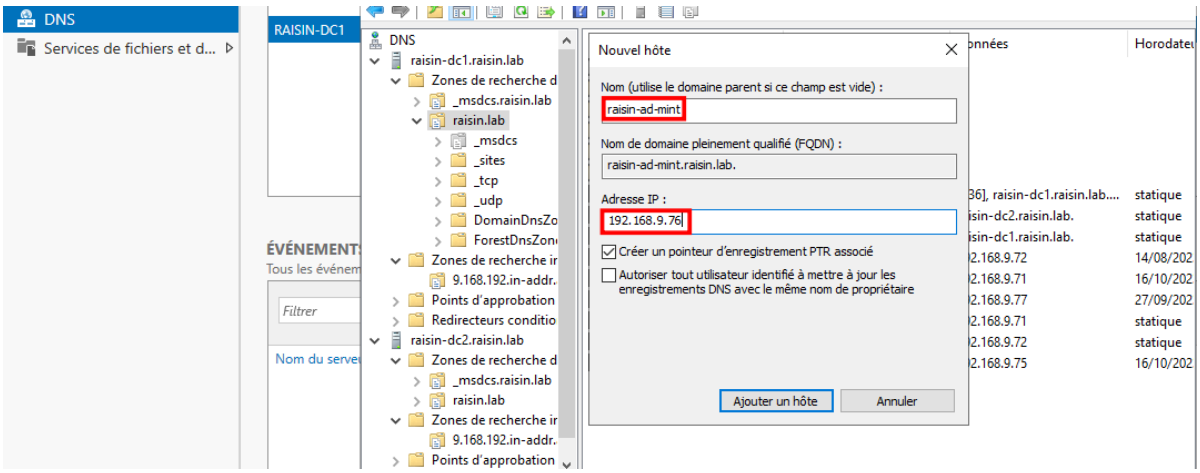
Jonction au domaine d'un linux

1°/ Déclaration DNS sur le contrôleur de domaine



2°/ Installation des paquets nécessaires sur la machine linux

```
$ apt install realmd krb5-user krb5-config libpam-krb5 krb5-auth-dialog
--------------------------------------------------------------------------
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  libgssrpc4 libkadm5clnt-mit12 libkadm5srv-mit12 libkdb5-10
Paquets suggérés :
  krb5-k5tls krb5-doc
Les NOUVEAUX paquets suivants seront installés :
  krb5-auth-dialog krb5-config krb5-user libgssrpc4 libkadm5clnt-mit12
  libkadm5srv-mit12 libkdb5-10 libpam-krb5 realmd
0 mis à jour, 9 nouvellement installés, 0 à enlever et 70 non mis à jour.
Il est nécessaire de prendre 170 ko/853 ko dans les archives.
Après cette opération, 4 145 ko d'espace disque supplémentaires seront utilisés.
Souhaitez-vous continuer ? [O/n]
```

3°/ Préparation du client

Il est recommandé de désactiver la configuration IPv6 du client.

Pour cela on peut commencer par commenter les lignes suivantes dans le fichier **/etc/hosts** :

```
#::1     ip6-localhost ip6-loopback
#fe00::0 ip6-localnet
#ff00::0 ip6-mcastprefix
#ff02::1 ip6-allnodes
#ff02::2 ip6-allrouters
```

Méthode 1 : Network-Manager

```
$ nmtui
```

Méthode 2 : Modification du GRUB

On modifie le fichier **/etc/default/grub** comme ceci :

```
GRUB_CMDLINE_LINUX_DEFAULT="ipv6.disable=1 quiet splash"
GRUB_CMDLINE_LINUX="ipv6.disable=1"
```

4°/ Préparation de la configuration Kerberos et jonction via Realmd

On remplace la configuration existante dans le fichier **/etc/krb5.conf** par celle-ci :

```
[libdefaults]
default_realm = RAISIN.LAB

# The following krb5.conf variables are only for MIT Kerberos.
kdc_timesync = 1
ccache_type = 4
forwardable = true
proxiable = true

default_tgs_enctypes = aes256-cts-hmac-sha1-96
default_tkt_enctypes = aes256-cts-hmac-sha1-96
permitted_enctypes = aes256-cts-hmac-sha1-96
```

```
# The following libdefaults parameters are only for Heimdal Kerberos.
fcc-mit-ticketflags = true

[realms]
RAISIN.LAB = {
    kdc = raisin-dc1.RAISIN.LAB
    kdc = raisin-dc2.RAISIN.LAB
    admin_server = raisin-dc1.RAISIN.LAB
    admin_server = raisin-dc2.RAISIN.LAB
    default_domain = RAISIN.LAB
}

[domain_realm]
.RAISIN.LAB = RAISIN.LAB
RAISIN.LAB = RAISIN.LAB

[appdefaults]
forwardable = true
```

Sachant que les valeurs de chiffrement possibles avec Windows sont :

- **DES_CBC_CRC**
- **DES_CBC_MD5**
- **RC4_HMAC_MD5**
- **AES128_HMAC_SHA1**
- **AES256_HMAC_SHA1**

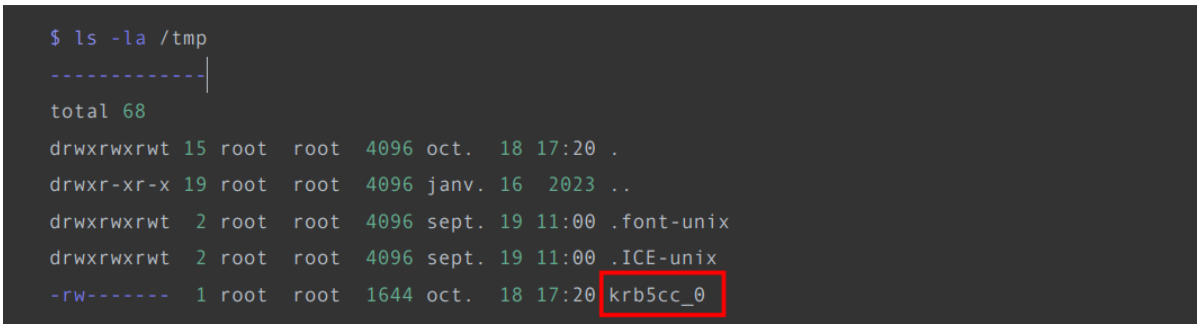
Rapide explication sur les autres paramètres :

kdc_timesync

La valeur **kdc_timesync** est un paramètre qui contrôle la manière dont le client Kerberos gère les horodatages (timestamps) reçus du Key Distribution Center (KDC), qui est le serveur d'authentification central de Kerberos.
La valeur 1 signifie que le client Kerberos doit synchroniser les horloges avec le KDC. Cela garantit que les horodatages utilisés dans les tickets Kerberos sont cohérents et que les communications entre le client et le KDC sont protégées contre les attaques de rejeu.

ccache_type

Le paramètre **ccache_type** spécifie le type de cache de tickets (credentials cache) utilisé par le client Kerberos pour stocker les tickets Kerberos.
Ici nous utilisons la valeur 4 qui correspond généralement à un cache de type "FILE-based credentials cache" (cache basé sur un fichier). Cela signifie que les tickets Kerberos seront stockés dans un fichier local sur le système de l'utilisateur (habituellement dans **/tmp**).



Les autres valeurs possibles sont :

- **MEMORY** : Ce type de cache stocke les tickets en mémoire vive
- **KEYRING** : Ce type de cache stocke les tickets dans le trousseau (keyring) du système, qui est un mécanisme de stockage sécurisé géré par le système d'exploitation (courant sur les systèmes Linux).
- **DIR** : Ce type de cache stocke les tickets dans un répertoire spécifié. Chaque ticket est généralement stocké dans un fichier distinct dans le répertoire.
- **KCM** : Le KCM est un gestionnaire de cache de tickets Kerberos qui offre des avantages supplémentaires en matière de performances et de sécurité par rapport aux autres types.
- **NONE** : L'utilisation de `NONE` signifie que le cache de tickets est désactivé. Cela empêche le stockage des tickets et oblige l'utilisateur à s'authentifier à chaque fois.

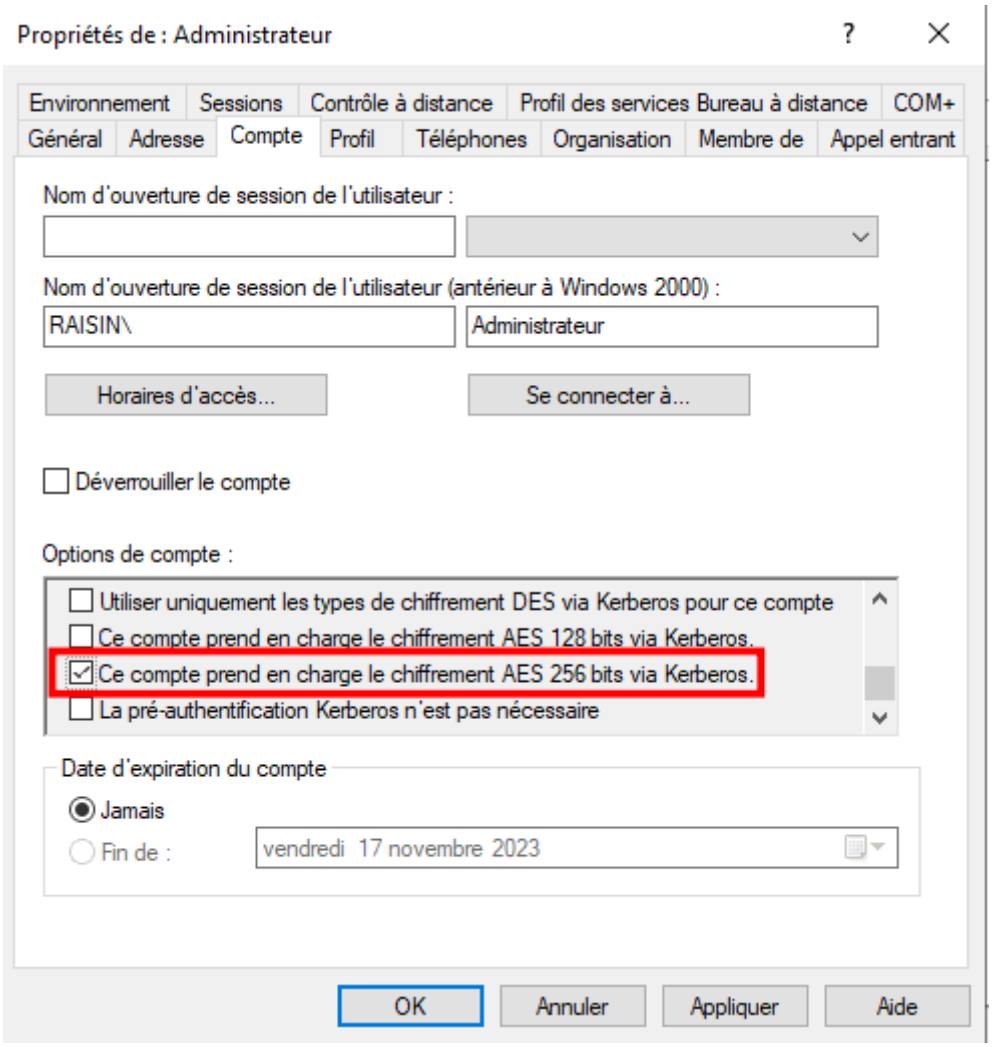
forwardable

Le paramètre **forwardable** indique si les tickets Kerberos obtenus par le client peuvent être transmis (forwarded) à un autre service. La valeur true signifie que les tickets sont transmissibles. Cela permet à un utilisateur de demander à un service de transférer son ticket à un autre service sans avoir à s'authentifier à nouveau. Pratique dans certaines situations, mais cela doit être géré avec prudence pour des raisons de sécurité.

proxiable

Le paramètre **proxiable** indique si les tickets Kerberos obtenus par le client peuvent être utilisés pour l'authentification par procuration (proxy). La valeur true signifie que les tickets sont proxy-capables. Cela signifie qu'un utilisateur peut utiliser un ticket pour demander à un service de l'authentifier auprès d'un autre service. Cette fonctionnalité est souvent utilisée dans des scénarios où un service intermédiaire agit en tant que proxy pour un utilisateur.

Ensuite, on va éditer le compte administrateur depuis le contrôleur de domaine et cocher l'option suivante :



Cela permettra à ce compte d'utiliser l'algorithme de chiffrement AES 256 bits pour sécuriser les échanges de données lors de l'authentification via le protocole Kerberos ce qui renforce la sécurité des échanges d'authentification et protège contre les attaques par force brute et autres méthodes d'attaque.

Depuis un terminal sur notre client linux, on procède à une demande de ticket kerberos pour le compte **administrateur** puis on vérifie les informations obtenues :

```
$ kinit administrateur
$ klist
-------
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: administrateur@RAISIN.LAB

Valid starting       Expires              Service principal
18/10/2023 17:20:41  19/10/2023 03:20:41  krbtgt/RAISIN.LAB@RAISIN.LAB
    renew until 19/10/2023 17:20:35
```

Notre ticket étant présent et vu que l'on a placé les bonnes options dans notre configuration **Kerberos** (forwardable/proxiable), nous allons pouvoir utiliser **realmd** afin qu'il procède à la jonction au domaine de notre machine en tant qu'**administrateur** (via le ticket **Kerberos**) sans avoir à nous authentifier.

On commence par demander à **realmd** de rechercher les informations disponibles sur le domaine que l'on souhaite rejoindre :

```
$ realm discover raisin.lab -v
-----------------------------
 * Resolving: _ldap._tcp.raisin.lab
 * Performing LDAP DSE lookup on: 192.168.9.71
 * Performing LDAP DSE lookup on: 192.168.9.72
 * Successfully discovered: raisin.lab
raisin.lab
  type: kerberos
  realm-name: RAISIN.LAB
  domain-name: raisin.lab
  configured: no
  server-software: active-directory
  client-software: sssd
  required-package: sssd-tools
  required-package: sssd
  required-package: libnss-sss
  required-package: libpam-sss
  required-package: adcli
  required-package: samba-common-bin
```

Les paquets nécessaires seront télécharger puis installer de manière automatique pendant le processus.

```
$ realm join raisin.lab -v
---------------------------
 * Resolving: _ldap._tcp.raisin.lab
 * Performing LDAP DSE lookup on: 192.168.9.71
 * Performing LDAP DSE lookup on: 192.168.9.72
 * Successfully discovered: raisin.lab
 * Unconditionally checking packages
 * Resolving required packages
 * Installing necessary packages: sssd-tools adcli sssd libnss-sss libpam-sss
 * LANG=C /usr/sbin/adcli join --verbose --domain raisin.lab --domain-realm
RAISIN.LAB --domain-controller 192.168.9.71 --login-type user --login-
ccache=/var/cache/realmd/realm-ad-kerberos-LP9JC2
 * Using domain name: raisin.lab
 * Calculated computer account name from fqdn: RAISIN-AD-MINT
 * Using domain realm: raisin.lab
 * Sending NetLogon ping to domain controller: 192.168.9.71
 * Received NetLogon info from: raisin-dc1.raisin.lab
 * Wrote out krb5.conf snippet to /var/cache/realmd/adcli-krb5-
ZTAOo8/krb5.d/adcli-krb5-conf-ZwwtOm
 * Using GSS-SPNEGO for SASL bind
 * Looked up short domain name: RAISIN
 * Looked up domain SID: S-1-5-21-236823562-4164089350-3284220594
 * Using fully qualified name: raisin-ad-mint.raisin.lab
 * Using domain name: raisin.lab
 * Using computer account name: RAISIN-AD-MINT
 * Using domain realm: raisin.lab
 * Calculated computer account name from fqdn: RAISIN-AD-MINT
 * Generated 120 character computer password
 * Using keytab: FILE:/etc/krb5.keytab
 * A computer account for RAISIN-AD-MINT$ does not exist
 * Found well known computer container at: CN=Computers,DC=raisin,DC=lab
 * Calculated computer account: CN=RAISIN-AD-MINT,CN=Computers,DC=raisin,DC=lab
```

```
 * Encryption type [3] not permitted.
 * Encryption type [1] not permitted.
 * Created computer account: CN=RAISIN-AD-MINT,CN=Computers,DC=raisin,DC=lab
 * Sending NetLogon ping to domain controller: 192.168.9.71
 * Received NetLogon info from: raisin-dc1.raisin.lab
 * Set computer password
 * Retrieved kvno '2' for computer account in directory: CN=RAISIN-AD-
MINT,CN=Computers,DC=raisin,DC=lab
 * Checking RestrictedKrbHost/raisin-ad-mint.raisin.lab
 *    Added RestrictedKrbHost/raisin-ad-mint.raisin.lab
 * Checking RestrictedKrbHost/RAISIN-AD-MINT
 *    Added RestrictedKrbHost/RAISIN-AD-MINT
 * Checking host/raisin-ad-mint.raisin.lab
 *    Added host/raisin-ad-mint.raisin.lab
 * Checking host/RAISIN-AD-MINT
 *    Added host/RAISIN-AD-MINT
 * Discovered which keytab salt to use
 * Added the entries to the keytab: RAISIN-AD-MINT$@RAISIN.LAB:
FILE:/etc/krb5.keytab
 * Added the entries to the keytab: host/RAISIN-AD-MINT@RAISIN.LAB:
FILE:/etc/krb5.keytab
 * Added the entries to the keytab: host/raisin-ad-mint.raisin.lab@RAISIN.LAB:
FILE:/etc/krb5.keytab
 * Added the entries to the keytab: RestrictedKrbHost/RAISIN-AD-
MINT@RAISIN.LAB: FILE:/etc/krb5.keytab
 * Added the entries to the keytab: RestrictedKrbHost/raisin-ad-
mint.raisin.lab@RAISIN.LAB: FILE:/etc/krb5.keytab
 ! Failed to update Kerberos configuration, not fatal, please check manually:
Setting attribute standard::type not supported
 * /usr/sbin/update-rc.d sssd enable
 * /usr/sbin/service sssd restart
 * Successfully enrolled machine in realm
```

Une fois cela fait, nous allons désactiver 3 services inutiles qui peuvent conduire à des bugs :

```
$ systemctl disable --now sssd-nss.socket sssd-pam-priv.socket sssd-pam.socket
-------------------------------------------------------------------------------
Removed /etc/systemd/system/sssd.service.wants/sssd-nss.socket.
Removed /etc/systemd/system/sssd.service.wants/sssd-pam.socket.
Removed /etc/systemd/system/sssd.service.wants/sssd-pam-priv.socket.
```

On en profitera pour regarder si la configuration NSS (Name Service Switch) a bien été mise à jour :

> nsswitch.conf est un fichier de configuration essentiel sur les systèmes Unix-like (y compris Linux) qui détermine comment le système résout les noms et les identités (par exemple, noms d'utilisateurs, de groupes, d'hôtes, de services, etc.) en fonction de différentes sources de données. En d'autres termes, il détermine l'ordre et les méthodes utilisés par le système pour rechercher des informations de services tels que les utilisateurs, les groupes, les hôtes, etc.)

```
$ cat /etc/nsswitch.conf
```

```
# /etc/nsswitch.conf
#
# Example configuration of GNU Name Service Switch functionality.
# If you have the `glibc-doc-reference' and `info' packages installed, try:
# `info libc "Name Service Switch"' for information about this file.

passwd:         files systemd sss
group:          files systemd sss
shadow:         files sss
gshadow:        files

hosts:          files mdns4_minimal [NOTFOUND=return] dns myhostname
networks:       files

protocols:      db files
services:       db files sss
ethers:         db files
rpc:            db files

netgroup:       nis sss
automount:      sss
```

5°/ Préparation de la configuration SSSD

Le fichier de configuration **sssd.conf** se trouve à l'emplacement **/etc/sssd/**.

Voici le contenu du fichier :

```
[sssd]
domains = raisin.lab
config_file_version = 2
services = nss, pam

[domain/raisin.lab]
default_shell = /bin/bash
krb5_store_password_if_offline = True
cache_credentials = True
krb5_realm = RAISIN.LAB
realmd_tags = manages-system joined-with-adcli
id_provider = ad
fallback_homedir = /home/%u@%d
ad_domain = raisin.lab
use_fully_qualified_names = True
ldap_id_mapping = True
access_provider = ad
```

default_shell

On peut remplacer la valeur par autre chose selon le type de shell que l'on préfère : zsh, ksh, ...

krb5_store_password_if_offline

On laissera la valeur sur **True**. Cela permettra aux utilisateurs de s'authentifier même lorsque le système ne peut pas accéder au serveur Kerberos, par exemple, en cas de panne du serveur ou en mode hors ligne.

cache_credentials

On laissera la valeur sur True. Cela signifie que les tickets Kerberos obtenus lors de l'authentification sont stockés temporairement sur le système local. Cela améliore également l'expérience de l'utilisateur (pas besoin de se réauthentifier pour chaque accès à une ressource) et permet de réduire la charge sur les serveurs d'authentification Kerberos.

id_provider

On a 4 choix possibles :

- **ad**
- **ldap**
- **ipa**
- **local**

Dans notre cas, on laissera sur **ad**.

fallback_homedir

Si le chemin du répertoire utilisateur n'est pas mentionnée dans les attributs étendus de son compte, alors c'est ce chemin qui sera effectif.

Ici nous avons **/home/%u@%d** ce qui donnerait pour l'utilisateur toto :

/home/toto@raisin.lab/

Pour plus de clarté, je recommande toutefois la valeur : **/home/%u**

use_fully_qualified_names

Je recommande de passer la valeur à **False**.

Si on laisse la valeur sur **True**, alors le système attendra à chaque fois des valeurs de noms d'utilisateurs ou de groupes de type **FQN** : par exemple, avec une valeur à **True**, l'utilisateur devra s'authentifier en tant que **toto@raisin.lab** à chaque fois au lieu de **toto**.

ldap_id_mapping

Je recommande de laisser la valeur sur **True**.

Cela va permettre de générer des **UID**/**GID** locaux pour chaque utilisateur. Ceux-ci sont générés à partir du **SID** (Security Identifier) de l'utilisateur.

Rappel : le **SID** est l'identificateur de sécurité global unique pour un objet dans un domaine Windows, tandis que le **RID** est une partie du SID qui identifie l'objet de manière unique au sein de ce domaine. En somme, le **SID** est une valeur qui est composée d'un partie fixe (domaine) ainsi que d'une partie variable **RID** (objet).

De même ici, l'**UID** de l'utilisateur ainsi généré aura une partie fixe et variable :

```
$ id administrateur|grep uid|cut -d " " -f 1
uid=211000500(administrateur)
$ id gcorle|grep uid|cut -d " " -f 1
uid=211001110(gcorle)
$ id ylegallais|grep uid|cut -d " " -f 1
uid=211001111(ylegallais)
```

On note la partie "domaine" fixe **21100** puis les parties variables en fonction des utilisateurs. Noter également que la partie variable s'incrémente de +1 pour chaque nouvel utilisateur créé. (Je suis le 1er à avoir créé un compte puis ce fut au tour de Yann par la suite)

access_provider

Cette variable peut prendre 3 valeurs :

- **ad**
- **ipa**
- **simple**

On laissera sur **ad** dans notre cas.

Voilà qui couvre l'explication des valeurs par défaut. Je vous propose un fichier maison qui comporte d'autre variables et qui est plus complet :

```
[sssd]
domains = UMR5107.LAB
config_file_version = 2
services = nss, pam

[domain/umr5107.lab]
ad_gpo_ignore_unreadable = True
#ad_gpo_access_control = permissive
default_shell = /bin/bash
krb5_store_password_if_offline = True
cache_credentials = True
krb5_realm = UMR5107.LAB
realmd_tags = manages-system joined-with-adcli
id_provider = ad
fallback_homedir = /home/%u
ad_domain = UMR5107.LAB
ad_server = DC1.UMR5107.LAB
ad_backup_server = DC1.UMR5107.LAB, DC3.UMR5107.LAB
use_fully_qualified_names = False
ldap_id_mapping = True
access_provider = ad
auth_provider = ad
chpass_provider = krb5
ldap_schema = ad
dyndns_update = true
dyndns_refresh_interval = 43200
dyndns_update_ptr = true
dyndns_ttl = 3600
```

Je vous laisse le soin de rechercher les autres options disponibles et de les modifier selon vos besoins.

Une fois que l'on a copié / collé ces informations dans notre fichier **sssd.conf**, on peut redémarrer le service :

```
$ service sssd restart
```

Le service **sssd** possède un système de cache. Si vous le souhaitez, vous pouvez purger celui-ci afin de bénéficier des dernières modifications que vous auriez faites sur votre **Active Directory** :

```
$ sss_cache -E
```

6°/ Mise à jour des attributs étendus de l'utilisateur

⚠️ Cette partie, qui peut être un peu longue et rébarbative en fonction du nombre d'utilisateurs que vous avez dans votre domaine, peut être facultative dans certains cas et selon les certaines configurations. Cependant je la recommande **fortement** afin de couvrir une interopérabilité maximale entre les environnements **Linux** et **Windows** sur votre production !⚠️

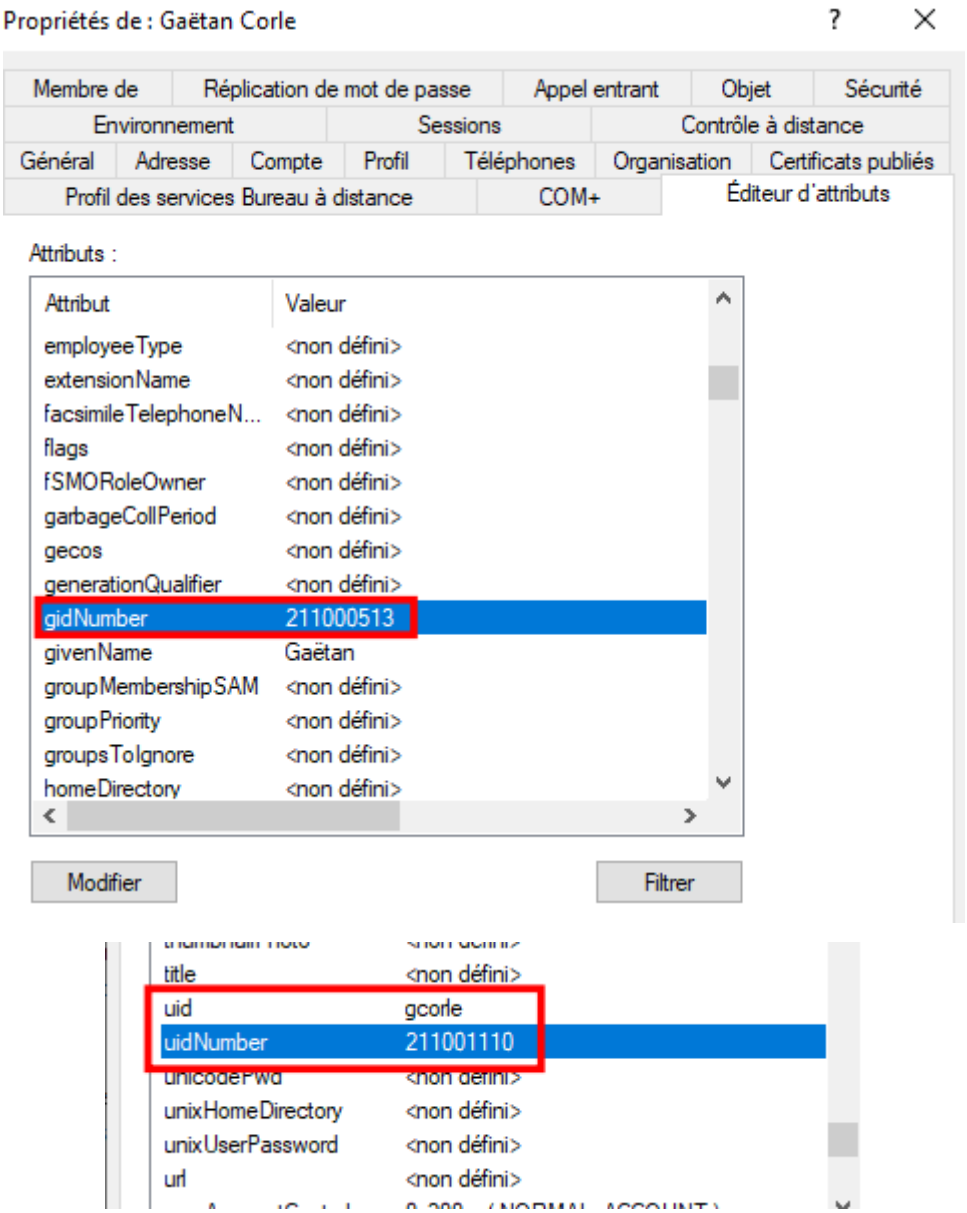
Maintenant que **SSSD** est configuré et que les utilisateurs et groupes possèdent leur propres **UID**/**GID**, il convient de les renseigner au sein d'**Active Directory** afin que les services tiers qui auraient besoin de connaître ces valeurs puissent y avoir accès (par exemple, dans le cas d'un NAS qui partage des données via SMB/NFS).

On se connecte alors à notre **DC** et on édite un utilisateur, ici par exemple **gcorle**.

En même temps, depuis notre Linux, on ouvre un terminal et on tape la commande suivante :

```
$ id gcorle
-----------
uid=211001110(gcorle) gid=211000513(utilisateurs du domaine)
groupes=211000513(utilisateurs du domaine),211000519(administrateurs de
l'entreprise),211000518(administrateurs du schéma),211000520(propriétaires
créateurs de la stratégie de groupe),211000512(admins du
domaine),211000572(groupe de réplication dont le mot de passe rodc est refusé)
```

Il suffit ensuite de reporter les valeurs dans les bons attributs :


Propriétés de : Gaëtan Corle
Membre de
Réplication de mot de passe
Appel entrant
Objet
Sécurité
Environnement
Sessions
Contrôle à distance
Général
Adresse
Compte
Profil
Téléphones
Organisation
Certificats publiés
Profil des services Bureau à distance
COM+
Éditeur d'attributs
Attributs :
Attribut
Valeur
employeeType <non défini>
extensionName <non défini>
facsimileTelephoneN... <non défini>
flags <non défini>
fSMORoleOwner <non défini>
garbageCollPeriod <non défini>
gecos <non défini>
generationQualifier <non défini>
gidNumber 211000513
givenName Gaëtan
groupMembershipSAM <non défini>
groupPriority <non défini>
groupsToIgnore <non défini>
homeDirectory <non défini>
Modifier
Filtrer

title <non défini>
uid gcorle
uidNumber 211001110
unixHomeDirectory <non défini>
unixUserPassword <non défini>
url <non défini>