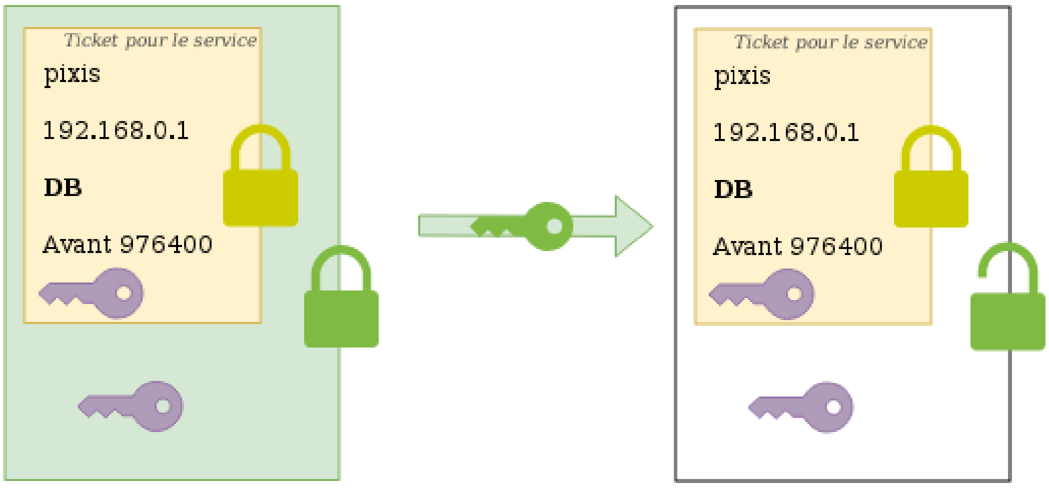
Kerberos in Active Directory

en.hackndo.com/kerberos

Pixis

February 2, 2019



02 Feb 2019 · 9 min

Active Directory is a Microsoft solution used for Windows network management, and provides the following services:

Author : **Pixis**

- Directory service (LDAP)
- Authentication (Kerberos)
- Name resolution (DNS)
- Homogeneous software policy

In this article, we will focus on the authentication part within Active Directory, based on Kerberos.

Kerberos is a protocol that allows users to authenticate on the network, and access services once authenticated.

How it works

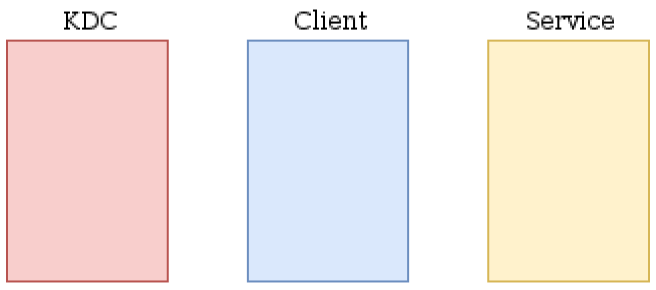
Kerberos is used whenever a user wants to access some services on the network. Thanks to Kerberos the user won't need to type his password every time and the server won't need to know every user's password. This is centralized authentication.

In order to do this, at least three entities are required

- A **client**
- A **service**

- A **Key Distribution Center** (KDC) which is a **Domain Controller** (DC) in Active Directory environment.

The idea is that when a client wants to access a service, no password will be sent over the network, thus avoiding password leaks that could compromise the network.

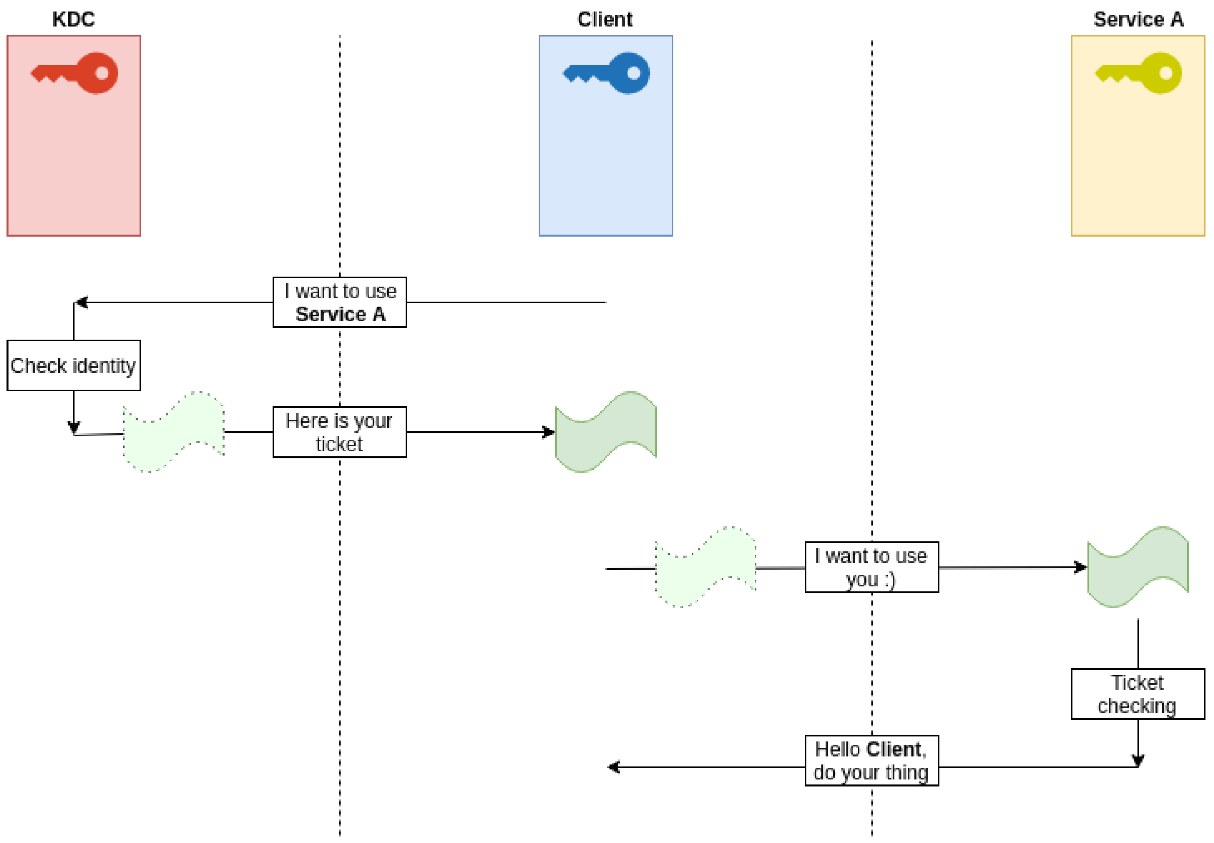


For this, the process is a bit cumbersome, and is divided into three steps:

1. Authentication Service (AS): The client must authenticate itself to the KDC.
2. Ticket-Granting Ticket (TGT): He must then request a ticket to access the chosen service (e.g. CIFS, HTTP, SQL, …).
3. Application Request (AP): He finally uses the service by providing the ticket.

It’s kind of like when you go to a party. You have your ID that you had made and that proves that you are who you say you are (TGT). If you want to drink something, you have to go to the cashier with this ID (TGT) to ask for a drink ticket (TGS). The cashier will then give you a stamped, non-forgeable drink ticket with your age on it. Once you have it, you can go to the bar and ask for your drink by presenting the ticket. The bar can check that the ticket comes from the cash desk thanks to the stamp, and serves you a good ol’ fresh beer if you are not underaged.

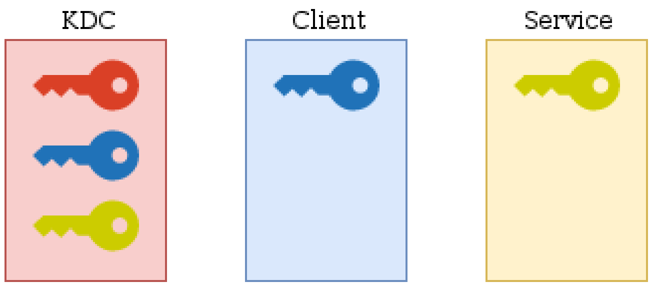
In a nutshell, the process looks like this:



All right, but how does it work in practice?

We are in Active Directory context, so the KDC is also the Domain Controller (DC). The KDC contains all the domain information, including the secrets of each service, machine, user. Thus, except for the DC, everyone only know his own secret, and therefore do not know the secrets of the other objects in Active Directory.

Let's take `pixis` user as an example. He wants to use a given service. To do so, he will need to authenticate himself to the KDC and he will then send a request to use the service. This phase is called **Authentication Service** (AS).

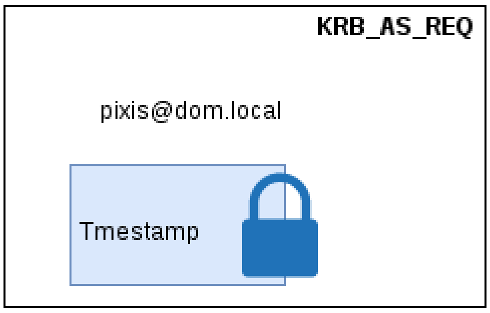


Authentication Service (AS)

KRB_AS_REQ

`pixis` will first send a request for a **Ticket Granting Ticket** (TGT) to the Domain Controller (DC). This request is called **KRB_AS_REQ** (*Kerberos Authentication Service Request*). The TGT requested by the client is a piece of encrypted information containing, among other things, a session key and user information (ID, name, groups, …).

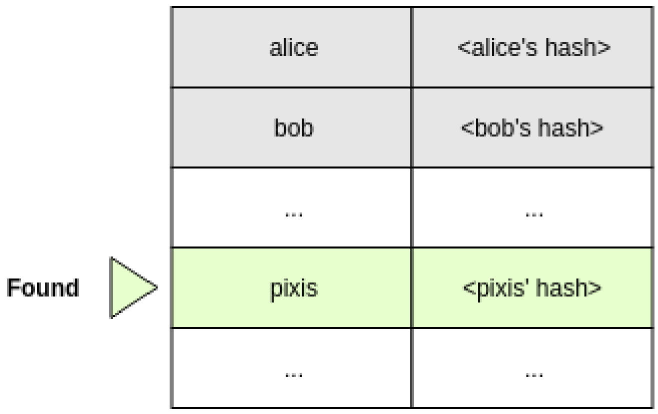
In order to perform this TGT request, `pixis` will send its name to the KDC as well as the exact request's time, encrypted with a hashed version of his password.



The KDC will receive this username, and will verify that it exists in its database.

If it finds it, it will then retrieve `pixis` hashed password which it will use to try to decrypt encrypted timestamp. If it can't, then the client didn't use the correct password to encrypt this timestamp.

If it does, however, the KDC is assured that it is really `pixis` who is talking to him. It will generate a unique session key tied to this user and limited in time.



alice	<alice's hash>
bob	<bob's hash>
...	...
pixis	<pixis' hash>
...	...

KRB_AS_REP

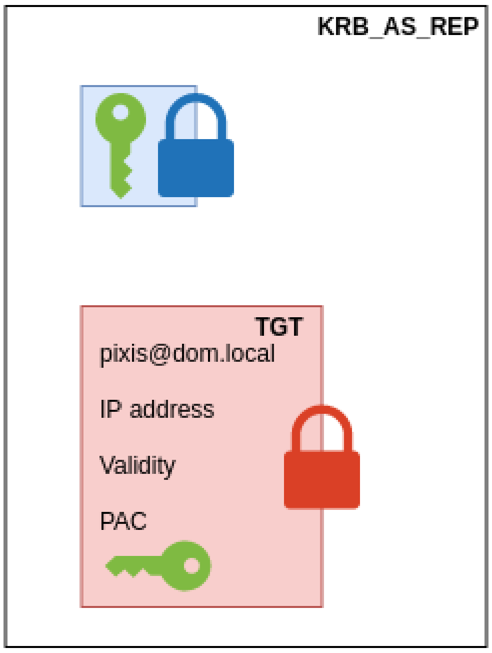
The KDC will send back different things to `pixis` (**KRB_AS_REP**).

- The **session key**, encrypted with `pixis` hashed password;
- The **TGT**, containing various information like:
  - Username (pixis)
  - Validity period
  - Generated session key
  - The **Privilege Attribute Certificate** (PAC) which contains a lot of specific information about the user, including his identifier (SID) and all the groups he is member of.

  TGT will be encrypted with the KDC key. Thus, **only the KDC is able to decipher and read this ticket's content**.

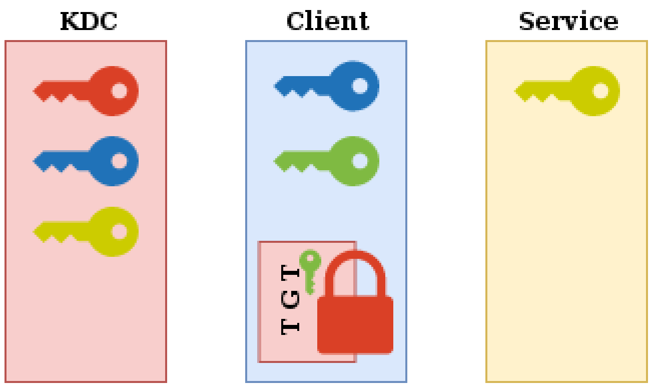
*Note that this TGT is considered public information. It can very well be intercepted during the authentication phase. We will see in the following paragraph the importance of the **authenticator** that accompanies the TGT when the client communicates with the KDC.*

The client receives these pieces of information. Using his hashed password, the first part will be decrypted in order to retrieve the session key that will be necessary for further exchanges.



Ticket-Granting Service (TGS)

Now that the user is authenticated, we are in the following situation: The user has his own key as well as a time-limited session key that only he currently knows, and a KDC-encrypted TGT that contains, among other things, this same session key.



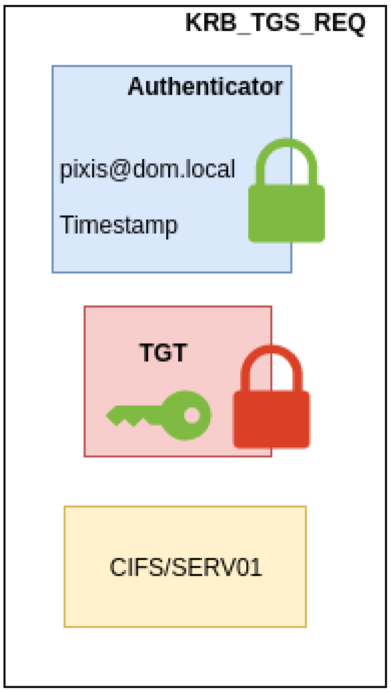
KRB_TGS_REQ

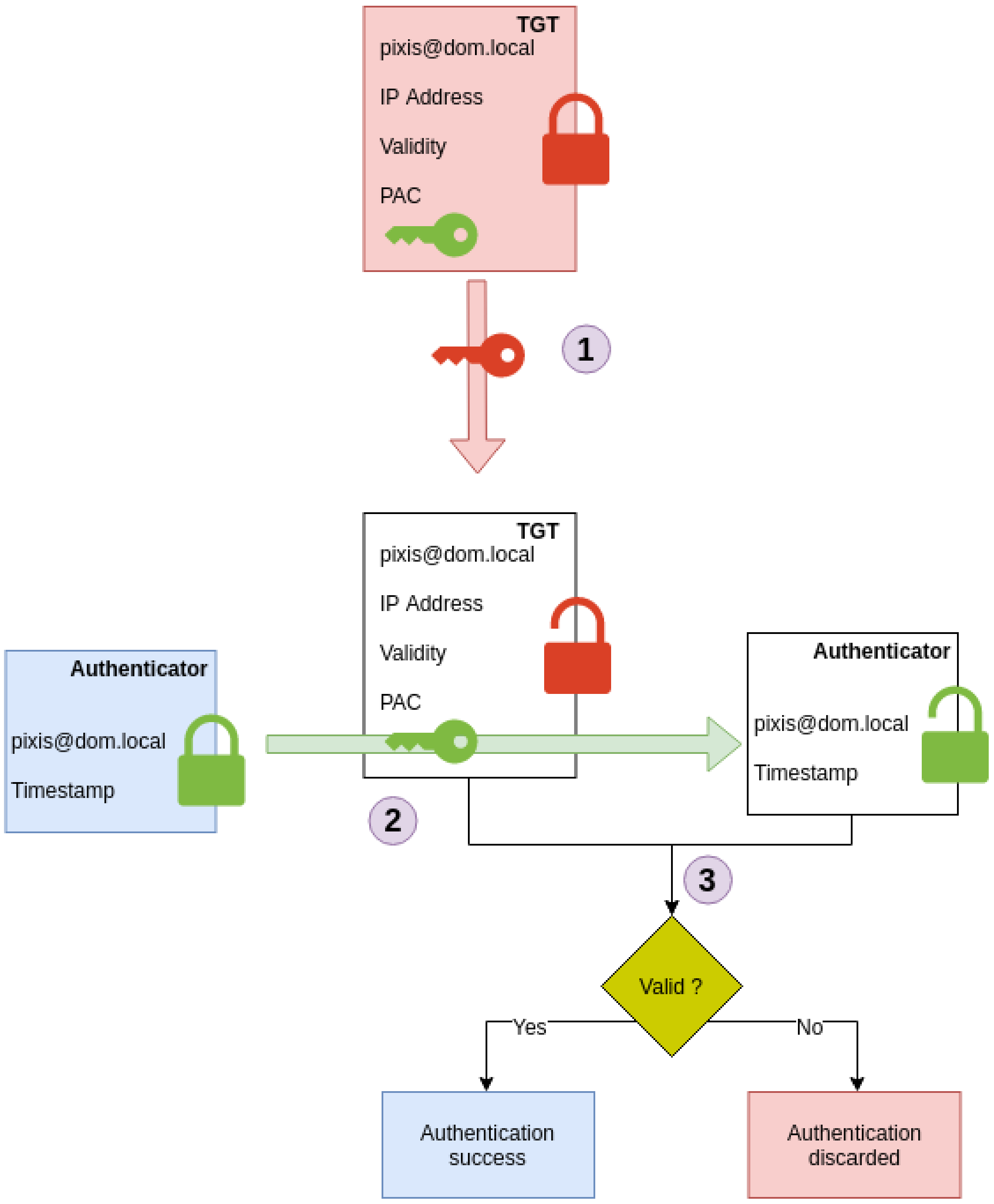
If `pixis` wants to use a service, e.g. `CIFS` on `SERVER01`, it will send several pieces of information to the KDC so that the KDC can send back a Service Ticket.

- The TGT;
- The service he wants to use and the associated host, so `CIFS/SERV01` in this example;
- An **authenticator**, which contains his username and current timestamp, all encrypted with the session key.

The **authenticator** is sent to make sure that it is `pixis` that is making the request. To do this, the KDC will compare TGT and **authenticator** contents. Since only the KDC can read TGT content, it could not have been tampered with. The KDC will read TGT content, including the TGT's owner and the associated session key. Then it will decrypt **authenticator** content with the same session key. If the decryption works, and the data in the **authenticator** matches the data in the TGT, then `pixis` is who it claims to be. The KDC is assured that whoever made the request has the TGT **and** knowledge of the negotiated session key.

Here is a diagram which summarizes this verification process at KDC level:





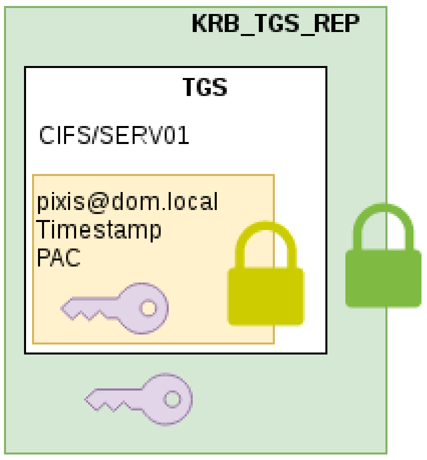
KRB_TGS_REP

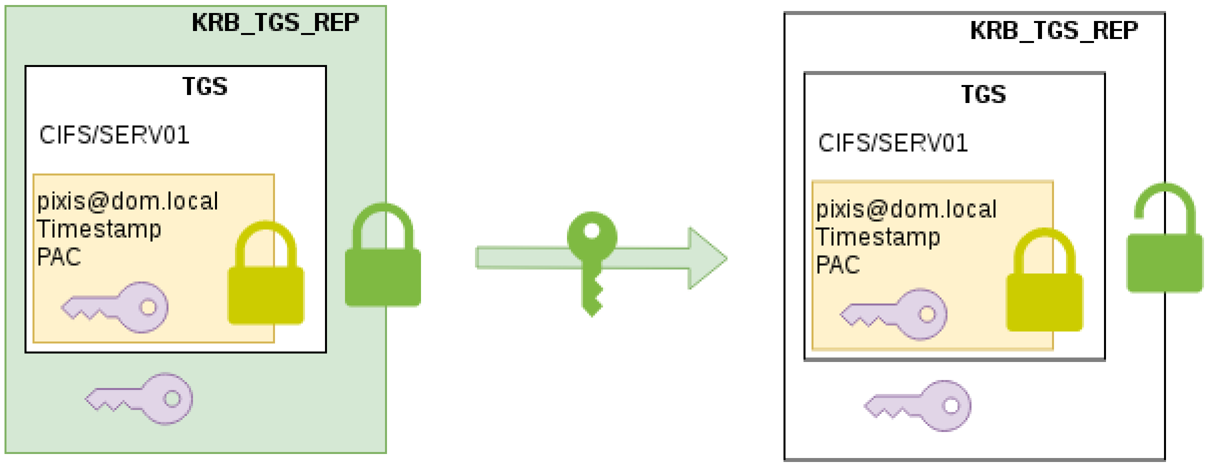
Now that the KDC has been able to verify that the user is `pixis`, it will send back information that will allow the user to make a request to the service. This message is the **KRB_TGS_REP**. It includes the following elements:

- A ticket containing the name and host of the requested service (`CIFS/SERV01`), user's username (`pixis`), the PAC, and a new session key which is only valid for communications between `pixis` and `SERVER01` for a certain period of time. This ticket is encrypted with the service's key (i.e. the host key, since CIFS service runs under the host account);
- The new session key

These two pieces of information (the ticket and the session key) are encrypted with the first session key, the one that was initially exchanged between the KDC and the client.

The client will receive this message, and will be able to decrypt the first layer to get the session key created for communication with the service, as well as the ticket generated to use this service. This ticket is commonly called **Ticket-Granting-Service** (TGS).



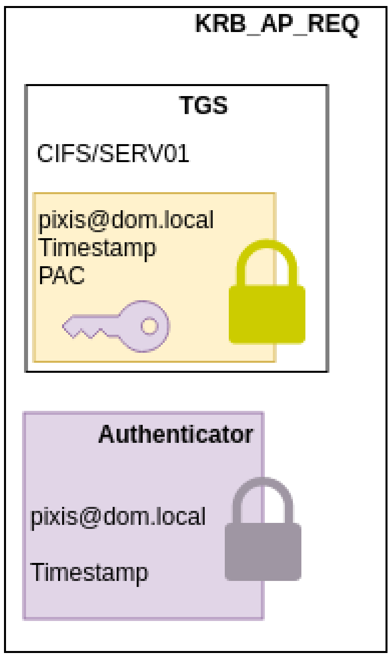


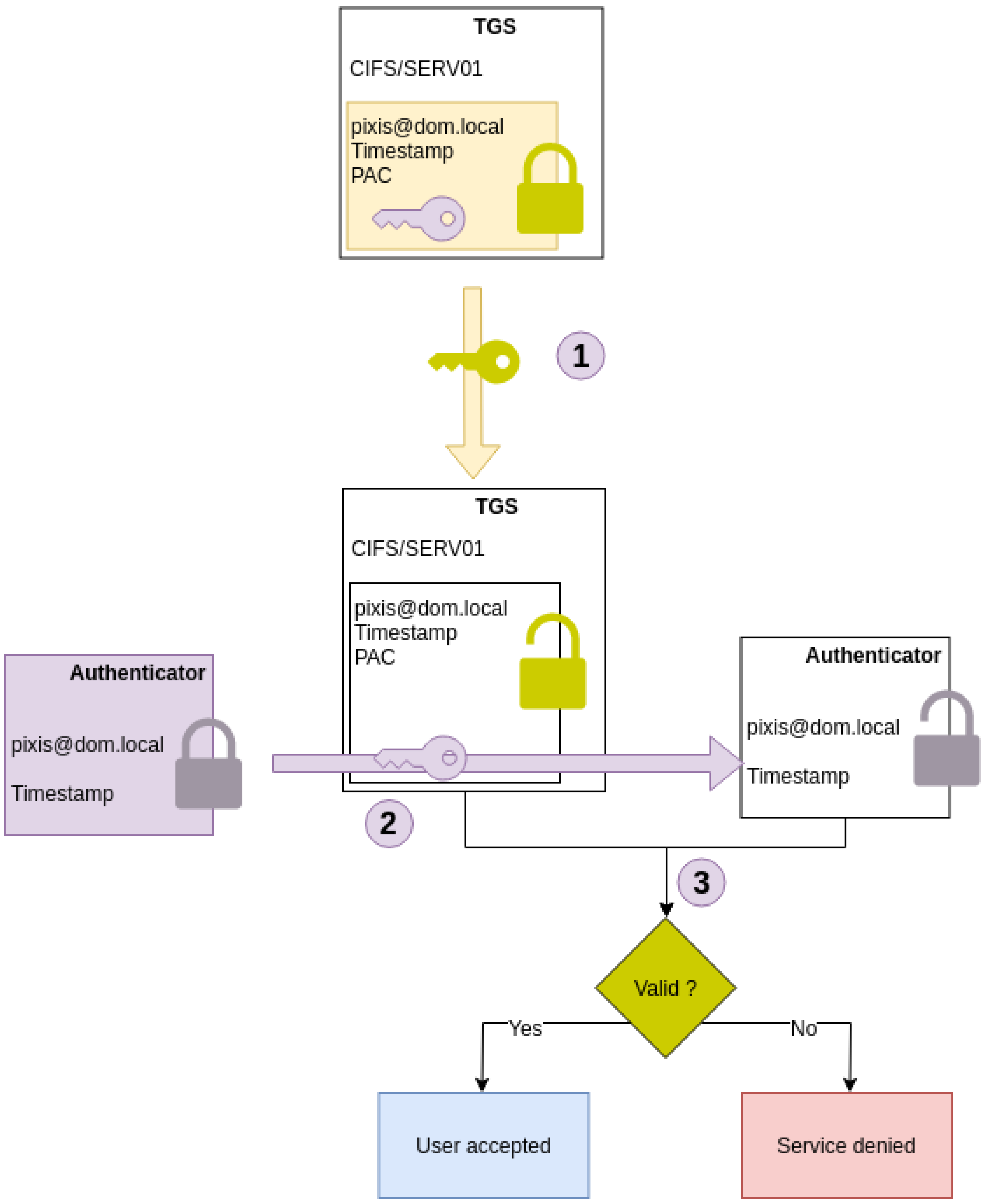
Application Request (AP)

KRB_AP_REQ

`pixis` will then generate a new authenticator which it will encrypt with this new session key, along with the TGS. This is the same process as with the KDC.

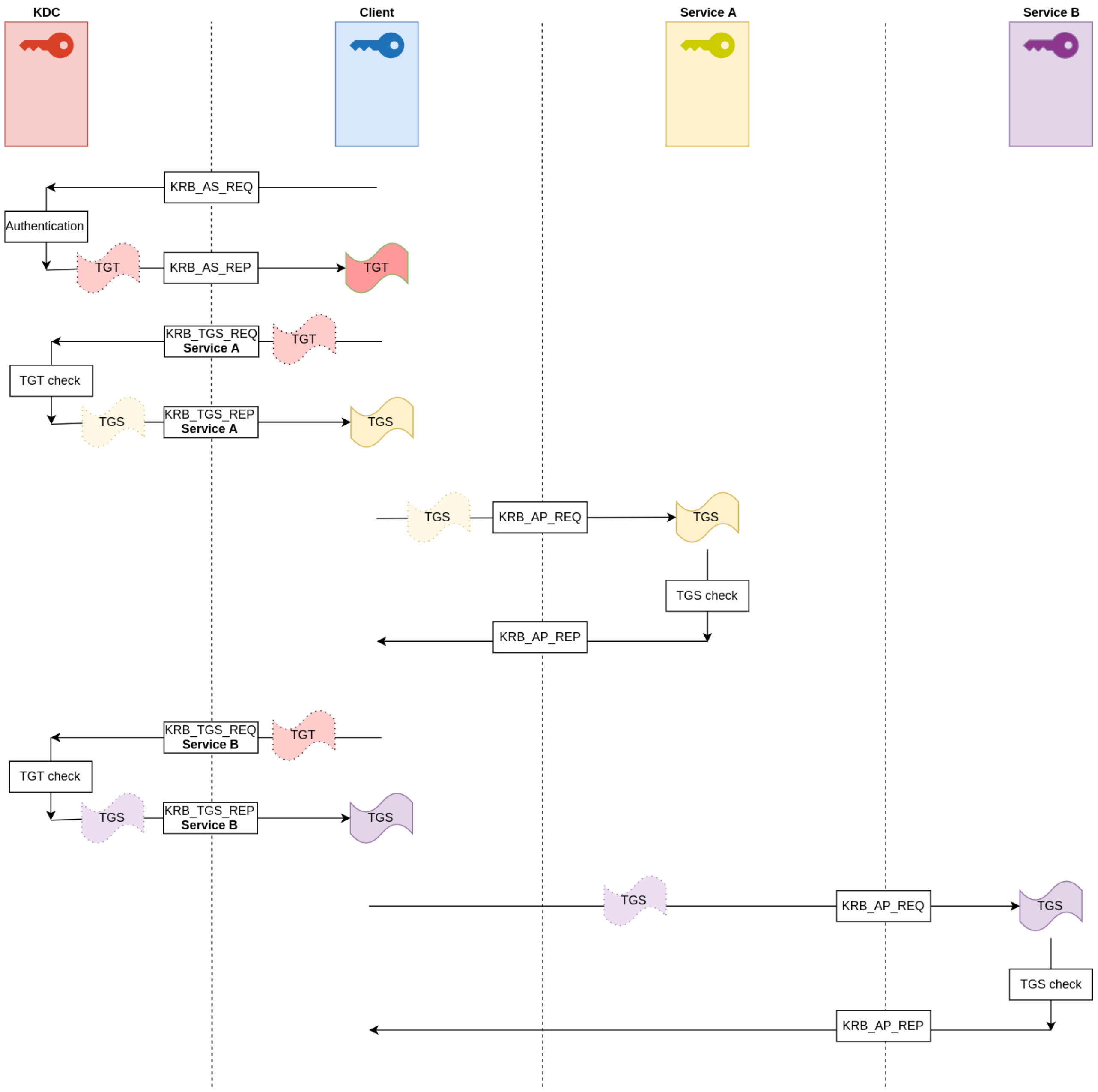
`CIFS` service receives the TGS and can decrypt it with its own secret. Since only the DCS knows its secret, he rests assured that this TGS is authentic. This TGS contains the session key it will use to decrypt the authenticator. By comparing the TGS and authenticator contents, the service can be certain of the user's authenticity.


KRB_AP_REQ
TGS
CIFS/SERV01
pixis@dom.local
Timestamp
PAC
Authenticator
pixis@dom.local
Timestamp



Summary

It is a relatively complex process, but once the steps have been looked at in detail, the usefulness of each step is better understood. Here is a summary diagram of the three steps for a client requesting access to two different services (click on the image if you want to zoom):



Conclusion

Kerberos protocol is very robust against a large number of attacks. Its design ensures that user or service authentication secrets are never exposed.

Next posts will detail some attacks on misconfiguration issues, or built-in issues that can lead to domain comprision.

Author : Pixis
Blog author, follow me on twitter or discord

Similar posts