Privilege Escalation with DCShadow

blog.netwrix.com/2022/10/04/privilege-escalation-with-dcshadow

Jeff Warren

DCShadow is a feature in the open-source tool mimikatz. In another blog post, we cover how attackers can use DCShadow to achieve persistence in a domain without detection once they've obtained admin credentials. But DCShadow can also enable an attacker to elevate their privileges.

How can a Domain Admin elevate their access even higher? By obtaining admin rights in other forests. Leveraging SID History, an attacker can add administrative SIDs to their user account and obtain admin level rights in other trusted domains and forests. In this post, we'll take a look at how this works.

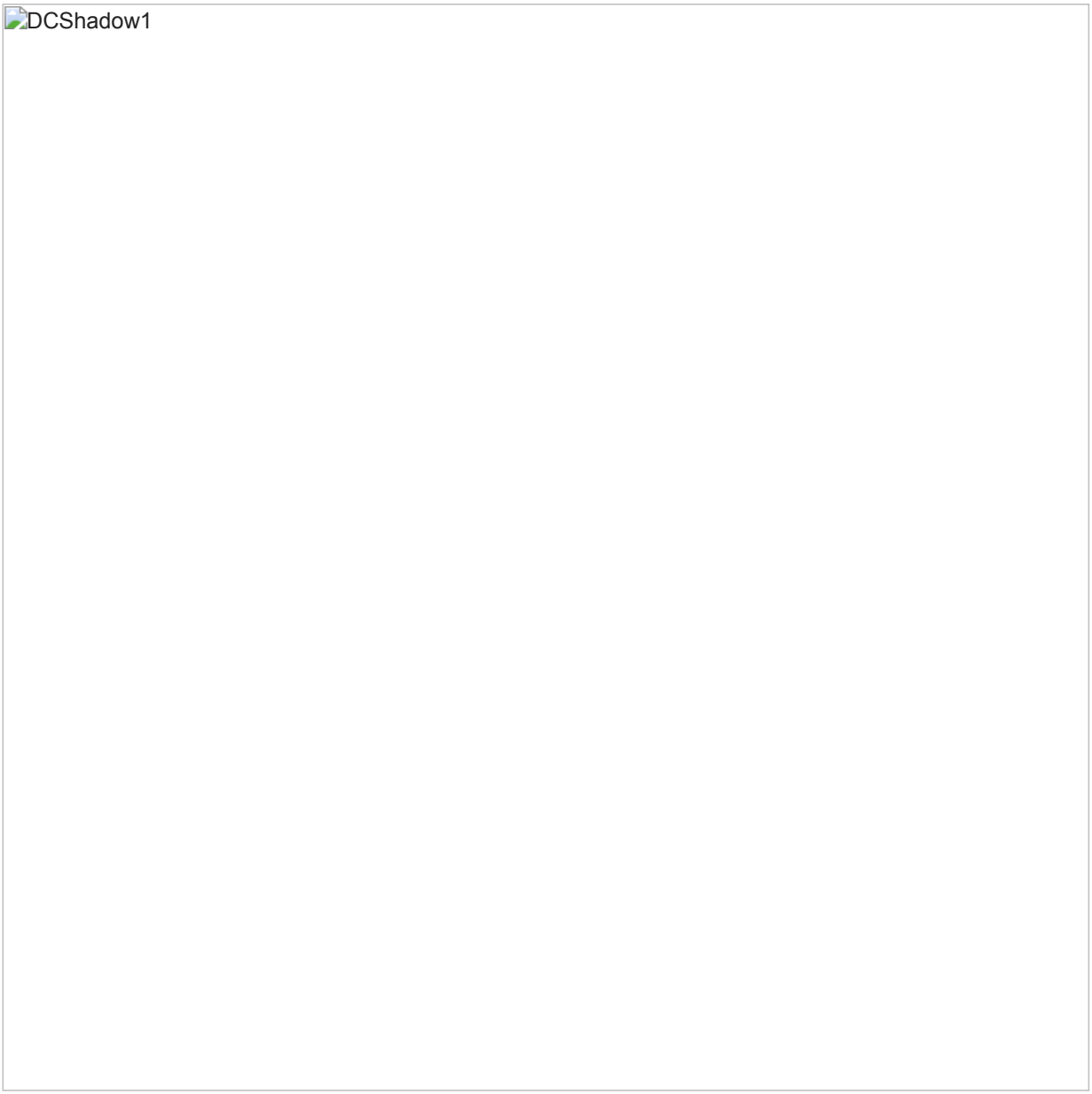
Step 1. Discover Trusts

The first step is to find out what trusts exist. There are several ways to do this, but two we will leverage through PowerShell are the PowerSploit framework and the Active Directory PowerShell module.

For each trust we find, we need to check whether SID filtering is enabled. If it is, then historical SIDs cannot be used to access the forest on the other side of the trust. However, if it is disabled, we are in business. Often this option is left disabled after migrations to ensure users don't lose access to any systems and data they need. The following PowerShell command will discover trusts and enumerate their options, including SID filtering:

```
Get-NetDomainTrust | ForEach-Object{Get-ADTrust –filter * -server $_.TargetName}
```

The output of this command is provided below. You can see there is a trust to the gobias.local domain where SID filtering is disabled (SidFilteringQuarantined = False), so we will be able to use historical SIDs to access resources in that domain.



To learn more about SID filtering and trusts, read this post on TechNet.

Step 2. Elevate Privileges using SID History

Next, we need to add an administrative SID to our user account so we can access resources in the trusted forest. DCShadow is going to come in handy here for two reasons:

- You cannot natively change SID History through applications like AD Users & Computers.
- DCShadow will make this change without any detection.

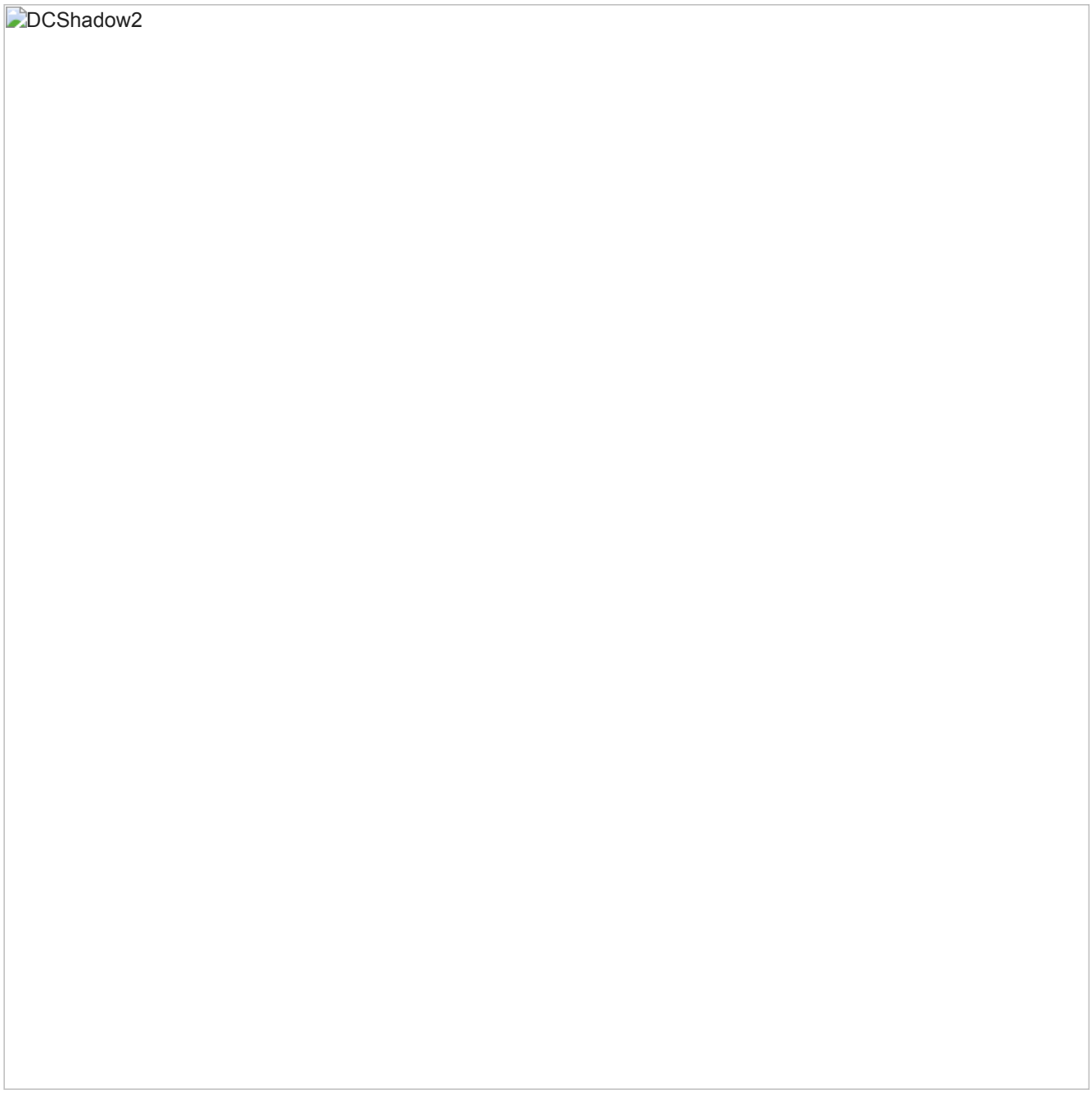
We just need to pick a SID to add to our SID History. We will avoid using any well-known SIDs and built-in users or groups such as Administrator and Domain Admins, since there are controls in place to allow these SIDs to be assigned only to their equivalent objects in other domains. Using domain reconnaissance, we should be able to find a domain user or group which we want to add to our access token to gain elevated rights.

Let's add the AD-Admins group from the gobias.local forest to our user account using the following DCShadow command:

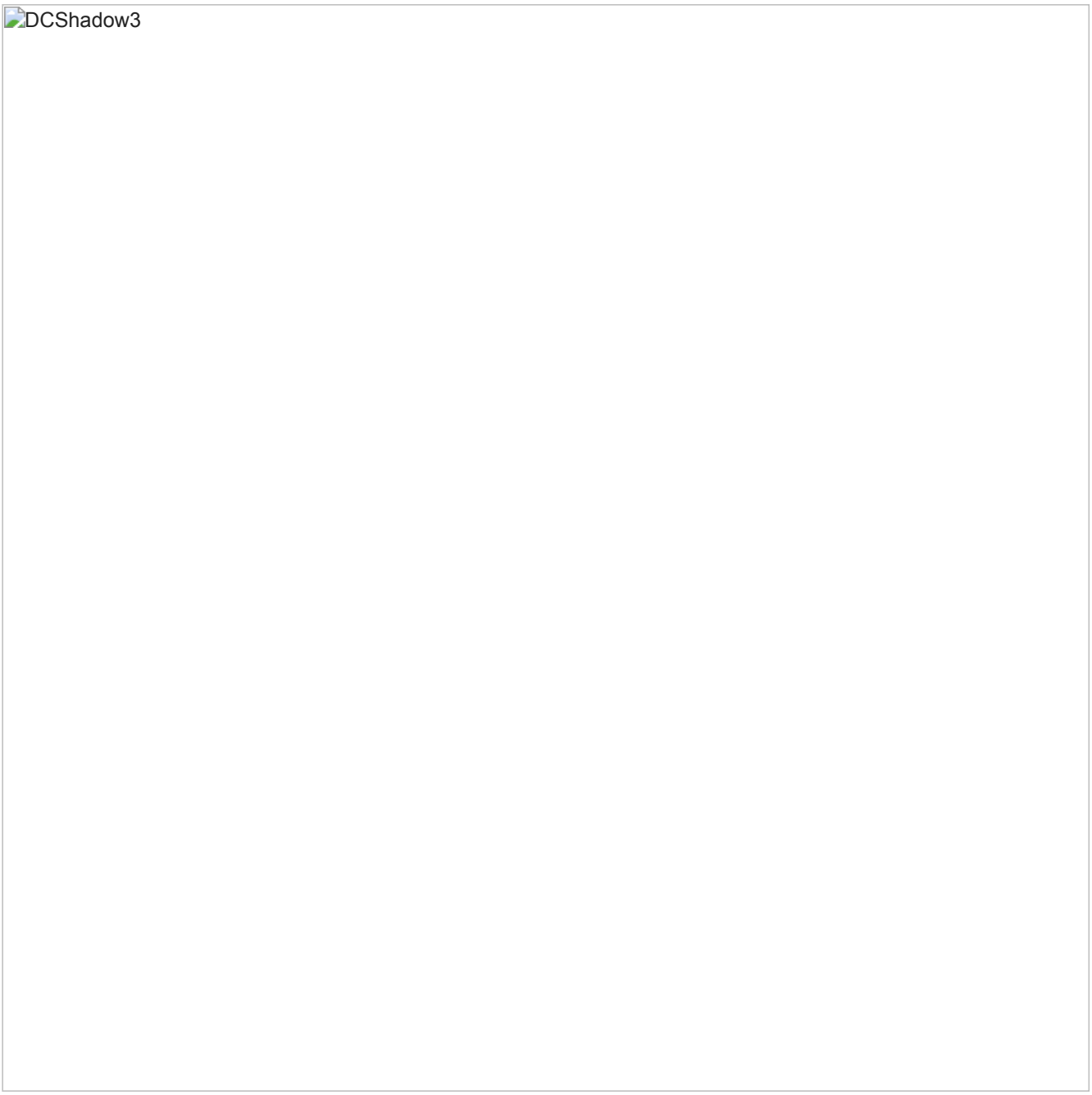
```
lsadump::dcshadow /object:"CN=Jeff Warren,OU=Administrators,OU=Users,OU=JEFFLAB,DC=JEFFLAB,DC=local"
/attribute:sidhistory /value:S-1-5-21-1722627474-2472677011-3296483304-1113
```

To see our newly added SIDhistory value, we can run the following script:

```
Get-ADUser Jeff -Properties sIDHistory
```



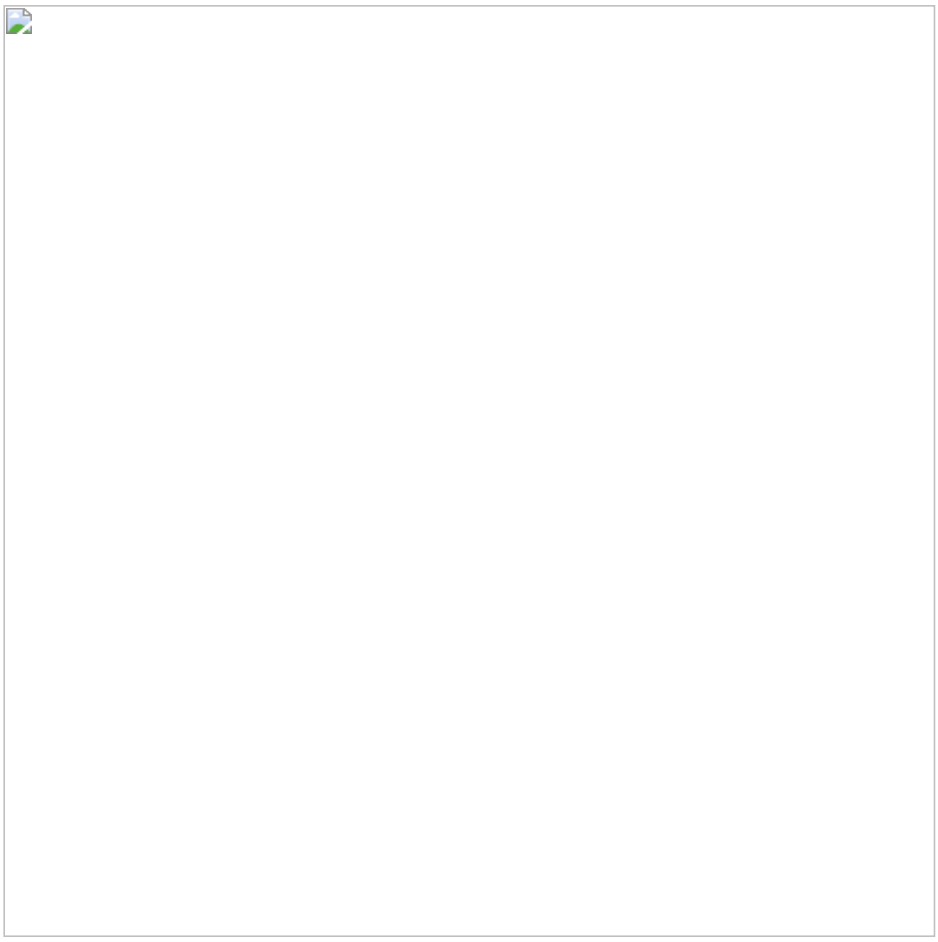
We can confirm this all worked by logging in again as this user and running a whoami /groups command to see the new group membership. Our user is only getting this group in its token through SID history.



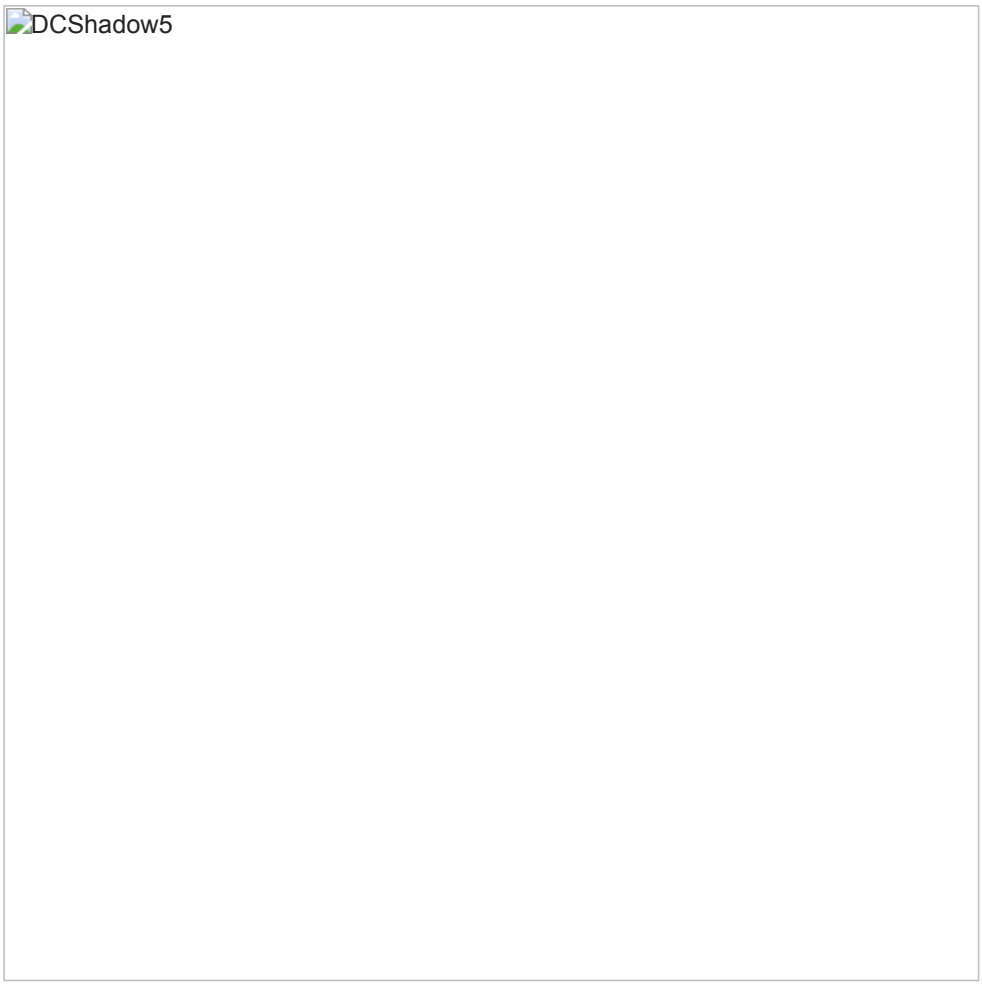
Step 3. Use the Elevated Privileges

Once we have access rights, there are any number of ways to extract data from the trusted forests. One of the most efficient is to use DCSync because it does not require any code to be run on the target domain controller.

Before we added the SID history to our account, attempting to run DCSync against the target forest would result in access being denied:



But after adding the historical SID to our user account, we are able to run the same command successfully and obtain the password hash to any account, including the extremely valuable krbtgt Kerberos service account.



DCShadow Detection and Response

The primary method used to detect DCShadow is finding patterns of behavior matching the registration and unregistration of rogue domain controllers and monitoring the replication traffic being pushed by them. Out of the box, Netwrix StealthDEFEND actively monitors all domain replication and change events for signs of DCShadow.

Netwrix StealthINTERCEPT blocking policies can be used to prevent the perpetrating account or workstation from executing additional replication, authentication and other activities, which can slow down an attacker and give responders more time to completely eliminate the threat.

Jeff Warren