المدرسة الوطنية للذكاء الاصطناعي والرقمنة - بركان
ÉCOLE NATIONALE DE L'INTELLIGENCE ARTIFICIELLE ET DU DIGITAL - BERKANE
ⵓⵏⵎⵓⴷⵓ ⵓⵏⵓⵎⵉⵔⵓ ⵏⵓⵏⵓⵔⵓ ⵓⵏⵓⴳⵎⵉⵓⵏⵉⵏ ⵍⵓⵏⵓⴷⵎⵍⵓ – ⴱⴰⵔⴽⵓ

Rapport de Projet de Fin d'Année

Filière : Ingénierie Réseaux et Sécurité Informatique

Conception et sécurisation d'une maison intelligente

Présenté le 24 juin 2025 par :

- DARRAZ Anas
- BAHMOU Lahoucine
- ERBATI Abdellatif

Devant le jury :

Pr. FARTITCHOU Mohamed	Encadrant
Pr. KHARBACH Amina	*Présidente*

Année Universitaire 2024-2025

Remerciements

Nous tenons à remercier toutes les personnes qui ont contribué au succès de notre projet de fin d'année et qui nous ont aidés lors de la rédaction de ce rapport.

Tout d'abord, nous adressons nos remerciements à notre encadrant
Pr. FARTITCHOU Mohamed qui nous a beaucoup aidé dans notre PFA, pour son écoute, ses conseils, et le temps qu'il a passé avec nous et le partage de son expertise au quotidien.

Par ailleurs, nous remercions également tous nos professeurs pour leurs orientations et recommandations tout au long de nos études.

Nos remerciements vont aussi à tous les membres de jury qui nous ont fait honneur d'évaluer et de juger notre PFA.

Nous remercions également tous nos collègues pour leur assistance, leur esprit d'équipe et leur partage de connaissances.

Enfin, nous tenons à remercier chaleureusement tous les membres de nos familles pour leur assistance.

Résumé

Ce projet porte sur la conception et la simulation d'une maison intelligente sécurisée à l'aide de deux environnements de simulation réseau.

Dans un premier temps, l'environnement Cisco Packet Tracer a été utilisé pour réaliser une maquette réseau de la maison. Cette topologie comprend des objets connectés (capteurs, caméras IP, actionneurs) interconnectés via un routeur. Des listes de contrôle d'accès (ACL) ont été mises en place afin de filtrer les communications réseau selon des règles précises, assurant un contrôle d'accès de base entre les différents segments.

Dans un second temps, une migration vers GNS3 a permis d'enrichir la simulation grâce à des outils plus avancés et réalistes. Le pare-feu pfSense a été installé pour appliquer des politiques de sécurité réseau strictes, incluant le filtrage de ports, la segmentation du réseau et la gestion des interfaces WAN/LAN. De plus, le protocole MQTT, largement utilisé dans les environnements IoT, a été intégré via le broker EMQX. La sécurité des échanges a été renforcée par l'implémentation d'un chiffrement TLS/SSL et d'un mécanisme d'authentification des clients,
garantissant la protection des données échangées entre les objets connectés et les systèmes de contrôle.

Le projet propose une approche à la fois pratique et pédagogique pour appréhender les principes de sécurisation des systèmes IoT, en combinant modélisation réseau, contrôle d'accès, chiffrement et configuration de pare-feu. Il constitue une base solide pour des développements futurs intégrant des outils de surveillance et de détection d'intrusion.

Mots-clés : Maison intelligente, IoT sécurisé, pfSense, MQTT (EMQX), TLS/SSL

Abstract

This project focuses on the design and simulation of a secure smart home using two network simulation environments.

In the first phase, Cisco Packet Tracer was used to build a basic network model of the smart home. The topology includes various IoT devices such as sensors, IP cameras, and actuators, all connected via a router. Access Control Lists (ACLs) were implemented to filter network traffic based on specific rules, providing fundamental access control between different network segments.

In the second phase, the project migrated to GNS3 to enable more advanced and realistic network configurations. The pfSense firewall was deployed to enforce strict security policies such as port filtering, network segmentation, and WAN/LAN interface management. In addition, the MQTT protocol—widely used in IoT systems were integrated through the EMQX broker. Secure communication was ensured using TLS/SSL encryption and client authentication mechanisms, thereby protecting data exchanged between connected devices and control systems.

This project adopts a practical and educational approach to understanding IoT security principles, combining network modeling, access control, encryption, and firewall configuration. It provides a solid foundation for future developments, including intrusion detection and monitoring tools.

Keywords : Smart Home, Secure IoT, pfSense, MQTT (EMQX), TLS/SSL

Liste des abréviations

PFA	Projet de fin d'année
ACL	Access Control Lists
GNS3	Graphical Network Simulator 3
WAN	Wide Area Network
LAN	Local Area Network
MQTT	Message Queuing Telemetry Transport
IoT	Internet of Things
TLS	Transport Layer Security
SSL	Secure Sockets Layer
LTE-M	Long Term Evolution for Machines
NB-IoT	Narrowband IoT
LPWAN	Low-power Wide-area Network
NFC	Near Field Communication
API	Application Programming Interface
VLAN	Virtual Local Area Network
CIAA	Confidentiality, Integrity, Availability, Authentication
VPN	Virtual Private Network
WPA3	Wi-Fi Protected Access version 3
HMAC	Hash-based Message Authentication Code
DoS	Denial of Service
IP	Internet Protocol
DMZ	Demilitarized Zone
CA	Certificate Authority
CCNA	Cisco Certified Network Associate
RFID	Radio Frequency Identification
AC	Air Cooler
HE	Heating Element
CO_2	Carbon Dioxide
SBC	Single Board Computer
R1, R2, R3	Routeur1, Routeur2, Routeur3
VM	Machine virtuelle

BSD	Berkeley Software Distribution
OS	Operating System
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
FAI	Fournisseur d'Accès à Internet
NAT	Network Address Translation
DHCP	Dynamic Host Configuration Protocol
SSH	Secure Shell
IPv4	Internet Protocol version 4
ICMP	Internet Control Message Protocol
CSR	Certificate Signing Request

Liste des figures

1.1 Architecture en couches d'un système IoT 18

2.1 Schéma de la maison intelligente sur Cisco Packet Tracer 28
2.2 configuration de base de passerelle domestique (Home Gateway) . . 32
2.3 Activation DHCP sur le serveur 32
2.4 Activation DHCP sur les composants IoT (caméra de surveillance) . 33
2.5 Activation de services IoT dans le serveur 33
2.6 Création d'un utilisateur sur le serveur 33
2.7 vérification des utilisateurs 34
2.8 Activation du contrôle à distance 34
2.9 Affichage des composants IoT connectés au serveur 35
2.10 L'ajout des conditions 35
2.11 Configuration des conditions 36
2.12 script du contrôle 39
2.13 Topologie utilisée pour la détection du mouvement 40
2.14 Topologie utilisée pour la détection de CO2 et le feu 40
2.15 Accès RFID autorisé 41
2.16 Accès RFID refusé 41
2.17 Architecture utilisée pour la détection de la température 42
2.18 Installation de MQTT 43
2.19 Configuration de Liste de Contrôle d'Accès (ACL) 44
2.20 Test de Connectivité Depuis l'extérieur vers le serveur (broker) . . . 44
2.21 Test de Connectivité Depuis l'extérieur vers l'interne (les objets IoT) 44
2.22 Configuration de la liste de contrôle de MQTT 44
2.23 La création des utilisateurs dans le broker 45
2.24 Vérification de la création des utilisateurs 45
2.25 Tentative de connecter au broker 46
2.26 Le transfert d'un message depuis le client 46
2.27 Vérification de la réception de message 47

3.1 Topologie utilisée sur gns3 51
3.2 Configuration de l'interface WAN 54
3.3 Configuration de l'interface LAN 55

3.4 Création des Vlans sur Switch 55
3.5 Assignation des ports aux Vlans(Access) 55
3.6 Assignation des ports aux Vlans(Trunk) 56
3.7 Création des Vlans sur pfsense 56
3.8 l'ajout des VLANs sur pfsense 56
3.9 Affectation des IP au VLAN broker 57
3.10 Affectation des IP au VLAN IoT 57
3.11 Configuration de DHCP sur LAN 58
3.12 Configuration de DHCP sur Vlan IoT 58
3.13 Configuration de DHCP sur Vlan broker 59
3.14 Attribution d'une adresse IP spécifique à la machine admin 59
3.15 Création d'un Alias 60
3.16 Définition des régles sur LAN 60
3.17 Définition des régles sur Vlan Broker 61
3.18 Définition des régles sur vlan IoT 61
3.19 Test d'un ping depuis l'admin vers le broker (vlan10) 62
3.20 Test d'un ping depuis l'admin vers vlan 20 62
3.21 Test d'un ping depuis l'admin vers l'exterieure 62
3.22 Test d'un ping depuis le Broker vers LAN 62
3.23 Test d'un ping depuis un composant IoT vers le Broker et l'exterieure 63
3.24 Status de serveur EMQX 64
3.25 Dashboard de EMQX 64
3.26 Génération de la clé de l'autorité du certificat 65
3.27 Certificat de l'autorité de certification. 65
3.28 Génération de la Clé privée du serveur 65
3.29 Génération de CSR du serveur 65
3.30 Fichier SAN (SubjectAltName) 66
3.31 Certificat de serveur signé 66
3.32 Importation des certificats dans EMQX 66
3.33 Configuration du chiffrement TLS 67
3.34 Importation des certificats 68
3.35 Activation du port sécurisé 68
3.36 Authentification et contrôle d'accès 69
3.37 Les clés TLS avec CA généré 70
3.38 Génération d'un certificat pour le client 70
3.39 Signature de CSR par CA 70
3.40 Test de validation 71
3.41 Authentification réussie par utilisateur pfa 72
3.42 La communication établie entre le client et le broker 72

3.43 La réception de message hello dans le broker 73
3.44 Les utilisateurs qui peuvent contacter le broker 73

Liste des tableaux

2.1 Les principales fonctionnalités de Cisco Packet Tracer 27
2.2 Composants d'infrastructure réseau 28
2.3 Dispositifs de surveillance 29
2.4 Capteurs de mouvement et de la lumière 29
2.5 Gestion de la température et de l'air 29
2.6 Contrôle d'accès par RFID 30
2.7 Sécurité incendie 30
2.8 Capteurs environnementaux complémentaires 30
2.9 Gestion de véhicules (garage) 31
2.10 Éléments d'ouverture 31

3.1 Tableau comparatif : Cisco Packet Tracer vs GNS3 50

Table des matières

Introduction générale **13**
Problématique 13

1 Présentation de la maison intelligente et de ses enjeux **14**
1.1 Introduction 14
1.2 Définition d'une maison intelligente 14
1.3 Architecture générale d'un système IoT 15
1.3.1 Couche perception (Perception Layer) 15
1.3.2 Couche réseau (Network Layer) 16
1.3.3 Couche de traitement (Processing Layer) 17
1.3.4 Couche application (Application Layer) 17
1.4 Problèmes de sécurité dans les environnements domestiques connectés 18
1.5 Objectifs de sécurité dans une maison intelligente 20
1.5.1 Confidentialité (Confidentiality) 20
1.5.2 Intégrité (Integrity) 20
1.5.3 Disponibilité (Availability) 21
1.5.4 Authentification (Authentication) 21
1.6 Choix des outils et technologies utilisés 22
1.7 Limites du projet 24
1.8 Conclusion 24

2 Simulation réseau initiale sous Cisco Packet Tracer **26**
2.1 Introduction 26
2.2 Présentation de Cisco Packet Tracer 26
2.3 Conception de la topologie réseau 27
2.4 Les étapes de configuration 31
2.4.1 Configuration du Routeur 31
2.4.2 Configuration de la passerelle domestique (Home Gateway) . . . 32
2.4.3 Activation DHCP sur le serveur IOT et les autres objets : 32
2.4.4 Activation de services IoT dans le serveur : 33
2.4.5 Créer un compte IoT pour gérer et lier les objets connectés : . . . 33

2.4.6 Activation de contrôle à distance : 34
2.5 Configuration des règles conditionnelles dans la maison intelligente . 35
2.6 Système d'automatisation basé sur des scripts 36
2.7 Vérification des scénarios d'automatisation via les scripts et les conditions 39
2.7.1 Activation automatique des lumières, alarme et caméras par détection de mouvement 39
2.7.2 Sécurité incendie automatisée 39
2.7.3 Contrôle d'accès RFID intelligent 41
2.7.4 Régulation intelligente de la température 42
2.7.5 Installation de MQTT Broker et Client 42
2.8 Configuration des ACLs (Access Control List) 43
2.8.1 Qu'est-ce qu'une ACL ? 43
2.8.2 Types des ACLs (populaires) 43
2.8.3 Applications des règles 43
2.8.4 Vérification 45
2.9 Conclusion 47

3 Mise en œuvre de la sécurité dans la maison intelligente via GNS3 48
3.1 Introduction 48
3.2 Transition vers GNS3 pour une simulation avancée 49
3.2.1 Présentation de GNS3 et ses avantages face à Packet Tracer 49
3.2.2 Objectifs de la migration 49
3.2.3 Description de la nouvelle topologie 51
3.2.4 Outils utilisés dans la simulation avancée 52
3.3 Sécurisation réseau avec pfSense 53
3.3.1 Présentation de pfSense 53
3.3.2 Déploiement de pfSense dans GNS3 53
3.3.3 Résultats des tests avec pfSense 61
3.4 Mise en place du broker MQTT sécurisé avec EMQX 63
3.4.1 Pourquoi MQTT dans une maison intelligente ? 63
3.4.2 Installation et configuration de EMQX 63
3.4.3 Sécurisation des communications avec TLS/SSL 65
3.4.4 Authentification mutuelle par certificats 69
3.4.5 Tests et validation 70
3.5 Conclusion 73

Conclusion générale 75

Bibliographie 77

Annexes **78**
Annexe A 78
Annexe B 80
Annexe C 82
Annexe D 84
Annexe E 85
Annexe F 86
Annexe G 86

Introduction générale

Au fil des dernières décennies, les progrès rapides dans les technologies de l'information, des télécommunications et de l'électronique ont profondément transformé notre quotidien. Parmi ces innovations, la maison intelligente s'impose comme l'un des symboles majeurs de cette révolution technologique. Elle intègre un ensemble de dispositifs interconnectés permettant d'automatiser, de contrôler à distance et d'optimiser différentes fonctions domestiques : éclairage, chauffage, sécurité, consommation énergétique, et bien plus encore.

Cependant, cette interconnexion croissante des objets du quotidien, bien qu'elle apporte confort, efficacité et gain de temps, soulève également d'importants défis, notamment en matière de sécurité et de protection des données. En effet, chaque appareil connecté constitue une porte d'entrée potentielle pour des cyberattaques, menaçant non seulement la vie privée des occupants mais aussi leur sécurité physique. Face à cette réalité, il devient impératif de concevoir des systèmes domotiques à la fois performants, fiables et sécurisés.

En effet, la majorité des systèmes domotiques repose sur des protocoles sans fil, des capteurs et des interfaces accessibles à distance, ce qui en fait des cibles privilégiées pour des attaques malveillantes telles que le piratage, le vol de données personnelles, ou encore la prise de contrôle à distance des équipements. Par ailleurs, les utilisateurs finaux ne sont pas toujours conscients des risques liés à l'utilisation de ces technologies, ni formés à appliquer des pratiques de sécurité de base.

Problématique

Comment concevoir une maison intelligente capable de répondre aux exigences de confort et d'automatisation, tout en garantissant un haut niveau de sécurité pour les données, les dispositifs et les utilisateurs ? Quels choix technologiques, méthodologiques et organisationnels doivent être privilégiés pour atteindre cet équilibre entre innovation, performance et cybersécurité ?

Chapitre 1

Présentation de la maison intelligente et de ses enjeux

1.1 Introduction

Avec l'essor rapide de l'Internet des Objets (IoT), les maisons intelligentes s'imposent comme une composante majeure de la transformation numérique de notre quotidien. En intégrant des dispositifs interconnectés capables d'automatiser et de contrôler les fonctions de l'habitat, elles offrent des bénéfices notables en matière de confort, de sécurité et d'efficacité énergétique. Toutefois, cette modernisation du cadre de vie soulève également de nouveaux défis, notamment sur le plan de la cybersécurité. Ce chapitre propose une présentation globale du concept de maison intelligente, en décrivant son architecture technique, les problématiques de sécurité qu'elle engendre, ainsi que les objectifs fondamentaux à atteindre pour garantir un environnement domestique fiable et sécurisé.

1.2 Définition d'une maison intelligente

Une maison intelligente, ou *smart home*, est une habitation équipée de dispositifs technologiques interconnectés qui permettent d'automatiser, contrôler et optimiser divers aspects de la vie domestique. Ces dispositifs peuvent inclure des capteurs, des actionneurs, des caméras, des systèmes d'éclairage, des thermostats, des serrures électroniques ou encore des assistants vocaux intelligents. Ils sont généralement connectés à un réseau local ou à Internet et sont pilotables à distance via une application mobile, une interface web ou des

commandes vocales.

L'objectif principal d'une maison intelligente est d'améliorer le confort, la sécurité, l'efficacité énergétique et la gestion des ressources de l'habitat. Par exemple, l'utilisateur peut programmer l'extinction automatique des lumières, la régulation du chauffage en fonction de sa présence, ou encore recevoir une alerte en cas d'intrusion détectée par une caméra ou un capteur de mouvement.

Le fonctionnement d'une maison intelligente repose sur des technologies de communication comme le Wi-Fi, Bluetooth, ZigBee, Z-Wave, ou encore le MQTT, qui permettent aux appareils de communiquer entre eux ou avec un serveur central (souvent appelé hub domotique ou *home gateway*).

Avec l'essor de l'Internet des Objets (IoT), les maisons intelligentes sont devenues un secteur clé de la transformation numérique de l'habitat, mais elles introduisent également de nouveaux défis en matière de cybersécurité, car chaque objet connecté représente une porte d'entrée potentielle pour les attaques.

1.3 Architecture générale d'un système IoT

Un système IoT (*Internet of Things* - Internet des objets) repose sur une architecture en couches qui permet de connecter des objets physiques au monde numérique, afin de collecter, transmettre, traiter et exploiter des données en temps réel. Dans le contexte d'une maison intelligente, cette architecture comprend généralement trois niveaux principaux :

1.3.1 Couche perception (Perception Layer)

Cette couche est responsable de convertir des signaux analogiques en données numériques et vice versa. L'étape initiale de tout système IoT englobe un large éventail d'objets qui agissent comme un pont entre les mondes réel et numérique.

Leur forme et leur taille varient, des minuscules puces de silicium aux gros véhicules. Par leurs fonctions, les objets IoT peuvent être divisés en groupes :

— **Capteurs** : tels que sondes, gauges, compteurs et autres. Ils collectent des paramètres physiques tels que la température ou l'humidité, les transforment en signaux électriques et les envoient au système IoT. Les capteurs IoT sont généralement petits et consomment peu d'énergie.

— **Actionneurs** : traduisant les signaux électriques du système IoT en actions physiques. Les actionneurs sont utilisés dans les contrôleurs de moteur, les lasers, les bras robotiques.
— **Machines et dispositifs** : connectés à des capteurs et des actionneurs ou comme étant des parties intégrantes.

Il est important de noter que l'architecture n'impose aucune restriction sur la portée de ses composants ou leur emplacement. En effet, la couche latérale peut inclure quelques objets physiquement placés dans une pièce ou des myriades de capteurs et d'appareils répartis dans le monde entier.

1.3.2 Couche réseau (Network Layer)

Les données collectées par tous ces appareils doivent être transmises et traitées. C'est le travail de la couche réseau, elle connecte donc ces appareils à d'autres objets intelligents, serveurs et appareils réseau. Elle gère également la transmission de toutes les données.

Les communications entre les appareils et les services cloud ou les passerelles impliquent différentes technologies :

— **Ethernet** : connecte des appareils IoT fixes tels que des caméras de sécurité et vidéo, des équipements industriels installés en permanence.
— **Réseaux cellulaires** : Les plus connus actuellement étant la 5G et la 4G, elles offrent un transfert de données fiable et une couverture presque mondiale. Il existe aussi deux normes cellulaires développées spécifiquement pour les objets IoT. LTE-M (*Long Term Evolution for Machines*) permet aux appareils de communiquer directement avec le cloud et d'échanger de gros volumes de données. NB-IoT (*Narrowband IoT*) utilise des canaux basse fréquence pour envoyer de petits paquets de données.
— **LPWAN** (*Low-power Wide-area Network*) : a été créée spécifiquement pour les appareils IoT. Cette technologie offre une connectivité sans fil longue portée avec une faible consommation d'énergie avec une autonomie de plus de 10 ans.
— **WiFi** : la technologie de réseau sans fil la plus populaire,convient idéalement aux solutions IoT gourmandes en données, faciles à recharger et à utiliser dans une petite zone. Un bon exemple d'utilisation est celui des appareils domestiques intelligents connectés au réseau électrique.

Il existe aussi d'autres technologies permettant la connectivité comme le NFC (*Near Field Communication*), le Bluetooth ou le ZigBee.

1.3.3 Couche de traitement (Processing Layer)

La couche de traitement accumule, stocke et traite les données provenant de la couche précédente. Toutes ces tâches sont généralement traitées via des plateformes IoT et comprennent deux étapes principales.

Étape d'accumulation des données Les données en temps réel sont capturées via une API et mises au repos pour répondre aux exigences des applications non temps réel. L'étape du composant d'accumulation de données fonctionne comme une plaque tournante entre la génération de données basée sur les événements et la consommation de données basée sur les requêtes.

Entre autres choses, l'étape définit si les données sont pertinentes pour les besoins de l'entreprise et où elles doivent être placées. Elle enregistre les données dans une large gamme de solutions de stockage, des *data lakes* capables de contenir des données non structurées telles que des images et des flux vidéo.

L'objectif global étant de trier une grande quantité de données diverses et de les stocker de la manière la plus efficace.

Étape d'abstraction des données Ici, la préparation des données est finalisée afin que les applications grand public puissent les utiliser pour générer des informations. L'ensemble du processus comprend les étapes suivantes :

— Combiner des données provenant de différentes sources, à la fois IoT et non-IoT
— Concilier plusieurs formats de données
— Agréger les données en un seul endroit ou les rendre accessibles quel que soit leur emplacement grâce à la virtualisation des données.

De même, les données collectées au niveau de la couche application sont reformatées ici pour être envoyées au niveau physique afin que les appareils puissent les comprendre. Ensemble, les étapes d'accumulation et d'abstraction des données masquent les détails du matériel, améliorant ainsi l'interopérabilité des appareils intelligents.

1.3.4 Couche application (Application Layer)

La couche application est ce avec quoi l'utilisateur interagit. C'est ce qui est chargé de fournir des services spécifiques à l'application à l'utilisateur.

Actuellement, les applications peuvent être construites directement sur les plates-formes IoT qui offrent une infrastructure de développement logiciel avec des outils prêts à l'emploi pour l'exploration de données, l'analyse avancée et la visualisation de données. Sinon, les applications IoT utilisent des API pour s'intégrer à la couche précédente.

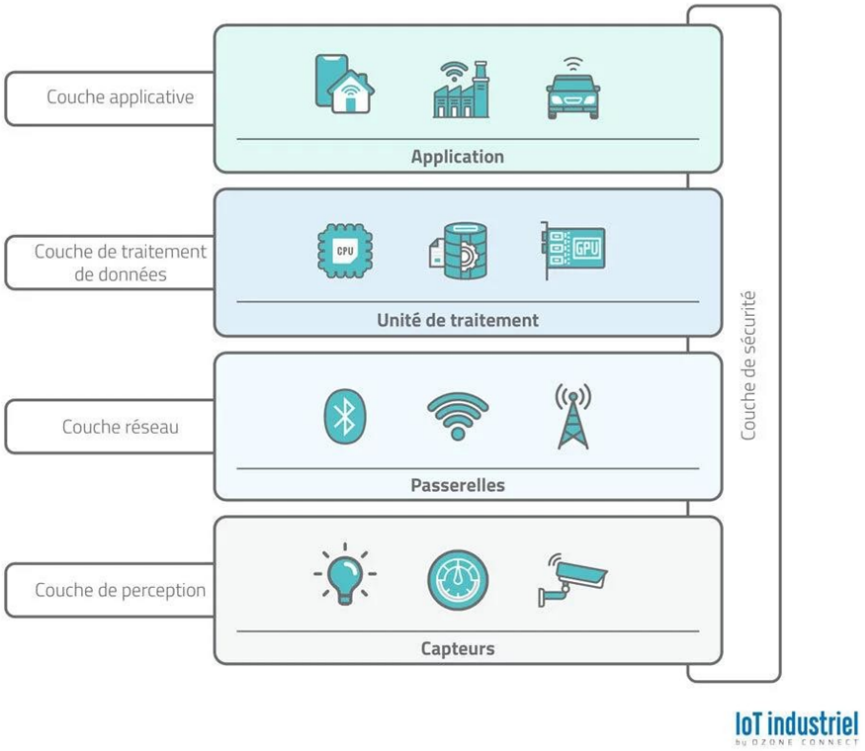

FIGURE 1.1 – Architecture en couches d'un système IoT

1.4 Problèmes de sécurité dans les environnements domestiques connectés

L'émergence des maisons intelligentes s'accompagne de nombreux avantages, mais elle soulève également des problèmes de sécurité majeurs, car chaque objet connecté devient une porte d'entrée potentielle pour les cyberattaques. La nature distribuée et souvent mal protégée des dispositifs IoT rend ces environnements particulièrement vulnérables.

Voici les principales menaces et problématiques de sécurité rencontrées dans les systèmes domotiques :

- **Manque de chiffrement des communications** : Beaucoup d'objets connectés transmettent les données en clair, ce qui expose les échanges à des attaques de type :
  - Interception de données sensibles (*sniffing*)
  - Altération ou injection de commandes via des attaques *Man-in-the-Middle*

  Sans protocole de sécurité comme TLS/SSL, un attaquant peut facilement intercepter ou modifier les données.
- **Authentification faible ou absente** : Certains dispositifs utilisent encore des identifiants par défaut (ex. : admin/admin) ou n'intègrent aucun système d'authentification robuste. Cela facilite les attaques :
  - Accès non autorisé aux équipements
  - Prise de contrôle à distance de la maison intelligente
- **Absence de mises à jour logicielles** : De nombreux objets IoT ne reçoivent pas ou très peu de mises à jour *firmware*, ce qui laisse des failles non corrigées exploitables à long terme par les attaquants.
- **Topologies mal segmentées** : Les dispositifs connectés partagent souvent le même réseau que les ordinateurs personnels, ce qui permet à une attaque sur un objet IoT de s'étendre à l'ensemble du réseau domestique. Cela rend nécessaire :
  - L'usage de VLAN ou de zones de sécurité
  - L'intégration de pare-feux comme pfSense
- **Manque de journalisation et de surveillance** : Contrairement aux systèmes professionnels, les environnements domestiques manquent souvent de mécanismes de détection d'intrusion ou de *logs* d'activité. Cela rend difficile :
  - L'identification des comportements suspects
  - La réponse rapide en cas d'attaque
- **Problèmes de confidentialité** : Les objets connectés collectent souvent des données personnelles sensibles (habitudes de vie, présence, vidéos...). Une mauvaise gestion de la confidentialité peut entraîner :
  - Des atteintes à la vie privée
  - Des risques de profilage ou d'espionnage à des fins malveillantes

En résumé, la sécurisation d'un environnement domestique connecté ne se limite pas à installer des équipements performants. Elle nécessite :

- Une approche globale incluant chiffrement, segmentation, authentification forte

— Une configuration rigoureuse des dispositifs
— Une supervision minimale pour détecter les anomalies

Ces risques justifient pleinement l'intégration de mécanismes comme des ACL, des certificats TLS, et des pare-feux dédiés dans les maisons intelligentes, comme l'a démontré ce projet.

1.5 Objectifs de sécurité dans une maison intelligente

La sécurité d'un système de maison intelligente repose sur les principes fondamentaux de la sécurité de l'information, souvent résumés par le modèle CIAA : Confidentialité, Intégrité, Disponibilité et Authentification. Ces objectifs sont essentiels pour garantir la confiance, la fiabilité et la résilience des dispositifs IoT utilisés dans l'environnement domestique.

1.5.1 Confidentialité (Confidentiality)

La confidentialité vise à empêcher l'accès non autorisé aux données échangées ou stockées dans le système. Dans une *smart home*, cela inclut :

— Les données des capteurs (présence, température, caméras)
— Les historiques de commandes (fermeture de porte, allumage de lumière, etc.)
— Les identifiants de connexion aux interfaces de contrôle

Mesures à appliquer :

— Utilisation du chiffrement (TLS/SSL, VPN)
— Réseaux Wi-Fi protégés (WPA3)
— Cloisonnement réseau et pare-feu (pfSense)

1.5.2 Intégrité (Integrity)

L'intégrité garantit que les données n'ont pas été modifiées ou altérées volontairement ou accidentellement. Dans une *maison intelligente*, il est crucial que :

— Les commandes envoyées aux dispositifs soient valides (ex. : ne pas allumer une alarme par erreur)
— les informations collectées par les capteurs soient fiables

Mesures à appliquer :

- Signatures numériques
- Contrôle d'intégrité des messages MQTT (par exemple via TLS avec HMAC)
- Détection d'anomalies ou de comportements incohérents

1.5.3 Disponibilité (Availability)

La disponibilité vise à garantir que les services sont accessibles et fonctionnels quand l'utilisateur en a besoin. Dans un système domotique, cela signifie :

- Que les capteurs et actionneurs répondent en temps réel
- Que le réseau domestique n'est pas surchargé ou bloqué
- Que l'utilisateur peut toujours accéder à l'interface de contrôle (localement ou à distance)

Menaces courantes :

- Attaques de type DoS (*Denial of Service*)
- Défaillance de l'équipement réseau (box, routeur, séparément local)

Solutions possibles :

- Supervision réseau
- Systèmes redondants ou sauvegardes
- Règles de pare-feu pour limiter le trafic anormal

1.5.4 Authentification (Authentication)

L'authentification permet de vérifier l'identité des utilisateurs ou des objets connectés avant de leur accorder un accès au système. Dans une *maison intelligente*, il est essentiel de :

- Contrôler l'accès des utilisateurs à l'interface mobile ou web
- Vérifier l'identité des objets connectés avant de leur autoriser des échanges

Mesures à appliquer :

- Authentification par mot de passe fort ou double facteur
- Utilisation de certificats numériques pour les objets (ex :EMQX authentification TLS mutuelle)
- Limitation des accès aux seuls périphériques autorisés

1.6 Choix des outils et technologies utilisés

La mise en œuvre d'une maison intelligente sécurisée nécessite le recours à un ensemble d'outils logiciels et matériels permettant à la fois la simulation du réseau, la gestion des communications IoT, et la mise en place de mécanismes de sécurité. Les choix effectués dans ce projet ont été guidés par des critères de réalisme, de modularité et de capacité à intégrer des fonctionnalités avancées (comme les certificats TLS ou la segmentation réseau). Voici les principales technologies utilisées :

Cisco Packet Tracer

Rôle : Simulation initiale de la maison intelligente
Utilisation :

- Conception d'un réseau domestique simple (PC, capteurs simulés, routeur, switch)
- Mise en place de listes de contrôles d'accès (ACL) sur un routeur Cisco pour filtrer le trafic selon des règles prédéfinies
- Simulation du comportement réseau de la maison intelligente de manière visuelle et interactive

Avantages :

- Facile à utiliser pour des tests rapides
- Représentation claire des flux réseau et règles ACL
- Outil éducatif très répandu

GNS3 (Graphical Network Simulator 3)

Rôle : Simulation avancée et environnement réseau réaliste
Utilisation :

- Simulation de réseaux plus complexes et réalistes avec de vrais systèmes d'exploitation
- Intégration de pfSense comme pare-feu virtuel
- Interconnexion entre objets IoT, pare-feu, brokers MQTT et clients distants

Avantages :

- Supporte de vrais systèmes (ISO, images QEMU, etc.)

— Permet de tester des scénarios de cybersécurité réels
— Émulation proche d'un déploiement réel

pfSense

Rôle : Pare-feu et routeur sécurisé
Utilisation :

— Définition de règles de pare-feu précises (filtrage IP, ports, VLAN)
— Gestion des zones de sécurité (LAN, WAN, DMZ)
— Intégration de certificats TLS/SSL
— Contrôle des accès aux services comme MQTT

Avantages :

— Interface web intuitive
— Large communauté et documentation
— Support des fonctions avancées de sécurité réseau

EMQX (Eclipse MQTT Broker)

Rôle : Broker MQTT sécurisé pour communications IoT
Utilisation :

— Gestion des communications éditeur/abonné (*publish/subscribe*) entre objets connectés
— Mise en place d'un serveur MQTT sécurisé avec certificats TLS
— Authentification des clients MQTT

Avantages :

— Performant et compatible avec les standards IoT
— Support de TLS, ACL, authentification et plugins
— Interface de gestion et journalisation des connexions

OpenSSL

Rôle : Génération de certificats et clés pour TLS
Utilisation :

— Création d'une autorité de certification (CA)
— Génération de certificats client/serveur pour sécuriser MQTT avec TLS

— Signature et vérification des certificats

Avantages :

— Outil en ligne de commande puissant
— Utilisé dans de nombreux systèmes de sécurité
— Permet de comprendre en détail le fonctionnement des certificats

1.7 Limites du projet

Malgré les résultats satisfaisants obtenus et les apprentissages techniques mobilisés, ce projet présente un certain nombre de limites qu'il convient de reconnaître pour garder une vision réaliste et critique de la solution développée.

A. Simulation uniquement logicielle

Toutes les phases du projet ont été réalisées dans des environnements de simulation (Cisco Packet Tracer, GNS3), sans connexion à du matériel physique réel (Raspberry Pi, capteurs physiques, etc.). Cela limite l'observation de certains comportements réels comme les temps de réponse, la consommation électrique, ou les variations environnementales.

B. Fonctionnalités encore incomplètes

Certaines fonctionnalités prévues ou envisagées sont partiellement intégrées :

— Le système de détection d'intrusion (**Snort**) est installé mais pas encore pleinement opérationnel.
— Le contrôle à distance via une interface mobile ou cloud n'a pas été implémenté.
— La gestion des logs et des alertes de sécurité est minimale.

1.8 Conclusion

La maison intelligente représente une avancée technologique prometteuse, transformant les usages quotidiens grâce à l'automatisation et à l'interconnexion des objets. Néanmoins, cette évolution s'accompagne de menaces importantes liées à la sécurité des données, à la confidentialité des utilisateurs et à la fiabilité

des équipements. Pour faire face à ces enjeux, il est indispensable d'adopter une approche sécurisée dès la conception du système, en s'appuyant sur des technologies éprouvées, une architecture bien segmentée, et des mécanismes de protection robustes comme le chiffrement, l'authentification forte et les pare-feux. Ce cadre conceptuel servira de fondation aux chapitres suivants, qui détailleront la mise en œuvre pratique et les solutions adoptées dans le cadre du projet.

Chapitre 2

Simulation réseau initiale sous Cisco Packet Tracer

2.1 Introduction

La maison intelligente repose sur l'intégration d'objets connectés capables de percevoir leur environnement et d'agir de manière autonome selon des règles programmées. L'utilisation de simulateurs comme Packet Tracer permet de modéliser et d'analyser le fonctionnement de ces systèmes complexes, notamment en configurant les interactions entre capteurs, actionneurs et serveurs. Ce type d'environnement facilite également l'étude des protocoles de communication, tels que MQTT, indispensables pour assurer un échange efficace des données entre les différents composants.

Par ailleurs, la sécurisation du réseau domestique par des mécanismes comme les listes de contrôle d'accès (ACL) est essentielle pour protéger les communications et garantir la fiabilité du système domotique.

2.2 Présentation de Cisco Packet Tracer

Cisco Packet Tracer est un outil de simulation réseau développé par Cisco Systems. Il permet de concevoir, configurer et tester des réseaux informatiques dans un environnement virtuel. Ce simulateur est largement utilisé dans les cursus de formation réseau (notamment Cisco CCNA) grâce à sa simplicité d'utilisation et à sa capacité à reproduire de manière graphique le comportement des équipements réseau.

Objectifs dans le cadre du projet

Dans le cadre de ce projet de maison intelligente sécurisée, Cisco Packet Tracer a été utilisé comme première étape de simulation, afin de :

— Visualiser l'architecture réseau d'une maison intelligente : poste utilisateur, objets connectés, routeur, switch, serveur local.
— Configurer des ACL (*Access Control Lists*) sur un routeur Cisco pour filtrer le trafic réseau selon des critères de sécurité (adresse IP, port, protocole).
— Comprendre la circulation des paquets et les effets des politiques de filtrage.
— Tester des scénarios simples d'accès autorisé/interdit entre équipements.

Fonctionnalités pertinentes

Fonctionnalité	**Utilisation dans le projet**
Ajout de périphériques	Routeur, switch, PC, capteurs simulés
Câblage virtuel	Connexions entre équipements (Ethernet)
Configuration CLI du routeur	Application d'ACL en ligne de commande
Simulation de paquets	Visualisation du chemin réseau et des paquets bloqués/autorisés

TABLE 2.1 – Les principales fonctionnalités de Cisco Packet Tracer

Avantages de Packet Tracer

— Interface intuitive et graphique, adaptée aux débutants.
— Ne nécessite pas d'équipements réels.
— Permet des tests rapides de configuration.
— Facilité de sauvegarde et de duplication des topologies.

2.3 Conception de la topologie réseau

La topologie réseau mise en place dans Cisco Packet Tracer représente une maison intelligente interconnectée via un réseau local filaire et sans fil. Elle inclut plusieurs types de dispositifs : routeurs, switchs, caméras, capteurs IoT, lecteurs RFID,
actionneurs, et éléments de sécurité. Chaque composant a un rôle précis dans l'automatisation et la sécurisation de l'environnement domestique.

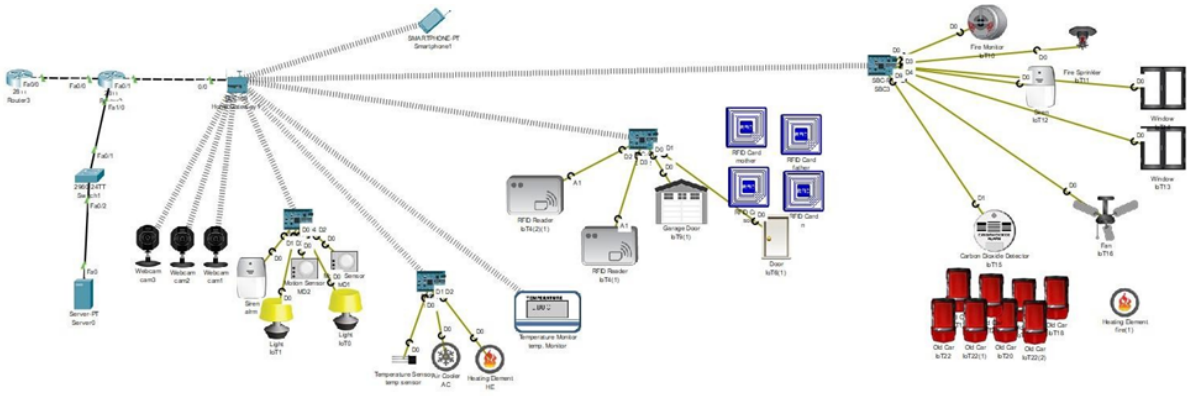

FIGURE 2.1 – Schéma de la maison intelligente sur Cisco Packet Tracer

Dispositifs utilisés sur l'architecture

Composant	**Description**	**Utilité**
Router-PT (Router3)	Routeur Cisco (2811)	Permet de relier le réseau local (LAN) à un réseau externe ou au cloud, et de configurer les ACL pour filtrer les flux.
Switch-PT (Switch0)	Switch Cisco 2960	Relie les périphériques du réseau local et assure la commutation des paquets.
Switch-PT-24TT (Switch1)	Switch 24 ports	Connecte un ensemble de capteurs IoT filaires et caméras.
Server-PT (Server0)	Serveur local	Peut servir pour l'hébergement web, la gestion de bases de données ou de services domotiques internes.
Home Gateway0	Passerelle domestique	Point de connexion central entre les objets connectés et le reste du réseau (possiblement routeur Wi-Fi intégré).
Smartphone-PT	Smartphone connecté	Appareil de l'utilisateur servant à piloter la maison à distance via Wi-Fi.

TABLE 2.2 – Composants d'infrastructure réseau

Composant	Description	Utilité
Webcam cam1/cam2/cam3	Caméras de surveillance	Surveillance vidéo des différentes zones (intérieur, garage, entrée).
Siren (Siren alarm)	Sirène d'alarme	Déclenchée en cas d'intrusion ou d'incident détecté.

TABLE 2.3 – Dispositifs de surveillance

Composant	Description	Utilité
Motion Sensor MD1/MD2	Capteurs de mouvement	Détectent la présence ou le passage dans une pièce, déclenchant la lumière ou une alarme.
Light IoT1/IoT2	Lampes connectées	Allumées automatiquement en cas de mouvement détecté ou via commande mobile.

TABLE 2.4 – Capteurs de mouvement et de la lumière

Composant	Description	Utilité
Temperature Sensor	Capteur de température	Relevé de la température ambiante pour régulation automatique.
Temperature Monitor	Affichage de température	Permet de visualiser les données du capteur.
Air Cooler (AC)	Climatisation	Refroidit l'air en fonction des données du capteur de température.
Heating Element (HE)	Chauffage	Chauffe l'environnement si la température est trop basse.

TABLE 2.5 – Gestion de la température et de l'air

Composant	Description	Utilité
RFID Reader bT4/bT4(1)	Lecteurs RFID	Permettent l'identification des personnes via des cartes RFID.
RFID Cards/Mother/-Father/Son	Badges d'accès	Chaque carte est associée à un utilisateur autorisé.
Garage Door bT5(1)	Porte de garage connectée	S'ouvre automatiquement si une carte RFID autorisée est détectée.
Door IoT6(1)	Porte d'entrée connectée	Gérée aussi par RFID, pour contrôler les accès à la maison.

TABLE 2.6 – Contrôle d'accès par RFID

Composant	Description	Utilité
Fire Monitor bT10	Détecteur de fumée	Déclenche une alarme en cas de détection d'incendie.
Fire Sprinkler bT11	Arroseur automatique	Active le système d'arrosage en cas d'incendie.

TABLE 2.7 – Sécurité incendie

Composant	Description	Utilité
Smoke IoT12	Détecteur de fumée secondaire	Redondance de la sécurité incendie.
Carbon Dioxide Detector bT15	Détecteur de CO_2	Alerte en cas de niveau dangereux de gaz dans l'air.
Fan bT16	Ventilateur intelligent	Aère automatiquement en cas de pollution ou de chaleur.

TABLE 2.8 – Capteurs environnementaux complémentaires

Composant	Description	Utilité
Old Car bT22(1-6)	Voitures simulées	ÉLEMENTS décoratifs de simulation représentant les véhicules dans le garage.
Heating Element fire(1)	Élément chauffant	Simulation d'un incident thermique dans la zone du garage.

TABLE 2.9 – Gestion de véhicules (garage)

Composant	Description	Utilité
Windows bT13 (x2)	Fenêtres connectées	Peuvent être fermées ou ouvertes automatiquement (sécurité ou ventilation).

TABLE 2.10 – Éléments d'ouverture

2.4 Les étapes de configuration

2.4.1 Configuration du Routeur

Étape 1 : Ajouter une adresse IP au routeur (Gateway)

```
Router>enable
Router#configure terminal
Router(config)#interface g0/0
Router(config-if)#ip address 192.168.25.1 255.255.255.0
Router(config-if)#no shutdown
Router(config-if)#exit
```

Étape 2 : Créer un pool DHCP sur le routeur

```
Router(config)#ip dhcp pool IOT
Router(dhcp-config)#network 192.168.1.1 255.255.255.0
Router(dhcp-config)#default-router 192.168.1.1
Router(dhcp-config)#dns-server 192.168.1.2
Router(dhcp-config)#end
Router#wr
```

2.4.2 Configuration de la passerelle domestique (Home Gateway)

— Ajouter une passerelle domestique.

— Configuration du système de commande à distance (Wi-Fi), changer le SSID en 'pilote', cliquer sur WPA2-PSK et taper 'pilote@2025'.

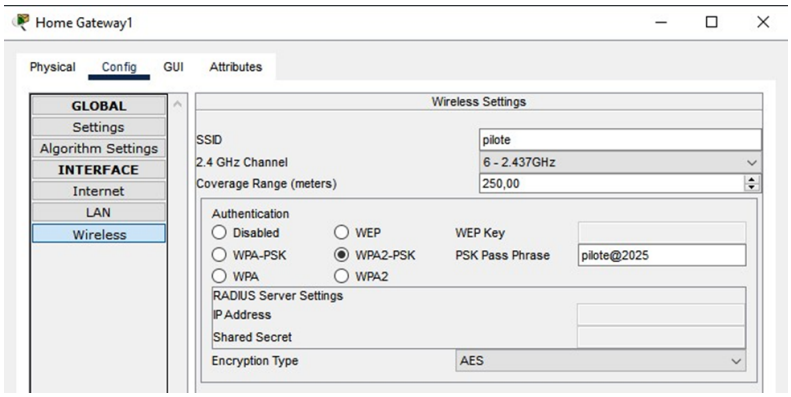

FIGURE 2.2 – configuration de base de passerelle domestique (Home Gateway)

2.4.3 Activation DHCP sur le serveur IOT et les autres objets :

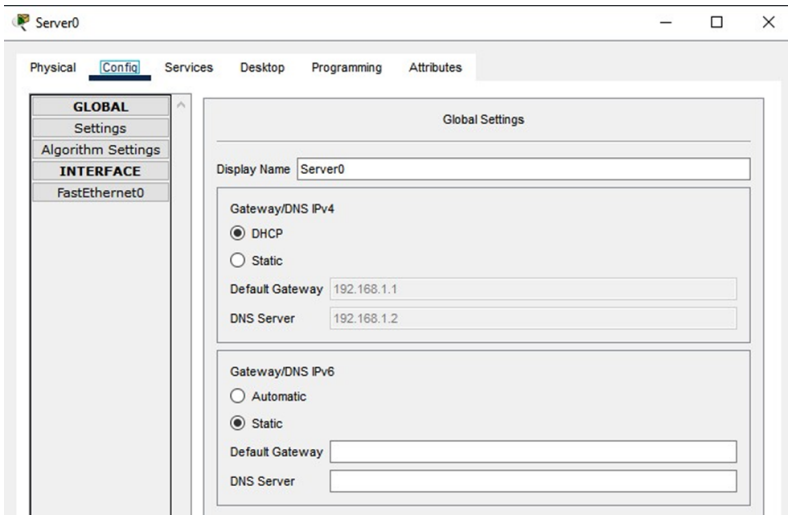

FIGURE 2.3 – Activation DHCP sur le serveur

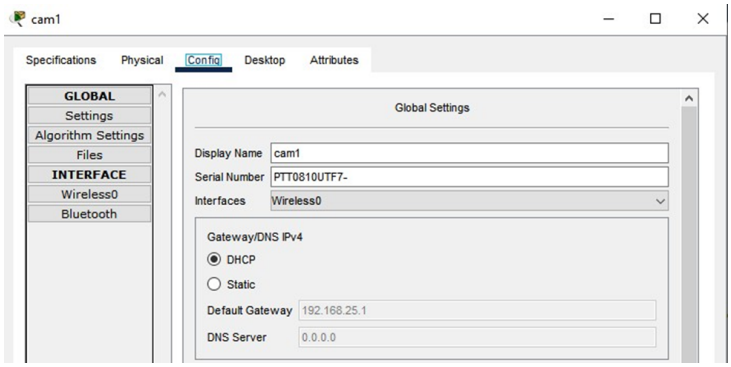

FIGURE 2.4 – Activation DHCP sur les composants IoT (caméra de surveillance)

2.4.4 Activation de services IoT dans le serveur :

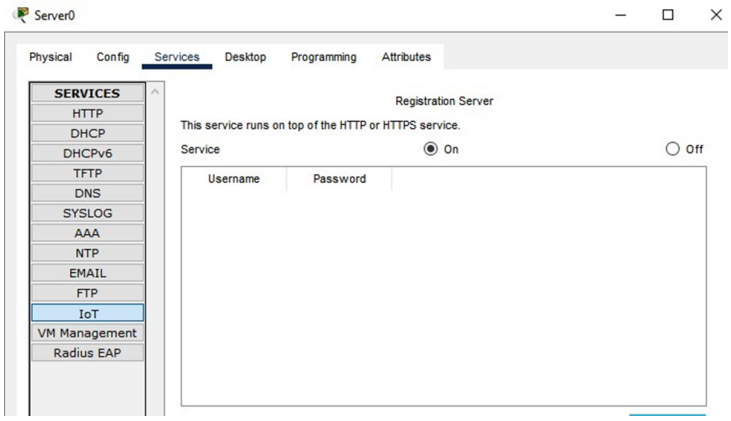

FIGURE 2.5 – Activation de services IoT dans le serveur

2.4.5 Créer un compte IoT pour gérer et lier les objets connectés :

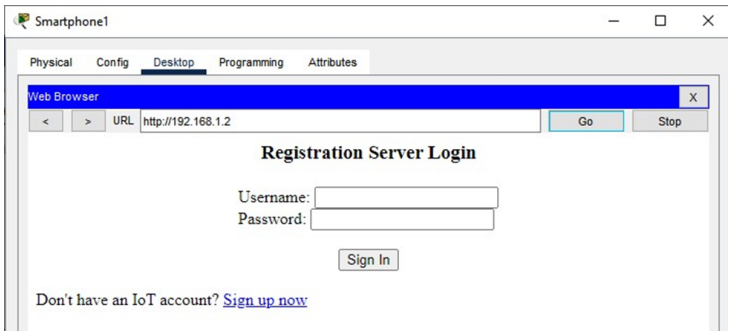

FIGURE 2.6 – Création d'un utilisateur sur le serveur

On appuie sur « sign up now » pour la création d'un compte. Pour confirmer l'ajout du compte, nous consultons le serveur pour examiner le service IoT. Comme illustré dans la figure ci-dessus, le compte a été correctement intégré.

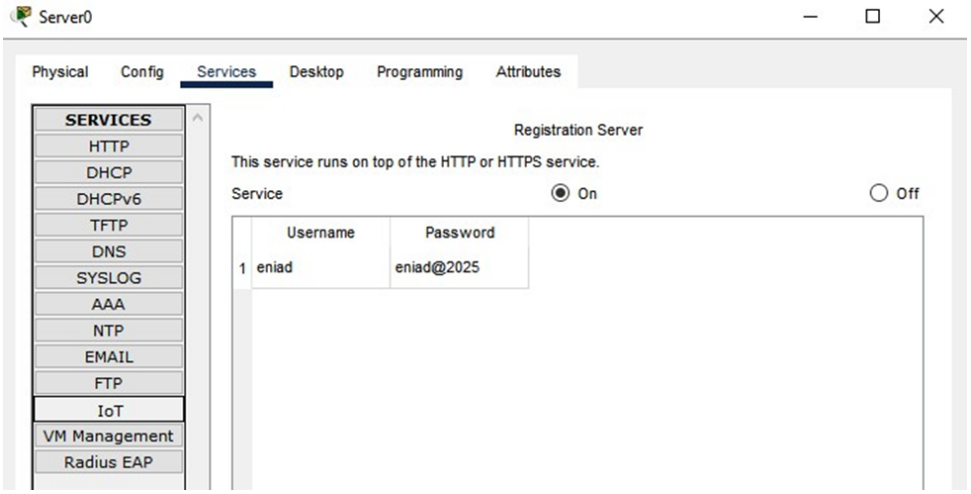

FIGURE 2.7 – vérification des utilisateurs

2.4.6 Activation de contrôle à distance :

Chaque composant est configuré avec le mode Remote Server activé, et les informations de connexion au serveur IoT (adresse, identifiant, mot de passe) sont saisies.

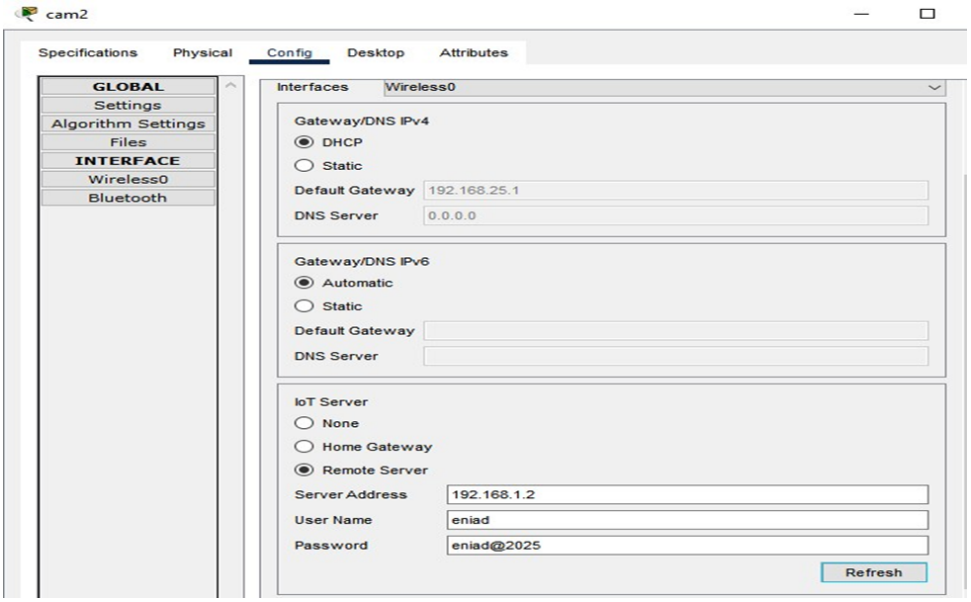

FIGURE 2.8 – Activation du contrôle à distance

Après avoir ajouté remote server dans tous les composants, voici la liste du contrôle à distance.

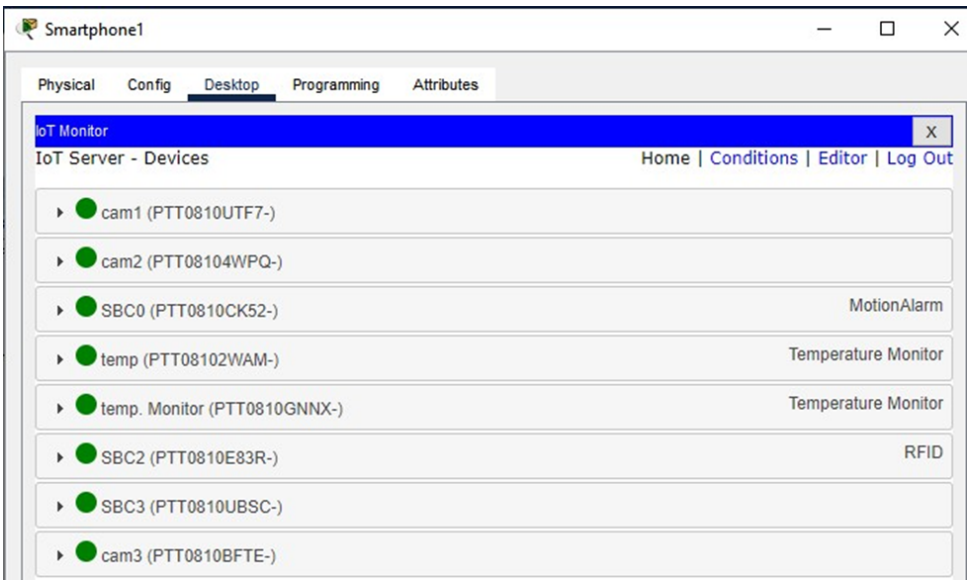

FIGURE 2.9 – Affichage des composants IoT connectés au serveur

2.5 Configuration des règles conditionnelles dans la maison intelligente

Permettre à la maison de réagir automatiquement à certains événements détectés par des capteurs grâce à des conditions logiques configurées dans le portail IoT.

Pour ajouter une condition, nous accédons à l'IoT Server et entrons dans la section dédiée aux conditions, comme indiqué dans la figure Nous cliquons sur

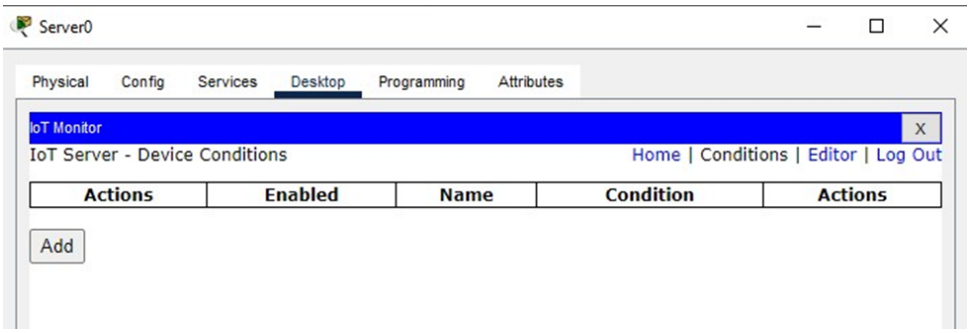

FIGURE 2.10 – L'ajout des conditions

« Add » pour ajouter une règle. Dans cet exemple, la caméra de surveillance est sélectionnée. Comme illustré dans la figure, une règle a été définie : lorsque le

capteur de mouvement détecte une activité, la caméra s'active automatiquement.

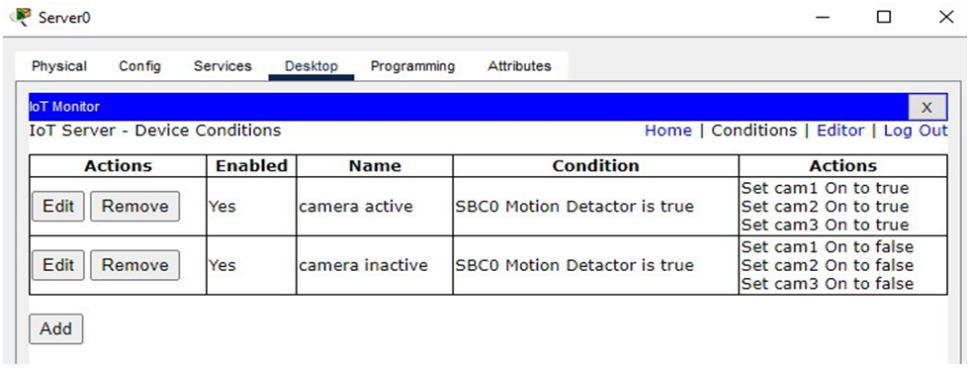

FIGURE 2.11 – Configuration des conditions

2.6 Système d'automatisation basé sur des scripts

Dans le cadre de notre projet, nous avons opté pour la configuration des règles d'automatisation non pas via les conditions simples proposées par l'interface web, mais en recourant à des scripts personnalisés.

Pourquoi utiliser des scripts ?

L'utilisation de scripts dans les objets IoT de Packet Tracer offre plusieurs avantages :

— Plus de contrôle logique : possibilité de combiner plusieurs conditions complexes.

— Flexibilité : gestion précise des actions, temporisations, ou scénarios multiples.

— Centralisation : logique de contrôle intégrée à un seul objet (par exemple un capteur ou la passerelle).

Exemple de script

Voici un exemple simple de script utilisé dans SBC0 (capteur de mouvement) :

```
var state = 0;

function setup() {
  pinMode(1, OUTPUT);
  pinMode(0, INPUT);
  pinMode(2, INPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);

  IoEClient.setup({
    type: "MotionAlarm",
    states: [{
      name: "On",
      type: "bool",
      controllable: true
    },
    {
      name: "Motion Detector",
      type: "bool"
    },
    {
      name: "Alarm",
      type: "bool"
    }
    ]
  });

  IoEClient.onInputReceive = function(input) {
    processData(input, true);
  };

  setState(state);
  }

function processData(data) {
  if ( data.length <= 0  )
    return;
  setState(parseInt(data));
```

```
}

function setState(newState,motion,alarm) {
  state = newState;

  if ( state === 0 ){
    IoEClient.reportStates([state,false,false]);
  }
  else{
    IoEClient.reportStates([state,motion,alarm]);
  }
}

function loop() {

  if (state !== 0){
    var motion1 = digitalRead(0);
    var motion2 = digitalRead(2);
    var mt = false;
    //Serial.println(motion);

    if (motion1 !== 0 || motion2 !== 0){
      mt = true;
      customWrite(1,1);
      customWrite(3,1);
      customWrite(4,1);
    }else {
      mt = false;
      customWrite(1,0);
      customWrite(3,0);
      customWrite(4,0);
    }
    IoEClient.reportStates([state,mt,mt]);
  }

  delay(1000);
}
```

Voici dans la figure l'emplacement des scripts

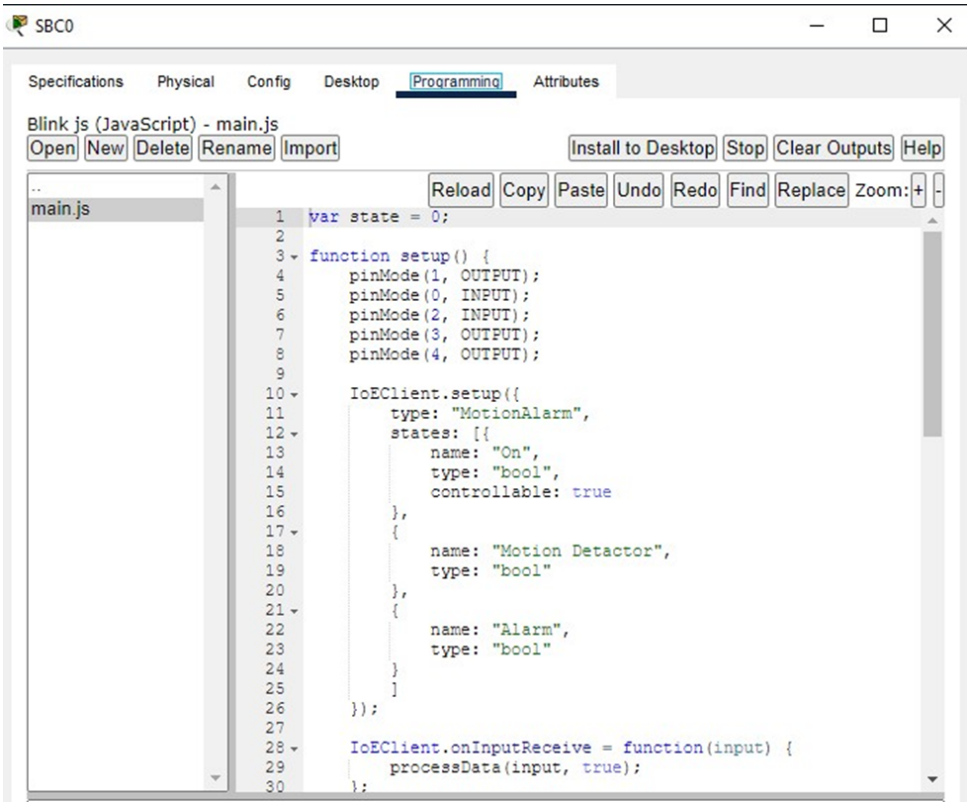

FIGURE 2.12 – script du contrôle

2.7 Vérification des scénarios d'automatisation via les scripts et les conditions

2.7.1 Activation automatique des lumières, alarme et caméras par détection de mouvement

Description :

— Script déclenché lorsqu'un capteur de mouvement détecte une présence.

— Les lumières, l'alarme et les caméras s'allument automatiquement.

2.7.2 Sécurité incendie automatisée

Description :

— Lorsque le détecteur de CO_2, de fumée ou d'incendie est activé :

— L'alarme sonore (*Siren*) se déclenche pour alerter les occupants.

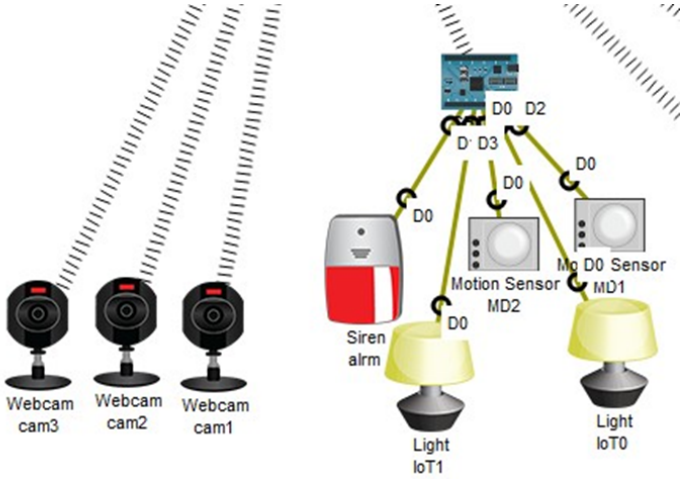

FIGURE 2.13 – Topologie utilisée pour la détection du mouvement

— Les sprinklers s'activent pour éteindre ou limiter les flammes.
— Les fenêtres s'ouvrent automatiquement pour permettre l'aération ou évacuer les fumées.
— Le ventilateur (*Fan*) se met en marche pour accélérer l'évacuation de l'air vicié.

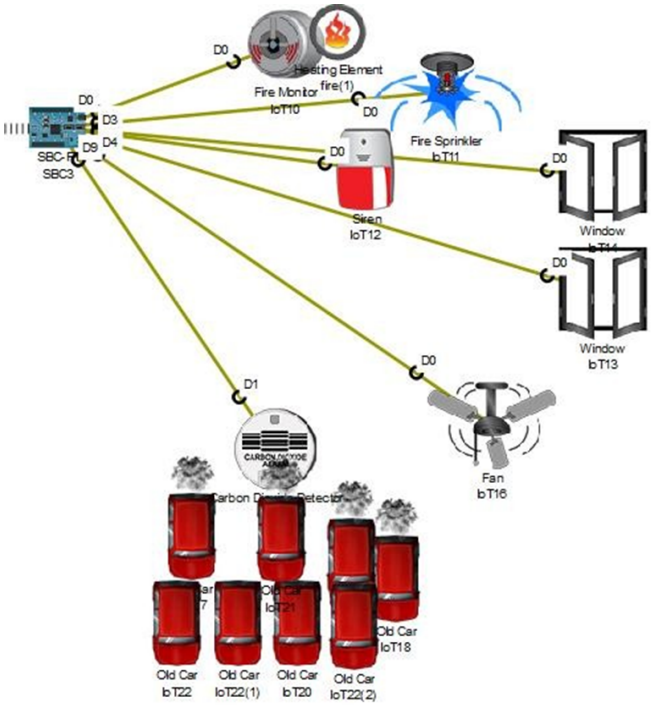

FIGURE 2.14 – Topologie utilisée pour la détection de CO2 et le feu

2.7.3 Contrôle d'accès RFID intelligent

Description :

— Quand un badge RFID valide est détecté : La porte du garage ou l'entrée principale s'ouvre.

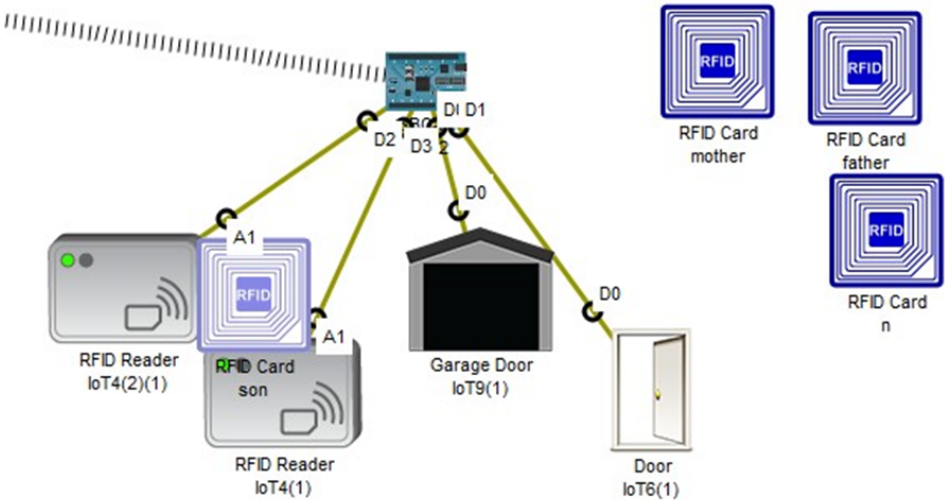

FIGURE 2.15 – Accès RFID autorisé

— Quand un badge RFID invalide est détecté : La porte du garage ou l'entrée principale ne s'ouvre pas.

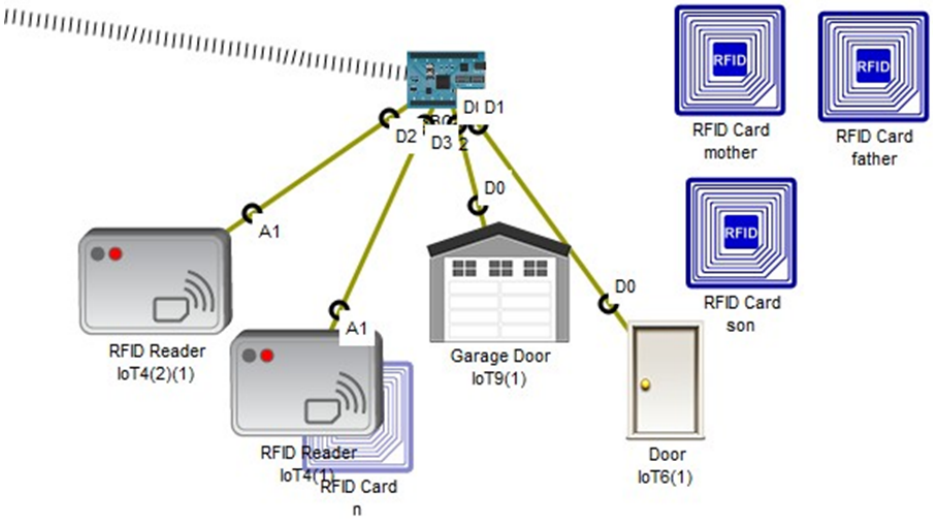

FIGURE 2.16 – Accès RFID refusé

2.7.4 Régulation intelligente de la température

Ce sous-système vise à automatiser le contrôle thermique dans la maison intelligente en s'appuyant sur les données d'un capteur de température. En fonction de la température mesurée, le système décide automatiquement d'activer ou de désactiver les éléments de chauffage et de refroidissement, de manière à maintenir une température ambiante confortable.

Ce type de régulation est essentiel pour le confort des occupants, la gestion énergétique efficace, ainsi que pour prévenir les surchauffes ou les refroidissements excessifs.

Le système repose sur un script JavaScript embarqué. La logique est la suivante :

— Si la température est inférieure à $15°$C, le chauffage est activé.
— Si la température est supérieure à $25°$C, le climatiseur est activé.
— Si la température comprise entre $15°$C et $25°$C, aucun système n'est activé (zone de confort).

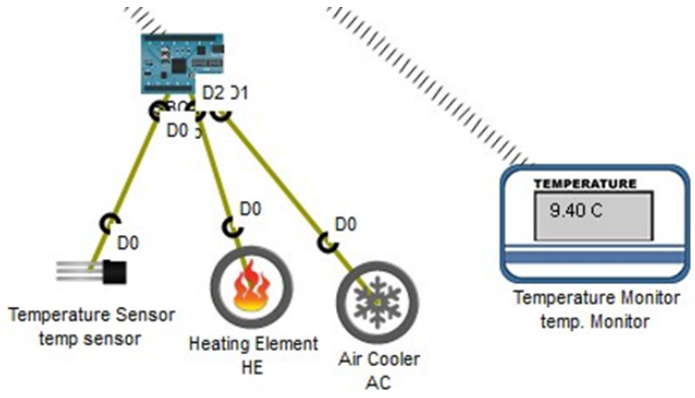

FIGURE 2.17 – Architecture utilisée pour la détection de la température

2.7.5 Installation de MQTT Broker et Client

Afin de permettre aux appareils IoT de communiquer entre eux, nous installons le protocole MQTT à la fois du côté client (au niveau des objets IoT) et du côté broker (au niveau du serveur pour gérer les échanges).

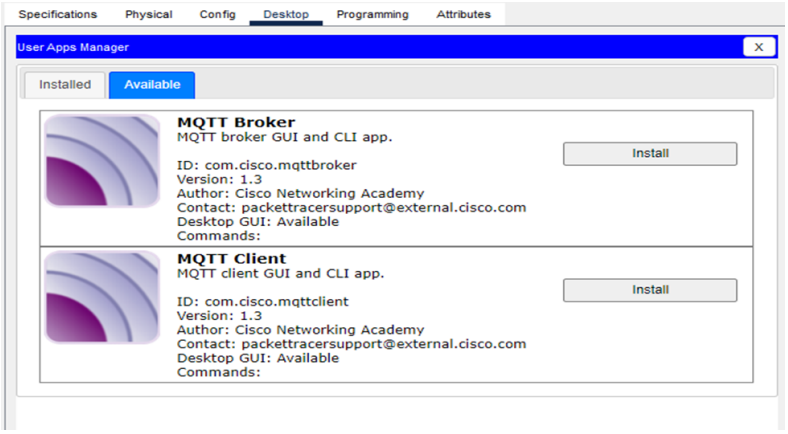

FIGURE 2.18 – Installation de MQTT

2.8 Configuration des ACLs (Access Control List)

2.8.1 Qu'est-ce qu'une ACL ?

Une ACL est une liste de règles qui spécifie qui peut accéder à une ressource et ce qu'il peut faire avec cette ressource.

2.8.2 Types des ACLs (populaires)

— **ACL standard** : Filtre le trafic en fonction de l'adresse IP source uniquement. Utilisée pour autoriser ou bloquer globalement un hôte ou un réseau.

— **ACL étendue** : Filtre le trafic selon l'adresse IP source et destination, le protocole et les ports.

2.8.3 Applications des règles

Dans le cadre de la sécurisation du réseau interne, nous avons mis en place une stratégie de filtrage visant à bloquer tout trafic entrant depuis l'extérieur, notamment depuis le Routeur 3, considéré ici comme une source non fiable. Pour cela, nous avons utilisé une liste de contrôle d'accès étendue (ACL) appliquée sur le Routeur 2, à l'entrée de l'interface connectée à R3. Cette ACL permet de refuser toute tentative de communication en provenance du réseau externe, tout en maintenant les échanges nécessaires à l'intérieur du réseau, comme la communication MQTT entre les objets IoT et le serveur. Ce filtrage préventif renforce la sécurité du cœur du réseau en stoppant les flux non autorisés avant qu'ils n'atteignent les ressources internes sensibles.

```
Router>enable
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#access-list 102 deny ip 24.17.2.0 0.0.0.255 any
Router(config)#access-list 102 permit ip any any
Router(config)#interface f0/0
Router(config-if)#ip access-group 102 in
```

FIGURE 2.19 – Configuration de Liste de Contrôle d'Accès (ACL)

Afin de valider le bon fonctionnement de la liste de contrôle, un test d'accès est réalisé depuis un appareil situé à l'extérieur du réseau interne, afin de confirmer que toute tentative de connexion non autorisée est effectivement bloquée.

Depuis l'extérieur vers le serveur (broker) :

```
Router>enable
Router#ping 192.168.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.2, timeout is 2 seconds:
UUUUU
Success rate is 0 percent (0/5)
```

FIGURE 2.20 – Test de Connectivité Depuis l'extérieur vers le serveur (broker)

Depuis l'extérieur vers l'interne (les objets IoT) :

```
Router#ping 24.17.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 24.17.1.1, timeout is 2 seconds:
UUUUU
Success rate is 0 percent (0/5)
```

FIGURE 2.21 – Test de Connectivité Depuis l'extérieur vers l'interne (les objets IoT)

Dans ce cas, nous avons choisi d'utiliser une liste de contrôle d'accès étendue afin d'autoriser uniquement la communication via le protocole MQTT, qui utilise le port 1883 (ou 8883 en version sécurisée), et de permettre aux appareils IoT de communiquer avec le broker, qui est dans ce cas le serveur.

```
Router>enable
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#access-list 101 permit tcp 24.17.1.0 0.0.0.255 host 192.168.1.2 eq 1883
Router(config)#access-list 101 permit tcp 24.17.1.0 0.0.0.255 host 192.168.1.2 eq 8883
Router(config)#access-list 101 deny ip any any
Router(config)#interface f0/1
Router(config-if)#ip access-group in
% Incomplete command.
Router(config-if)#ip access-group 101 in
```

FIGURE 2.22 – Configuration de la liste de contrôle de MQTT

2.8.4 Vérification

Tout d'abord, afin de sécuriser l'accès au broker MQTT, nous avons commencer par créer les utilisateurs autorisés à s'y connecter.

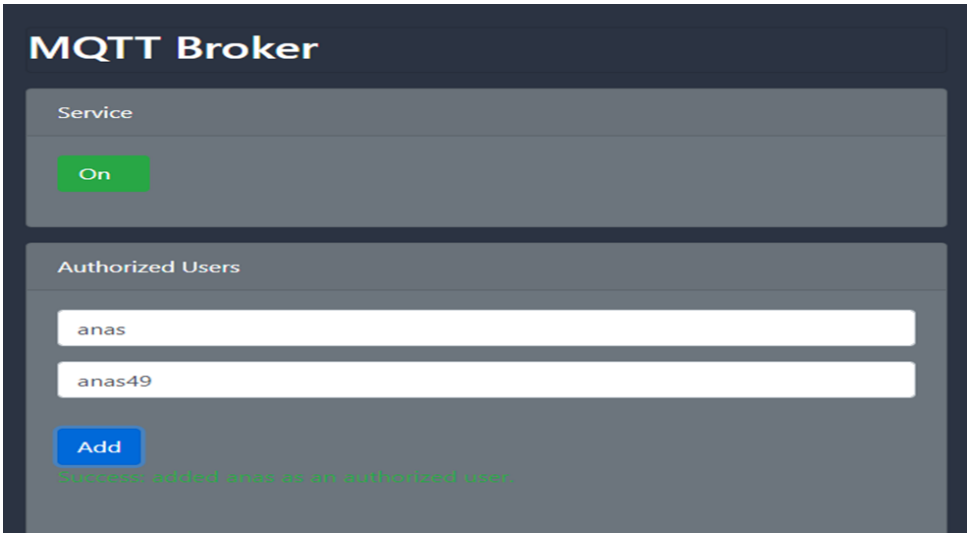

FIGURE 2.23 – La création des utilisateurs dans le broker

La liste des utilisateurs créés dans le serveur (Broker) :

Username	Password
anas	anas49
abdellatif	abdellatif@
lahocine	lahocine!!

FIGURE 2.24 – Vérification de la création des utilisateurs

Nous effectuons maintenant un test afin de vérifier si la communication entre les appareils IoT et le broker MQTT est bien établie.

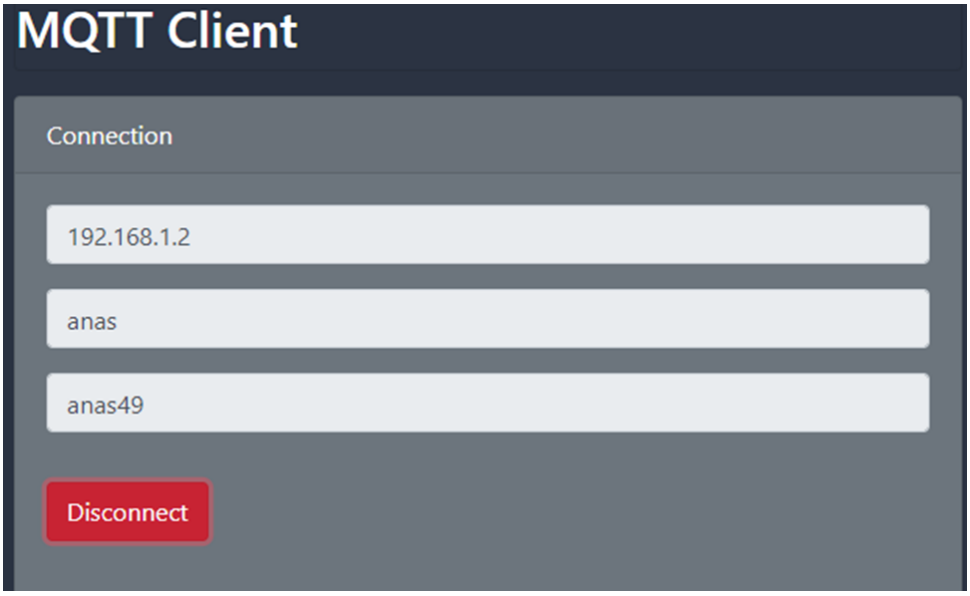

FIGURE 2.25 – Tentative de connecter au broker

Côté Client :

FIGURE 2.26 – Le transfert d'un message depuis le client

Côté Broker :

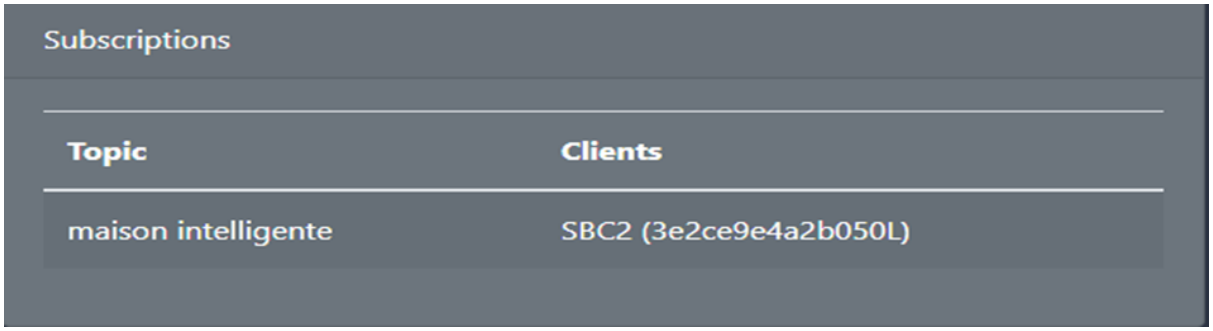

FIGURE 2.27 – Vérification de la réception de message

Comme vous pouvez le constater, le message a bien été reçu par sa destination, ce qui confirme que la communication fonctionne correctement.

2.9 Conclusion

Le travail réalisé avec Packet Tracer a permis de concevoir et tester un système domotique intelligent complet, intégrant la détection, l'automatisation et la communication sécurisée entre objets connectés. L'utilisation de scripts personnalisés a offert une grande flexibilité dans la gestion des scénarios automatisés, tandis que le protocole MQTT a facilité la communication entre les composants.

Enfin, la mise en place de mécanismes de sécurité via des listes de contrôle d'accès a renforcé la protection du réseau, assurant ainsi un environnement IoT fiable et sécurisé. Cette simulation complète dans Packet Tracer constitue une base solide pour le développement de solutions domotiques réelles.

Chapitre 3

Mise en œuvre de la sécurité dans la maison intelligente via GNS3

3.1 Introduction

Ce chapitre traite de la conception, de la configuration et de la sécurisation d'un environnement réseau dédié à l'Internet des objets (IoT) dans le contexte d'une maison intelligente. La segmentation du réseau en VLANs distincts, tels que les VLANs Broker et IoT, permet d'assurer une isolation efficace et un contrôle d'accès précis entre les différents équipements et services.

La gestion du trafic est assurée par le pare-feu pfSense a été configuré pour appliquer des règles de sécurité strictes ,et assure la gestion efficace des ressources tout en facilitant l'administration. Par ailleurs, le déploiement du broker MQTT EMQX assure une communication légère, rapide et sécurisée entre capteurs et applications IoT.

La sécurisation des échanges MQTT est renforcée par la mise en place du chiffrement TLS/SSL et l'authentification mutuelle, incluant la génération et l'intégration des certificats. Enfin, des tests de validation confirment l'efficacité et la robustesse de cette architecture sécurisée

3.2 Transition vers GNS3 pour une simulation avancée

3.2.1 Présentation de GNS3 et ses avantages face à Packet Tracer

GNS3 (Graphical Network Simulator 3) est un outil de simulation réseau qui permet d'assembler, tester et simuler des infrastructures complètes à l'aide de véritables systèmes d'exploitation réseau. Contrairement à Cisco Packet Tracer, qui se limite à des équipements Cisco virtuels et à un environnement éducatif restreint, GNS3 offre une plus grande flexibilité et un réalisme accru grâce à l'intégration de VMs (machines virtuelles), conteneurs Docker, systèmes BSD/Linux, pare-feu, et services réseau complets.

Avantages principaux de GNS3 :

- **Réalité de la simulation** : possibilité de travailler avec des OS réels comme pfSense, Ubuntu, Kali Linux, etc.
- **Support de services avancés** : intégration de pare-feu, IDS/IPS, brokers MQTT, etc.
- **Fonctionnalité VLAN et routage avancé** : prise en charge du VLAN tagging, du routage dynamique et du firewalling complexe.
- **Évolutivité** : possibilité d'ajouter autant de composants que nécessaire (selon les ressources disponibles).
- **Interopérabilité** : les machines peuvent interagir avec le réseau physique de l'hôte (accès Internet réel, mises à jour, etc.).

3.2.2 Objectifs de la migration

La migration de Cisco Packet Tracer vers GNS3 s'inscrit dans une volonté d'améliorer la fiabilité, la précision et la portée de la simulation. Plus précisément, les objectifs visés sont :

- Créer une infrastructure IoT sécurisée à travers une topologie réseau réaliste.
- Déployer et tester des outils de cybersécurité réels tels que pfSense, EMQX, TLS, VLAN, etc.
- Séparer les flux réseau entre composants critiques via des VLANs (isolation des IoT, du broker, de l'admin).
- Simuler des attaques et vérifier la réaction du système de détection (Snort).
- Mettre en place des services IoT (MQTT) avec authentification, contrôle d'accès et chiffrement TLS.

Dans le cadre de ce projet, la première phase a été réalisée sur Cisco Packet Tracer, principalement pour concevoir une maquette fonctionnelle de la maison intelligente et simuler les objets IoT dans un environnement simple et visuel. Cependant, bien que Packet Tracer soit un outil pédagogique très utile, il présente certaines limitations en matière de sécurité et de virtualisation réseau avancée.

Pour intégrer des fonctionnalités plus réalistes et approfondir l'aspect sécurité réseau, une migration vers GNS3 s'est imposée. Cet outil permet de travailler avec de véritables images de systèmes (ex. : pfSense, Linux, etc.), offrant une simulation plus proche d'un environnement de production.

Critère	**Cisco Packet Tracer**	**GNS3 (Graphical Network Simulator 3)**
Type d'outil	Simulateur pédagogique	Émulateur réseau avancé
Interface utilisateur	Simple, visuelle, adaptée aux débutants	Complexe, orientée professionnel et réseau réel
Objets IoT intégrés	Oui (capteurs, actionneurs, scripts IoT)	Non directement (nécessite intégration manuelle)
Simulation de protocoles IoT	Basique (MQTT non supporté)	Possible via installation de brokers (ex. EMQX)
Fonctionnalités réseau	Limitées (routage, ACLs simples)	Complètes (pfSense, VLAN, DMZ, sniffing, etc.)
Firewall / IDS / IPS	Non supportés	Oui (pfSense, Snort, Suricata, etc.)
TLS / SSL, certificats	Non supportés	Oui, via systèmes réels
Performance et réalisme	Faible réalisme	Haute fidélité (réseaux réels)
Utilisation de systèmes réels	Non (systèmes fictifs Cisco)	Oui (machines virtuelles, OS réels)

TABLE 3.1 – Tableau comparatif : Cisco Packet Tracer vs GNS3

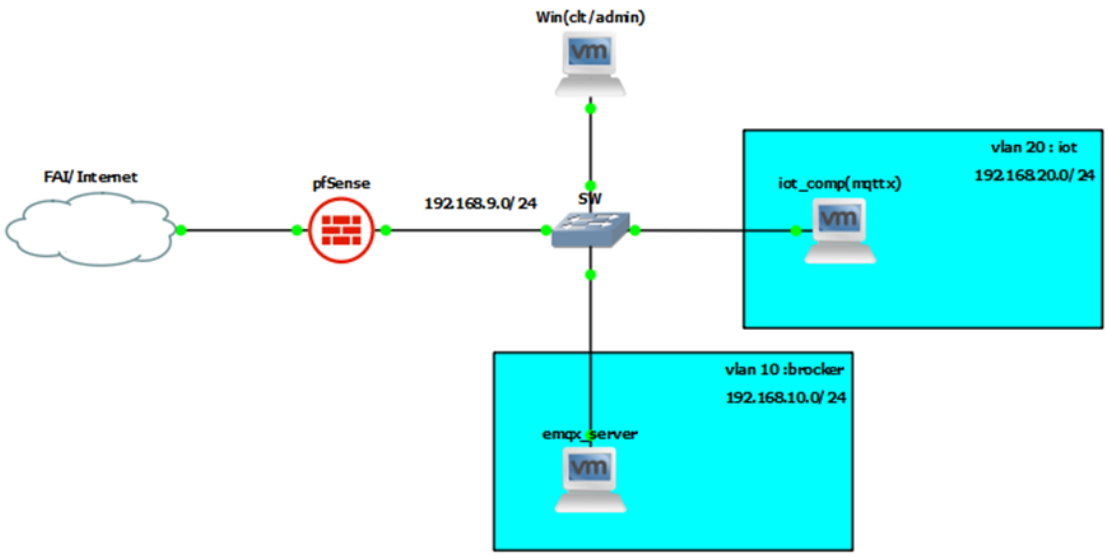

FIGURE 3.1 – Topologie utilisée sur gns3

3.2.3 Description de la nouvelle topologie

La topologie illustrée montre un réseau sécurisé pour une maison intelligente. Elle inclut :

- **pfSense** : Agit comme pare-feu et routeur, connecté à l'Internet (FAI/Internet) via une interface WAN et protégeant les VLAN internes.
- **Switch (SW)** : Gère la segmentation en VLAN pour isoler les trafics.
- **VLAN 10 (broker)** : Contient un serveur employé (`emp_server`) hébergeant le broker MQTT (EMQX) pour la communication IoT.
- **VLAN 20 (iot)** : Inclut un client IoT simulé (`mqttx`) pour interagir avec le broker.
- **Win (admin)** : Une machine d'administration connectée au switch pour gérer le réseau.
- **Sécurité** : Le pare-feu pfSense applique des règles pour bloquer les accès non autorisés et segmenter les VLAN.

Le schéma fourni reflète cette architecture, avec une séparation claire entre les zones (Internet, administration, IoT, broker) et l'utilisation de VLAN pour une isolation renforcée.

3.2.4 Outils utilisés dans la simulation avancée

pfSense est un pare-feu open source basé sur FreeBSD, intégré dans la simulation sous forme de machine virtuelle. Il assure plusieurs fonctions de sécurité critiques : filtrage de paquets, traduction d'adresses réseau (NAT), routage inter-VLAN, et possibilité d'ajouter des services complémentaires comme **Snort** pour la détection d'intrusion. pfSense joue le rôle de passerelle principale entre le réseau interne et l'extérieur (Internet), tout en contrôlant finement les flux autorisés entre les VLANs.

EMQX (Erlang MQTT Broker) est le serveur MQTT choisi pour cette architecture. Ce broker open source, compatible avec les standards industriels, est utilisé pour gérer les échanges de messages entre les objets connectés. EMQX permet la mise en place de la sécurité au niveau de la communication grâce à la prise en charge du chiffrement TLS/SSL, à l'authentification des clients, et à la gestion des autorisations d'accès aux topics. Il est déployé dans un VLAN isolé (VLAN 10), servant de zone tampon pour limiter les risques d'intrusion.

MQTTX est un client MQTT graphique utilisé pour simuler le comportement des objets connectés. Il permet d'établir des connexions avec le broker EMQX, de publier ou de s'abonner à des topics, et de tester les mécanismes de sécurité comme l'authentification et le chiffrement. Il est exécuté depuis une machine virtuelle située dans un VLAN spécifique aux objets IoT (VLAN 20), renforçant ainsi la séparation des rôles dans le réseau.

3.3 Sécurisation réseau avec pfSense

3.3.1 Présentation de pfSense

pfSense est une distribution open source basée sur FreeBSD, largement utilisée pour la mise en œuvre de pare-feu, de routeurs et de services de sécurité réseau. Elle est connue pour sa robustesse, sa fiabilité et sa richesse fonctionnelle. pfSense dispose d'une interface Web intuitive qui permet de configurer facilement des règles de filtrage, de traductions d'adresses (NAT), de VLANs, de VPNs, ainsi que des fonctionnalités avancées comme la redondance (CARP), la détection d'intrusion (Snort), ou la surveillance du trafic.

Les fonctions principales de pfSense utilisées dans ce projet sont :

— Pare-feu à état (*stateful firewall*) avec filtrage par IP, protocole, port et interface.

— NAT (*Network Address Translation*), permettant aux machines internes d'accéder à Internet tout en restant masquées.

— Routage inter-VLAN, grâce à l'attribution de sous-réseaux distincts aux VLANs créés dans GNS3.

— Gestion centralisée des règles, permettant d'appliquer une politique de sécurité stricte et cohérente sur toutes les interfaces.

3.3.2 Déploiement de pfSense dans GNS3

Le déploiement de pfSense dans GNS3 commence par la configuration des interfaces réseau, essentielles pour connecter le pare-feu au reste de la topologie. Une interface WAN est assignée pour la connexion à l'Internet (FAI), tandis que les interfaces LAN sont utilisées pour gérer les segments internes, notamment les

VLAN dédiés aux dispositifs IoT et au broker. Les règles de sécurité sont ensuite ajoutées via l'interface web de pfSense.

Configuration des interfaces LAN/WAN

L'interface WAN de pfSense est connectée au cloud GNS3, lequel est relié à l'interface réseau de l'hôte via NAT VMware. Cette configuration permet à la machine pfSense d'accéder à Internet tout en conservant une isolation du réseau local simulé :

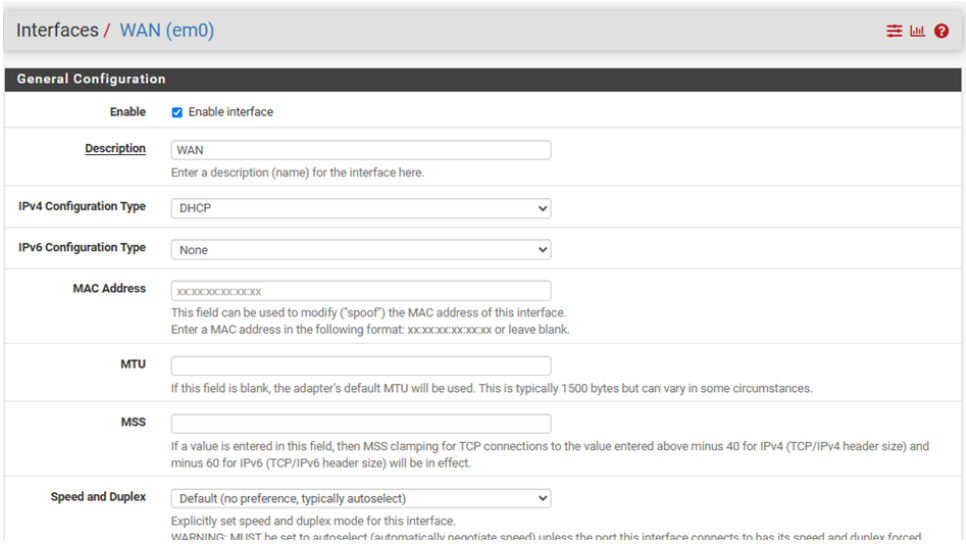

FIGURE 3.2 – Configuration de l'interface WAN

L'interface LAN est configurée avec une adresse IP statique (192.168.9.1/24), offrant une base solide pour la gestion des sous-réseaux internes et la segmentation réseau via les VLANs.

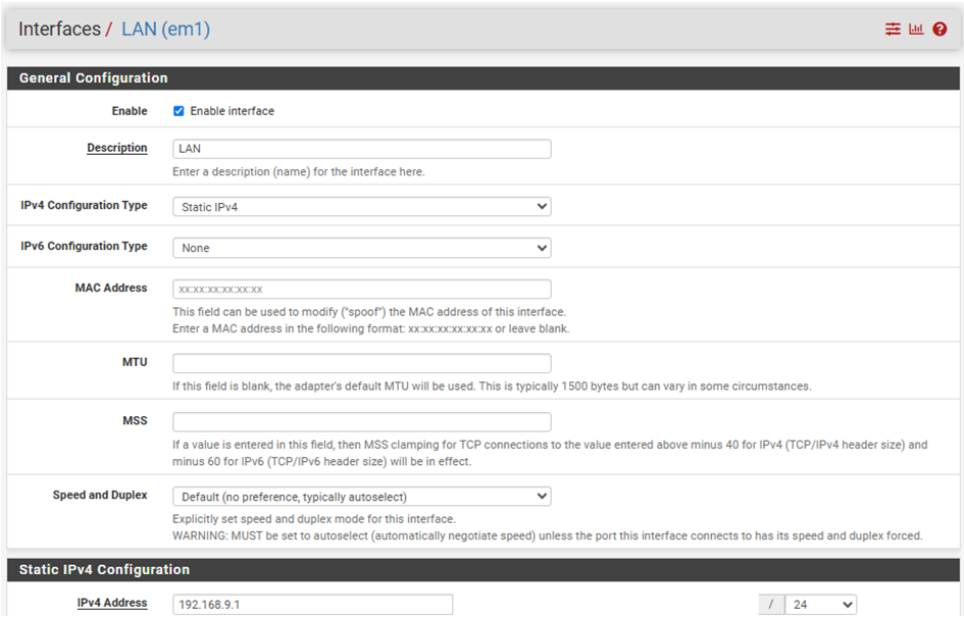

FIGURE 3.3 – Configuration de l'interface LAN

Configuration des VLAN

La configuration des VLANs commence par leur mise en place sur le switch virtuel intégré à GNS3 :

— **Création des VLANs** : Les VLANs BROKER et IOT sont créés dans l'interface de gestion du switch pour définir des segments distincts.

```
SW(config)#vlan 10
SW(config-vlan)#name broker
SW(config-vlan)#ex
SW(config)#vlan 20
SW(config-vlan)#name iot
SW(config-vlan)#ex
```

FIGURE 3.4 – Création des Vlans sur Switch

— **Assignation des ports aux VLANs** : Les ports sont configurés en mode access, attribuant des appareils spécifiques (comme des capteurs IoT ou le broker) à chaque VLAN.

```
SW(config)#interface range ethernet 3/0-3
SW(config-if-range)#switchport mode access
SW(config-if-range)#switchport access vlan 20
SW(config-if-range)#ex
```

FIGURE 3.5 – Assignation des ports aux Vlans(Access)

— **Configuration d'un port trunk** : Un port trunk est établi vers pfSense pour acheminer le trafic inter-VLAN, reliant les différents segments au pare-feu.

```
SW(config)#interface ethernet 0/0
SW(config-if)#switchport trunk encapsulation dot1q
SW(config-if)#switchport mode trunk
SW(config-if)#
*Jun 13 23:32:28.761: %LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0/0, changed state to down
SW(config-if)#
*Jun 13 23:32:31.772: %LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0/0, changed state to up
SW(config-if)#switchport trunk allowed vlan 10,20
SW(config-if)#ex
```

FIGURE 3.6 – Assignation des ports aux Vlans(Trunk)

Sur pfSense, les VLANs sont ensuite configurés sur l'interface physique `em1` (LAN) :

— **Création des interfaces virtuelles** : Des interfaces virtuelles sont créées et associées aux VLANs BROKER et IOT.

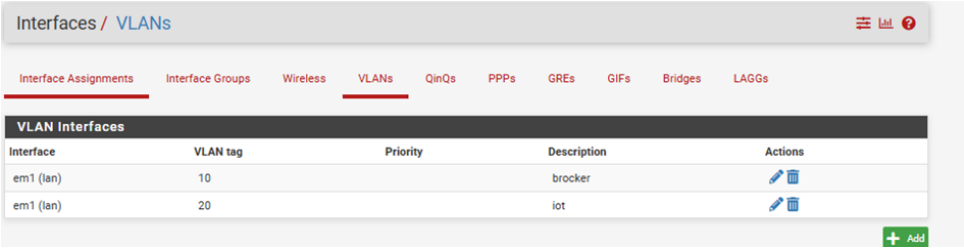

FIGURE 3.7 – Création des Vlans sur pfsense

— **Ajout des VLANs comme interfaces logiques** : Ces VLANs sont intégrés comme des interfaces logiques distinctes dans pfSense.

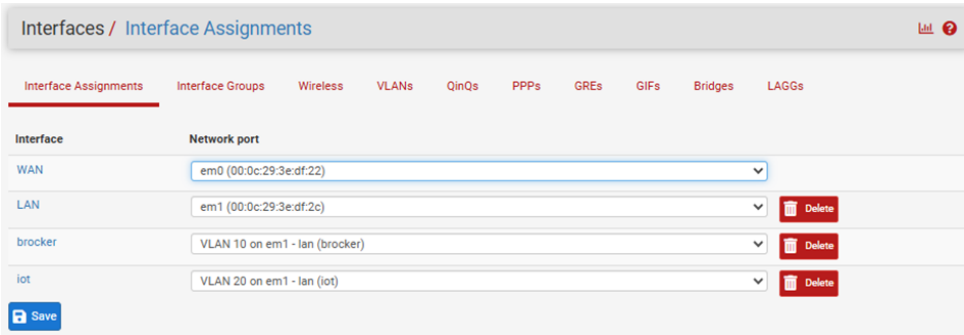

FIGURE 3.8 – l'ajout des VLANs sur pfsense

— **Activation et affectation des IP** : Les interfaces sont activées, et des adresses IP sont attribuées (par exemple, 192.168.10.1 pour BROKER et 192.168.20.1 pour IoT), assurant une identification réseau unique pour chaque segment.

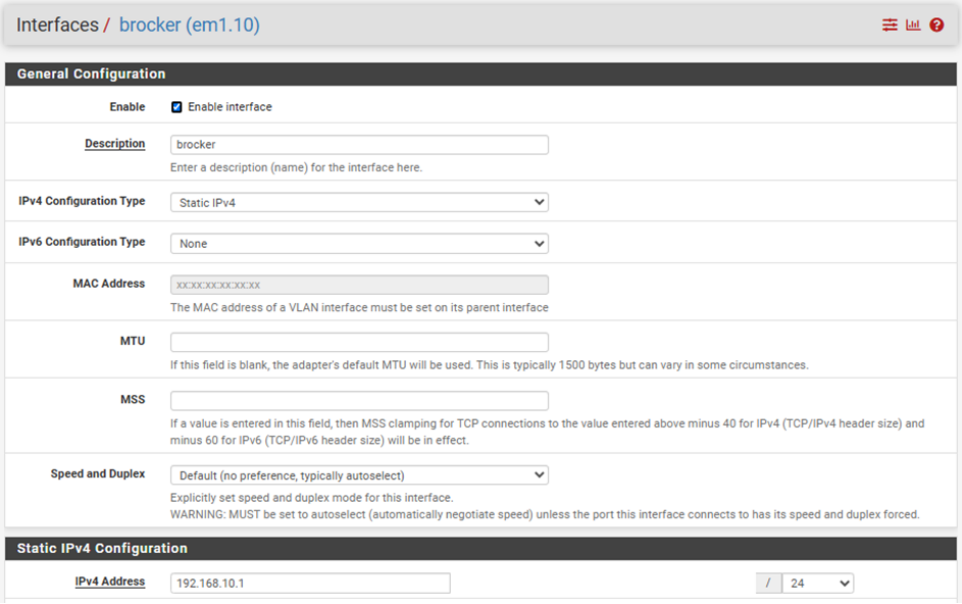

FIGURE 3.9 – Affectation des IP au VLAN broker

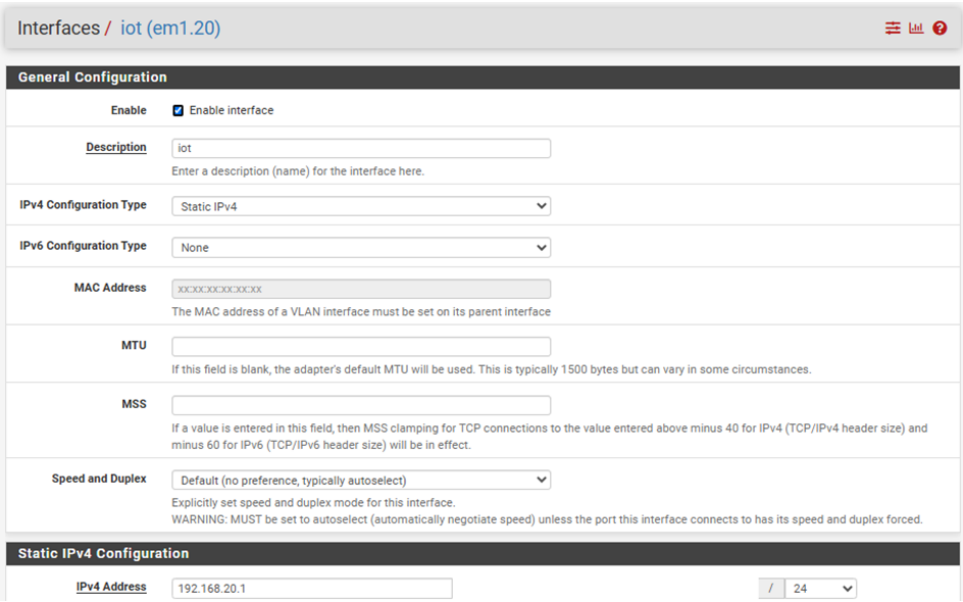

FIGURE 3.10 – Affectation des IP au VLAN IoT

Configuration DHCP

Des serveurs DHCP sont configurés pour automatiser l'attribution des adresses IP :

— **LAN** : Un serveur DHCP est mis en place pour le sous-réseau LAN, distribuant des adresses dans la plage 192.168.9.100–192.168.9.200.

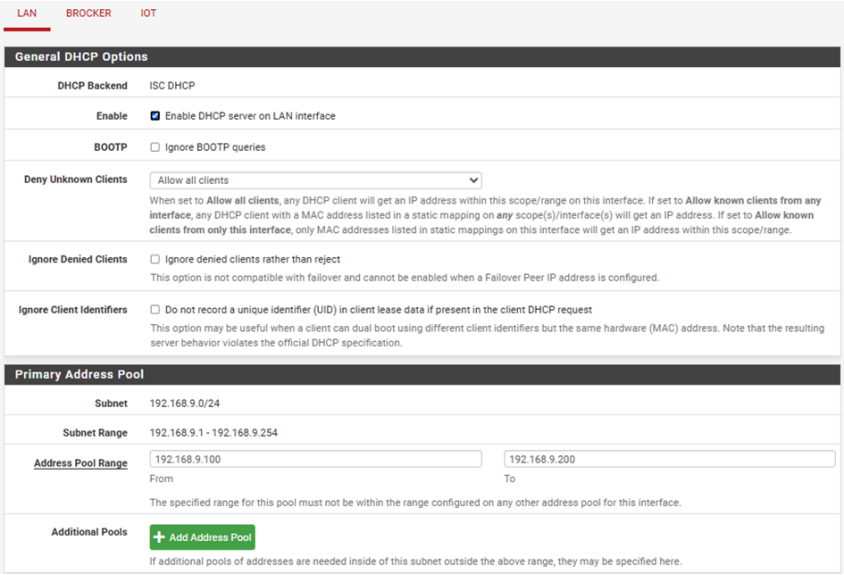

FIGURE 3.11 – Configuration de DHCP sur LAN

— **IoT** : Un serveur DHCP est activé pour le VLAN IOT, attribuant des adresses dans la plage 192.168.11.x, facilitant la connexion des dispositifs IoT

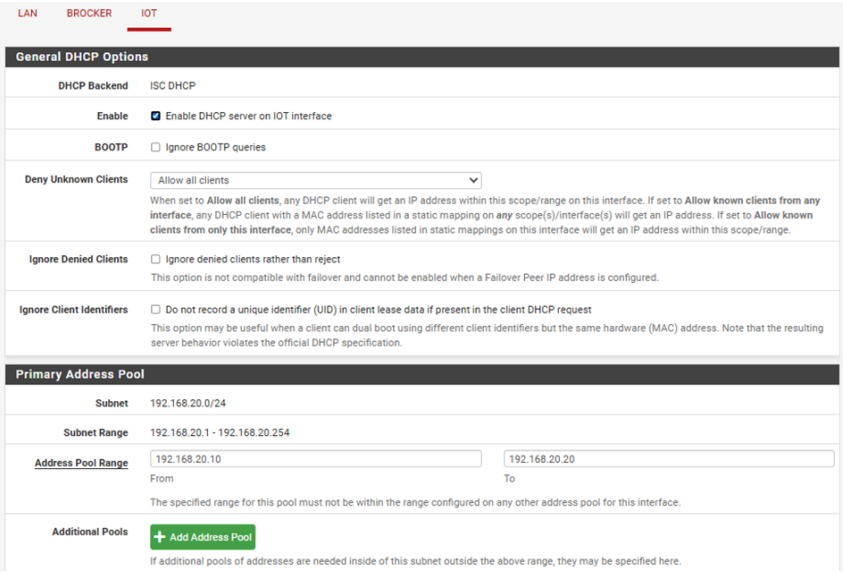

FIGURE 3.12 – Configuration de DHCP sur Vlan IoT

— **BROKER** : Un serveur DHCP spécifique est configuré pour le VLAN BROKER, assignant des adresses dans la plage 192.168.10.10 – 192.168.10.20

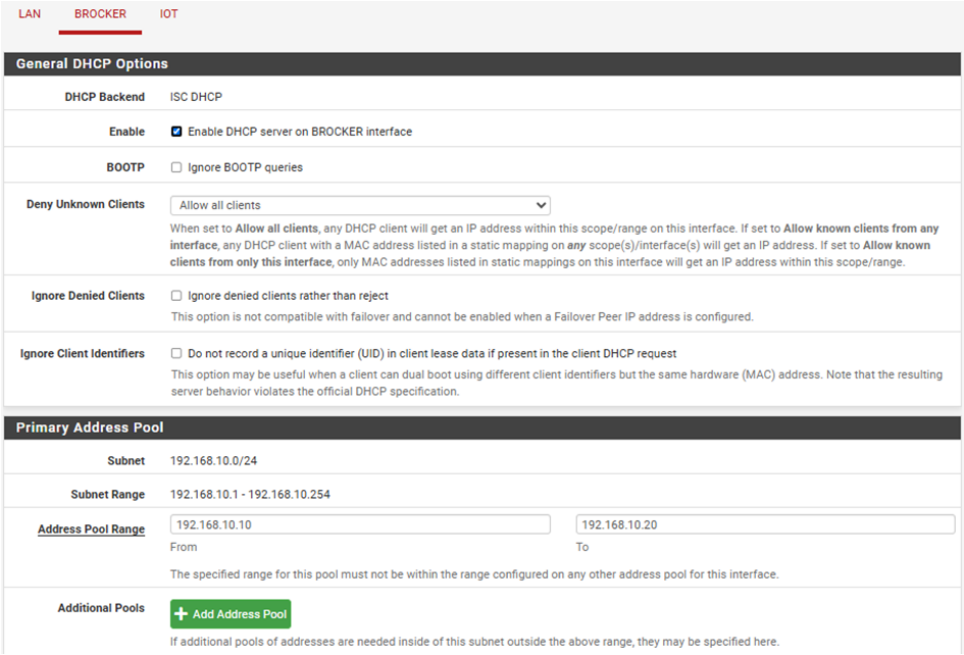

FIGURE 3.13 – Configuration de DHCP sur Vlan broker

Configuration des règles du pare-feu

Avant d'appliquer les règles, une adresse IP spécifique (192.168.9.9) est attribuée à la machine admin (mod_dar)

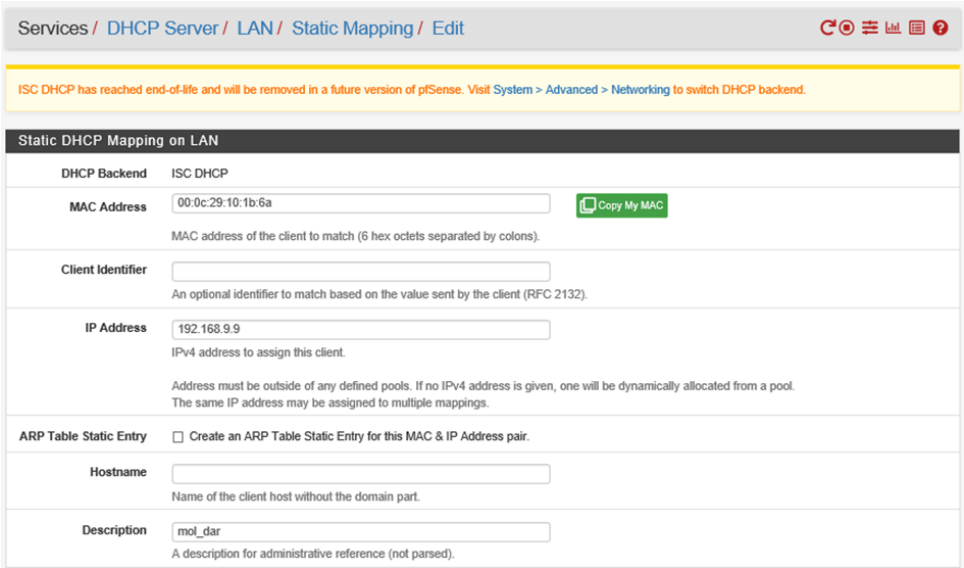

FIGURE 3.14 – Attribution d'une adresse IP spécifique à la machine admin

et un alias est créé pour cette machine dans pfSense afin de la reconnaître facilement

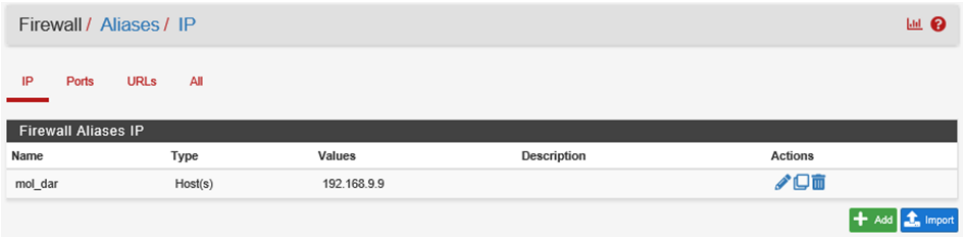

FIGURE 3.15 – Création d'un Alias

La configuration des règles sur l'interface LAN de pfSense vise à assurer à la fois l'accès nécessaire pour l'administration et la sécurité du réseau. Une règle spéciale (Anti-Lockout) garantit que l'accès à l'interface Web et SSH de pfSense reste toujours possible depuis le LAN. L'administrateur (mol_dar) bénéficie d'un accès complet à tous les sous-réseaux, ce qui facilite la gestion du réseau. Ensuite, une règle de sécurité interdit explicitement aux machines du LAN d'accéder à l'interface pfSense elle-même, limitant ainsi les risques d'attaques internes. De plus, le trafic du LAN vers les VLANs "BROCKER" et "IOT" est bloqué pour assurer l'isolation entre les réseaux. Enfin, une règle par défaut permet au reste du trafic LAN de sortir librement vers Internet, tout en maintenant les restrictions imposées par les règles précédentes.

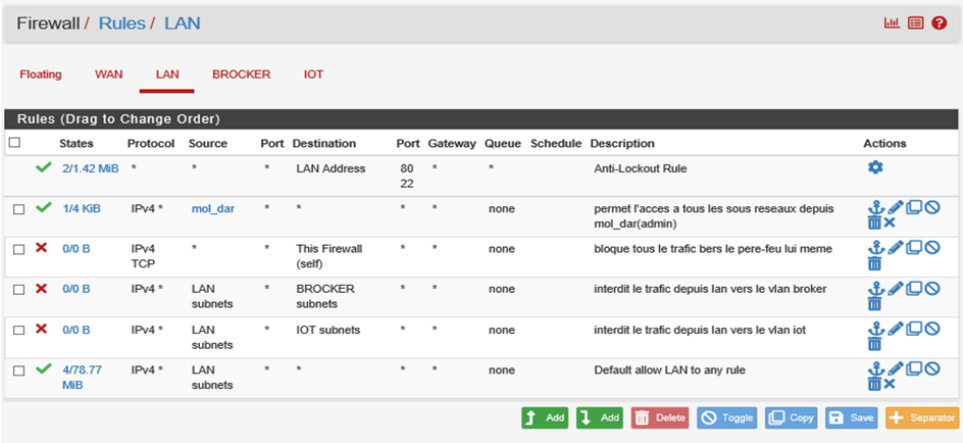

FIGURE 3.16 – Définition des régles sur LAN

La configuration du VLAN BROCKER sur pfSense accorde un accès complet à l'administrateur (mol_dar), tout en appliquant un contrôle strict sur le trafic sortant. Deux règles précises autorisent la communication vers le VLAN IOT uniquement sur les ports 1883 et 8883, nécessaires au fonctionnement du

protocole MQTT, couramment utilisé dans les environnements IoT. Toute autre tentative de connexion depuis BROCKER vers d'autres réseaux ou vers lui-même est bloquée par une règle de refus générale, garantissant ainsi une isolation rigoureuse et sécurisée de ce VLAN

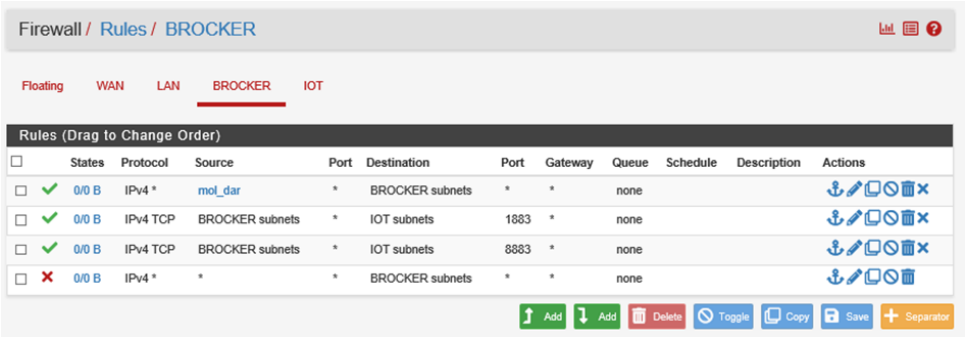

FIGURE 3.17 – Définition des régles sur Vlan Broker

configuratpion des règles de pare-feu pour l'interface IOT dans pfSense.Elle inclut une règle autorisant tout trafic IPv4 depuis mod_dar vers les sous-réseaux IOT, ainsi que deux règles permettant le trafic TCP IPv4 des sous-réseaux IOT vers les sous-réseaux BROKER sur les ports 1883 et 8883, essentiels pour le protocole MQTT. Ces règles, toutes activées, assurent une communication sécurisée entre les dispositifs IOT et le broker, tout en offrant un accès administratif contrôlé depuis mod_dar, garantissant une gestion efficace du réseau IoT.

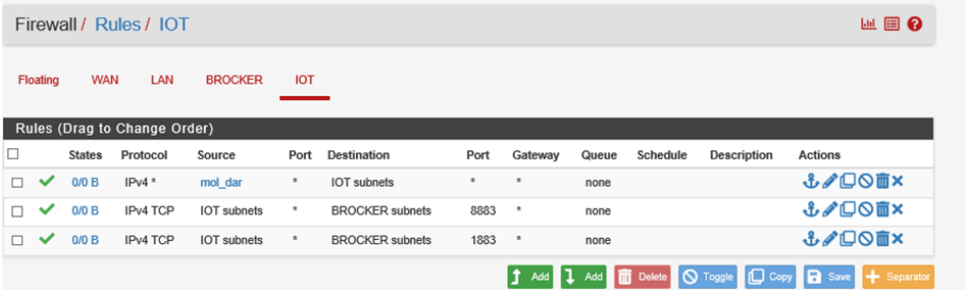

FIGURE 3.18 – Définition des régles sur vlan IoT

3.3.3 Résultats des tests avec pfSense

Les tests effectués avec pfSense ont permis d'évaluer la configuration réseau. Le test de connectivité depuis l'administrateur a été un succès, confirmant que l'accès depuis la machine `mol_dar` (192.168.9.9) aux sous-réseaux LAN, BROKER, et IOT est pleinement opérationnel.

```
C:\Users\Administrateur>ping 192.168.10.10

Envoi d'une requête 'Ping'  192.168.10.10 avec 32 octets de données :
Réponse de 192.168.10.10 : octets=32 temps=5 ms TTL=63
Réponse de 192.168.10.10 : octets=32 temps=6 ms TTL=63
Réponse de 192.168.10.10 : octets=32 temps=7 ms TTL=63
Réponse de 192.168.10.10 : octets=32 temps=7 ms TTL=63

Statistiques Ping pour 192.168.10.10:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 5ms, Maximum = 7ms, Moyenne = 6ms
```

FIGURE 3.19 – Test d'un ping depuis l'admin vers le broker (vlan10)

```
C:\Users\Administrateur>ping 192.168.20.11

Envoi d'une requête 'Ping'  192.168.20.11 avec 32 octets de données :
Réponse de 192.168.20.11 : octets=32 temps=6 ms TTL=63
Réponse de 192.168.20.11 : octets=32 temps=9 ms TTL=63
Réponse de 192.168.20.11 : octets=32 temps=9 ms TTL=63
Réponse de 192.168.20.11 : octets=32 temps=7 ms TTL=63

Statistiques Ping pour 192.168.20.11:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 6ms, Maximum = 9ms, Moyenne = 7ms
```

FIGURE 3.20 – Test d'un ping depuis l'admin vers vlan 20

```
C:\Users\Administrateur>ping 15.15.15.130

Envoi d'une requête 'Ping'  15.15.15.130 avec 32 octets de données :
Réponse de 15.15.15.130 : octets=32 temps=2 ms TTL=64
Réponse de 15.15.15.130 : octets=32 temps=5 ms TTL=64
Réponse de 15.15.15.130 : octets=32 temps=5 ms TTL=64
Réponse de 15.15.15.130 : octets=32 temps=4 ms TTL=64

Statistiques Ping pour 15.15.15.130:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 2ms, Maximum = 5ms, Moyenne = 4ms
```

FIGURE 3.21 – Test d'un ping depuis l'admin vers l'exterieure

En revanche, le test de connectivité ICMP depuis les composants BROKER et IOT vers d'autres réseaux a été refusé, validant l'efficacité des règles de blocage qui isolent ces segments, conformément aux politiques de sécurité établies

```
┌──(lahoucine㉿Server)-[~]
└─$ ping 192.168.9.9
PING 192.168.9.9 (192.168.9.9) 56(84) bytes of data.


^C
--- 192.168.9.9 ping statistics ---
15 packets transmitted, 0 received, 100% packet loss, time 14326ms
```

FIGURE 3.22 – Test d'un ping depuis le Broker vers LAN

```
┌──(lahoucine㉿kali)-[~]
└─$ ping 192.168.10.10
PING 192.168.10.10 (192.168.10.10) 56(84) bytes of data.
^C
--- 192.168.10.10 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5108ms

┌──(lahoucine㉿kali)-[~]
└─$ ping 15.15.15.130
PING 15.15.15.130 (15.15.15.130) 56(84) bytes of data.
^C
--- 15.15.15.130 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5108ms
```

FIGURE 3.23 – Test d'un ping depuis un composant IoT vers le Broker et l'exterieure

3.4 Mise en place du broker MQTT sécurisé avec EMQX

3.4.1 Pourquoi MQTT dans une maison intelligente ?

Le protocole MQTT (Message Queuing Telemetry Transport) s'impose comme une solution idéale pour les communications dans les environnements de maison intelligente. Conçu pour les réseaux à faible bande passante et pour les dispositifs embarqués, il offre une communication légère, rapide et en temps réel entre capteurs, contrôleurs et applications.

Son architecture basée sur un modèle publish/subscribe permet une grande flexibilité : les capteurs publient leurs données sur des "topics", tandis que les dispositifs ou applications intéressés s'y abonnent. Grâce à sa faible consommation de ressources et sa simplicité d'implémentation, MQTT est parfaitement adapté aux contraintes d'un réseau domestique IoT.

3.4.2 Installation et configuration de EMQX

Le broker EMQX (Erlang MQTT Broker) a été choisi pour ce projet en raison de sa compatibilité avec MQTT v3.1.1 et v5.0, sa gestion fine de la sécurité et sa capacité à fonctionner dans des environnements virtualisés.

Dans GNS3, EMQX a été déployé sur une machine virtuelle Linux connectée au VLAN réservé au "broker" (VLAN 10). Après l'installation du paquet EMQX via les dépôts officiels, une configuration de base a été effectuée pour permettre la communication sur les ports standards :

— Port 1883 : communication MQTT non sécurisée (désactivée par la suite).

— Port 8883 : communication MQTT sécurisée via TLS.

L'interface Web d'administration (dashboard) d'EMQX, accessible via le port 18083, a permis de gérer facilement les utilisateurs, les règles de sécurité et la surveillance des connexions.

Les commandes pour l'installation et l'activation :

```
wget https://www.emqx.com/en/downloads/broker/5.7.0/emqx-5.7.0-
    ubuntu22.04-amd64.deb
sudo dpkg -i emqx-5.7.0-ubuntu22.04-amd64.deb
sudo systemctl start emqx
sudo systemctl enable emqx
```

Vérification :

FIGURE 3.24 – Status de serveur EMQX

L'interface graphique de EMQX :

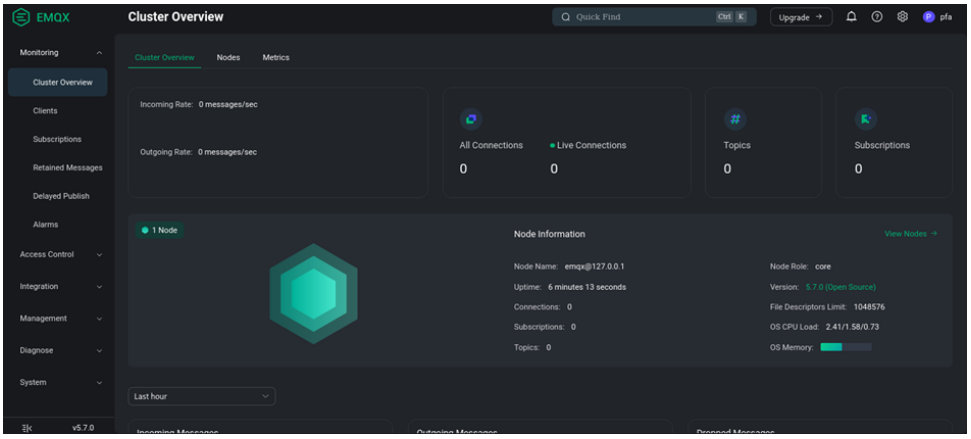

FIGURE 3.25 – Dashboard de EMQX

3.4.3 Sécurisation des communications avec TLS/SSL

Afin de garantir la confidentialité et l'intégrité des données échangées entre les dispositifs IoT et le broker MQTT, un chiffrement des communications via TLS/SSL a été mis en place. Cette sécurisation s'est déroulée en deux grandes étapes : la génération des certificats puis leur intégration dans EMQX.

Étape 1 : Génération des certificats

La première étape consiste à créer un jeu de certificats permettant de chiffrer les échanges entre clients et serveur MQTT. Cela a été fait à l'aide de l'outil OpenSSL, directement depuis une machine Linux.

— **Clé de l'autorité de certification (CA)** :

```
┌──(lahoucine㉿Server)-[~/emqx-tls]
└─$ openssl genrsa -out ca-key.pem 2048
```

FIGURE 3.26 – Génération de la clé de l'autorité du certificat

— **Certificat de l'autorité de certification** :

```
┌──(lahoucine㉿Server)-[~/emqx-tls]
└─$ openssl req -x509 -new -nodes -key ca-key.pem -sha256 -days 365 \
 -out ca-cert.pem -subj "/C=MA/ST=MAROC/L=BEKANE/O=ENIAD/CN=EMQX_CA"
```

FIGURE 3.27 – Certificat de l'autorité de certification.

— **Clé privée du serveur** :

```
┌──(lahoucine㉿Server)-[~/emqx-tls]
└─$ openssl genrsa -out server-key.pem 2048
```

FIGURE 3.28 – Génération de la Clé privée du serveur

— **CSR du serveur** :

```
┌──(lahoucine㉿Server)-[~/emqx-tls]
└─$ openssl req -new -key server-key.pem -out server.csr \
 -subj "/C=MA/ST=MAROC/L=BERKANE/O=ENIAD/CN=localhost"
```

FIGURE 3.29 – Génération de CSR du serveur

— **Fichier SAN (SubjectAltName)** :

```
┌──(lahoucine㉿Server)-[~/emqx-tls]
└─$ cat > san.cnf <<EOF
subjectAltName=DNS:localhost,IP:127.0.0.1
EOF
```

FIGURE 3.30 – Fichier SAN (SubjectAltName)

— **Certificat du serveur signé** :

```
┌──(lahoucine㉿Server)-[~/emqx-tls]
└─$ openssl x509 -req -in server.csr -CA ca-cert.pem -CAkey ca-key.pem -CAcreateserial \
 -out server-cert.pem -days 365 -sha256 -extfile san.cnf

Certificate request self-signature ok
subject=C=MA, ST=MAROC, L=BERKANE, O=ENIAD, CN=localhost
```

FIGURE 3.31 – Certificat de serveur signé

Étape 2 : Importation des certificats dans EMQX et configuration du chiffrement TLS

Une fois les certificats générés, ils sont importés dans EMQX pour activer TLS. Les fichiers `ca-cert.pem`, `server-key.pem`, et `server-cert.pem` sont copiés dans le répertoire spécifié, généralement `/etc/emqx/certs` :

```
┌──(lahoucine㉿Server)-[~/emqx-tls]
└─$ sudo mkdir -p /etc/emqx/certs

┌──(lahoucine㉿Server)-[~/emqx-tls]
└─$ sudo cp *.pem /etc/emqx/certs
```

FIGURE 3.32 – Importation des certificats dans EMQX

Ensuite, la modification des paramètres se fait dans le fichier de configuration emqx.conf. Il faut :

— **Ajouter le listener sécurisé : Un nouveau listener sécurisé est configuré, comme illustré dans la capture d'écran** :

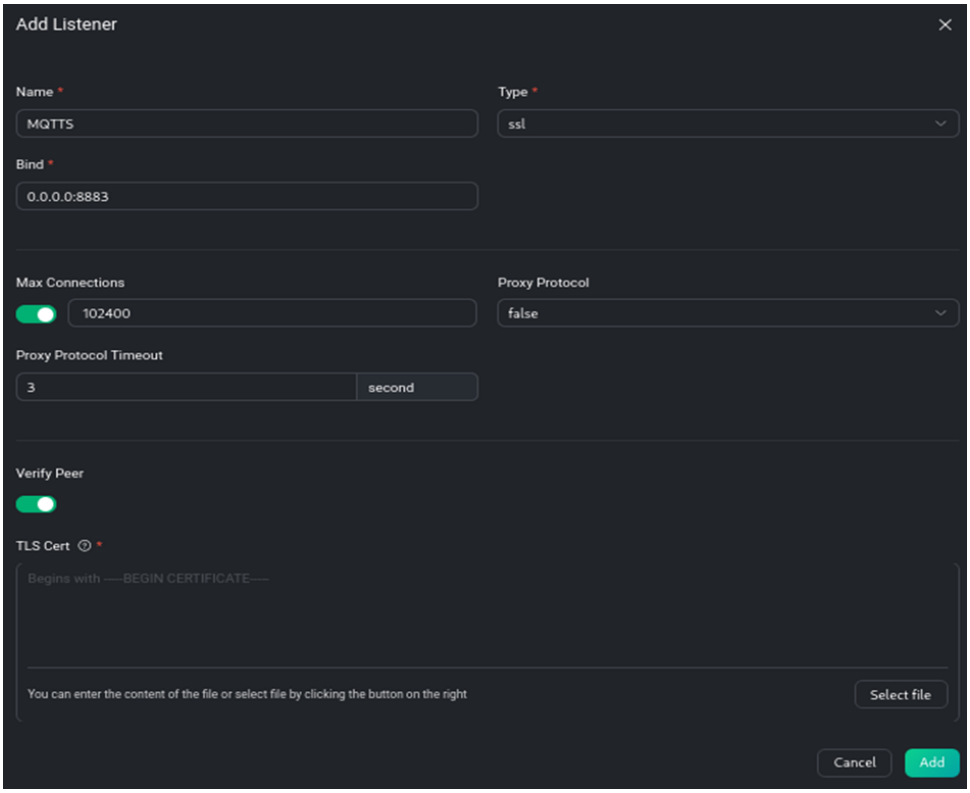

FIGURE 3.33 – Configuration du chiffrement TLS

— **Importation des certifications sur l'interface web : :**

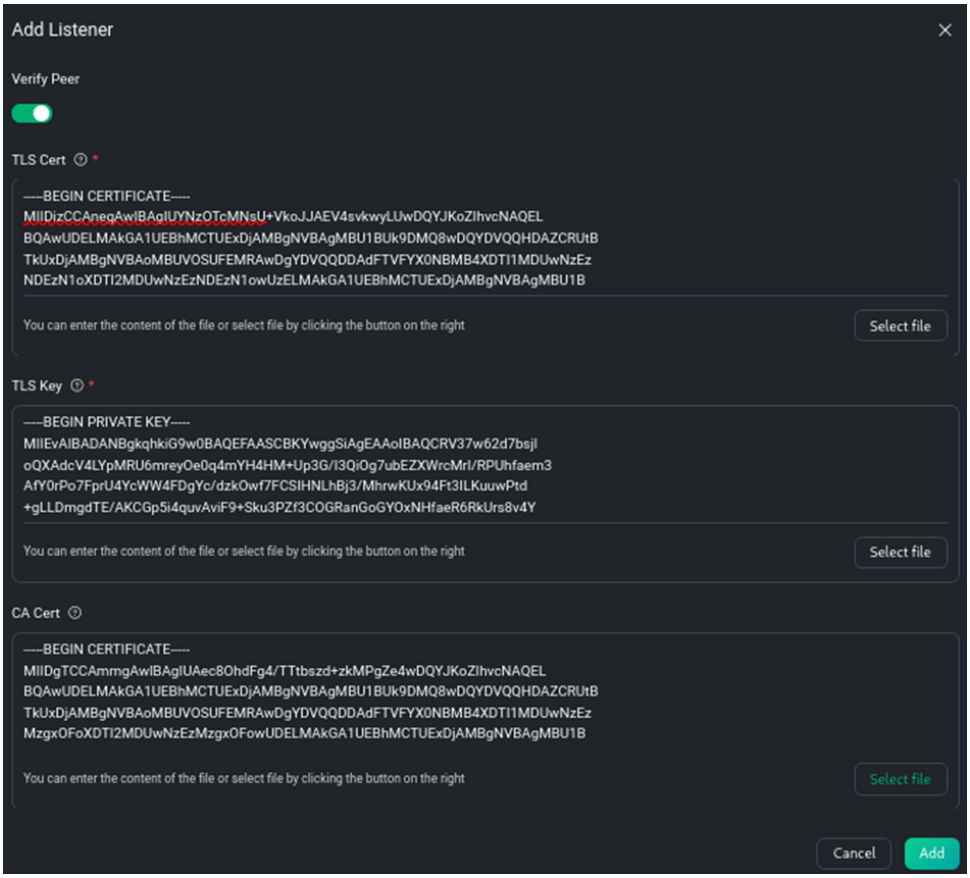

FIGURE 3.34 – Importation des certificats

— **Activer le port sécurisé : Une fois les certificats importés, le port sécurisé 8883 est activé en sauvegardant la configuration via le bouton "Add" : :**

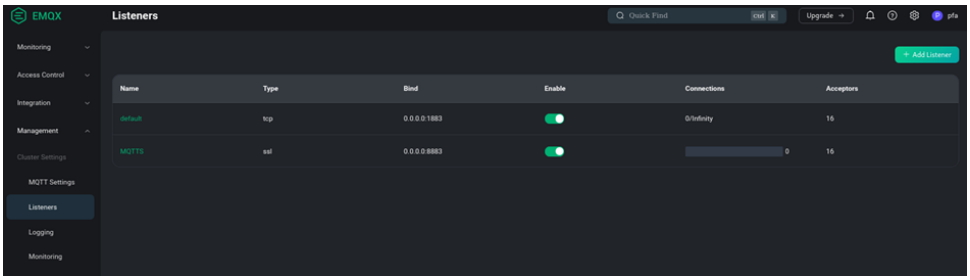

FIGURE 3.35 – Activation du port sécurisé

3.4.4 Authentification mutuelle par certificats

Afin de compléter la sécurité offerte par le chiffrement TLS/SSL, EMQX intègre plusieurs mécanismes d'authentification et de contrôle d'accès, permettant de renforcer la protection contre les connexions non autorisées ou les abus au niveau applicatif. Dans notre projet on utilise la méthode classique basée sur nom d'utilisateur et mot de passe. Les identifiants sont définis via l'interface Web et sont requis pour chaque client souhaitant établir une session MQTT.

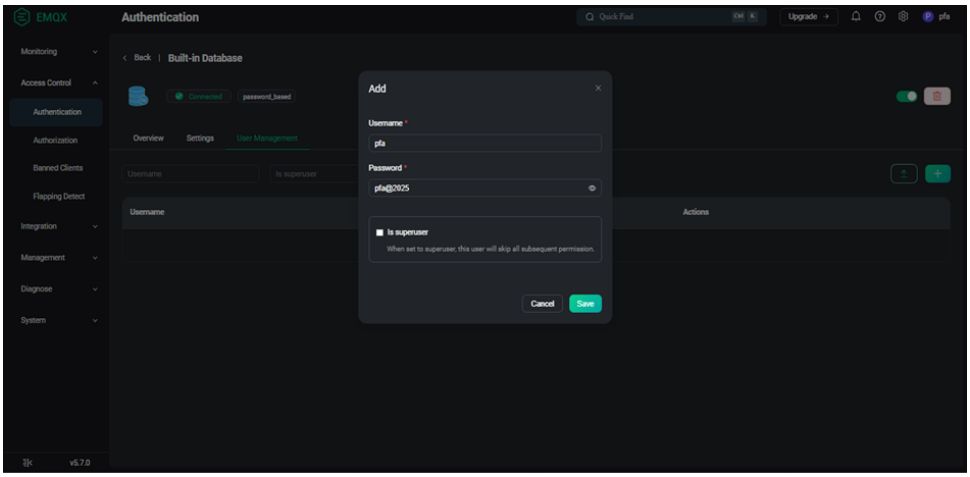

FIGURE 3.36 – Authentification et contrôle d'accès

En parallèle, EMQX permet une authentification mutuelle par certificats numériques (mutual TLS), dans laquelle le client ne se contente pas de valider l'identité du serveur, mais doit également présenter son propre certificat au broker. Ce mécanisme assure une authentification bidirectionnelle, renforçant considérablement la sécurité dans des environnements sensibles. Pour activer cette fonctionnalité, il est nécessaire de configurer le paramètre force<_verify_peer à true dans la section TLS du fichier de configuration ou via l'interface Web. Ainsi, le broker exigera que tout client souhaitant se connecter via TLS présente un certificat valide, signé par une autorité reconnue.

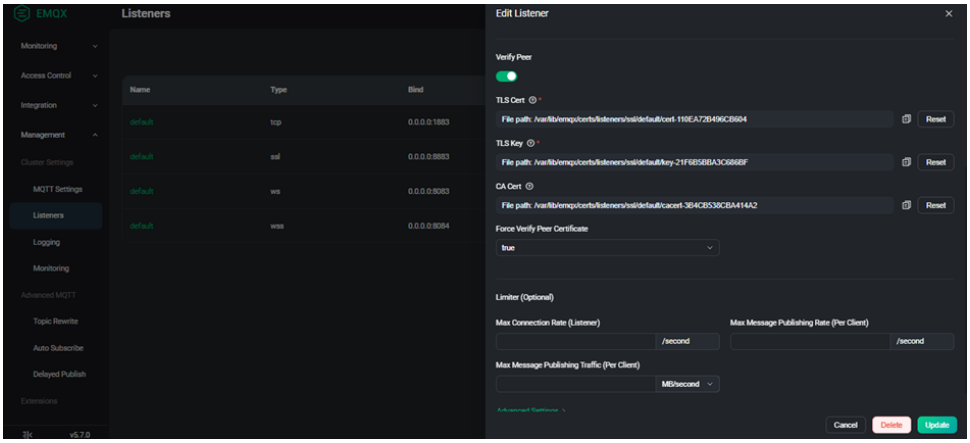

FIGURE 3.37 – Les clés TLS avec CA généré

3.4.5 Tests et validation

Avant de procéder aux tests de connexion, il est nécessaire de générer un certificat pour le client afin d'assurer une authentification mutuelle conforme à la configuration TLS avec "Force Verify Peer Certificate". Le processus débute sur le côté client (machine iot_comp) avec la génération du certificat CSR (Certificate Signing Request) à l'aide d'OpenSSL.

```
┌──(lahoucine㉿kali)-[~/emqx/clt_certs]
└─$ openssl genrsa -out client-key.pem 2048

┌──(lahoucine㉿kali)-[~/emqx/clt_certs]
└─$ openssl req -new -key client-key.pem -out client.csr \
 -subj "/C=MA/ST=MAROC/L=BERKANE/O=ENIAD/CN=clt"
```

FIGURE 3.38 – Génération d'un certificat pour le client

Ensuite, après la génération, le fichier client.csr doit être transféré vers le serveur EMQX, qui agit également comme une autorité de certification (CA) dans ce scénario. Sur le serveur, le CSR est signé avec la clé CA existante (ca-key.pem et ca-cert.pem) , produisant ainsi un certificat client signé (client-cert.pem).

```
┌──(lahoucine㉿Server)-[~/emqx-tls]
└─$ ls clts
client.csr

┌──(lahoucine㉿Server)-[~/emqx-tls]
└─$ openssl x509 -req -in clts/client.csr -CA ca-cert.pem -CAkey ca-key.pem \
  -CAcreateserial -out clts/client-cert.pem -days 365 -sha256
Certificate request self-signature ok
subject=C=MA, ST=MAROC, L=BERKANE, O=ENIAD, CN=clt

┌──(lahoucine㉿Server)-[~/emqx-tls]
└─$ ls clts
client-cert.pem  client.csr
```

FIGURE 3.39 – Signature de CSR par CA

Enfin, le certificat signé (client.crt) est renvoyé au client (iot_comp), où il est associé à la clé privée client.key pour établir une connexion sécurisée avec EMQX. Ce certificat, validé par la même CA utilisée pour le serveur, permet au client de répondre aux exigences de l'authentification mutuelle

Test et vérification :

Simulation de clients MQTT : des outils comme MQTTX ont permis de simuler des capteurs IoT et des applications mobiles. Les connexions ont été établies avec succès sur le port 8883, en utilisant les identifiants et certificats configurés. Sur la machine iot_comp (MQTTX) : Une connexion a été établie avec succès entre le client MQTTX et le serveur EMQX, utilisant les certificats client générés et les identifiants appropriés.

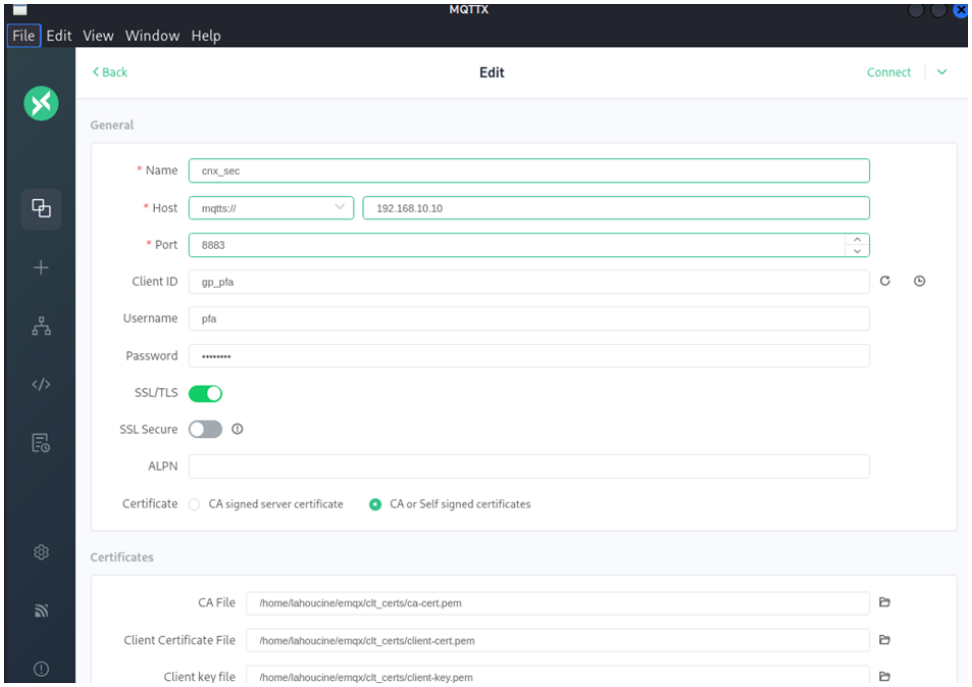

FIGURE 3.40 – Test de validation

Sur la machine emqx_server : La connexion initiée depuis le client iot_comp (MQTTX) a été acceptée avec succès, comme confirmé par les logs du serveur EMQX

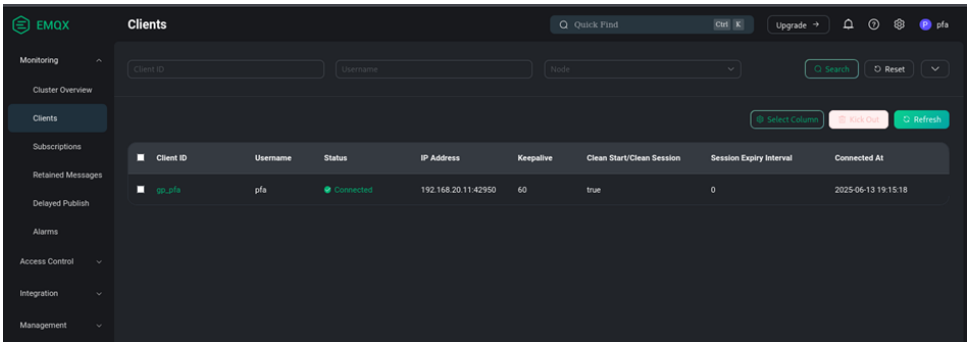

FIGURE 3.41 – Authentification réussie par utilisateur pfa

Client MQTT iot_comp (MQTTX) souscrivant et publiant : Le client iot_comp (MQTTX) s'est abonné au topic pfa_test, puis a publié le message « hello from client », vérifiant ainsi la fonctionnalité de publication dans un environnement sécurisé.

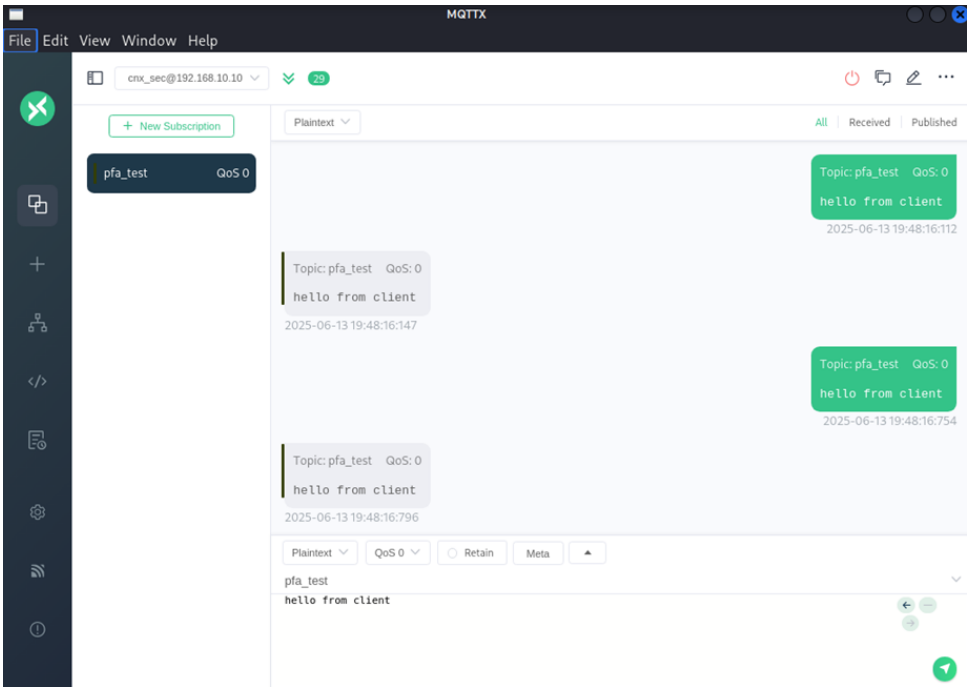

FIGURE 3.42 – La communication établie entre le client et le broker

Test sur le serveur avec MQTTX : Sur la machine emqx_server, un second instance de MQTTX a été lancée en tant que client. Après s'être abonné au même topic pfa_test, ce client a reçu le message « hello from client » envoyé par iot_comp (MQTTX), confirmant la transmission bidirectionnelle et la cohérence du système MQTT

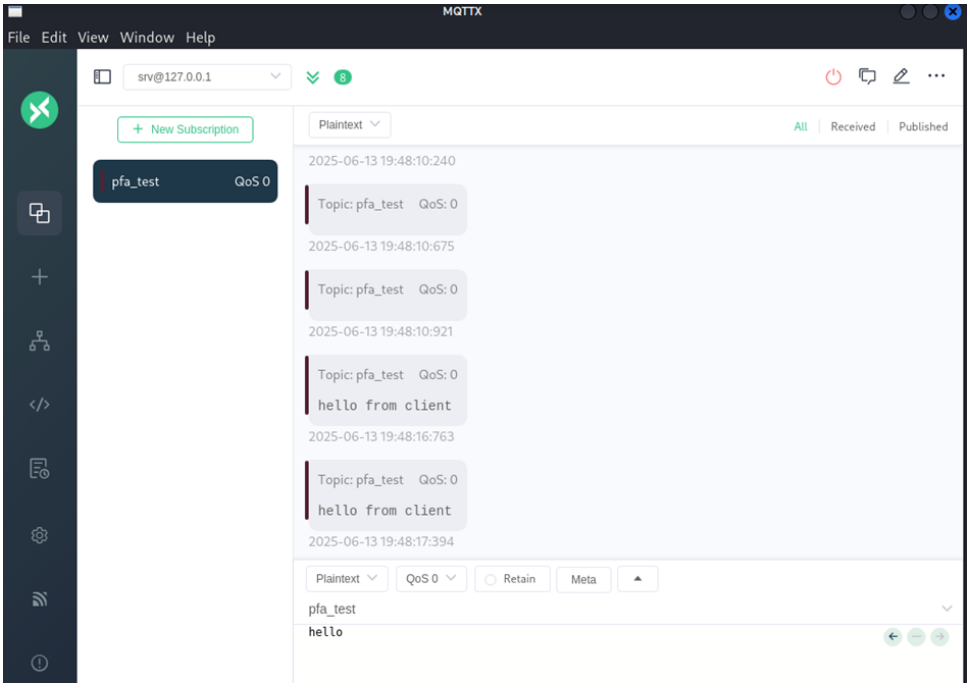

FIGURE 3.43 – La réception de message hello dans le broker

Finalement ,cette image illustre les abonnements effectués sur le topic 'pfa_test'

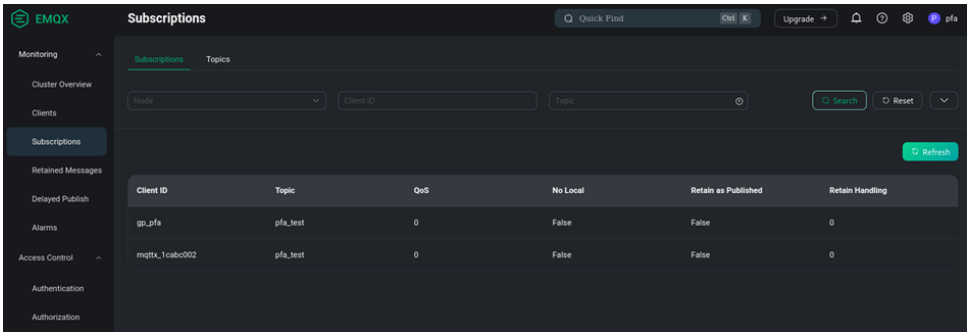

FIGURE 3.44 – Les utilisateurs qui peuvent contacter le broker

3.5 Conclusion

Ce chapitre a présenté la mise en place complète d'une infrastructure réseau sécurisée pour une maison intelligente, intégrant la segmentation VLAN, la configuration fine des règles de pare-feu pfSense, et le déploiement d'un broker MQTT performant et sécurisé avec EMQX.

La mise en œuvre du chiffrement TLS/SSL et de l'authentification mutuelle par certificats a permis d'assurer la confidentialité, l'intégrité, et l'authenticité des communications entre les dispositifs IoT et le broker. Les tests réalisés ont

démontré la réussite de la configuration, garantissant une isolation rigoureuse des réseaux, un contrôle d'accès adapté et une communication fiable et sécurisée.

Ainsi, cette architecture constitue une base solide pour le développement et la gestion sécurisée des environnements IoT dans le cadre des maisons intelligentes, répondant aux exigences actuelles de sécurité et de performance.

Conclusion générale

La maison intelligente représente aujourd'hui bien plus qu'un simple confort technologique. Elle incarne une nouvelle manière d'habiter, de gérer son environnement, et d'interagir avec les objets du quotidien. L'automatisation, le contrôle à distance et l'adaptabilité aux besoins des occupants redéfinissent profondément l'expérience domestique. Mais cette modernité, aussi séduisante soit-elle, doit impérativement s'accompagner d'une réflexion sérieuse autour de la sécurité.

Ce projet a permis de valider une architecture IoT sécurisée dans un contexte domestique, en s'appuyant sur une approche en deux phases.

Dans un premier temps, **Cisco Packet Tracer** a été utilisé pour modéliser une maison intelligente connectée, intégrer des objets IoT, concevoir une topologie réseau et appliquer des ACL simples pour segmenter le trafic et filtrer les accès.

Dans un second temps, la migration vers **GNS3** a permis d'apporter une couche de sécurité plus réaliste en intégrant le pare-feu **pfSense**, en configurant des règles avancées, en simulant des flux **MQTT** sécurisés via **EMQX**, et en mettant en place des certificats numériques **TLS/SSL**.

Ce travail a été l'occasion de mobiliser des compétences variées : configuration réseau, logique domotique, cybersécurité, gestion d'accès, ainsi que scripting et test. Sur le plan personnel, ce projet nous a permis de mieux comprendre l'interconnexion entre sécurité réseau et IoT, tout en développant une méthode de travail rigoureuse et structurée. Sur le plan technique, il a représenté une véritable mise en pratique des notions théoriques apprises dans le cadre académique.

Limites rencontrées

— L'environnement **Packet Tracer** reste limité en termes de sécurité avancée..
— L'ensemble du système n'a pas encore été testé dans un environnement physique réel.

Perspectives d'évolution

Plusieurs pistes pourraient enrichir ce travail à l'avenir :

— Intégration d'un **IDS/IPS** (*Snort* ou *Suricata*) pour la détection d'intrusion.

— Connexion à des services cloud (*AWS IoT*, *Azure IoT Hub*) pour la supervision à distance.
— Ajout de capteurs mobiles et gestion via smartphone/tablette.
— Utilisation d'algorithmes d'**IA** pour apprendre et adapter les comportements (température, présence…).
— Test d'une maquette réelle avec **Raspberry Pi** ou **ESP32** pour valider l'implémentation sur le terrain.

En définitive, ce projet montre qu'il est tout à fait envisageable de mettre en place une maison connectée efficace et sécurisée, à condition d'intégrer la cybersécurité dès la conception. L'innovation technologique n'est réellement durable que si elle s'accompagne d'une protection robuste et intelligente.

Bibliographie

— Article sur l'architecture IoT :
https://iotindustriel.com/iot-iiot/architecture-iot-lessentiel-a-savoir/

— Article sur la définition d'une maison intelligente :
https://www.hitachiclimat.fr/actualites/maison-intelligente-connectee-climatiseurs

— Article sur les principales vulnérabilités des appareils IoT :
https://www.fortinet.com/fr/resources/cyberglossary/iot-device-vulnerabilities

— Article publié sur le site de Cisco :
https://www.netacad.com/fr/cisco-packet-tracer

— Cour universitaire sur les listes d'accès (ACL) :
https://www.mediafire.com/file/upe76i4luva0ju3/Cour_S%C3%A9curit%C3%A9_des_r%C3%A9seaux_Partie_1.pdf/file

— Vidéo éducative sur YouTube pour la simulation de MQTT :
https://youtu.be/fT6L1LwRr8M?si=XIanJKeesbIWkC9D

— Article publié sur ce site pour GNS3 :
https://tcastus.github.io/TChelp/GNS3/Presentation.html

— Site officiel de pfSense pour son fonctionnement :
https://www.pfsense.org/

— Site officiel d'EMQX pour son fonctionnement :
https://www.emqx.com/en

— Articles sur la sécurité des maisons intelligentes :
https://www.globalsign.com/fr/blog/maison-intelligente-securite-iot

— Mémoire sur détection d'intrusions réaliste dans les maisons connectées :
https://theses.hal.science/tel-04453102v2/preview/These_LOURME_Olivier.pdf

Annexes

Annexe A – Script JavaScript sur SBC0 (Détection de mouvement)

Ce script a été utilisé dans le module SBC0 pour gérer une alarme domotique intelligente. Il active une alarme visuelle et sonore en cas de détection de mouvement par l'un des deux capteurs connectés aux broches d'entrée.

```
var state = 0;

function setup() {
  pinMode(1, OUTPUT);      // LED d'alarme
  pinMode(0, INPUT);       // Capteur de mouvement 1
  pinMode(2, INPUT);       // Capteur de mouvement 2
  pinMode(3, OUTPUT);      // Sirne ou LED 2
  pinMode(4, OUTPUT);      // Sirne ou LED 3

  IoEClient.setup({
    type: "MotionAlarm",
    states: [{
      name: "On",
      type: "bool",
      controllable: true
    }, {
      name: "Motion Detector",
      type: "bool"
    }, {
      name: "Alarm",
      type: "bool"
    }]
  });

  IoEClient.onInputReceive = function(input) {
    processData(input, true);
  };

  setState(state);
}

function processData(data) {
  if (data.length <= 0)
```

```
    return;
  setState(parseInt(data));
}

function setState(newState, motion, alarm) {
  state = newState;

  if (state === 0) {
    IoEClient.reportStates([state, false, false]);
  } else {
    IoEClient.reportStates([state, motion, alarm]);
  }
}

function loop() {
  if (state !== 0) {
    var motion1 = digitalRead(0);
    var motion2 = digitalRead(2);
    var mt = false;

    if (motion1 !== 0 || motion2 !== 0) {
      mt = true;
      customWrite(1, 1);
      customWrite(3, 1);
      customWrite(4, 1);
    } else {
      mt = false;
      customWrite(1, 0);
      customWrite(3, 0);
      customWrite(4, 0);
    }
    IoEClient.reportStates([state, mt, mt]);
  }

  delay(1000);
}
```

- Le script fonctionne comme un système de sécurité à double capteur.
- Il utilise l'API IoEClient pour communiquer l'état vers l'interface utilisateur dans Packet Tracer.
- Les sorties 1, 3 et 4 sont activées quand un mouvement est détecté.

Annexe B – Script JavaScript sur SBC1 (Contrôle automatique de température)

Ce script a été programmé dans le composant SBC1 pour simuler la gestion thermique intelligente d'une pièce. Il lit une valeur de température numérique (représentée ici par une lecture numérique simple) et active une sortie :

- Chauffage si la température < 15°C
- Climatisation si la température > 25°C
- Rien entre 15°C et 25°C

```
var state = 0;

function setup() {
  pinMode(0, INPUT);   // Capteur de temperature
  pinMode(1, OUTPUT);  // Chauffage
  pinMode(2, OUTPUT);  // Climatisation

  IoEClient.setup({
    type: "Temperature Monitor",
    states: [{
      name: "Auto",
      type: "bool",
      controllable: true
    }]
  });

  IoEClient.onInputReceive = function(input) {
    processData(input, true);
  };

  attachInterrupt(0, function() {
    processData(customRead(0), false);
  });

  setState(state);
}

function processData(data, bIsRemote) {
  if (data.length <= 0)
    return;
  setState(parseInt(data));
}
```

```
33
34 function setState(newState) {
35   state = newState;
36
37   if (state === 0) {
38     digitalWrite(0, LOW);
39   } else {
40     digitalWrite(0, HIGH);
41   }
42
43   customWrite(0, state);
44   IoEClient.reportStates([state]);
45   setDeviceProperty(getName(), "state", state);
46 }
47
48 function loop() {
49   if (state !== 0) {
50     var temp = digitalRead(0);
51     console.log("Temperature:", temp);
52
53     if (temp >= 25) {
54       digitalWrite(1, HIGH);  // Clim OFF
55       digitalWrite(2, LOW);   // Chauff ON
56     } else if (temp < 15) {
57       digitalWrite(2, HIGH);  // Chauff OFF
58       digitalWrite(1, LOW);   // Clim ON
59     } else {
60       digitalWrite(1, LOW);   // Tout OFF
61       digitalWrite(2, LOW);
62     }
63
64     IoEClient.reportStates([temp]);
65   } else {
66     digitalWrite(1, LOW);
67     digitalWrite(2, LOW);
68   }
69   delay(1000);
70 }
```

• digitalRead(0) lit ici une température simulée (probablement générée par le simulateur Cisco IoT).

• Les broches 1 et 2 sont utilisées pour simuler la chauffage et la climatisation.

Annexe C – Script JavaScript sur SBC2 (Contrôle des portes avec capteurs)

Ce script, déployé sur le composant SBC2, permet la gestion automatique des ouvertures de la maison :

• Ouverture de la porte principale (door) basée sur un détecteur à droite (detectorD)

• Ouverture de la porte du garage (garage) basée sur un détecteur à gauche (detectorG)

• Il utilise des lectures analogiques simulées pour activer ou désactiver les sorties

```
var detectorD = 2;
var detectorG = 3;
var door = 1;
var garage = 0;
var state = 0;
var doorState = 0;
var garageState = 0;

function setup() {
  pinMode(door, OUTPUT);
  pinMode(detectorD, INPUT);
  pinMode(garage, OUTPUT);
  pinMode(detectorG, INPUT);

  IoEClient.setup({
    type: "RFID",
    states: [{
      name: "On",
      type: "bool",
      controllable: true
    }, {
      name: "door status",
      type: "bool"
    }, {
      name: "garage status",
      type: "bool"
    }]
  });

```

```
30    IoEClient.onInputReceive = function(input) {
31      processData(input, true);
32    };
33
34    setState(state);
35  }
36
37  function processData(data) {
38    if (data.length <= 0)
39      return;
40    setState(parseInt(data));
41  }
42
43  function setState(newState) {
44    state = newState;
45
46    if (state === 0) {
47      IoEClient.reportStates([state, false, false]);
48    } else {
49      IoEClient.reportStates([state, doorState, detectorG]);
50    }
51  }
52
53  function loop() {
54    if (state !== 0) {
55      if (analogRead(detectorD) === 0) {
56        customWrite(door, 1);
57        doorState = 1;
58      } else {
59        customWrite(door, 0);
60        doorState = 0;
61      }
62
63      if (analogRead(detectorG) === 0) {
64        customWrite(garage, 1);
65        garageState = 1;
66      } else {
67        customWrite(garage, 0);
68        garageState = 0;
69      }
70
```

```
    IoEClient.reportStates([state, doorState, garageState]);
  }
  delay(1000);
}
```

• Ce script utilise analogRead() pour détecter l'état des capteurs d'entrée. Si les détecteurs simulent des RFID ou capteurs de proximité, c'est bien adapté.
• Les états (doorState, garageState) sont envoyés en retour à l'interface utilisateur.

Annexe D – Script JavaScript sur SBC3 (Détection Incendie et CO)

Ce script a été déployé sur le composant SBC3 pour assurer la détection de fumée (capteur de feu) et de dioxyde de carbone (CO). Il contrôle plusieurs sorties pour signaler l'alerte :

- Détection d'incendie
- Détection de gaz CO
- Alarme sonore (sirène)
- Activation d'un système de sécurité incendie

```
function setup() {
  pinMode(1, OUTPUT); // Sir ne
  pinMode(2, OUTPUT); // Alarme g n rale
  pinMode(3, OUTPUT); // LED feu
  pinMode(0, INPUT);  // Capteur feu

  pinMode(9, INPUT);  // Capteur CO2
  pinMode(4, OUTPUT); // LED CO2
  pinMode(5, OUTPUT); // Clignotant / syst me d' vacuation
}

function loop() {
  var fireD = digitalRead(0);   // D tection incendie
  var CO2D = analogRead(9);     // D tection gaz CO2
  Serial.println(CO2D);

  // Si incendie d tect
  if (fireD !== 0) {
    customWrite(3, 1); // LED feu ON
    customWrite(1, 1); // Sir ne ON
  } else {
    customWrite(3, 0);
```

```
    customWrite(1, 0);
  }

  // Si gaz d tect
  if (CO2D !== 0)
    customWrite(4, 1); // LED CO2 ON
  else
    customWrite(4, 0);

  // Si au moins une alerte est active
  if (fireD !== 0 || CO2D !== 0) {
    customWrite(5, 1); // Clignotant / syst me  vacuation  ON
    customWrite(2, 1); // Alarme g n rale ON
  } else {
    customWrite(5, 0);
    customWrite(2, 0);
  }

  delay(1000);
}
```

Ce script assure la surveillance des risques domestiques dans la maison intelligente à l'aide de deux capteurs :
• Un capteur de feu (digitalRead(0)) pour détecter un départ d'incendie.
• Un capteur de gaz CO (analogRead(9)) pour detecter une concentration anormale de dioxyde de carbone.

Annexe E – Installation de OpenSSL

OpenSSL est utilisé pour générer les certificats nécessaires à la sécurisation des communications TLS/SSL entre les dispositifs IoT et le broker MQTT.

Étapes d'installation sur Ubuntu :

```
sudo apt update
sudo apt install openssl
```

Génération d'un certificat auto-signé (valide 1 an) :

```
# Generation de la cle privee de l'autorite de certification
openssl genrsa -out ca.key 2048

```

```
# Generation du certificat de l'autorite (CA)
openssl req -x509 -new -nodes -key ca.key -sha256 -days 365
-out ca.crt

# Generation de cle privee pour le serveur
openssl genrsa -out server.key 2048

# Generation de la requ te de certificat pour le serveur
openssl req -new -key server.key -out server.csr

# Signer le certificat serveur avec la cle CA
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key
-CAcreateserial -out server.crt -days 365 -sha256
```

Annexe F – Installation et configuration de EMQX

EMQX est un broker MQTT haute performance utilisé pour gérer la communication entre les objets connectés.

Étapes d'installation (Ubuntu 22.04) :

```
wget https://www.emqx.com/en/downloads/broker/5.7.0/emqx-5.7.0-
    ubuntu22.04-amd64.deb
sudo dpkg -i emqx-5.7.0-ubuntu22.04-amd64.deb
sudo systemctl start emqx
sudo systemctl enable emqx
```

Interface d'administration Web :

- Adresse : http ://localhost :18083
- Identifiants par défaut :
  - Nom d'utilisateur : admin
  - Mot de passe : public

Annexe G – Installation de MQTTX (client MQTT)

MQTTX est une application graphique permettant de tester les connexions MQTT avec ou sans chiffrement.

```
wget https://www.emqx.com/en/downloads/MQTTX/v1.11.1/MQTTX_1.11.1
    _amd64.deb
```

```
sudo dpkg -i MQTTX_1.11.1_amd64.deb
```

Configuration TLS dans MQTTX :

- Protocole : mqtts
- Port : 8883
- Certificat CA : ca.crt
- Certificat client : client.crt
- Clé privée : client.key