Cracking Kerberos TGS Tickets Using Kerberoast – Exploiting Kerberos to Compromise the Active Directory Domain

By Sean Metcalf

Published: 2015-12-31 · Archived: 2026-04-05 23:43:58 UTC

Microsoft's Kerberos implementation in Active Directory has been targeted over the past couple of years by security researchers and attackers alike. The issues are primarily related to the legacy support in Kerberos when Active Directory was released in the year 2000 with Windows Server 2000. This legacy support is enabled when using Kerberos RC4 encryption (RC4_HMAC_MD5) since the NTLM password hash is used extensively with this encryption type.

There are several Kerberos attacks that take advantage of Microsoft's legacy support in Active Directory. When Microsoft released Windows 2000 and Active Directory along with it, they needed to support Windows NT and Windows 95 which meant a wide variety of security (and less secure configurations). This support meant that Microsoft needed to support several different clients and enable them to speak Kerberos. The easy way to do this was to use the NTLM password hash as the Kerberos RC4 encryption private key used to encrypt/sign Kerberos tickets. Once the NTLM password hash is discovered, it can be used in a variety of ways, including re-compromising the Active Directory domain (think Golden Tickets & Silver Tickets).

RC4 Kerberos encryption is still supported even now, 15 years later. In fact, AES encryption wasn't available as an option on Windows until Windows Vista and Windows Server 2008. While AES Kerberos encryption is now used by default on the newer operating systems, there may still be significant use of RC4 Kerberos encryption on the network, involving some interesting network devices that have AES Kerberos encryption disabled by default.

With the introduction of AES as a Kerberos encryption option, Windows uses AES for hashing which is a break from traditional Windows password hashing methods. This means that while Kerberos RC4 encryption leveraged the NTLM password hash as encryption key, Kerberos AES encryption uses the AES hash to encrypt the Kerberos tickets. (in other words, when AES is the Kerberos encryption

Update:
Will @harmj0y Schroeder (blog.harmj0y.net) and I spoke at DerbyCon 6 in September, 2016 and demonstrated how Kerberoast works. The slides and video from our talk are now available. The other demos Will did during the talk are here.
All of the slides and most videos of my talks are on the Presentations page.
This article describes how Service Principal Names work and how to use Kerberoast to crack passwords offline.
Will also posted on how to Kerberoast without using Mimikatz.

Active Directory Kerberos Attacks:

There are several different types of Kerberos attacks ranging from recon (SPN Scanning), to offline service account password cracking (Kerberoast), to persistence (Silver & Golden Tickets).

Here are the most popular AD Kerberos attacks:

- SPN Scanning – finding services by requesting service principal names of a specific SPN class/type.
- Silver Ticket – forged Kerberos TGS service ticket
- Golden Ticket – forged Kerberos TGT authentication ticket
- MS14-068 Forged PAC Exploit – exploitation of the Kerberos vulnerability on Domain Controllers.
- Diamond PAC – blended attack type using elements of the Golden Ticket and the MS14-068 forged PAC.
- Skeleton Key In-memory Malware – malware "patches" the LSASS authentication process in-memory on Domain Controllers to enable a second, valid "skeleton key" password with which can be used to authenticate any domain account.

This post covers another type of Kerberos attack that involves Kerberos TGS service ticket cracking using Kerberoast. This information is based on the presentations I gave at several security conferences in 2015 (BSides, Shakacon, Black Hat, DEF CON, & DerbyCon) and Tim Medin's DerbyCon "Attacking Microsoft Kerberos Kicking the Guard Dog of Hades" presentation in 2014 (slides & video) where he released the Kerberoast Python TGS cracker.

SPN Scanning

Traditionally, attackers have performed recon using network port scanning, though this is usually not required on modern networks thanks to use of Active Directory and Kerberos. I have previously written about "SPN Scanning" for recon which involves requesting specific Service Principal Name (SPN) types from a Domain Controller (requires a user or computer account). For the attacker, one of the most useful SPN types to scan for is "SQL" to discover all SQL servers registered in Active Directory. All service types that leverage Kerberos authentication have SPNs registered in Active Directory since SPNs are required for Kerberos to work. First, let's review how Kerberos works.

I maintain a Service Principal Name (SPN) directory with the most common SPNs and what they are used for.

SPN Scanning for Services

```
Domain             : lab.adsecurity.org
ServerName         : adsMSSQL02.lab.adsecurity.org
Port               : 9834
Instance           :
ServiceAccountDN   : {CN=svc-adsSQLSA,OU=TestServiceAccounts,DC=lab,DC=adsecurity,DC=org}
OperatingSystem    : {Windows Server 2008 R2 Datacenter}
OSServicePack      : {Service Pack 1}
LastBootup         : 3/8/2015 1:07:25 AM
OSVersion          : {6.1 (7601)}
Description        : {Production SQL Server}
SrvAcctUserID      : svc-adsSQLSA
SrvAcctDescription : SQL Server Service Account
```

SPN Scanning for Service Accounts

```
Domain               : lab.adsecurity.org
UserID               : svc-SQLAgent01
PasswordLastSet      : 01/03/2015 18:42:01
LastLogon            : 12/29/2014 00:18:02
Description          :
SPNServers           : {ADSAPPSQL01.lab.adsecurity.org, ADSAPPSQL02.lab.adsecurity.org, ADSAPPSQL03.lab.adsecurity.org}
SPNTypes             : {MSSQLSvc}
ServicePrincipalNames : {MSSQLSvc/ADSAPPSQL01.lab.adsecurity.org:1433, MSSQLSvc/ADSAPPSQL02.lab.adsecurity.org:1433,
                       MSSQLSvc/ADSAPPSQL03.lab.adsecurity.org:1433}
```

Once the attacker has a list of Service Principal Names (SPNs) associated with service accounts, these SPNs can be used to request Kerberos TGS service tickets useful for offline TGS password cracking.

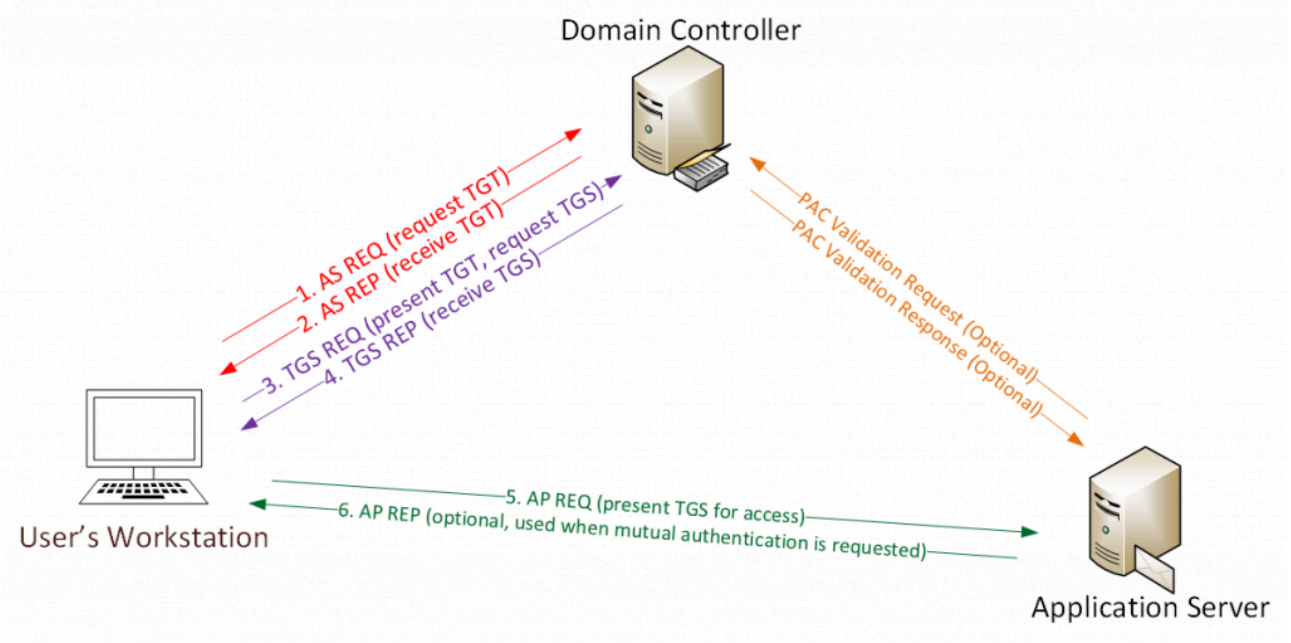
Note: Both of these screenshots are from PowerShell functions I wrote to perform SPN Scanning in my GitHub repository.

If you have the Active Directory PowerShell module installed, you can easily find all SPNs of a specific type with Get-ADObject

*get-adobject -filter {serviceprincipalname -like "*sql*"} -prop serviceprincipalname*

The AD PowerShell module quickly installs on Windows Server 2008R2 and newer:
import-module servermanager ; add-windowsfeature RSAT-AD-PowerShell

Kerberos Overview & Communication Process:



User logs on with username & password.

1a. Password converted to NTLM hash, a timestamp is encrypted with the hash and sent to the KDC as an authenticator in the authentication ticket (TGT) request (AS-REQ).
1b. The Domain Controller (KDC) checks user information (logon restrictions, group membership, etc) & creates Ticket-Granting Ticket (TGT).

2. The TGT is encrypted, signed, & delivered to the user (AS-REP). *Only the Kerberos service (KRBTGT) in the domain can open and read TGT data.*

3. The User presents the TGT to the DC when requesting a Ticket Granting Service (TGS) ticket (TGS-REQ). The DC opens the TGT & validates PAC checksum – If the DC can open the ticket & the checksum check out, TGT = valid. The data in the TGT is effectively copied to create the TGS ticket.

4. The TGS is encrypted using the target service accounts' NTLM password hash and sent to the user (TGS-REP).

5.The user connects to the server hosting the service on the appropriate port & presents the TGS (AP-REQ). The service opens the TGS ticket using its NTLM password hash.

6. If mutual authentication is required by the client (think MS15-011: the Group Policy patch from February that added UNC hardening).

Unless PAC validation is required (rare), the service accepts all data in the TGS ticket with no communication to the DC.

Cracking Service Account Passwords with Kerberoast:

Kerberoast can be an effective method for extracting service account credentials from Active Directory as a regular user without sending any packets to the target system. This attack is effective since people tend to create poor passwords. The reason why this attack is successful is that most service account passwords are the same length as the domain password minimum (often 10 or 12 characters long) meaning that even brute force cracking doesn't likely take longer than the password maximum password age (expiration). Most service accounts don't have passwords set to expire, so it's likely the same password will be in effect for months if not years. Furthermore, most service accounts are over-permissioned and are often members of Domain Admins providing full admin rights to Active Directory (even when the service account only needs to modify an attribute on certain object types or admin rights on specific servers).

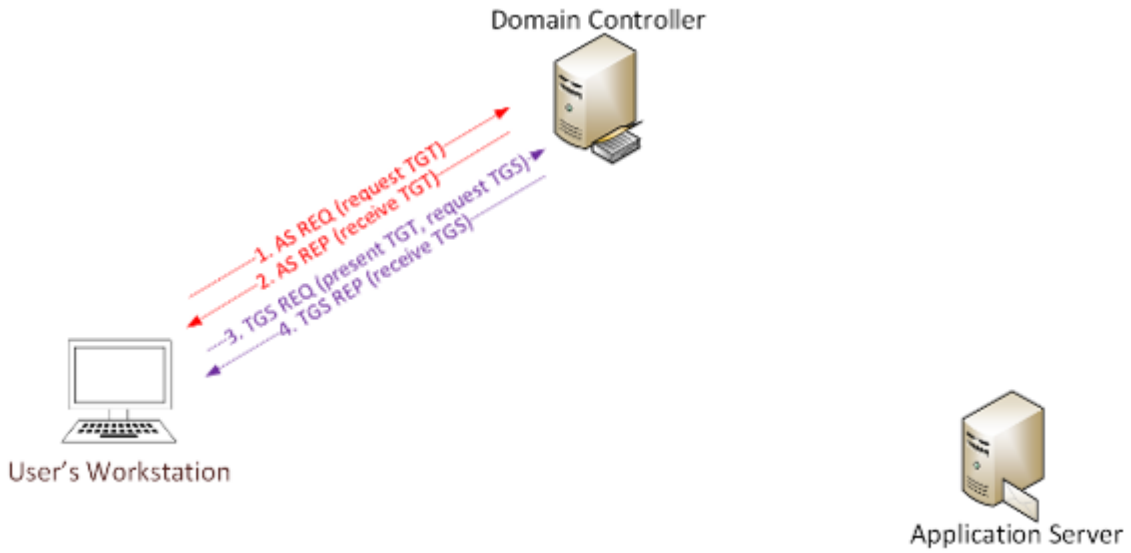
The most effective mitigation of this attack is ensuring service account passwords are longer than 25 characters. Managed Service Accounts and Group Managed Service Accounts are a good method to ensure that service account passwords are long, complex, and change regularly. A third party product that provides password vaulting is also a solid solution for managing service account passwords.

Note: This attack will not be successful when targeting services hosted by the Windows system since these services are mapped to the computer account in Active Directory which has an associated 128 character password which won't be cracked anytime soon.

This attack involves requesting a Kerberos service ticket(s) (TGS) for the Service Principal Name (SPN) of the target service account. This request uses a valid domain user's authentication ticket (TGT) to request one or several service tickets for a target service running on a server. The Domain Controller doesn't track if the user ever actually connects to these resources (or even if the user has access). The Domain Controller looks up the SPN in Active Directory and encrypts the ticket using the service account associated with the SPN in order for the service to validate user access. The encryption type of the requested Kerberos service ticket is RC4_HMAC_MD5 which means the service account's NTLM password hash is used to encrypt the service ticket. This means that Kerberoast can attempt to open the Kerberos ticket by trying different NTLM hashes and when the ticket is successfully opened, the correct service account password is discovered.

No elevated rights are required to get the service tickets and no traffic is sent to the target.

Tim Medin released the Kerberoast Python TGS cracker and discussed these methods at DerbyCon 2014.



An attacker can crack service account passwords without ever getting admin access to the server or the network. The attacker gets a foothold on a computer & Requests TGS tickets for several services with service accounts. Exports the TGS tickets from memory, saves them to files, & uploads to a website or webservice (Google Drive). An attacker owned computer on the internet grabs these files & runs Kerberoast against them until it identifies the correct NTLM password hash that will open one. Success in opening a TGS means the service account password was found!

Note that this attack can also work by sniffing network traffic and grabbing Kerberos TGS tickets encrypted using RC4_HMAC_MD5 off the wire.

I'll walk through this attack using a PowerShell script I wrote called Discover-PSMSSQLServers.ps1. This script discovers all the SQL servers in the domain/forest and identifies the associated service account. If it has a domain user account, it is very likely the associated password is not very strong, so that account is targeted.

1. SPN scan for SQL servers with service accounts.

```
Domain               : lab.adsecurity.org
ServerName           : adsMSSQL02.lab.adsecurity.org
Port                 : 9834
Instance             :
ServiceAccountDN     : {CN=svc-adsSQLSA,OU=TestServiceAccounts,DC=lab,DC=adsecurity,DC=org}
OperatingSystem      : {Windows Server 2008 R2 Datacenter}
OSServicePack        : {Service Pack 1}
LastBootup           : 3/8/2015 1:07:25 AM
OSVersion            : {6.1 (7601)}
Description          : {Production SQL Server}
SrvAcctUserID        : svc-adsSQLSA
SrvAcctDescription   : SQL Server Service Account
```

```
Domain               : lab.adsecurity.org
UserID               : svc-SQLAgent01
PasswordLastSet      : 01/03/2015 18:42:01
LastLogon            : 12/29/2014 00:18:02
Description          :
SPNServers           : {ADSAPPSQL01.lab.adsecurity.org, ADSAPPSQL02.lab.adsecurity.org, ADSAPPSQL03.lab.adsecurity.org}
SPNTypes             : {MSSQLSvc}
ServicePrincipalNames : {MSSQLSvc/ADSAPPSQL01.lab.adsecurity.org:1433, MSSQLSvc/ADSAPPSQL02.lab.adsecurity.org:1433,
                       MSSQLSvc/ADSAPPSQL03.lab.adsecurity.org:1433}
```

2. After identifying the target, we use PowerShell to request the service ticket for this Service Principal Name (SPN).

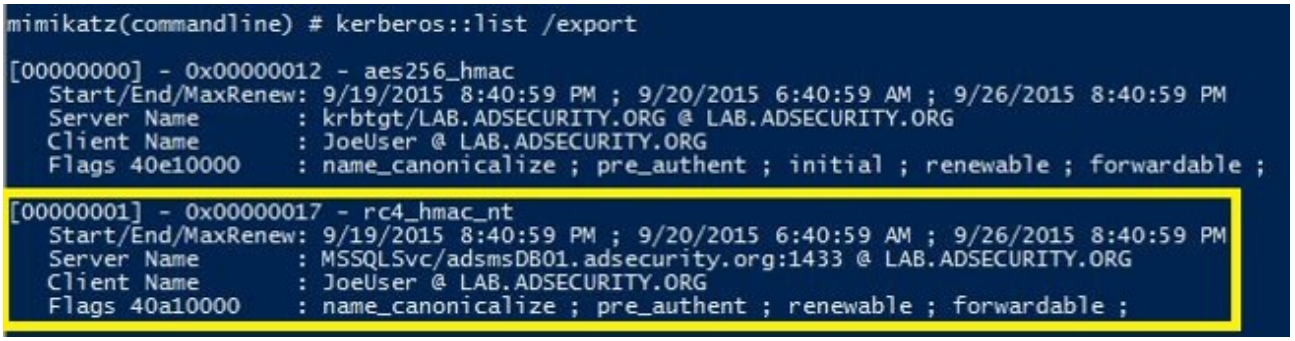
Note that the service ticket requested has the RC4 encryption type.

```
PS C:\> Add-Type -AssemblyName System.IdentityModel
PS C:\> New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken `
>>  -ArgumentList 'MSSQLSvc/adsmsDB01.adsecurity.org:1433'
>>


Id                   : uuid-2262c868-429e-4581-ae12-8e6ce2c0aa22-3
SecurityKeys         : {System.IdentityModel.Tokens.InMemorySymmetricSecurityKey}
ValidFrom            : 9/20/2015 12:40:59 AM
ValidTo              : 9/20/2015 10:40:59 AM
ServicePrincipalName : MSSQLSvc/adsmsDB01.adsecurity.org:1433
SecurityKey          : System.IdentityModel.Tokens.InMemorySymmetricSecurityKey


PS C:\> klist

Current LogonId is 0:0xbf51b3

Cached Tickets: (2)

#0>     Client: JoeUser @ LAB.ADSECURITY.ORG
        Server: krbtgt/LAB.ADSECURITY.ORG @ LAB.ADSECURITY.ORG
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent name_canonicalize
        Start Time: 9/19/2015 20:40:59 (local)
        End Time:   9/20/2015 6:40:59 (local)
        Renew Time: 9/26/2015 20:40:59 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96

#1>     Client: JoeUser @ LAB.ADSECURITY.ORG
        Server: MSSQLSvc/adsmsDB01.adsecurity.org:1433 @ LAB.ADSECURITY.ORG
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40a10000 -> forwardable renewable pre_authent name_canonicalize
        Start Time: 9/19/2015 20:40:59 (local)
        End Time:   9/20/2015 6:40:59 (local)
        Renew Time: 9/26/2015 20:40:59 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
```

Looking at a packet capture, we can see the Kerberos communication and note that the ticket is RC4-HMAC-MD5.

```
    62 11.0397850 172.16.11.12      172.16.11.101      KRB5     1594 TGS-REP
⊞ Frame 62: 1594 bytes on wire (12752 bits), 1594 bytes captured (12752 bits) on interface 0
⊞ Ethernet II, Src: Microsof_17:c1:98 (00:15:5d:17:c1:98), Dst: Microsof_17:c1:a6 (00:15:5d:17:c1:a6)
⊞ Internet Protocol Version 4, Src: 172.16.11.12 (172.16.11.12), Dst: 172.16.11.101 (172.16.11.101)
⊞ Transmission Control Protocol, Src Port: 88 (88), Dst Port: 51087 (51087), Seq: 1, Ack: 1592, Len: 1540
⊟ Kerberos
  ⊞ Record Mark: 1536 bytes
  ⊟ tgs-rep
      pvno: 5
      msg-type: krb-tgs-rep (13)
      crealm: LAB.ADSECURITY.ORG
    ⊟ cname
        name-type: kRB5-NT-PRINCIPAL (1)
      ⊟ name-string: 1 item
          KerberosString: JoeUser
    ⊟ ticket
        tkt-vno: 5
        realm: LAB.ADSECURITY.ORG
      ⊟ sname
          name-type: kRB5-NT-SRV-INST (2)
        ⊟ name-string: 2 items
            KerberosString: MSSQLSvc
            KerberosString: adsmsDB01.adsecurity.org:1433
      ⊟ enc-part
          etype: eTYPE-ARCFOUR-HMAC-MD5 (23)
          kvno: 2
          cipher: a0c70bf983f16b744fdd06e0ad69fc7710d77afb2dd8d790...
```

3. Once the ticket is is received by the client, we can use Mimikatz (or other) to export all Kerberos tickets in the user's memory space without elevated rights.



4. After exporting the service ticket to a file, that file can be sent to our attacker machine running Kali Linux with Kerberoast. Depending on our wordlist file, we may be able to crack the service account's password associated with the ticket (file).

The attacker now knows the service account username and password valid on a server (or servers) and likely has administrator rights.

Since service accounts are typically over-permissioned in many enterprises, often with weak passwords, this is an easy way for an attacker to go from domain user to domain admin.

Mitigation:

Ensure all service accounts (user accounts with Service Principal Names) have long, complex passwords greater than 25 characters, preferably 30 or more. This makes cracking these password far more difficult. Service Accounts with elevated AD permissions should be the focus on ensuring they have long, complex passwords. Ensure all Service Account passwords are changed regularly (at least once a year). If possible use group managed service accounts which have random, complex passwords (>100 characters) and are managed automatically by Active Directory.

Detection:

Detection is a lot tougher since requesting service tickets (Kerberos TGS tickets) happens all the time when users need to access resources.
Looking for TGS-REQ packets with RC4 encryption is probably the best method, though false positives are likely. Monitoring for numerous Kerberos service ticket requests in Active Directory is possible by enabling Kerberos service ticket request monitoring ("Audit Kerberos Service Ticket Operations") and searching for users with excessive 4769 events (Eventid 4769 "A Kerberos service ticket was requested").

References:

- Sean Metcalf's Presentations on Active Directory Security

- Kerberoast (GitHub)
- Tim Medin's DerbyCon "Attacking Microsoft Kerberos Kicking the Guard Dog of Hades" presentation in 2014 (slides & video).
- My GitHub repository

(Visited 133,635 times, 4 visits today)
