SAÉ S4.Deploi.01 : Documentation technique de Vortex Network

Nolan Dupont, Nils Rayot, Kylian Metayer, Jess Rousselard, Flavien Riondet

4 avril 2025

Table des matières

1 Introduction **3**

2 Architecture du Réseau **3**
2.1 Utilisation Proxmox . . . 3
2.2 Configuration du SDN (Software-Defined Network) . . . 3
2.2.1 Zone . . . 3
2.2.2 Vnets . . . 4
2.2.3 Subnet . . . 5
2.2.4 IPAM . . . 5
2.3 Inventaire des Équipements . . . 5
2.4 Schéma du réseau . . . 5

3 Structure des serveurs **7**
3.1 Le fonctionnement général du réseau . . . 7
3.2 Routeur NAT . . . 7
3.2.1 Configuration du routeur . . . 7
3.2.2 Pare-feu . . . 7
3.2.3 Configuration du routeur . . . 7
3.2.4 Pare-feu . . . 8
3.3 Les serveurs DHCP . . . 8
3.4 Les serveurs DNS . . . 8

4 Test de connectivité **8**
4.1 Script de test local . . . 9
4.1.1 Test de la passerelle par défaut . . . 9
4.1.2 Test de résolution DNS . . . 9
4.1.3 Test de connectivité externe (ping) . . . 9
4.1.4 Test de ping vers une machine distante . . . 9
4.1.5 Test de routage (Traceroute) . . . 9
4.2 Mode client (test local) . . . 10

4.3 Mode distant (test d'une machine externe) 10

5 Script d'installation de serveur 11

6 Utilisation de Matrix pour les communication 11
6.1 Organisation 11
6.2 Fonctionnement 11
6.3 Utilisabilité 11
6.4 Sécurité 12

Liste des tableaux

1 Tableau configuration du Vnets 4
2 Tableau configuration du Vnets 5
3 Inventaire des Serveurs 5

Table des figures

1 Configuration VXLAN 4
2 Réseau représenté dans un graphe 6
3 Test distant de passerelle par défaut et DNS 10
4 Test d'accesibilité d'une machine 11

1 Introduction

Ce document vise à fournir une vue d'ensemble technique de notre réseau informatique, virtualisé sur Proxmox usant d'hyperviseur sous KVM. Il décrit l'architecture du réseau. Il est documenté les scripts utilisés, l'infrastructure réseau (Plan d'adressage, nom des machines, Zone Proxmox,...), Infrastructure Système (OS utilisé, Distribution, Logiciel,...). Enfin les moyens de communication utilisée.

2 Architecture du Réseau

2.1 Utilisation Proxmox

La supervision du réseau et le paramétrage hardware des machines virtuelles est configuré avec le GUI de Proxmox, elle est composé d'un cluster de 5 hyperviseurs (appelé nœud) de 20 GO d'espace de stockage chaqu'un. Nous avons dispersé les machines virtuelles sur différents hyperviseurs pour obtenir le maximum d'espace mémoire soit 100 G, le SDN est configuré en conséquence pour que les machines puissent communiquer ensemble en étant sur différents superviseurs, mais dans le même réseau. Nous utilisons la version 8.2.7 de Proxmox, la dernière version n'étant par supporter par le datacenter actuellement et pouvant occasionner une perte du cluster et de ces hyperviseurs suite à une mise à jour de SDN avec le bouton 'Apply'.

2.2 Configuration du SDN (Software-Defined Network)

Pour superviser un réseau, Proxmox utilise une approche SDN qui consiste à administrer un réseau et à gérer les services de réseau par abstraction de fonctionnalités. Dans notre cas, Proxmox est composé de 4 fonctionnalités sous la version 8.2.7.

- Zones - permet de définir virtuellement des réseau séparer restreint donc isolé le réseau suivant 5 types de zones : Simple, VLAN, QinQ, VXLAN, EVPN
- VNets - permet de configurer les interfaces de connexion à la zone utilisée par les machines virtuelle sur leur carte réseau
- Subnet - permet de configurer la plage d'adressage IP du réseau sur une interface
- IPAM (IP adresse Management) - manager d'adressage des machines virtuelles par rapport à leur adresse MAC cela permet de gérer automatiquement le DHCP par l'hyperviseur, dans notre cas inutile, car ne fonctionne que sur de la zone Simple

2.2.1 Zone

Les différentes machines virtuelles étant hypervisé dans différents hyperviseurs à la contrainte d'espace de stockage limité. Nous avons dû passer par des zones en type VXLAN dont ils sont possibles d'offrir les IP des hyperviseur pour les connecter entre eux. Finalement permettre la communication entre deux machines virtuelles dans deux hyperviseurs différents. (Voir section Test de connectivité)

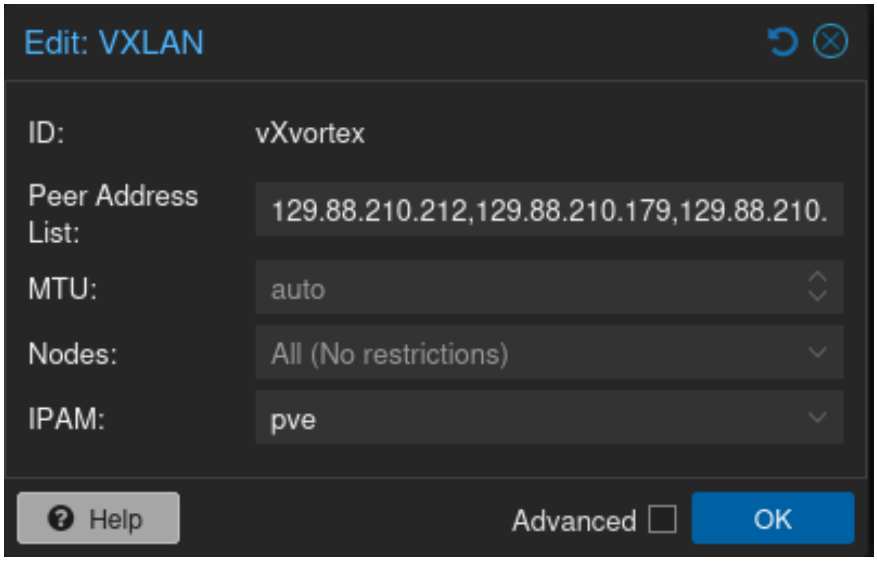

FIGURE 1 – Configuration VXLAN

Comme écrit, le nom du VXLAN est vXvortex référence au VXLAN et au nom de l'entreprise Vortex Network. Les Peer Adresse listent qui accueil les IP des hyperviseur interconnecté est écrit sous là forme, "[IP],[IP],[IP],[IP],[IP]".

Nous avons aussi une zone Simple qui permet de connecter un routeur NAT lui-même connecté au VXLAN pour connecter les machines des réseaux à Internet. La zone Simple, il s'agit du plugin le plus simple. Il crée un pont Vnet isolé. Ce pont n'est pas lié à une interface physique, et le trafic VM est uniquement local sur chaque nœud. Il peut être utilisé dans des configurations NAT ou routé, pour connecter d'autres réseaux à Internet passant par le nœud propriétaire de la VM routeur NAT.

2.2.2 Vnets

Les Vnets sont les interfaces qui permettent de connecter les zones aux interfaces nous possédons 5 interface suivant l'adressage suivant.

ID Interface	**Alias**	**Tag**	**zone**
vnet10	Poste	10	vXvortex
vnet1	DataCenter	1	vXvortex
vnet2	Admin	0	vXvortex
vnet20	DMZ	20	vXvortex
vnet3	routeurs	AUCUN	private

TABLE 1 – Tableau configuration du Vnets

2.2.3 Subnet

Les sous-réseaux permettent de configurer la plage d'adressage des Vnet, nous avons un subnet par Vnet pour plus de simplicité dans la configuration.

Subnet	Gateway	SNAT	DHCP Range	Interface
192.168.10.0/24	192.168.10.1	1	192.168.10.2 - 192.168.10.254	vnet10
192.168.1.0/24	192.168.1.1	0	192.168.1.2 - 192.168.1.254	vnet1
192.168.2.0/24	192.168.2.1	0	192.168.2.2 - 192.168.2.254	vnet2
192.168.0.0/24	192.168.0.1	0	192.168.0.2 - 192.168.0.254	vnet20
192.168.3.0/24	192.168.3.1	0	192.168.3.2 - 192.168.3.254	vnet3

TABLE 2 – Tableau configuration du Vnets

2.2.4 IPAM

Nous utilisons le système d'IPAM pour la zone simple détenant les routeurs. nous avons la zone private et l'interface Vnet0 avec l'adresse MAC des VM et l'IP assigné pour les machines virtuelle des routeurs.

2.3 Inventaire des Équipements

Nom	Catégorie	@IP	Services hébergés	Rôle
DHCP principal	DHCP	192.168.1.10/24	KEA	Fournir des @IP
DHCP DMZ	DHCP	192.168.0.4/24	KEA	Fournir des @IP
DNS principal	DNS	192.068.1.20/24	Bind9	Résolution de noms
DNS employés	DNS	192.068.10.2/24	Bind9	Résolution de noms
DNS administrateurs	DNS	192.068.2.2/24	Bind9	Résolution de noms
DNS DMZ	DNS	192.068.0.2/24	Bind9	Résolution de noms
Wiki	Web	192.168.1.30/24	Apache	Hébergement Web
Web DMZ	Web	192.168.0.3/24	Apache	Hébergement Web
SGBD	Base de données	192.168.1.40/24	PostgreSQL	Stocker des données
Fichier	SMB/CIFS	192.168.1.50/24	Samba	Stocker des fichiers
Log	CLI	19.068.1.60/24	Systemd-journald	Avoir un suivi des acti
CAS	Web	192.168.1.70/24	LDAPS	Authentification
IPAM	CLI	192.168.1.80/24	Apache	Hébergement Web

TABLE 3 – Inventaire des Serveurs

2.4 Schéma du réseau

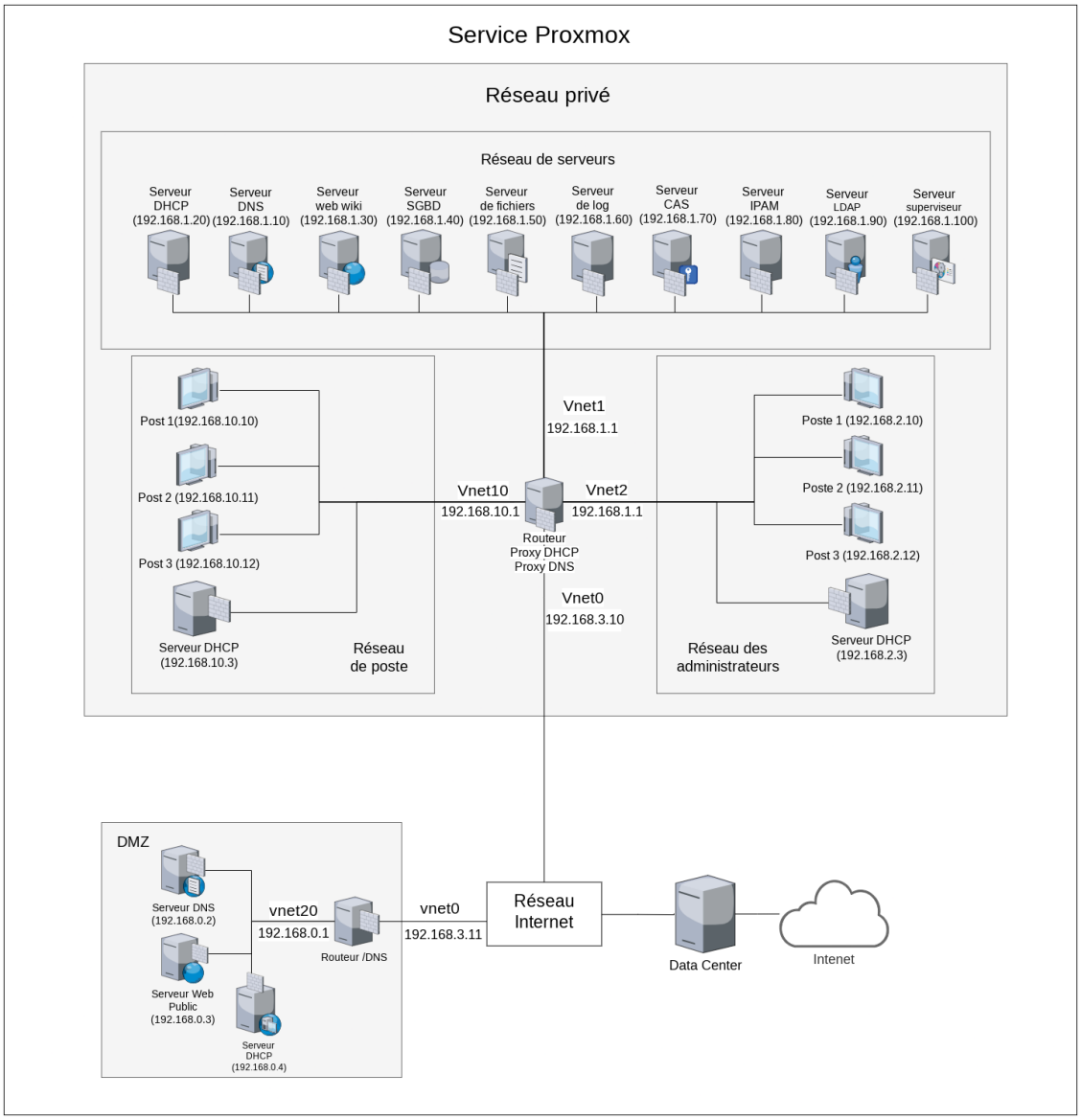

FIGURE 2 – Réseau représenté dans un graphe

3 Structure des serveurs

3.1 Le fonctionnement général du réseau

Nous avons créé un réseau qui est partagé en 4 sous-réseaux. Un pour le personnel qui travail dans l'entreprise, un pour les administrateurs, un pour les différents serveurs et un dernier pour la DMZ qui contient juste un serveur web. Nous avons deux routeurs, un pour les sous-réseaux principaux et un pour la DMZ. Sur le réseau principal, le routeur va servir de proxy pour le DHCP (voir 3.3) et le DNS (voir 3.4).

3.2 Routeur NAT

3.2.1 Configuration du routeur

Nous avons créé un routeur servant d'intersection entre nos réseaux de machines virtuelles et un de nos hyperviseurs. Le routeur est configuré de telle sorte à ce que les paquets soient redirigée sur le sous-réseau visé, et dans le cas où l'IP de destination est inconnue, les paquets sont envoyés à l'hyperviseur.

Cette configuration permet aux sous-réseaux de machines virtuelles de communiquer entre eux, mais aussi d'accéder à Internet.

3.2.2 Pare-feu

Nous avons créé un fichier de configuration firewall sur le routeur. Nous utilisons `nftables` pour notre configuration de pare-feu. Dans la configuration de notre pare-feu, nous retrouvons deux tables :

— la table `filter`, permettant de filtrer les trames entrantes et sortantes du routeur (hooks `input` et `output`).

— la table `nat`, s'occupant de filtrer et rediriger les trames qui ne sont pas à destinations du routeur.

Dans la table `nat` table, on retrouve un ensemble d'adresses IP autorisées, ainsi qu'un ensemble de ports autorisés. Nous avons aussi une chaîne `forward`, dans laquelle nous filtrons quelles trames sont autorisées à poursuivre, en fonction des ensembles d'adresses IP et de ports définis précédemment. Dans cette table, se trouve aussi une chaîne `postrouting`, utilisant de la traduction d'adresses avec la règle `masquerade`, re-routant les paquets vers l'interface reliant le routeur à l'hyperviseur.

3.2.3 Configuration du routeur

Nous avons créé un routeur servant d'intersection entre nos réseaux de machines virtuelles et un de nos hyperviseur. Le routeur est configuré de telle sorte à ce que les paquets soient redirigé sur le sous réseau visé, et dans le cas où l'IP de destination est inconnue, les paquets sont envoyé à l'hyperviseur.

Cette configuration permet aux sous réseaux de machines virtuelles de communiquer entre eux, mais aussi d'accéder à Internet.

3.2.4 Pare-feu

Nous avons créé un fichier de configuration firewall sur le routeur. Nous utilisons `nftables` pour notre configuration de pare-feu. Dans la configuration de notre pare-feu, nous retrouvons deux tables :

- la table `filter`, permettant de filtrer les trames entrantes et sortantes du routeur (hooks `input` et `output`).
- la table `nat`, s'occupant de filtrer et rediriger les trames qui ne sont pas à destinations du routeur.

Dans la table `nat` table, on retrouve un ensemble d'adresses IP autorisées, ainsi qu'un ensemble de ports autorisés. Nous avons aussi une chaîne `forward`, dans laquelle nous filtrons quelles trames sont autorisées à poursuivre, en fonction des ensembles d'adresses IP et de ports définis précédement. Dans cette table se trouve aussi une chaîne `postrouting`, utilisant de la traduction d'adresses avec la règle `masquerade`, reroutant les paquets vers l'interface reliant le routeur à l'hyperviseur.

3.3 Les serveurs DHCP

Le serveur DHCP a été testé en local et configuré, mais il n'a pas encore été ajouté au réseau. Afin de distribuer des adresses IP à l'ensemble des machines virtuelles (VM), nous avons mis en place un serveur DHCP par sous-réseau. Ces serveurs ont été déployés à l'aide de Kea, un logiciel open source dédié à la gestion de l'attribution des adresses IP. Ils sont configurés pour fournir des baux d'une durée de huit jours, renouvelables lorsqu'une machine est connectée. Nous avons configuré Kea d'après suivant l'article https ://www.it-connect.fr/linux-installer-et-configurer-un-serveur-dhcp-kea-sur-debian/, nous avons créant un fichier qui nous sert de base à notre configuration.

3.4 Les serveurs DNS

Le serveur DNS est en préparation mais n'a pour le moment pas encore étais ajouter au réseau. Pour la résolution des noms de domaine au sein de nos réseaux, nous avons mis en place des serveurs DNS utilisant BIND 9. Chaque sous-réseau dispose de son propre serveur DNS, ce qui permet de limiter les dégâts en cas d'attaque, notamment en cas de DNS poisoning. Pour les résolutions DNS des machines en dehors de notre réseau, nous utilisons le serveur DNS de l'UIT.

4 Test de connectivité

Dans le cadre de ce projet, nous avons développé une solution automatisée permettant de tester la connectivité réseau de manière complète. Cette solution repose sur trois scripts principaux :

- Un script de test local : permet de tester la connectivité réseau de la machine exécutant le script.
- Un script de test distant : permet de tester la connectivité d'une machine distante.

— Un script d'exécution globale (*run_test*) : permet d'exécuter les tests locaux et distants de manière automatisée.

L'objectif de ces tests est de vérifier la disponibilité des services réseau, diagnostiquer d'éventuelles pannes et s'assurer que la configuration réseau est correcte.

4.1 Script de test local

4.1.1 Test de la passerelle par défaut

Le premier script est dédié à la vérification de la connectivité de la machine exécutant client. Il permet de tester :

— Récupération de la passerelle par défaut à l'aide de la commande : ip route
— Si une passerelle est détectée, un test de ping est lancé pour vérifier sa disponibilité.
— Si le test est positif, la machine est correctement connectée au réseau local.

4.1.2 Test de résolution DNS

— Vérification de la capacité du système à résoudre un nom de domaine en adresse IP.
— La fonction utilise socket.gethostbyname() pour convertir le nom de domaine en adresse IP.
— Si le test échoue, cela indique un problème de configuration DNS.

4.1.3 Test de connectivité externe (ping)

— Envoi de 4 paquets à une machine hors du réseau local (exemple : google.com).
— Vérification de :
  — Temps de réponse moyen (RTT)
  — Perte de paquets
  — Si le test échoue, cela indique une perte de connectivité avec l'extérieur.

Script de test distant de passerelle par défaut et DNS :

4.1.4 Test de ping vers une machine distante

Le deuxième script permet de tester la connectivité réseau d'une machine distante. Il réalise plusieurs types de tests pour diagnostiquer les problèmes de communication réseau.

— Envoi de 4 paquets à la machine distante à l'aide de la commande : ping -c 4 <IP>
— Vérification de :
  — Temps de réponse moyen
  — Perte de paquets
— Si le test échoue, cela indique un problème de routage ou de connectivité.

4.1.5 Test de routage (Traceroute)

— Utilisation de la commande traceroute pour suivre le chemin des paquets vers la machine distante.
— Si le test échoue, cela peut indiquer :

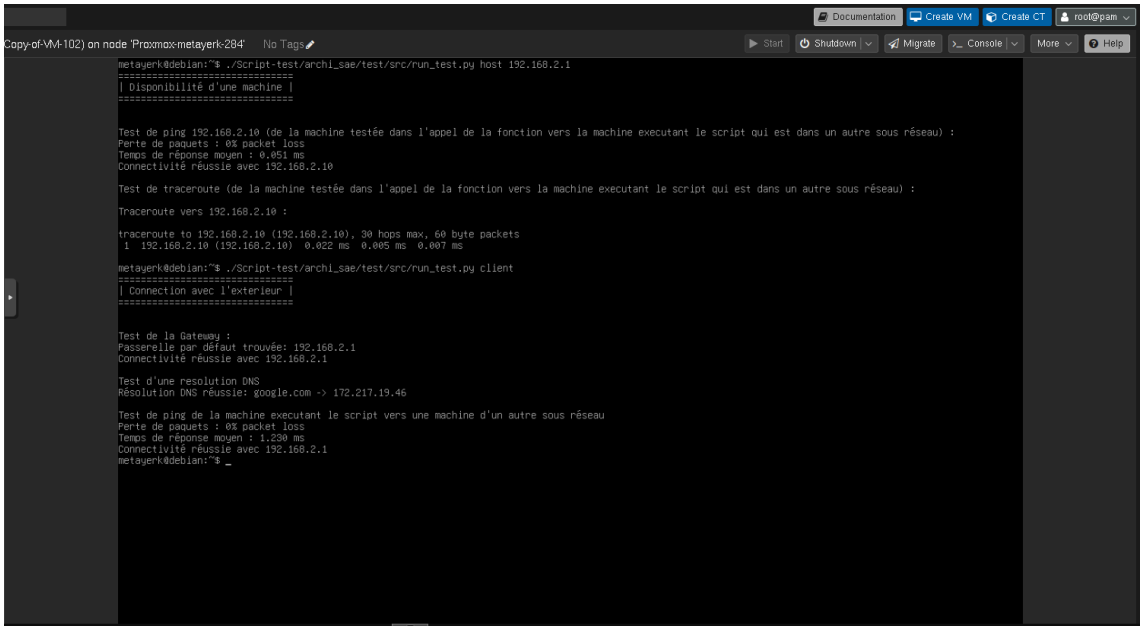

FIGURE 3 – Test distant de passerelle par défaut et DNS

  - Un problème de routage
  - Un pare-feu bloquant les paquets
- Affichage du chemin pris par les paquets et des éventuelles latences sur le trajet.
- Script global d'exécution (*main*)

4.2 Mode client (test local)

Le troisième script (*main*) permet de centraliser et d'automatiser l'exécution des tests précédents. Il utilise la bibliothèque argparse pour exécuter les tests en fonction des paramètres fournis.

La commande : python *main.py* client

Exécute les tests suivants :

- Test de la passerelle
- Test DNS
- Test de ping vers une machine externe

4.3 Mode distant (test d'une machine externe)

La commande : python *main.py* host <IP>

Exécute les tests suivants :

- Test de ping vers une machine distante
- Test de traceroute

Script de test distant pour vérifié l'accesibilité d'une machine :

FIGURE 4 – Test d'accesibilité d'une machine

5 Script d'installation de serveur

Pour le moment, nous avons réalisé le script de configuration de serveur DHCP. Il est écrit en python, prend en argument une IP. Il permet de générer le masque de sous-réseau, installer le logiciel KEA et de modifier le fichier "/etc/network/interfaces". Voir script dans les documents fournit, le nom : "$DHCP\_config.py$"

6 Utilisation de Matrix pour les communication

6.1 Organisation

Pour organiser nos communications, nous avons mis en place plusieurs salons sur notre serveur Matrix :

- **Général** : pour partager les informations courantes.
- **Journal de bord** : pour informer les autres membres du groupe de l'achèvement d'une tâche et envoyer un compte-rendu expliquant comment elle a été réalisée.
- **Réunion** : pour organiser des appels vocaux ou visioconférences.
- **Outils** : pour partager les outils utilisés lors de la SAE et poser des questions sur leur fonctionnement.

6.2 Fonctionnement

Nous profitons du fait que Matrix est un moyen de communication chiffré pour échanger des fichiers de configuration, des scripts ou d'autres informations devant rester confidentielles.

6.3 Utilisabilité

Malgré quelques petits problèmes rencontrés avec Matrix au début du projet, nous n'en avons plus constaté par la suite. Le logiciel est complet et dispose de toutes les

fonctionnalités dont nous avons eu besoin.

6.4 Sécurité

Matrix offre une grande confidentialité aux discussions de ses utilisateurs. Il utilise les fonctions de chiffrement AES-256 et HMAC-SHA-256 pour protéger et lire les messages du groupe. Pour accéder aux messages, les utilisateurs doivent s'authentifier mutuellement à l'aide d'une paire de clés. La clé publique utilisée est générée via la courbe `Curve25519`.