What a DCShadow Attack Is and How to Defend Against It

blog.netwrix.com/2022/09/28/dcshadow_attack

DCShadow is a late-stage kill chain attack that allows an adversary with compromised privileged credentials to register a rogue Active Directory domain controller (DC) and replicate malicious changes, such as modifications that help them establish persistence. More specifically, DCShadow is a command in the lsadump module of the open-source hacking tool Mimikatz. Introduced in early 2018, it utilizes specific instructions in the Microsoft Directory Replication Service Remote (MS-DRSR) protocol.

Handpicked related content:

[Free Guide] Privileged Access Management Best Practices

DCShadow attacks are difficult to prevent. Like DCSync, it does not abuse a vulnerability that could be patched; it exploits valid and necessary functions of the replication process, which cannot be turned off or disabled. DCShadow attacks are also difficult to detect, since the changes that the adversary requests are registered, processed and committed as legitimate domain replication.

Overview of a DCShadow Attack

Here is an overview of the steps in a DCShadow attack:

1. An attacker obtains Domain Admin or Enterprise Admin permissions, for example, by compromising a poorly secured group-managed service account.
2. The attacker uses DCShadow to register a computer object (such as a workstation) as a domain controller by making changes to the configuration schema and the workstation's SPN. Now, AD thinks the workstation is a DC so it's trusted to replicate changes.
3. The attacker submits changes for replication, such as changes to password data, account details or security group membership. Once replication is triggered, changes are published and committed by the other DCs.
4. The attacker uses DCShadow to remove the rogue domain controller from the AD database to cover their activity.

A Simple Example: Replicating a Minor Change Using DCShadow

Let's assume you are an attacker who has registered a rogue DC using DCShadow. Here are the steps you could take to replicate a simple change across all the DCs in the domain.

Step 1. Switch to using the SYSTEM account.

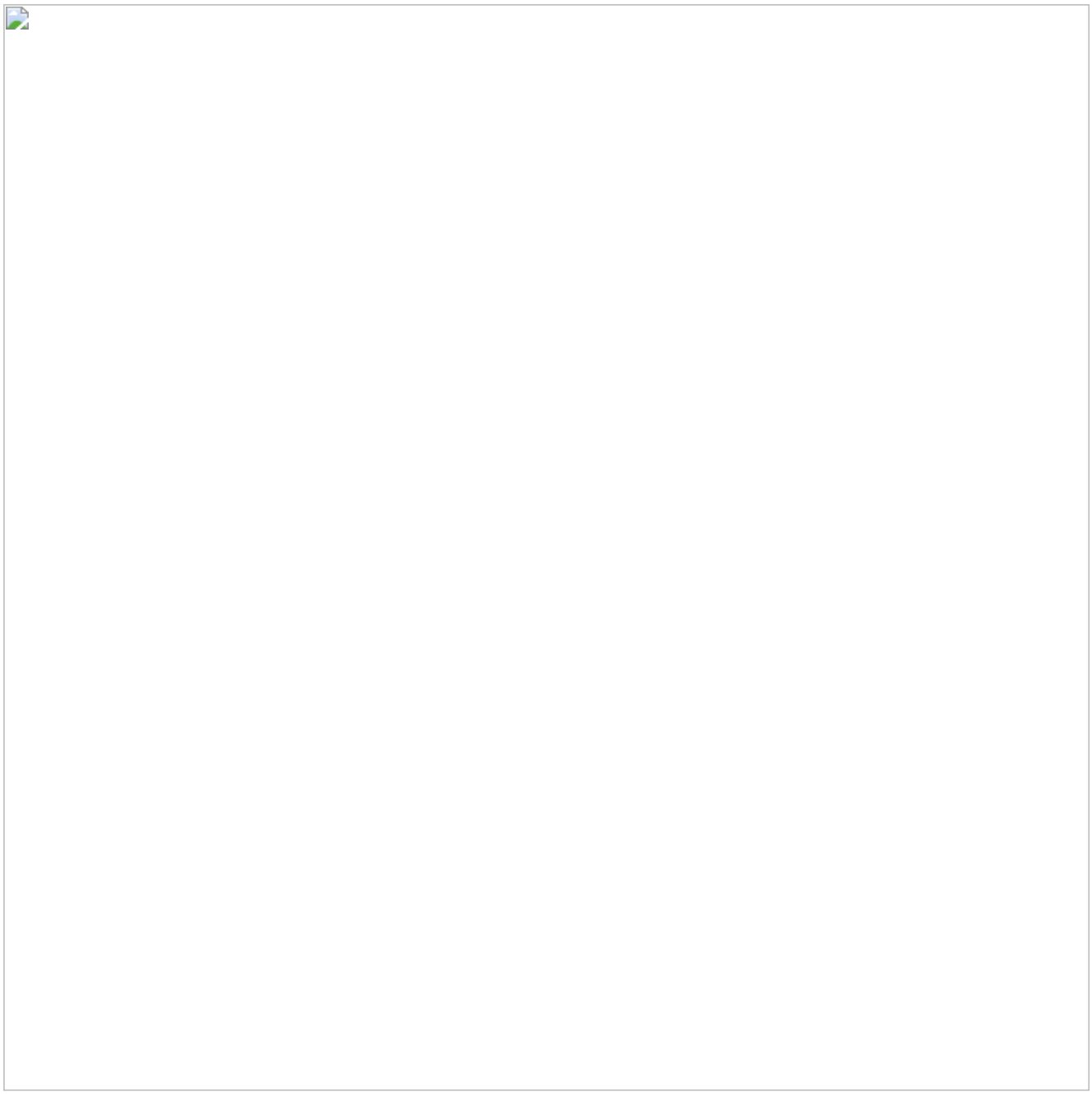
The process that creates the change that is to be replicated must be run as the SYSTEM account, rather than a domain user account, since only changes from registered DCs will be replicated. If you fail to do this, you may see this error code:

```
ERROR kuhl_m_lsadump_dcshadow_force_sync_partition ; IDL_DRSReplicaAdd DC=JEFFLAB,DC=local
0x80090322 (2148074274)
```

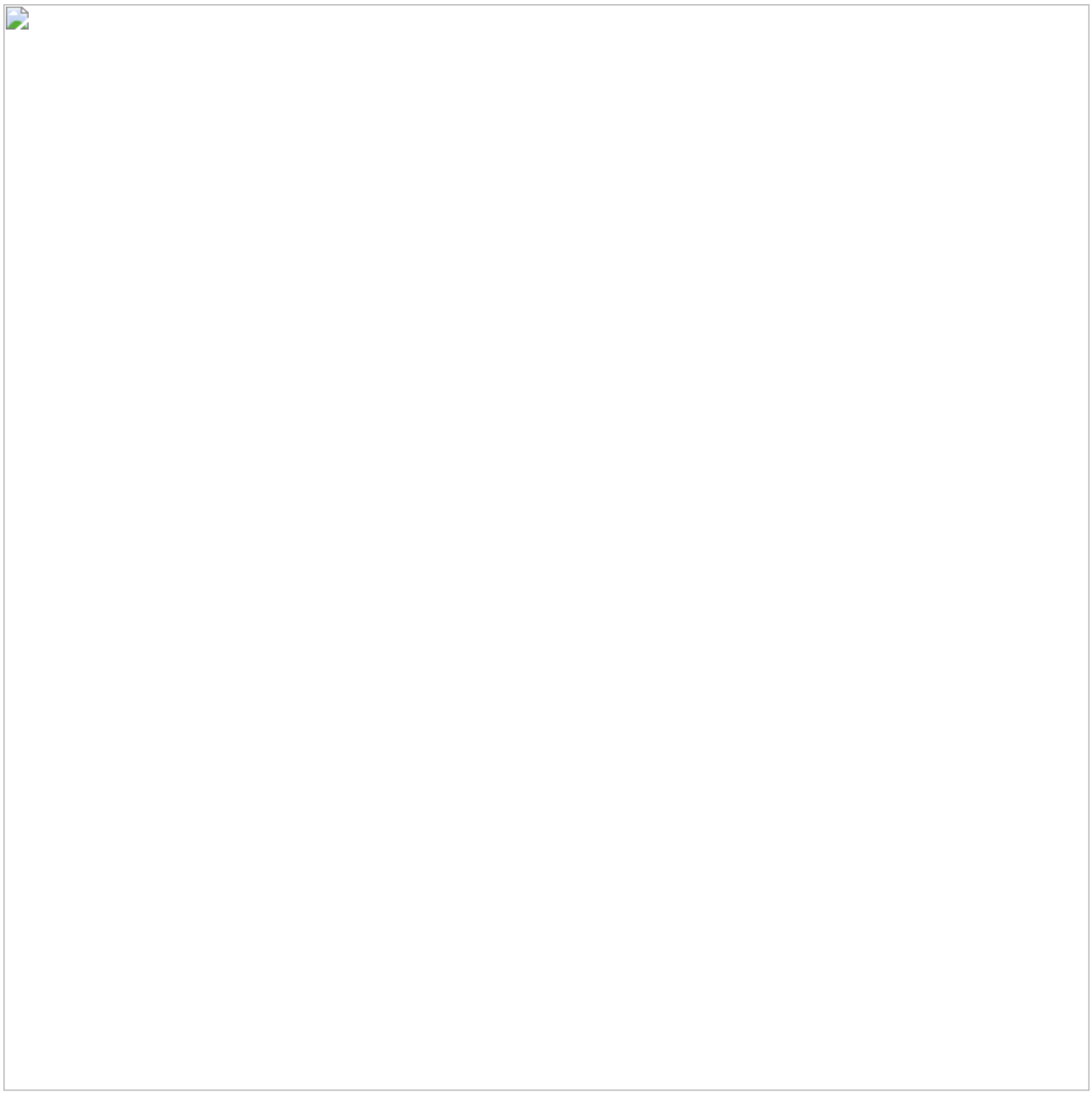
One way to run a process as SYSTEM in Mimikatz is to use PSExec:

```
PsExec.exe –i –s cmd.exe
```

To confirm that you are now running under SYSTEM, use the **whoami** command.



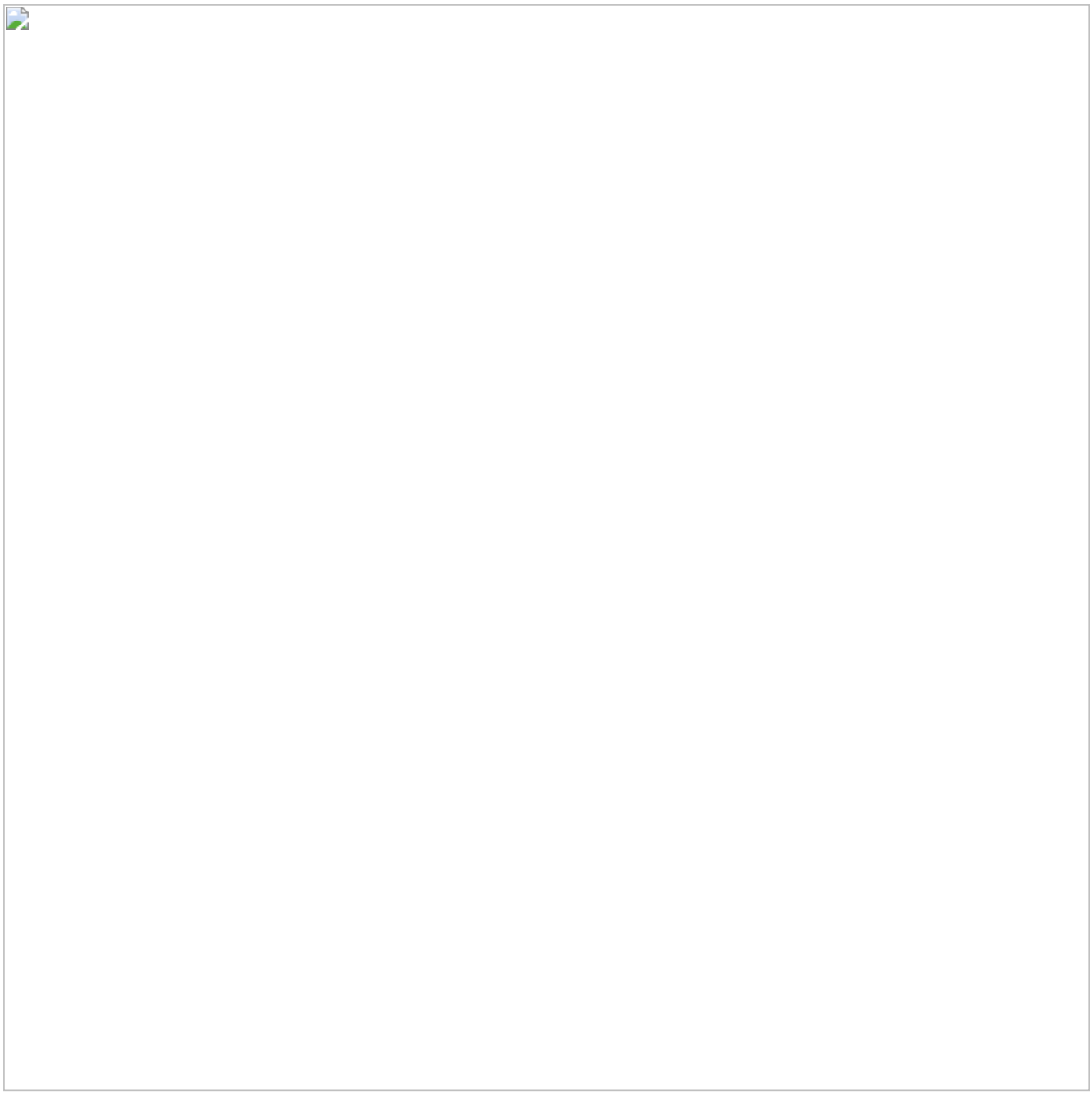
Then use **token::elevate** to be sure all threads will run as SYSTEM.



Step 2. Push a desired change.

Now you will be able to push a change. For this simple example, let's change the description of the Administrator account:

```
lsadump::dcshadow /object:CN=Administrator,CN=Users,DC=JEFFLAB,DC=local /attribute:description
/value:"DCShadow was here!"
```

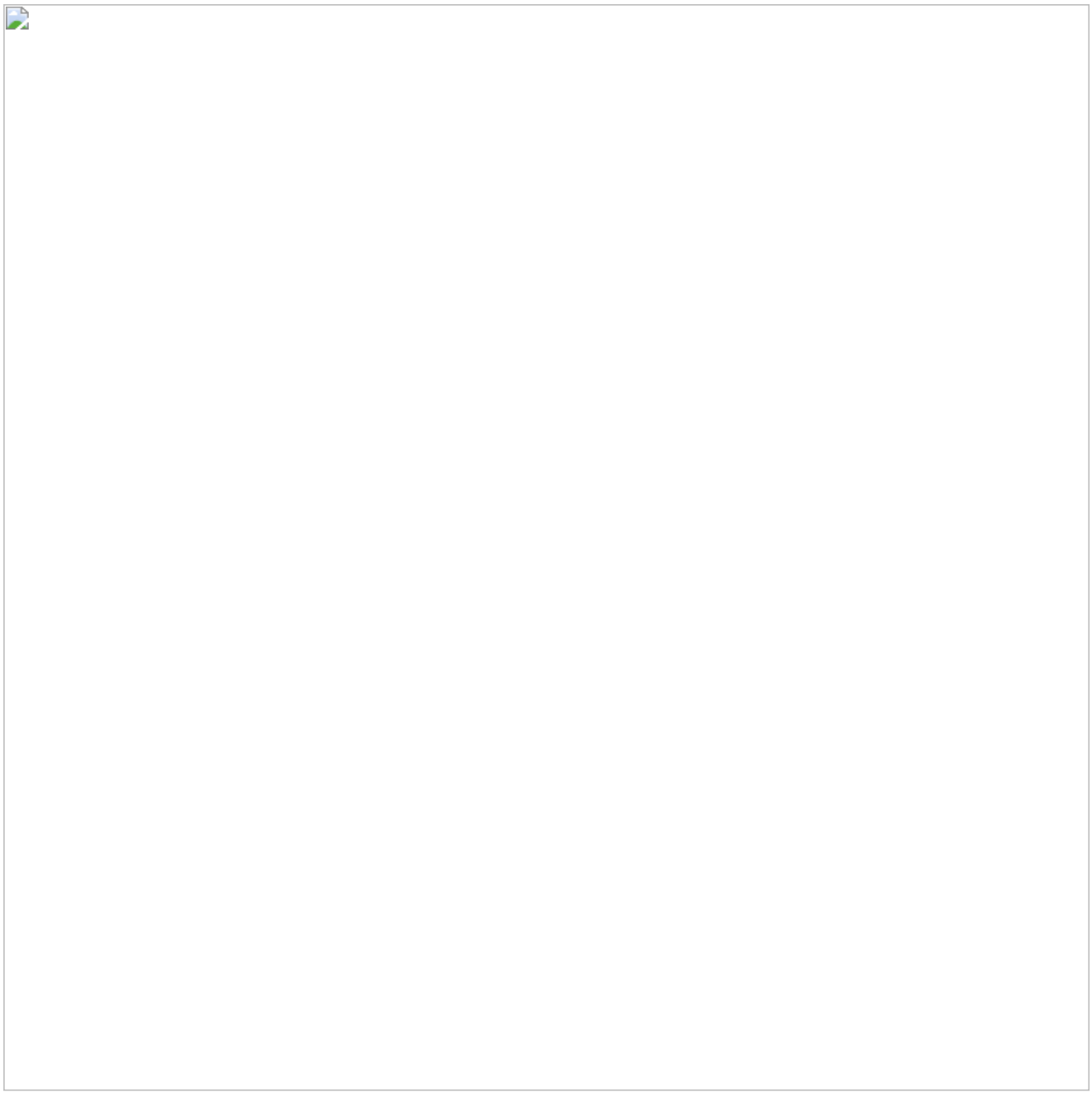


Step 3. Trigger replication.

To trigger replication, open another window and run the following command as the compromised Domain Admin account (not as SYSTEM):

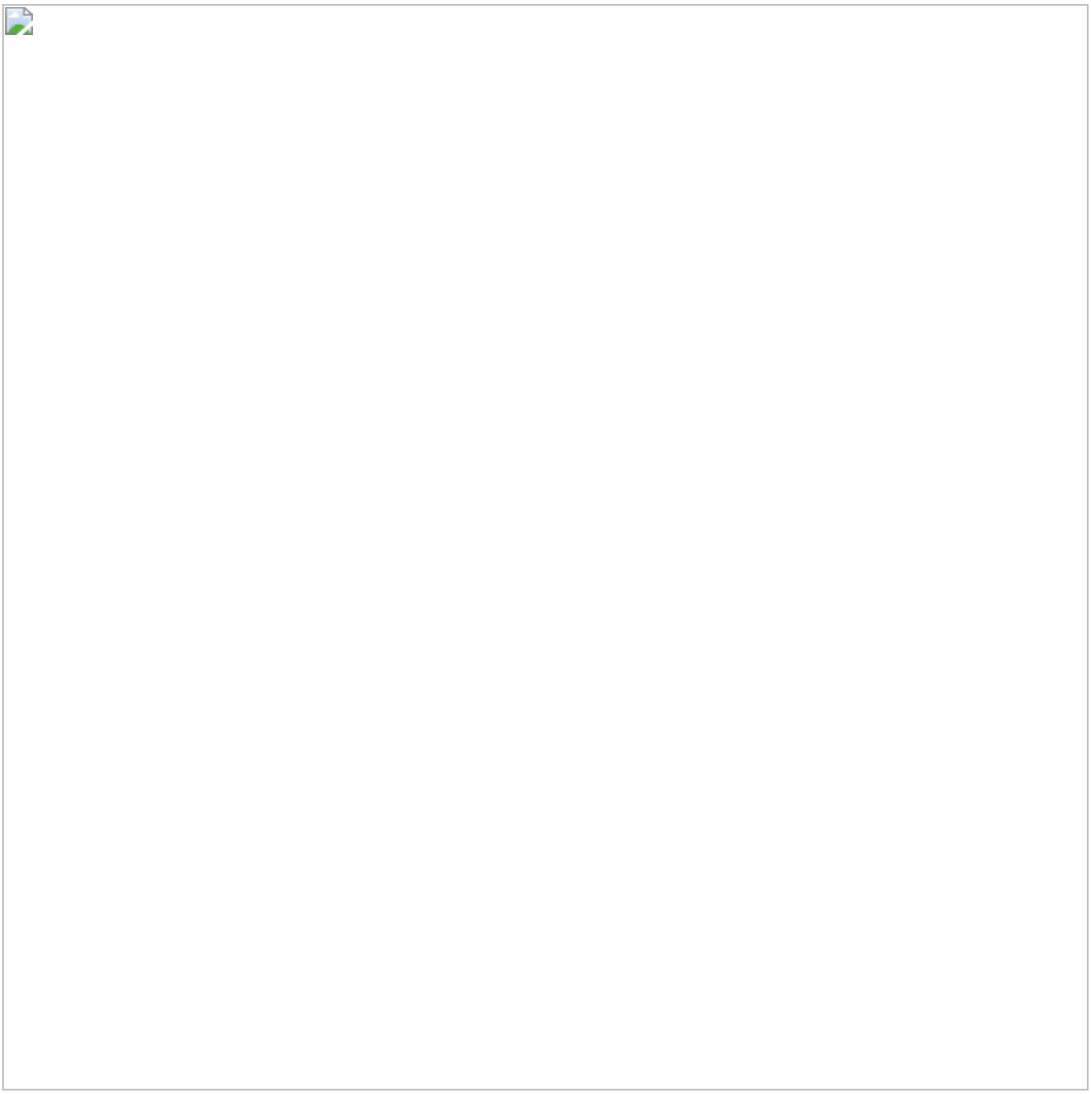
```
lsadump::dcshadow /push
```

The original window will show the resulting events:



Step 4. Confirm the change.

Go to the domain controller and check the description field:



A Real-World Attack: Using DCShadow to Achieve Persistence

Now let's see how an adversary would DCShadow in a real-world attack. To run DCShadow, they must already have Domain Admin or Enterprise Admin rights, so why would they need to use DCShadow?

One reason is that they may not want to use a Domain Admin account for data exfiltration, since those accounts are often closely monitored and trigger alarms more easily. Another reason is to obtain administrative rights in other forests.

But one of the most common reasons is to create persistence — so that even if they lose access to the admin account they compromised, they will retain a foothold in the domain. Let's step through one way we could use DCShadow to achieve persistence.

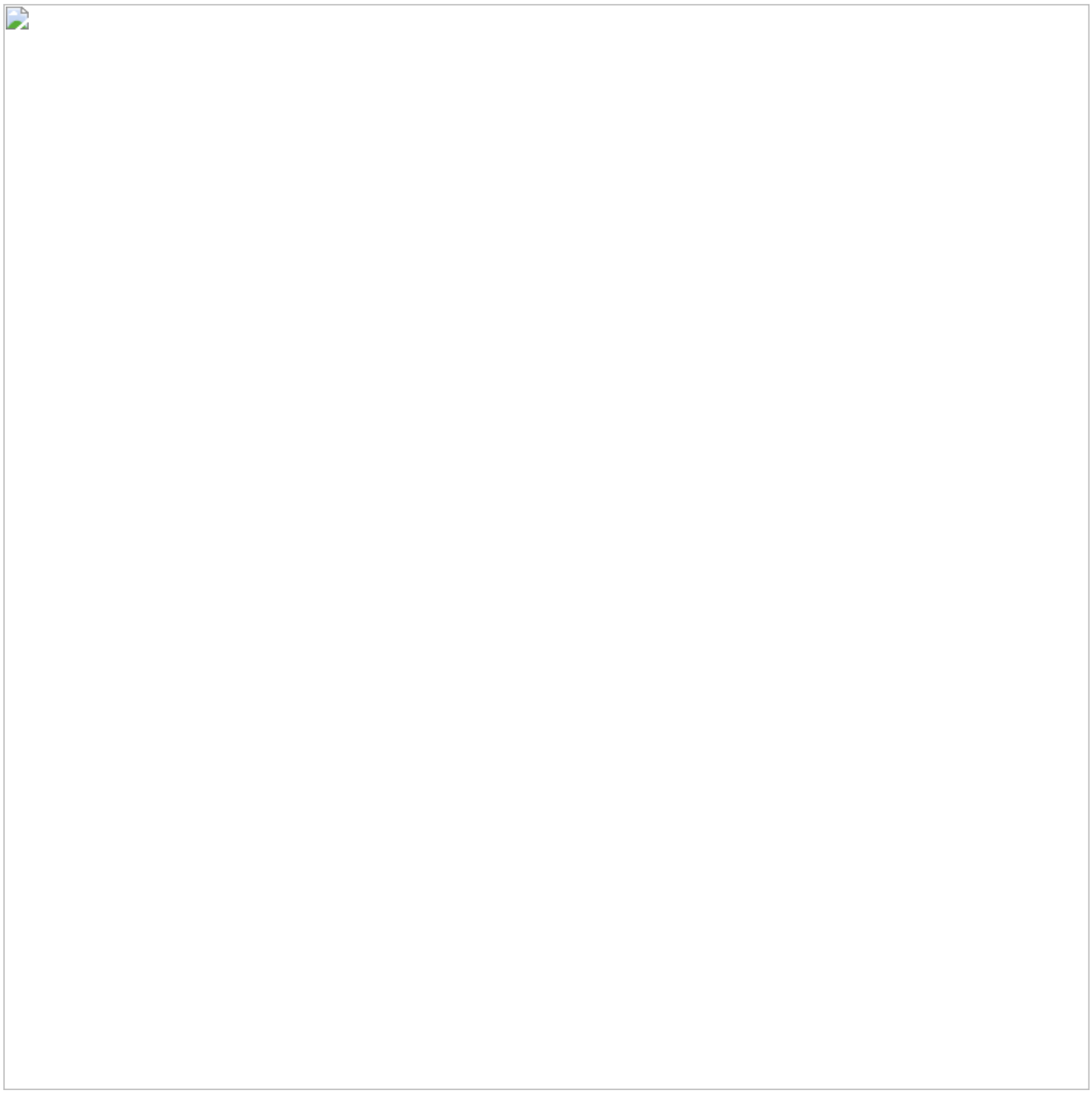
The basic workflow is as follows:

1. Review the AdminSDHolder object, which provides template permissions for the protected accounts and groups in the domain.
2. Add a new account with Full Control permissions to AdminSDHolder.
3. Replicate the change using DCShadow.

Step 1. Review the AdminSDHolder object.

To review the AdminSDHolder object, we will use some basic PowerShell:

```
$AdminSDHolder = [adsi]'LDAP://CN=AdminSDHolder,CN=System,DC=JEFFLAB,DC=local'

$SDDL = $AdminSDHolder.ObjectSecurity.Sddl
```

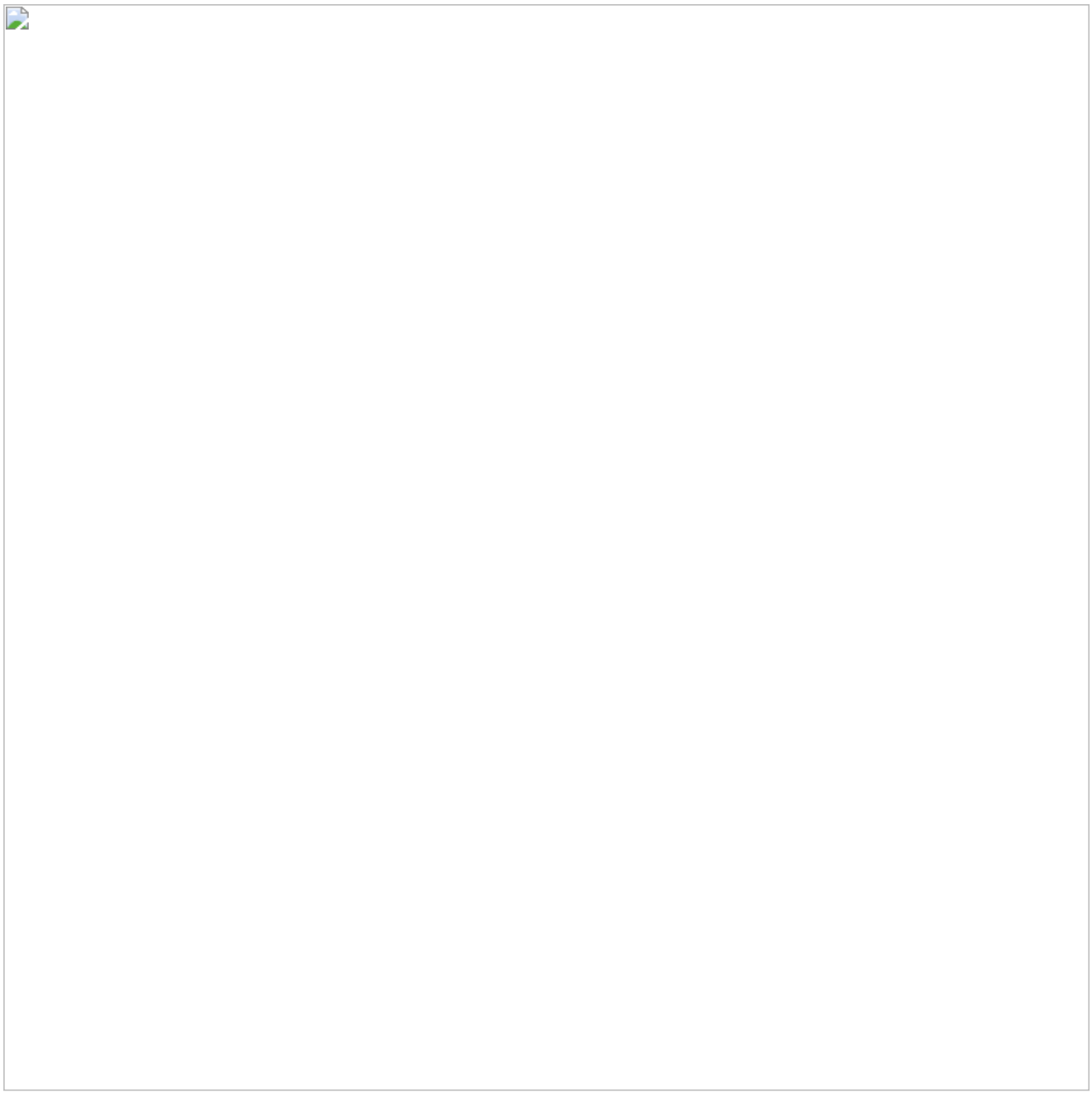


We can use the ConvertFrom-SDDLString command to convert the result to a more readable format:

Step 2. Add an Account to the AdminSDHolder Container.

To create persistence, we must add an account to AdminSDHolder using its SID. All we need to know is the distinguished name of the object, and the following command will store its SID in the variable **$UserSid**:

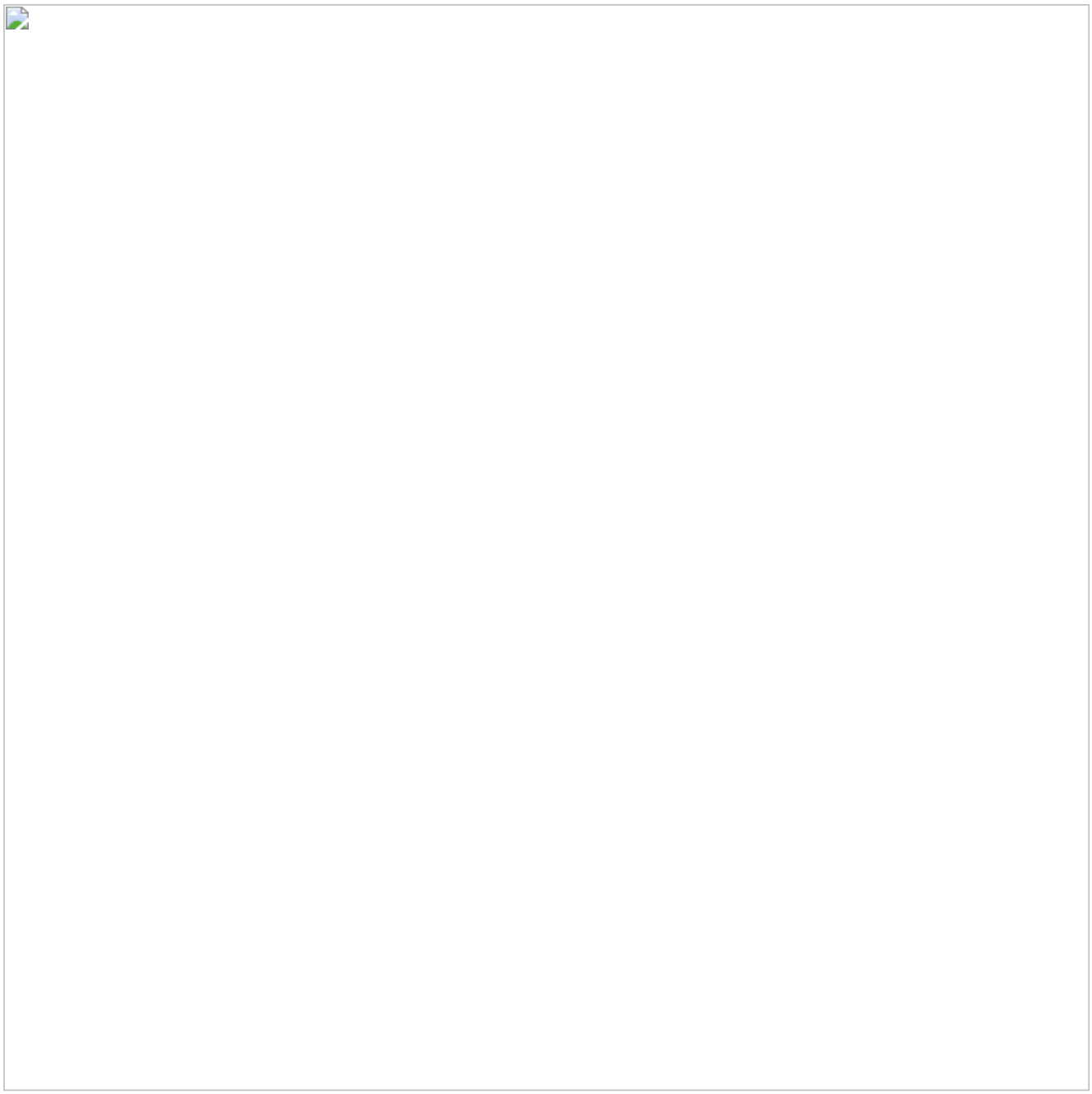
```
$UserToAdd = [adsi]'LDAP://CN=Bob Loblaw,OU=Business,OU=Users,OU=JEFFLAB,DC=JEFFLAB,DC=local'

$UserSid  = New-Object System.Security.Principal.SecurityIdentifier($UserToAdd.objectSid[0], 0)
```



Now we can use that variable to add the account to AdminSDHolder, giving it Full Control permissions. We will use the following PowerShell command:

```
$NewSDDL = $SDDL + "(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;" + $UserSid.Value + ")"
```

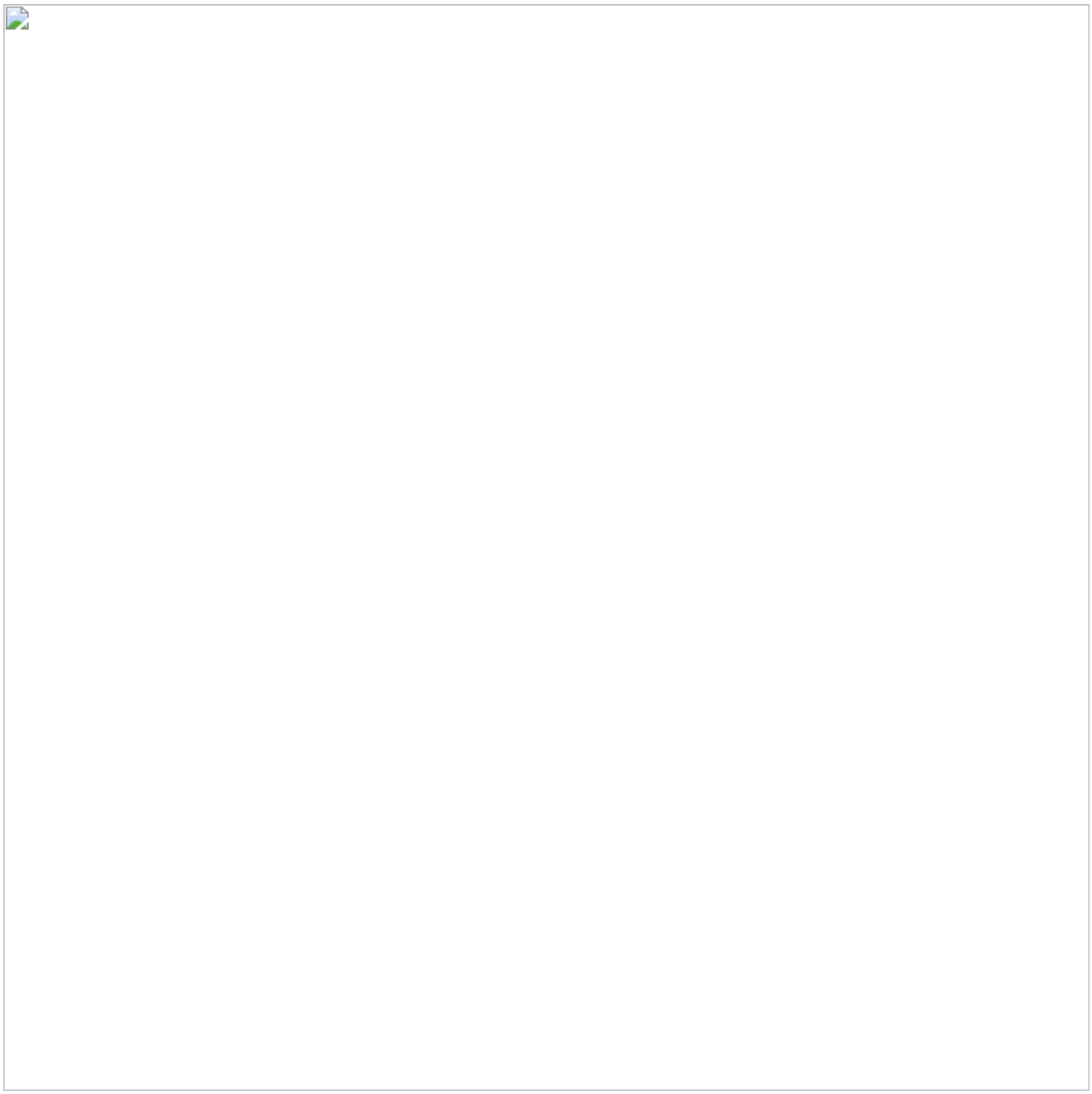
If we take another look at AdminSDHolder, we now see the user as the last entry:



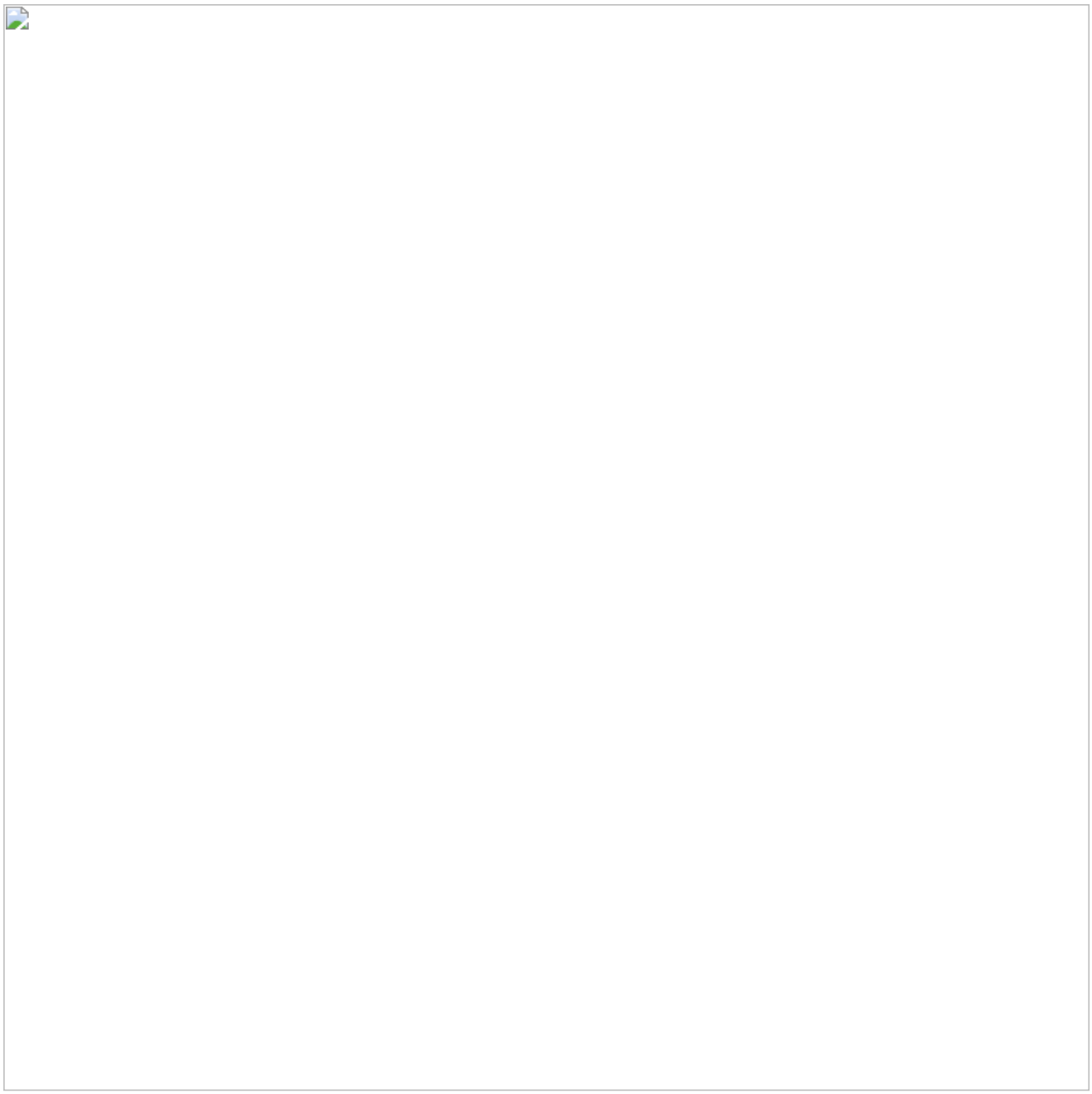
Step 3. Replicate the change using DCShadow.

Now that we have the new permissions in the form of an attribute value, it is easy to apply them with DCShadow. Running as SYSTEM, we can use the following command to make the desired change:

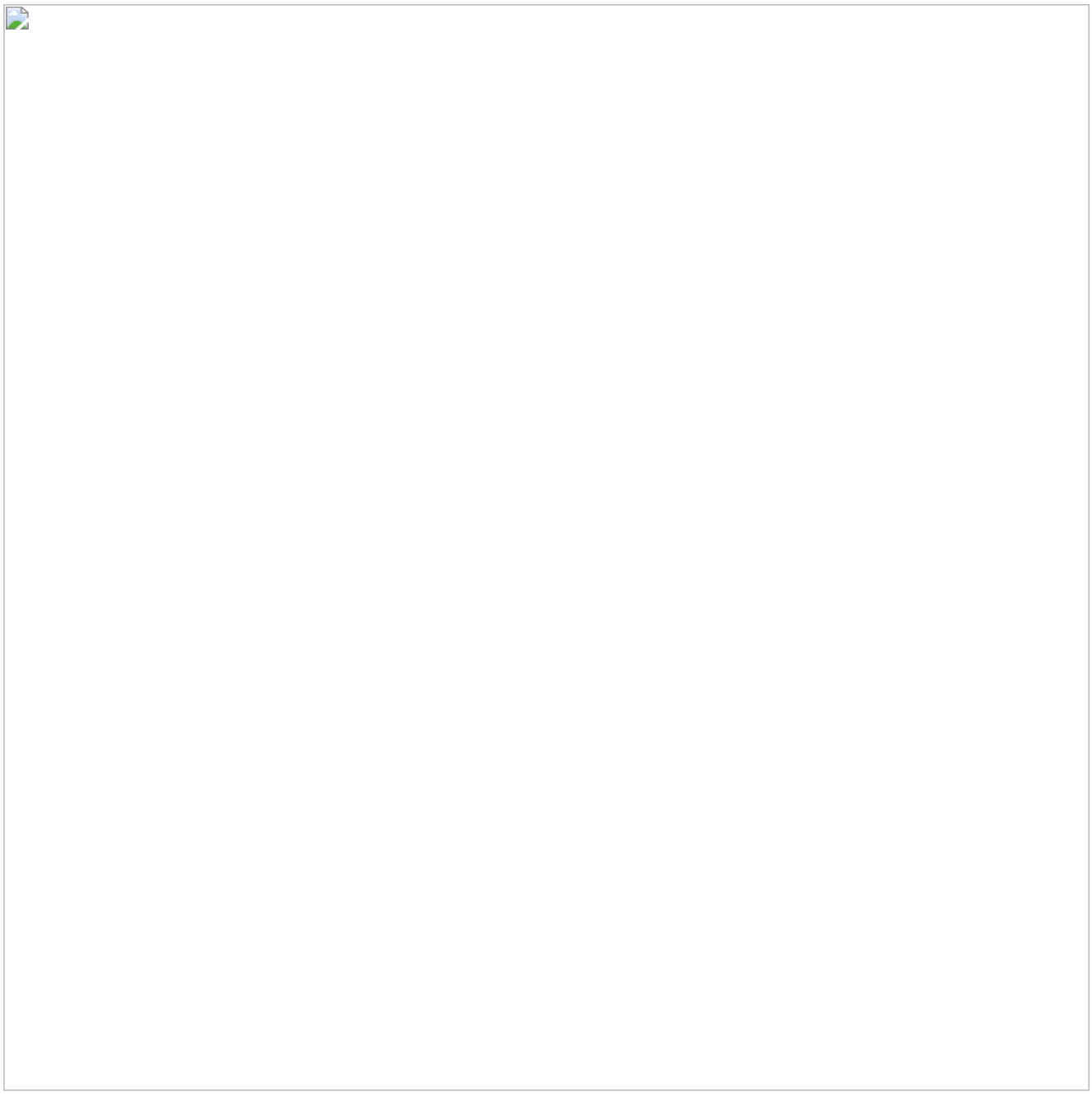
```
mimikatz.exe "lsadump::dcshadow /object:CN=AdminSDHolder,CN=System,DC=JEFFLAB,DC=local
/attribute:ntsecuritydescriptor /value:[output from the previous command]"
```



Here, you can see the change is ready to be replicated:



Then we use the **lsadump::dcshadow/push** command to trigger the replication. Here you can see the updated AdminSDHolder object with our user added; soon this will be on every protected group in the domain.



Preventing DCShadow Attacks

This is just one way that DCShadow can be used to establish persistence or otherwise undermine Active Directory security. It is challenging to prevent these attacks because they leverage native features of Active Directory, not flaws that can be fixed by patching.

The primary key to defense lies in the fact that an attacker must have Domain Admin or Enterprise Admin rights to perform the DCShadow attack. Accordingly, the most effective strategy for blocking this attack is to prevent anyone from gaining unauthorized membership in these powerful security groups.

Detecting DCShadow Attacks

A key benefit of using DCShadow is that an attacker can make changes that will not be recorded by event logs — since the changes are made through replication, they are seen as legitimate. Nevertheless, you can find signs that an attack is happening using the event logs. Here are the key events to look for.

Changes to Computer SPNs

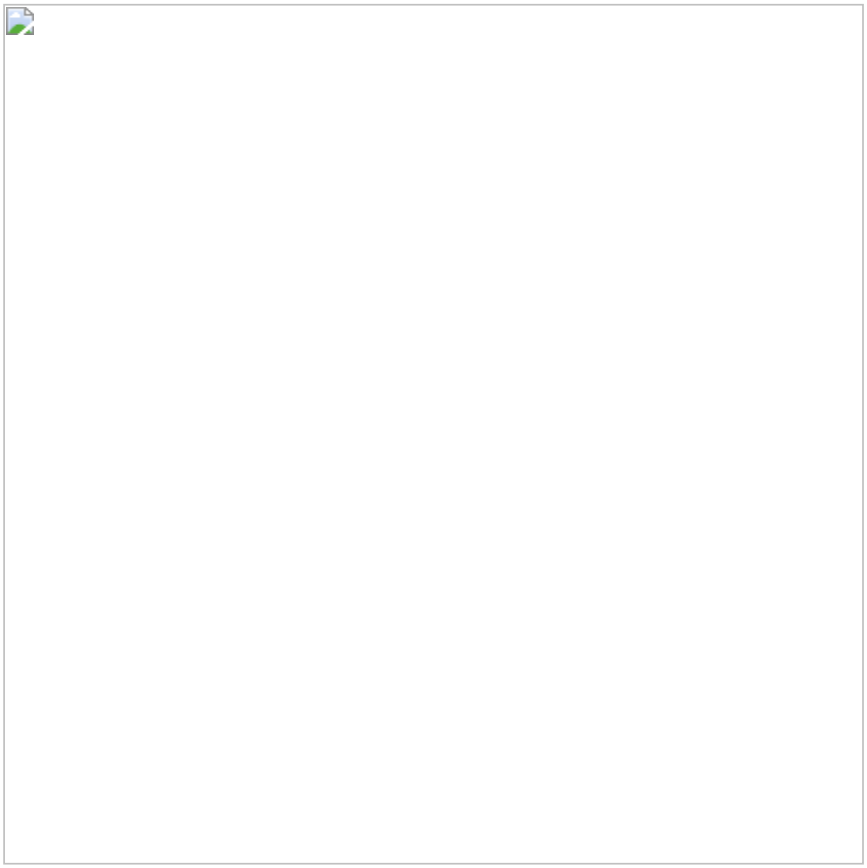
For DCShadow to work, two particular service principal name (SPN) values must be added to the computer that will be impersonating the domain controller (the computer from which the attacker is executing DCShadow). This can be any domain-joined computer.

Those two SPNS are:

- **Global Catalog server SPN** — This has the format *GC/<DNS hostname>/<DNS forest name>.*
- **Directory Replication Service (DRS) SPN** — This has the format *<DRS interface GUID>/<DSA GUID>/<DNS domain name>*, where *<DRS interface GUID>* is always *E3514235–4B06–11D1-AB04–00C04FC2DCD2***.**

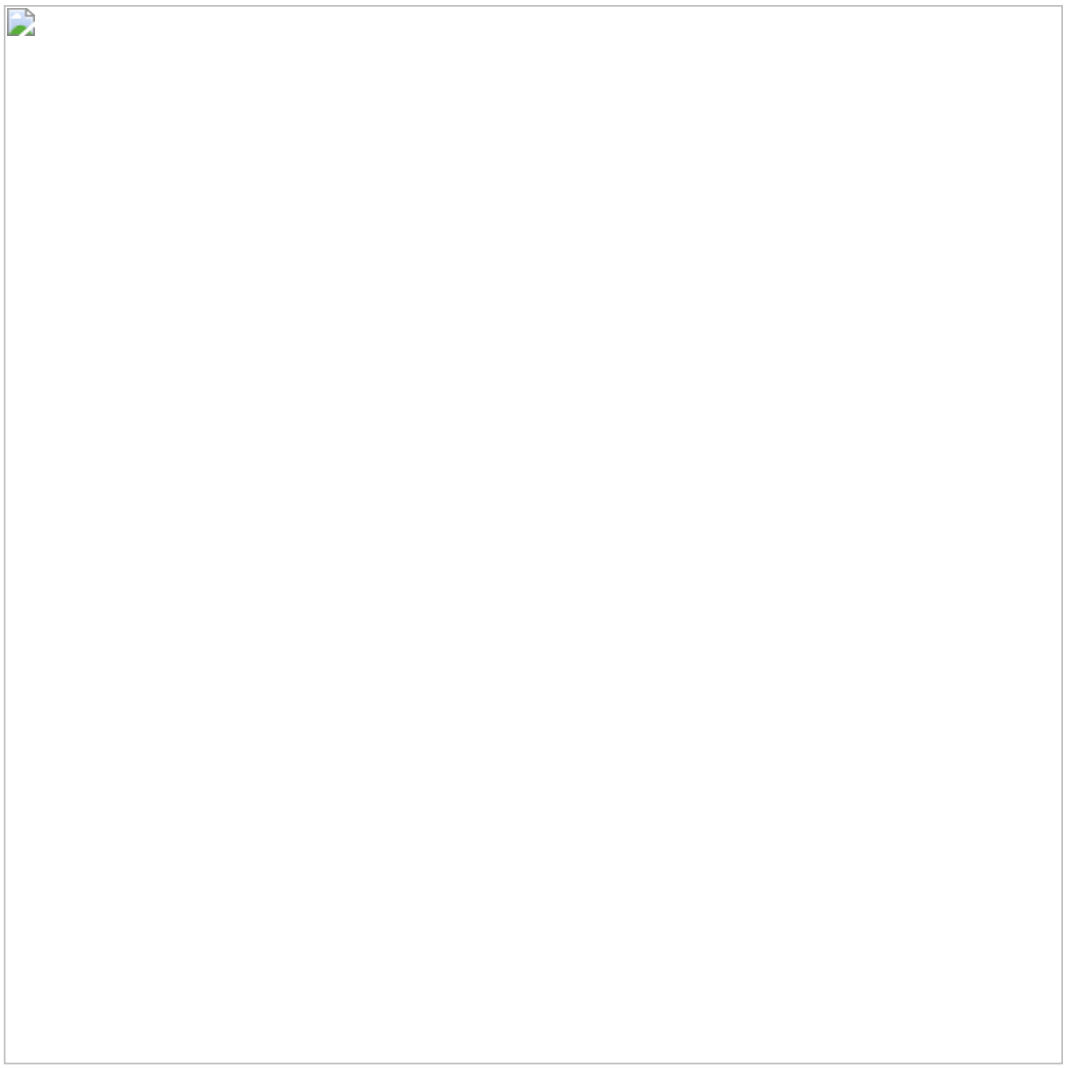
We can spot signs of DCShadow attacks by looking for the addition of these SPNs to a computer that is not a domain controller, followed by the removal of those SPNs. (Note that in our lab, DCShadow removes only the Global Catalog server SPN; it leaves the DRS SPN.)

We can use Event ID 4742 to monitor for these changes. This event shows which user initiated the change, so you know which Domain Admin account is being used to perform the attack.



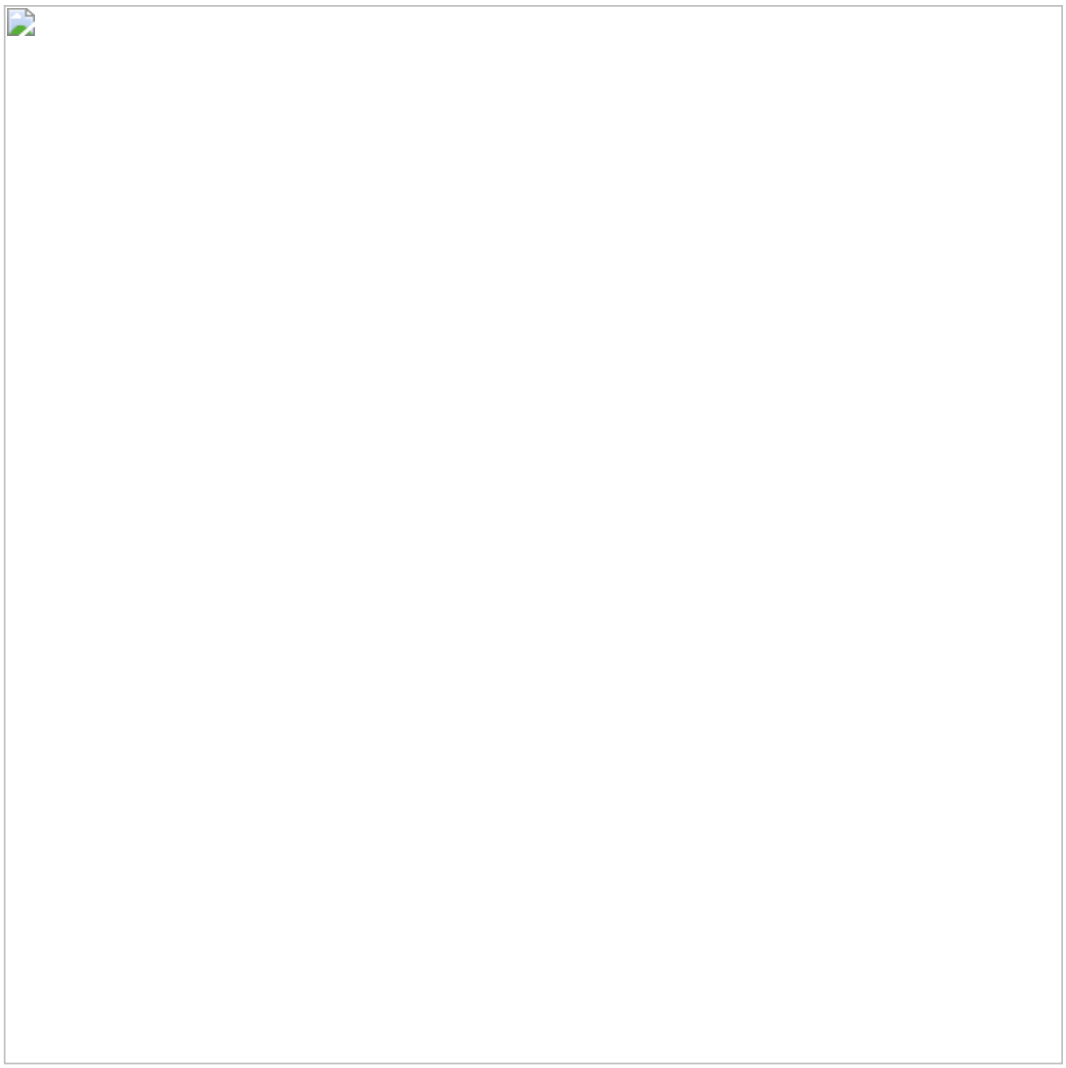
Creation and Deletion of a DC

Another step in the DCShadow attack is to create a DC in the Sites container of the Configuration Namespace. This is done by creating a server and NTDS settings for that server. You can see these objects for our legitimate domain controller below:

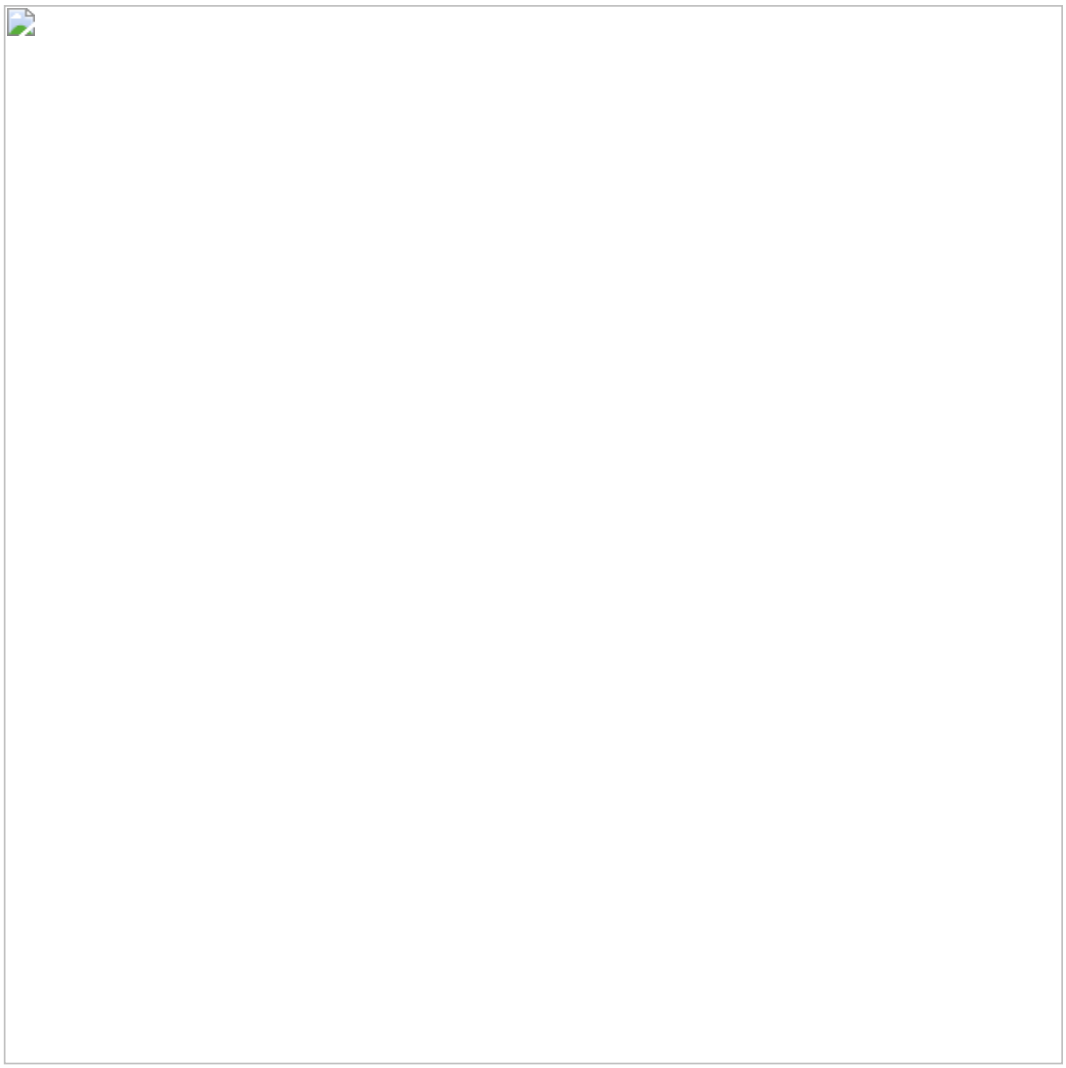


DCShadow will create a DC and its settings and then, once the change is replicated, it will immediately delete the entries to cover its tracks. This will leave behind a strange sequence of events of a new DC being added and then removed.

The addition can be tracked with Event ID 5137, which contains the name of the rogue DC, its GUID and object class, and the account responsible for creating it:



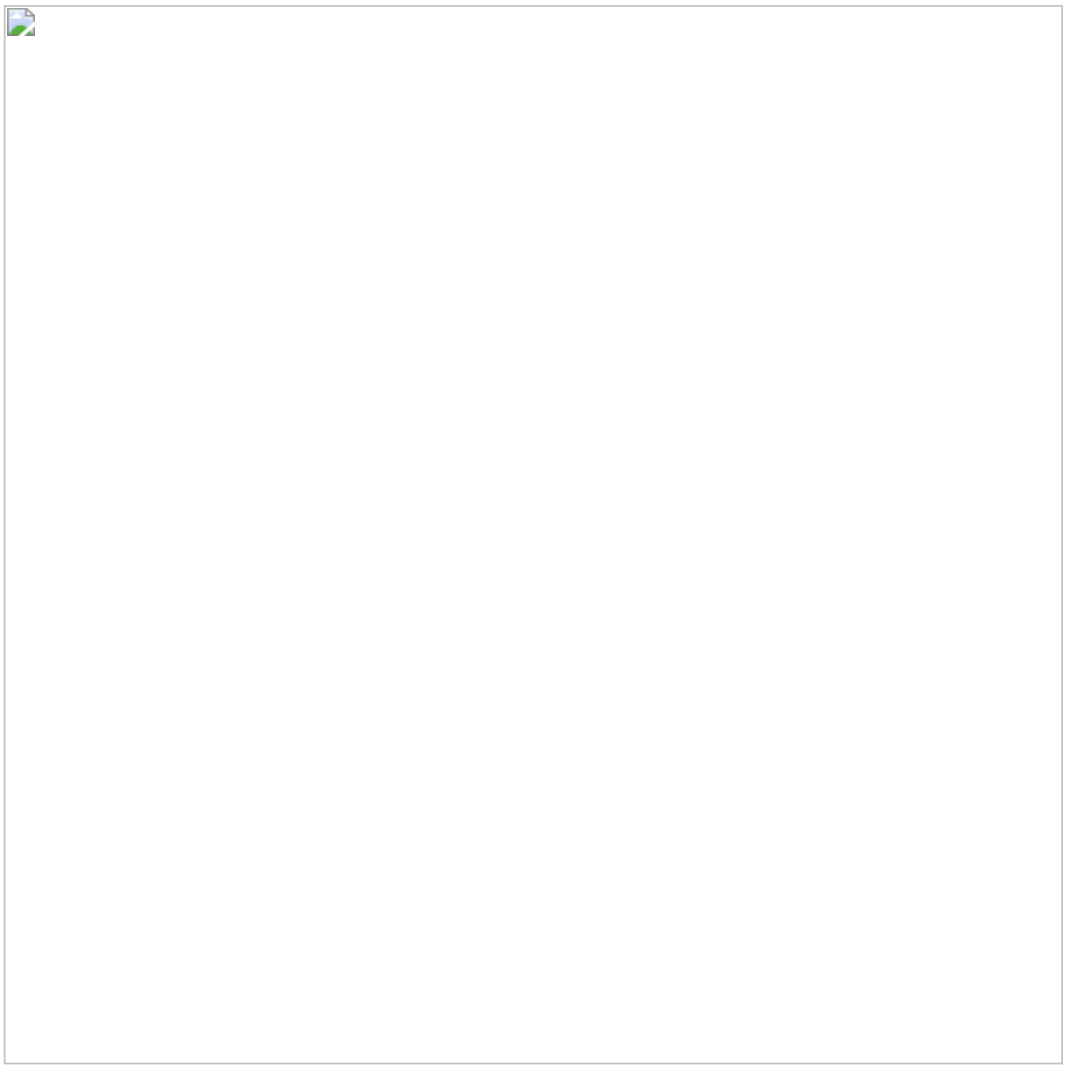
Event ID 5141 will show the same information for the deletion of the DC:



Unusual Replication Event

It is also possible to detect DCShadow attacks by monitoring replication. However, the replication events that are triggered may be difficult to differentiate from genuine replication events.

Still, Event ID 4929 can be a useful clue, since it indicates that a source naming context has been removed and will point to the rogue DC as the source. Seeing this event for a computer that is not a recognized domain controller should raise a red flag.



Replication Failure

In addition, the pair of events 4935 & 4936 indicate a replication failure that you can often tie to DCShadow.

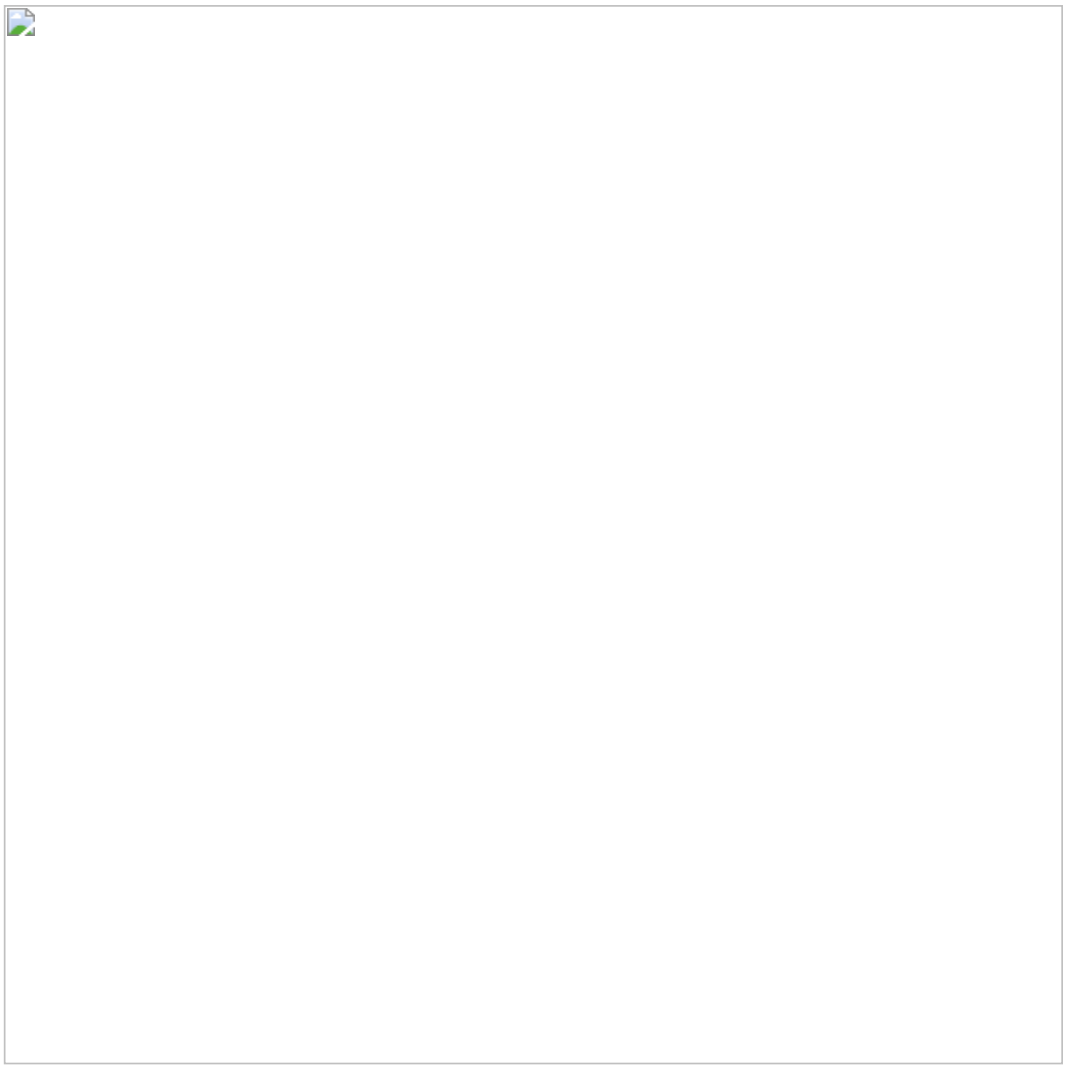
How Netwrix Can Help

Netwrix StealthDEFEND is a best-in-class solution for detecting and responding to DCShadow attacks:

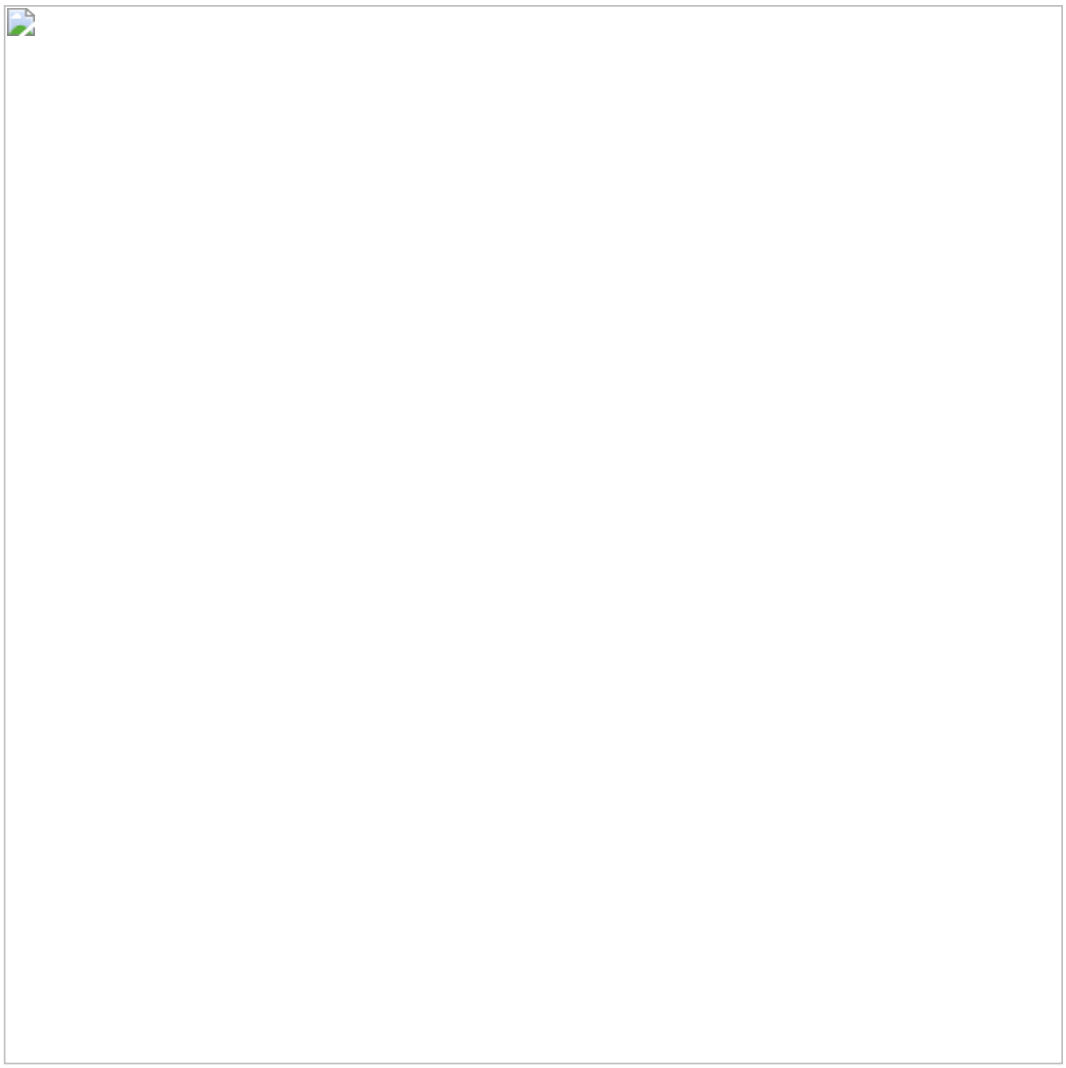
Detection

Netwrix StealthDEFEND offers built-in DCShadow threat detection. It monitors all domain replication and change events in real time for behavior indicative of DCShadow attacks. In particular, it watches for the addition and deletion of DCs and monitors monitoring replication traffic.

Below is an example in which Netwrix StealthDEFEND has detected the addition of new domain controller:



The addition of a DC is suspicious enough, but the report also highlights that the machine is running Windows 10, which doesn't support the domain controller role. By expanding the event details, we can see the specific changes that were made as part of the DCShadow attack:

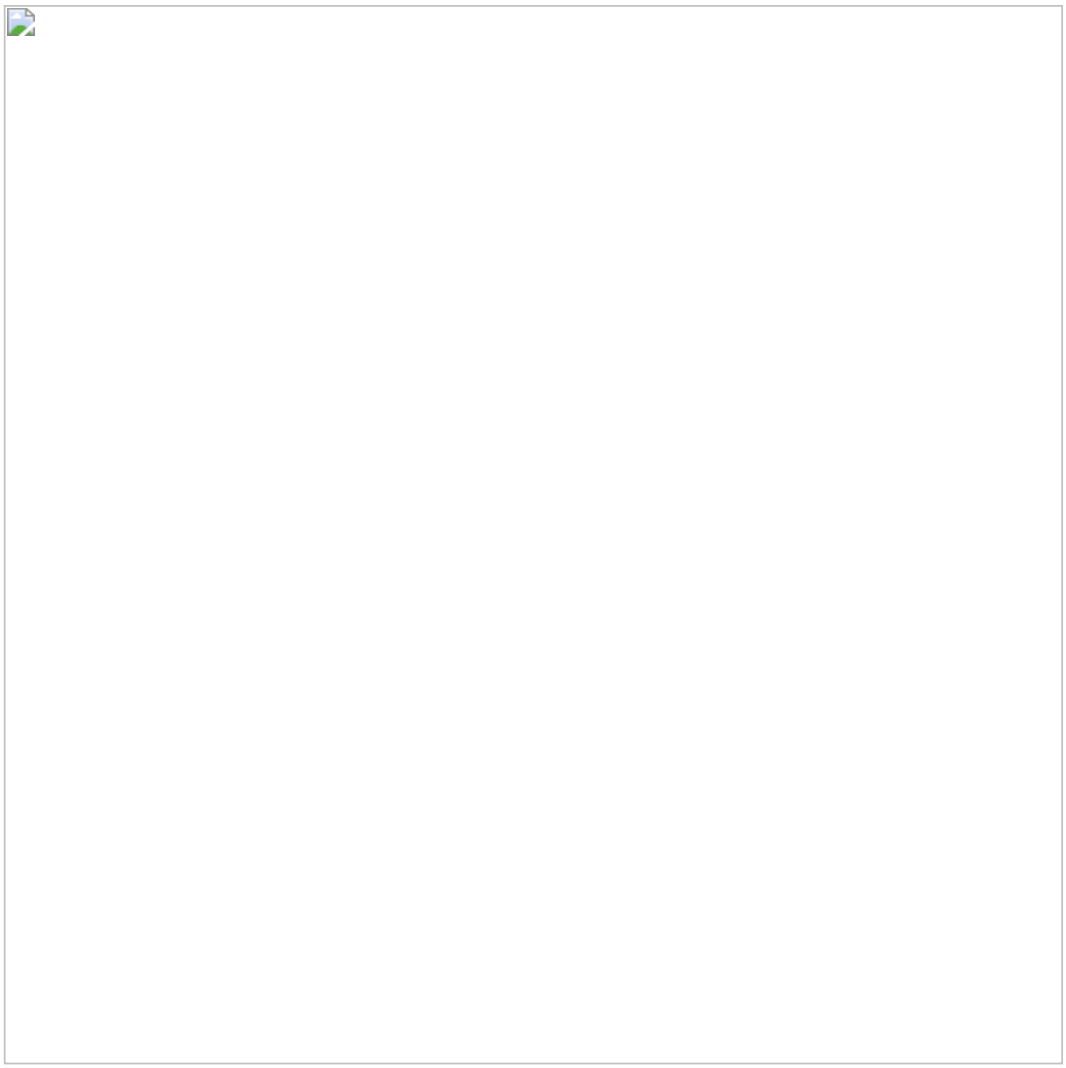


Response

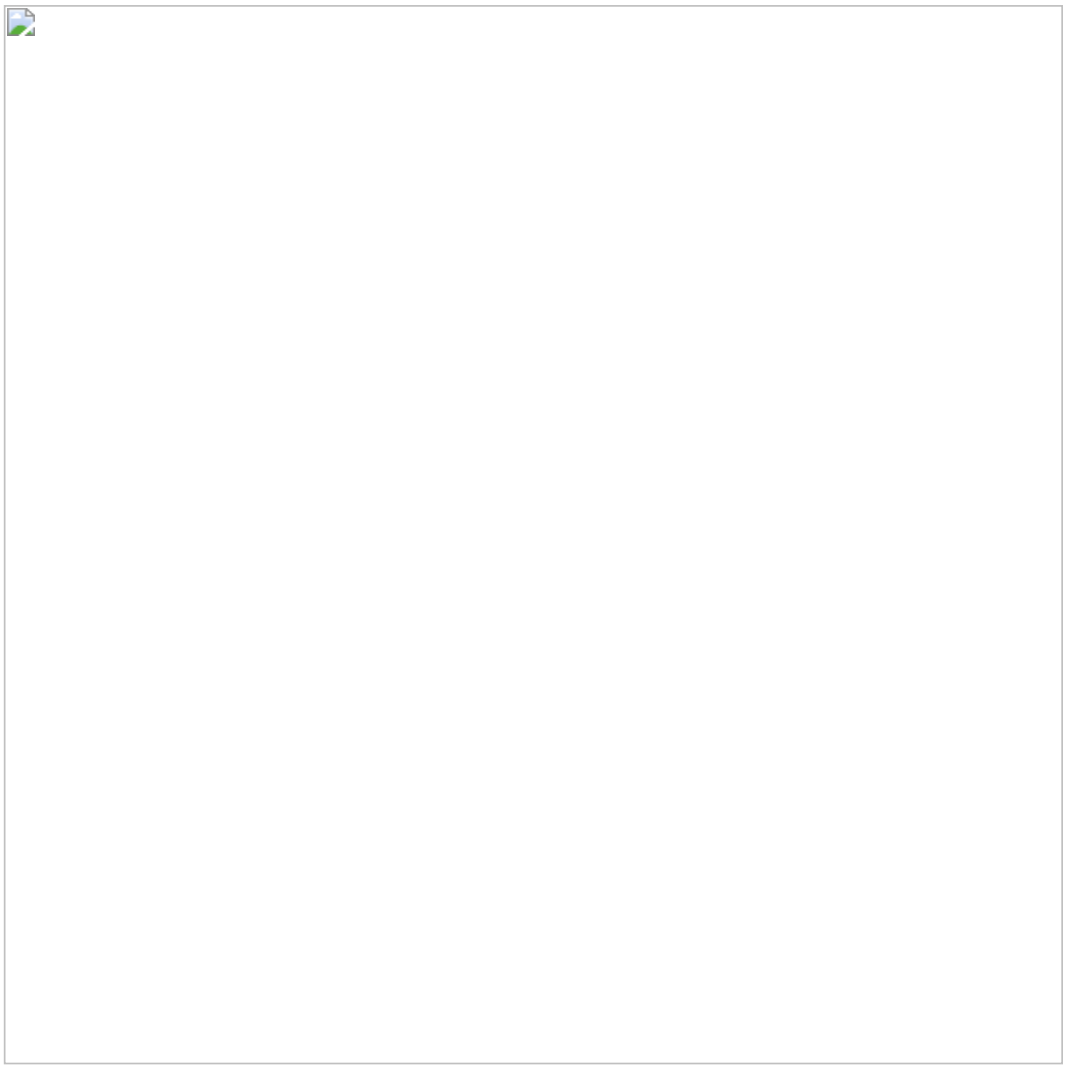
While prompt detection of DCShadow attacks is critical, it's not sufficient, especially since it means an adversary has compromised a highly privileged account. But simply disabling that account may be insufficient, because by the time you spot a DCShadow attack, the adversary likely has a host of other network resources and options available and in use.

Netwrix StealthDEFEND provides a wealth of automated response options that empower you to easily build an effective playbook for each anticipated threat or vulnerability. In the case of a DCShadow attack, the playbook should include the following steps:

1. Notify the right people in the organization that an attack has occurred and provide them with the information they need to respond effectively. Netwrix StealthDEFEND provides all critical details of the attack, including the perpetrator, source, and target. Moreover, it facilitates quick communication through easy integration with Slack, Microsoft Teams, ServiceNow and other systems using PowerShell or webhook facilities.



Block the perpetrating account or workstation from executing additional replication, authentication or other actions:



FAQs

What is DCShadow?

DCShadow is a command in the Mimikatz tool that enables an adversary to register a rogue domain controller and replicate malicious changes across the domain.

How does a DCShadow attack work?

An attacker registers their computer as a domain controller by making changes to the AD configuration schema and the workstation's SPN value. Then they can replicate changes, including changes to ensure their persistence in the domain.

How can DCShadow attacks be detected?

The best way to detect DCShadow attacks is to use an automated solution that continuously watches for the suspicious addition of domain controllers and monitors replication traffic for abnormal activity.

What is the best way to respond to a DCShadow attack?

When a DCShadow attack is detected, time is of the essence. Therefore, it's best to have an automated workflow that immediately reports the event to the security team and blocks the account from making any further changes in the domain.

Jeff Warren