FICHE INDIVIDUELLE DE PROJET : ONIONROUTER

Nom / Prénom : Arjanit

Groupe : RayAnit

Projet : OnionRouter (R3.09 / SAÉ 3.02)

Rôle : Responsable Master, Données & IHM

1. Introduction : Mon rôle

Dans le cadre de ce projet de routeur virtuel distribué, j'ai occupé le rôle de **Responsable Master, Données et IHM**. Ma mission consistait à concevoir l'infrastructure centrale et l'expérience utilisateur du système OnionRouter. Pendant que mon binôme, Rayan, s'occupait de la logique réseau "bas niveau" et de la cryptographie, j'ai assuré la gestion du "cerveau" du réseau (le Master), le stockage des informations critiques (Base de données) et la création des interfaces graphiques permettant d'interagir avec le système.

2. Réalisations Techniques (Détail du code)

Mon travail s'est articulé autour de trois piliers majeurs que j'ai développés intégralement en Python :

A. Le Serveur Master (Le superviseur réseau)

J'ai codé le script central master.py qui agit comme l'autorité du réseau:

- **Gestion des Sockets :** J'ai mis en place un serveur TCP utilisant la bibliothèque socket pour écouter les routeurs. À chaque démarrage, un routeur se connecte au Master pour lui transmettre son adresse IP, son port et sa clé publique.

- **Distribution des Clés :** J'ai implémenté la logique permettant de répondre aux requêtes des clients. Le Master fournit la liste des routeurs actifs et leurs clés respectives afin que le client puisse construire ses couches de chiffrement ("l'oignon").

- **Maintien de la Topologie :** Le code gère dynamiquement une liste des nœuds actifs pour s'assurer que les routes proposées aux clients sont valides.

B. Gestion des Données (MariaDB)

Conformément aux contraintes, j'ai intégré une base de données MariaDB pour la persistance:

- **Schéma SQL :** J'ai conçu les tables pour stocker les clés publiques des routeurs, les tables de routage (Next Hop) et les logs système.

- **Automatisation :** J'ai utilisé le connecteur MariaDB pour Python afin d'automatiser l'insertion et la récupération des données en temps réel. J'ai veillé à ce que les journaux de bord soient **anonymisés** : on peut voir qu'un message transite, mais il est impossible de lier l'expéditeur initial au destinataire final via la base.

C. Interfaces Graphiques (PyQt)

J'ai développé deux interfaces distinctes avec PyQt5/PyQt6 pour rendre le système utilisable:

- **Interface Master :** Elle permet de visualiser l'état du réseau, de voir quels routeurs sont connectés et de consulter les logs de routage en direct.
- **Interface Client :** J'ai créé une fenêtre intuitive permettant de saisir un message et de l'envoyer de manière **non bloquante**.
- **Multi-threading :** Pour éviter que les interfaces ne "gèlent" pendant les échanges réseau, j'ai utilisé des Threads(QThread). Cela permet à l'interface de

rester réactive pendant que la logique de socket traite les données en arrière-plan.

3. Analyse des compétences (AC / CE)

Mon travail valide plusieurs compétences spécifiques du référentiel R&T :

- AC23.04 | Installer, administrer un système de gestion de données : J'ai validé cette compétence par la mise en place complète du serveur MariaDB et la définition du schéma des tables nécessaires au stockage des routes et des clés.

- AC23.05 | Accéder à un ensemble de données depuis une application : J'ai implémenté les requêtes SQL permettant au Master et aux routeurs d'interagir avec la base de données en temps réel via le code Python.
- AC23.02 | Développer une application à partir d'un cahier des charges : En utilisant PyQt, j'ai répondu à la contrainte de fournir des outils visuels pour l'administration et l'utilisation du réseau distribué.

- CE3.02 | En documentant le travail réalisé : J'ai rédigé le planning prévisionnel et les guides d'installation des interfaces graphiques.

- CE3.04 | En choisissant les outils de développement adaptés : Le choix de séparer l'IHM de la logique réseau via des classes distinctes a permis une intégration fluide avec le code de cryptographie de Rayan.

4. Conclusion sur mon apprentissage

Cette SAÉ a été une étape clé dans mon apprentissage de la **programmation événementielle** et de la gestion de systèmes distribués. J'ai découvert la complexité de synchroniser plusieurs machines virtuelles (Linux et Windows) et l'importance cruciale du multi-threading pour maintenir une interface utilisateur fluide.

Le plus gratifiant a été de voir, lors de nos tests finaux, le Master coordonner avec succès le passage d'un message à travers trois routeurs différents avant d'arriver au client destinataire. Malgré les difficultés liées à la gestion des ports et aux pannes réseau entre VM, ce projet m'a permis de comprendre concrètement comment fonctionnent les architectures de type "Tor".