Kerberos Attacks Exploited: Part 1

redfoxsec.com/blog/attacking-kerberos-part-1

Shashi Kant Prasad December 21, 2022

Kerberos Attacks- Part 1

- December 21, 2022
- Active Directory
- Shashi Kant Prasad

As discussed in the Active Directory Basics blog, Kerberos is an authentication mechanism used to authenticate users and services.

The two main components of Kerberos are:

- Authentication Server (AS), which authenticates user and grants Ticket Granting Ticket (TGT)
- Ticket Granting Server (TGS), which issues the service tickets (TGS)

The main goal of an attacker is to gain access to the most privileged account in a network to extract all the data from it. Attackers compromise an arbitrary system in the network and move laterally with the credentials exfiltrated, escalating privileges to achieve their goals.

In a Windows environment, the most targeted account for an attacker is the one with Domain Administrator privileges. Once these accounts are compromised, an attacker has complete and unrestricted control over the infrastructure and can generate Kerberos tickets to get unauthorized access.

Kerberos is the point of authentication and authorization of users and services, which becomes another primary target for attacks. Once compromised, an attacker can forge and use TGTs through the domain, thus giving them unlimited access.

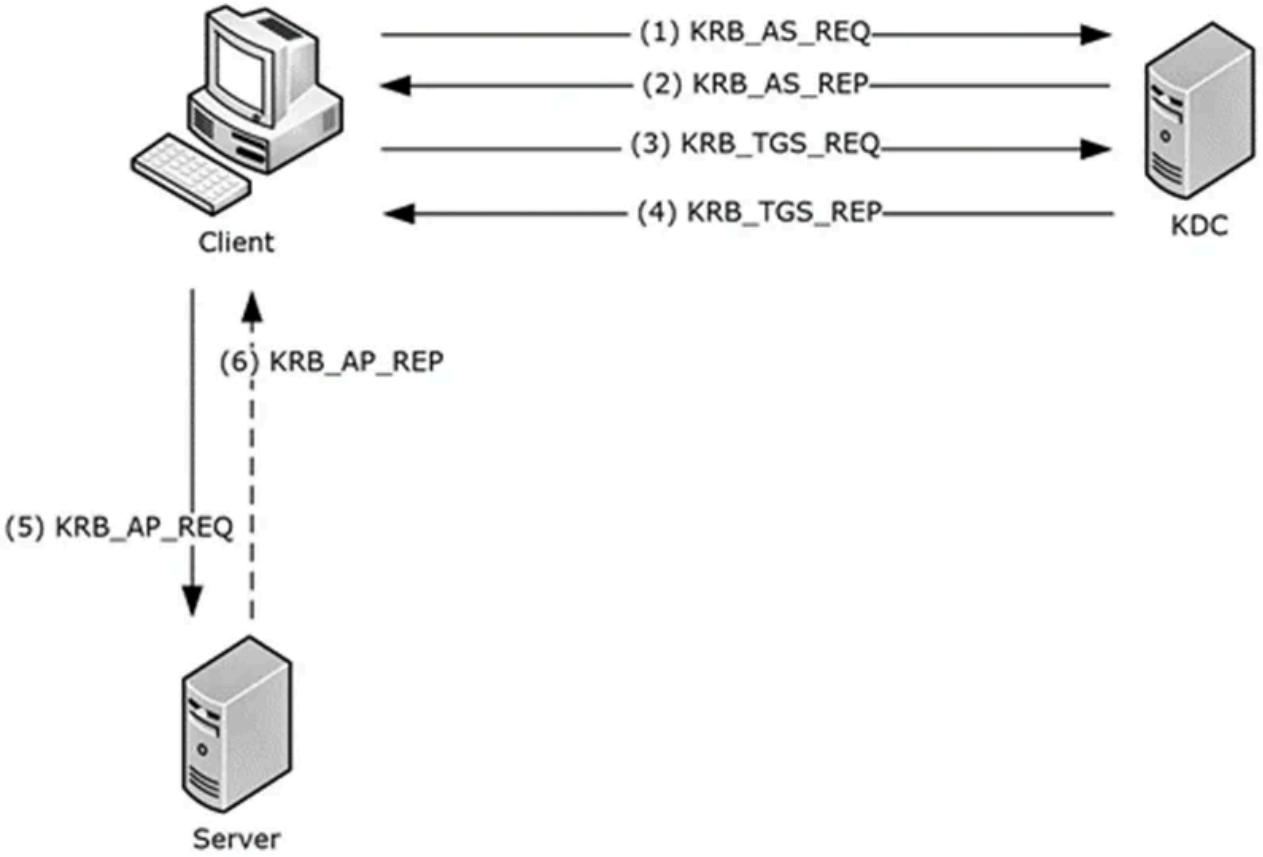
Kerberos Authentication

Let us look at the authentication mechanism used by Kerberos along with an attack that can exploit it called Skeleton Key Attack.

This is a post-exploitation attack, wherein once an attacker has taken control of a domain controller, deploys this attack to stay persistent in the network. The attack uses a piece of code that injects itself into the memory, more specifically into the LSASS process, and creates a master password that can be used with any account in the domain.

The existing account passwords continue to work, so domain credentials are not being modified. This makes the attack harder to detect. Since the malware is run in the memory, rebooting the domain controller will remove it, which rarely happens in an enterprise.

Before going into depth about how this malware facilitates the attack, let us understand how the Authentication server (AS) works.



Below are the packets captured by Wireshark of a Kerberos authentication at a domain controller for a domain user. We will be observing each packet to understand the mechanism of AS better.

No.	Time	Source	Destination	Protocol	Length	Info
1	495 147.921776	10.0.2.15	10.0.2.2	KRB5	283	AS-REQ
2	496 147.923520	10.0.2.2	10.0.2.15	KRB5	237	KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
3	503 147.928114	10.0.2.15	10.0.2.2	KRB5	363	AS-REQ
4	504 147.928556	10.0.2.2	10.0.2.15	KRB5	1570	AS-REP

1) Once the user (client machine) logs in and enters their password, the client machine sends an AS-REQ to the Authentication Server (Domain Controller), which contains the username and the encryption keys of the user along with encryption protocols supported by the client machine such as AES256, AES128, DES, RC4 and so on.

```
> Record Mark: 225 bytes
v as-req
    pvno: 5
    msg-type: krb-as-req (10)
  v padata: 1 item
    v PA-DATA pA-PAC-REQUEST
      v padata-type: pA-PAC-REQUEST (128)
        > padata-value: 3005a0030101ff
  v req-body
      Padding: 0
    > kdc-options: 40810010
    > cname
      realm: FOX.LOCAL
    > sname
      till: Sep 12, 2037 19:48:05.000000000 Pacific Daylight Time
      rtime: Sep 12, 2037 19:48:05.000000000 Pacific Daylight Time
      nonce: 621110978
    v etype: 6 items
        ENCTYPE: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
        ENCTYPE: eTYPE-AES128-CTS-HMAC-SHA1-96 (17)
        ENCTYPE: eTYPE-ARCFOUR-HMAC-MD5 (23)
        ENCTYPE: eTYPE-ARCFOUR-HMAC-MD5-56 (24)
        ENCTYPE: eTYPE-ARCFOUR-HMAC-OLD-EXP (-135)
        ENCTYPE: eTYPE-DES-CBC-MD5 (3)
```

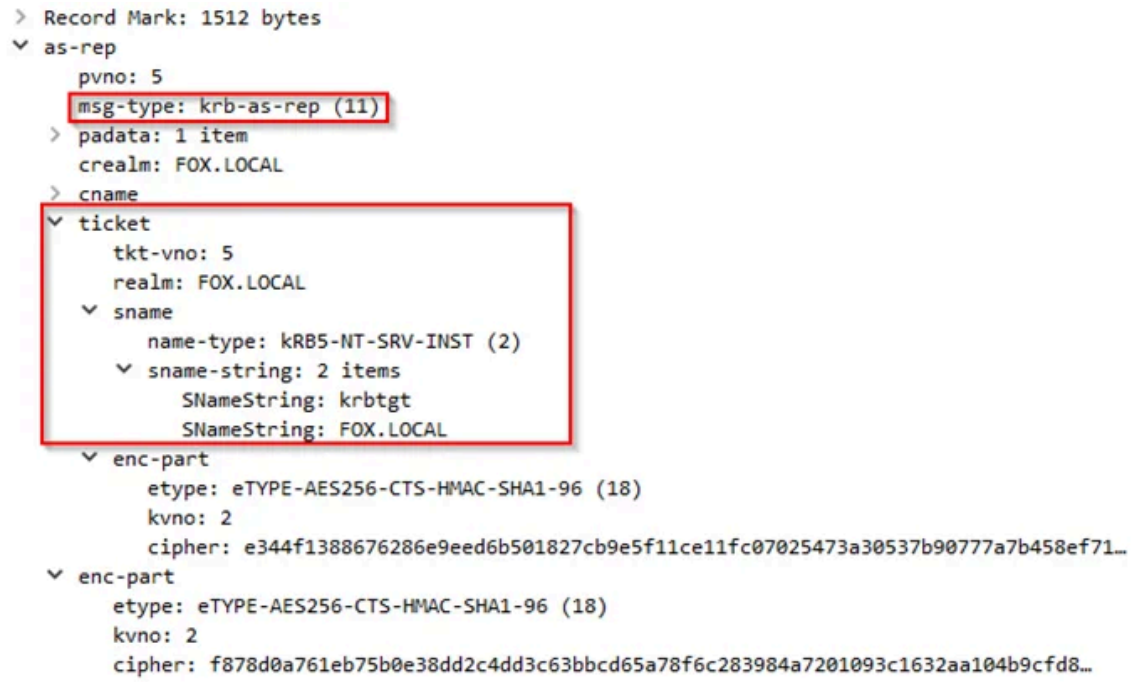
2) The Authentication server checks the encryption protocols of the user keys stored in its database and responds with a PRE-AUTH-REQ, which provides all the different encryption protocols it can communicate with. It also includes salt for hashing, the realm name (domain name) and a username to make each encryption key unique.

```
> Record Mark: 179 bytes
v krb-error
    pvno: 5
    msg-type: krb-error (30)
    stime: Nov  2, 2022 12:38:11.000000000 Pacific Daylight Time
    susec: 862821
    error-code: eRR-PREAUTH-REQUIRED (25)
    realm: FOX.LOCAL
  > sname
  v e-data: 3051302ea103020113a22704253023301aa003020112a1131b11464f582e4c4f43414c76…
    v PA-DATA pA-ETYPE-INFO2
      v padata-type: pA-ETYPE-INFO2 (19)
        v padata-value: 3023301aa003020112a1131b11464f582e4c4f43414c7668656c73696e673005a0030201…
          v ETYPE-INFO2-ENTRY
              etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
              salt: FOX.LOCALvhelsing
          v ETYPE-INFO2-ENTRY
              etype: eTYPE-ARCFOUR-HMAC-MD5 (23)
    v PA-DATA pA-ENC-TIMESTAMP
      > padata-type: pA-ENC-TIMESTAMP (2)
    v PA-DATA pA-PK-AS-REQ
      > padata-type: pA-PK-AS-REQ (16)
    v PA-DATA pA-PK-AS-REP-19
      > padata-type: pA-PK-AS-REP-19 (15)
```

3) The client chooses an encryption protocol, then encrypts the timestamp using the user password hash with the selected protocol, and sends an AS-REQ PA-ENC-TS to the authentication server.

```
> Record Mark: 305 bytes
v as-req
    pvno: 5
    msg-type: krb-as-req (10)
  v padata: 2 items
    v PA-DATA pA-ENC-TIMESTAMP
      v padata-type: pA-ENC-TIMESTAMP (2)
        v padata-value: 3041a003020112a23a04380520ca0c722b4397efe3a9b776ea299e3a73b296e9ac50f996…
            etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
            cipher: 0520ca0c722b4397efe3a9b776ea299e3a73b296e9ac50f996b4d01e94eaac9e4bcd6e70…
    > PA-DATA pA-PAC-REQUEST
  > req-body
```

4) The AS decrypts the timestamp with the agreed-upon encryption protocol using the user's password hash stored in its database. If it is successful, the AS sends an AS-REP containing the Ticket Granting Ticket (TGT) encrypted with the user's password to the client.



Forged Key: Skeleton Key Attack

Now that we know how the Authentication Server grants the TGT let us further look into how this attack works.

First, we must understand the difference between AES and RC4 encryption and how RC4 can help attackers to remain persistent in the domain easily. The main difference between these encryption protocols is that AES supports **Salting** (adding a random string (here the username) to the password before encrypting it so that it remains unique, even if the passwords are the same for different users) and **Key stretching** (reiterating through multiple rounds of hashing to provide more security) while RC4 does not.

For an attacker to remain persistent in the domain while using AES, they would have to recalculate the encryption key of the master password every time since it is salted with the username; thereby different for each user. This requires a lot of computing, memory and time. So, the attackers downgrade the Kerberos encryption mechanism from AES to RC4 to create the master password without any overhead.

The code does this by patching the in-memory function "**Kerberos-Newer-Keys**" which is a function where the Authentication Server checks if the user has AES keys; and returns it as false, thereby forcing the AS to downgrade to RC4 protocol. This renders all the AES keys stored in the database useless.

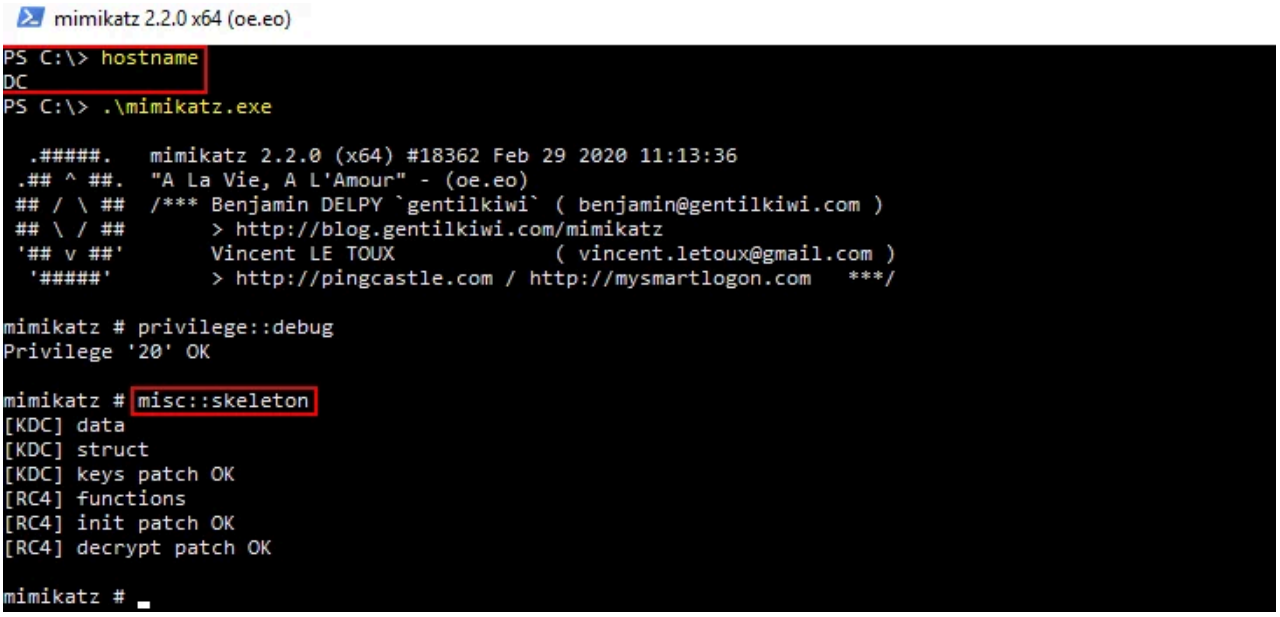
It also patches the "**decrypt**" function, essentially used to decrypt the timestamp with the user key stored in the DC to validate the user. It does so in a way that if the decrypt function fails the first time due to a bad password, it calls it again, but this time replacing the user key with the master password hash (skeleton key) from the DC for decryption and checking.

This is done so that the user can still use their original password and authenticate. They can also use the master password to authenticate, as the decrypt function will fail the first time due to a bad user password but will succeed the second time as it gets replaced with the hash of the master password at the DC. So, once this malware infects the DC, a user will have 2 RC4 keys in the database; one is the original password key, and the other is the skeleton key.

Now that's out of the way, let us see the skeleton key in action. We will be performing this attack on a Domain Controller using Mimikatz.

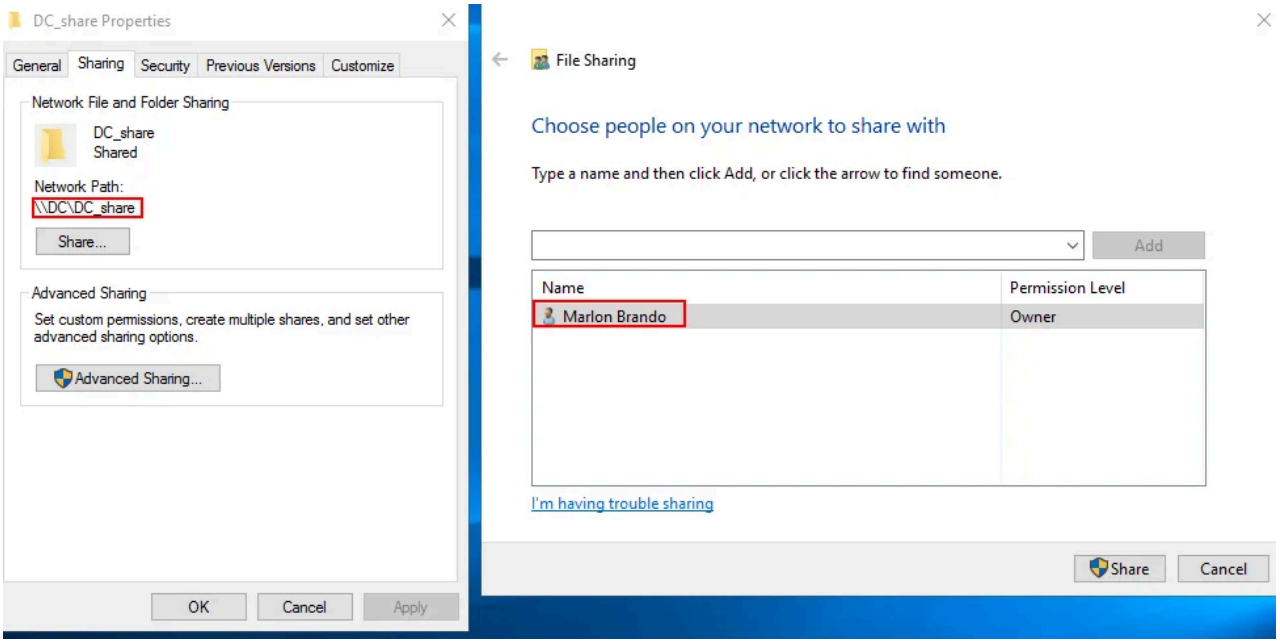
Mimikatz has a module called "misc::skeleton" which will create a master password, "mimikatz" for all users in the domain.

```
privilege::debug
misc::skeleton
```



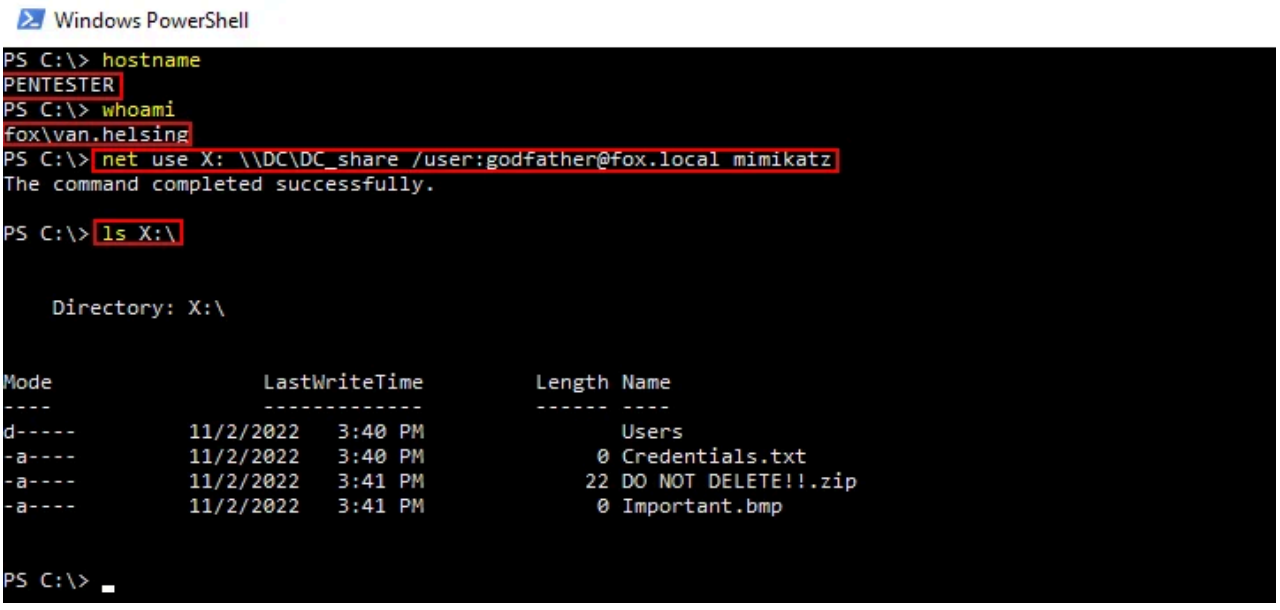
Let us confirm this by accessing a share only accessible to the domain user.

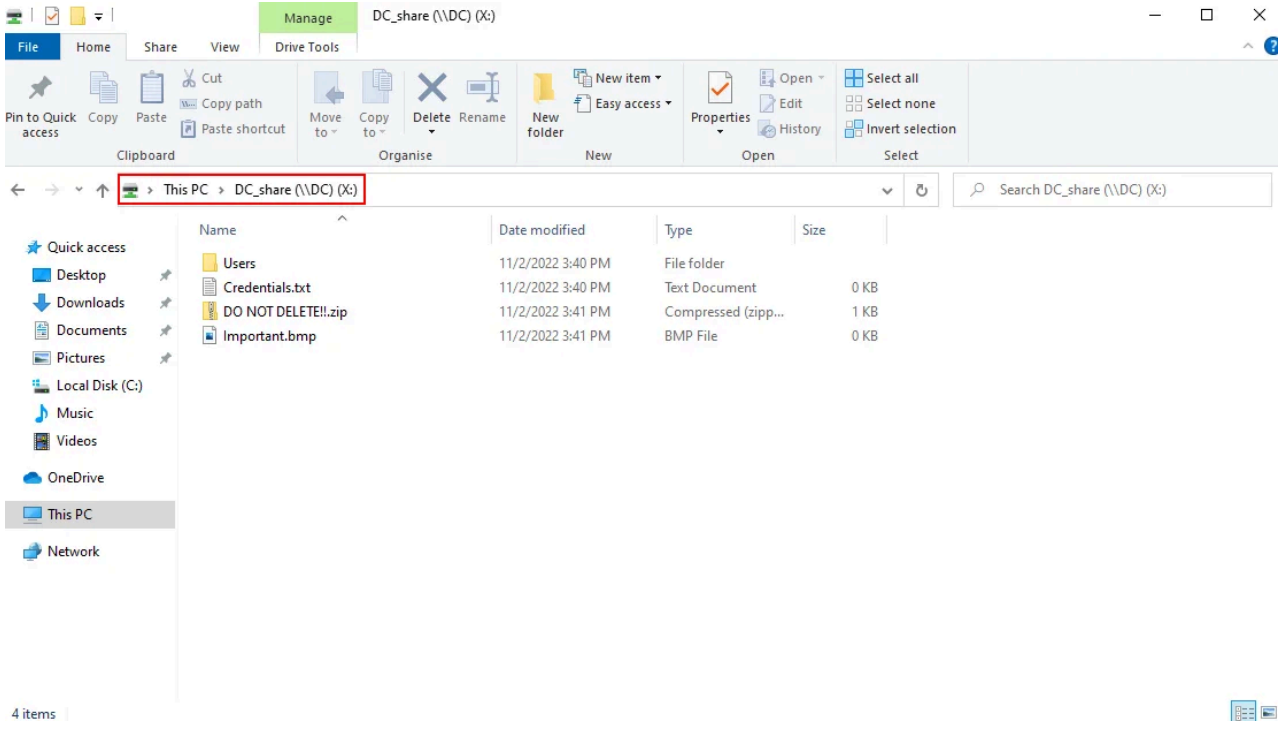
We have created a share in the Domain Controller (DC) named DC_share, which is only accessible by the Domain Admin Marlon Brando (GODFATHER).



Now we can try to mount and access the share as the user Van Helsing from the host PENTESTER using the net use command and the master password “mimikatz” for the domain admin GODFATHER.

```
net use X: \\DC\DC_share /user:godfather@fox.local mimikatz
```





We successfully authenticated as the Domain Admin godfather and mapped the share.

Kerberos Authorization

Till now, we have been discussing the authentication mechanism of Kerberos and how an attacker can exploit it. Now we will move on to the authorization part of Kerberos.

In Windows, Authorization is implemented using a Privilege Attribute Certificate (PAC), which contains all the authorization data for domain objects, i.e. membership groups, as well as their permissions and privileges.

After authentication by the AS server, the PAC of the user is embedded with the TGT and sent back to the user. When the user request access to a specific service, the PAC gets transferred to the TGS and validated by the service in the AS_REP response. Later when the user requests a TGS, the PAC is copied from the TGT to the TGS and validated. That is how a user gets authorized to access a service.

To maintain the integrity of the TGT, it is encrypted with the **krbtgt key,** i.e. the secret key of the Kerberos account. Since the PAC is embedded within the TGT, it is also encrypted along with the TGT. It is also signed twice by the TGT, as shown.

Thus, the attacker cannot modify the PAC or the TGT without obtaining the krbtgt key.

This brings us to the Golden Ticket attack, wherein the attacker steals the krbtgt key from the domain controller to create any number of arbitrary tickets.

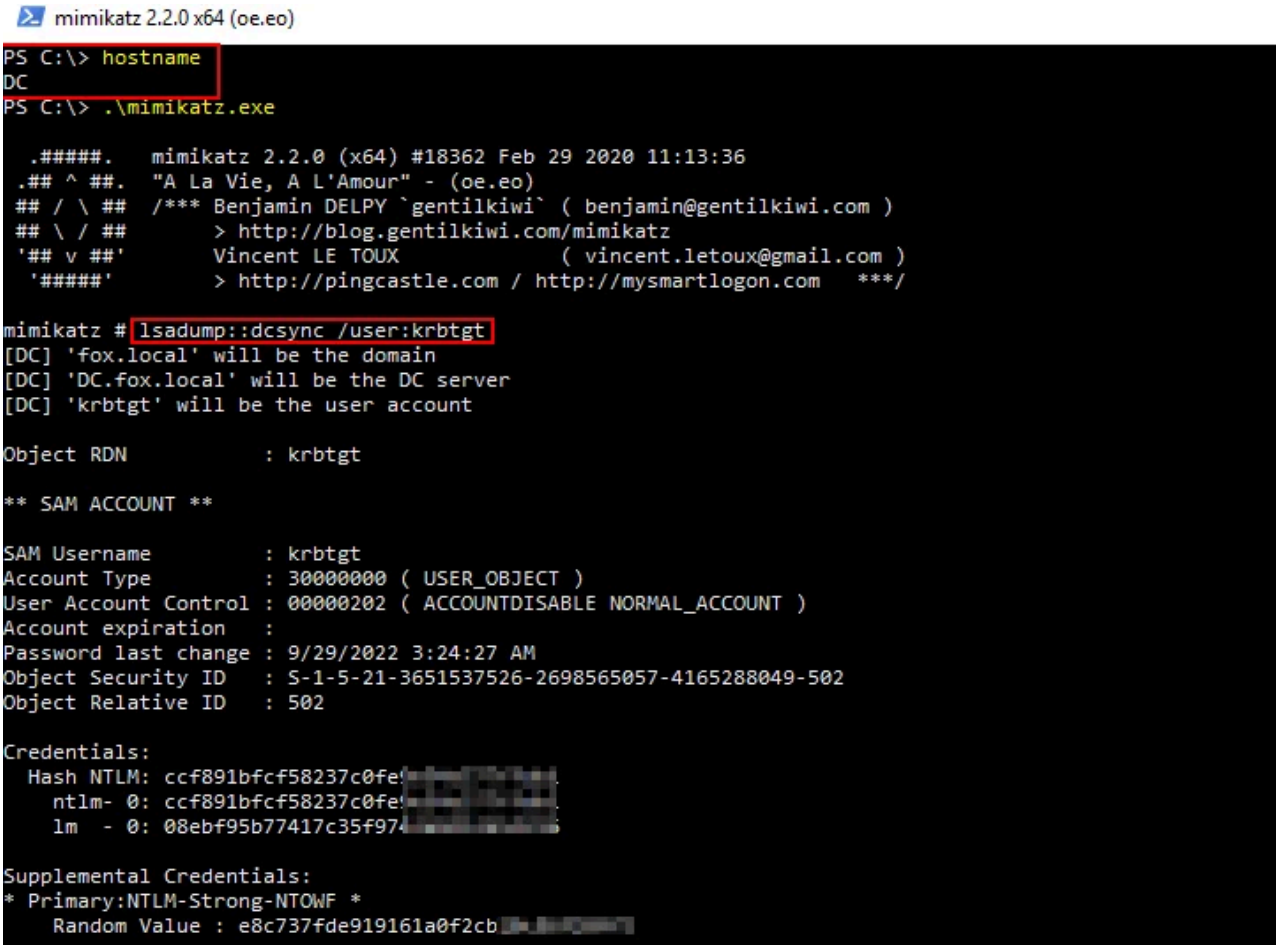
Forged Ticket: Golden Ticket Attack

Using the golden ticket attack, the attacker can forge a TGT, thus skipping the authentication mechanism altogether. It can also be used to request any service for an indefinite period (which can be specified), thus giving it the name “golden” ticket. Once an attacker compromises the krbtgt key, they can forge any ticket, thus giving them full control over the domain and the access control system.

Let us look at the golden ticket attack. To do so, we must assume that the attacker has gotten their hands on the krbtgt key.

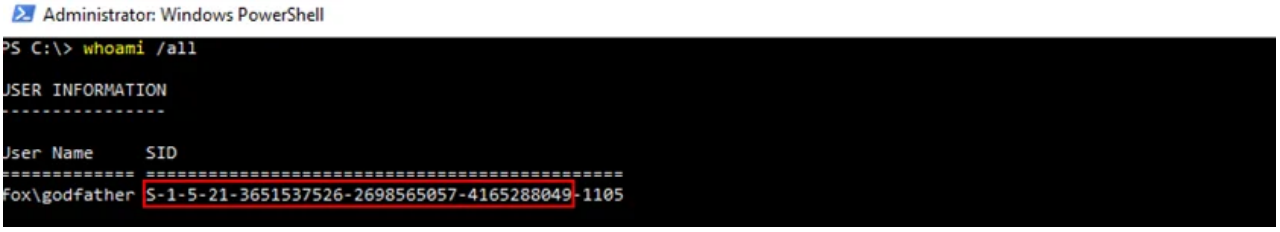
Here, for the sake of this blog, we are extracting the key directly from the DC using Mimikatz.

```
lsadump::dcsync /user:krbtgt
```



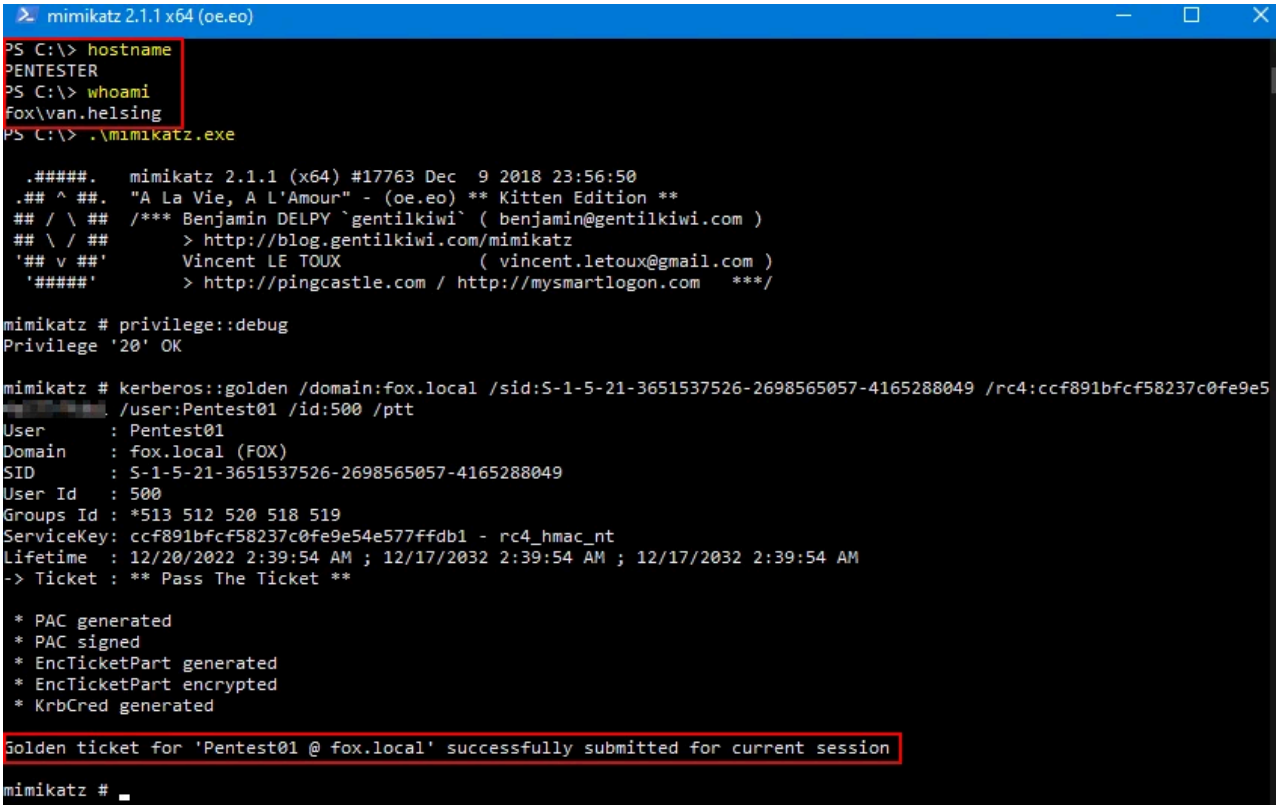
We can now use this hash to forge a ticket for new user PENTEST01 with admin privileges from the host PENTEST01.

For this, we need to get the SID of the domain, which can be obtained by running the “whoami /all” command on any domain joined host and leaving out the last part.



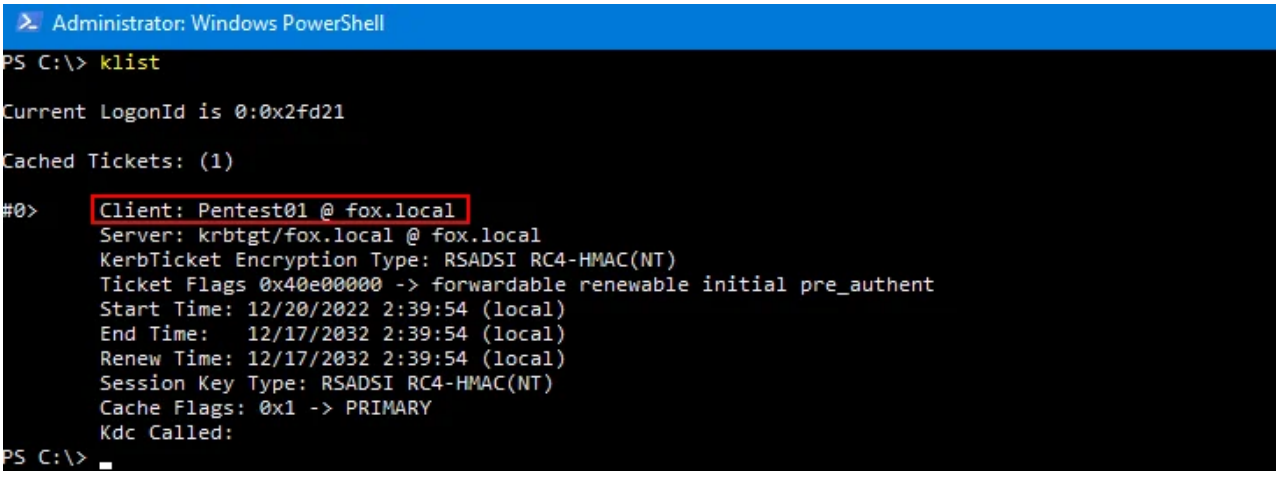
We can create a new user with the Domain Admin privileges and get their session.

```
kerberos::golden /domain:fox.local /sid:S-1-5-21-3651537526-2698565057-4165288049
/rc4:ccf891bfcf58237c0fe9e5********** /user:Pentest01 /id:500 /ptt
```



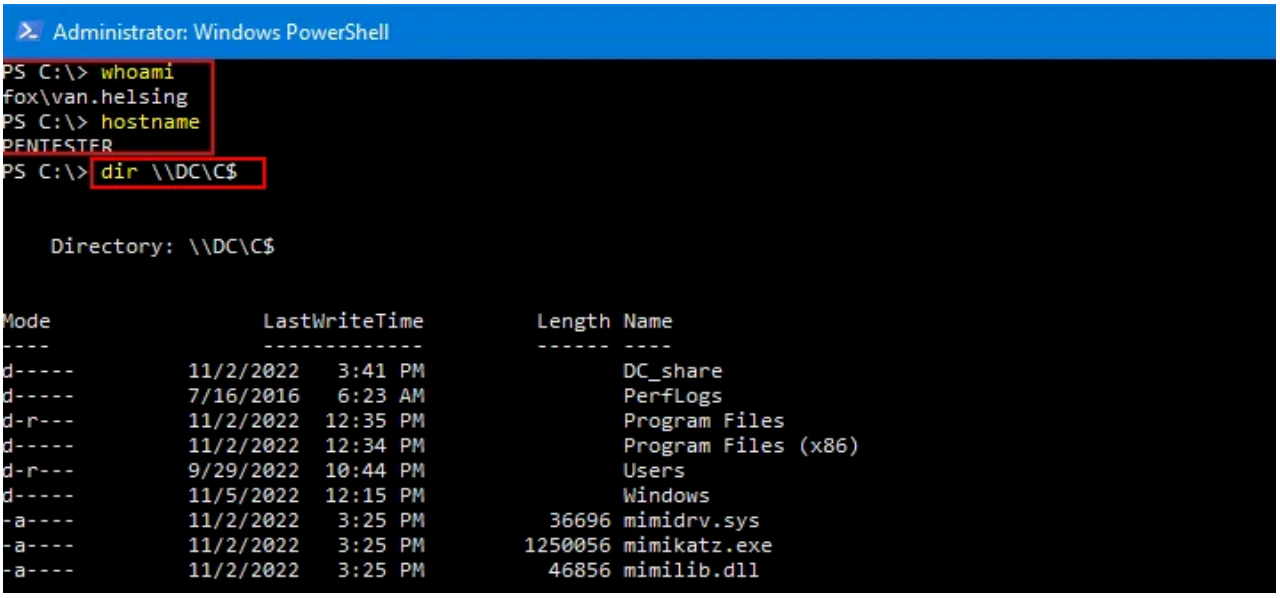
Here we are assigning it an ID of 500 which gives it administrative privileges, and also using the flag ‘ptt’ to pass the ticket generated to get a session of the newly created user.

We can verify that the session is still active by exiting from Mimkatz and checking the klist for the active ticket of the newly created domain Admin PENTEST01.



We can confirm the privileges of the newly created Domain Admin by accessing the C: drive of the domain controller DC from the test machine PENTEST using the session of PENTEST01. This drive can only be accessed by the domain or local administrators of the DC.

```
dir \\DC\C$
```



TL;DR Walkthrough

Our upcoming blog will discuss more Kerberos functionalities and how an attacker could abuse them.

By partnering with Redfox Security, you'll get the best security and technical skills to execute an effective and thorough penetration test. Our offensive security experts have years of experience assisting organizations in protecting their digital assets through **penetration testing services**. To schedule a call with one of our technical specialists, call 1-800-917-0850 now.

Redfox Security is a diverse network of expert security consultants with a global mindset and a collaborative culture. We proudly deliver robust security solutions with a combination of data-driven, research-based, and manual testing methodologies.

"Join us on our journey of growth and development by signing up for our comprehensive **courses** if you want to excel in cybersecurity."

PreviousDependency Confusion Attack and its Mitigation
NextAndroid Webview Vulnerabilities

Recent Blog

September 09, 2025
Is APK Decompilation Legal? What You Need To Know
September 06, 2025
When Hackers Hit the Road: The Jaguar Land Rover Cyberattack
September 05, 2025
This Is the Hacker's Swiss Army Knife. Have You Heard About It?