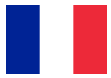
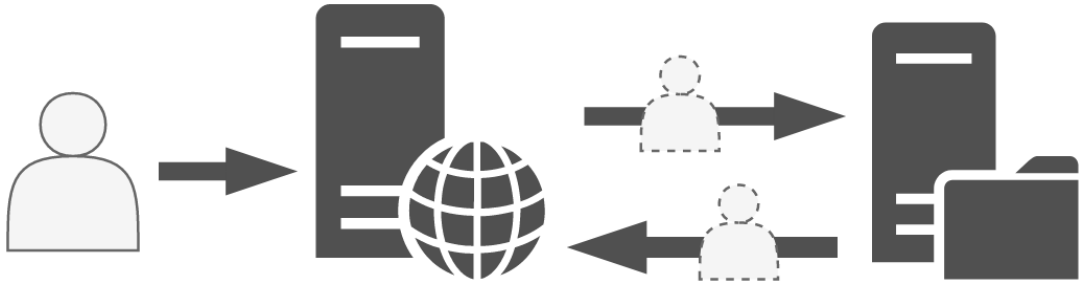
Kerberos Delegation

en.hackndo.com/constrained-unconstrained-delegation

Pixis April 18, 2020





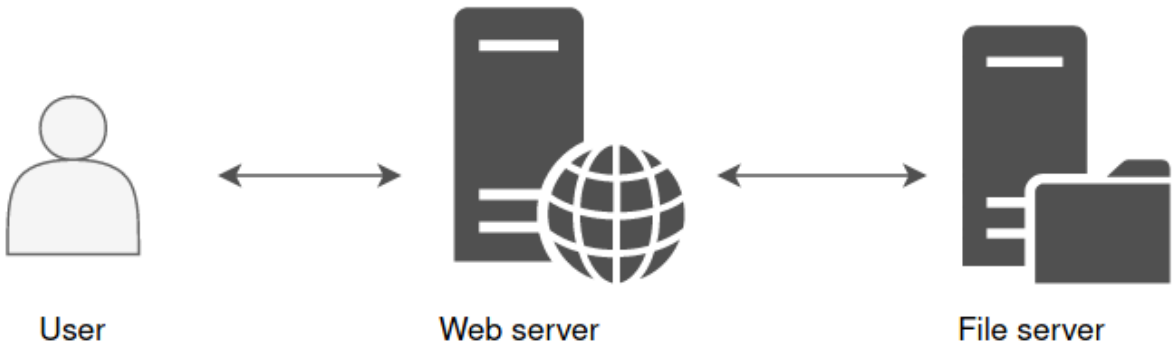
18 Apr 2020 · 14 min

Author : **Pixis**

Within an Active Directory, services can be used by users. Sometimes these services need to contact others, on behalf of the user, like a web service might need to contact a file server. In order to allow a service to access another service **on behalf of the user**, a solution has been implemented (introduced in Windows Server 2000) to meet this need : **Kerberos Delegation.**

Delegation principle

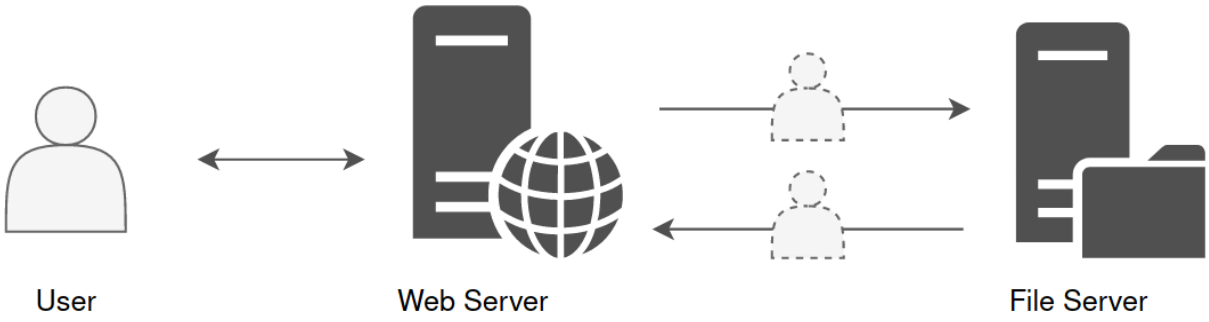
In order to understand what is Kerberos Delegation, let's take a concrete example. A web server with a nice interface allows a user to access his personal folder, hosted on a file server. We are in the following situation :



The web server is front-end, and it's this web server that will fetch the information instead of the user on the file server in order to display the content of a file, for example.

However, the web server does not know what belongs to the user on the file server. It is not his role to unpack the user's PAC to make a specific demand to the file server. This is where the **delegation** comes in. This mechanism allows the web server to **impersonate** the user, and to authenticate on the user's behalf to the file server. Thus, from the file server's point of view, it is the user who makes the request. The file server will be able to

read and check user's rightsr, then send back the information to which this account has access. This is how the web server can then display this information in a nice interface back to the user.



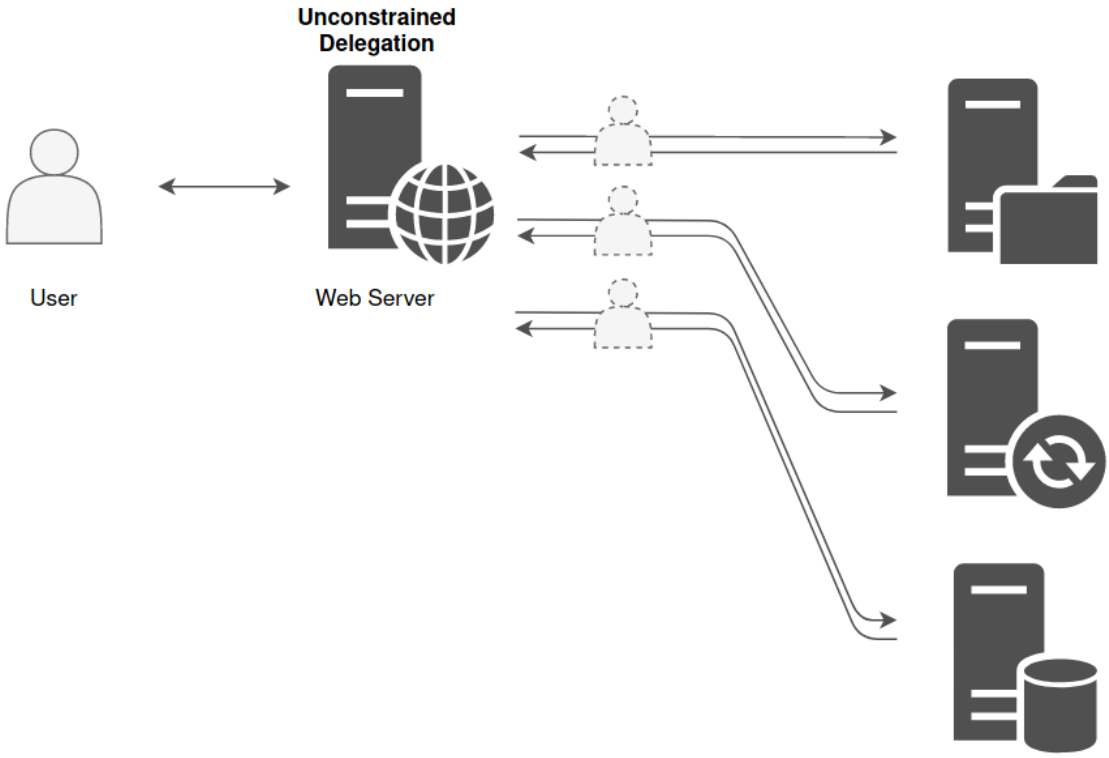
Constrained & Unconstrained Delegation

The ability to relay credentials can be given to an account with at least one SPN attribute set. It could be a computer account or a service account.

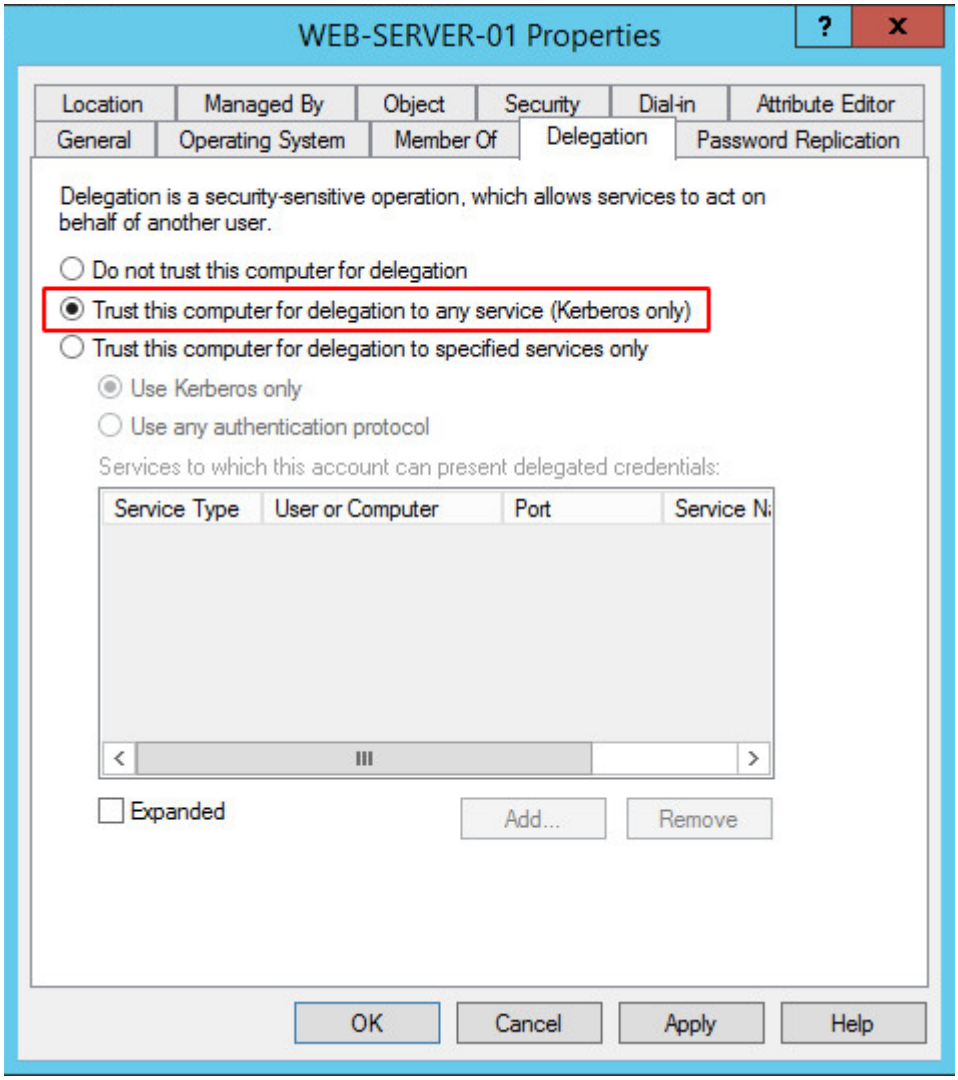
Today, there are three ways to authorize a computer or service account to impersonate a user in order to communicate with one or more other service(s) : **Unconstrained Delegation**, **Constrained Delegation** and **Resource Based Constrained Delegation**.

Unconstrained Delegation

With **Unconstrained Delegation**, the server or the service account that is granted this right is able to impersonate a user to authenticate to **any services** on **any host**.



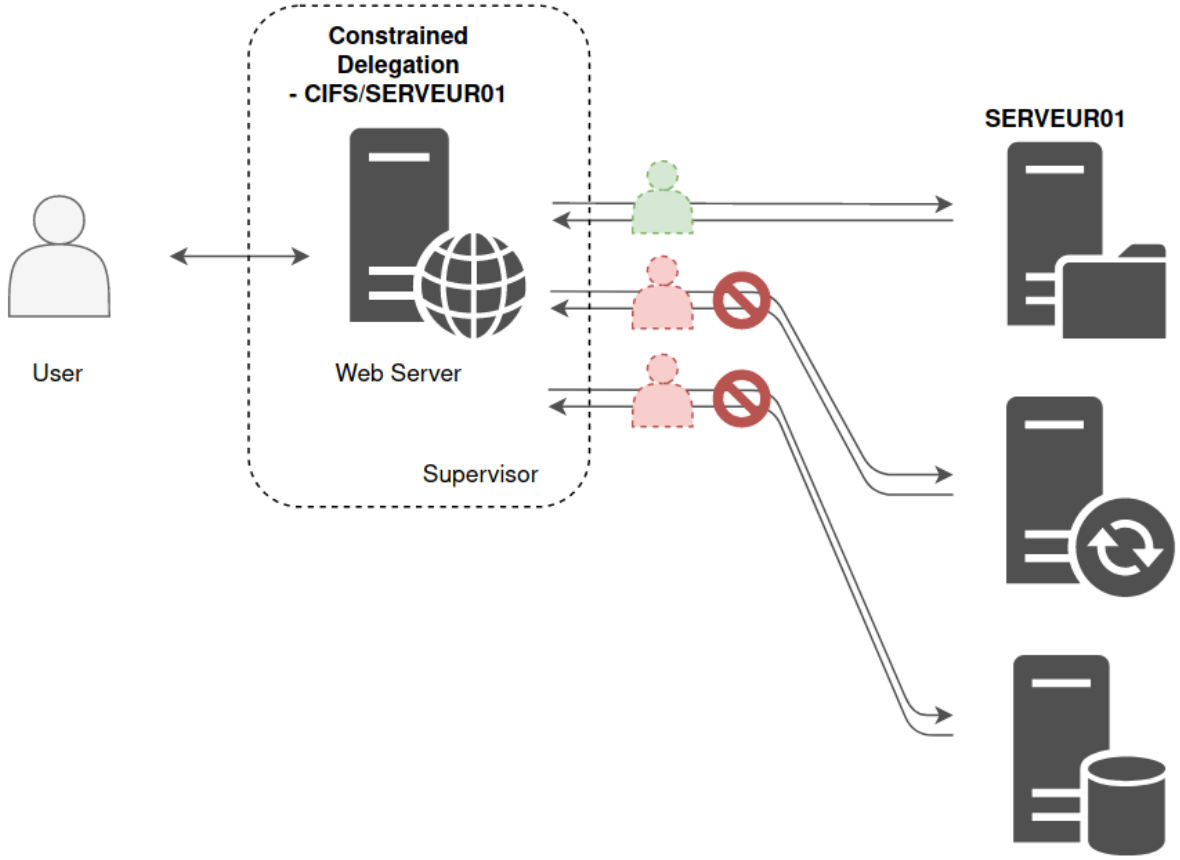
Here is an example, in my lab, of a machine that is in **Unconstrained Delegation** :



It is historically the only choice there was when the delegation principle was introduced. But for obvious security reason, **Constrained Delegation** has been added.

Constrained Delegation

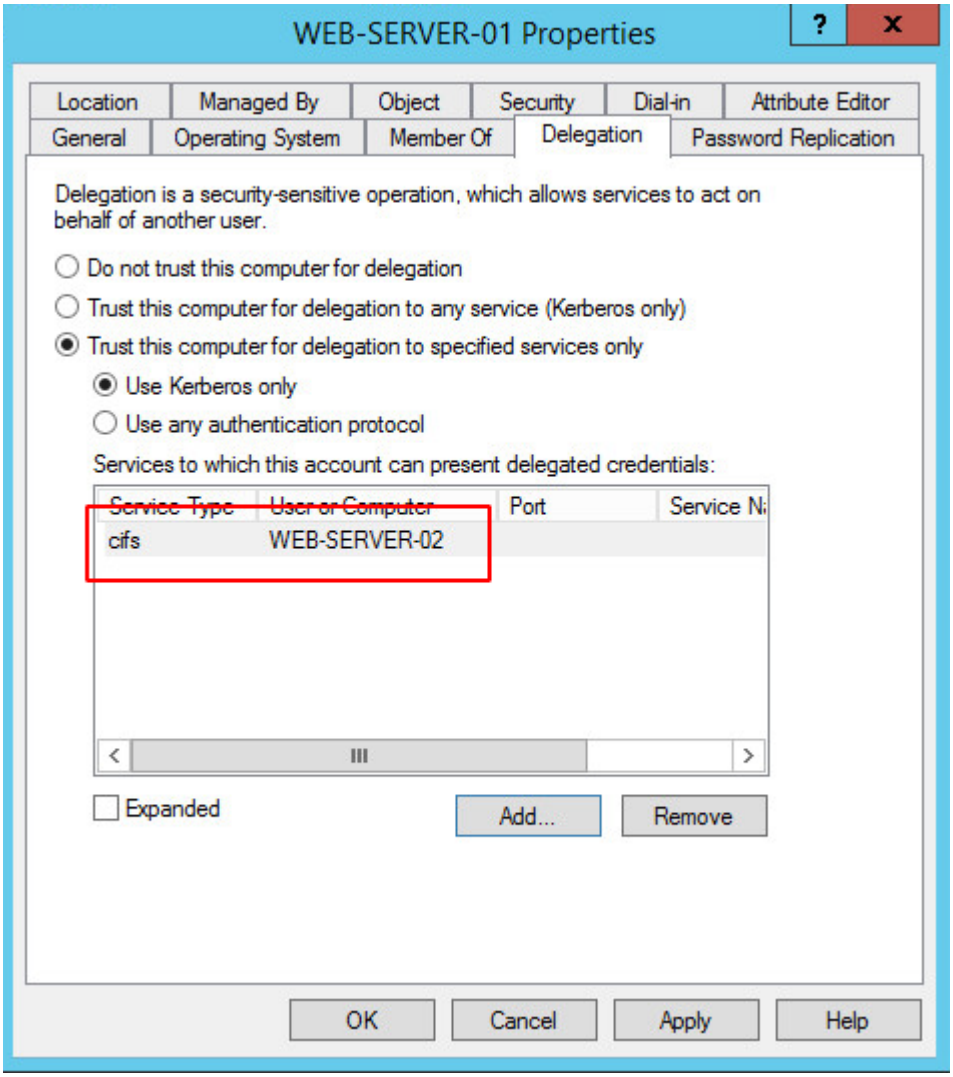
If a computer or a service account has the **Constrained Delegation** flag set, a list of authorized services shall be associated to this flag. For example, in previous example (web server and file server), the web server will have the **Constrained Delegation** flag indicating that this account can only impersonate some users against `CIFS` service hosted by `SERVER01`, the file server.



This list of allowed SPN is set on the relaying account, the computer account hosting the web server..

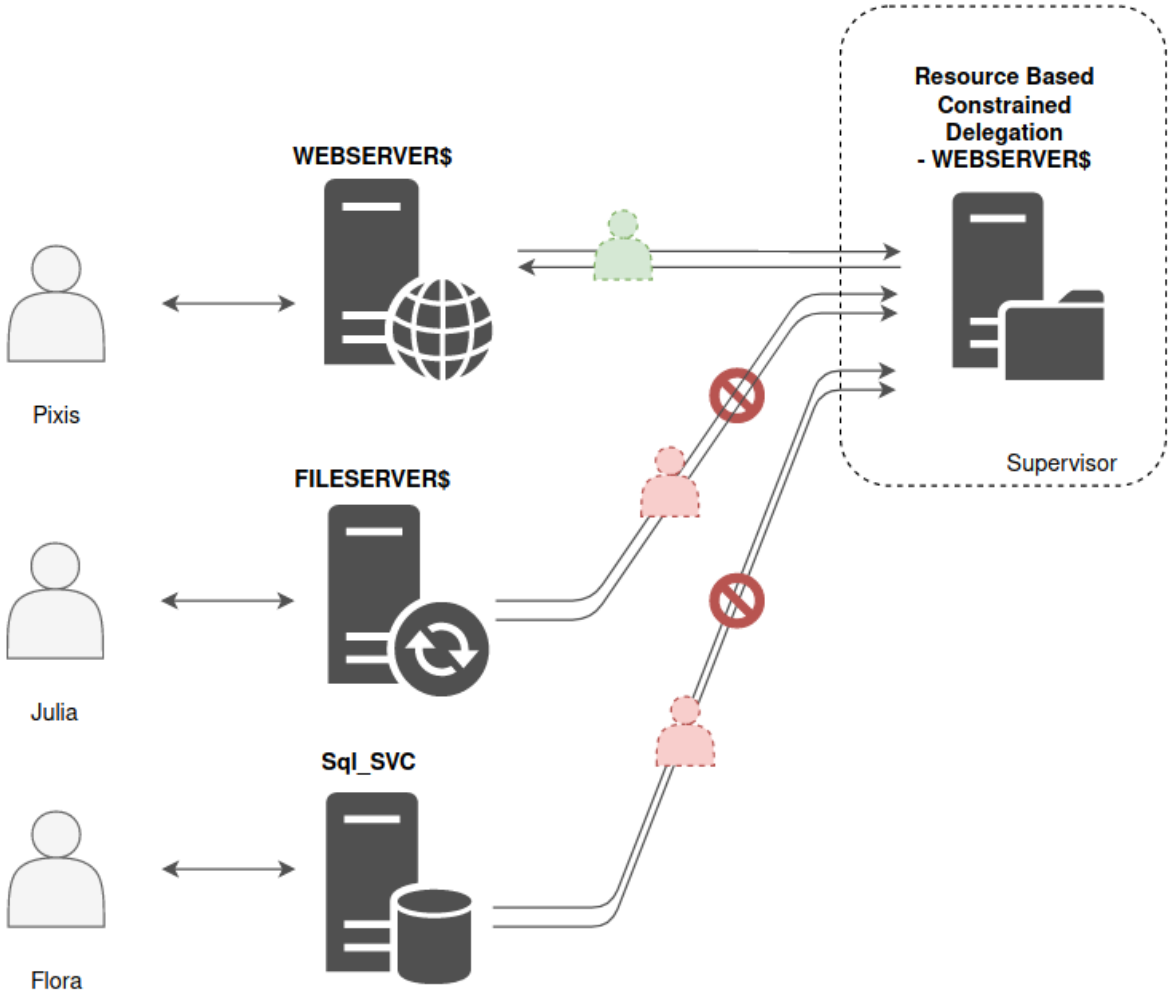
In other words, the Domain Controller will read the SPN list on this account, and will decide: “This account is allowed to impersonate a user, so he can authenticate against one of these services on behalf of the user”.

In my lab, the web server is `WEB-SERVER-01` and the one with file sharing is `WEB-SERVER-02`. So here is what the list of services for which `WEB-SERVER-01` can pretend to be the user looks like :



Resource Based Constrained Delegation - RBCD

Finally, we have the **Resource Based Constrained Delegation** (RBCD) case. Introduced with Windows Server 2012, this solution allows to overcome some problems related to the **Constrained Delegation** (Responsibility, inter-domains delegation, …). Without going into too much details, the delegation responsibility is moved. Whereas in **Constrained Delegation**, it's the relaying server that holds the list of allowed target services, in the case of **Resource Based Constrained Delegation**, it's the resources (or services) that have a list of accounts they trust for delegation. Thus, the diagram is as follows :



The responsibility is shifted, it's at the level of the resource that receives the delegated authentications that the information of whether or not the delegation is accepted is found.

In other words, it's the end resource that says "I allow this list of account […] to authenticate to me on behalf of someone else".

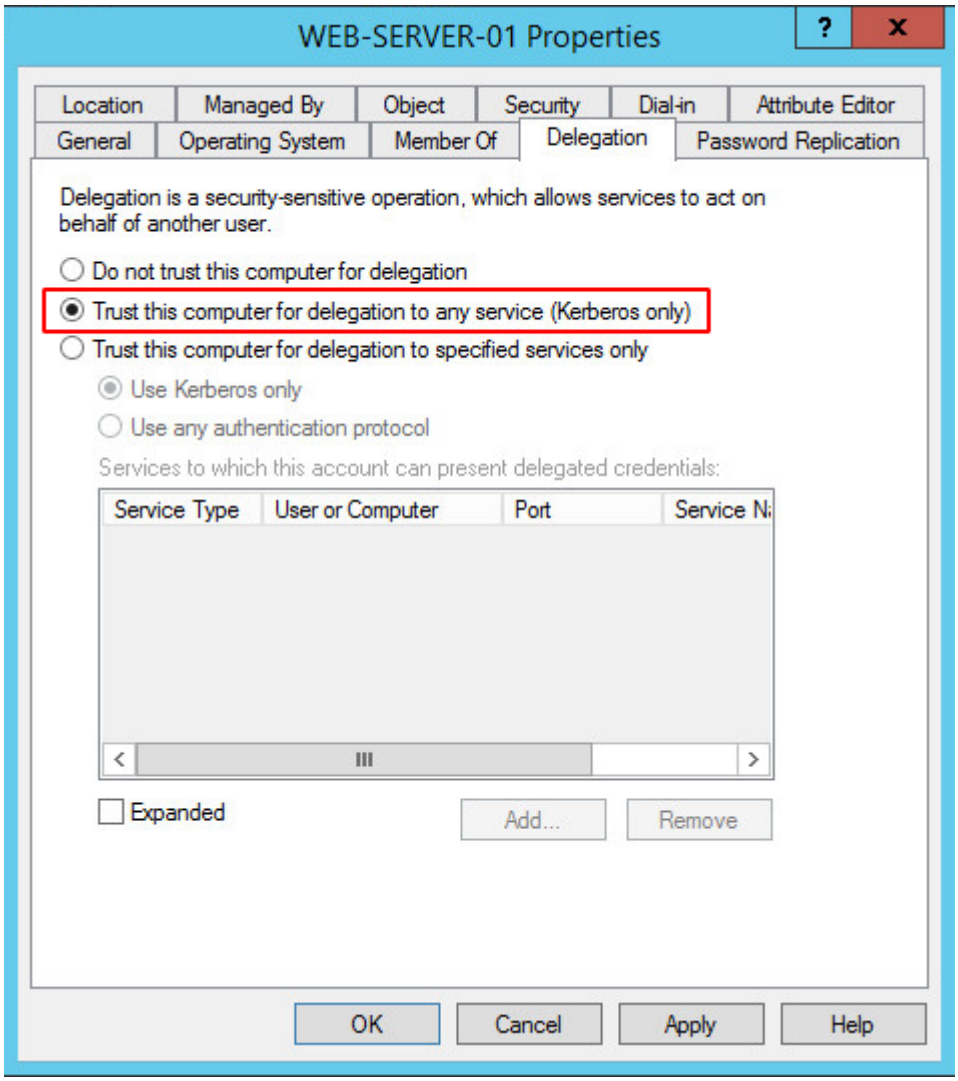
Technical details

Now that the high-level behavior is understood (at least I hope so), let's go into a little more detail about this process. Concretely, how can a machine or an account can pretend to be a user when using a resource ? That's what we are going to see now. Details between every different techniques are relatively different, that's why each of them will be explained separately. Stay close on your seat, it's gonna get dirty.

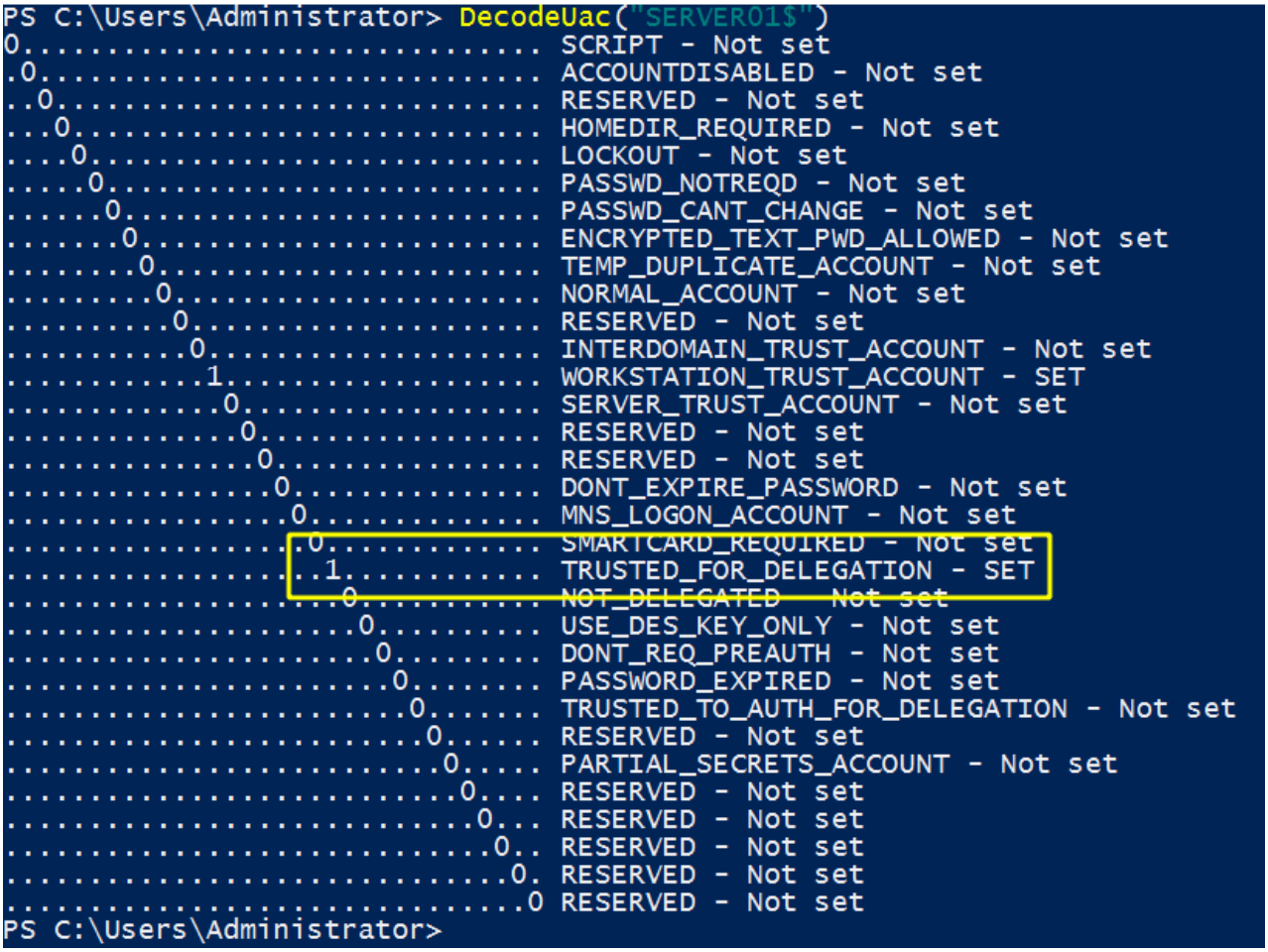
Unconstrained Delegation

As seen before, in this case, the server or the service account can authenticate on behalf of the user to any other services. For this to be possible, two prerequisites are required :

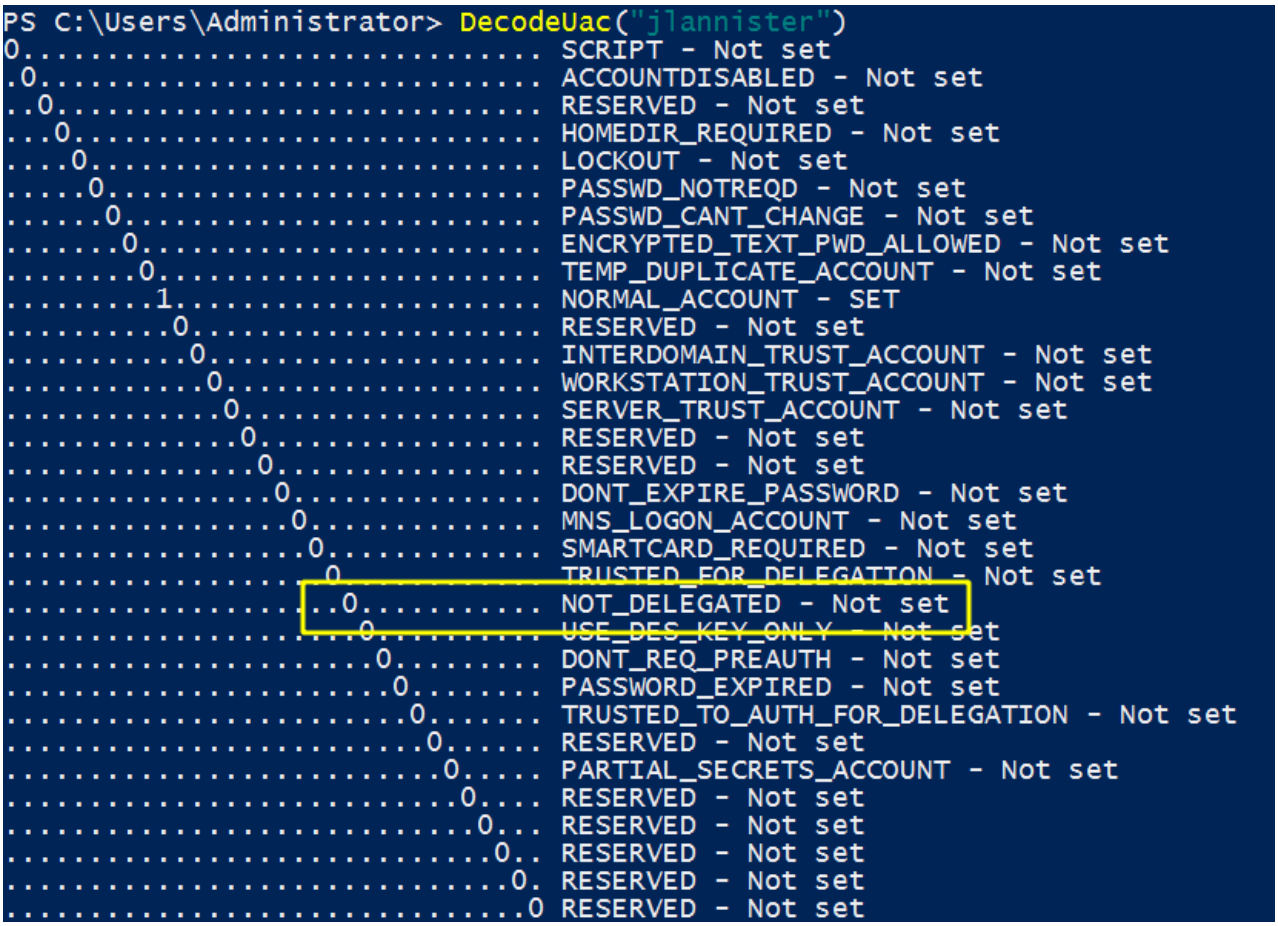
The first one is that the account that wants to delegate an authentication has the `TRUSTED_FOR_DELEGATION` flag in his UAC - User Account Control flags. In order to set this flag, you need to have the `SeEnableDelegationPrivilege` right, which is usually only available for domain administrators. Here is how the flag is set on the account (machine or service account):



Once set, we can list UAC flags and check that `TRUSTED_FOR_DELEGATION` has been set.



The second one is that the user account which will be relayed is effectively “relayable”. To **disable** relaying capabilities on an account, NOT_DELEGATED flag is set. By default, no account on the AD has this flag set, so they are all “relayable”.

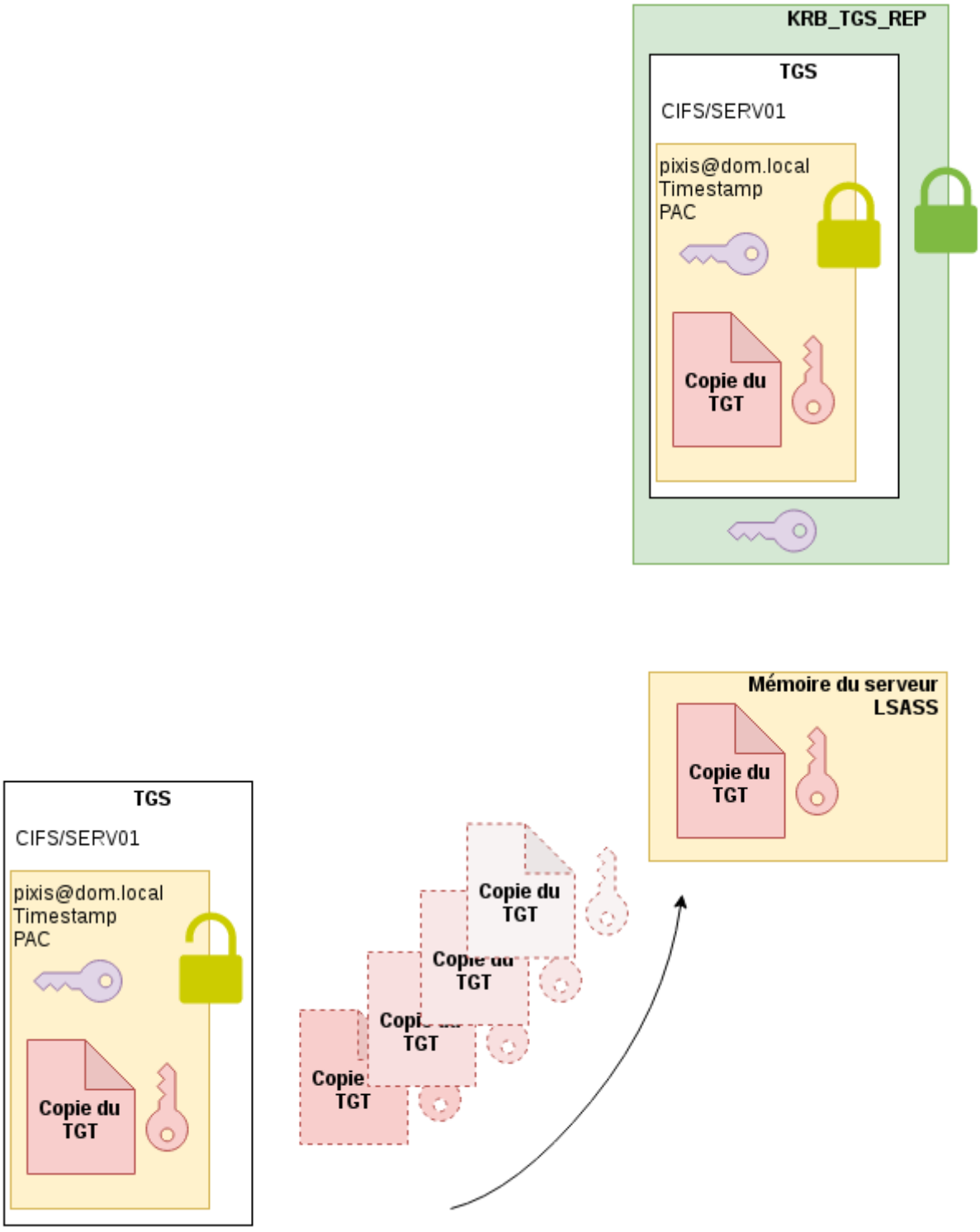


Concretely, during exchanges with the Domain Controller as described in the Kerberos in Active Directory article, when the user asks for a TGS (KRB_TGS_REQ), he will specify the SPN of the service he wants to use. It is at this point that the Domain Controller will look for the two prerequisites :

- Is the `TRUSTED_FOR_DELEGATION` flag set in the attributes of the account associated to the SPN.
- Is the `NOT_DELEGATED` flag **not** set for the requesting user.

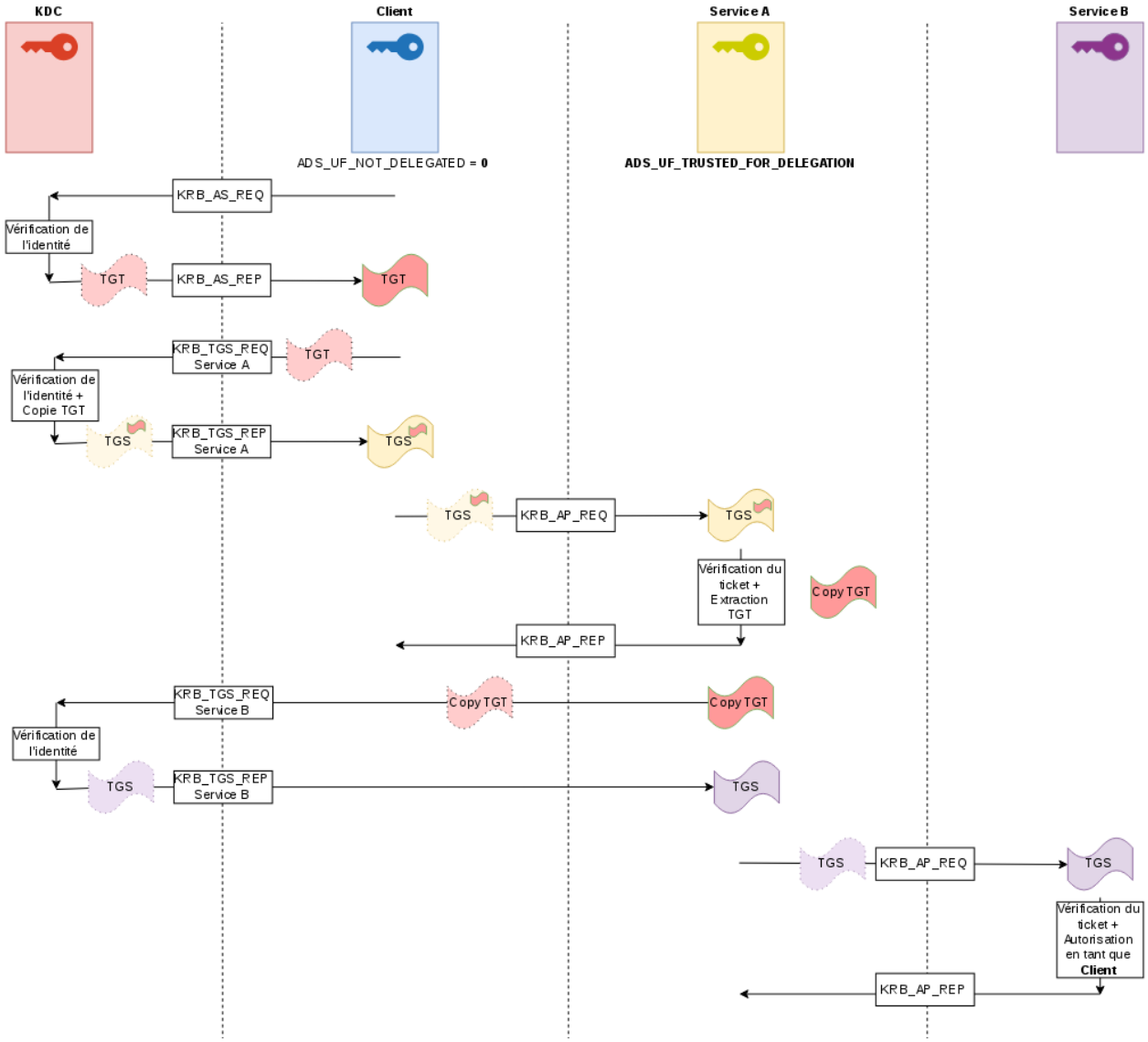
If both prerequisites are met, then the Domain Controller will respond to the user with a KRB_TGS_REP containing standard information, but it will also contains a **copy of the user's TGT** in his response, and a new associated session key.

Once in possession of these elements, the user will continue the classic process by sending a request to the service (KRB_AP_REQ) by sending the TGS and an authenticator. The service will be able to decrypt the content of the TGS, verify the user's identity by decrypting the authenticator, but above all it will be able to retrieve the copy of the TGT and the associated session key, in order to pretend to be the user at the Domain Controller.



Now in possession of a copy of the user's TGT and a valid session key, the service can authenticate to any other service on the user's behalf by requesting the TGS from the Domain Controller, by providing this TGT and by encrypting an authenticator with the session key. It's the **Unconstrained Delegation** principle.

Here is a summary diagram :



Constrained Delegation

For the **Constrained Delegation**, a list of SPN or of authorized accounts will be provided to indicate the services/accounts accepted for the delegation. Therefore, the process is not the same. The service involved will not be in possession of the user's TGT, otherwise there is no way to control service's authentication. A different mechanism is used.

Let's consider the case where the user authenticate to `Service A` and then this `Service A` has to impersonate the user to access `Resource B`.

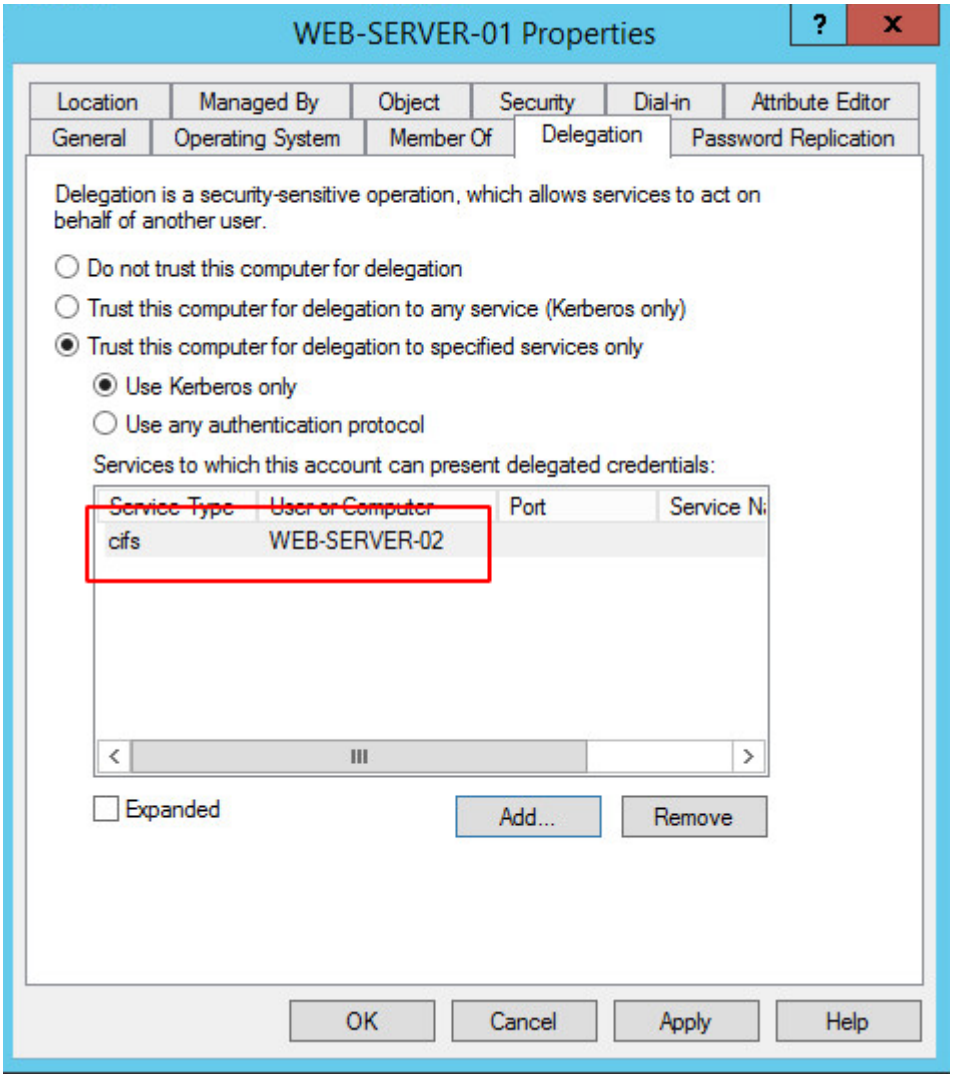
The user makes a TGS request, then sends it to `Service A`. Since this service needs to access `Resource B` as the user, it will request a TGS to the Domain Controller on behalf of the user. This request is governed by the S4U2Proxy extension. To tell the Domain Controller it wants to authenticate on behalf of someone else, two attributes will be defined in the KRB_TGS_REQ ticket request:

- The field `additional-tickets`, usually empty, must contain the user's TGS (given that the `NOT_DELEGATED` flag is **not** set for the requesting user. If that was the case, the user's TGS would **not** be `forwardable`, and the Domain Controller would not accept it in the rest of the process)
- The cname-in-addl-tkt flag, which should be set to indicate to the Domain Controller that it should not use the server information, but the ticket information in `additional-tickets`, i.e. the information of the user the server wants to pretend to be.

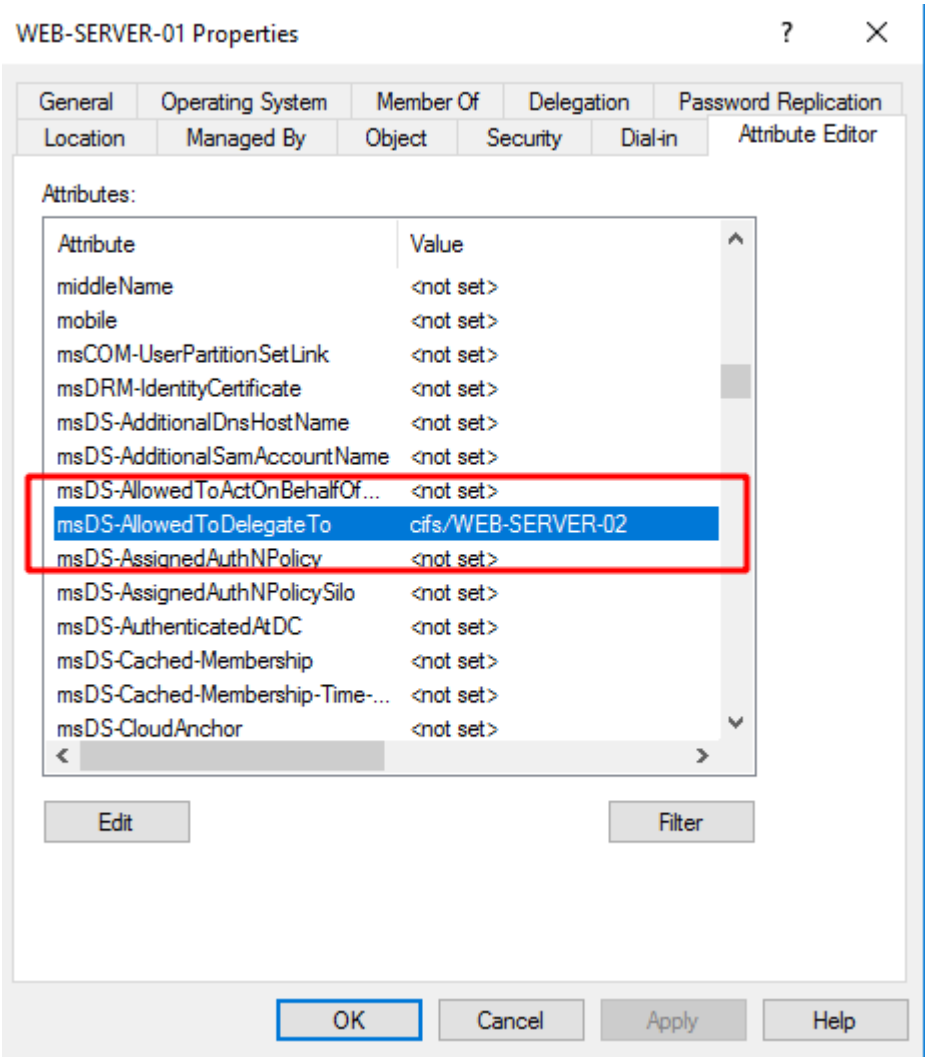
It is during this request that the Domain Controller, upon seeing this information, will verify that `Service A` has the right to authenticate to `Resource B` on behalf of the user.

Constrained Delegation - Classique

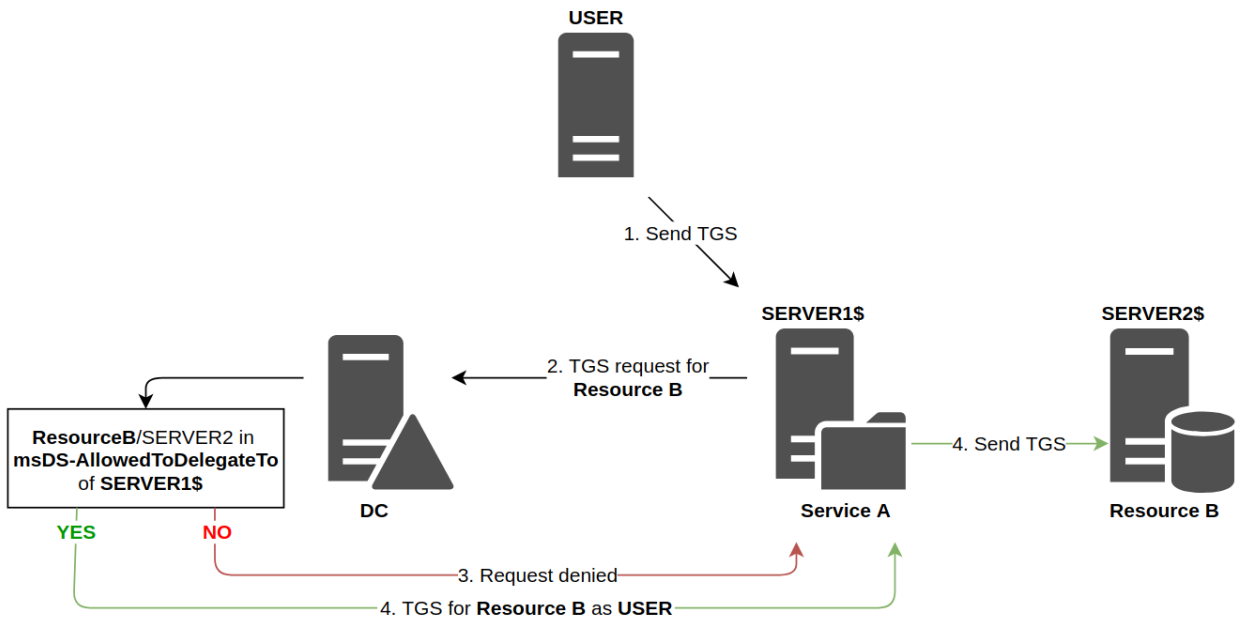
In the classic **Constrained Delegation** case (when delegation information is located in `Service A`), this information is found in the msDS-AllowedToDelegateTo attribute **of the requesting account**, thus of `Service A`. This attribute specifies the list of authorized SPN for the delegation.



For example here the `msDS-AllowedToDelegateTo` attribute will contain `cifs/WEB-SERVER-02`.



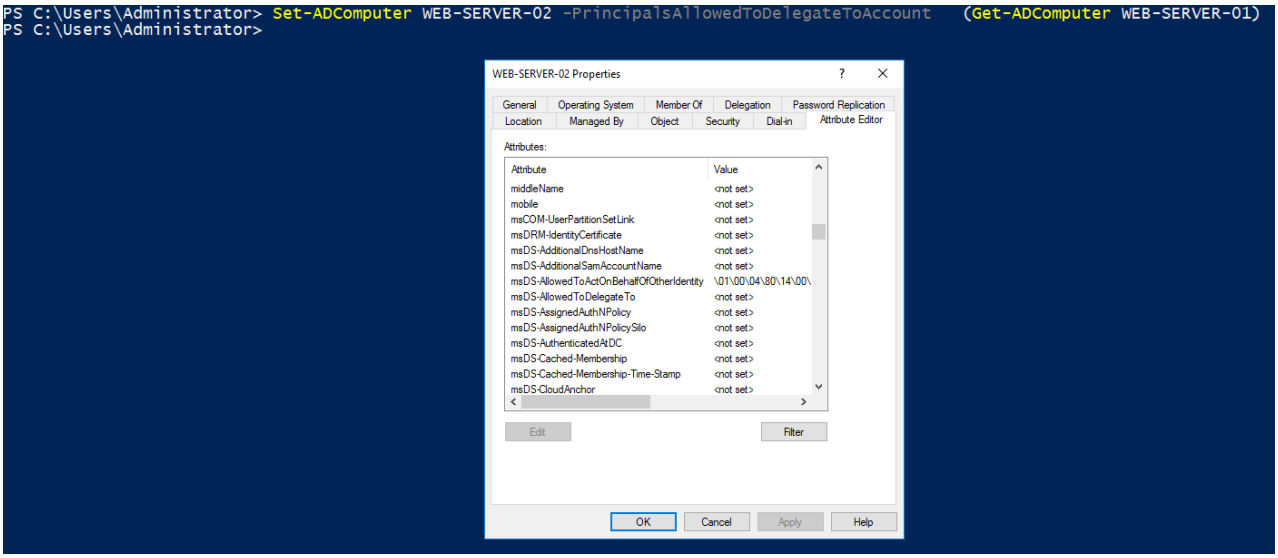
If the targeted SPN is present, then the Domain Controller sends back a valid TGS, with the name of the user, for the requested service. Here is a summary diagram :



Constrained Delegation - Resource Based

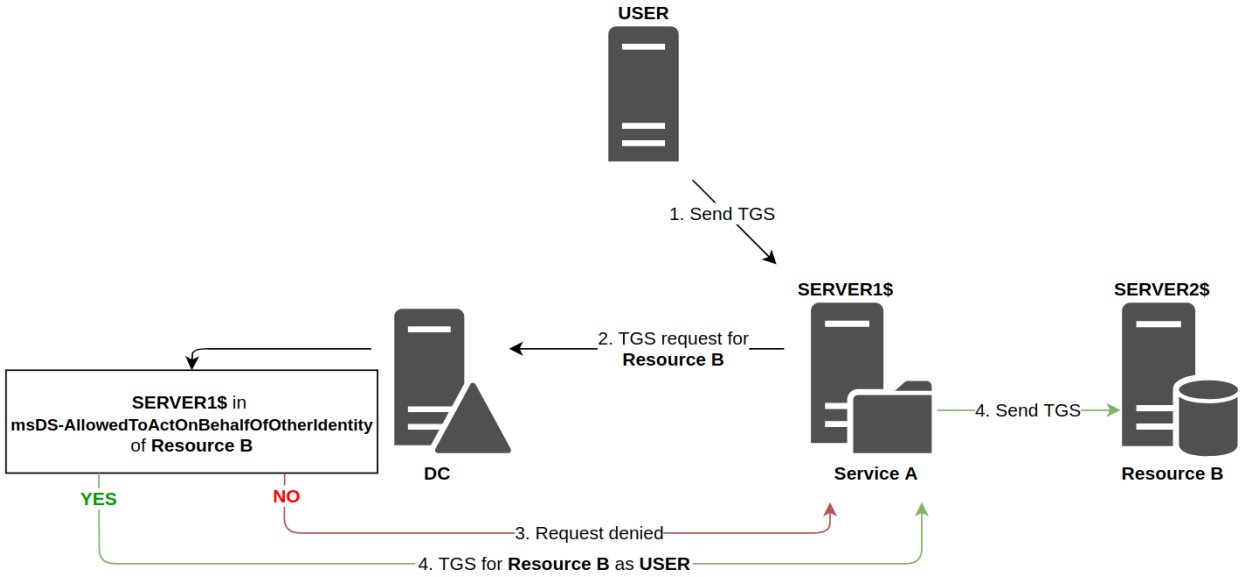
This time, the Domain Controller will look at the attributes of **Resource B** (instead of `Service A`). It will check that the account associated with `Service A` is present in the msDS-AllowedToActOnBehalfOfOtherIdentity attribute of the account linked to `Resource B`.

In order to set up RBCD on a resource, we need to use powershell.



`WEB-SERVER-01` is added to `WEB-SERVER-02` trusting list, so **msDS-AllowedToActOnBehalfOfOtherIdentity** is updated.

Then, when a service wants to access a resource, here's how it works:



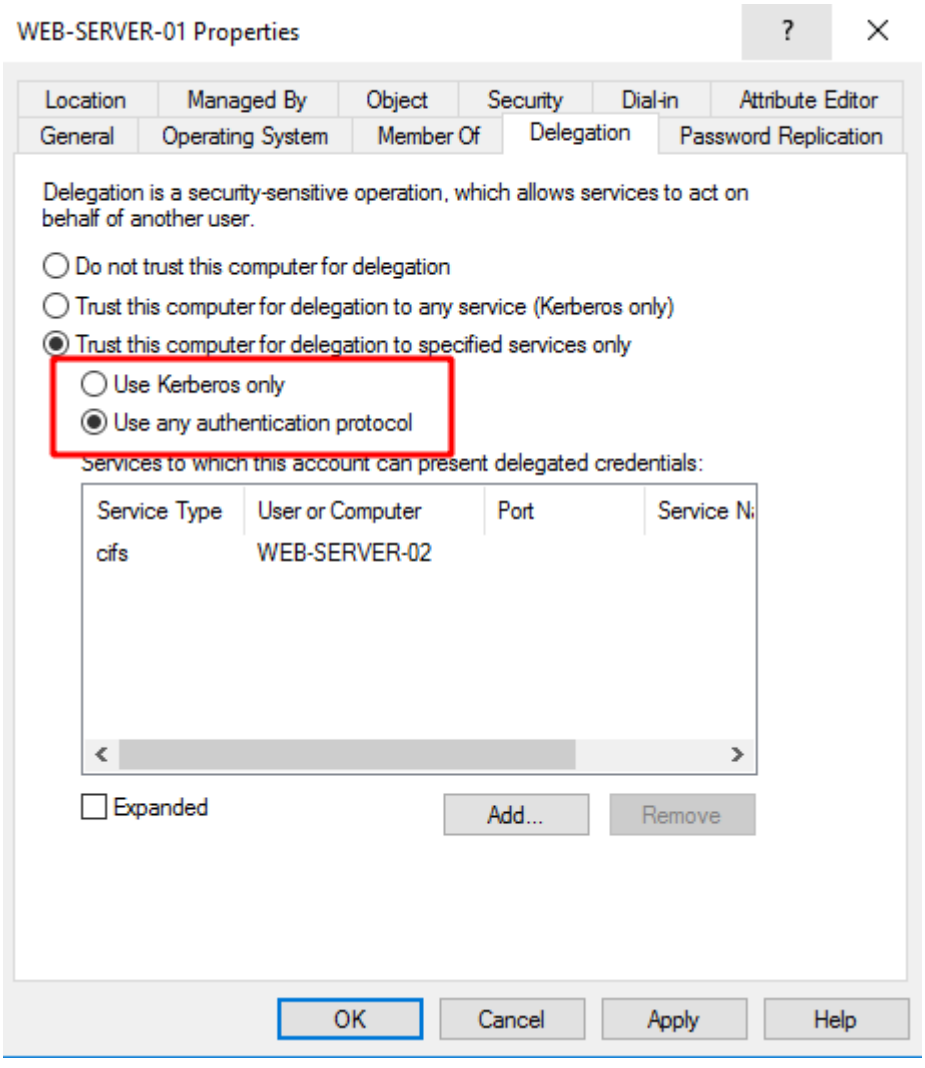
As the diagram shows, the technical functioning is similar to classic constrained delegation, however the responsibilities of each entity is radically different.

S4U2Self

If you are still there and you are not lost, you should have noticed that we haven't touched on the notion of protocol transition in this article. Indeed, in the explanation of constrained delegation, we assumed that `Service A` had a service ticket coming from `USER`, which was added in the `additional-tickets` field of the TGS request (**S4U2Proxy**). But sometimes the user may authenticate to the server in other ways than the Kerberos protocol (e.g. via NTLM, or even a Web form). In this case, the server is not in possession of the TGS sent by the user. Thus, `Service A` cannot fill the `additional-tickets` field as it did in the case described above.

That is why there is an extra step, possible through the S4U2Self extension, that `Service A` must perform. This step allows it to obtain a TGS for a user **arbitrarily chosen**. To do this, it makes a classic TGS request (KRB_TGS_REQ) except that instead of putting his own name in the `PA-FOR-USER` block (present in the pre-authentication part), it puts the name of a user **it chooses**.

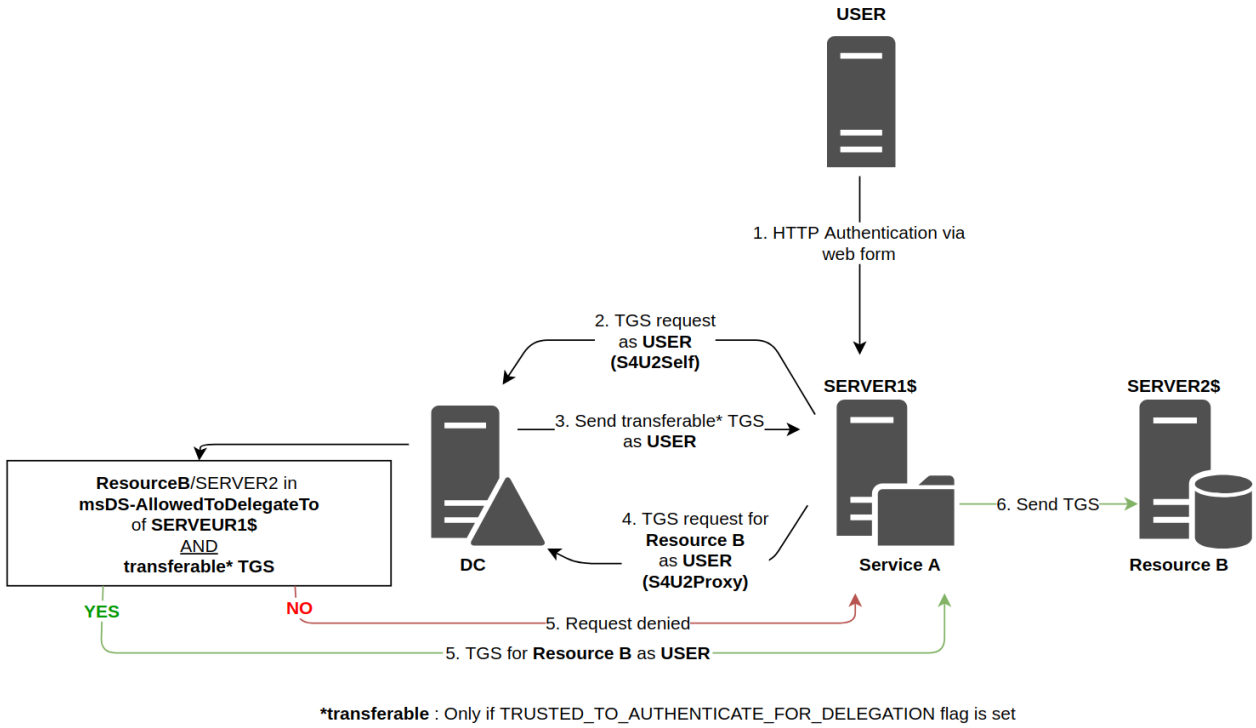
This ability to manage **protocol transition** is accepted by the Domain Controller only if it has been explicitly granted to the service account wishing to manage this delegation. It is in delegation management tab that an administrator can choose a constrained delegation using Kerberos only, therefore without protocol transition, or using any protocol, thus allowing **protocol transition**.



In the first case, the **TRUSTED_FOR_DELEGATION** flag is set on the account, and the service can only relay kerberos authentication. It cannot use the S4U2Self extension to create a ticket out of nowhere.

In the second case, the **TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION** flag is set. If this is the case, then the service with this flag **can pretend to be anyone** when accessing services in its list via the **S4U2Self** extension.

This diagram is getting a little bit more complicated, but I hope it remains relatively clear.



***transferable** : Only if TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION flag is set

If such an account is compromised, then all services to which that account is entitled to authenticate via delegation will also be compromised, since the attacker can create service tickets on behalf of arbitrary users, such as administrators.

Conclusion

I was thinking of doing an article that would describe kerberos delegation as well as associated attacks, however the explanations are much more dense than expected, so this article remains devoted to explanation. Associated attacks will be presented in other articles.

To sum up, there are three types of delegation:

1. **Unconstrained delegation**: In this case, the client sends a copy of his TGT to a service, and that service uses it to impersonate the client to any other service. Only an administrator can set this option on an account.

2. **Constrained delegation**: A list of resources is set on the service that wishes to delegate authentication. If **protocol transition** is allowed, then the service can pretend to be anyone when accessing resources in its list. In any case, only an administrator can set this option.
3. **Resource-based Constraint Delegation**: The final resource has a list of trusted accounts. All accounts in this list can delegate authentication when accessing the resource. Resources can modify this list as they wish, they don't need an administrator to update it.

If you have any questions or suggestions, do not hesitate, I'm all ears.

Resources

Big thanks to them for their clear explanations.

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

Similar posts