Livrable 5
Rapport final

SAÉ 4.Deploi.01 – Mars 2025

Lilya Benkheira, Maëlys Deschaux-Beaume, Felix Martins, Kenzo Sechi, Manu Thuillier

IUT2 Université Grenoble Alpes
Département INFO

Table des matières

Table des figures

Table des *listings*

1 Introduction et mise en contexte

Ce projet vise à concevoir et déployer une infrastructure réseau sécurisée adaptée à la structure de la mairie de Voiron. Elle fournit plusieurs services réseaux (DNS, DHCP...) ainsi que des postes de travail pour des utilisateurs.

Dans le cadre de ce livrable 5, nous rappellerons les choix architecturaux précédents, tout en détaillant les évolutions apportées à l'infrastructure actuelle. Pour une compréhension approfondie du processus de déploiement, nous expliquerons les différentes étapes d'installation et de configuration mises en place. Nous aborderons également les tests effectués ainsi que les mesures de sécurité appliquées pour assurer la fiabilité et la protection du système.

2 Rappel de l'architecture et éventuelles évolutions

Notre architecture sépare les machines en quatre VLAN :

- Une DMZ qui contient tous les services qui doivent être accessibles depuis l'extérieur - le Wiki, le CAS et un DNS externe
- Un VLAN pour les postes de travail des utilisateurs
- Un VLAN pour les postes de travail des administrateurs
- Un VLAN pour les services internes comme la base de données ou le serveur IPAM

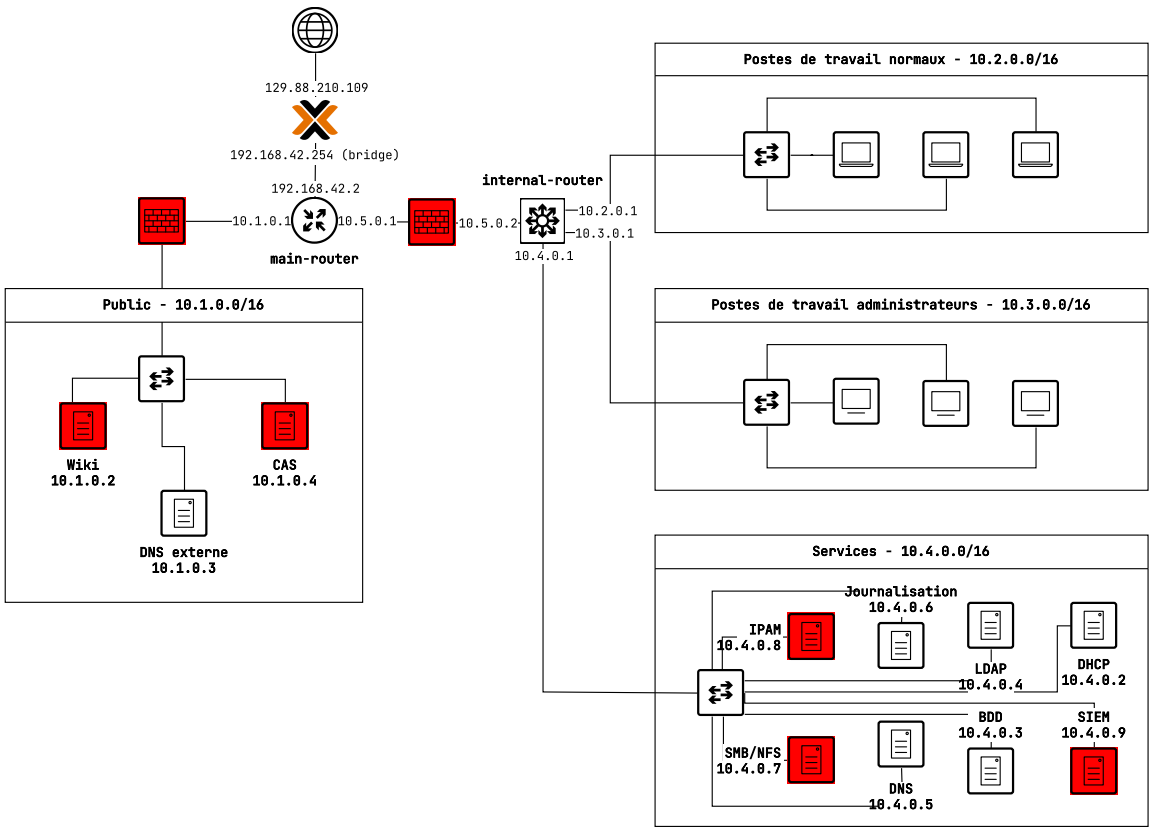

FIGURE 1 – Ancienne architecture, en rouge les éléments qui n'étaient pas encore réalisés.

En raison de contraintes de ressources, nous n'avons pas été en mesure de déployer le serveur SIEM avec Wazuh. Nous avons donc opté pour l'installation du serveur Kerberos à la place.

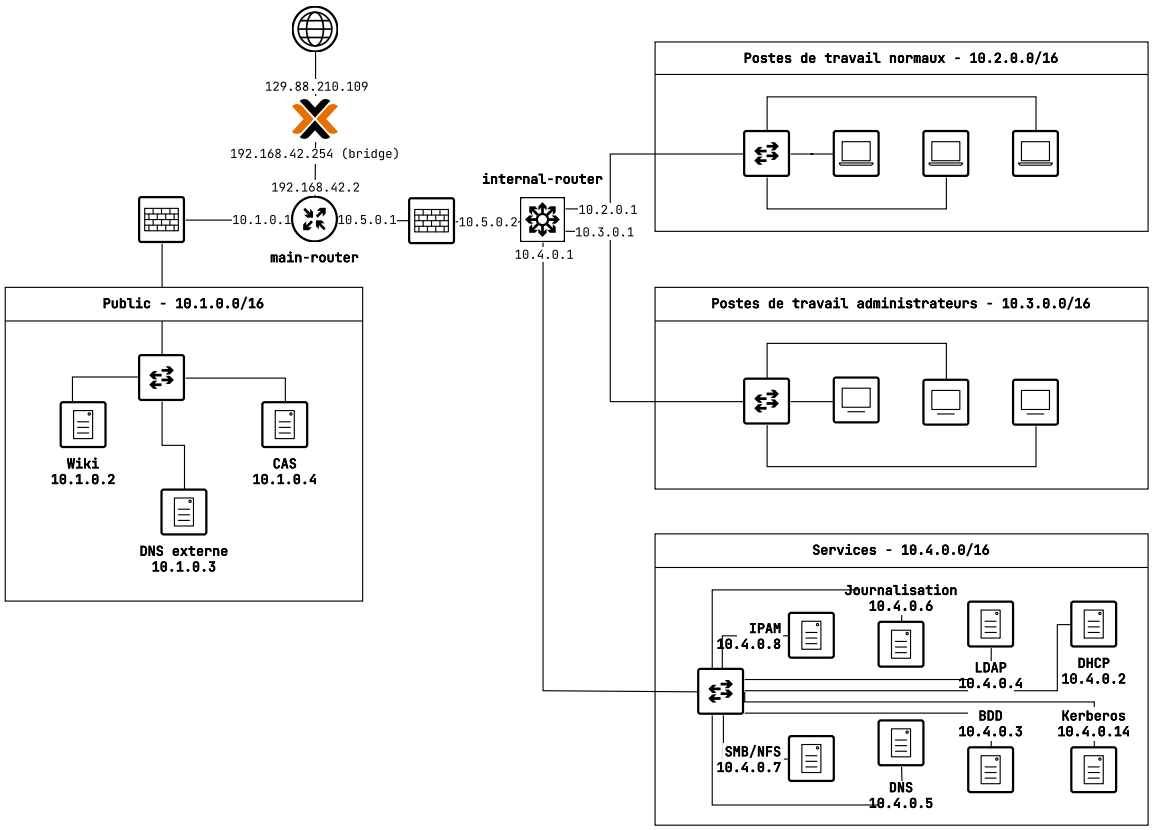

FIGURE 2 – Architecture actuelle

3 Récapitulatif des ressources matérielles utilisées

Pour le déploiement de notre infrastructure, nous disposons de 6 hyperviseurs, offrant un total de 128Go de stockage. Chaque hyperviseur, se voit attribuer 10Go de RAM et 4 CPU. L'ensemble comprend 16 machines, dont leur allocation en ressources dépend de leur rôle et de leur utilisation.

4 Documentation technique des différents réseaux et services

4.1 Configuration du réseau inter-nœuds

L'infrastructure repose sur trois réseaux VXLAN distincts pour segmenter le trafic en fonction des usages : `internal`, `public` et `dmz`. Ces réseaux sont configurés sur Proxmox (avec 1450 de MTU car VXLAN prend 50 octets sur une trame Ethernet) et interconnectent plusieurs nœuds, assurant une isolation et une sécurité adaptées aux différents types de communications.

4.1.1 VXLAN `public`

Le VXLAN `public` regroupe les services accessibles depuis l'extérieur. Il est défini sur un unique *vnet* :

— **10.1.0.0/16** : utilisé pour les services publics nécessitant une exposition externe.

4.1.2 VXLAN `internal`

Le VXLAN `internal` est dédié aux communications internes et se divise en trois *vnets* avec chacun un subnet :

- `users` avec **10.2.0.0/16** : réseau destiné aux postes utilisateurs.
- `admin` avec **10.3.0.0/16** : réseau réservé aux postes administrateurs.
- `services` **10.4.0.0/16** : réseau des services internes.

4.1.3 VXLAN `dmz`

Le VXLAN `dmz` est utilisé pour isoler les échanges entre le routeur principal et le réseau interne :

- **10.5.0.0/16** : segment réseau entre le routeur principal et le routeur interne, permettant un filtrage et une sécurisation du trafic.

4.2 Routeur principal

Le routeur est équipé de plusieurs interfaces réseau :

- **Interface bridge** `vmbr1` (`ens18`) – `192.168.42.0/24` : bridge vers l'extérieur.
- **Interface VNet** `public` (`ens19`) – `10.1.0.1/16` : réseau public.
- **Interface VLAN** `dmz` (`ens20`) – `10.5.0.1/16` : réseau intermédiaire entre le routeur principal et interne.

Le fichier `/etc/network/interfaces` du routeur principal est configuré comme suit :

```
source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto ens18
iface ens18 inet static
        address 192.168.42.2/24
        gateway 192.168.42.254

# Externe
auto ens19
iface ens19 inet static
        address 10.1.0.1/16

# DMZ
auto ens20
iface ens20 inet static
        address 10.5.0.1/16
        post-up ip route add 10.0.0.0/8 via 10.5.0.2 dev ens20
        pre-down ip route add 10.0.0.0/8 via 10.5.0.2 dev ens20
```

Listing 1 – Fichier `/etc/network/interfaces` pour le routeur principal

De plus, décommentez la ligne `net.ipv4.ip_forward=1` dans le fichier `/etc/sysctl.conf`. Ensuite, exécutez la commande `sysctl -p`.

4.3 Routeur interne

Le routeur interne a été configuré pour assurer la communication entre les différents VLAN internes et le VLAN `dmz`. Il est responsable du routage entre ces réseaux et de la gestion des

requêtes DHCP relayées vers le serveur principal.

Le fichier `/etc/network/interfaces` du routeur interne est configuré comme suit :

```
source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto ens18
iface ens18 inet static
       address 10.5.0.2/16
       gateway 10.5.0.1

# Reseau users
auto ens19
iface ens19 inet static
       address 10.2.0.1/16

# Reseau admin
auto ens20
iface ens20 inet static
       address 10.3.0.1/16

# Reseau services
auto ens21
iface ens21 inet static
       address 10.4.0.1/16
```

Listing 2 – Fichier `/etc/network/interfaces` pour le routeur interne

De plus, décommentez la ligne `net.ipv4.ip_forward=1` dans le fichier `/etc/sysctl.conf`. Ensuite, exécutez la commande `sysctl -p`.

Configuration du relais DHCP

Afin d'assurer l'attribution dynamique des adresses IP aux machines connectées aux différents VLANs, le paquet `dhcp-relay` a été configuré. Ce relais permet de transmettre les requêtes DHCP vers le serveur principal pour garantir une distribution centralisée des adresses IP.

Le fichier de configuration du relais DHCP est fourni ci-dessous :

- **INTERFACES** représente le nom des différentes interfaces des sous-réseaux devant avoir accès au relais.
- SERVERS spécifie l'IP du serveur DHCP.

```
# Defaults for isc-dhcp-relay initscript
# sourced by /etc/init.d/isc-dhcp-relay
# installed at /etc/default/isc-dhcp-relay by the maintainer scripts

#
# This is a POSIX shell fragment
#

# What servers should the DHCP relay forward requests to?
SERVERS="10.4.0.2"

# On what interfaces should the DHCP relay (dhrelay) serve DHCP requests?
INTERFACES="ens19 ens20 ens21"
```

```
# Additional options that are passed to the DHCP relay daemon?
OPTIONS=""
```

Listing 3 – Fichier `/etc/default/isc-dhcp-relay` du `internal-router`

4.4 DHCP

Le serveur DHCP est configuré avec Kea afin de gérer dynamiquement l'attribution des adresses IP aux différents équipements du réseau. Il prend en charge plusieurs sous-réseaux correspondant aux différents VLANs définis dans l'architecture.

4.4.1 Gestion des pools d'adresses

Des pools d'adresses sont définis pour chaque VLAN afin de permettre l'attribution dynamique d'IP aux machines qui en font la demande. Chaque réseau interne dispose de son propre pool, garantissant une séparation claire des espaces d'adressage :

- **10.2.0.0/16** : pool d'adresses pour les utilisateurs.
- **10.3.0.0/16** : pool d'adresses pour les administrateurs.
- **10.4.0.0/16** : pool d'adresses pour les services internes.

Chaque pool est configuré avec des paramètres tels que la durée de validité des baux et les délais de renouvellement, afin d'optimiser la gestion des ressources IP.

Tous nos services internes nécessitent une adresse IP fixe. Des réservations sont donc mises en place en associant des adresses spécifiques aux adresses MAC correspondantes aux machines en question.

La configuration de Kea se trouve également au *listing* 8 en annexe

4.5 Serveurs DNS

- Installer Bind9 avec la commande :

```
apt install bind9
```

4.5.1 Configuration du DNS externe

- Modifier `/etc/bind/named.conf.options` :
  - Activer la récursion
  - Définir un serveur DNS externe dans `forwarders`

```
options {
        directory "/var/cache/bind";
        recursion yes;
        allow-query { any; };
        forward only;

        forwarders {
                152.77.1.22; // Ici, notre serveur DNS externe
        };

        dnssec-validation auto;
};
```

Listing 4 – Fichier `/etc/bind/named.conf.options` pour le DNS externe

— Vérifier la configuration :

```
named-checkconf
```

— Redémarrer le service :

```
systemctl restart bind9
```

— Tester la résolution avec :

```
nslookup univ-grenoble-alpes.fr
dig univ-grenoble-alpes.fr
```

4.5.2 Configuration du DNS interne

— Modifier `/etc/bind/named.conf.options` :
  — Activer la récursion
  — Définir le DNS externe dans `forwarders`
  — Activer l'écoute via TLS et HTTPS en utilisant un certificat signé par notre autorité de certification

```
tls mycert {
    cert-file "/var/cache/bind/internal-dns.cert.pem";
    key-file "/var/cache/bind/internal-dns.key.pem";
};

options {
        directory "/var/cache/bind";

        // If there is a firewall between you and nameservers you want
        // to talk to, you may need to fix the firewall to allow multiple
        // ports to talk. See http://www.kb.cert.org/vuls/id/800113

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0s placeholder.
        recursion yes;
        allow-recursion { any; };
        forwarders { 10.1.0.3; };

        dnssec-validation auto;
        listen-on { any; };
        listen-on tls mycert { any; };
        listen-on tls mycert http default { any; };
        // listen-on-v6 { any; };
};
```

Listing 5 – Fichier `/etc/bind/named.conf.options` pour le DNS interne

— Déclarer la zone locale dans `/etc/bind/named.conf.local`
— Créer le fichier de zone `/etc/bind/db.local` contenant :

```
$TTL 86400
@ IN SOA local. admin.local. (
               2024031201 ; Serial
               3600 ; Refresh
               1800 ; Retry
               604800 ; Expire
               86400 ) ; Minimum TTL

; Serveur DNS
```

```
@ IN NS ns.local.

; Adresse du serveur DNS
ns IN A 10.4.0.5
ext-ns IN A 10.1.0.3

; éRsolution noms des services

ipam IN A 10.4.0.13
journalisation IN A 10.4.0.6
ldap IN A 10.4.0.4
krb IN A 10.4.0.14
fileserver IN A 10.4.0.7
db IN A 10.4.0.3
dhcp IN A 10.4.0.2
zabbix IN A 10.4.0.12
```

Listing 6 – Fichier `/etc/bind/db.local` pour le DNS interne

— Inclure `named.conf.local` dans `named.conf` :

```
include "/etc/bind/named.conf.local";
```

— Vérifier la configuration :

```
named-checkzone local /etc/bind/db.local
```

— Redémarrer le service :

```
systemctl restart bind9
```

— Tester la résolution avec :

```
dig +trace ns.local
dig +trace univ-grenoble-alpes.fr
```

4.6 Serveur Wiki

4.6.1 Prérequis système

Avant d'installer `MediaWiki`, il faut s'assurer que le serveur dispose des composants suivants :

— Serveur web : `Apache` ou `Nginx`
— `PHP 8.1.0` ou supérieure
— Base de données :
  — MariaDB 10.3.0+ ou MySQL 5.7.0+
  — PostgreSQL 10.0+
  — SQLite 3.8.0+

4.6.2 Téléchargement de MediaWiki

— Installer les dépendances et les modules requis :

```
sudo apt-get install apache2 mariadb-server php php-intl php-mbstring
    php-apcu php-curl php-mysql
```

— Télécharger le fichier `tar.gz` sur la page officielle : https://www.mediawiki.org/wiki/Download
— Décompresser l'archive téléchargée avec la commande :

```
tar -xzvf mediawiki-*.tar.gz
```

— Déplacer le dossier extrait dans le répertoire web du serveur (ici Apache).

```
# Directory
/var/www/html/wiki
```

4.6.3 Configuration de la base de données avec MariaDB

— Pour stocker les données du Wiki et gérer l'accès de façon sécurisé, créer une base de données, un utilisateur et attribuer des permissions.

```
CREATE DATABASE my_wiki;
CREATE USER 'wikiuser'@'localhost' IDENTIFIED BY 'database_password';
GRANT ALL PRIVILEGES ON my_wiki.* TO 'wikiuser'@'localhost' WITH GRANT
    OPTION;
```

Listing 7 – Requête SQL : Création de BDD et d'utilisateur et attribution de permissions

4.6.4 Installation de MediaWiki

— Accéder à l'URL pour lancer le script d'installation de MediaWiki : `http://localhost/`
— Suivre les instructions pour connecter le wiki à la base de données, créer le compte administrateur et configurer d'autres paramètres.
— Le fichier `LocalSettings.php` est généré à la fin de l'installation. Ce fichier contient toutes les configurations de l'installation de MediaWiki. Il doit être placé dans `/var/www/html/wiki`.
— Une fois l'installation terminée et `LocalSettings.php` configuré, le serveur MediaWiki est accessible depuis `http://wiki.voiron.fr/`

4.7 Serveur journalisation

Notre serveur de journalisation permet de centraliser et de collecter les journaux provenant de plusieurs machines distantes. Ces journaux contiennent des informations essentielles pour la gestion, la surveillance, et le dépannage des systèmes Linux.

La configuration de `systemd-journal-remote` du serveur et du client `systemd-journal-upload` sont disponibles dans le livrable n°4.

4.8 Serveur LDAP

Nous mettons en place un serveur implémentant le protocole LDAP, permettant l'accès à un annuaire informatique. Cet annuaire est utile afin de stocker des données relatives aux personnes ou aux machines, par exemple un couple login/mot de passe dans le cas d'authentification d'utilisateurs sur des postes anonymes. Nous utilisons `OpenLDAP`, qui permet l'interrogation d'une base de données à travers un daemon. Cette installation est reproductible à l'aide de scripts fournis dans le livrable n°4.

Nous relions ce service au PAM des stations utilisateurs afin de permettre une connexion sur n'importe quel poste avec n'importe quel login, tout en centralisant les données et en évitant donc la duplication. Nous fournissons un script permettant d'installer le client LDAP et de faire les configurations nécessaires, ainsi qu'un script permettant d'ajouter un utilisateur en choisissant son login, mot de passe, UID...

4.9 Serveur NFS

Nous mettons en place un serveur de stockage de fichier implémentant le protocole NFS. Ce serveur permet le stockage centralisé des données personnelles des utilisateurs, en particulier de leur homedir. Nous mettons à disposition un script serveur, permettant de répliquer cette installation, ainsi qu'un script client, automatisant le montage d'un disque du serveur en

`/home/` avec le fichier `/etc/fstab`.

La configuration du serveur NFS et clients NFS sont précisés dans les *listings 10* et *11* du livrable n°4.

4.10 Serveur Kerberos

Nous mettons en place un serveur Kerberos, permettant de paramétrer le déploiement de clés cryptographiques. Ce serveur utilise un backend LDAP afin de gérer la création et la distribution de clés à des agents identifiés sur le réseau. L'objectif est de sécuriser d'autres protocoles, comme NFS. Nous mettons à disposition un script d'installation du serveur Kerberos, un script à exécuter sur le serveur LDAP permettant de configurer les utilisateurs et permissions appropriées ainsi qu'un script permettant de rajouter les attributs Kerberos à un utilisateur enregistré dans la base LDAP.

La mise en place de Kerberos est expliquée dans le livrable n°4.

4.11 Serveur IPAM

NetBox est une application web open-source de modélisation des ressources d'infrastructure, développée avec le framework Django en Python. Elle sert à la gestion des centres de données et des réseaux.

Prérequis

Avant de commencer l'installation, assurez-vous que votre système Debian est à jour et que vous disposez des privilèges administratifs.

Installation des dépendances système

NetBox nécessite Python 3.10 ou une version supérieure. Commencez par mettre à jour les dépôts et installer les paquets nécessaires :

```
apt update
apt install -y python3 python3-pip python3-venv python3-dev build-essential
    libxml2-dev libxslt1-dev libffi-dev libpq-dev libssl-dev zlib1g-dev git
```

Vérifiez que votre version de Python est compatible :

```
python3 --version
```

Installation de PostgreSQL

NetBox utilise PostgreSQL comme base de données. Installez-le et configurez une base de données dédiée :

```
apt install -y postgresql
```

Connectez-vous à PostgreSQL pour créer la base de données et l'utilisateur :

```
-u postgres psql
```

Dans le shell PostgreSQL, exécutez :

```
CREATE DATABASE netbox;
CREATE USER netbox WITH PASSWORD 'votre_mot_de_passe';
ALTER ROLE netbox SET client_encoding TO 'utf8';
ALTER ROLE netbox SET default_transaction_isolation TO 'read committed';
ALTER ROLE netbox SET timezone TO 'UTC';
GRANT ALL PRIVILEGES ON DATABASE netbox TO netbox;
\q
```

Installation de Redis

Redis est utilisé pour la mise en cache et la gestion des tâches en arrière-plan. Installez-le et assurez-vous qu'il fonctionne :

```
apt install -y redis-server
systemctl enable --now redis-server
```

Vérifiez que Redis est opérationnel :

```
redis-cli ping
```

Une réponse `PONG` confirme que Redis fonctionne.

Téléchargement et installation de NetBox

Clonez le dépôt Git de NetBox et installez les dépendances Python :

```
mkdir -p /opt/netbox
cd /opt/netbox
git clone -b master https://github.com/netbox-community/netbox.git .
pip3 install -r /opt/netbox/requirements.txt
```

Configuration de NetBox

Créez un utilisateur système pour exécuter NetBox :

```
adduser --system --group --disabled-login netbox
```

Assurez-vous que l'utilisateur `netbox` a les droits appropriés sur les répertoires nécessaires :

```
chown -R netbox:netbox /opt/netbox/netbox/media /opt/netbox/netbox/reports \
       /opt/netbox/netbox/scripts
```

Copiez le fichier de configuration exemple et modifiez-le :

```
cd /opt/netbox/netbox/netbox
cp configuration_example.py configuration.py
nano configuration.py
```

Dans `configuration.py`, ajustez les paramètres suivants :
- **ALLOWED_HOSTS** : Liste des hôtes autorisés.

```
  ALLOWED_HOSTS = ['ipam.local']
```

- **DATABASE** : Paramètres de connexion à PostgreSQL.

```
  DATABASE = {
      'NAME': 'netbox',
      'USER': 'netbox',
      'PASSWORD': 'votre_mot_de_passe',
      'HOST': 'localhost',
      'PORT': '',
      'CONN_MAX_AGE': 300,
  }
```

- **REDIS** : Paramètres de connexion à Redis.

```
REDIS = {
    'tasks': {
        'HOST': 'localhost',
        'PORT': 6379,
        'PASSWORD': '',
        'DATABASE': 0,
        'SSL': False,
    },
    'caching': {
        'HOST': 'localhost',
        'PORT': 6379,
        'PASSWORD': '',
        'DATABASE': 1,
        'SSL': False,
    }
}
```

- **SECRET_KEY** : Clé secrète pour les opérations cryptographiques. Générez-en une avec :

```
python3 ../generate_secret_key.py
```

Copiez la clé générée et remplacez la valeur de `SECRET_KEY` dans le fichier `configuration.py`.

Finalisation de l'installation

Appliquez les migrations de la base de données et créez un superutilisateur :

```
python3 manage.py migrate
python3 manage.py createsuperuser
```

Installation du connecteur entre Netbox et Kea

Le connecteur `netbox-kea-dhcp` permet de synchroniser les données de NetBox avec le serveur DHCP ISC Kea. Il exporte les préfixes, les plages d'IP et les adresses IP de NetBox vers les sous-réseaux, les pools et les réservations de Kea DHCP. Le connecteur écoute les événements Webhook de NetBox et met à jour la configuration de Kea en conséquence.

Installation avec pip

Pour installer `netbox-kea-dhcp` avec `pip`, exécutez la commande suivante :

```
pip install netbox-kea-dhcp
```

Configuration des Webhooks NetBox

Configurez des webhooks dans NetBox pour notifier les événements liés aux objets DHCP. Voici les configurations recommandées :

- **Webhook 1** :
  - **Types de contenu** : `IPAM > Prefix`, `IPAM > IP Range`, `IPAM > IP Address`, `DCIM > Device`, `DCIM > Interface`, `Virtualization > Virtual Machine`, `Virtualization > Interface`.
  - **Événements** : `Updates`.
  - **Conditions** : Aucune.
- **Webhook 2** :
  - **Types de contenu** : `IPAM > IP Address`.
  - **Événements** : `Creations`, `Deletions`.

— **Conditions** : `{ "and": [ { "attr": "status.value", "value": "dhcp" } ] }`.
— **Webhook 3** :
— **Types de contenu** : `IPAM > IP Range`.
— **Événements** : `Creations`, `Deletions`.
— **Conditions** : `{ "and": [ { "attr": "status.value", "value": "dhcp" } ] }`.
— **Webhook 4** :
— **Types de contenu** : `IPAM > Prefix`.
— **Événements** : `Creations`, `Deletions`.
— **Conditions** : Aucune.

Pour plus de détails, consultez la section *Recommended Netbox webhooks* de la documentation officielle.

Exécution du Connecteur

Après avoir configuré les webhooks, lancez le connecteur avec la commande suivante :

```
netbox-kea-dhcp --netbox-url https://ipam.local \
    --netbox-token votre_token_netbox \
    --kea-url http://dhcp.local \
    --sync-now \
    --listen \
    -v
```

Cette commande effectue une synchronisation complète au démarrage et écoute les événements de NetBox pour des mises à jour en temps réel.
Pour des informations détaillées sur l'installation, la configuration et les options avancées, référez-vous à la documentation officielle du projet sur GitHub.

4.12 Serveur Zabbix

4.12.1 Installation de Zabbix

— Installation du dépôt

```
wget
    https://repo.zabbix.com/zabbix/7.2/release/debian/pool/main/z/zabbix-release
    zabbix-release_latest_7.2+debian12_all.deb
dpkg -i zabbix-release_latest_7.2+debian12_all.deb
apt update
```

— Installer Zabbix `server, frontend`

```
apt install zabbix-server-mysql zabbix-frontend-php zabbix-apache-conf
    zabbix-sql-scripts
```

Pour installer Zabbix `agent`, nous avons réalisé un script dans le *listing* 13 du livrable n°4.

4.12.2 Configuration de Zabbix

— Création de la base de données initiale

```
# mysql -uroot -p -h db.local
password
mysql> create database zabbix character set utf8mb4 collate utf8mb4_bin;
mysql> create user zabbix@localhost identified by 'password';
mysql> grant all privileges on zabbix.* to zabbix@localhost;
```

```
mysql> set global log_bin_trust_function_creators = 1;
mysql> quit;
```

— Importer le schéma et les données de Zabbix

```
zcat /usr/share/zabbix/sql-scripts/mysql/server.sql.gz | mysql
    --default-character-set=utf8mb4 -uzabbix -p zabbix -h db.local
```

— Désactiver l'option `log_bin_trust_function_creators`

```
mysql -uroot -p -h db.local
mysql> set global log_bin_trust_function_creators = 0;
mysql> quit;
```

— Modifier le fichier de configuration `/etc/zabbix/zabbix_server.conf` pour ajouter le mot de passe de la base de données et l'hôte

```
DBPassword=password
DBHost=db.local
```

— Éditer le fichier `/etc/zabbix/nginx.conf`, et décommenter `listen` et `server_name` :

```
# listen 80;
# server_name zabbix.local;
```

— Démarrer les services Zabbix et Nginx

```
systemctl restart zabbix-server zabbix-agent nginx php8.2-fpm
systemctl enable zabbix-server zabbix-agent nginx php8.2-fpm
```

— L'interface web de Zabbix est accessible à l'adresse suivante :

```
https://zabbix.local
```

5 CAS

Dans le cadre de l'authentification centralisée, nous avons choisi de mettre en place le serveur CAS d'Apereo. Voici les étapes suivies pour l'installation et la configuration du serveur.

Utilisation du CAS Initializr

Génération du projet

Nous avons utilisé le CAS Initializr pour générer un projet CAS avec les dépendances nécessaires, notamment l'authentification LDAP pour l'intégration avec notre annuaire.

Installation de Java 21

Téléchargement et installation

Nous avons installé Java 21 en téléchargeant l'archive Debian depuis le site d'Oracle, car cette version est requise par le serveur CAS.

Compilation du projet

Compilation du projet avec Gradle

Après configuration, nous avons compilé le projet avec la commande suivante :

```
./gradlew build
```

Cela a généré le fichier '.war' du serveur CAS.

Configuration du serveur CAS

Fichier de configuration

Nous avons créé le répertoire /etc/cas/config et y avons placé le fichier de configuration cas.properties. Voici un extrait des configurations principales :

```
cas.webflow.crypto.signing.key=X
cas.webflow.crypto.encryption.key=Y
cas.tgc.crypto.signing.key=Z
cas.tgc.crypto.encryption.key=T
cas.webflow.crypto.encryption.key-size=128
cas.server.name=https://cas.voiron.fr
cas.authn.ldap[0].base-dn=dc=local
cas.authn.ldap[0].bind-dn=cn=admin,dc=local
cas.authn.ldap[0].bind-credential=ldap
cas.authn.ldap[0].ldap-url=ldap://10.4.0.4
cas.authn.ldap[0].search-filter=cn={user}
cas.locale.default-value=fr
server.port=443
cas.authn.accept.enabled=false
otel.metrics.exporter=none
cas.service-registry.json.location=file:/etc/cas/config/services
cas.service-registry.core.init-from-json=true
```

Les valeurs `X`, `Y`, `Z` et `T` sont à remplacer par les valeurs données lors du premier démarrage du CAS.

Création du service systemd

Configuration du service

Nous avons créé un fichier de service systemd pour gérer le démarrage du serveur CAS. Voici le contenu du fichier :

```
[Unit]
Description=CAS Server
After=network.target

[Service]
User=cas
Group=cas
ExecStart=/usr/bin/java -Dotel.metrics.exporter=none -jar /opt/cas/cas.war
Restart=always
SuccessExitStatus=143
WorkingDirectory=/opt/cas
StandardOutput=journal
StandardError=journal
AmbientCapabilities=CAP_NET_BIND_SERVICE

[Install]
WantedBy=multi-user.target
```

Démarrage du service

Après avoir configuré le service, nous avons exécuté les commandes suivantes pour démarrer le serveur CAS :

```
systemctl daemon-reload
systemctl start cas
```

Configuration des services

Ajout des services

Dans le dossier /etc/cas/config/services, nous avons créé un sous-dossier `services` et ajouté un fichier pour chaque service nécessitant une authentification via CAS. Par exemple, pour un service générique souhaitant s'enregistrer avec CAS, nous avons utilisé la configuration suivante :

```
{
  /*
    Generic service definition that applies to https/imaps urls
    that wish to register with CAS for authentication.
  */
  "@class" : "org.apereo.cas.services.CasRegisteredService",
  "serviceId" : "^(https|imaps)://.*",
  "name" : "wiki",
  # "description": "This is commented out"
  "id" : 10000001
}
```

Chaque service est enregistré avec un identifiant unique et un motif serviceId correspondant à l'URL du service. Cette configuration permet à CAS d'authentifier les utilisateurs pour ces services spécifiques.

6 Tests sur l'infrastructure

Nous fournissons une suite de tests permettant de tester différents aspects de notre infrastructure. Les programmes sont écrits en Python et sont modulables à l'aide de fichiers `CSV` ; les données renseignées montrent des exemples de test qui peuvent être conduits. La suite permet des tests de connectivité (ou non-connectivité pour vérifier l'efficacité d'un pare-feu) et de disponibilité de plusieurs services (LDAP, DNS, DHCP...). La suite est à déployer sur l'hyperviseur. Elle utilise les clés du ssh-agent (ou équivalent) de l'hyperviseur pour réaliser une connexion SSH aux hôtes du réseau et faire des tests à distance.

7 Aspects liés à la sécurité

7.1 Analyse de la résistance de l'infrastructure

L'infrastructure a été conçue avec des mécanismes de sécurité visant à limiter les risques d'intrusion :

- **Segmentation réseau par VLANs** : les différents services sont isolés dans des VLANs distincts, limitant les communications aux seuls échanges nécessaires.
- **Filtrage des ports** : seuls les ports strictement nécessaires sont ouverts (par exemple, 80 et 443 pour le web, 22 pour l'administration via SSH, et il peut être modifié pour certains services critiques).
- **Accès SSH sécurisé** : l'authentification se fait uniquement par clé SSH. Les connexions par mot de passe sont désactivées, et seules certaines adresses IP peuvent accéder au service SSH.

7.2 Veille sur les failles de sécurité et processus de correction

— **Veille régulière** : l'équipe technique consulte les bulletins de sécurité (CVE, listes de diffusion Debian, Proxmox, etc.) pour se tenir informée des vulnérabilités récentes.
— **Mises à jour système fréquentes** : les serveurs sont mis à jour via `apt` au moins une fois par semaine.
— **Application immédiate des patchs critiques** : lorsqu'une faille critique est identifiée, un correctif est appliqué dans les plus brefs délais.

7.3 Bonnes pratiques sur les postes d'administration

— **Sécurité physique et logique** : les postes d'administration sont réservés à l'équipe technique et protégés par des mots de passe robustes.
— **Systèmes à jour** : les postes utilisent des distributions Debian maintenues à jour.
— **Utilisation encadrée** : les postes sont dédiés à un usage professionnel. La navigation personnelle y est proscrite.
— **Gestion des mots de passe** : aucun mot de passe n'est stocké en clair. Un gestionnaire de mots de passe est utilisé (par exemple KeePassXC).

8 Conclusion

Nous obtenons à l'issue de ce projet une infrastructure fonctionnelle opérationnalisant les services suivants :

— Serveurs DNS récursif et autoritaire
— Serveurs DHCP et IPAM pour la gestion des IP
— Serveur LDAP et synchronisation avec les postes utilisateur
— Serveur NFS et synchronisation avec les homedir des postes
— Serveur de centralisation de la journalisation
— Serveur CAS, utilisant le serveur LDAP en backend
— Serveur Web, hébergeant un Wiki et intégré avec le CAS
— Serveur de distribution de clefs Kerberos, avec le serveur LDAP en backend
— Monitoring du réseau avec Zabbix

L'infrastructure intègre des éléments de sécurité ; les hôtes sont répartis dans plusieurs VLAN isolés par des pare-feux. Les services sont mis en oeuvre à travers des protocoles sécurisés - SSL, clefs Kerberos...

Nous fournissons l'infrastructure avec des scripts d'installation, permettant de reproduire l'ensemble des configurations sur un site tiers, ainsi que des procédures de tests permettant de vérifier plusieurs aspects de l'infrastructure.

8.1 Retour d'expérience

Ce projet nous a permis d'opérationnaliser plusieurs services utiles et récurrents dans les infrastructures informatiques. Nous avons pu réinvestir les acquis théoriques de R4.B.10 et R4.B.12 afin de choisir les bons logiciels et protocoles pendant la phase de lancement de projet. Les connaissances de R4.B.11 nous ont permis de mettre en oeuvre l'infrastructure au niveau réseau et transport, avec la configuration des routeurs et pare-feux virtuels.

8.2 Pistes d'amélioration

Nous avons réussi à implémenter une grande partie des services de l'infrastructure améliorée. Cependant, certains services n'implémentent pas les meilleurs protocoles. Par exemple, le serveur DNS n'utilise pas SSL, et le serveur NFS n'utilise pas Kerberos. Ces failles sont des points d'attention puisqu'elle représentent des vulnérabilités connues du système.

Annexes

```
{
"Dhcp4":{
   "interfaces-config":{
      "interfaces":[
         "ens18" // services
      ]
   },
   "control-socket":{
      "socket-type":"unix",
      "socket-name":"/run/kea/kea4-ctrl-socket"
   },
   [...]
   "option-data":[
      {
         "name":"domain-name-servers",
         "data":"10.4.0.5" // DNS interne
      },
      {
         "name":"interface-mtu",
         "data":"1450" // Necessaire car le VXLAN prend 50 bits
      }
   ],
   "subnet4":[
      {
        // users
         "subnet":"10.2.0.0/16",
         "pools":[
            {
               "pool":"10.2.0.2 - 10.2.254.254"
            }
         ],
         "option-data":[
            {
               "name":"routers",
               "data":"10.2.0.1"
            }
         ]
      },
      {
        // admin
         "subnet":"10.3.0.0/16",
         "pools":[
            {
               "pool":"10.3.0.2 - 10.3.254.254"
            }
         ],
         "option-data":[
            {
               "name":"routers",
```

```
                "data":"10.3.0.1"
            }
        ]
      },
      {
        // services
        "subnet":"10.4.0.0/16",
        "pools":[
            {
                "pool":"10.4.0.3 - 10.4.254.254"
            }
        ],
        "reservations":[
            {
                "hw-address":"BC:24:11:51:CC:2A",
                "ip-address":"10.4.0.3",
                "hostname":"postgres"
            },
            // [...] pour chacun des services necessitant des IP fixes
        ],
        "option-data":[
            {
                "name":"routers",
                "data":"10.4.0.1"
            }
        ]
      }
    ]
}
}
```

Listing 8 – Configuration (partielle) de Kea