HE2B
ISIB

SYSTEMES OPEN SOURCE

Linux et RabbitMQ

Par Jean-Hubert ABA'A

Enseignant : Thierry SEVERS

2021●2022

Table des matières

Partie I : Linux

Objectifs :

- ✓ Installer et configurer un réseau avec des solutions Open Source
- ✓ Maitriser la configuration du réseau via les fichier .yaml
- ✓ Maitriser les commandes de base de l'administration système sous Linux
- ✓ Connaitre SAMBA et Kerberos et pourvoir les utiliser
- ✓ Administrer un domaine via RSAT
- ✓ Créer des GPOs

I. Description de l'environnement de travail

Dans l'optique d'effectuer notre travail, nous avons mis en place un environnement de travail virtuel sous le logiciel VirtualBox d'Oracle. Notre banc d'essai est constitué de 3 machines virtuelles. Les configurations sont répertoriées dans le tableau suivant :

Ordinateur (OS)	**IP**		**DNS**		**Rôle**
Ubuntu OS 20.04	NAT	10.0.2.15	NAT		Passerelle
	Intnet	192.168.1.6	Intnet		
Ubuntu Server	192.168.1.7		192.168.1.7		Serveur
Windows 10 Ent	192.168.1.12		192.168.1.7		Client/RSAT

La configuration TCP/IP du banc d'essai peut se faire intuitivement via l'interface graphique sur les machines Windows et Ubuntu client. En revanche, Ubuntu Server est dénué d'interface

graphique type Gnome ; nous configurerons donc notre server via Netplan[1] et son fichier yaml que voici :

```
network:
        ethernets:
                enp0s3:
                        addresses: [192.168.1.7/24]
                        gateway4: 192.168.1.6
                        nameservers:
                                addresses: [192.168.1.7]

        version: 2
```

PS : Il convient de mettre le serveur client DNS de 8.8.8.8 durant la configuration au cas où il est client DNS de lui-même et que la résolution DNS ne se fait pas.

Il suffit ensuite de lancer la commande suivante pour activer le routage et permettre aux machines sur le LAN de se connecter à internet:

```
IPTABLES -T NAT -A POSTROUTING -O ENP0S3 -J MASQUERADE
```

II. Le Domaine de Sécurité

Un domaine est un ensemble de ressources sur un réseau Windows NT utilisant les mêmes règles de sécurité.

Dans notre cas, et au vu de notre installation majoritairement Open Source, nous devrons résoudre un problème majeur : le protocole SMB[2] utilisé sur Windows NT n'est pas implémenté nativement avec Ubuntu Server. Pour l'utiliser, nous installerons Samba, qui est une implémentation SMB sous Unix. C'est un logiciel libre qui permet à n'importe quelle petite machine sous Unix d'être contrôleur de domaine NT (Nous le verrons à l'œuvre dans nos manipulations). Son fichier de configuration est accessible avec la commande[3]:

```
SUDO VIM /ETC/SAMBA/SMB.CONF
```

Kerberos[4] que nous installons aussi est un protocole d'authentification réseau qui repose sur un mécanisme de clés secrètes et l'utilisation de tickets, et non de mots de passe en clair, évitant ainsi le risque d'interception frauduleuse des mots de passe des utilisateurs (Petters, s.d.).

Le principe de Kerberos est de s'adresser à un serveur d'authentification qui remet un '*ticket*' (on parle plutôt de certificat), avec lequel nous pouvons accéder à la ressource demandée.

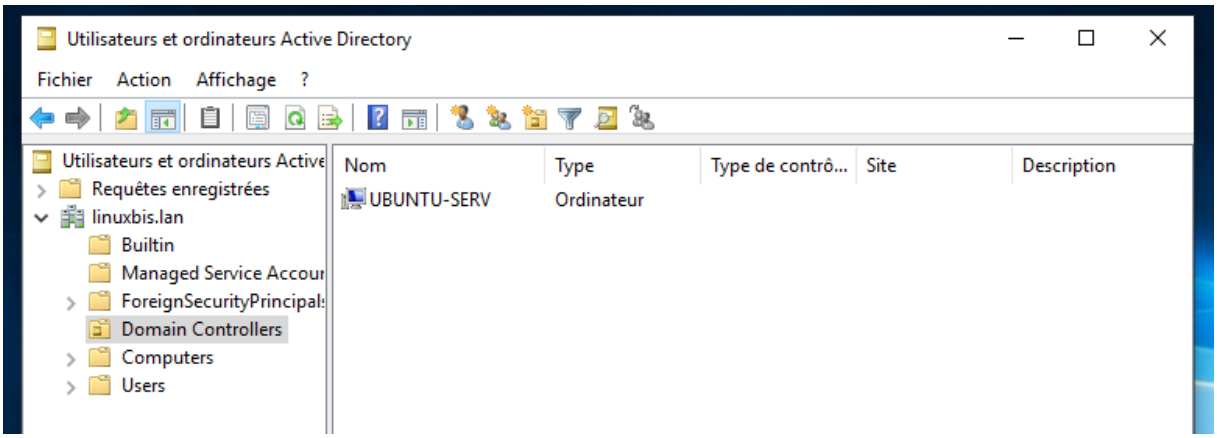
[1] Concrètement Netplan permet de créer une description de la configuration du réseau à partir d'un fichier yaml et la soumettre au Gestionnaire de Réseau.
[2] Server message Block : Protocole de Microsoft et d'Intel fonctionnant sur NetBIOS et permettant le partage des ressources à travers un réseau. L'une de ses variantes est Windows NT
[3] Nous avons suivi le tuto disponible sur la page du cours pour l'installation de Samba et Kerberos. Il n'est pas nécessaire de réécrire cette procédure ici.
[4] Le nom vient du grec, qui a donné Cerbère : chien à 3 têtes qui gardait les enfers.

Nous nous servirons de Kerberos pour se connecter en tant qu'administrateur en suivant le consignes du cours (SEVERS, 2021). Nous mettrons en annexes les liens vers ces fichiers[5].

Problème : Impossible de résoudre le DNS du server Linux sur Windows.

Solution : supprimer le DNS 8.8.8.8 et le remplacer par le DNS du server Linux.

A ce stade de notre projet, Samba est installé et configuré. Kerberos aussi. Nous pouvons le voir à partir de la machine Windows.



III. Ajout de la machine Ubuntu20 dans le domaine.

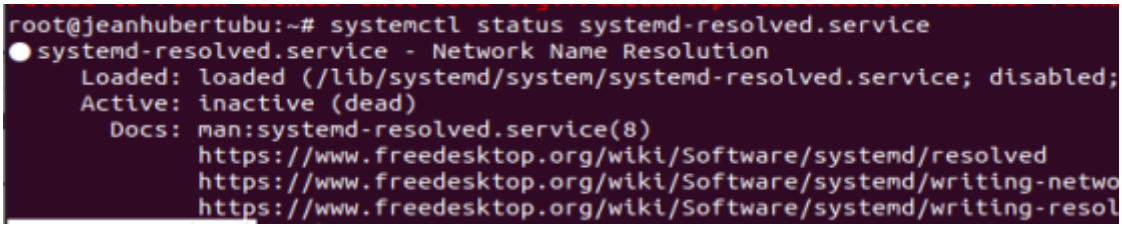
```
# Let NetworkManager manage all devices on this system
network:
        version: 2
        renderer: networkd
        ethernets:
                enp0s8:
                        addresses: [192.168.1.6/24]
                        gateway4: 10.0.2.15
                        nameservers:
                                addresses: [192.168.1.7, 8.8.8.8]
                        dhcp4: false
                        dhcp6: false
~
~
```

Pour la suite de la configuration, notre machine dois pourvoir résoudre les noms. Mais nous avons été confrontés au problème suivant :

```
root@jeanhubertubu:~# nslookup ubuntu-serv.linuxbis.lan
;; connection timed out; no servers could be reached
```

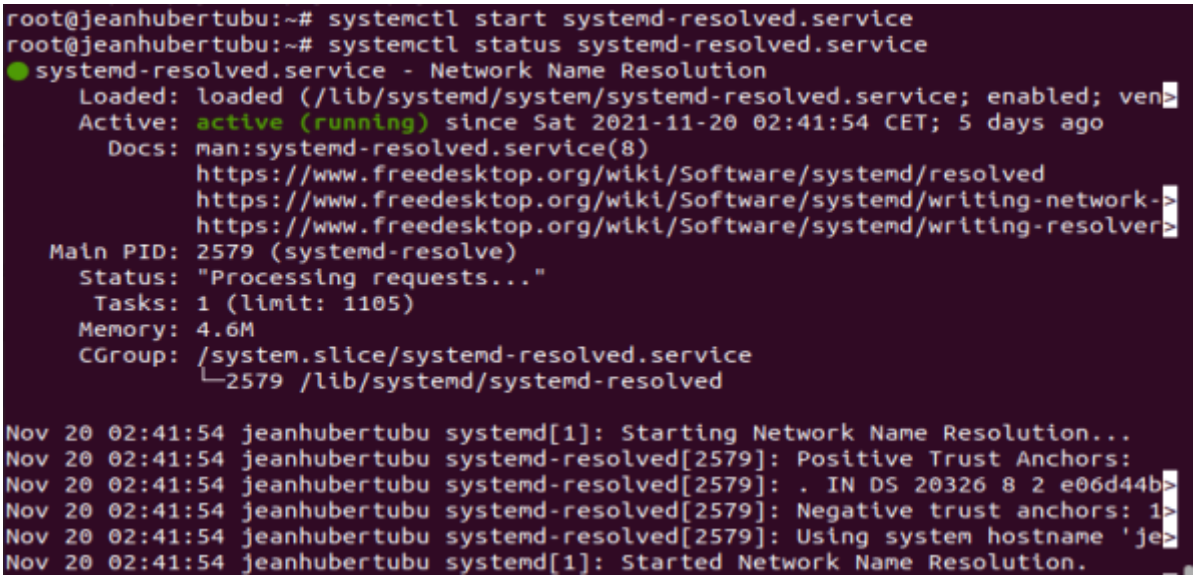
Et les statuts :

[5] Tous disponibles sur le Git Du cours : https://github.com/jhabaa/OpenSource-RabbitMQ.git

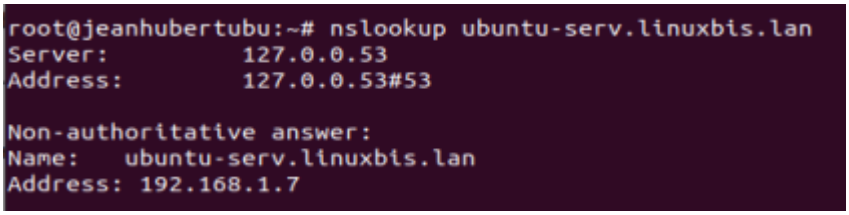


L'erreur nous indique que le daemon de résolution de noms n'est pas actif. Nous allons l'activer avec les commandes :

```
SYSTEMCTL ENABLE SYSTEMD-RESOLVED.SERVICE

SYSTEMCTL START SYSTEMD-RESOLVED.SERVICE

SYSTEMCTL STATUS SYSTEMD-RESOLVED.SERVICE
```



Maintenant que le service est lancé et actif, nous pouvons résoudre le nom du server à partir de la machine Linux:



Réglage de la date

Pour pouvoir intégrer un domaine, il faudrait que les dates et heures de la machine soit synchronisée avec l'heure du serveur. Nous allons donc activer le daemon NTP sur le serveur, et récupérer l'heure sur la machine cliente :

```
TIMEDATECTL SET-NTP TRUE

SYSTEMCTL RESTART SYSTEMD-TIMESYNCD.SERVICE
```

```
TIMEDATECTL –ADJUST-SYSTEM-CLOCK
```

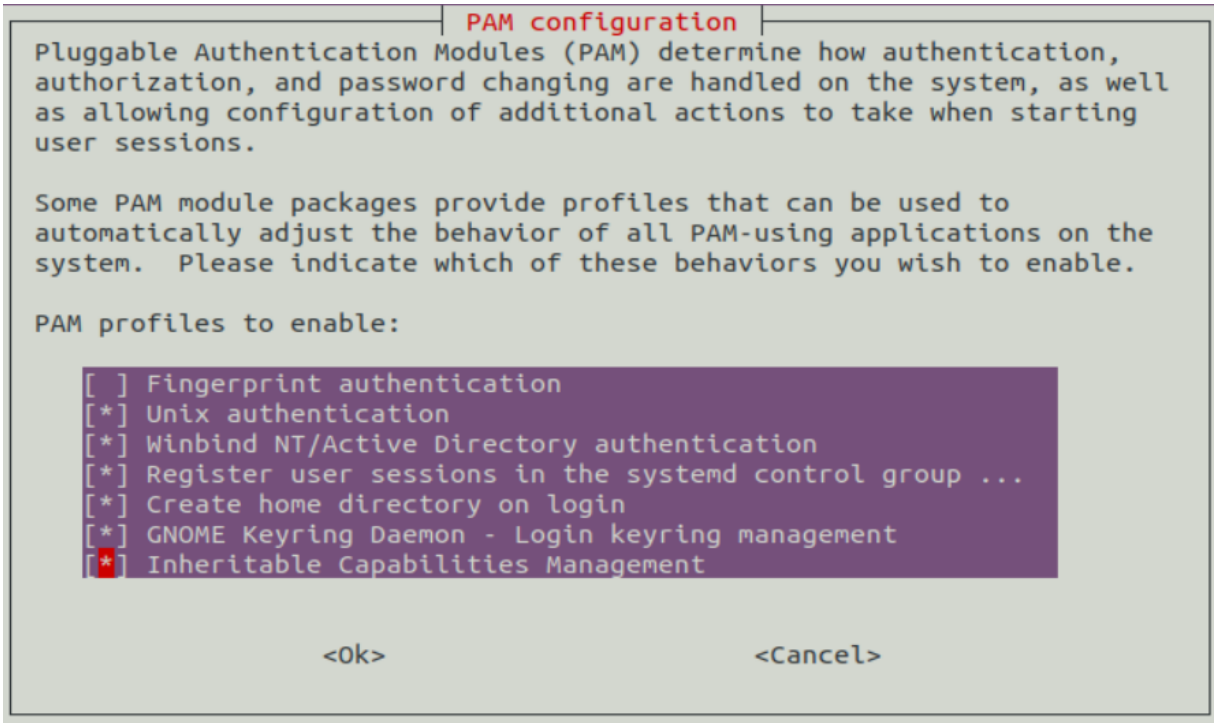
```
root@jeanhubertubu:~# vim /etc/systemd/timesyncd.conf
root@jeanhubertubu:~# timedatectl set-ntp true
root@jeanhubertubu:~# systemctl restart systemd-timesyncd.service
root@jeanhubertubu:~# timedatectl --adjust-system-clock
               Local time: Do 2021-11-25 06:47:17 CET
           Universal time: Do 2021-11-25 05:47:17 UTC
                 RTC time: Do 2021-11-25 05:47:17
                Time zone: Europe/Brussels (CET, +0100)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

Une fois les heures des machines locales et du serveur synchronisées, nous pouvons procéder à l'installation de Samba

```
SUDO APT INSTALL SAMBA KRB5-CONFIG KRB5-USER WINBIND LIBPAM-WINBIND LIBNSS-WINBIND
```

Ensuite mettre à jour la configuration PAM

```
PAM-AUTH-UPDATE
```



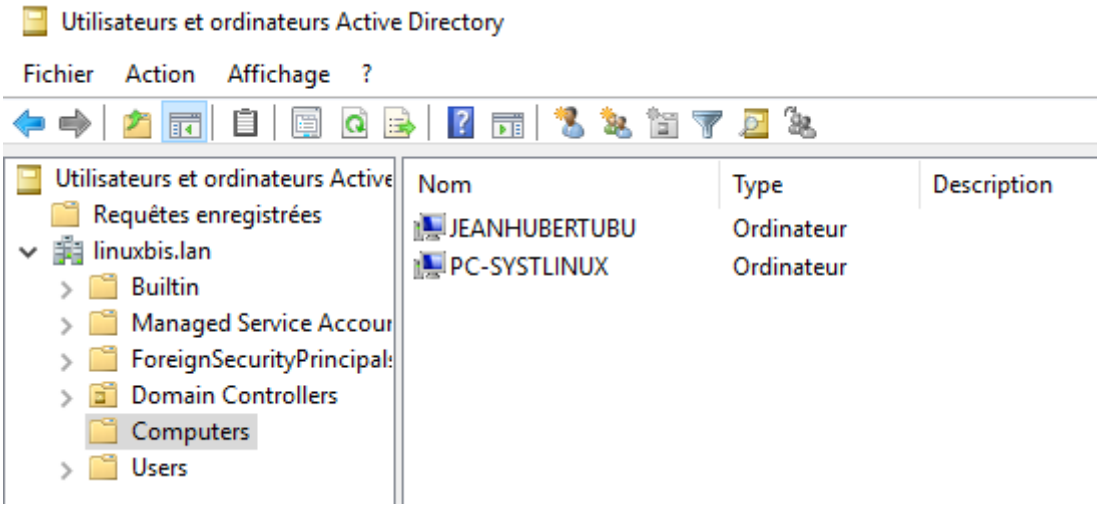

```
127.0.0.1       localhost
127.0.1.1       jeanhubertUBU.linuxbis.lan jeanhubertUBU
```

```
[realms]
        LINUXBIS.LAN = {
                kdc = ubuntu-serv.linuxbis.lan
                admin_server = ubuntu-serv.linuxbis.lan
        }
```

```
root@jeanhubertubu:~# kinit Administrator
Password for Administrator@LINUXBIS.LAN:
Warning: Your password will expire in 4 days on Do 25 Nov 2021 14:44:34
root@jeanhubertubu:~#
```

```
root@jeanhubertubu:~# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: Administrator@LINUXBIS.LAN

Valid starting       Expires              Service principal
2021-11-21 07:32:05  2021-11-21 17:32:05  krbtgt/LINUXBIS.LAN@LINUXBIS.LAN
        renew until 2021-11-22 07:32:01
```

```
root@jeanhubertubu:~# kinit -VR
Using principal: Administrator@LINUXBIS.LAN
Initialized cache
Stored credentials
Authenticated to Kerberos v5
root@jeanhubertubu:~# sudo net ads join -U Administrator
Enter Administrator's password:
gse_get_client_auth_token: gss_init_sec_context failed with [ Miscellaneous fai
lure (see text): Clock skew too great](2529638949)
kinit succeeded but ads_sasl_spnego_gensec_bind(KRB5) failed for ldap/ubuntu-se
rv.linuxbis.lan with user[Administrator] realm[LINUXBIS.LAN]: The attempted log
on is invalid. This is either due to a bad username or authentication informati
on.
gse_get_client_auth_token: gss_init_sec_context failed with [ Miscellaneous fai
lure (see text): Clock skew too great](2529638949)
kinit succeeded but ads_sasl_spnego_gensec_bind(KRB5) failed for ldap/ubuntu-se
rv.linuxbis.lan with user[JEANHUBERTUBU$] realm[LINUXBIS.LAN]: The attempted lo
gon is invalid. This is either due to a bad username or authentication informat
ion.
gse_get_client_auth_token: gss_init_sec_context failed with [ Miscellaneous fai
lure (see text): Clock skew too great](2529638949)
Using short domain name -- LINUXBIS
Joined 'JEANHUBERTUBU' to dns domain 'linuxbis.lan'
DNS Update for jeanhubertubu.linuxbis.lan failed: ERROR_DNS_GSS_ERROR
DNS update failed: NT_STATUS_UNSUCCESSFUL
```



La machine Ubuntu est bien ajoutée au domaine LINUXBIS.LAN.

IV. Gestion du domaine avec RSAT et l'outil samba-tool sur le contrôleur Linux.

Création des utilisateurs, des groupes et des UO[6]

Unité Organisationnelle

Nous allons en créer une à l'aide samba-tool. Il serait dommage de s'en priver quand notre objectif est effectivement de voir à l'œuvre l'outil Samba. La commande est simple :

```
SAMBA-TOOL OU CREATE ISIB
```

Utilisateurs

Avec samba-tool nous pouvons créer des comptes utilisateurs à partir du serveur Linux Controleur de domaine. Nous allons en créer 3 : MA1user, Hugo et Hélène. Ces deux derniers feront partie de l'UO ISIB.

```
SAMBA-TOOL USER ADD MA1USER ISIB1234.

SAMBA-TOOL USER ADD HUGO ISIB1234. –USEROU OU=ISIB

SAMBA-TOOL USER ADD HELENE ISIB1234. –USEROU OU=ISIB
```

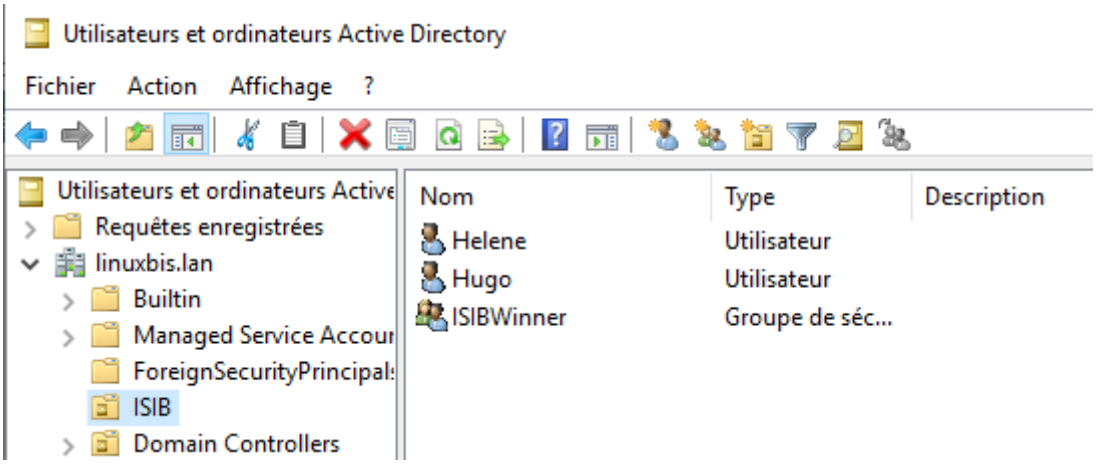
Groupes

Nous pouvons aussi créer un groupe dans l'unité organisationnelle ISIB

```
SAMBA-TOOL GROUP ADD ISIBWINNER –GROUPOU OU=ISIB
```

Et nous pouvons retrouver les résultats de nos commandes à partir de RSAT sur Windows.

[6] Organizational Unit : partition administrative d'un Active Directory. En fait, c'en est juste un morceau, pouvant lui-même contenir d'autres OU, de façon a pouvoir gérer plus facilement l'annuaire.




```
SAMBA-TOOL USER LIST
```

GPOs[7]

Samba-tool dispose une fonction pour la gestion des GPO. Mais si nous décidons de directement l'appliquer, le terminal nous renvoie ce message :

```
root@ubuntu-serv:~# samba-tool gpo create newGpotest
ERROR(runtime): uncaught exception - (3221225524, 'The object name is not found.')
  File "/usr/lib/python3/dist-packages/samba/netcmd/__init__.py", line 186, in _run
    return self.run(*args, **kwargs)
  File "/usr/lib/python3/dist-packages/samba/netcmd/gpo.py", line 1192, in run
    cldap_ret = net.finddc(domain=self.lp.get('realm'), flags=flags)
root@ubuntu-serv:~#
```

A présent, rentrons dans la configuration. Winbind est installé, et sa configuration est[8] :

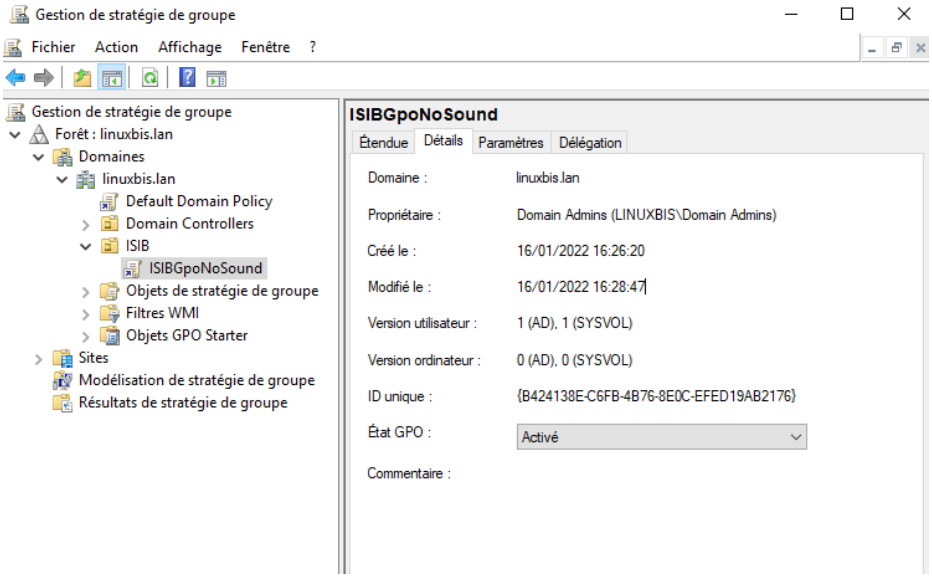
```
# /etc/nsswitch.conf
#
# Example configuration of GNU Name Service Switch functionality.
# If you have the `glibc-doc-reference' and `info' packages installed, try:
# `info libc "Name Service Switch"' for information about this file.

passwd:         files systemd winbind
group:          files systemd winbind
shadow:         files
gshadow:        files

hosts:          files dns wins
networks:       files

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

netgroup:       nis
~
~
```

[7] Group Policy Object : Objet stockant une politique de groupe dans un GPC(Group Policy container) ou un GPT(Group Policy Template) sur une machine.

[8] Dans le fichier /etc/nsswitch.conf

Dans l'optique de configurer les polices de groupes Samba, nous avons premièrement installé les templates ADMX sur la machine windows. L'idée, était de créer des GPO à partir de samba-tool en se servant des admx.

Nous n'avons à date malheureusement pas encore résolu ce problème. Nous créerons donc une GPO à travers l'outil RSAT de Windows qui nous permettra de désactiver le son des utilisateurs de l'UO ISIB.



V. Conclusion de partie

Au terme de ces manipulations de prise en main de Samba avec kerberos pour la gestion d'un domaine basé sur de l'OpenSource, nous pouvons dire sans rougir que l'OpenSource peut aisément remplacer les solutions propriétaires, et qu'il est tout à fait possible de gérer son domaine Active Directory avec Samba. Néanmoins, l'ajout d'une machine Windows dans notre configuration nous montre bien les failles possibles de notre système pour une implémentation plus large. Malgré une large communauté, les problèmes restent légion, et il est parfois difficile d'avoir une aide rapide et concise.

Partie II : RabbitMQ

I. Présentation du programme

RabbitMQ est un broker de messages : Son rôle est d'accepter et de faire suivre les messages. Pour faire une analogie avec le monde réel RabbitMQ est comme une boite aux lettres, un bureau postal et postier.

La seule différence avec un véritable bureau de poste, c'est que RabbitMQ ne traite pas de papiers, mais des données binaires. Dans le jargon MQ, nous avons principalement :

- Le producteur : Qui est en fait le programme expéditeur du message.
- La file : C'est la version informatique de notre boite aux lettres. Les messages peuvent juste y être stockés en attente de réception.

II. Installation

Dans le cadre de notre démonstration, nous installerons RabbitMQ à partir d'un conteneur Docker. Pour cela, il suffit de taper la commande :

```
Docker pull rabbitmq :management
```

Cette version nous permet de bénéficier d'une interface web pour la gestion et le contrôle de notre programme.

```
docker run –d --hostname my-rabbit --name some-rabbit –p 15672:15672 –p 5672:5672 rabbitmq:management
```

Et voilà. Notre container RabbitMQ est lancé et fonctionnel. Nous passerons dorénavant par l'interface graphique pour avoir accès aux données obtenues.

III. Fonctionnement

Il est possible une fois RabbitMQ lancé, de pouvoir envoyer et recevoir des messages. C'est bien ici que l'on doit mentionner l'avantage de l'OpenSource. En effet il est possible d'écrire soit même un programme de messagerie[9] pour envoyer et recevoir des messages avec RabbitMQ. Dans notre démonstration, nous le ferons en Python avec la librairie Pika[10].

[9] Dans les faits, c'est parce que RabbitMQ ne dispose pas d'une application dédiée.
[10] En effet il existe plusieurs librairies disponibles

Rabbit MQ fonctionne avec de multiples protocoles de communication. Il implémente AMQP[11] qui est à son tour libre, mais aussi les plugins STOMP et MQTT.

Il existe différents modes de fonctionnement

Expéditeur vers Destinataire

Dans ce mode, un message est stocké dans la file pour un et un seul destinataire. Nous utiliserons pour ce faire une file d'attente nommée.

Publication / Souscription

Pour illustrer ce modèle, nous construirons un système de journalisation simple. Il se composera de deux programmes : le premier émettre des messages de journal et le second les recevra. Ici chaque copie en cours d'exécution du programme recevra les messages. Essentiellement, les messages de journal publiés vont être diffusés à tous les récepteurs.

Routage

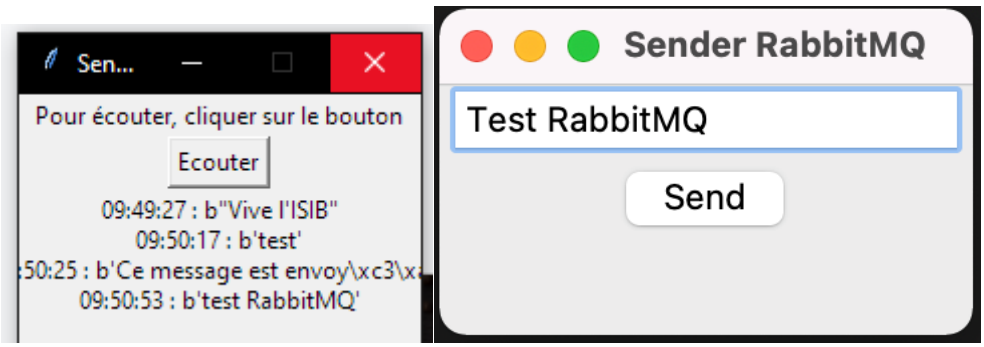
Il est similaire au mode précédent, avec la nuance que dans ce cas, on ne permet que de s'abonner à un sous-ensemble des messages. Par exemple, nous pouvons diriger uniquement des messages d'erreurs critiques vers le fichier journal (Pour économiser de l'espace disque), tout en étant en mesure d'imprimer tous les messages du journal sur la console.

Topics

Ce système est le plus connu du protocole MQTT. En fait le principe est d'avoir des publications et des souscriptions. On souscrit à un topic, et on publie sur un topic. Dans ce cas, le topic fonctionne comme un canal, et tous ceux qui écoutent sur un topic reçoivent les messages qui sont par la suite libérés pour faire de l'espace mémoire.

IV. Application

A qui se poserait la question de savoir en quoi les solutions OpenSource, et RabbitMQ dans notre cas sont utiles, nous répondront que la réponse se trouve dans gestion des données et le coût réduit des solutions. En voici l'exemple d'une petite implémentation basée sur MQTT qui peut permettre une communication dans un réseau.



[11] Advanced Message Queuing Protocol : Norme open source pour les systèmes de messagerie asynchrone par le réseau.

Il existe en effet d'autre moyens de configurer et d'utiliser RabbitMQ, mais cela irait bien au-delà du cadre de notre étude tant le champ d'application est vaste. Cependant nous retenons que les solutions OpenSource offrent des possibilités généralement gratuites et assez vastes que pour couvrir tous les domaines d'applications. La seule condition reste une implémentation correcte.

Références