Active Directory Federation Services (ADFS) and Kerberos

medium.com/@robert.broeckelmann/active-directory-federation-services-adfs-and-kerberos-f36c71e13be5

Robert Broeckelmann 9 августа 2019 г.

Robert Broeckelmann

Hemlock Ridge Preserve (8) /

While researching an upcoming blog post about Kerberos and Mobile, I needed to understand how Identity Providers (like ADFS or Ping Federate) use Kerberos (and possibly Kerberos Delegation) to perform authentication via username and password. This blog post captures what I found for ADFS.

The Kerberos protocol interaction between ADFS and the Domain Controller has two phases: user authentication and delegation to the ADFS service (obtains a service ticket for the ADFS service using the S4U2Self delegation sub-protocol).

Assumptions

- EC2AMAZ-A6G81N3.rcbj.net is the domain controller in this example.
- adfs-server1.rcbj.net is the ADFS server.

- fs.rcbj.net is an alias for the ADFS server.
- The user being authenticated is rcbj1@rcbj.net.

Authentication Stage

The authentication stage looks more-or-less the same as what happens when a user logs into a Windows workstation or server. I covered the details of this here.

For completeness, I include the messages that are exchanged between the ADFS server and the domain controller here.

The Kerberos messages described below were sent between the ADFS server and the Domain Controller (KDC) in response to the submission of the following SAMLRequest message to ADFS by a web application:

```
Request URL: Request Method: GET
```

This corresponds to the following SAML AuthnRequst message:

```
<?xml version="1.0" encoding="UTF-8"?><saml2p:AuthnRequest
xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
AssertionConsumerServiceURL="https://blogdemo.auth.us-west-
2.amazoncognito.com/saml2/idpresponse"
Destination="https://fs.rcbj.net/adfs/ls/"   ID="_ee8a0d1c-8277-414c-956b-
7c5b82c028b6"   IssueInstant="2018-09-07T04:53:15.267Z" Version="2.0">
<saml2:Issuer     xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:entity">urn:amazon:cognito:sp:us-west-2_WaPN6Bb7H</saml2:Issuer>
</saml2p:AuthnRequest>
```

This value can be extracted by using samltool.com to URL decode, base64 decode, and decompress the XML data structure.

For the details of what all that means, check out this older post.

Since there is no MSISAuth cookie for tracking the ADFS security session for the user that generated this SAMLRequest message, the user must be authenticated via Kerberos against the Domain Controller (the Kerberos KDC). On subsequent authentication requests in the same browser session, the MISISAuth cookie would be sent along with the SAMLRequest message. In this case, the Kerberos authentication protocol described in the rest of this section would not be needed.

The response to the first SAMLRequest message will be a login workflow that allows the user to enter a username and password. Those details are outside the scope of this blog post. Validation of the username and password that is provided as described in the rest of this blog post.

AS-REQ:

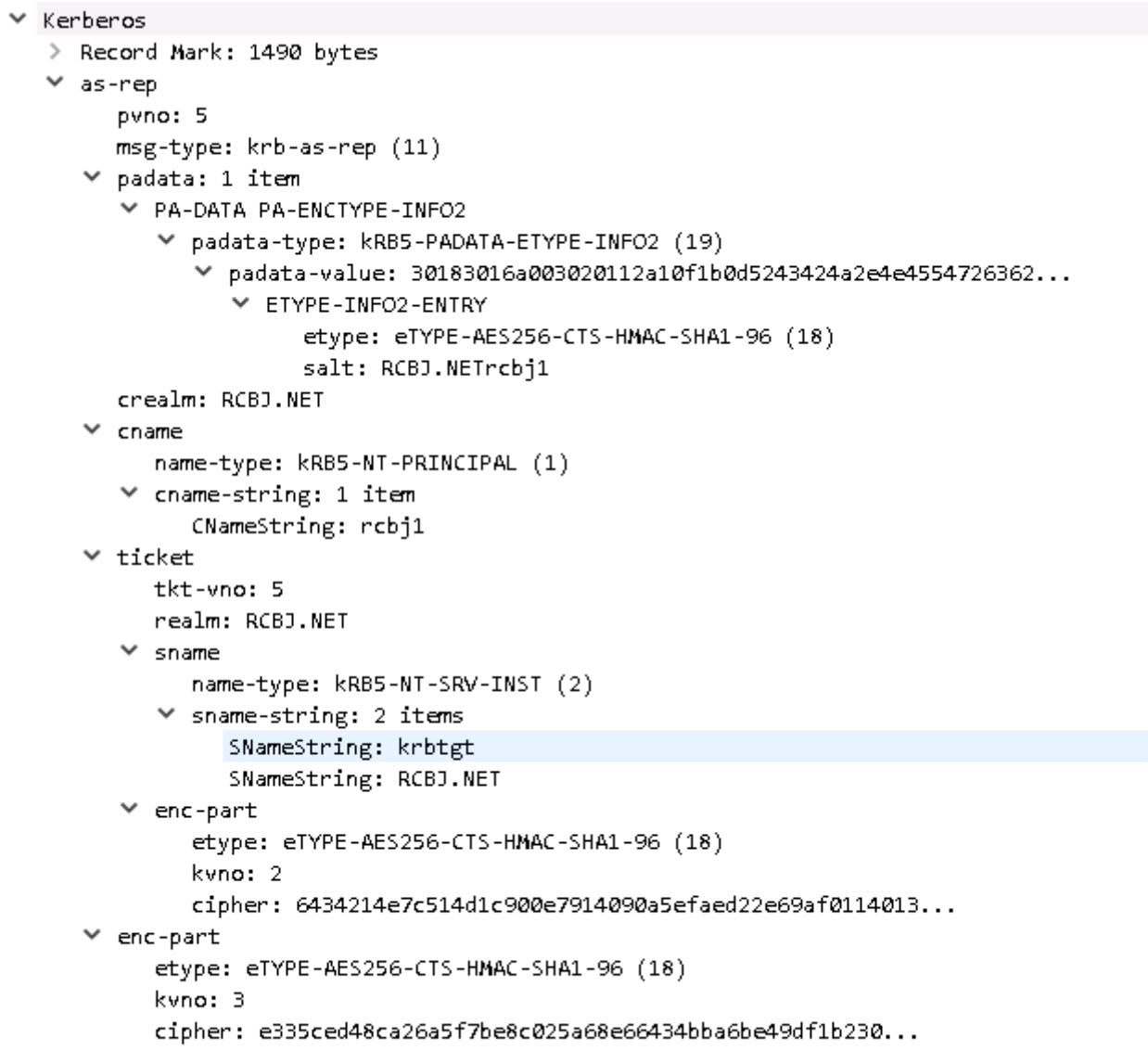
This is the AS-REQ message being sent to the domain controller’s authentication service.

```
Kerberos
  Record Mark: 226 bytes
  as-req
    pvno: 5
    msg-type: krb-as-req (10)
    padata: 1 item
      PA-DATA PA-PAC-REQUEST
        padata-type: kRB5-PADATA-PA-PAC-REQUEST (128)
          padata-value: 3005a0030101ff
              include-pac: True
    req-body
      Padding: 0
      kdc-options: 40810010 (forwardable, renewable, canonicalize, renewable-ok)
      cname
        name-type: kRB5-NT-ENTERPRISE-PRINCIPAL (10)
        cname-string: 1 item
          CNameString: rcbj1@rcbj.net
      realm: RCBJ.NET
      sname
        name-type: kRB5-NT-SRV-INST (2)
        sname-string: 2 items
          SNameString: krbtgt
          SNameString: RCBJ.NET
      till: 2037-09-13 02:48:05 (UTC)
      rtime: 2037-09-13 02:48:05 (UTC)
      nonce: 124990701
      etype: 6 items
        ENCTYPE: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
        ENCTYPE: eTYPE-AES128-CTS-HMAC-SHA1-96 (17)
        ENCTYPE: eTYPE-ARCFOUR-HMAC-MD5 (23)
        ENCTYPE: eTYPE-ARCFOUR-HMAC-MD5-56 (24)
        ENCTYPE: eTYPE-ARCFOUR-HMAC-OLD-EXP (-135)
        ENCTYPE: eTYPE-DES-CBC-MD5 (3)
      addresses: 1 item ADFS-SERVER1<20>
        HostAddress ADFS-SERVER1<20>
```

We can see that the user being authenticated is rcbj1@rcbj.net (or the rcbj1 user for the rcbj.net domain). We can also see that the name of the ADFS server is ADFS-SERVER1.

AS-REP:

The AS-REP message returned from the authentication service contains a TGT for rcbj1@rcbj.net.



TGS-REQ:

Next, the TGT and an authenticator are passed into Ticket Granting Service (TGS).

```
tgs-req
    pvno: 5
    msg-type: krb-tgs-req (12)
  padata: 2 items
    PA-DATA PA-TGS-REQ
      padata-type: kRB5-PADATA-TGS-REQ (1)
        padata-value: 6e8204cb308204c7a003020105a10302010ea20703050000...
          ap-req
              pvno: 5
              msg-type: krb-ap-req (14)
              Padding: 0
            > ap-options: 00000000
            ticket
                tkt-vno: 5
                realm: RCBJ.NET
              sname
                  name-type: kRB5-NT-SRV-INST (2)
                sname-string: 2 items
                    SNameString: krbtgt
                    SNameString: RCBJ.NET
              enc-part
                  etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
                  kvno: 2
                  cipher: 6434214e7c514d1c900e7914090a5efaed22e69af0114013...
            authenticator
                etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
                cipher: f1a457a1e70d13106d8e518c36d98f5d208b64aec204b99f...
    PA-DATA Unknown:167
      padata-type: Unknown (167)
        padata-value: 3009a0070305004000000
  req-body
      Padding: 0
    > kdc-options: 40810000 (forwardable, renewable, canonicalize)
      realm: RCBJ.NET
    sname
        name-type: kRB5-NT-SRV-HST (3)
      sname-string: 2 items
          SNameString: host
          SNameString: adfs-server1.rcbj.net
      till: 2037-09-13 02:48:05 (UTC)
      nonce: 124891528
    etype: 5 items
        ENCTYPE: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
        ENCTYPE: eTYPE-AES128-CTS-HMAC-SHA1-96 (17)
        ENCTYPE: eTYPE-ARCFOUR-HMAC-MD5 (23)
        ENCTYPE: eTYPE-ARCFOUR-HMAC-MD5-56 (24)
        ENCTYPE: eTYPE-ARCFOUR-HMAC-OLD-EXP (-135)
```

TGS-REP:

The TGS returns the following message with a Service Ticket for the adfs-server1.rcbj.net server (computer).

```
> Record Mark: 1439 bytes
v tgs-rep
    pvno: 5
    msg-type: krb-tgs-rep (13)
    crealm: RCBJ.NET
  v cname
      name-type: kRB5-NT-PRINCIPAL (1)
    v cname-string: 1 item
        CNameString: rcbj1
  v ticket
      tkt-vno: 5
      realm: RCBJ.NET
    v sname
        name-type: kRB5-NT-SRV-HST (3)
      v sname-string: 2 items
          SNameString: host
          SNameString: adfs-server1.rcbj.net
    v enc-part
        etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
        kvno: 1
        cipher: c9d2ca4f5dd4899a1fd51a1a0750e9765995ab1368586b4d...
  v enc-part
      etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
      cipher: 0c4e5b15112bb4f55f9f7bef303986901cb447dd73ef345c...
```

Delegation Call for ADFS Service Ticket:

Using the S4U2Self delegation sub-protocol, a service ticket describing the rcbj1@rcbj.net user for the adfs service is obtained.

The S4U2Self protocol is a Microsoft proprietary extension to Kerberos Delegation. It is described in detail here.

In order to make the S4U2Self call, the ADFS service must have already obtained a TGT. Those calls are not described here.

TGS-REQ

Per the S4U2Self protocol, the pre-authentication data includes a PA-FOR-USER data structure. This data structure looks like:

The following code defines the ASN.1 structure of the PA-FOR-USER padata type.

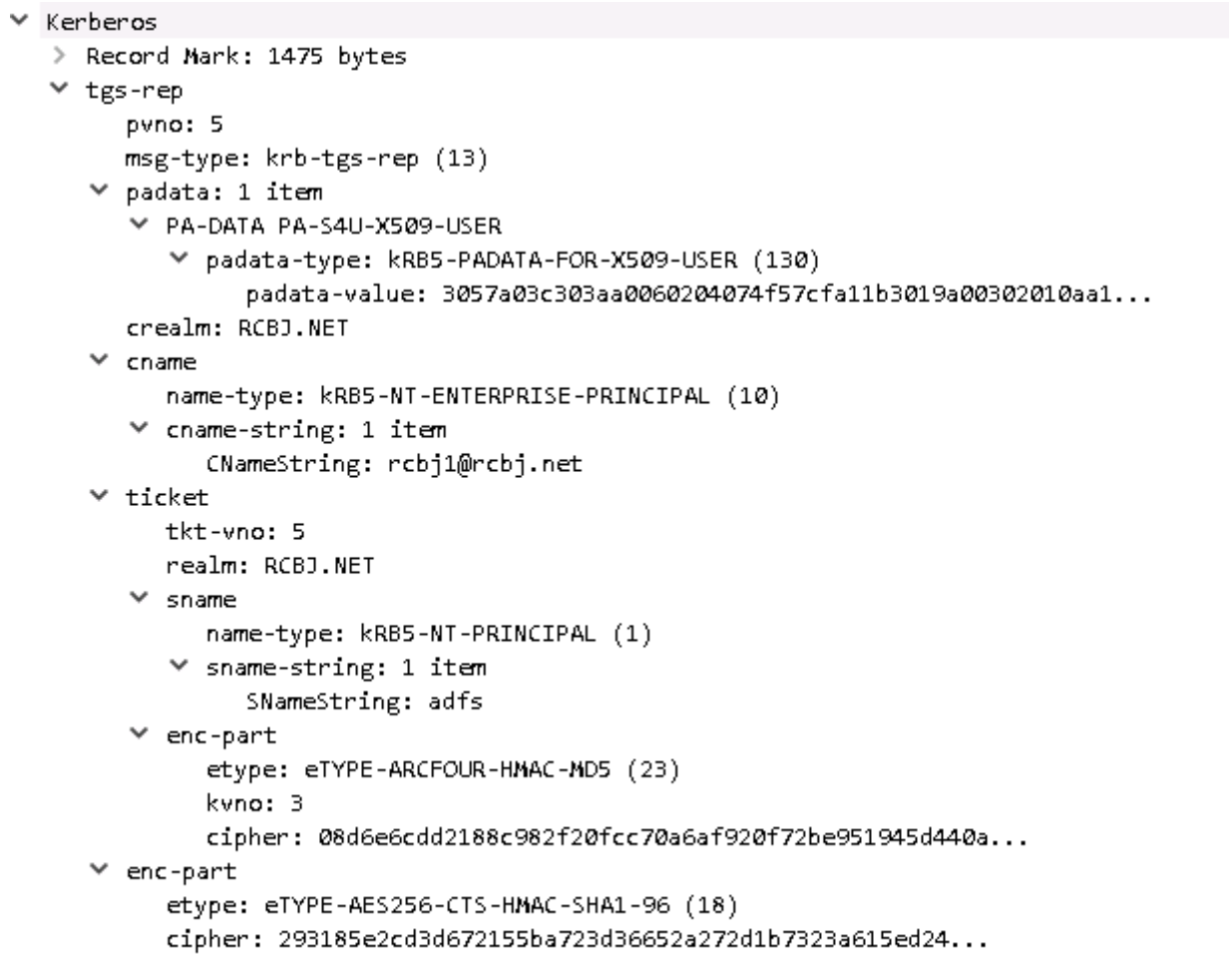
```
padata-type ::= PA-FOR-USER  — value 129padata-value ::= EncryptedData— PA-FOR-USER-ENCPA-FOR-USER-ENC ::= SEQUENCE {userName[0] PrincipalName,userRealm[1] Realm,cksum[2] Checksum,auth-package[3] KerberosString}
```

You can see this data structure in the padata field in the TGS-REQ message below.

The TGS-REQ message looks like.

TGS-REP:

The response looks like the following and includes a Service Ticket for the ADFS service that describes the user.



All of the tickets and session keys that have been produced during this exchange will be cached by the ADFS-Server1 server's Kerberos cache. We can read this cache with the klist command. We can use this PowerShell script to get the adfs TGT and the corresponding service tickets that have been generated.

```
Kerberos Tickets for LogonID 0x74bba*****************************Logon Type:
5Session ID: 0x74bbaAuth Method: KerberosCurrent LogonId is 0:0x21957b6Targeted
LogonId is 0:0x74bbaCached TGT:ServiceName : krbtgtTargetName (SPN) :
krbtgtClientName : adfsDomainName : RCBJ.NETTargetDomainName :
RCBJ.NETAltTargetDomainName: RCBJ.NETTicket Flags : 0x40e10000 -> forwardable
renewable initial pre_authent name_canonicalizeSession Key : KeyType 0x12 – AES-
256-CTS-HMAC-SHA1–96 : KeyLength 32–25 2b a5 21 cb 56 5f f1 f6 ad e9 35 73 7b 31
06 2e e8 8e 67 f7 bb 9f 24 a7 a5 50 fdd8 d0 43 48StartTime : 9/6/2018 23:23:25
(local)EndTime : 9/7/2018 9:23:25 (local)RenewUntil : 9/9/2018 2:27:40
(local)TimeSkew : + 0:00 minute(s)EncodedTticket : (size: 1003)0000 61 82 03 e7 30
82 03 e3:a0 03 02 01 05 a1 0a 1b a…0……..0010 08 52 43 42 4a 2e 4e 45:54 a2 1d 30
1b a0 03 02 .RCBJ.NET..0….0020 01 02 a1 14 30 12 1b 06:6b 72 62 74 67 74 1b 08
….0…krbtgt..0030 52 43 42 4a 2e 4e 45 54:a3 82 03 af 30 82 03 ab RCBJ.NET….0…0040
a0 03 02 01 12 a1 03 02:01 02 a2 82 03 9d 04 82 …….0050 03 99 6f 06 82 a8 18
3f:60 c7 b2 b7 2a 2b 18 e5 ..o….?`…*+.. : @{Client=adfs @ RCBJ.NET;
Server=krbtgt/RCBJ.NET @ RCBJ.NET; KerbTicket Encryption Type=AES-256-CTS-HMAC-
SHA1–96; Ticket Flags=0x40e10000 -> forwardable renewable initial pre_authent
name_canonicalize ; Start Time=9/6/2018 23:23:25 (local); End Time=9/7/2018
9:23:25 (local); Renew Time=9/9/2018 2:27:40 (local); Session Key Type=AES-256-
CTS-HMAC-SHA1–96} : @{Client=adfs @ RCBJ.NET; Server=krbtgt/RCBJ.NET @ RCBJ.NET;
KerbTicket Encryption Type=AES-256-CTS-HMAC-SHA1–96; Ticket Flags=0x60a10000 ->
forwardable forwarded renewable pre_authent name_canonicalize ; Start
Time=9/2/2018 2:27:49 (local); End Time=9/2/2018 12:27:40 (local); Renew
Time=9/9/2018 2:27:40 (local); Session Key Type=AES-256-CTS-HMAC-SHA1–96} :
@{Client=adfs @ RCBJ.NET; Server=ldap/EC2AMAZ-A6G81N3.rcbj.net @ RCBJ.NET;
KerbTicket Encryption Type=AES-256-CTS-HMAC-SHA1–96; Ticket Flags=0x40a50000 ->
forwardable renewable pre_authent ok_as_delegate name_canonicalize ; Start
Time=9/7/2018 1:38:28 (local); End Time=9/7/2018 9:23:25 (local); Renew
Time=9/9/2018 2:27:40 (local); Session Key Type=AES-256-CTS-HMAC-SHA1–96} :
@{Client=adfs @ RCBJ.NET; Server=ldap/EC2AMAZ-A6G81N3.rcbj.net/rcbj.net @
RCBJ.NET; KerbTicket Encryption Type=AES-256-CTS-HMAC-SHA1–96; Ticket
Flags=0x40a50000 -> forwardable renewable pre_authent ok_as_delegate
name_canonicalize ; Start Time=9/6/2018 23:28:02 (local); End Time=9/7/2018
9:23:25 (local); Renew Time=9/9/2018 2:27:40 (local); Session Key Type=AES-256-
CTS-HMAC-SHA1–96} : @{Client=adfs @ RCBJ.NET; Server=cifs/EC2AMAZ-A6G81N3.rcbj.net
@ RCBJ.NET; KerbTicket Encryption Type=AES-256-CTS-HMAC-SHA1–96; Ticket
Flags=0x40a50000 -> forwardable renewable pre_authent ok_as_delegate
name_canonicalize ; Start Time=9/2/2018 2:27:49 (local); End Time=9/2/2018
12:27:40 (local); Renew Time=9/9/2018 2:27:40 (local); Session Key Type=AES-256-
CTS-HMAC-SHA1–96}
```

The TGT for the ADFS service is the first entry. This is used to issue service tickets.

Now, these five service tickets are for krbtgt/DC, ldap, and cifs. So, this is not the tickets that were obtained for users. So, where are the tickets for the rcbj1@rcbj.net user?

This is the TGT and Service Tickets that were reported by klist after authenticating rcbj1@rcbj.net.

```
Kerberos Tickets for LogonID 0x5b05d****************************Logon Type:
10Session ID: 0x5b05dAuth Method: KerberosCurrent LogonId is 0:0x21957b6Targeted
LogonId is 0:0x5b05dCached TGT:ServiceName : krbtgtTargetName (SPN) :
krbtgtClientName : rcbj1DomainName : RCBJ.NETTargetDomainName :
RCBJ.NETAltTargetDomainName: RCBJ.NETTicket Flags : 0x40e10000 -> forwardable
renewable initial pre_authent name_canonicalizeSession Key : KeyType 0x12 – AES-
256-CTS-HMAC-SHA1–96 : KeyLength 32–6e b5 20 8b 89 53 88 23 33 48 60 82 a9 5b 16
32 1e ab 1a 6a d8 23 a5 64 0a c5 c6 bab9 7e 33 a4StartTime : 9/8/2018 1:14:40
(local)EndTime : 9/8/2018 11:14:40 (local)RenewUntil : 9/14/2018 5:44:38
(local)TimeSkew : + 0:00 minute(s)EncodedTicket : (size: 1052)0000 61 82 04 18 30
82 04 14:a0 03 02 01 05 a1 0a 1b a…0……..0010 08 52 43 42 4a 2e 4e 45:54 a2 1d 30
1b a0 03 02 .RCBJ.NET..0….0020 01 02 a1 14 30 12 1b 06:6b 72 62 74 67 74 1b 08
….0…krbtgt..0030 52 43 42 4a 2e 4e 45 54:a3 82 03 e0 30 82 03 dc RCBJ.NET….0…0040
a0 03 02 01 12 a1 03 02:01 02 a2 82 03 ce 04 82 ……. : @{Client=rcbj1 @
RCBJ.NET; Server=krbtgt/RCBJ.NET @ RCBJ.NET; KerbTicket Encryption Type=AES-256-
CTS-HMAC-SHA1–96; Ticket Flags=0x40e10000 -> forwardable renewable initial
pre_authent name_canonicalize ; Start Time=9/8/2018 1:14:40 (local); End
Time=9/8/2018 11:14:40 (local); Renew Time=9/14/2018 5:44:38 (local); Session Key
Type=AES-256-CTS-HMAC-SHA1–96} : @{Client=rcbj1 @ RCBJ.NET; Server=LDAP/EC2AMAZ-
A6G81N3.rcbj.net/rcbj.net @ RCBJ.NET; KerbTicket Encryption Type=AES-256-CTS-HMAC-
SHA1–96; Ticket Flags=0x40a50000 -> forwardable renewable pre_authent
ok_as_delegate name_canonicalize ; Start Time=9/8/2018 2:11:49 (local); End
Time=9/8/2018 11:14:40 (local); Renew Time=9/14/2018 5:44:38 (local); Session Key
Type=AES-256-CTS-HMAC-SHA1–96}
```

Ticket #0 is the TGT ticket for the rcbj1@rcbj.net user.

From the Kerberos calls above the following tickets were received:

- ,
- , service=adfs-server1.rcbj.net
- ,

That doesn't seem to match up with what we got out of klist. Maybe these tickets are cached separately from Windows by ADFS? Not sure about that part and those details are not critical for this post. I'll come back to it in a later blog post.

In theory, the adfs@rcbj.net service ticket that describes the rcbj1@rcbj.net user represents the authentication that is backing the SAML token generation. ADFS will generate a SAML token in response to the SAML AuthnRequest message. ADFS will also have a claims mapping configuration that will map user attributes from Active Directory (and possibly other sources) into the SAML Assertion claims list.

The end result is the SAML Assertion being returned as part of a SAMLResponse message.

Image: Hemlock Ridge Preserve (8) /