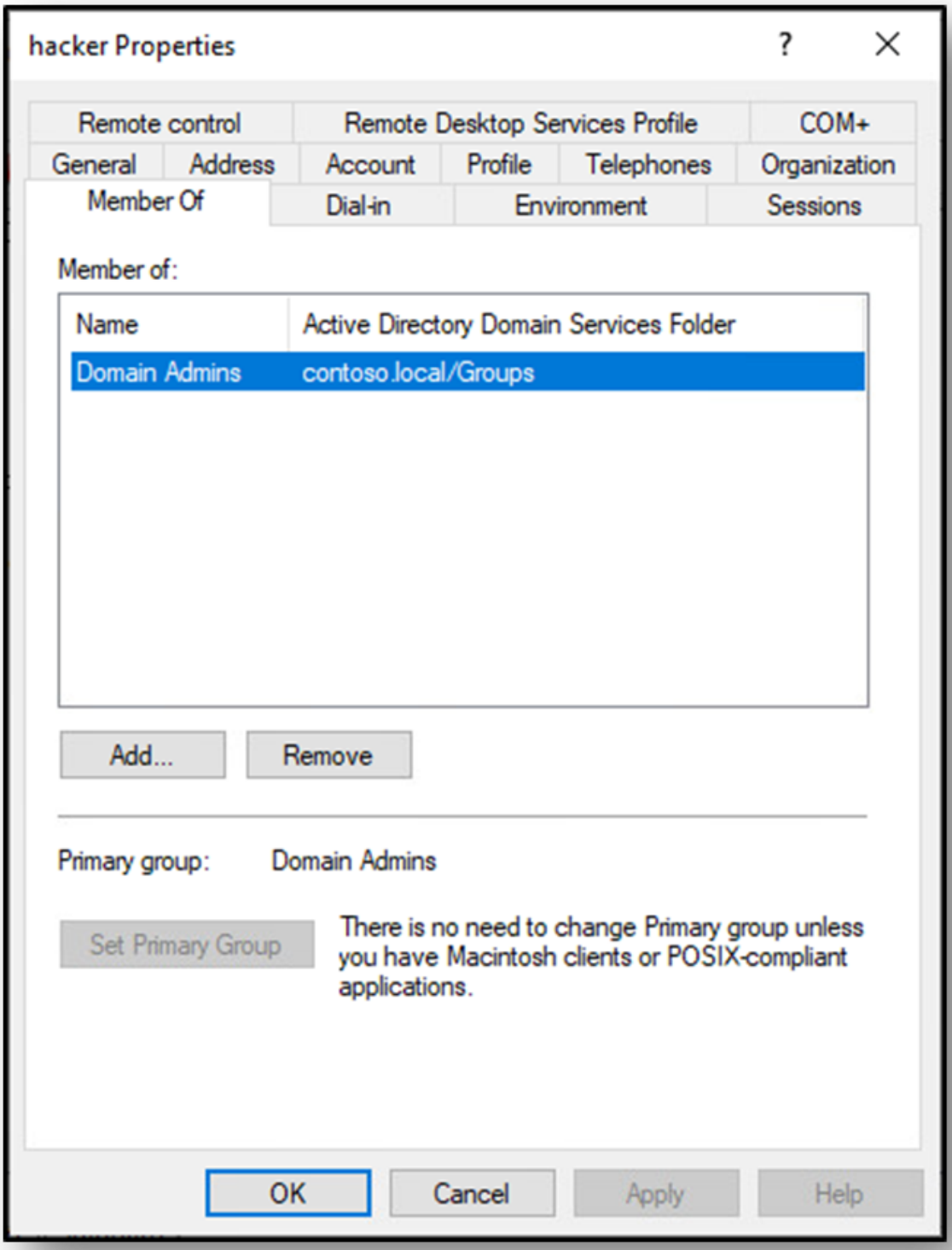
The next logical step was to avoid traditional tools and attack the AD database directly. Performing a DCShadow attack would bypass LSASS and would allow direct modification of the primaryGroupID attribute. So, loading up mimikatz, I performed an attack to set *primaryGroupID* to 512 outlined in the screenshot below:



Prior to these changes, the **Hacker** user account was only a member of the Domain Users group. By default, Domain Users is also configured as **Hacker's** Primary Group.



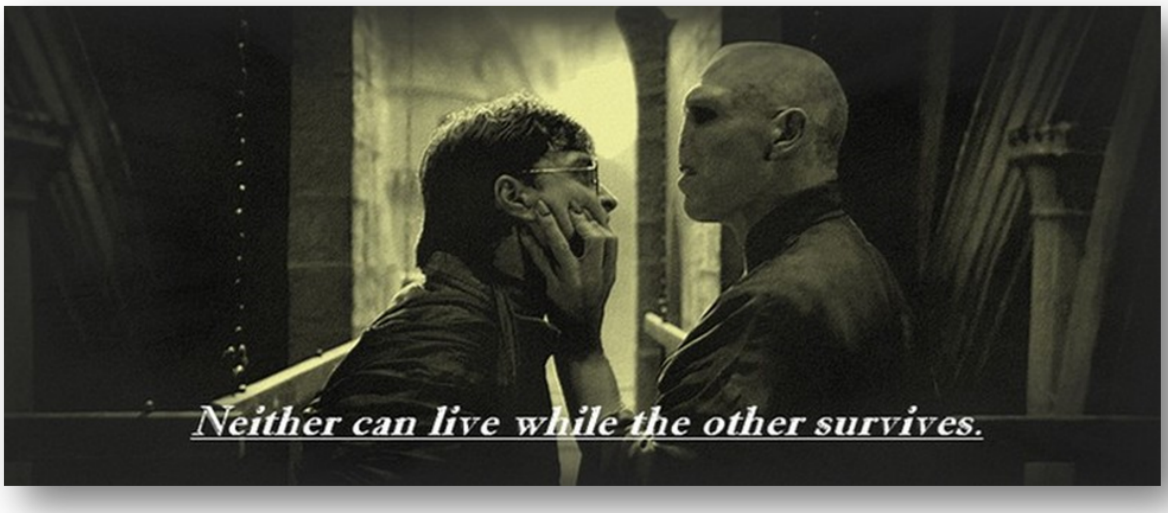
After the attack, the **Hacker** user account Primary Group has been changed to Domain Admins but it has lost its membership to the Domain Users group.



A DCShadow attack replaces the Primary Group but it also strips the corresponding group membership for the previously set Primary group. This attack is not limited to only Domain Admins group, using any group will result in the same outcome. Using the DSInternals function *Set-ADDBPrimaryGroup* yielded the same results:

```
PS C:\Users\Administrator> Set-ADDBPrimaryGroup -SamAccountName hacker
```

In testing, I have determined it is not possible to decouple these two attributes. Changing the Primary Group ID on a user will always coincide with a group addition (see above). Further proof of this is the removal of membership from the Domain Users group. To reiterate, changing a user's Primary Group ID also removes their membership from the original Primary Group.



While these two attacks can be "sneaky" (by directly updating the AD Database), existence of a new Domain Admin would eventually be discovered unless you are using the wrong commands to report on group membership. 😈

Behavior and Misconceptions

Further research into the *primaryGroupID* revealed that group membership reports differently depending on the method used to perform the query. Regardless of how the *primaryGroupID* is set, using the PowerShell cmdlets "*Get-ADGroup*" and "*Get-ADGroupMember*" return different results:

```
PS C:\Users\Administrator> Get-ADGroupMember "Domain Admins"


distinguishedName : CN=Administrator,CN=Users,DC=contoso,DC=local
name              : Administrator
objectClass       : user
objectGUID        : c3287246-4045-4d15-a8c1-f8ab4468530a
SamAccountName    : Administrator
SID               : S-1-5-21-4268493626-3948267598-2515514451-500

distinguishedName : CN=Admin User1,CN=Users,DC=contoso,DC=local
name              : Admin User1
objectClass       : user
objectGUID        : 16058a33-5b4a-4d2b-89c5-c913a9629169
SamAccountName    : admin_user1
SID               : S-1-5-21-4268493626-3948267598-2515514451-1103

distinguishedName : CN=Service Account1,CN=Users,DC=contoso,DC=local
name              : Service Account1
objectClass       : user
objectGUID        : 696b3841-0158-466d-a6c5-0ad08545dfd9
SamAccountName    : Service_Act1
SID               : S-1-5-21-4268493626-3948267598-2515514451-1105

distinguishedName : CN=test,CN=Users,DC=contoso,DC=local
name              : test
objectClass       : user
objectGUID        : 91ac6cef-78df-4261-a848-a8793b603ee3
SamAccountName    : test
SID               : S-1-5-21-4268493626-3948267598-2515514451-1110

distinguishedName : CN=test,CN=Users,DC=contoso,DC=local
name              : test
objectClass       : user
objectGUID        : 91ac6cef-78df-4261-a848-a8793b603ee3
SamAccountName    : test
SID               : S-1-5-21-4268493626-3948267598-2515514451-1110

distinguishedName : CN=hacker,CN=Users,DC=contoso,DC=local
name              : hacker
objectClass       : user
objectGUID        : 7808f040-42e4-4fa6-8397-1dcb7dc80944
SamAccountName    : hacker
SID               : S-1-5-21-4268493626-3948267598-2515514451-1112
```

```
PS C:\Users\Administrator> Get-ADGroup "Domain Admins" -Properties members | select -ExpandProperty members
CN=test,CN=Users,DC=contoso,DC=local
CN=Service Account1,CN=Users,DC=contoso,DC=local
CN=Admin User1,CN=Users,DC=contoso,DC=local
CN=Administrator,CN=Users,DC=contoso,DC=local
PS C:\Users\Administrator>
```

The **Hacker** user account has been added to the “Domain Admins” group and set as *primaryGroupID*. *Get-ADGroup* does not list **Hacker** as a member; *Get-ADGroupMember* does. Furthermore, when using *Get-ADUser* the *memberOf* attribute is empty yet, running

net group will list **Hacker** as a member.

```
PS C:\Users\Administrator> Get-ADUser hacker -Properties memberof


DistinguishedName : CN=hacker,CN=Users,DC=contoso,DC=local
Enabled           : True
GivenName         : hacker
MemberOf          : {}
Name              : hacker
ObjectClass       : user
ObjectGUID        : 7808f040-42e4-4fa6-8397-1dcb7dc80944
SamAccountName    : hacker
SID               : S-1-5-21-4268493626-3948267598-2515514451-1112
Surname           :
UserPrincipalName : hacker@contoso.local



PS C:\Users\Administrator> net group "Domain Admins"
Group name     Domain Admins
Comment        Designated administrators of the domain

Members

-------------------------------------------------------------------------------
admin_user1              Administrator            hacker
Service_Act1             test                     test
The command completed successfully.
```

ADUC and Admin Center will both list all members including those with primaryGroupID set. However, if digging directly into ADSI Edit, the member attribute for the Domain Admins group does not include any user with the primaryGroupID configured.

Here is a recap of which tools return *primaryGroupID* members and which don't:

Returns members with primaryGroupID	Does not return members with primaryGroupID
Net group "Domain Admins"	ADSI (Looking at Group)
ADUC / Admin Center	Get-adgroup "Domain Admins" -properties member(s)
Get-adgroupmember "Domain Admins"	Get-aduser <user> -properties memberOf

Once recursive queries and nested groups are introduced into the equation, nothing can be trusted. In this example, The Domain Admins group contains 2 users and 1 *nestedDAs* group. The *nestedDAs* group contains 2 additional users. The *hidden* user account has its primary group set to *nestedDAs*. Performing a recursive query using *Get-ADGroupMember* or by using an LDAP filter does not return *hidden* as a member.

```
PS C:\Users\Administrator> Get-ADGroupMember "Domain Admins" | Format-Table name, objectClass

name          objectClass
----          -----------
Administrator user
Admin User1   user
nestedDAs     group


PS C:\Users\Administrator> Get-ADGroupMember "nestedDAs" | Format-Table name, objectClass

name   objectClass
----   -----------
user 2 user
hidden user


PS C:\Users\Administrator> Get-ADGroupMember "Domain Admins" -Recursive | Format-Table name, objectClass

name          objectClass
----          -----------
Administrator user
Admin User1   user
user 2        user


PS C:\Users\Administrator> Get-ADObject -LDAPFilter "(memberof:1.2.840.113556.1.4.1941:=CN=Domain Admins,OU=Groups,DC=co
ntoso,DC=local)"

DistinguishedName                           Name          ObjectClass ObjectGUID
-----------------                           ----          ----------- ----------
CN=Administrator,CN=Users,DC=contoso,DC=local Administrator user        c3287246-4045-4d15-a8c1-f8ab4468530a
CN=Admin User1,CN=Users,DC=contoso,DC=local   Admin User1   user        16058a33-5b4a-4d2b-89c5-c913a9629169
CN=nestedDAs,CN=Users,DC=contoso,DC=local     nestedDAs     group       42f67eb3-102b-4876-bbd8-9024ae19288d
CN=user 2,CN=Users,DC=contoso,DC=local        user 2        user        69cc059a-3483-4635-a748-4bee9d6d78e8
```

Monitor Your Monitoring

Trimarc asks all clients two very important questions.

1. Are you monitoring privileged groups for membership changes?

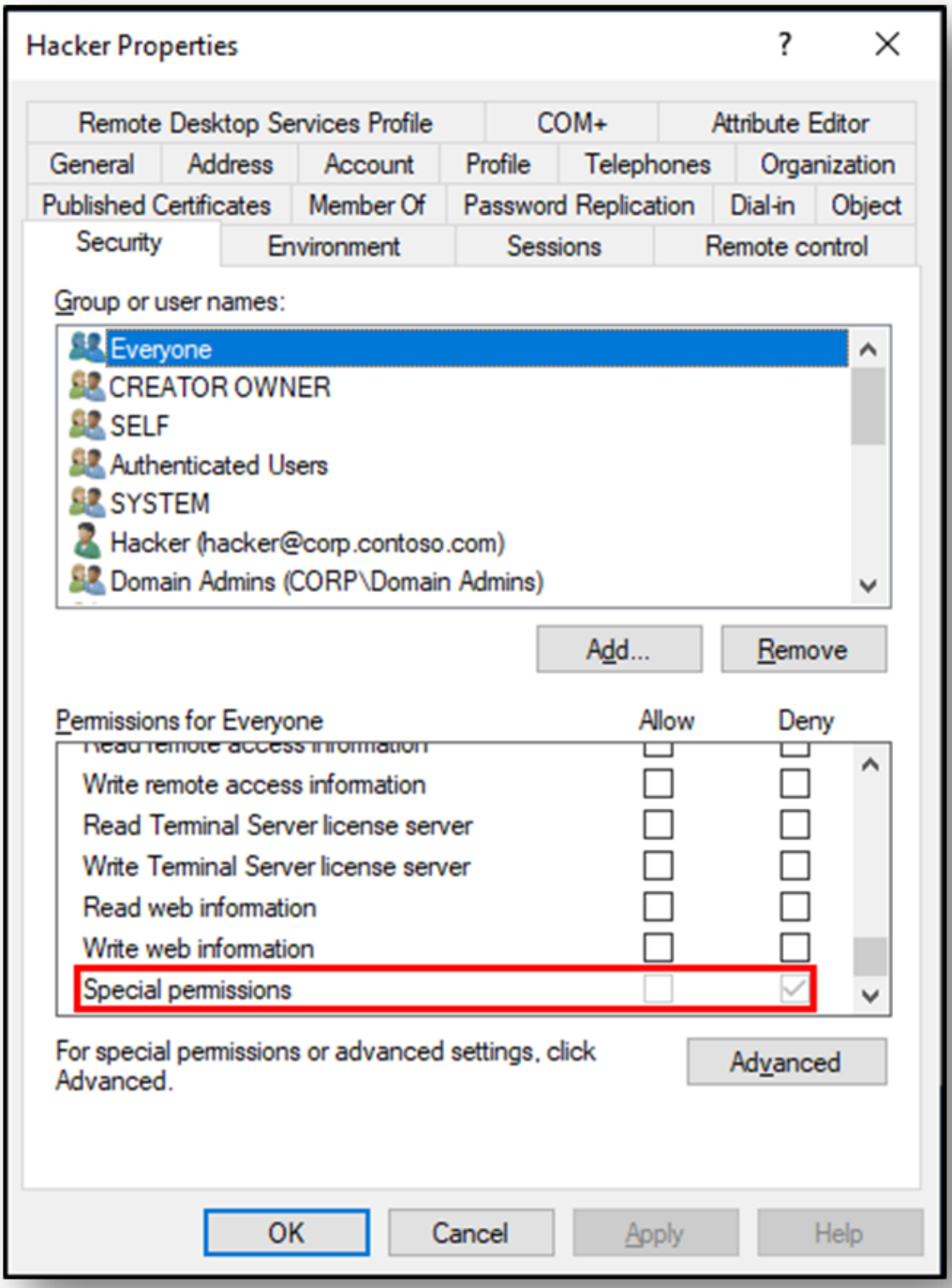
2. Are you monitoring privileged groups for enumeration?

While these questions seem straight forward, how does *primaryGroupID* impact these questions? If directly looking at the group objects, is it possible that user objects are slipping through the cracks? Setting up test scenarios may be the best way to truly identify if this blind spot exists in your environment.

In short, check your scripts and tools. Confirm that any reporting or monitoring being used is properly reporting on members with *primaryGroupID* configured. Failure to do so may result in inconsistent reporting or could mean you have some unknown Administrators.

Real Exploitation

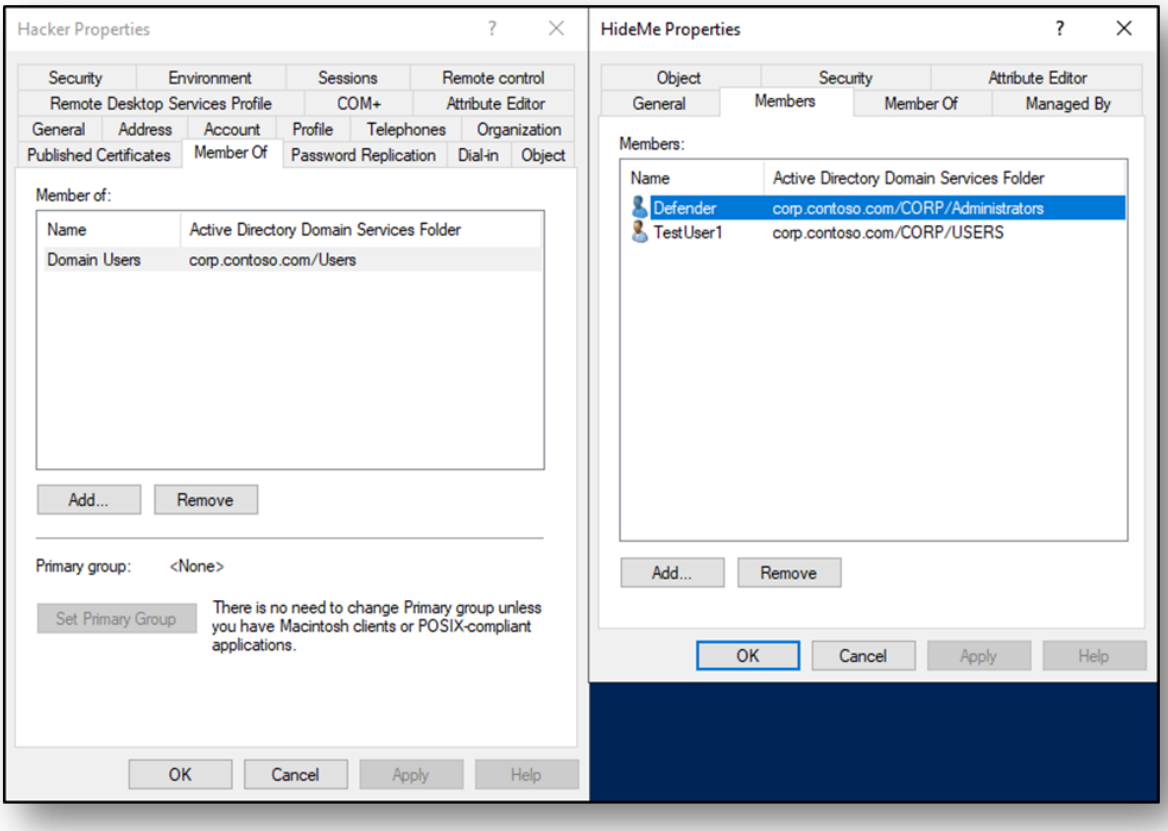
Now that we have an understanding of how the *primaryGroupID* behaves, it's possible to comprehend a real attack strategy. Yuval Gordon wrote a paper that explains a method of abusing the *primaryGroupID* by modifying the DACL on a user object. In the context of our examples, this DACL would be applied to the **Hacker** user account. It effectively sets a deny permission for the "Everyone" group on the ability to read the *primaryGroupID* attribute.



This technique only works for objects not protected by SDProp. Membership in any of these groups periodically have permissions restamped and would remove the deny DACL rights.

Once a user has these permissions configured any (unprotected) group set as the Primary Group becomes invisible on both the user and group object. In this example, the user **Hacker** is a member of the HideMe group. Notice the Primary Group appears as

<None> and **Hacker** does not appear in the Members tab.



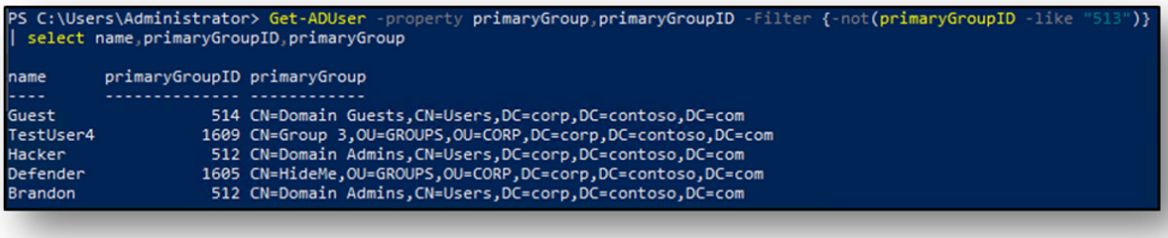
Even more impressive, it's illusive to all PowerShell commands. The only tested method that properly reports the membership is "*net group*":

```
PS C:\Windows\system32> Get-ADGroup hideme -Properties members | select -ExpandProperty members
CN=TestUser1,OU=USERS,OU=CORP,DC=corp,DC=contoso,DC=com
PS C:\Windows\system32> Get-ADGroupMember hideme | select name

name
----
TestUser1
Defender


PS C:\Windows\system32> Get-ADUser testuser1 -properties memberof, primarygroupid | fl memberof, primarygroupid


memberof       : {CN=HideMe,OU=GROUPS,OU=CORP,DC=corp,DC=contoso,DC=com,
                 CN=Marketing,OU=GROUPS,OU=CORP,DC=corp,DC=contoso,DC=com,
                 CN=TestUserGroup,OU=GROUPS,OU=CORP,DC=corp,DC=contoso,DC=com}
primarygroupid : 513



PS C:\Windows\system32> Get-ADUser defender -properties memberof, primarygroupid | fl memberof, primarygroupid


memberof       : {CN=Domain Users,CN=Users,DC=corp,DC=contoso,DC=com, CN=Domain
                 Admins,CN=Users,DC=corp,DC=contoso,DC=com}
primarygroupid : 1605



PS C:\Windows\system32> Get-ADUser hacker -properties memberof, primarygroupid | fl memberof, primarygroupid


memberof       : {CN=Domain Users,CN=Users,DC=corp,DC=contoso,DC=com}
primarygroupid :



PS C:\Windows\system32> net group hideme
Group name     HideMe
Comment

Members

-------------------------------------------------------------------------------
Defender                 hacker                   TestUser1
The command completed successfully.
```

Yuval offers the following script to query for users without a primaryGroupID, or more accurately, without a readable *primaryGroupID:*

```
PS C:\Windows\system32> Get-ADUser -Filter {-not(primaryGroupID -like "*")} | select name

name
----
Hacker
```

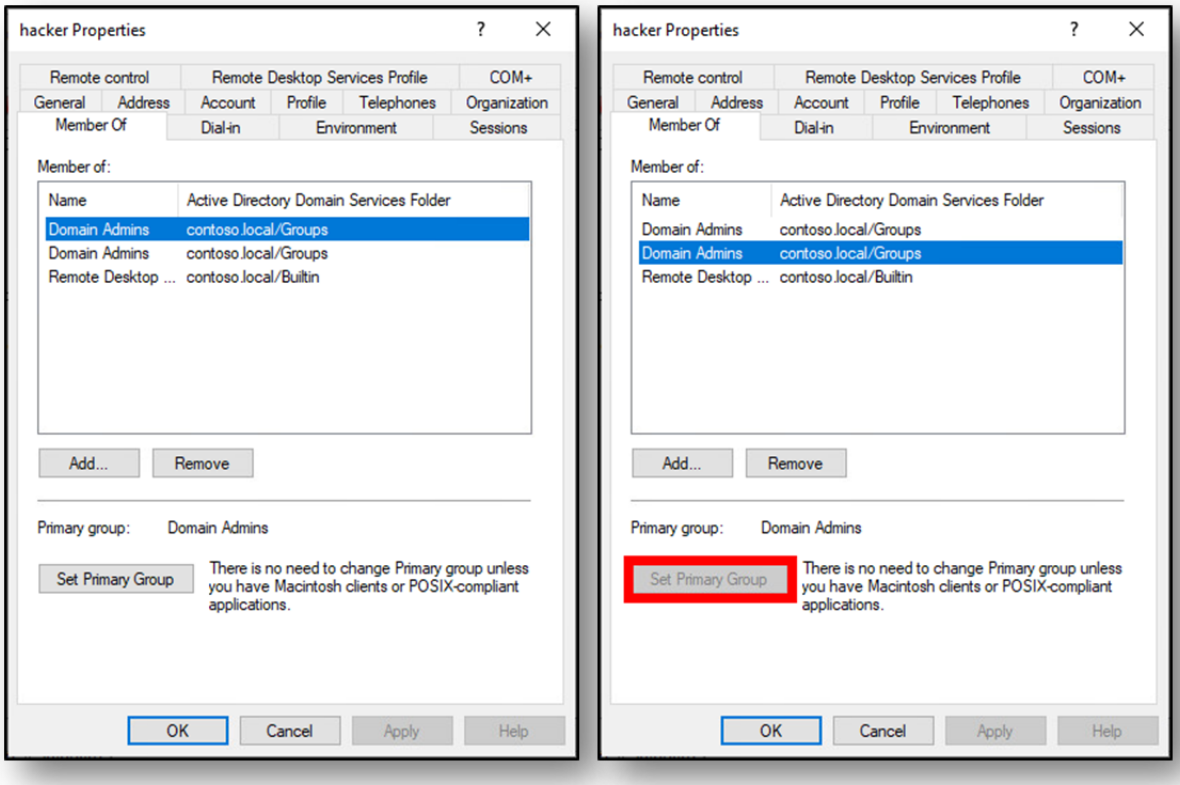
In addition, checking for any non-standard *primaryGroupID* may reveal any misconfigured users. A similar attack method is possible when setting the deny DACL on the group object. This will also hide group membership, but the Primary Group is still readable on the user object:



Unless there is a valid reason to retain a non-standard Primary Group, all users should be set to Domain Users (513).

Goofing Off

Hackers like to make things do something they aren't supposed to do. Just goofing off during testing, I found that it's possible to add a user to a group twice. Performing a DCShadow or DSInternals attack on an account that is already a member of Domain Admins, and has a Primary Group set as anything else results in a double-DA state. I don't know what this means but I hope it's the only time you see a hacker as a Domain Admin.



So, what's it all mean?

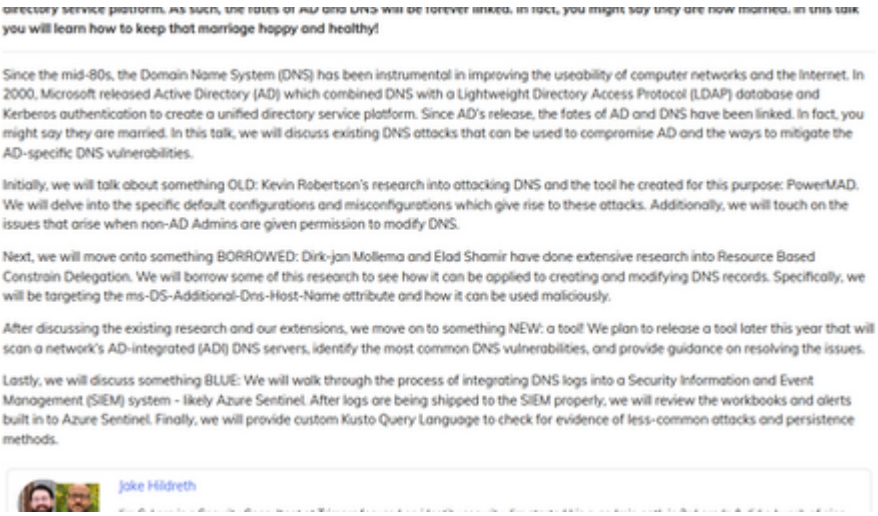
This adventure into the *primaryGroupID* touched on the use of attack tools and sneaky methods that may allow attackers to persist in your environment. The two key takeaways that everyone should understand are as follows:

1. You should be reporting on the current **primaryGroupID** of all user objects in AD. Scrutinize any user that may be granted privileged group membership in this way. Note that "privileged" does not always mean Domain Admin. **Any** non-standard users should be reset to Domain Users.

2. You should be reviewing and testing all scripts, tools, and monitoring that report on group membership. Membership using **primaryGroupIDs** may be missed depending on the PowerShell cmdlet or API used to report on group members. Logging and reporting on group changes may have a blind spot when it comes to the use of **primaryGroupIDs.**

Recent Posts

See All



Video: BSides Charm 2023 - AD & DNS: A Match Made in Heck

1,172 0

Video: Protecting Users with “Protected Users”

Despite being around for 9 years, organizations are unaware that the Protected Users AD group exists let alone its benefits. In this talk, you'll learn the What, Why, and How of utilizing this group i

923 0

Comments