République Algérienne Démocratique et Populaire

Ministère de l'enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahleb Blida 1

Faculté des Science

Département d'Informatique

Projet de fin d'étude pour l'obtention du diplôme de Master en sécurité des systèmes d'information

Thème

Étude des problèmes de sécurité liés au contrôle collaboratif d'une plateforme IoT via le Web

Organisme d'accueil : CDTA

Encadreur :

HENTOUT Abdelfetah

Co-Encadreur :

TIBERKAK Allal

Réalisé par :

BENMEDDAH Mohamed

Soutenue le 22/09/2019 devant le jury composé de :

Mm. ARKAM	**Présidente**
Mr. OULD KHAOUA	**Examinateur**

Année universitaire : 2018/2019

REMERCIEMENTS

Louanges à celui qui sans lui rien n'aurait pu être existé

J'adresse ensuite mes remerciements les plus sincères et les plus chaleureux à mon promoteur M. TIBERKAK Allal pour son encadrement, ses conseils, et son aide.

Mes profonds remerciements vont à M. HENTOUT Abdelfetah, mon Encadreur pour m'avoir proposé ce thème et aidé inconditionnellement à le réaliser tout au long de mon stage, ainsi que pour son constant et son réconfortant soutien.

Je tiens à formuler ma reconnaissance aux membres du jury.

Nous tenons à remercier tous les employés du service de sécurité des systèmes d'information de l'université pour l'aide qu'ils m'ont apporté.

Et enfin, merci à tous ceux qui ont contribué de près ou de loin à m'apporter de l'aide.

DEDICACE

Une fleur d'amour et de reconnaissance est dédiée à mes parents.

À mes sœurs

À mes frères

À toute ma famille.

Et à tous mes amis.

Mohamed.

Résumé

Le domaine de WWW (*World Wide Web*) où on trouve WoT (*Web of Things*) rend le monde physique 100% communicant. Cependant, les environnements où l'IoT (*Internet of Things*) évolue doivent également s'adapter pour être sécurisés. Le domaine de Web avait plusieurs problèmes de sécurité ; ceci ne supporte pas de propager l'utilisation des objets connectés dans ce domaine. C'est le défi qui a encouragé le travail sur les plateformes IoT.

Nous avons choisi d'utiliser un modèle de gestion d'identité centré sur l'utilisateur pour avoir une meilleure intégration des services dans le système d'informations. Pour valider l'approche proposée, nous avons implémenté cette solution pour sécuriser la partie liée à l'accès multiple à cette plateforme, en accomplissant quelques tests qui affirment les résultats.

Mots clés :

HTTPS, WebSockets sécurisés, WebRTC, IoT, Gestion de l'identité numérique.

Abstract

The World Wide Web domain where we find WoT (Web of Things) make the physical world 100% communicating. However, environments where IoT is evolving must also adapt to be secure. The Web domain had several security issues; this does not support propagating the use of connected objects in this domain. This is the challenge that has encouraged work on IoT platforms.

We chose to use a user-centric identity management model to better integrate services into the information system. To validate our approach we showed the implementation of this solution to secure the part related to multiple access to this platform, supported with some tests that affirm the results.

Keywords

HTTPS, Secure WebSockets, WebRTC, IoT, Digital Identity Management.

ملخص

إن شبكة الويب العالمية (WWW) حيث نجد WoT (عالم الأشياء)، جعلت العالم المادي في حالة اتصال بنسبة 100٪، ومع ذلك، يجب على البيئة التي تتطور فيها انترنت الأشياء أن تضبط لتكون آمنة. الويب كبيئة يحتوي العديد من مشاكل أمن المعلومات؛ مما لا يدعم انتشار استخدام الأشياء المتصلة في هذه البيئة. هذا ما يعتبر تحديا شجع على العمل في منصات إنترنت الأشياء.

لقد اخترنا استخدام نموذج إدارة الهوية المتمحور حول المستخدم لدمج الخدمات بشكل أفضل في نظام المعلومات. للتحقق من صحة النموذج المقترح، قمنا بعمل تطبيقي لهذا الحل لتأمين الجزء المتعلق بالمدخل المتعدد لهذه المنصة، من خلال إجراء بعض الاختبارات التي تؤكد صحة النتائج.

كلمات مفتاحية:

HTTPS، WebSockets آمنة، WebRTC، انترنت الأشياء، إدارة الهوية الرقمية.

Table des matières

Chapitre 02 : Problèmes de sécurité liés à la plateforme IoT

Chapitre 03 : Solution pour sécuriser l'accès multiple à la plateforme IoT

Chapitre 04 : Validation de la solution proposée

Liste des figures

Introduction Générale

Contexte

Les transformations du monde informatique telles que le Cloud et les plateformes IoT sont des accélérateurs de l'évolution actuelle des réseaux d'entreprise. Ainsi, l'accès et le contrôle adaptés au contexte sont désormais une priorité pour les administrateurs de la sécurité informatique, afin d'équilibrer leurs ressources pour garantir la sécurité des données et le respect des standards de réseau.

Problématique

De nos jours, nous vivons des avances incroyables dans la miniaturisation des composants électroniques de calcul et de communication, des moyens d'accès à internet, et des technologies de perception de l'environnement physique et biologique. Ces avancés ont encouragé le travail sur l'Internet des objets (IoT), où les objets peuvent interagir avec l'enveniment où ils se trouvent et communiquer entre eux via Internet, comme ils peuvent être contrôlés à distance.

Grâce aux standards du World Wide Web, l'univers des IoT devient plus ouvert et plus proche à l'internet traditionnel. Cependant, certains protocoles n'ont pas été spécialement conçus pour les objets IoT, certains d'autres souffrent d'une fragilité en matière de sécurité, plus précisément les protocoles de communication en temps réel. En conséquence, une vague de problèmes de sécurité Web liés aux plateformes IoT, avec un impact devient un risque en métiers physiques.

Dans un travail antérieur, une architecture de contrôle à distance d'une plateforme IoT a été proposée. Celle-ci peut être contrôlée par plusieurs opérateurs directement via le Web en utilisant les technologies HTTP, WebSocket ou WebRTC. De plus, les opérateurs peuvent communiquer et collaborer entre eux en utilisant la technologie WebRTC.

Objectifs

Le travail demandé dans le cadre de ce travail de Master concerne, d'une part, l'étude des problèmes liés à la sécurité informatique de l'architecture de la plateforme IoT proposée précédemment. D'autre part, il s'agit de développer une solution capable de sécuriser l'accès multiple à cette plateforme collaborative, qui

s'implique dans une communication en temps réel via le Web. Cette solution doit remédier aux problèmes de sécurité Web liés aux plateformes.

Organisation du mémoire

En plus de cette introduction générale, ce mémoire est organisé en quatre chapitres :

- Le premier consiste en un état de l'art sur la sécurité de Web et la gestion de l'identité numérique. Ceci représente le contexte général.
- Dans le deuxième chapitre, nous mettons l'accent sur les travaux connexes liés à l'architecture de plateformes IoT, afin d'illustrer les problèmes de sécurité Web.
- Le troisième chapitre est le cœur de notre travail ; il présentera le cadre conceptuel de la solution proposée.
- Dans le dernier chapitre, nous définissons les outils logiciels utilisés. Le chapitre montre également quelques tests et les résultats obtenus.

À la fin, une conclusion générale résumera le travail réalisé et donnera quelques perspectives futures.

Chapitre 01

État de l'art sur la sécurité du Web

I. Introduction

La sécurité Web est un ensemble de procédures de pratiques et de technologies destinées à protéger les serveurs Web, les utilisateurs Web et les organisations environnantes. Les utilisateurs et les fournisseurs de services Web ont également un ensemble d'hypothèses concernant le comportement attendu du Web en matière de sécurité [1]. Ce premier chapitre contient les détails requis sur cet ensemble.

II. Notions sur le Web

Le web a été créé en 1990 comme application de partage d'informations puis est devenu une plateforme à part entière sur laquelle sont développées régulièrement des nouvelles technologies, Les bases de ces technologies sont le protocole réseau HTTP, qui supporte la communication de type client/serveur [2].

II.1. Serveur Web

Dans un réseau informatique, un serveur est un ordinateur et un logiciel dont le rôle est de répondre automatiquement à des demandes envoyées par des clients - ordinateur et logiciel via le réseau. Parmi les services les plus connues on trouve le service Web. Traditionnellement, les applications Web n'étaient rien de plus qu'un ensemble de fichiers HTML statiques pouvant être récupérés à partir d'un serveur Web, puis restitués et affichés par un navigateur Web.

Aujourd'hui, Les serveurs Web utilisent une méthode standard c'est l'interface CGI (Common Gateway Interface), elle délègue la génération des pages Web à un exécutable (n'importe quelle langue est prise en charge par le serveur Web), également un langage de script, tel que Perl ou Python est un interpréteur approprié sur les serveurs Web.

Avec CGI, chaque demande à un exécutable provoque la création d'un nouveau processus sur le serveur Web. Étant donné que le temps nécessaire pour créer et détruire un processus peut prendre beaucoup plus de temps que la génération de la page Web elle-même, effectuer de petits calculs entraîne une surcharge significative. Pour cela, plusieurs approches alternatives à CGI ont été proposées. Les exemples incluent des techniques spécifiques au serveur Web tels que les modules Apache, les plug-ins IIS ISAPI et Fast CGI. En outre, de nouvelles architectures complètes pour les sites Web dynamiques ont été proposées. Ces

solutions couvrent les threads au lieu de créer des processus à la demande. Par conséquent, leur temps d'exécution est inférieur à celui des solutions qui nécessitent la création et la destruction de processus. Java Enterprise Edition et Microsoft ASP.NET sont les exemples de ces solutions [3].

II.2. Navigateur Web

Le navigateur Web est un logiciel capable de résoudre le nom en appelant un service DNS, et d'échanger avec les serveurs Web, Il utilise un modèle interne pour traiter les documents. Ce modèle est équivalent au DOM (Document Object Model), une manière indépendante de la plate-forme et du langage pour représenter le contenu HTML et XML. De nombreuses implémentations de DOM permettent aux langages côté client de manipuler le modèle via une interface de programmation (API) [4].

II.3. Communication client/serveur

II.3.1. Protocole HTTP

Le protocole de transfert hypertexte (HTTP, HyperText Transfer Protocol) est l'un des protocoles de la suite TCP/IP, qui a été développé pour publier et extraire des pages HTML. Il fonctionne comme un protocole de requête et de réponse. C'est un protocole générique, sans état, qui peut être utilisé pour de nombreuses tâches autres que l'hypertexte [5].

HTTP a été utilisé par l'initiative d'informations mondiales du World-Wide Web depuis 1990, qui est une mise à jour de la RFC 7540 (HTTP/2).

- **Fonctionnement :** Un navigateur Web ou un agent client envoie un message de requête HTTP à un serveur Web pour une ressource donnée identifiée par une URL. Le serveur Web répond à son tour avec un message de réponse HTTP. Le message de réponse contient l'état d'achèvement de la demande et peut contenir dans son corps le contenu demandé par l'agent d'utilisateur.

a. Requête HTTP

Le protocole HTTP prend en charge neuf actions différentes pouvant être effectuées par une ressource identifiée dans une demande. Ces actions sont appelées

méthodes de requête HTTP et sont les suivantes : HEAD, GET, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT et PATCH.

Les méthodes GET et POST sont les plus utilisées sur le Web. Les requêtes GET sont destinées à extraire une représentation de la ressource spécifiée et ne doivent pas être utilisées pour modifier l'état du serveur. Une requête GET peut être utilisée pour envoyer des informations au serveur en incluant une chaîne de requête contenant des paires attribut-valeur dans l'URL demandée. Les requêtes POST sont généralement utilisées pour envoyer des données à un serveur Web. Contrairement aux requêtes GET, les requêtes POST peuvent être utilisées pour créer ou mettre à jour de nouvelles ressources sur le serveur Web.

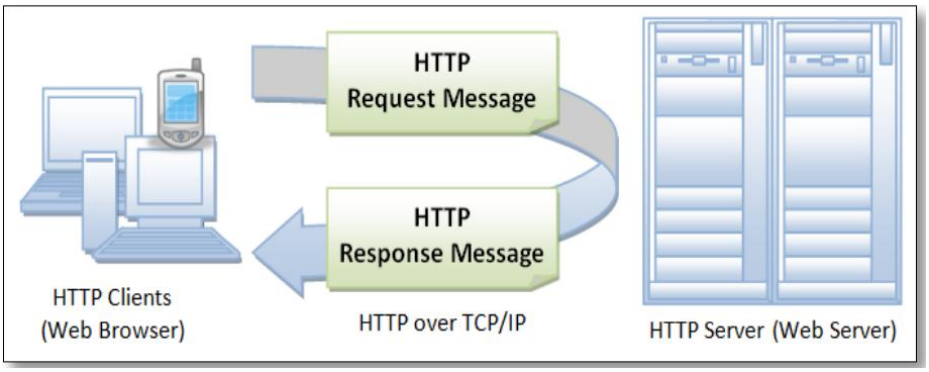

Figure 01 : Communication client/serveur via http [W1]

b. Réponse HTTP

Le message de réponse HTTP comprend un code qui indique l'état de message par exemple : 200 indique que le traitement de la demande a abouti, 301 indique que la ressource a été déplacée vers une nouvelle URL, 400, la requête n'a pas pu être comprise par le serveur Web en raison d'une syntaxe mal formée, 401, la requête nécessite une authentification et 404, le serveur Web n'a trouvé aucune ressource correspondante avec l'URL demandée.

II.3.2. Communication asynchrone

Il existe trois modèles principaux pour la communication asynchrone dans le modèle client/serveur : [6]

- ***Polling*** : où le client Web envoie une requête à intervalles réguliers et le serveur Web envoie une réponse immédiatement, puis ferme la connexion.

- ***Long polling*** : où le client Web envoie une demande et le serveur Web maintient la connexion ouverte pendant une période prolongée.
- ***Streaming*** : où le serveur Web maintient la connexion ouverte indéfiniment et diffuse les réponses vers le client Web jusqu'à ce que le client Web mette fin à la connexion.
- **AJAX :** JavaScript et XML asynchrones (AJAX, Asynchronous JavaScript and XML) est la méthode principale permettant d'activer la communication asynchrone sur le Web, elle utilise XML pour structurer les informations transmises et modifier le contenu de la page via le code JavaScript. Dans une application Web traditionnelle, chaque page web a une adresse URL[2] unique, qu'on appelle page web statiques, alors que dans une application AJAX, chaque état ne peut pas être représenté par une seule adresse. Une adresse particulière peut avoir beaucoup d'états avec un contenu différent. C'est le cas du web dynamique.

Pour le cas d'une communication de type HTTP-Polling avec AJAX, une application doit répéter les en-têtes HTTP dans chaque demande de client et chaque réponse de serveur, cela peut conduire à une surcharge de communication.

II.3.3. Communication P2P peer-to-peer

a. WebSockets

WebSockets est une technologie émergente intégrant des mécanismes de communication bidirectionnelle dans le Web, il a le potentiel de relever de nombreux défis introduits par AJAX. Par exemple, WebSocket n'ajoute pas de nouveau contenu de manière inhérente au contenu existant, comme c'est le cas avec AJAX [7].

Le protocole WebSocket comprend deux étapes : Le handshake se compose d'un message du client et de la réponse du serveur. La deuxième étape est le transfert de données.

b. WebRTC

WebRTC est un Framework ouvert pour le Web qui active les communications en temps réel dans le navigateur. Il comprend les éléments fondamentaux pour des

communications de haute qualité sur le Web, tels que les composants réseau, audio et vidéo utilisés dans les applications de conversation vocale et vidéo [8].

Ces composants, lorsqu'ils sont implémentés dans un navigateur, sont accessibles via une API JavaScript, permettant aux développeurs d'implémenter facilement leur propre application WebRTC. Bien que la communication de base WebRTC utilise le mode peer-to-peer, l'étape initiale d'établissement de cette communication requiert de la coordination.

- **Signalisation :** Un canal de signalisation est nécessaire afin d'échanger trois types d'information entre les Hôtes :
  - Contrôle de session media : établir et libérer la communication.
  - Configuration réseau des nœuds : adresse de transport (adresse IP et numéro de port) pour l'échange des données en temps réels même en présence de NAT.
  - Capacités multimédia des nœuds : medias supportés, codecs disponibles, résolutions supportées, fréquence d'envoi des paquets, etc.

Aucun flux media ne peut être échangé tant que les informations ci-dessus n'ont pas été proprement échangées et négociées.

III. Vulnérabilités du Web

Le « Open Web Application Security Project (OWASP) » a défini dans l'un de ses projets nommé "TOP 10" les dix classes de vulnérabilités Web les plus critiques. L'objectif principal du Top 10 de l'OWASP est d'informer les développeurs, concepteurs, architectes, managers, et les entreprises au sujet des conséquences des faiblesses les plus importantes inhérentes à la sécurité des applications Web. Le Top 10 fournit des techniques de base pour se protéger contre ces vulnérabilités [W2].

La dernière version la plus récente date de 2017. Les vulnérabilités listées sont les suivantes, classées par ordre d'importance :

III.1. Faille d'injection

Les injections profitent de paramètres d'entrées non vérifiés, leur but est d'injecter du code dans une requête. Ainsi, il est possible de récupérer des

informations se trouvant dans la base de données ou encore d'exécuter des commandes.

Presque toutes les sources de données peuvent être un vecteur d'injection, des variables d'environnement, des paramètres, des services Web externes et internes et tous les types d'utilisateurs.

Les vulnérabilités d'injection se trouvent souvent dans les requêtes SQL, LDAP, XPath ou NoSQL, les commandes de système d'exploitation, les analyseurs syntaxiques XML, les en-têtes SMTP, les langages d'expression et les requêtes ORM [9].

- **Impact :** L'injection peut entraîner une perte de données, une corruption ou une divulgation à des parties non autorisées, une perte de responsabilité ou un refus d'accès. L'injection peut parfois mener à la prise de contrôle de l'hôte.

III.2. Violation d'authentification

Les fonctions d'application liées à l'authentification et à la gestion de session sont souvent implémentées de manière incorrecte, permettant aux attaquants de compromettre les mots de passe, les clés ou les jetons de session, ou d'exploiter d'autres failles d'implémentation afin de prendre l'identité de l'utilisateur temporairement ou définitivement. Deux principales FAILLES de violation d'authentification [10] :

a. Réinitialisation du mot de passe non-sécurisé

L'attaquant profite de la similitude des processus d'enregistrement et de réinitialisation du mot de passe pour lancer une attaque par intermédiaire (MiTM) au niveau de l'application. Il initie un processus de réinitialisation du mot de passe avec un site Web, et transmet chaque défi à la victime qui souhaite s'inscrire sur le site attaquant ou accéder à une ressource particulière de celui-ci.

L'attaque comporte plusieurs variantes, notamment l'exploitation d'un processus de réinitialisation du mot de passe reposant sur le téléphone portable de la victime, par SMS ou appel téléphonique. Google, Facebook et D'autres sites Web et certaines applications mobiles populaires sont vulnérables.

b. ID de session non sécurisé

L'identifiant de session est simplement une chaîne de caractères ou de chiffres et fonctionne un peu comme un numéro de sécurité sociale. Le serveur garde en mémoire que l'identifiant de session (SID) a été donné à l'utilisateur et autorise l'accès lorsqu'il est présenté.

Avoir l'identifiant de session évite de devoir fournir le nom d'utilisateur et le mot de passe, ce qui est donc une cible pour les pirates potentiels.

- **Impact :** Selon le domaine de l'application, cela peut permettre le blanchiment d'argent, la fraude à la sécurité sociale et le vol d'identité, ou la divulgation d'informations hautement sensibles protégées par la loi.

III.3. Exposition de données sensibles

La faille est simplement de ne pas chiffrer les données sensibles. Même si le chiffrement est utilisé, une gestion faible de clés (générateur/échange), ainsi que l'utilisation d'algorithmes et de protocoles faibles, et les mauvaises techniques de stockage avec hashage de mots de passe, l'exposition reste exister.

Les données sensibles méritent une protection supplémentaire tel un chiffrement statique ou en transit, ainsi que des précautions particulières lors de l'échange avec le navigateur.

- **Impact :** Au cours des dernières années, cette attaque a été la plus percutant, compromet toutes les données confidentielles telles que les dossiers médicaux, les informations d'identification, les données personnelles et les cartes de crédit, qui nécessitent souvent une protection telle que définie par les lois ou les réglementations.

III.4. XML External Entities (XXE)

De nombreux processeurs XML anciens ou mal configurés évaluent les références d'entités externes au sein de documents XML. Les entités externes peuvent être utilisées pour divulguer des fichiers internes à l'aide du gestionnaire de fichiers URI, des partages de fichiers internes, de l'analyse du port interne, de l'exécution de code à distance et des attaques par déni de service.

- **Impact :** Ces failles peuvent être utilisées pour extraire des données,

exécuter une requête à distance du serveur, analyser des systèmes internes, lancer une attaque par déni de service et exécuter d'autres attaques.

III.5. Violation de contrôle d'accès

Les restrictions sur ce que les utilisateurs authentifiés sont autorisés à faire ne sont souvent pas correctement appliquées. Les pirates peuvent exploiter ces vulnérabilités pour accéder à des fonctionnalités et/ou à des données non autorisées, telles qu'accéder aux comptes d'autres utilisateurs, afficher des fichiers sensibles, modifier les données d'autres utilisateurs, modifier les droits d'accès, etc.

- **Impact :** L'impact est constitué par des attaquants agissant en tant qu'utilisateurs ou administrateurs, ou par des utilisateurs de fonctions privilégiées, ou par la création, l'accès, la mise à jour ou la suppression de chaque enregistrement.

III.6. Mauvaise configuration sécurité

Le problème concerne le système d'exploitation ainsi que les outils installés pour servir l'application web : des options sont installées par défaut alors qu'elles ne sont pas nécessaires au bon fonctionnement de l'application, comptes créés avec des mots de passe par défaut. Ces comptes et mots de passe sont les cibles privilégiées des usurpations d'identité. S'ils ne sont pas mis à jour, l'attaquant peut exploiter les failles non corrigées. Une bonne sécurité nécessite de disposer d'une configuration sécurisée définie et déployée pour l'application, contextes, serveur d'application, serveur web, serveur de base de données et la plate-forme. Tous ces paramètres doivent être définis, mis en œuvre et maintenus, car beaucoup ne sont pas livrés sécurisés par défaut. Cela implique de tenir tous les logiciels à jour [11].

- **Impact :** Ces failles donnent souvent aux attaquants un accès non autorisé à certaines données ou fonctionnalités du système. Parfois, de tels défauts entraînent un compromis complet du système.

III.7. Cross-Site Scripting (XSS)

Une attaque XSS est une attaque par injection de contenu (script), L'attaque consiste à injecter un code malveillant dans une page Web, afin qu'il soit exécuté dans le navigateur de l'utilisateur victime. Une fois injecté, le code malveillant

gagne le même pouvoir que n'importe quel code dans le contexte de la page, ne peut pas être distingué du code légitime. Ce qui permet à l'attaquant de voler des informations d'utilisateur [12]. Il existe en fait trois types d'attaque XSS :

a. **XSS par réflexion** (reflected XSS) L'utilisateur devra interagir avec certains liens malveillants pointant vers une page contrôlée par un attaquant, tels que des sites Web malveillants, des publicités ou similaires.

b. **XSS stockée** (stored XSS) s'appuie sur le fait que l'attaquant réussisse à stocker dans la base de données du code frauduleux qui sera exécuté par la victime lorsqu'elle tentera d'afficher la donnée malveillante.

c. **XSS basé sur DOM :** La page elle-même (la réponse HTTP) ne change pas, mais le code côté client contenu dans la page s'exécute différemment en raison des modifications malveillantes survenues dans l'environnement DOM.

- **Impact :** XSS permet à des attaquants d'exécuter du script dans le navigateur de la victime afin de détourner des sessions utilisateur, défigurer des sites web, afficher des formulaires dont les saisies, récupérer les cookies présents sur la machine de la victime, exécuter des commandes systèmes et construire des liens déguisés vers des sites malveillants.

III.8. Désérialisation non sécurisée

La désérialisation, c'est-à-dire l'extraction de données d'une séquence d'octets, est un processus continu entre les applications qui communiquent entre elles. Une désérialisation mal sécurisée peut notamment conduire à l'exécution de code à distance et à des attaques, y compris par relecture, injection ou escalade de privilèges.

- **Impact :** L'impact des défauts de désérialisation ne peut pas être surestimé. Ces failles peuvent conduire à des attaques à l'exécution de code à distance, l'une des attaques les plus sérieuses possibles.

III.9. Utilisation de composants avec des vulnérabilités connues

Les composants vulnérables, tels que bibliothèques, contextes et autres modules logiciels fonctionnent presque toujours avec des privilèges maximums. Ainsi, si

exploités, ils peuvent causer des pertes de données sérieuses ou une prise de contrôle du serveur. Les applications utilisant ces composants vulnérables peuvent compromettre leurs défenses et permettre une série d'attaques et d'impacts potentiels.

- **Impact :** Certaines vulnérabilités connues n'ont que des incidences mineures, mais certaines des plus grandes violations à ce jour reposent sur l'exploitation de vulnérabilités connues des composants. Selon les actifs que vous protégez, ce risque devrait peut-être figurer en haut de la liste.

III.10. Journalisation et surveillance insuffisantes

Est la cause de presque tous les incidents majeurs, une journalisation et une surveillance insuffisantes permettent aux pirates de renforcer et de prolonger leurs stratégies d'attaque, de maintenir la persistance, de « pivoter vers d'autres systèmes, et d'altérer, extraire ou détruire des données ».

Les attaques reposent sur le manque de surveillance et de réaction rapide pour atteindre leurs objectifs sans être détectées.

- **Impact :** La plupart des attaques réussies commencent par une analyse de vulnérabilité. Le fait de permettre à de telles enquêtes de continuer peut augmenter la probabilité de réussite de l'exploitation à près de 100%.

IV. Mécanismes de la sécurité Web

- **Mécanismes au côté serveur**

La sécurité Web coté serveur assure que le demandeur de service soit légitime et responsable. Légitime dans le sens où l'utilisateur a été identifié avec précision. Responsable c'est-à-dire l'utilisateur ne tentera pas d'accéder à des documents restreints, ou d'obtenir un accès illégal à un autre système informatique [1].

IV.1 Gestion d'identité numérique

IV.1.1 Identité numérique

Dans ce monde internet, le besoin de savoir avec qui ou quoi nous communiquons est devenu indispensable et un nouveau terme "identité numérique" est né. Il peut être considéré comme une « représentation informatique » d'une

entité. Ce sont les moyens informatiques et technologiques qui permettent à cette entité de se projeter dans le monde du numérique, cette projection pouvant prendre plusieurs formes selon son contexte (administratif, professionnel, réseaux sociaux, etc.). La notion d'entité peut concerner une personne (physique ou morale), une ressource ou un groupe d'individus (personnes physiques). [13]

Le but d'une identité numérique est de permettre la création d'une relation de confiance entre deux entités à travers l'application de politiques prédéfinies.

- **Autorité de délivrance :** délivre l'identité numérique, en faisant le lien initial entre la personne physique ou morale et l'identité numérique.

IV.1.2. Attributs

L'identité consiste à associer à leur entité un ensemble de données numériques, également appelées « attributs ». Ces attributs peuvent être relativement statiques dans le temps comme nom, prénom, adresse, empreinte vocale d'un individu, ou fortement dynamiques comme ses centres d'intérêt, le nom de l'application utilisée avec son heure de démarrage et la durée de l'utilisation, voire la géolocalisation de la personne s'il s'agit d'une application sur mobile.

- **Fournisseurs d'attributs** : sont des acteurs facultatifs qui fournissent les attributs « supplémentaires » décrivant l'utilisateur. Ainsi, le fournisseur d'identités confirme au fournisseur de services l'identité numérique elle-même et les attributs de base, le fournisseur d'attributs confirme les attributs supplémentaires comme le revenu fiscal [14].

IV.1.3. Système de Gestion d'identité

On définit un système de Gestion d'identités (SGI) comme l'ensemble des processus permettant de gérer le cycle de vie d'une identité numérique, c'est-à-dire, sa création, sa manipulation et sa fin de vie. Le SGI s'occupe aussi des composants opérationnels, lesquels gèrent les différents aspects de la sécurité, c'est-à-dire, le processus d'authentification, le contrôle d'accès et l'audit. De plus, selon la norme ISO/IEC 24760 [15], la gestion des identités inclut la gouvernance, les politiques, les processus, les données, les technologies et les standards permettant notamment de [16] :

- Authentifier les identités.
- Établir la provenance des informations des identités.
- Établir le lien entre les informations sur les identités et les entités.
- Maintenir à jour les informations sur les identités.
- Assurer l'intégrité des informations sur les identités.
- Fournir les justificatifs d'identité et les services pour faciliter l'authentification d'une entité en tant qu'identité reconnue.
- Ajuster les risques de sécurité liés au vol d'information par exemple.

- **Fournisseur d'identités** (IdP, Identity Provider) : il gère l'identité numérique au quotidien et la confirme auprès du fournisseur de services. Dans sa fonction minimale, le fournisseur d'identités gère et confirme uniquement les attributs nécessaires pour l'authentification (par exemple nom, numéro de carte à puce et validité de l'identité numérique).

a. Acteurs de SGI

Un système de gestion des identités numériques implique schématiquement trois types d'acteurs sont :

- **Utilisateur** : est la personne physique ou le représentant d'une personne morale détenteur de l'identité. Il est représenté par un jeu d'attributs (nom, sexe, adresse, revenu fiscal, photos, etc.).
- **Fournisseur de services** (SP, Service Provider) : est un service électronique public ou privé qui permet à l'utilisateur d'accéder à des ressources en ligne ou d'accomplir des actions en ligne. Pour cela, il doit authentifier l'utilisateur, c'est-à-dire s'assurer qu'il s'agit bien de la bonne personne en recourant, en règle générale, à un tiers de confiance.
- **Tiers de confiance :** englobe trois fonctions qui peuvent être assurées par un seul organisme ou des organismes différents : L'autorité de délivrance, Fournisseur d'attributs et Fournisseur d'identité (IdP).

b. Types de SGI

Les technologies mises en œuvre pour la gestion de ces identités numériques sont constituées d'un ensemble d''éléments logiciels et matériels. Des projets comme [17] classifient ces systèmes de gestion d'identité (SGI) en trois types :

- **Type 1 :** SGI pour la gestion de comptes (authentification, autorisation, ...).
- **Type 2 :** SGI pour l'´établissement de profils.
- **Type 3 :** SGI personnel pour la gestion des rôles et pseudonymes.

Chacun de ces types de SGI adresse les problématiques de l'identité numérique différemment et utilise des modèles et des protocoles différents. Les SGI de type 1 et 3 ont pour but de permettre l'utilisation d'une identité numérique pour établir une relation de confiance alors que les SGI de type 2 sont destinés à surveiller l'utilisation d'une identité numérique.

c. Identification et Authentification

La distinction entre identification et authentification d'une entité réside dans le degré de confiance établi entre l'identité déclarée par l'entité et l'identité numérique dont elle est détenteur.

- **Identification :** repose en général sur la simple déclaration par l'entité de son identité sous la forme d'un identifiant numérique de type numéro, nom, alias, pseudonyme, etc.
- **Authentification :** L'authentification consiste à vérifier l'identité d'une entité ayant accès au système d'information, l'individu déclare son identité et apporte la preuve de cette identité, que cette preuve provienne de lui-même ou bien d'un tiers. Cette preuve s'appelle un « credential ». Un « credential » peut se présenter sous la forme d'un mot de passe, d'une réponse à un défi (quelque chose que l'on sait), d'une information fournie par une carte à puce ou un certificat numérique (quelque chose que l'on a), ou d'une information dérivée des caractéristiques de la personne, comme l'empreinte digitale, l'iris ou le timbre de la voix (ce que l'on est), avec des niveaux de robustesse et de fiabilité très hétérogènes. Les techniques modernes renforcent même le niveau d'authentification par le biais d'analyses comportementales. Il s'agit notamment de déterminer quels identificateurs d'entité sont plus importants que d'autres dans le processus d'identification et pourquoi certains identificateurs utilisés pour l'authentification ne devraient pas avoir la même valeur en termes d'authentification [18].

d. Protection de vie privée

Une solution technique pour protéger l'identité de l'individu consiste à couper le lien entre l'identité numérique et l'identité réelle de la personne. Deux propriétés entendent répondre à cet objectif :

- **Pseudonymat** : implique nécessairement l'utilisation d'un pseudonyme, c'est-à-dire un identifiant pouvant être relié à une identité réelle par une autorité légitime.
- **Anonymat** : se distingue du pseudonymat par la difficulté à établir ce lien. C'est sur ce niveau de difficulté à remonter à l'identité réelle de la personne que les scientifiques et les juristes divergent. Les juristes définissent l'anonymat comme l'impossibilité de remonter à la personne, et ce, de façon irréversible.Les scientifiques préfèrent définir un seuil de résistance d'un mécanisme d'anonymat face à des tentatives de levée d'anonymat initiées par un adversaire potentiel. [19]

La propriété d'inassociabilité (unlinkability en anglais) désigne quant à elle l'incapacité à relier au moins deux informations distinctes (messages, URL, identifiants) relatives à un individu.

IV.1.4. Modelés de gestion d'identités

Comme nous l'avons indiqué, les systèmes de gestion d'identités peuvent être classes en trois types, Les modèles que nous présentons ici appartiennent aux types 1 et 3 [20] :

VI.1.4.1. Gestion isolée

Dans ce modèle, chaque fournisseur de service utilise son propre domaine d'identité, donc, son propre fournisseur d'identité. Un utilisateur doit répéter les étapes d'authentification et d'identification auprès de chacun des domaines d'identité rattachés aux fournisseurs de services.

IV.1.4.2. Gestion centralisée d'identités

Dans ce modèle, seuls un identifiant et un justificatif sont utilisés par les fournisseurs de service, Jøsang donnent trois exemples d'implémentation de ce type de gestion d'identité :

a. Identité commun

Dans ce modèle, qui a été représenté dans la figure 02, une identité unique fournie par un fournisseur d'identités sert à authentifier un utilisateur chez plusieurs fournisseurs de services d'un même domaine de sécurité. De ce fait, l'utilisateur utilise les mêmes informations d'identifiant et de justificatif quel que soit le fournisseur de service sollicité, ce qui simplifie l'accès aux différents services.

Ce modèle de gestion impose que chaque fournisseur de service lié au domaine d'identité unique soit déclaré explicitement. Dans le cas contraire, il est difficile d'évaluer les conséquences d'un changement ou d'une altération vis-à-vis des services mis à disposition de l'utilisateur par le biais de cette architecture.

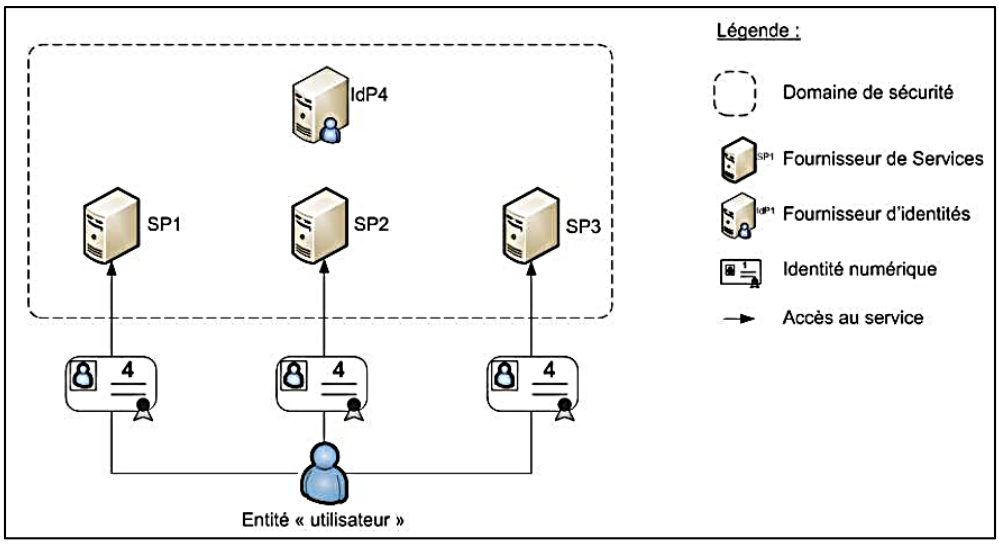

Figure 02 : Modèle de gestion avec une identité commune [23]

b. Méta-identité

Les fournisseurs de services peuvent partager un certain nombre de besoins en terme d'identité qui peuvent être regroupés sous une méta-identité. Dans ce modèle qui est dans la figure 03, les identités du sujet sont liées à une méta-entité qui va être utilisée pour accéder à des services présents dans un domaine de sécurité.

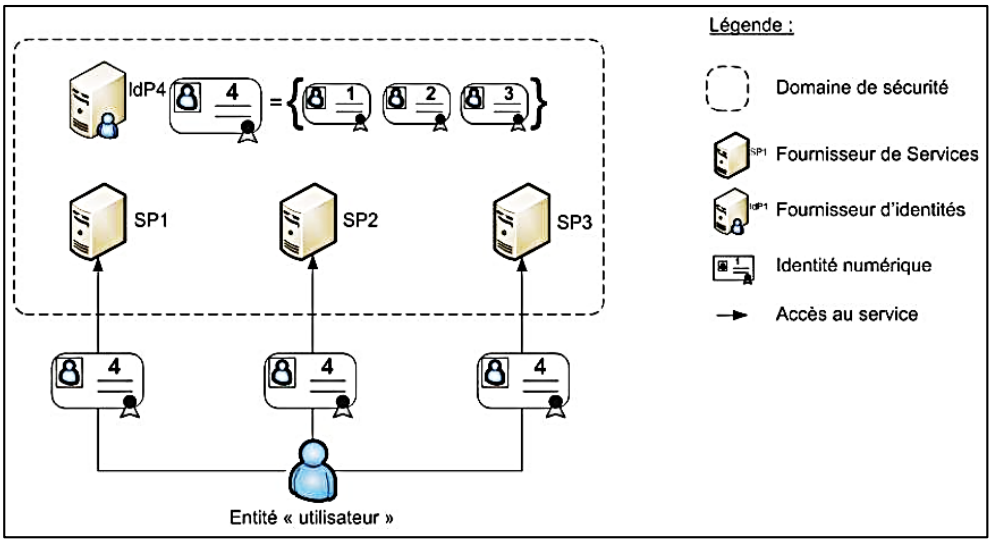

Figure 03 : Modèle de gestion avec méta-identité [22]

Les justificatifs peuvent être liés au méta-identifiant. Dans ce cas, les justificatifs sont les mêmes pour tous les domaines d'identités concernés. Du point de vue de l'utilisateur, cette architecture peut être perçue comme un mécanisme de synchronisation des justificatifs entre les différents domaines d'identité. Cette approche est un moyen de faciliter l'intégration de différents domaines d'identité, comme lors de la fusion de plusieurs organisations. Ainsi, le modèle de méta-identité permet à des domaines d'identité isolés de mettre en commun des informations et éviter la réplication des informations.

c. Single Sign-On (SSO)

L'approche Single Sign-On est similaire à une fédération d'identité, mais aucune correspondance d'identité n'est nécessaire, car il n'existe qu'un seul fournisseur d'identité. Dans cette architecture, un utilisateur n'a besoin de s'authentifier qu'une seule fois (en anglais « single sign-on ») auprès d'un fournisseur de service. Il est alors authentifié de facto auprès des autres fournisseurs de service.

Le modèle Single Sign-On peut être associé au modèle de fédération d'identité, permettant une authentification unique inter-domaine. Dans un environnement collaboratif, l'authentification unique peut être très intéressante.

IV.1.4.3. Gestion fédérée d'identités

Pour simplifier l'utilisation des identités pour les individus et les organisations, la notion de fédération d'identité, qui a été représenté dans la figure 04, permet le

partage d'identités numériques entre plusieurs domaines de sécurité à l'intérieur d'un cercle de confiance. Un cercle de confiance regroupe plusieurs fournisseurs de services qui vont faire confiance à un ou plusieurs fournisseurs d'identités pour authentifier les utilisateurs. Le but de cette fédération est de permettre aux utilisateurs d'accéder au système d'un autre domaine de sécurité sans avoir besoin de changer d'identité. Les scenarios classiques de fédération impliquent le déploiement de solutions d'authentification unique (SSO) entre domaines.

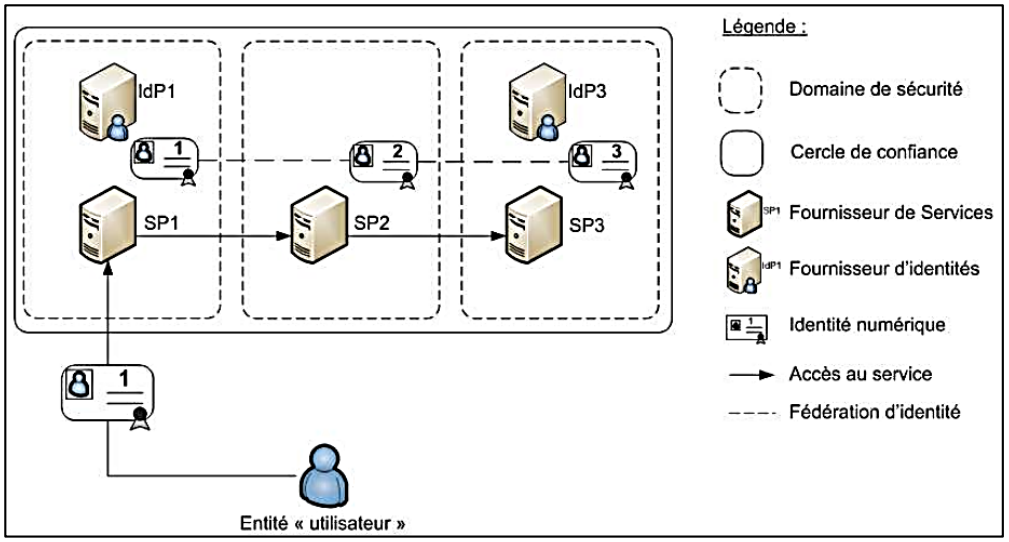

Figure 04 : Modèle de fédérée d'identité [23]

IV.1.5. Protocoles et langages pour la gestion d'identités

Pour permettre l'implémentation des modèles complexes que sont la gestion centrée sur l'utilisateur et la fédération d'identité par exemple, un certain nombre de protocoles et de langages spécifiques ont été proposés [24].

a. SAML2

SAML 2.0 [W3] est une version du standard SAML pour l'échange de données d'authentification et d'autorisation entre domaines de sécurité. SAML 2.0 est un protocole à base de grammaire XML qui utilise des jetons de sécurité contenant des assertions pour véhiculer des informations sur les utilisateurs entre un fournisseur d'identité (IdP) SAML et un fournisseur de service (SP) SAML. SAML 2.0 se prête aux scénarios d'authentification et d'autorisation web incluant un mécanisme de

connexion unique (SSO) inter domaines, ce qui évite de distribuer plusieurs jetons d'authentification à l'utilisateur.

SAML 2.0 définit un certain nombre de protocoles pour le traitement des requêtes et des réponses faites à l'aide de ces assertions, les assertions par exemple sont généralement transportées dans d'autres structures comme des requêtes POST HTTP ou des messages SOAP. Le standard définit donc des méthodes pour réaliser ce lien et transporter les assertions SAML 2.0.

b. OAuth2

Ce protocole [W3] permet à des applications tierces d'obtenir un accès limité à un service disponible via HTTP par le biais d'une autorisation préalable du détenteur des ressources.

L'accès est demandé par ce qu'on appelle « un client », et qui peut être un site internet ou une application mobile par exemple. Si les ressources n'appartiennent pas au client, alors ce dernier doit obtenir l'autorisation de l'utilisateur final, sinon il peut directement obtenir l'accès en s'authentifiant avec ses propres identifiants.

La version 2 est censée simplifier la version précédente du protocole et à faciliter l'interopérabilité entre les différentes applications. Les spécifications sont toujours en cours de rédaction et le protocole évolue sans cesse mais cela ne l'empêche pas d'être plébiscité et implémenté par de nombreux sites tels que Google ou Facebook.

Bien que la spécification OAuth2 n'impose pas de format pour les jetons d'accès, mais l'utilisation de JWT est largement adoptée pour ce protocole.

- **JWT :** JSON Web Token JWT, est un standard ouvert (RFC 7519), définit un moyen compact et autonome de transmettre des informations en toute sécurité entre les parties sous forme d'objet JSON. Ces informations peuvent être vérifiées et approuvées car elles sont signées numériquement. Les JWT peuvent être signés à l'aide d'un secret (avec l'algorithme HMAC) ou d'une paire de clés publique / privée à l'aide de RSA.

  Un JWT est simplement constitué de trois parties séparées par un point : *part1*, *part2*, *part3* encodé en base64.

- **Partie 1 "entête"** : contient un objet JavaScript indiquant l'algorithme utilisé pour *hasher* le contenu par exemple :

$$\{"typ":"JWT","alg":"HS256"\}$$
$$\Rightarrow eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9$$

- **Partie 2 "contenu"** : il est un simple objet avec les informations à échanger entre le client et le serveur, il inclut des champs personnalisés et des attributs réservés comme :
    - *exp* : Date d'expiration du token (expiration),
    - *iat* : Date de création du token (issued at),
- **Partie 3 "signature"** : il représente l'encodage de la concaténation des deux premières parties avec l'algorithme défini dans l'entête.

IV.2. Contrôle d'accès à distance :

Un mécanisme de contrôle d'accès à distance est une solution logicielle ou matérielle pour appliquer une politique de contrôle d'accès.

IV.2.1. Définition

Le contrôle d'accès est l'une des technologies de sécurité permettant de protéger les ressources du système en empêchant les accès non autorisés aux ressources et en limitant les accès des utilisateurs autorisés en fonction de leurs privilèges [25] :

IV.2.2. Facteurs de contrôle d'accès

Le modèle de contrôle d'accès est mis en œuvre à différents niveaux dans de nombreux domaines, tels que le système d'exploitation et le système de gestion de base de données. Tout contrôle d'accès est composé de cinq facteurs :

- Sujets : Entités actives sous la forme d'utilisateurs et de processus qui demandent l'accès à des objets.
- Objets : Entités passives contenant des informations auxquelles les sujets ont accès.
- Actions : Opération à effectuer sur un certain objet (lecture, écriture, exécution, etc.).

- Privilèges : Autorisations permettant d'effectuer certaines actions sur certains objets.
- Politique d'accès : Ensemble de règles déterminant la décision d'accès acceptée ou refusée.

La figure montre le déroulement d'une opération de contrôle d'accès. Il commence lorsqu'un sujet/utilisateur envoie une demande d'accès au gestionnaire de contrôle d'accès pour accéder à un certain objet. Le gestionnaire de contrôle d'accès utilise les détails de l'utilisateur, tels que le nom d'utilisateur ou le mot de passe, puis les compare aux stratégies de contrôle d'accès pour déterminer la décision d'accès. La décision sera soit acceptée, soit rejetée. Si l'accès est accepté, le gestionnaire de contrôle d'accès autorisera l'utilisateur à accéder à l'objet, tandis que si l'accès est refusé, le gestionnaire de contrôle d'accès mettra fin à la session après avoir envoyé un message d'avertissement concernant des informations d'identification insuffisantes.

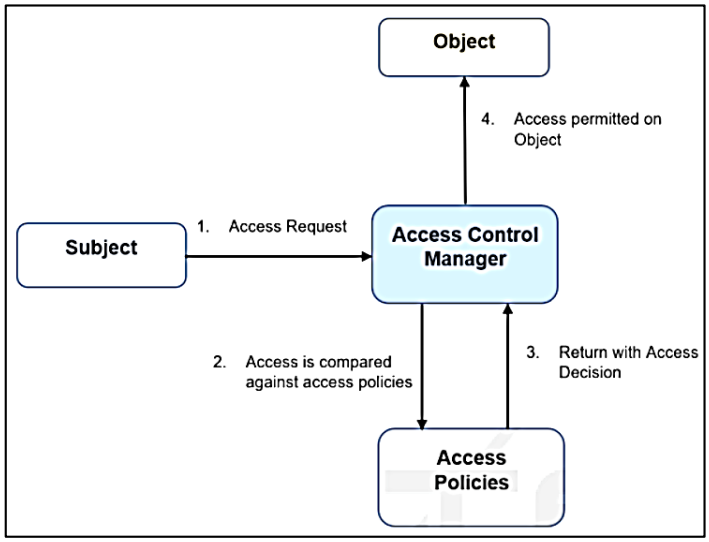

Figure 05 : Éléments principaux d'un modèle de contrôle d'accès [26]

IV.2.3. Modèles de contrôle d'accès

a. DAC : Contrôle d'accès discrétionnaire

Le modèle DAC (Discretionary Access Control) a été conçu pour les bases de données multi-utilisateurs et les systèmes avec quelques utilisateurs connus auparavant.

DAC accorde l'accès en fonction de l'identité et de l'autorisation de l'utilisateur, définies pour les stratégies ouvertes. Le propriétaire de la ressource peut accorder l'accès à n'importe quel utilisateur. Il est appelé discrétionnaire car il offre la possibilité d'autoriser les utilisateurs à transmettre librement leurs autorisations d'accès à d'autres utilisateurs. Par conséquent, toutes les ressources système sont sous le contrôle total de l'utilisateur.

b. MAC : Contrôle d'accès obligatoire

Le contrôle d'accès discrétionnaire est généralement défini par opposition au contrôle d'accès obligatoire ou MAC (Mandatory Access Control), qui impose des règles incontournables garantissant l'atteinte des objectifs de sécurité visés. Dans ce type de contrôle d'accès les sujets ne peuvent pas intervenir dans l'attribution des droits d'accès. Ce contrôle d'accès est plus rigide que le contrôle d'accès discrétionnaire. Le modèle de contrôle d'accès obligatoire est utilisé quand la politique de sécurité est d'une organisation définie que :

- Les décisions de protection d'une ressource ne doivent pas être sous la responsabilité de son propriétaire.
- Le système doit mettre en œuvre les mécanismes permettant de respecter la politique de sécurité et interdire au propriétaire d'une ressource d'agir à sa guise.

c. RBAC

Le modèle RBAC fournit une classification des utilisateurs en fonction de leurs rôles, RBAC limite l'accès aux objets en fonction du rôle du sujet plutôt que de leurs identifications. Les rôles sont attribués aux sujets en fonction de leurs autorisations, de leurs qualifications et de leurs responsabilités au sein de l'organisation. Un ensemble d'autorisations est regroupé pour former un rôle. Un utilisateur peut être affecté à différents rôles et le rôle peut être attribué à différents utilisateurs. Le modèle RBAC peut avoir plusieurs utilisateurs. Chaque utilisateur se voit attribuer un rôle spécifique ou plusieurs rôles et chaque rôle consiste en un ensemble d'autorisations/droits.

- **Mécanisme du côté client**

La sécurité Web coté client assure la légitimité, la sûreté et la confidentialité de service fournis. Légitime dans le sens que la source de service doit être la même

que laquelle l'utilisateur s'attend à fournis. En sûreté que le service ne contient pas de contenu malveillant [1].

IV.3 Politique de même origine (The Same-Origin Policy)

La politique de même origine est essentiellement un accord entre les constructeurs de navigateurs, principalement Microsoft, Apple, Google, Mozilla et Opera- sur, un moyen standard de limiter les fonctionnalités du code de script exécuté dans les navigateurs Web des utilisateurs. La politique de Même origine est également essentielle pour comprendre les mécanismes de sécurité dans les navigateurs. Ce concept garantit qu'un navigateur Web permet aux scripts contenus dans une page Web d'accéder aux données d'une autre page Web, mais uniquement si les deux pages ont la même origine. Une origine est définie comme une combinaison de schéma d'URI, nom d'hôte et numéro de port.

Cette politique empêche un script malveillant d'une page d'obtenir l'accès aux données sensibles d'une autre page Web via le modèle d'objet de document (DOM) de cette page. Ce concept est au cœur de la sécurité des applications Web et repose largement sur les cookies HTTP pour la maintenance des sessions utilisateur authentifiées. Les navigateurs Web conformes à la politique Même origine doivent pouvoir empêcher strictement le partage de données entre des pages côté client afin d'éviter toute perte de confidentialité et d'intégrité des données. Sans la politique de même origine, aucune donnée ne serait en sécurité [27].

- **Sécurité de la communication Web**

En vue global, la sécurité Web doit rend les communications entre le serveur et le client isolées d'espionnage et fiables où les transmissions ne seront pas modifiées par un tiers [1].

IV.4 Protocoles de cryptographie

- **SSL :** SSL [W4] (Secure Sockets Layers, que l'on pourrait traduire par couche de sockets sécurisée) est un procédé de sécurisation des transactions effectuées via Internet. Le standard SSL a été mis au point par Netscape, en collaboration avec Mastercard, Bank of America, MCI et Silicon Graphics. Il repose sur un procédé de cryptographie par clé publique afin de garantir la sécurité de la transmission de données sur Internet. Son principe consiste à établir un canal de

communication sécurisé (chiffré) entre deux machines (un client et un serveur) après une étape d'authentification. Le système SSL est indépendant du protocole utilisé, ce qui signifie qu'il peut aussi bien sécuriser des transactions faites sur le Web par le protocole HTTP que des connexions via les protocoles FTP, POP ou IMAP. En effet, SSL agit telle une couche supplémentaire permettant d'assurer la sécurité des données, située entre la couche application et la couche transport (protocole TCP par exemple).

- **TLS** [W4] **:** Au milieu de l'année 2001, le brevet de SSL qui appartenait jusqu'alors à Netscape a été racheté par l'IETF (Internet Engineering Task Force) et a été rebaptisé pour l'occasion TLS (Transport Layer Security). TLS a été conçu pour des protocoles de transport fiables. Par conséquent, il ne prévoit aucune perte ou réorganisation des messages de la couche de transport. Si un message est perdu ou semble hors d'usage, il assume une attaque et interrompt la connexion.
- **DTLS :** DTLS [W4] est dérivé de certaines caractéristiques de TLS et en hérite. Il permet de réutiliser les fonctionnalités de sécurité TLS en plus du protocole UDP (User Datagram Protocol). Avec l'émergence de CoAP en tant que protocole de transfert Web spécialisé pour les appareils soumis à des contraintes, DTLS est le protocole de sécurité préféré en IoT.

V. Conclusion

Pour lutter et prévenir les risques liés à la sécurité des informations sur le Web, nous avons vu plusieurs solutions qu'ont été proposées. Mais malgré cela la sécurité des données reste un sujet d'inquiétude et représente un défi pour adopter des applications Web, ceci à cause du développement des nouvelles technologies.

Chapitre 02

Problèmes de sécurité liés à la plateforme IoT

I. Introduction

Avec le développement des technologies Web et des applications distribuées, la détection des problèmes de sécurité sont de plus en plus compliqués ; il devient indispensable de concevoir une étude particulière sur ces nouveaux systèmes connectés. Le but de cette étude est d'illustrer les failles de sécurité selon l'architecture du système.

II. Internet des objets

II.1 Définition

L'Internet des objets est un système d'objets physiques pouvant être découverts, surveillés, contrôlés ou interagis, par des appareils électroniques qui communiquent via diverses interfaces réseau et peuvent éventuellement être connectés à Internet. Un objet IoT contient un ou plusieurs des éléments suivants :

- Capteurs (température, lumière, mouvement, etc.).
- Actionneurs (écrans, son, moteurs, etc.).
- Micro-Ordinateur (peut exécuter des programmes).
- Interfaces de communication (avec ou sans fil).

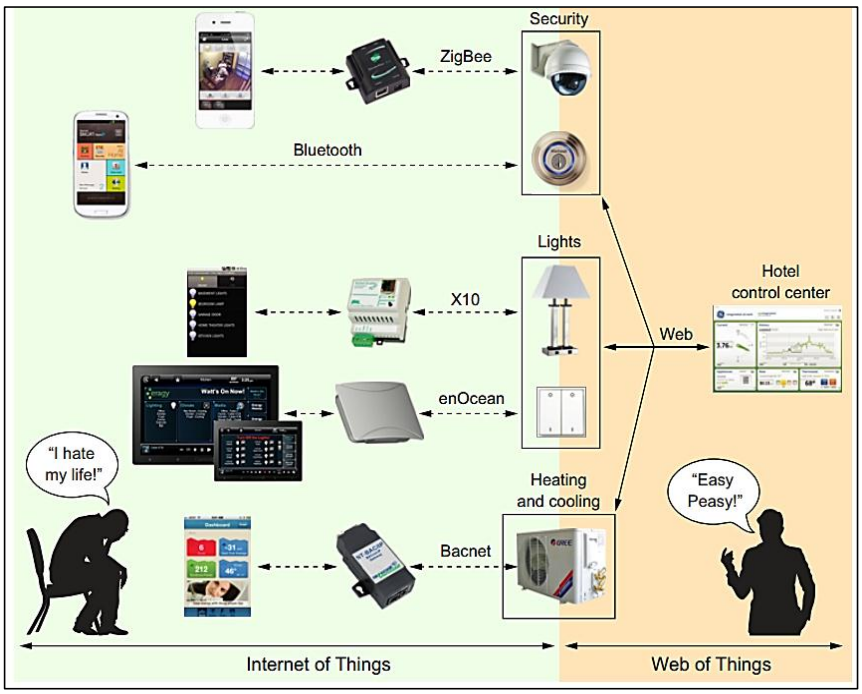

Figure 06 : Différence entre IoT et WoT [28]

La figure 06 montre la différence entre l'IoT et Le WoT. Dans le Web des objets (WoT), les appareils et leurs services sont entièrement intégrés au Web, car ils utilisent les mêmes normes et techniques que les sites Web traditionnels. Cela signifie que vous pouvez écrire des applications qui interagissent avec des périphériques intégrés de la même manière que vous interagissez avec tout autre service Web utilisant des API Web, en particulier des architectures REST [29].

II.2. API de script WoT

L'API de script WoT permet la mise en œuvre de la logique de l'appareil (l'objet connecté) à l'aide de scripts réutilisables exécutés dans un système pour des applications IoT similaires à celles d'un navigateur Web, et vise à améliorer la productivité et à réduire les coûts d'intégration. De plus, les API standardisées permettent la portabilité des modules d'application. Le système d'exécution de l'API de script WoT instancie des objets locaux jouant le rôle d'interface avec d'autres objets et leurs potentialités d'interaction (propriétés, actions et événements). Il permet également aux scripts d'exposer des objets, c'est-à-dire de définir et de mettre en œuvre la potentialité d'interaction et publier une description de l'objet TD [W5].

II.3. Description de l'objet TD

La description de l'objet (TD, Thing Description) est un élément central dans le Web des objets (WoT) et peut être considéré comme le point d'entrée d'un objet (un peu comme l'index.html d'un site Web). Une instance TD a quatre composants principaux : des métadonnées textuelles sur l'objet elle-même, un ensemble de potentialité d'interaction qui indiquent comment l'objet peut être utilisée, des schémas pour les données échangées avec l'objet pour la compréhensibilité machine, et enfin des liens Web pour exprimer toute relation formelle ou informelle avec d'autres objets ou documents sur le Web [W6].

II.4. Protocoles de Communication sur WoT

Les protocoles de communication forment un mappage d'interaction sur des messages concrets d'un protocole spécifique tel que HTTP, CoAP, WebSockets ou WebRTC. Il indique au consommateur comment activer l'interaction via une

interface réseau. Les protocoles suivent la contrainte d'interface uniforme de REST pour prendre en charge l'interopérabilité.

- **CoAP (Constrained Application Protocol) :** CoAP est un protocole utilisé par les périphériques IoT et ressemble à HTTP de nombreuses manières, à la différence qu'il repose sur UDP au lieu de TCP comme protocole de couche 4.

III. Vulnérabilités du Web des objets

En cas d'un attaquant ayant l'accès au réseau de l'objet et/ou l'interface WoT. Ou d'un utilisateur autorisé malveillant, ou d'un développeur de logiciels malveillants accès aux informations d'identification d'utilisateur disponibles sur l'interface de configuration (smartphone, tablette, etc.).

III.1. Menace liée au protocole de liaison.

Toute attaque à distance utilisant l'interface WoT ou directement à l'aide d'interfaces de protocole de liaison dans le but de compromettre le protocole. L'attaquent peut accéder à tous les actifs WoT disponibles

III.2. Menaces liées à l'interface WoT

La figure 07 illustre le client WoT de base utilisé pour exploiter directement à l'objet WoT. Il expose une interface réseau WoT et fournit également directement une description de l'objet.

- **Menace d'instance de l'objet**

Identique à première menace, mais l'attaquant cible directement l'instance d'objet dans l'interface WoT. L'attaquant peut exécuter du code avec les privilèges de l'objet sur le périphérique WoT.

- **Accès non autorisé à l'interface WoT**

En raison du manque d'authentification appropriée à l'interface WoT, un attaquant peut accéder à des informations des objets connectés ou exécuter une action à distance.

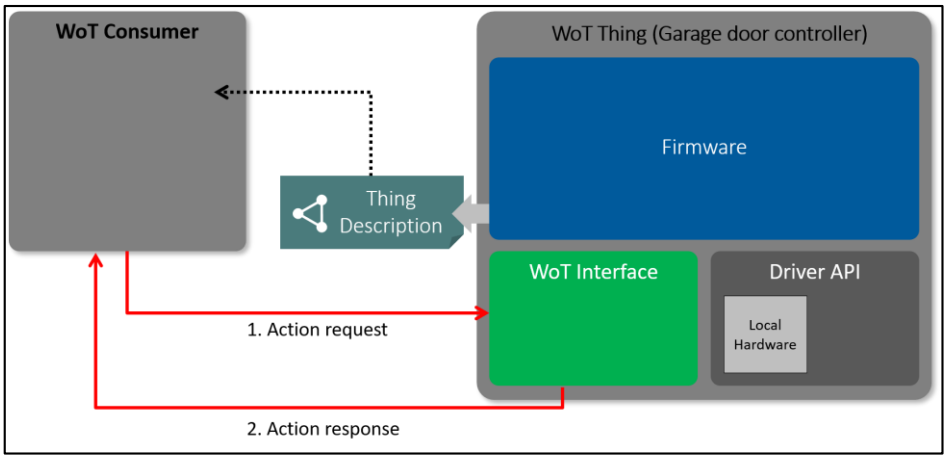

Figure 07 : Communication client/serveur dans WoT [W7]

III.3. Menace liée à fichier TD

Comme il montre dans les figures 07 et 08, au moins une représentation TD doit être disponible. La description est un format compréhensible par la machine, qui permet à l'utilisateur de découvrir et d'interpréter les fonctionnalités de l'objet, et de s'adapter à différentes implémentations (par exemple, différents protocoles ou structures de données) lors de leurs interactions, permettant ainsi l'interopérabilité entre différentes plates-formes IoT [W6].

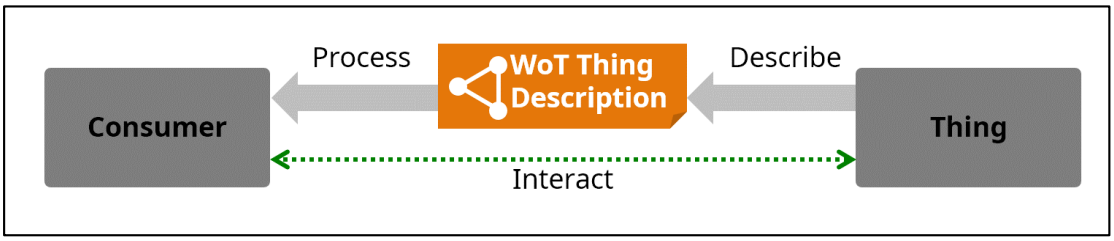

Figure 08 : Interaction entre utilisateur et un objet connecté [W6]

- **Intégrité du TD**

Une application malveillant écoute le réseau et intercepte un TD envoyé par l'un des périphériques WoT. Ensuite, il modifie le TD pour qu'il utilise une méthode ou une autorité d'authentification différente et le transmet au périphérique prévu.

Le périphérique suit les instructions du TD modifié et envoie une demande d'authentification à l'emplacement spécifié par l'attaquant, révélant potentiellement ses informations d'identification, telles que le nom d'utilisateur et le mot de passe. De même, au lieu de modifier le TD légitime, une attaque pourrait

le sauvegarder et l'utiliser ultérieurement, Cela peut être un tremplin pour mener d'autres attaques sur le réseau IoT.

- **Confidentialité et vie privé du TD**

Lors de l'inspection du TD, un utilisateur apprend des informations confidentielles sur l'hôte, telles que la présence d'un système de suivi médical ou d'assistance, le nom du prestataire de soins de santé, etc.

Un utilisateur malveillant peut diffuser publiquement et envoie ces données à un attaquant distant afin de pouvoir lire les profils dans les périphériques installés.

III.4. Intégrité des données du système.

Un attaquant intercepte les requêtes vers l'interface WoT pour définir certains paramètres sur un command à distance, L'interface WoT accepte ces paramètres.

Un exemple d'attaque consiste à rejouer une demande d'interface légitime de WoT pour définir un paramètre, par exemple augmenter la température de la maison de plusieurs degrés par une répétition à plusieurs reprises.

- **Attaque par déni de service dans WoT**

Un attaquant envoie un grand nombre de requêtes à un objet ou à tous les périphériques disponibles sur le réseau IoT. Il avoir une aide par l'interface WoT ou directement cibler le protocole de liaison. Ces requêtes utilisent toute la bande passante de traitement d'un objet ou d'un réseau IoT et empêchent les utilisateurs légitimes de communiquer avec des interfaces WoT ou avec des objets.

III.5. Confidentialité de données du système

Un attaquant utilise des mécanismes pour écouter l'échange d'interface WoT entre des entités légitimes. À partir de cette écoute, l'attaquent acquiert des informations confidentielles sur l'hôte [W7].

IV. Architecture de la plateforme IoT

IV.1. Description de composants de l'architecture

La plateforme qui inclut des objets connectés, utilise les protocoles d'internet (suite TCP/IP) pour relier des différents hôtes et avoir une communication en temps

réel via des technologies Web qui respectent les contraintes de réseau IoT. La figure 09 représente les différents composants de la plateforme :

IV.1.1. Objets connectés (IoT)

Les objets IoT sont connectés directement à l'internet. Ils utilisent la technologie WebSockets pour communiquer en temps réel, respectivement, avec l'*Opérateur1* et *l'Opérateur2*.

IV.1.2. Réseaux locaux :

Ils rassemblent aux réseaux (*LAN1* et *LAN2*) qui offrent une adresse privée pour l'*Opérateur1* et l'*Opérateur2*, et interdisent la connexion à travers des ports non autorisés par les règles de pare-feu.

IV.1.3. Serveur Web

Le serveur central fournit une application Web pour identifier les composants de la plateforme et gérer les différentes technologies, services et APIs utilisées**.**

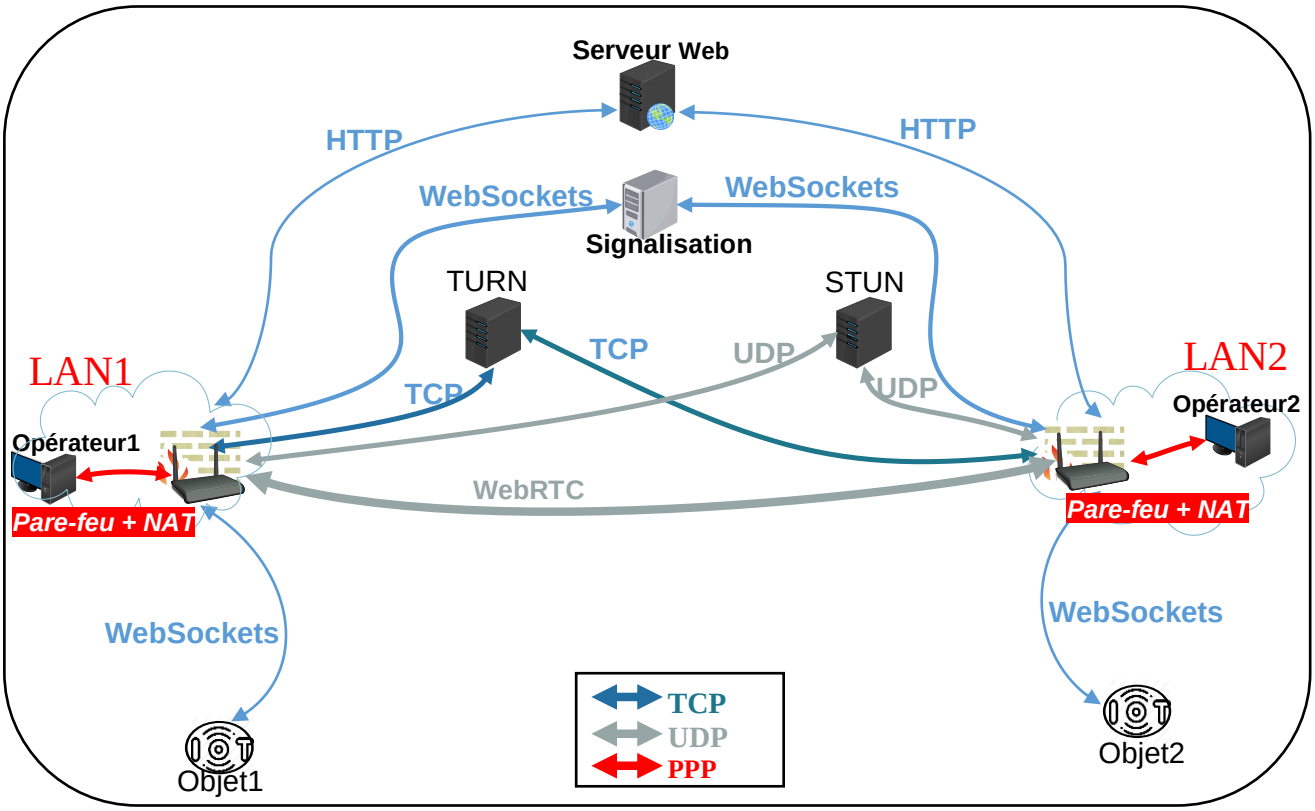

Figure 09 : Architecture fonctionnelle de la Plateforme de contrôle des IoT

IV.1.4. Serveurs du WebRTC

Le WebRTC offre une communication P2P, qui nécessite un serveur pour échanger des informations critiques entre les hôtes et exige la présence des serveurs pour traverser les pare-feu et les NAT.

a. Serveur de Signalisation

Un service de signalisation est nécessaire à la mise en relation des différents hôtes au cours d'une session. Pour établir un canal entre deux navigateurs (initialiser une session WebRTC), il faut que les clients se signalent entre eux. Le serveur permet l'échange de trois types d'information : les messages de contrôle sur la session, les configurations réseaux et les capacités du media du navigateur.

b. Serveur STUN [W8]

Il est utilisé pour permettre à un client situé derrière un routeur NAT de connaître son adresse IP publique et le type de serveur NAT. Le serveur STUN devient non fonctionnel dans le cas d'une NAT symétrique.

c. Serveur TURN [W9]

Il comble certains manques du serveur STUN : ceci permet à des hôtes, situés dans des réseaux équipés de pare-feu et de routeurs NAT (inclus le NAT symétrique), de communiquer en passant par une passerelle.

IV.2. Fonctionnement de la plateforme

L'architecture dans la figure 09 représente la circulation complexe des données, et englobe tous les composants qui permettent de résoudre éventuellement les problèmes de NAT et de pare-feu. Le fonctionnement a trois étapes :

a. Identification du système

Au niveau du serveur Web central, l'application doit identifier chaque composant de la plateforme IoT, et mettre en œuvre une interface pour l'utilisateur qui interagit avec les différents hôtes disponibles via HTTP. Le serveur Web doit contenir des instances TD liés aux deux Objets (Objet1, Objet2) à l'aide d'une base de données, pour faciliter la démarche d'étape suivante.

b. Contrôle de l'objet

Les objets 1 et 2 sont connectés à l'internet à l'aide des APIs WoT, qui supportent la connexion via WebSockets. Cette étape concerne l'établissement de la connexion P2P entre les objets et leurs contrôleurs, comme le montre la figure10.

Figure 10 : Connexion P2P via le Websockets

c. Communication via le WebRTC

Pour créer une connexion P2P entre les deux opérateurs et construire un canal de communication comme représentés dans la figure 11, il doit passer par trois phases :

- **Phase de signalisation :**

Les hôtes, qui participent, doivent initier la session par la création de canal P2P et d'échanger les messages de type offre et réponse à l'aide du serveur de signalisation. Les opérateurs derrière le NAT interagissent avec le serveur STUN pour le traverser via UDP, si le cas d'une NAT symétrique ou autre problème qui empêche la création du canal, l'hôte demande au serveur TURN de le relier à l'extérieur via TCP.

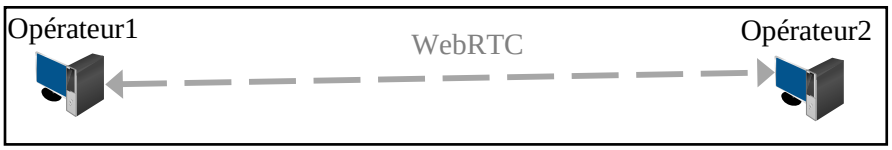

Figure 11 : Canal de Communication P2P entre deux Opérateurs

- **Phase de collecte de données**

L'objectif de cette phase est d'intégrer les données envoyées, par l'objet vers le participant, dans le WebRTC pour les permettre de rejoindre la session. Elle inclut la récupération d'autres media de l'opérateur. La signalisation de données supportées par chaque hôte est nécessaire.

- **Phase de gestion de session**

Une fois que l'établissement de la connexion P2P est prêt, les utilisateurs gèrent

la session entre eux. Ils ont la possibilité d'envoyer des signalisations pour mettre à jour les informations de configuration, mettre la communication en attente ou couper la session, en informant le serveur de signalisation.

V. Scénarios d'attaques liées à la plateforme

V.1 Attaques liés à l'application web

Ce sont les attaques qui ciblent la coche supérieur (La couche Application).

a. Attaque par Injection

Les formulaires, qui sont utilisées pour s'authentifier ou interagir avec l'interface WoT, sont vulnérables par la faille d'injection. Un utilisateur malveillant peut injecter des paramètres dynamiques de l'application Web du côté de l'opérateur, ou du côté des APIs non sécurisés. Ces derniers peuvent mener à la prise de contrôle de l'objet connecté.

b. Violation d'authentification

Comme il est défini dans le chapitre 1 les faibles fonctions d'authentification, au niveau du serveur Web, permettent l'usurpation d'identité. La gestion de session qui est mise en place d'une manière incorrecte, au niveau des serveurs de WebRTC, rend la réplication et la violation de jeton possible. Les attaques suivantes représentent quelques scénarios :

- **Au niveau du serveur Web**

Si l'application Web n'est pas limitée à un usage humain, elle peut être la cible d'un processus automatique qui permet la violation d'authentification tel que l'attaque par la force brute. L'attaquant va bombarder la page d'authentification avec des valeurs d'identifiant et de mots de passe jusqu'à ce qu'il se fasse accepter.

Un autre attaque cible l'identité de l'administrateur du serveur Web, qui peut avoir un mot de passe faible ou par défaut. Il a le droit de gérer la base de données et voir les mots de passe cryptés. Ceci permet à une attaque par dictionnaire de violer n'importe quelle identité.

- **Au niveau de l'opérateur**

Le HTTP est un protocole sans état, donc l'établissement d'une session primitive (première visite d'une page) est nécessaire pour pouvoir faire la différence entre les clients. Elle permet de garder en mémoire sur le serveur chaque client qui visite le site, (identifié ou non). Ceci accepte une attaque *hijacking* si la gestion de session n'est pas bien mise en place.

c. Exposition de données sensibles

- **Au niveau du serveur Web**

Les mots de passe et les descriptions des objets TD sont les données les plus sensibles, et sont stockés dans une base au niveau du serveur Web central. Cette base de données peut être exploitée si elle est vulnérable ou mal configurée, donc des données sensibles peuvent être exposées.

- **Au niveau de canaux WebRTC**

L'attaque par écoute clandestine vise à espionner les canaux du WebRTC, qui utilisent des algorithmes de chiffrement faibles ou un mécanisme d'échange de clés également faible.

d. Violation de contrôle d'accès

Attaque de la manipulation de métadonnées accessible depuis la racine Web, dont la liste de répertoires n'est pas désactivée sur le serveur., telle que la relecture ou l'altération d'un jeton de contrôle d'accès, d'un TD, d'un cookie ou d'un champ masqué pour élever des privilèges ou abuser de l'invalidation du jeton.

e. **Cross Site Scripting (XSS)**

- **Au niveau de l'opérateur**

Des attaquants peuvent utiliser XSS par réflexion et redirigent l'opérateur vers des sites malveillants pour détourner la session en cours de l'utilisateur, ou pour poster les credentials vers une fausse destination. Ces attaques entraînent de piéger l'utilisateur pour envoyer l'ID de session ou le mot de passe vers le site Web de l'attaquant.

L'opérateur peut avoir une attaque XSS basé sur DOM par des APIs incluent de manière dynamique des données contrôlables par un attaquant dans une page

vulnérable. Cependant, l'application Web s'exécute sans avoir examiné ces données contrôlables.

- **Au niveau du serveur Web**

Un attaquent peut exécute un code XSS pour le stocké dans la base de données, qui peut être exécuté à distance par le navigateur d'un opérateur. Cette attaque est de type XSS stocké, permet de livrer de logiciel malveillant.

V.2 Attaques liés à WebRTC

- **Au niveau du serveur STUN**

Une attaque qui s'appelle *Eavesdropping* (écoute) oblige le client à utiliser une adresse réflexive qui se dirige vers l'attaquant.

Un client non autorisé peut utiliser un serveur STUN comme réflecteur, en l'envoyant demandes avec une adresse IP source et un port falsifiés. Dans ce cas, la réponse serait transmise à cette adresse IP et à ce port source. Dans cette attaque, l'attaquant oblige le client à utiliser un réflexe adresse qui route à lui-même. Il transmet ensuite tous les paquets qu'il reçoit au client. Cette attaque permettrait à l'attaquant d'observer tous les paquets envoyés au client.

- **Au niveau du serveur TURN**

Le protocole TURN se préoccupe principalement de l'authentification et de l'intégrité des messages. La confidentialité n'est qu'une préoccupation secondaire, car les messages de contrôle TURN n'incluent pas d'informations particulièrement sensibles. La même attaque précédente (*Eavesdropping*) peut exploiter sans le besoin d'être placé l'attaquant au milieu, la figure 12 représente ce scénario.

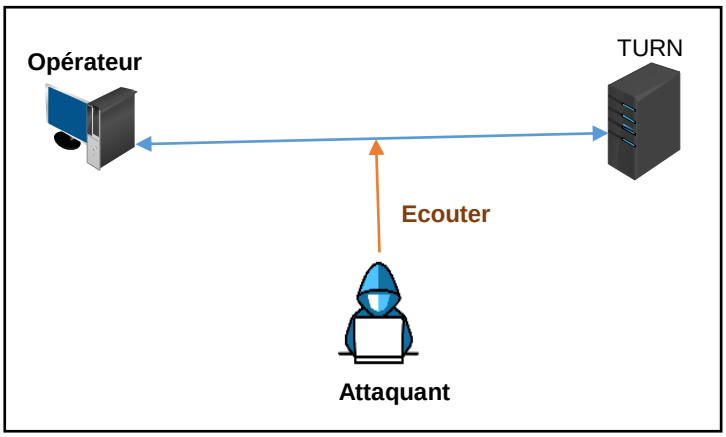

Figure 12 : Attaque par écoute au niveau de serveur TURN

- **Au niveau du serveur de Signalisation**

Une attaque par suivie des appels (Call tracking) peut résulter la récupération des données sensible sur l'appel de WebRTC. Elle permet de savoir qui communique, à quelle heure, et pendant combien de temps. Ces informations peuvent être employées pour mettre en place une autre attaque comme l'écoute par clandestine.

V.3. Attaques liés aux protocoles de WoT

V.3.1. Protocole WebSockets

Ce protocole ne prescrit aucun moyen particulier permettant aux serveurs d'authentifier les clients lors de l'établissement de la liaison WebSockets.

Cela permet effectivement à l'attaquant de notre scénario de lire l'entête des victimes transmises via la connexion WebSockets et de les mettre à jour en émettant des demandes d'écriture via cette connexion (attaque CSWSH [W10]). Cela est possible du fait si le code WebSockets du serveur repose sur les données d'authentification de session (cookies ou authentification HTTP) envoyées par le navigateur pendant la phase handshake/upgrade de WebSockets.

V.3.2. Protocole CoAP

CoAP utilisé par des appareils IoT dans une surface WoT, ce qui signifie qu'un attaquant peut utiliser des appareils vulnérables pour générer une grande quantité de paquets UDP facilement. Les périphériques ne savent pas comment traiter ces données, ce qui en fait un déni de service.

VI. Solution général

VI.1. Prévention d'injection

OWASP définit la prévention d'injection en trois règles :

a. Validation de la saisie

Effectuer une validation d'entrée correcte. Une validation d'entrée positive ou « liste blanche » avec une canonisation appropriée est également recommandée, mais ne constitue pas une défense complète, car de nombreuses applications nécessitent des caractères spéciaux dans leur entrée.

b. Utilisation d'une API sécurisée :

L'option privilégiée consiste à utiliser une API sécurisée qui évite l'utilisation de l'interpréteur ou fournit une interface paramétrée. Faites attention aux API, telles que les procédures stockées, qui sont paramétrées, mais peuvent toujours introduire une injection.

c. Échappement des données utilisateur

Si aucune API paramétrée n'est disponible, il doit soigneusement échapper les caractères spéciaux à l'aide de la syntaxe d'échappement spécifique de l'interpréteur.

VI. 2. Authentification sécurisée

- **Implémenter des contrôles de mot de passe**

L'application doit appliquer des règles pour augmenter la complexité des mots de passe. Ces mécanismes doivent permettre à pratiquer tous les caractères saisis par l'utilisateur et de faire partie de leur mot de passe compliqué.

- **Implémenter un mécanisme de récupération de mot de passe sécurisé**

L'application doit disposer d'un mécanisme sécurisé permettant à un utilisateur d'accéder à son compte s'il oublie son mot de passe.

- **Stocker les mots de passe de manière sécurisée**

Il est essentiel pour une application de stocker un mot de passe à l'aide de la technique cryptographique appropriée.

- **Transmettre les mots de passe uniquement via TLS ou DTLS**

La non-utilisation de TLS ou d'un autre moyen de transport puissant pour la page d'arrivée de connexion permet à un attaquant de modifier l'action de formulaire de connexion, entraînant la publication des informations d'identification de l'utilisateur à un emplacement arbitraire.

- **Envisager une authentification de transaction forte**

Certaines fonctionnalités doivent utiliser un deuxième facteur pour vérifier si un utilisateur peut effectuer des opérations sensibles

- **Authentification et messages d'erreur**

Il devrait que les messages d'échec d'authentification n'indiquer pas le statut d'un compte existant.

- **Prévenir les attaques par force brute**

Étant donné que le but du système de verrouillage du mot de passe est de protéger contre les attaques brutales, une stratégie judicieuse consiste à verrouiller les comptes pendant une période donnée.

- **Enregistrement et surveillance**

Activer la journalisation et la surveillance des fonctions d'authentification pour détecter les attaques ou les échecs en temps réel.

VI. 3. Sécurité de WoT

a. Bon pratiques de sécurité pour les TDs

- **Livraison et stockage sécurisée**

Les TD ne doivent pas être fournis sans authentification préalable du demandeur et vérification de son autorisation d'accès aux TD demandés.

Lorsqu'un TD est stocké sur l'équipement terminal, dans une passerelle (par exemple, un cache) ou dans un stockage distant (par exemple, une archive), son authenticité et la confidentialité doivent également être protégées à l'aide des meilleures méthodes locales disponibles.

- **Réduction au minimum les informations en TD**

Les TD de WoT devraient minimiser la quantité d'informations disponibles publiquement sur l'objet. Par exemple, ne doit pas identifier la version du logiciel ou du système d'exploitation utilisé par un périphérique.

- **Limiter l'exposition des TDs**

Limitez le nombre de clients pouvant obtenir une certaine TD via des méthodes d'authentification et d'autorisation.

b. Bon pratiques de sécurité pour l'interface WoT

- **Utiliser un modèle de contrôle d'accès appropriés**

Utilisez les options de métadonnées de sécurité pour configurer les méthodes d'authentification et d'autorisation appropriées pour les interfaces WoT. Minimisez le nombre d'interfaces WoT sans aucun contrôle d'accès défini (y compris les événements publics). Considérer différents niveaux d'accès pour différents utilisateurs.

- **Limiter les fonctionnalités sans authentification**

Toute interface réseau exposée publiquement doit éviter tout traitement lourd. Les interfaces WoT ne doivent fournir que la fonctionnalité minimale nécessaire. Cette recommandation permet de prévenir l'attaque DoS.

IV. Conclusion

Dans ce chapitre nous avons exposé une brève revue sur les problèmes de sécurité liés à la plateforme IoT. Ceci confirme que le développement des technologies Web doit impérativement être suivi du développement de sa sécurité. Le chapitre suivant concerne une solution particulière pour sécuriser l'identité et limiter l'accès.

Chapitre 03

Solution pour sécuriser l'accès multiple à la plateforme IoT

I. Introduction

Pour comprendre la puissance et l'importance de la gestion d'identité dans le cadre de la plateforme IoT et WebRTC ; nous verrons comment la solution a été conçue en suivant le modèle de gestion centralisée d'identités que nous avons intégré via le Web. Nous aborderons en détails les méthodes utilisés dans la conception pour assurer un niveau optimal de sécurité.

II. Description générale des solutions

La multiplicité des serveurs dans la plateforme implique le besoin d'engendrer un serveur de confiance commun, pour limiter l'accès publique, garantir l'intégrité de travail collaboratif pour la partie externe et identifier les entités de système.

Le serveur ajouté est un fournisseur d'identité **IdP**, il sert à identifier un utilisateur chez tous les serveurs interne de la plateforme selon le Modèle d'*Identité unique SSO* (cité dans le premier chapitre).

On a choisi d'utiliser un algorithme de chiffrement symétrique dans la construction de l'identité numérique, la clé de vérification a étais déclarer explicitement dans les autres serveurs. Ce choix permet à un serveur de renouveler l'identité numérique sans besoin de revenir à l'IdP en cas d'une session prolongé.

La gestion d'identité pour les objets connectés faite par son propriétaire, ce dernier doit s'occupe la clé de chiffrement pour pouvoir communiquer avec l'objet, le serveur utilise un chiffrement asymétrique pour générer un jeton temporaire, et seulement l'objet qui peut le vérifier avec sa clé privée au niveau de l'interface WoT.

III. Conception orienté objet

Notre conception consiste à définir un modèle UML, qui permet de montrer la gestion d'identité légal dans l'ensemble du système WebRTC et la plateforme IoT, grâce à une série de bon pratique et d'autre méthodes de sécurité Web.

III.1. Cas d'utilisation

Ce premier diagramme du modèle UML représente la structure des grandes fonctionnalités nécessaires aux acteurs du système.

III.1.1 Générale

Les personnes qui interagir avec le système doivent s'authentifier pour pouvoir accéder aux grandes fonctionnalités de système, le diagramme générale montre que toute personne non authentifier est un visiteur.

L'authentification et l'inscription sont disponibles au niveau du serveur fournisseur d'identité IdP. Si un visiteur, ou un utilisateur non authentifier, veut accéder à un fournisseur de service SP, il doit se redirige automatiquement vers le fournisseur d'identité.

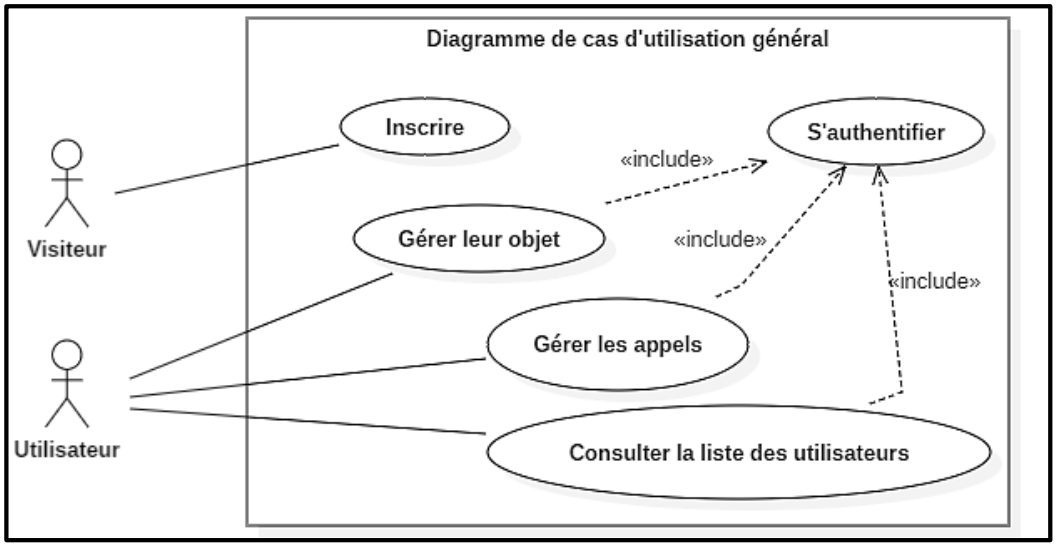

Figure 13 : Diagramme de cas d'utilisation générale

III.1.2 Cas d'inscription

La seule fonction accessible par un visiteur est l'inscription pour avoir un nouveau compte. Cette fonction nécessite d'entrer un nom d'utilisateur valide, qui exige de déclencher un teste et interagir avec la base de donnée avant de valider l'opération, cette vérification permet d'entrer un nom d'utilisateur valide qui assure une inscription légale.

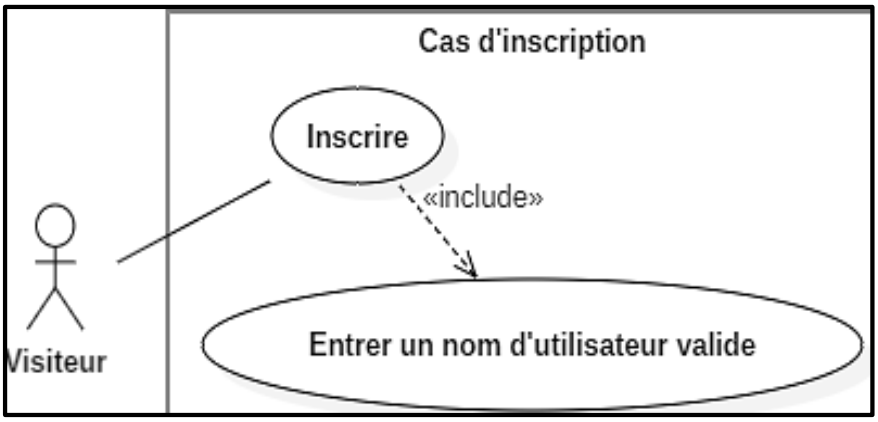

Figure 14 : Diagramme de cas d'utilisation : Inscrire

III.1.3 Gestion de l'objet connecté

L'objet connecté doit pouvoir générer un fichier de l'extension pem, qui comporte la clé de chiffrement. On a proposé de ne pas faire circuler cette clé dans le réseau à cause de l'impact d'une attaque liée à la violation de cette clé. L'ajout ou le changement de l'état de l'objet requiert une communication avec l'interface WoT.

L'initialisation de l'objet n'est qu'une opération préalable pour permettre à l'utilisateur d'interagir en temps réel avec leur objet pendant un appel WebRTC avec un autre utilisateur.

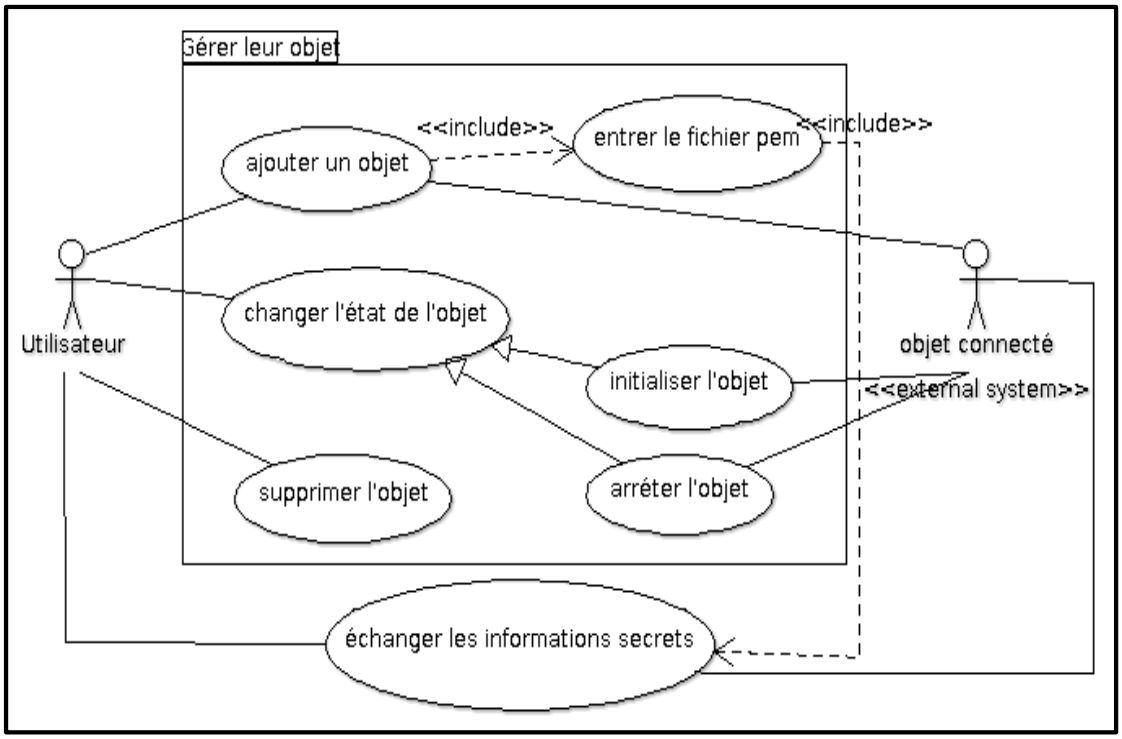

Figure 15 : Diagramme de cas d'utilisation : Gestion d'IoT

III.1.4 Gestion des appels

Le principe des appels est simple, il suffit de choisir un utilisateur d'état en ligne d'après la liste pour pouvoir déclencher un appel WebRTC. De l'autre côté l'utilisateur appelé peut répondre ou ignorer l'appel arrivé.

Si un utilisateur n'ajoute pas un objet, ou l'état de ce dernier est mise en repos, alors la communication avec WebRTC n'inclura pas le contrôle d'objet à distance, du côté de cet utilisateur.

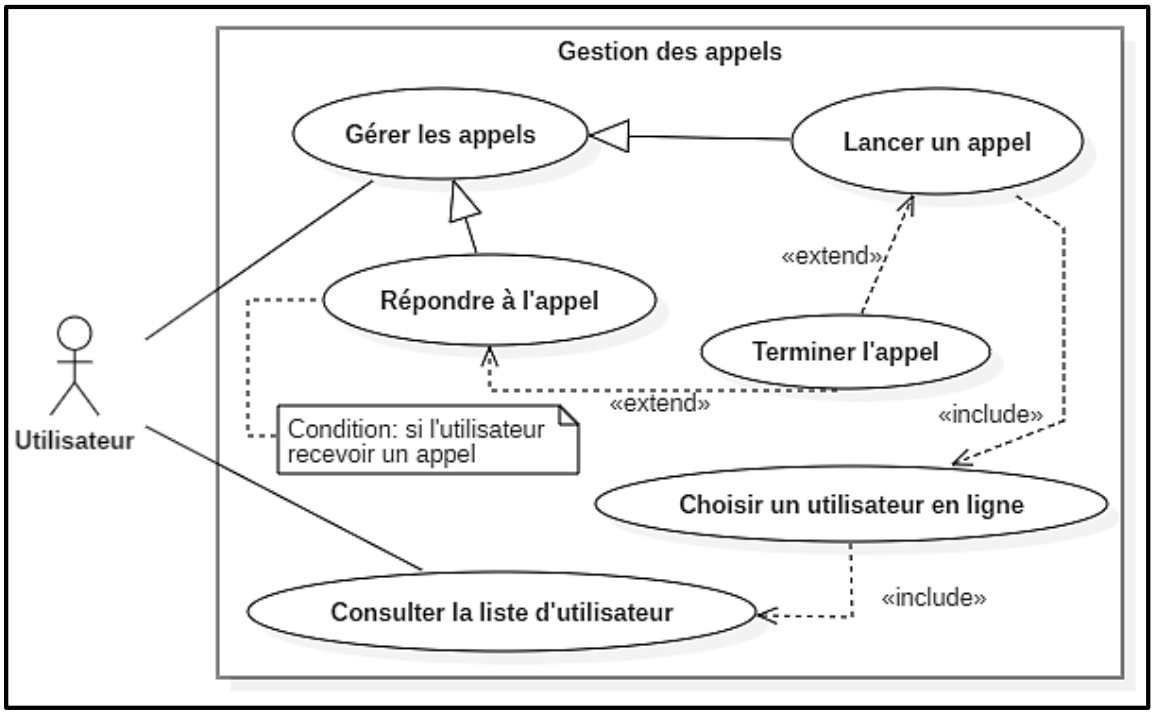

Figure 16 : Diagramme de cas d'utilisation : Gérer les appels

Le diagramme de cas d'utilisation celui où s'assure la relation entre l'utilisateur et les objets que le système met en œuvre. Le diagramme suivant démontrer les relations conceptuelles du système et de données.

III.2 Diagramme ER (Enttity Relationship Diagram)

L'inscription et l'ajout de l'objet sont des cas d'utilisation rassemble à des opérations de construction de deux entités, qu'ils sont l'utilisateur et l'objet connecté. Ce sont les seules entités que le système utilise durant tous les opérations.

L'Utilisateur s'occupe d'une clef $username$, qui est le nom d'utilisateur en anglais, d'un mot passe qui falloir être hashé après d'être concaténer avec une valeur fixe au début et à la fin, en respectant les bonnes pratiques cité en chapitre précédent.

L'Objet identifier par son nom, et chaque entité Objet relie à son propriétaire par leur nom d'utilisateur. La clé de chiffrement, entrer pendant l'ajout de l'objet, est laquelle qui prouve l'utilisateur propriétaire, donc elle ne doit pas être partagé dans le réseau.

Le dernier attribut est TD (Thing Description), rassemble à un fichier JSON (converti en base64), qui contient les informations de l'objet connecté, parmi eux

une URI qui permet à l'utilisateur de savoir comment il va contacter l'objet via le WebSockets. Nous allons voir comment récupérer le fichier TD à l'aide de diagramme de séquence.

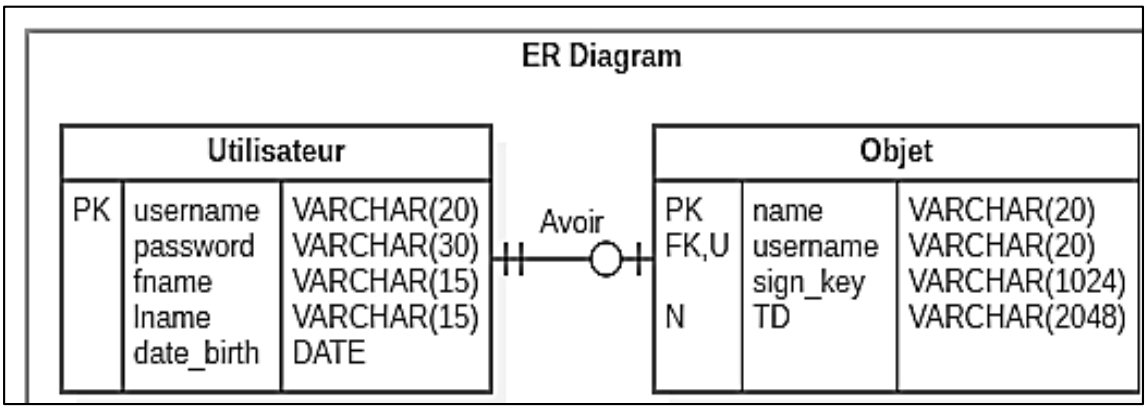

Figure 17 : Diagramme entité-relation

III.3 Diagramme de séquence :

Le diagramme de séquence concerne de représenter le déroulement des cas importants dans un ordre chronologique. Ceci nous permet de proposé des fonctions qui assurent la sécurité.

III.3.1 Vérification de nom d'utilisateur

L'absence de confiance entre le système et le visiteur doit limite les opérations d'interaction entre eux, surtout les actions qui fait appelle à la base de données. En conséquence : chaque cas d'utilisation de serveur IdP (s'authentifier, s'inscrire ou tester le nom d'utilisateur) doit pouvoir vérifier le nombre de tentative qui lui rassemble pour chaque visiteur.

L'inscription n'est qu'une insertion, d'un nouvel utilisateur dans la base de données, après la validation du forum HTML, ceci n'impose aucune fuite d'informations.

Contrairement à l'action qui précède l'inscription, qui représente le lancement de méthode POST avec Ajax pour vérifier le nom d'utilisateur. Si le visiteur entre cinq noms valides sans qu'il soit enregistré, le serveur doit arrêter de lui fournir le service d'inscription.

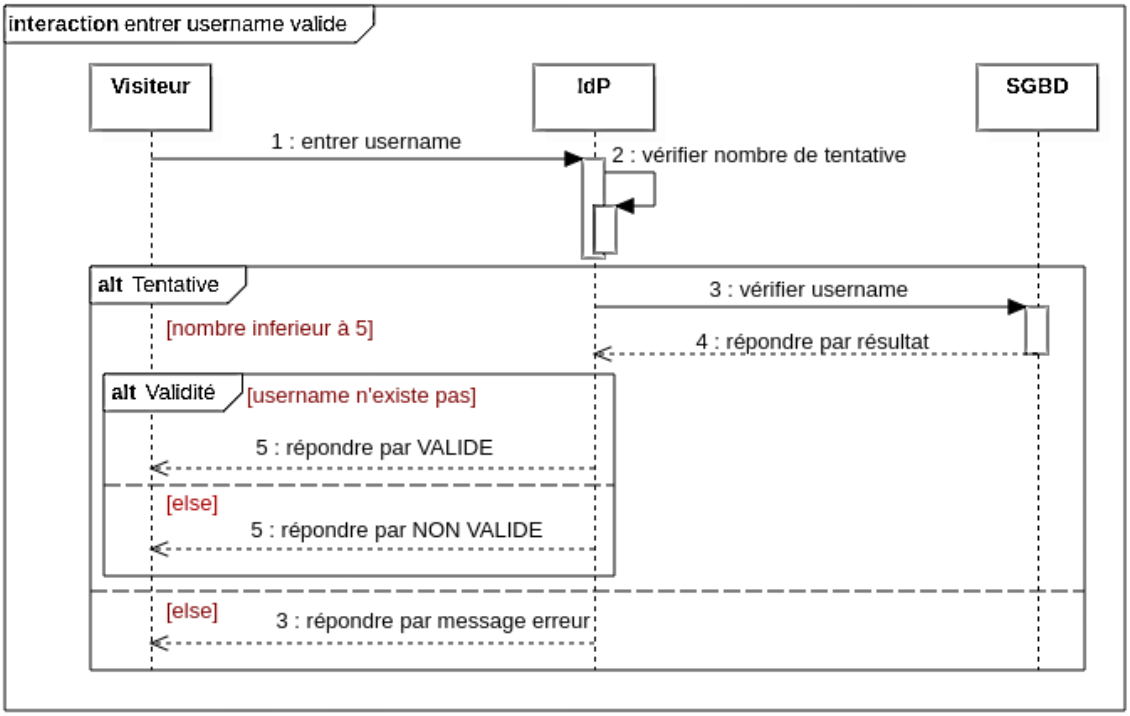

Figure 18 : Diagramme de séquence : Test de nom d'utilisateur entré

III.3.2 Authentification

La méthode sécurisé pour garder l'authenticité d'un opérateur est d'utiliser les cookies avec une bonne pratique (accessible avec HTTPS seulement). Les cookies peut être configuré pour devenir accessible à partir des serveurs de même domaine (et différents sous-domaines). Ceci permet de à deux serveurs (IdP et SP) de garder l'authentification de l'opérateur pendant l'authentification.

La deuxième méthode, qui est moins sécurité de première, est de définir une règle CORS (Cross-Origin Resource Sharing) dans le serveur IdP, pour qu'un opérateur de domaine extérieur puisse envoyer une requête d'authentification sécurisé (la méthode POST sous HTTPS), et de placer le Jeton obtenu, à partir de réponse, dans autre requête d'autorisation [W11] (d'utiliser la méthode *bearer* au niveau de header), destiné vers le serveur SP. Ces deux requêtes doivent utiliser une méthode asynchrone (comme Ajax) pour que le navigateur ne soit pas actualiser.

On propose d'utiliser la méthode suivante pour garder l'authenticité de l'opérateur en protégeant le mot de passe de toute extérieure utilisation.

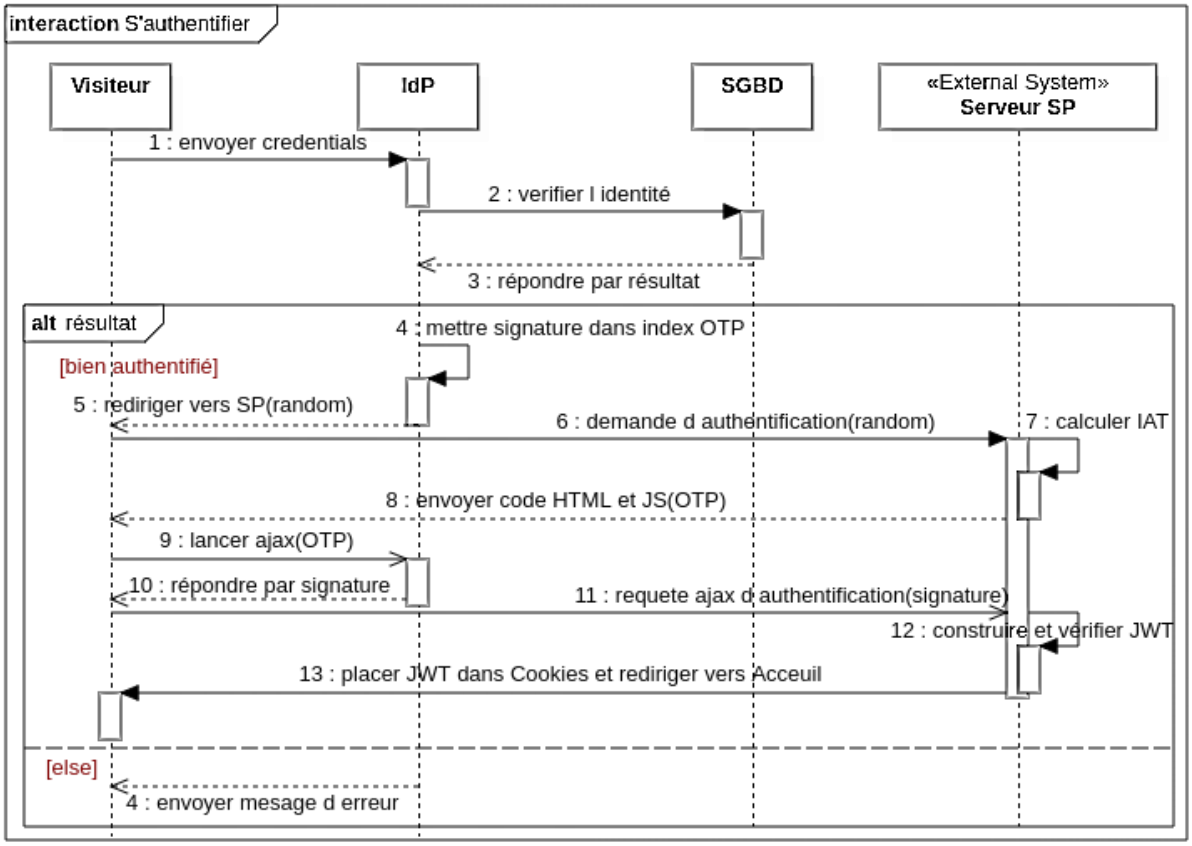

Figure 19 : Diagramme de séquence : Authentification

- **"Challenge–response" pour l'authentication**

Le serveur IdP utilise un défi pour sécuriser l'authenticité et le transmettre de Jeton sans exposé de vol. Pour cela on utilise un système OTP basé sur un nombre aléatoire et le temps instantané en seconde. Voici les rôles de chaque partie.

➢ **Serveur IdP :** Les étapes suivant montre le rôle de fournisseur d'identité.

1. La Création d'OTP selon l'instruction suivant :

$$OTP = fonction\_md5\,(instant + nb\_aleatoire)$$

2. L'OTP généré utilisé comme un index d'un tableau temporaire, ce dernier garde la signature de JWT seulement dans cet index, pendant une durée très petit (5 seconds par exemple).

3. Le serveur permet à un domaine extérieur spécifié de demander une valeur dans le tableau en fonction de son index.

4. Les demandes réussies doit détruire, à partir du tableau, la valeur obtenue.

5. Le serveur doit limiter le nombre de tentatives surtout pour les demandes échoueés sans perdre le fonctionnement normal du service.

- **Serveur SP :** Le fournisseur de service doit pouvoir accéder aux informations qui représentent le corps de Jeton, on distingue deux types d'information :

a. Information statique : Ce sont les informations qui été stocké dans la base de données, concerne l'utilisateur, sans inclus de mot de passe. Plusieurs méthodes sécurisées sont disponibles, parmi eux est d'utiliser le concept de Vue [W12].

b. Information dynamique : Deux attribues calculables sont utilisés dans le corps de JWT : le temps de création IAT, et le temps d'expiration EXP, du jeton. La première information permet de connaître la dernière. Voici les étapes nécessaires pour calculer IAT.

1. Utiliser le même service NTP pour les deux serveurs.

2. Envoyer les deux derniers chiffres d'IAT (Issue At Time) avec le nombre aléatoire par une simple concaténation, à partir de serveur IdP vers le serveur SP.

3. Utiliser ces deux chiffres comme un vecteur de correction dans la fonction de la figure 20, pour obtenir la valeur exact de l'attribut iat.

```
Fonction CorringerIAT(vecteur) : entier
Variable    IAT, cIAT : entier
Début
   IAT = instant( )
   cIAT = concatiner(IAT, vecteur)
   Si (IAT Mod 100 < vect)
   Alors Si ( vect – (IAT Mod 100) > 10 )
         Alors cIAT += 100 FinSi
   Sinon Si (vect < 10 et IAT Mod 100 > 90)
         Alors cIAT −= 100 Finsi
   FinSi
   Retourner ( cIAT )
Fin
```

Figure 20 : Fonction Corriger IAT

- **Opérateur :** L'opérateur doit pouvoir utiliser JavaScript, la méthode AJAX et une connectivité que ne dépasse pas le délai d'authentification pendant la redirection entre les serveurs.

III.3.3 Lister les utilisateurs

La communication HTTP marche en parallèle avec autre communication via le WebSockets, ce dernier permet de déduire l'état des utilisateurs (en/hors ligne), de lancer des appels, et de créer des canaux de signalisations.

Le protocole WebSockets ne gère pas l'autorisation ou l'authentification, qui exige le serveur d'envoyer le jeton dans la réponse HTTP, pour être utiliser dans le premier message de WebSockets à partir de client.

Le serveur donc gère le premier message d'authentification, vérifier le jeton, et déconnecter automatiquement dans le cas d'un jeton invalide ou ne reçoit aucun message d'authentification après 5 seconds de leur connexion.

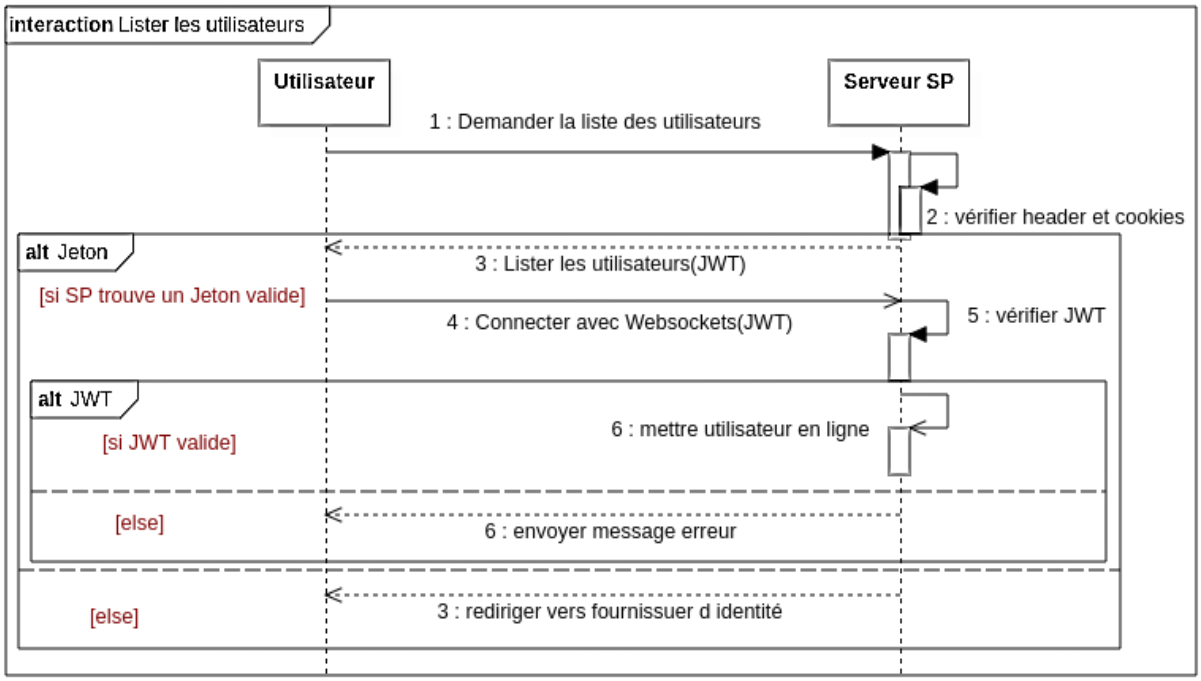

Figure 21 : Diagramme de séquence : consulter la liste des utilisateurs

III.3.4 Gestion d'IoT

Après la mise en place de l'objet via le propriétaire, et après la récupération de la clé de chiffrement, l'utilisateur peut contrôler l'objet à l'aide du serveur. Ce dernier est le responsable de communication COAP avec l'interface WoT de l'objet. Par contre dans la communication WebSockets, le serveur fourni le code HTML et JAVASCRIPT à l'utilisateur pour attendre une communication P2P.

Nous nous intéressons aux communications COAP qui sont déroulent comme le diagramme suivant :

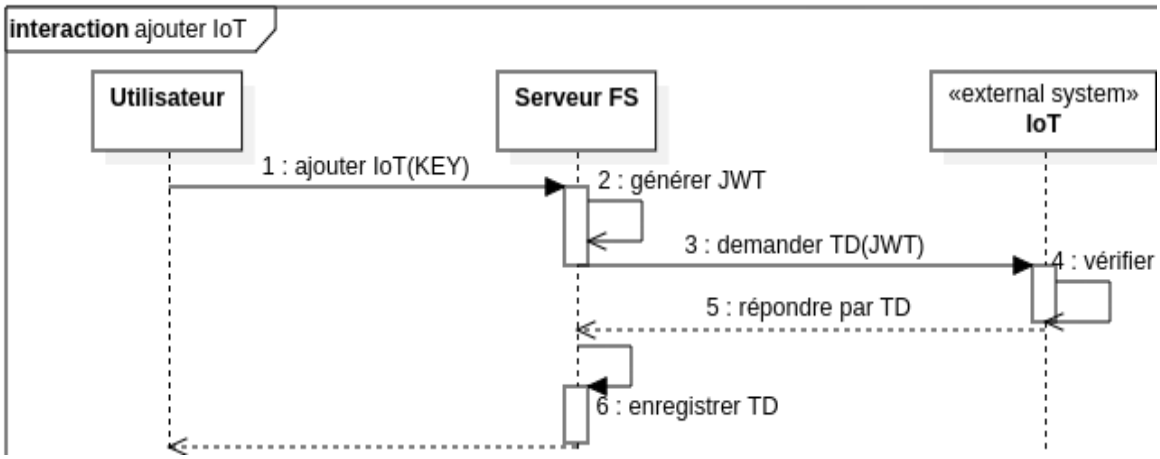

Figure 22 : Diagramme de séquence : Ajouter IoT

Dans les autres cas d'utilisation, le changement est dans la nature d'action, avec la même succession, par exemple : Si le cas de l'initialisation de l'objet, qui remplace la première action (1 : ajouter IoT), donc l'URI remplace le TD dans 3, 5 et 6. L'URI est le nouveau lien pour permettre d'utiliser le WebSockets après l'initialisation.

L'interface WoT de tous les objets inscrit dans le système doit accepter les requêtes COAP d'après l'adresse de serveur SP seulement pour avoir une sécurité élevée.

III.3.5 Gestion des Appels

Le diagramme suivant montre tous les messages de WebSockets géré par le serveur pour lancer ou recevoir un appel. L'action 12 (création d'appel) est responsable de la phase de signalisation (entre deux utilisateurs) cité dans le chapitre précédent.

Le serveur SP gère l'authentification de tous les sockets également comme la figure23, et fourni le code contient l'adresse des serveurs (STUN et TURN) confiances, avec le mécanisme de sécurité qui les rassemble.

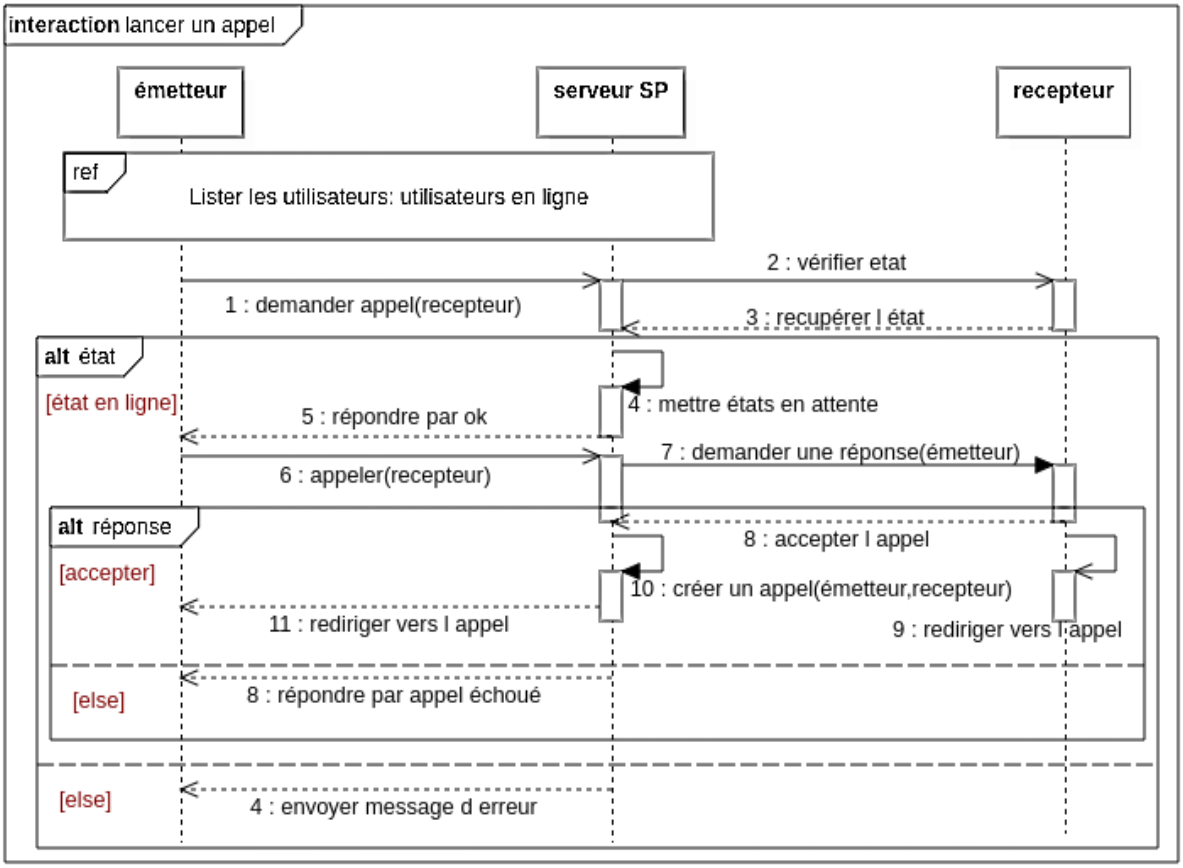

Figure 23 : Diagramme de séquence : Gestion d'appel

III.4 Diagramme de classe :

Pour un utilisateur authentifié, le serveur SP offre deux services principaux :

- Interaction avec des utilisateurs : La possibilité de consulter la liste d'utilisateurs et de recevoir leurs états en temps réel, plus la gestion des appels.
- Interaction avec un IoT : La possibilité d'accéder à l'interface WoT d'un objet approprié, plus le contrôle des outputs de cet objet.

La figure suivant montre une représentation abstraite des entités du système et leurs relations indépendantes pour réaliser les cas d'utilisation.

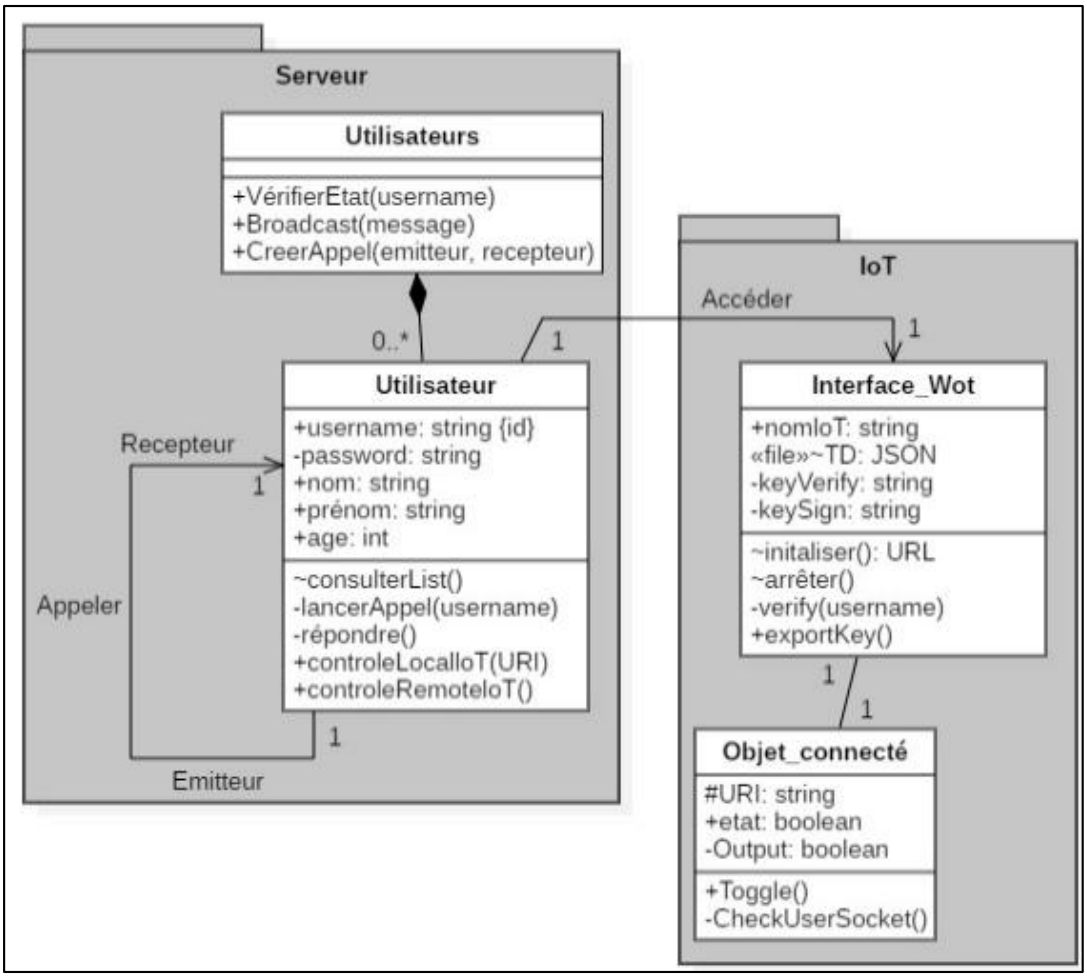

Figure 24 : Diagramme de classes

IV. Analyse de sécurité

IV. 1 Authentification

Deux types de protocole utilisent l'authentification :

- **HTTP**

L'échange de jeton d'authentification en utilisant notre amélioration de type *"Challenge–response authentication"* est laquelle qui relier la session HTTP avec le JWT. La bonne pratique pour protéger l'ID de cette session est responsable de la sécurité contre les attaques Hijacking (détournement de session HTTP), mais le serveur ne peut pas assurer cette sécurité pour un utilisateur qui avoir un navigateur non protégé (ne supporte pas le flag HttpOnly par exemple), et pour un utilisateur vulnérable d'un accès physique.

- **WebSockets**

WebSockets accessible pour des utilisateurs authentifiés seulement, qu'ils s'authentifient via le WebSockets lui-même. Cette authentification plus la restriction de CORS épargne le serveur SP d'être une cible des attaques extérieurs comme *Cross-Site WebSocket Hijacking (CSWSH).*

IV.2 Autorisation

Le contrôle d'accès par identité empêche les visiteurs non identifiés d'accéder aux serveurs fourni par l'application à part l'IdP. Également, il fait interdit les utilisateurs non propriétaires de contrôler un objet connecté, ou interagit avec leur interface WoT.

IV.3 Confidentialité

Il est souvent important d'utiliser des protocoles sécurisés garantissant l'authenticité et la confidentialité des données de l'utilisateur du système. Plus précisément, WebSockets Sécurisé (utilisant TLS) et CoAPS (utilisant DTLS). L'échange des informations sensibles via des différents mécanismes, soit par des requetés HTTP ou par une base de données centrale, diminue l'impact des attaques de sniffing.

IV.4 Non répudiation

L'échange de la signature de jeton (sans corps et entête) avec le mécanisme d'OTP qui est basé sur le temps instantané garantissent le non répudiation contre le phishing et les attaques par intermédiaire (MiTM Man in The Middle).

IV.5 Disponibilité

Les interfaces WoT gèrent les requetés CoAP qui arrivent depuis une seule source (le serveur SP) et refuse tous les autres requêtes. L'implémentation de cette règle dans les firewalls des réseaux de contraintes permettent de créer une barrière de sécurité contre les attaques DDoS. Cependant, les fonctions fournissent par l'interface WoT (initialiser et arrêter), mettent l'objet en repos la plupart des temps, cela évite plusieurs type d'attaques.

V. Conclusion

Nous allons vu que plusieurs principes, contraints et méthodes natives, peuvent limitent l'accès au système, et prévenant les attaques Web liés à la plateforme. Il nous reste la validation de cette solution dans le dernier chapitre.

Chapitre 04

Validation de la solution proposée

I. Introduction

L'objectif de ce chapitre est de présenter l'environnement de travail et l'ensemble des outils logiciels utilisés pour la réalisation de notre application Web. Et de démontrer les résultats obtenus au cours de ce travail.

II. Outils et environnement

II. 1. NodeJS [W13]

Node.js offre un environnement construit sur le moteur JavaScript V8 de Chrome, qui permet à un serveur Web de communiquer avec le système à travers de différentes librairies C++ et un langage familier**.** En gros, il vient en remplacement de langages serveur comme PHP, JavaEE, etc. grâce à une approche non bloquante permettant d'effectuer des entrées/sorties (I/O) de manière asynchrone.

- **NPM [W14]**

C'est le gestionnaire de paquets officiel de NodeJS (Node package manager en anglais). Les développeurs utilisent npm pour partager et emprunter des packages, et de nombreuses organisations utilisent également npm pour gérer le développement privé. Il est le plus grand écosystème de bibliothèques open-source au monde. Voici les modules principaux qu'on a utilisés pour réaliser notre projet.

II.1.1. Express [W15]

ExpressJS est un Framework qui nous permet de créer une application Web plus simplement qu'avec l'objet http de NodeJS. Elle fournit un ensemble de méthodes permettant de traiter les requêtes HTTP et fournit un système de middleware pour étendre ses fonctionnalités. Nous choisissons d'utiliser Express grâce à ces trois principes :

a. Routeur

Comme beaucoup de Framework web, ExpressJS se présente comme un routeur où l'on va déclarer les routes supportées par notre application ainsi que le traitement à effectuer lorsque cette dernière est rencontrée.

b. Middlewares

Les middlewares permettent d'effectuer un traitement avant celui défini par les routes. On trouvera de nombreux middleware disponible sur NPM comme par exemple express-session qui permet de gérer les données de session par exemple.

c. Session

Express-session est un middleware qui stocke les données de session sur le serveur ; il ne sauvegarde que l'ID de session dans le cookie lui-même, mais pas les données de session. Il implémente un stockage en mémoire supporté par plusieurs modules basés sur le stockage de session tel que Redis.

II.1.2. EJS [W16]

Embedded JavaScript est un moteur de Template. Il permet de rendre un fichier html dynamique, d'exécuter du code serveur, et accéder à des variables serveur à partir de ce fichier. Et permet également d'utiliser *if, while* ou toute autre commande JavaScript souhaitée.

II.1.3. Socket.io [W17]

Socket.io se base sur plusieurs techniques différentes qui permettent la communication en temps réel. Il nous permet d'utiliser les WebSockets très facilement.

II.1.4. Jsonwebtoken

La bibliothèque *jsonwebtoken* est une implémentation de JWT, il inclut deux méthodes principales : verify pour la vérification de JWT et sign pour la construction.

II.2. SGBD SQL et NoSQL

SQL est de nature relationnelle puisque toutes les données sont stockées dans différentes tables et que les relations sont établies à l'aide de clés primaires ou d'autres clés appelées clés étrangères.

Les bases de données NoSQL sont principalement utilisées dans les applications Big-Data et Web en temps réel. La structure de données utilisée par la base de données NoSQL est très différente de celle utilisée dans une base de données

relationnelle. Certaines opérations sont plus rapides que des bases de données relationnelles telles que MySQL.

II.2.1. Redis

Redis est système de gestion de base de données clef-valeur. Il est très léger et ses types de données lui confèrent un avantage concurrentiel. Il fait partie de la mouvance NoSQL et vise à fournir une base de données en mémoire ou un système de cache de hautes performances, simple à utiliser et hautement évolutif.

II.2.2. MySQL

MySQL est l'un des systèmes de gestion de base de données SQL largement utilisés. Il peut être cité comme l'un des meilleurs SGBDR utilisés pour le développement de diverses applications logicielles basées sur le Web.

MySQL est disponible avec un large éventail d'outils de création de rapports qui aident à la validité des applications, tandis que les bases de données NoSQL manquent d'outils de création de rapports pour l'analyse et les tests de performance.

II.3. Raspberry PI

Le Raspberry pi est un nano ordinateur de la taille d'une carte de crédit que l'on peut brancher à un écran et utilisé comme un ordinateur standard. Il peut être utilisé dans une vaste gamme de projets IoT.

Les projets intégrant cette technologie peuvent être autonomes et utilisés pour créer des objets interactifs indépendants ou opérer un logiciel. Raspberry Pi a la capacité d'interagir avec le monde extérieur grâce à des pins GPIO.

II.3.1. GPIO

Les GPIO utilisables comme entrée/sortie numérique sont au nombre de 26. Ils ne fonctionnent qu'en tout ou rien, 0 ou 1, 0V ou 3.3V. Ils permettent la connexion de cartes d'extension HAT (Hardware attached on top) ou d'autres composants électroniques pour réaliser des montages.

II.3.2. Raspbian [W18]

Raspbian est un système d'exploitation basé sur Debian optimisé pour fonctionner sur un Raspberry Pi. Il est fourni avec plusieurs environnements de

programmation telle que python 2 et 3, et fourni également la possibilité d'installer NodeJS depuis le gestionnaire de paquets.

a. Python

Python est un langage de programmation multiplateforme très rapide et simple de prise en main. C'est un langage installé en standard sur Raspbian.

- **Tornado [W19]**

Tornado est un Framework web asynchrone pour Python qui a sa propre boucle d'événements. Cela lui permet de supporter nativement les fonctionnalités en temps réel comme le WebSockets.

- **GPIOSimulator**

GPIOSimulator est une bibliothèque de python pour simuler certaines des fonctions utilisées dans la bibliothèque RPi.GPIO (qui gère les GPIO de Raspberry), l'intention de cet émulateur est éducative.

III. patron de conception MVC

Nous avons choisi d'utiliser le patron de conception MVC (ou design pattern en anglais) qui est très répandu pour réaliser des sites web. Le but est d'avoir un code bien structuré et organisé sans mélanger des requêtes SQL partout dans le code JavaScript de Node.

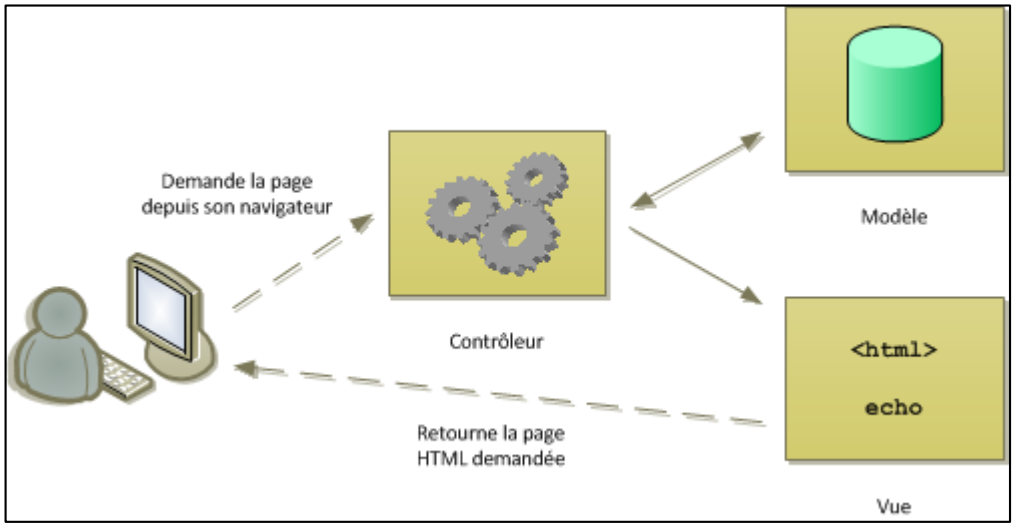

Figure 25 : Échange d'informations entre les éléments MVC [W20]

- **NodeJS avec MVC :**

Nous allons choisir d'utilisé le patron de conception *Modèle-Vue-Contrôleur* pour notre application Web, spécialement dans environnement NodeJS.

a. ***Modèle*** *:* Son rôle est d'aller récupérer les informations brutes dans la base de données, de les organiser et de les assembler pour qu'elles puissent ensuite être traitées par le contrôleur. On y trouve donc les requêtes SQL.

b. ***Contrôleur*** *:* C'est l'intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue. On a utilisé le Framework *Express* pour créer des *Middlewares* comme contrôleurs.

c. ***Vue*** : Cette partie ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu'elle doit afficher. On y trouve essentiellement du code HTML qui est engendré par le Framework EJS (Embedded *JavaScript*).

IV. Interface graphique (Vues)

IV.1. Serveur IdP

La figure 26 montre la page d'accueil du serveur IdP, cette page contient d'un entête et un pied de page, qui sont fixes dans toutes les autres pages. L'entête lui-même fournit d'une barre de menu pour basculer entre les pages.

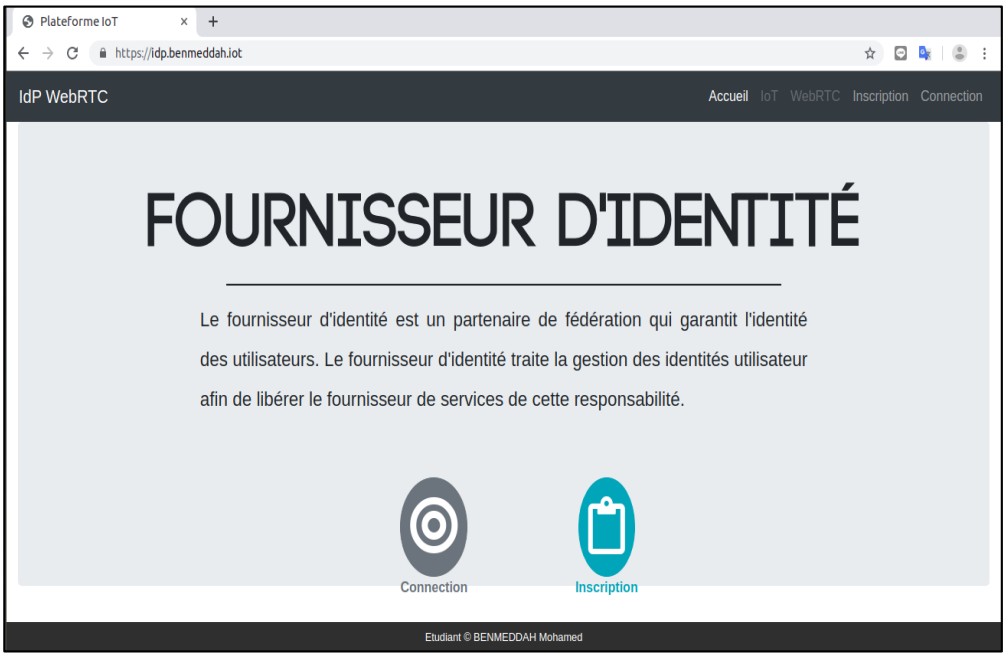

Figure 26 : Page d'accueil de serveur IdP

La page d'inscription fournit un formulaire de six entrés, avec un code JavaScript qui déclenche la requête de vérification du champ *username* sans actualiser la page. La figure 27 montre les cas d'utiliser une valeur valide et non valide.

Figure 27 : Page d'inscription

La page de connexion s'occupe des champs pour entrer les credentials (*username* et *password*) pour authentifier les utilisateurs.

Figure 28 : Page de connexion

IV.2. Serveur SP

Le domaine qui représente le serveur Web change automatiquement après une connexion réussie, c'est une redirection du serveur IdP vers le serveur SP inclut l'échange d'authenticité de l'utilisateur, la figure montre la page d'accueil du serveur SP.

Figure 29 : Page d'accueil de serveur SP

La gestion des objets nécessite la récupération d'un fichier de type *pem*, pour cela nous utilisons Raspbian dans une machine Virtual qui représente l'objet connecté, il génère le fichier *pem* pendant le démarrage d'exécution. Voici l'interface pour ajouter un objet.

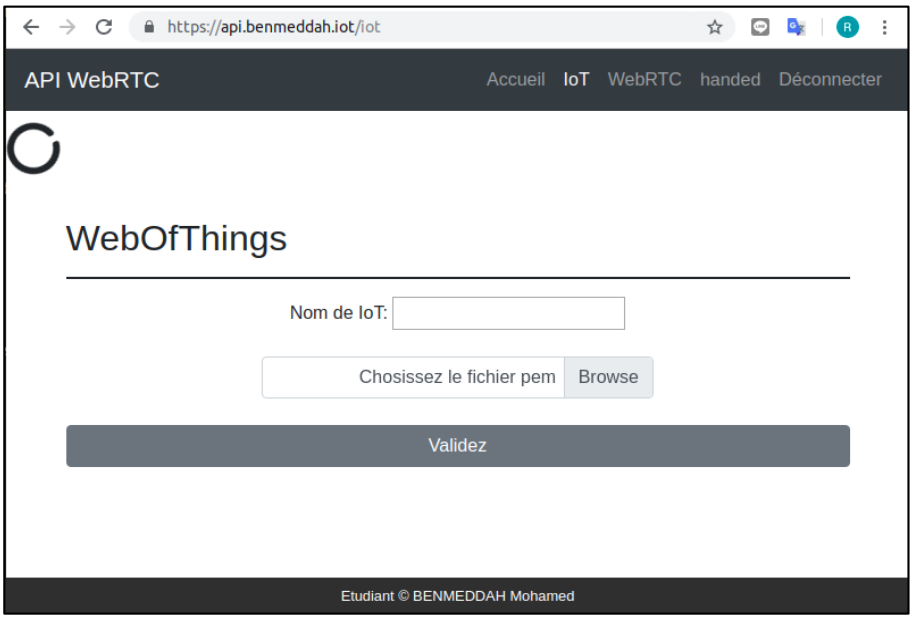

Figure 30 : Page d'ajouter un IoT

L'ajout de l'objet fourni des fonctionnalités d'interaction avec WoT à travers le serveur SP, voici la nouvelle interface pour la gestion IoT.

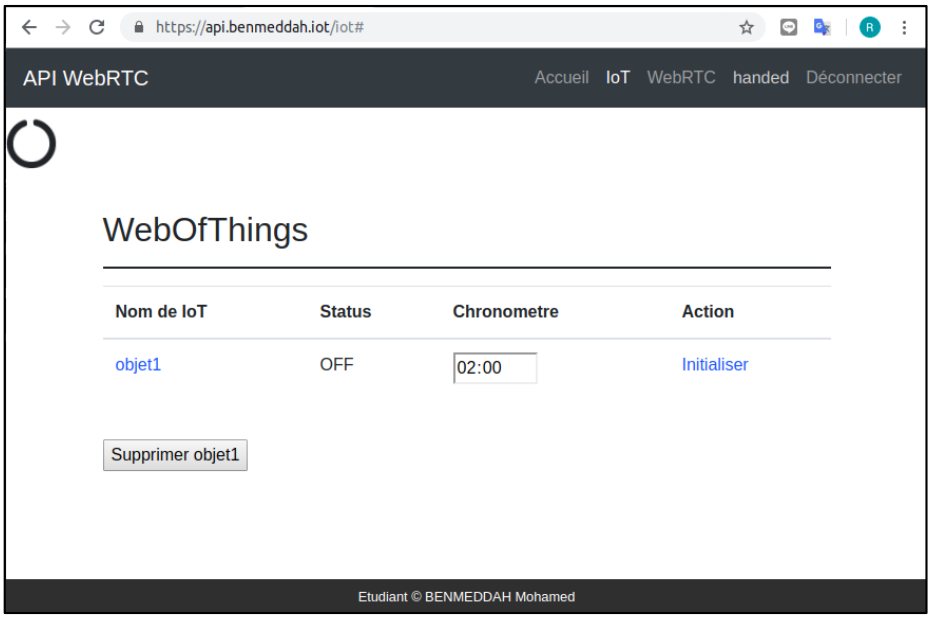

Figure 31 : Page d'un IoT en repos

L'initialisation permet au protocole WebSockets d'accepter la connexion avec l'utilisateur propriétaire. Cette ouverture est limitée par un chronomètre qui peut être arrêté par une clique (montré dans la figure 32).

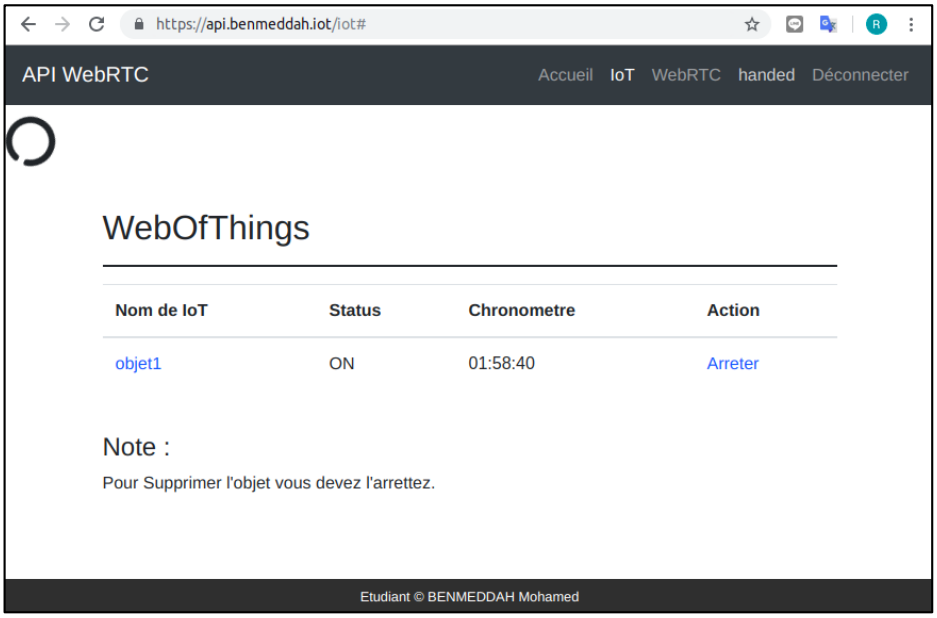

Figure 32 : Page d'un IoT en ligne

L'interface qui concerne WebRTC affiche la liste des utilisateurs avec leurs informations. Cette interface est dynamique, elle permet de sélectionner un utilisateur pour l'appeler.

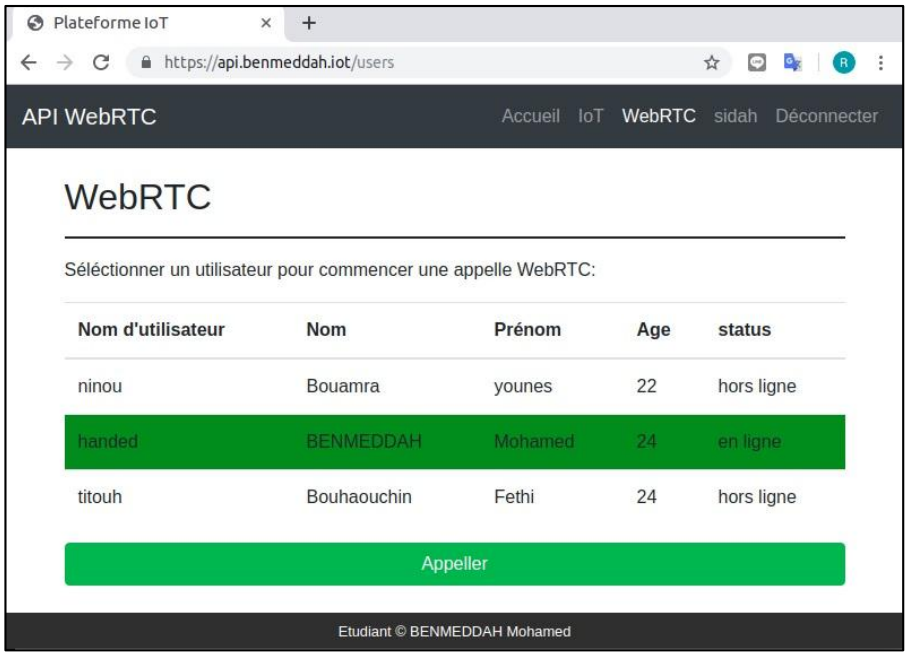

Figure 33 : Page de la liste des utilisateurs

La figure suivant montre une déclanchement d'un appel dans les deux coté (émetteur et récepteur).

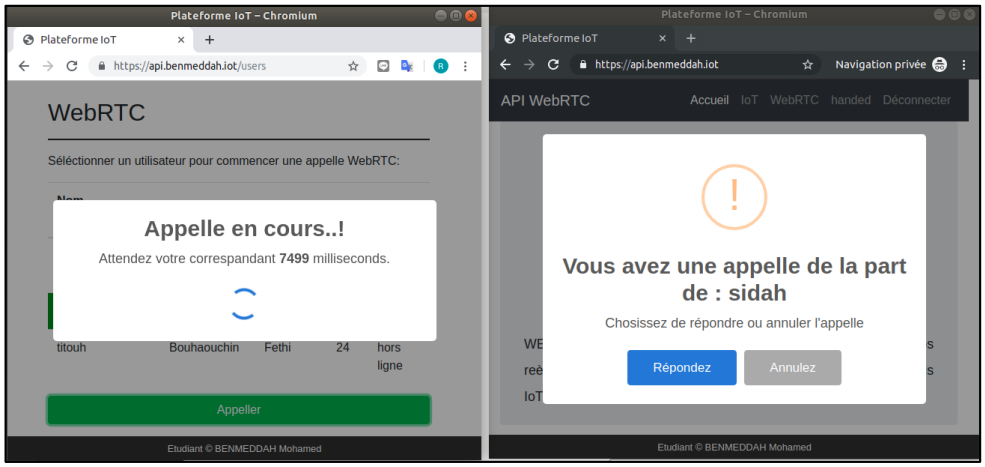

Figure 34 : Fenêtres de lancement et recevoir d'un appel

La page d'appel, dans la figure 35, montre plusieurs fonctionnalités offrir pour deux utilisateurs qui participent au même appel.

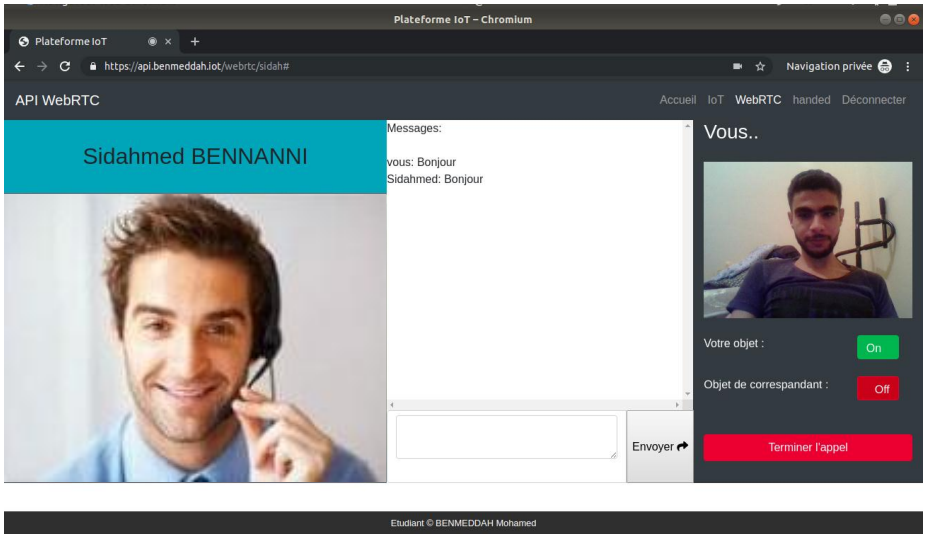

Figure 35 : Interface offre la communication WebRTC et WebSockets

IV.3. Objet Connecté

L'implémentation d'un objet Virtual permet de tester l'application avant de mettre en place un objet réel, grâce à la distribution raspbian qui permit cela, nous serons satisfait de cette virtualisation.

a. Interface WoT

Nous allons utiliser NodeJS pour gérer les requêtes CoAPs arrivés depuis le serveur SP, l'application dans ce cotée offre les fonctionnalités suivantes :

- Générer un couple des clés de chiffrement asymétrique : un pour la vérification des requêtes arrivées et l'autre est exporter dans un fichier.
- Répondre aux requêtes légales et interdire le reste.
- Initialiser et arrêter les communications de WebSockets à travers un lien URI dynamique.

b. WebSockets et GPIO

Pour la gestion de la communication en temps réel(WebSockets) avec GPIO, nous utilisons python3 qui est installé par défaut sur Raspbian. Et qui doit permet à l'objet Virtual de :

- Vérifier la sécurité des sockets arrivés (authentification et CORS).

- Ouvrir et refuser la connexion selon leur état.
- Contrôler GPIO selon les messages arrivés via le protocole WebSockets.

L'interface suivant montre l'état des GPIO avant et après une clique sur le bouton qui contrôle l'objet, au niveau de la page précédente.

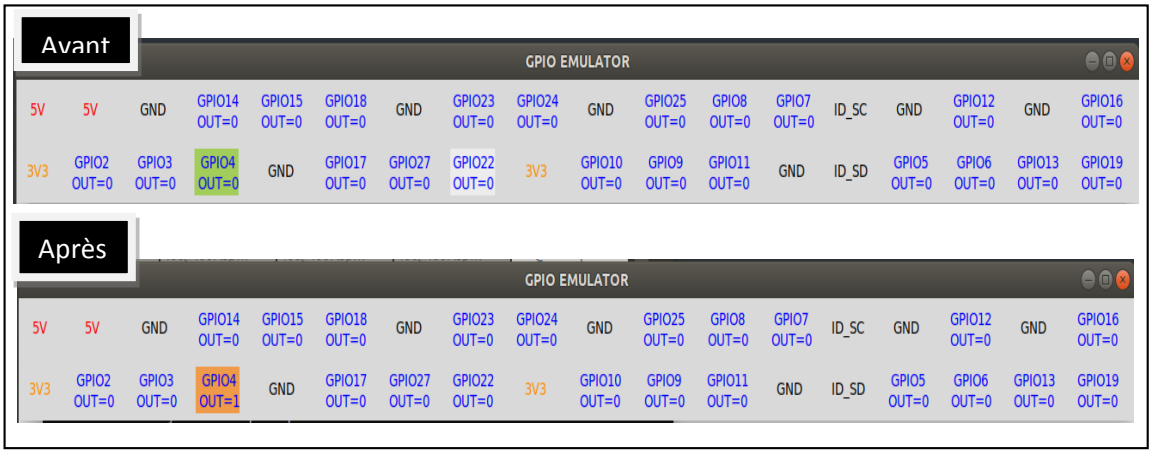

Figure 36 : Interface des GPIO d'un Raspberry virtual.

V. Tests et résultats (contrôleur)

V.1. Serveur IdP

- **Input**

Nous avons protégé les champs des formulaires par des expressions régulières définies dans le code HTML, de coté de l'utilisateur, et dans des middlewares de coté de serveur. La figure suivant montre une entrée non valide depuis le navigateur.

Figure 37 : Contrôle des Entrée via le Navigateur.

- **Fuite d'information**

Les attaquants utilisent tous les techniques pour collecter les informations possibles, donc nous utilisons un seul message d'erreur en cas d'une authentification non valide, et la vérification du *usernames* pendant l'inscription est limitée. La figure montre le déclanchement d'un événement de teste pour chaque changement au niveau de champ du *username*.

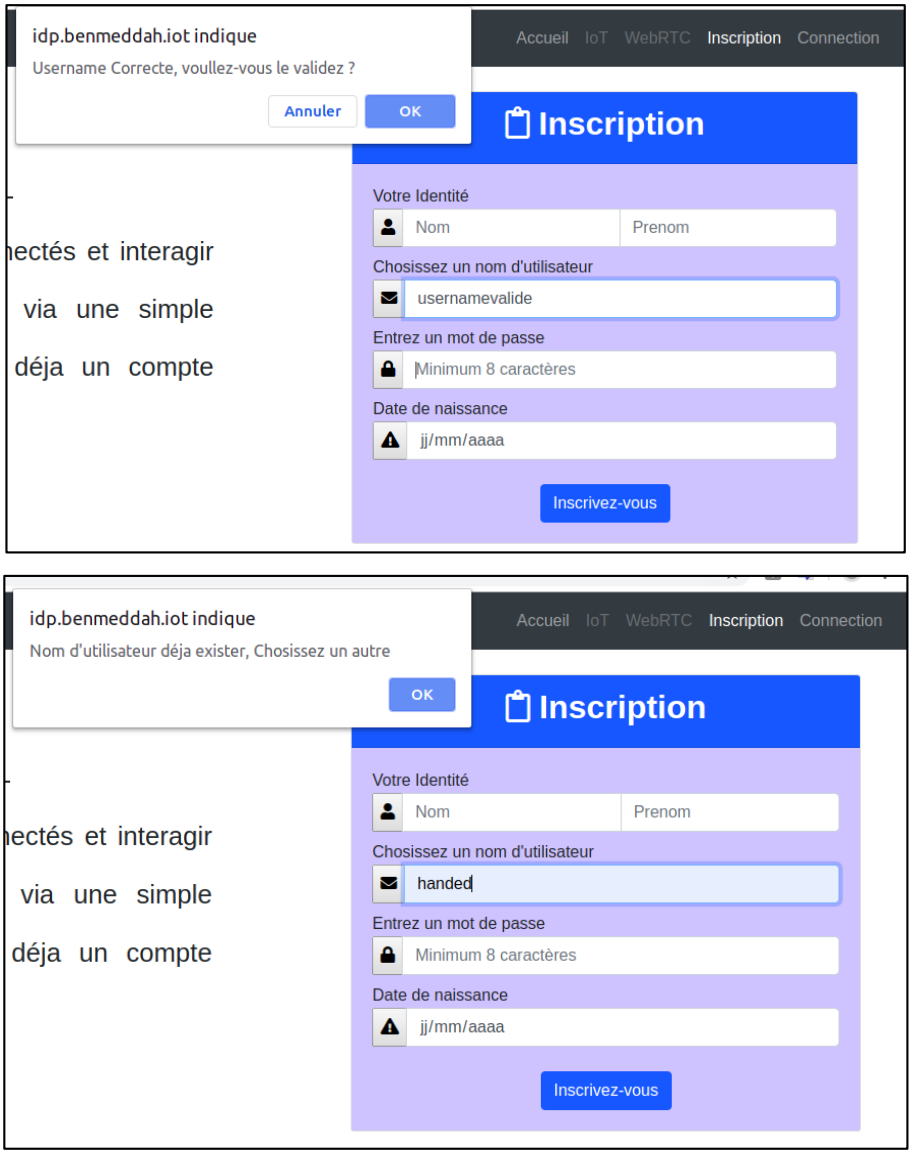

Figure 38 : Vérification du « username » en temps réel

- **Tentative**

Le serveur interdit l'accès aux pages d'inscription et de connexion après un nombre des tentatives pendant une durée de temps. Voici le message d'erreur qui spécifie le service bloqué.

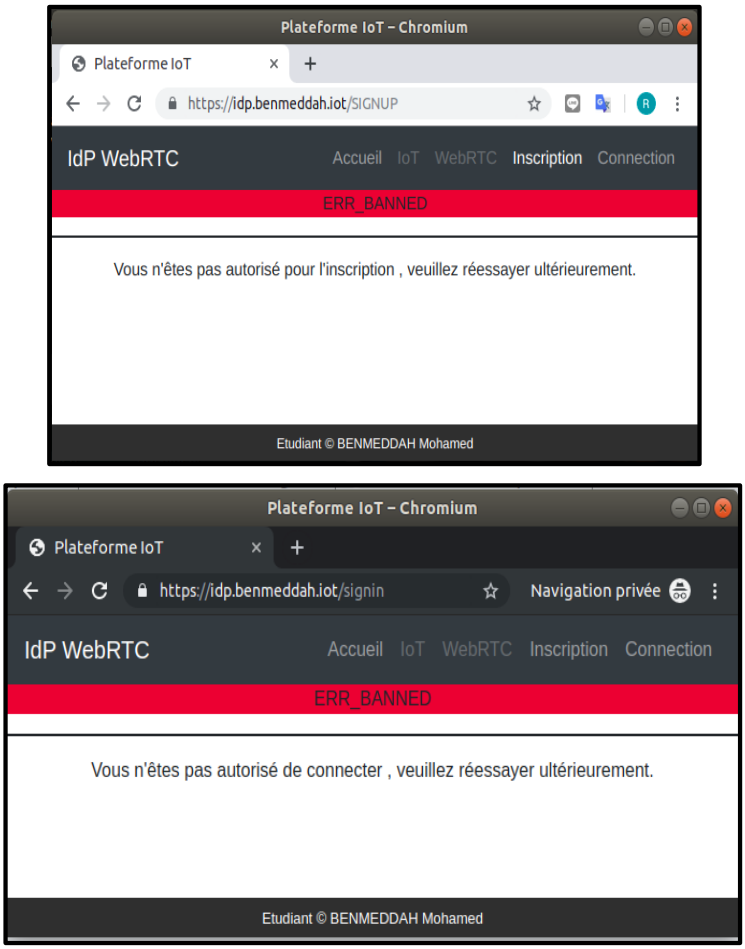

Figure 39 : Initialisation d'un objet avec une clef erronée

V.2. Serveur SP

Les paramètres entrés pour ajouter un objet IoT sont utilisé comme suivant :

- **Nom de l'objet** : Pour savoir le lien de l'interface WoT.
- **Fichier *pem*** : Pour la création de JWT qui permet l'interaction avec l'objet connecté et avec son Interface.

Voici l'action d'initialisation après la modification de la clef dans la base de données (la figure40).

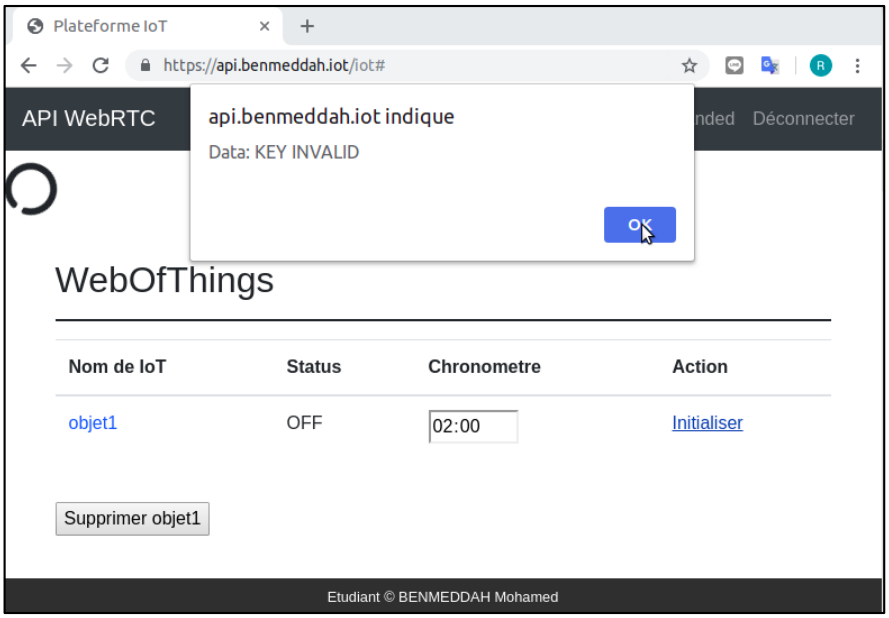

Figure 40 : Initialisation d'un objet avec une clef erronée

Du coté Interface WoT, le console affiche le message d'erreur « invalid signature » dans la figure 41.

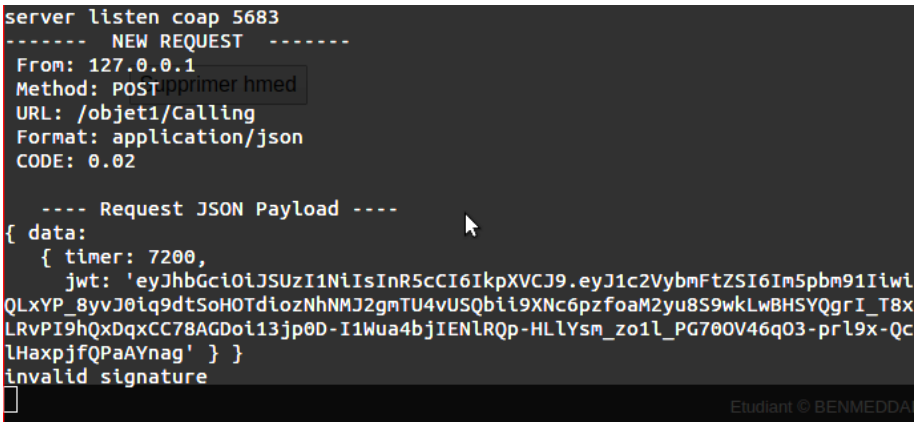

Figure 41 : Recevoir d'un message invalide par l'interface WoT

Le code JavaScript fournit à l'utilisateur ne le permet pas d'appeler un autre utilisateur qui n'est pas en ligne, voici (la figure 42) une teste d'appeler un utilisateur, avec état diffèrent de *en ligne*, après la modification du code.

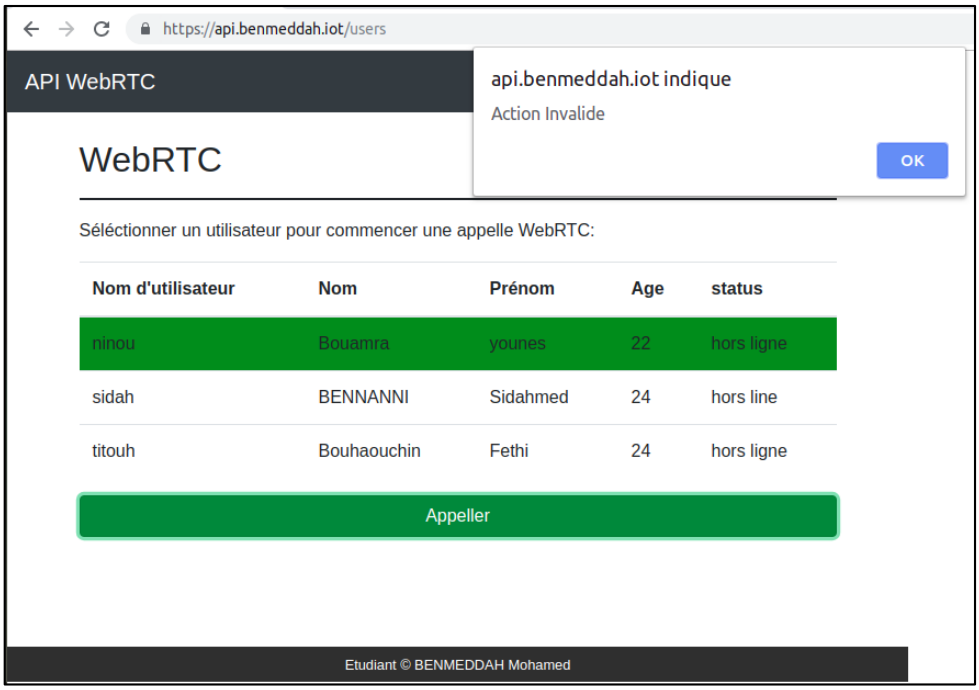

Figure 42 : Demande d'appeler un utilisateur hors ligne

- **Postman** [W21] : Cet outil permet d'envoyer des requêtes son passé par le code du navigateur, voici la requête post qui ne respecte pas les contraintes du formulaire de la page d'inscription (la figure).

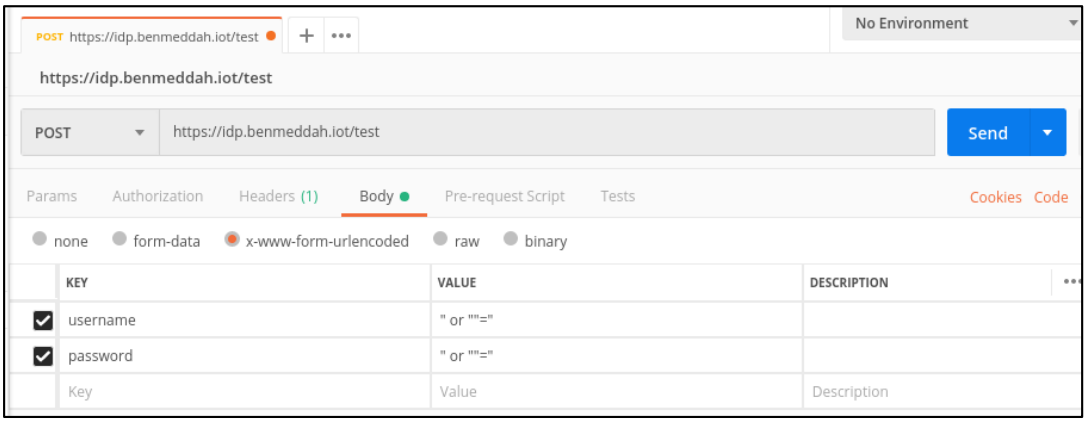

Figure 43 : Requête POST malveillant avec Postman

Le serveur SP vérifie les entrés et renvoie un message d'erreur tel qu'affiché dans la figure suivante :

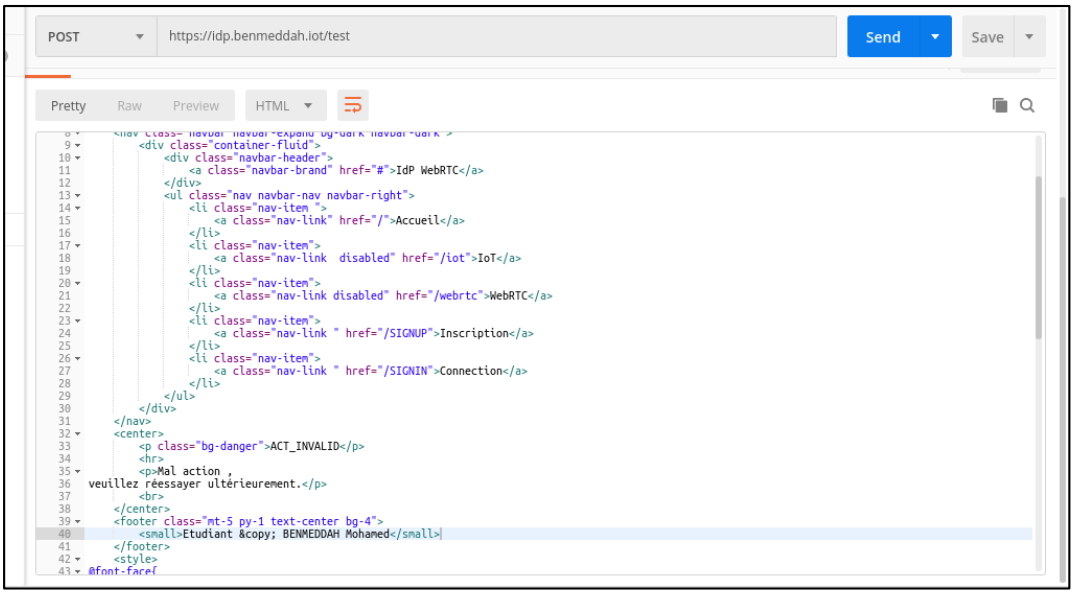

Figure 44 : Réponse à une requête malveillante

V.3. WebSockets

Le protocole WebSockets dans l'IoT utilise le même mécanisme d'authentification dans le serveur SP, la figure montre le résultat de teste par envoyer une demande de connexion à partir une origine non acceptable.

Figure 45 : Test de connexion WebSockets avec un entête invalide

La confidentialité est assurée par TLS pour le protocole TCP, la figure montre que nous avons utilisé le protocole HTTPS.

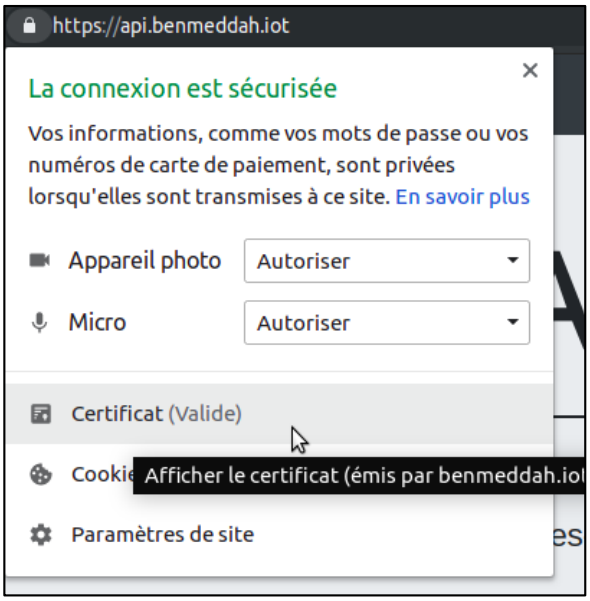

Figure 46 : Fenêtre démontre l'utilisation du protocole HTTPS

Après une modification au niveau de code JavaScript, le remplacement de protocole WSS par WS empêche la communication avec WebSockets.

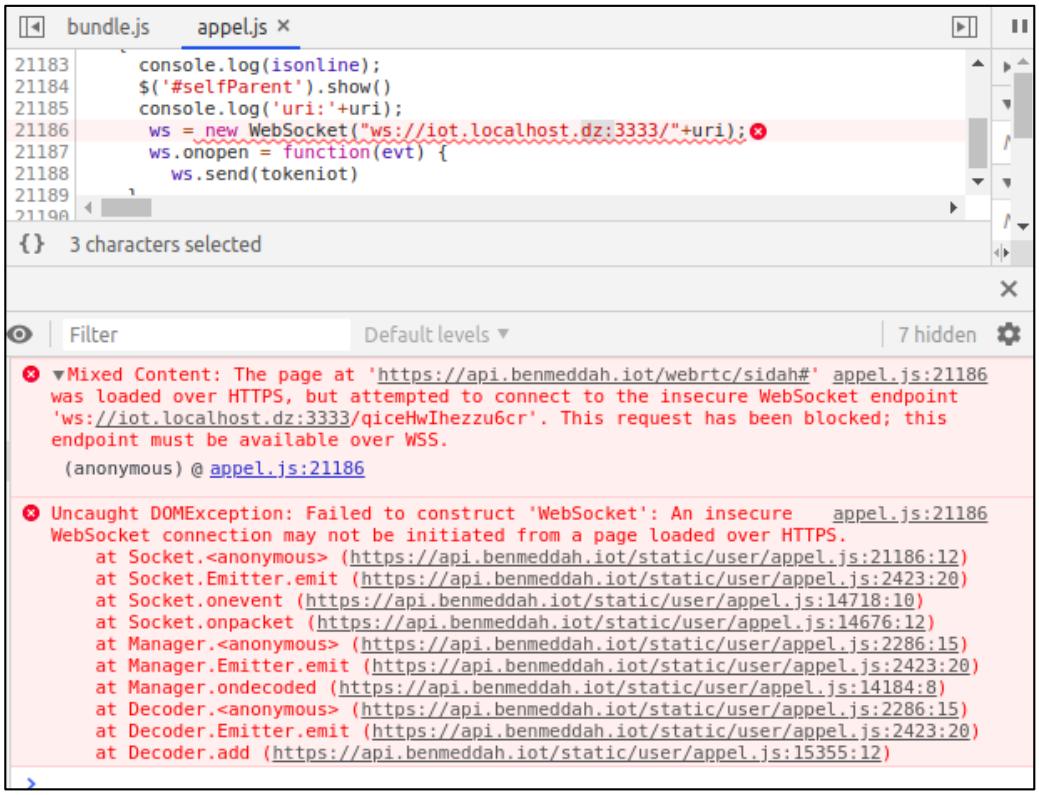

Figure 47 : Test de connexion avec WebSockets non sécurisé

V.4. Base de données

La figure suivant montre les informations des utilisateurs qui sont stockés dans la base de données. La colonne de password ressemble aux mots de passes après d'être modifiés et hashés, nous utilisons l'algorithme de SHA-256.

```
> select * from user;
+----------+------------------------------------------------------------------+----------+-------------+------------+
| username | password                                                         | fname    | lname       | d_birth    |
+----------+------------------------------------------------------------------+----------+-------------+------------+
| ninou    | d98b26d8a7b686a33eef4717b790d2e615158224a4f90e40b9d0d2d96f47e01d | younes   | Bouamra     | 1997-08-25 |
| handed   | eddd117f3b952f1929f0eb529435907fad7d19d726b09da4caacd0312f2fe387 | Mohamed  | BENMEDDAH   | 1995-01-06 |
| sidah    | a6dd3eb28605663a67a62ed9b23d5a14e78369cbff5ddbe89fc2f570a8ff24e6 | Sidahmed | BENNANNI    | 1994-11-10 |
| titouh   | a19ae80dc558b03c0e52c0c648c0465f6258ac94caf3a418c86800d76d5c2458 | Fethi    | Bouhaouchin | 1995-03-27 |
+----------+------------------------------------------------------------------+----------+-------------+------------+
```

Figure 48 : Table d'utilisateur montre des mots de passe non claire

Les valeurs ajoutées aux mots de passés avant le hashage sont affichés dans la figure :

```
var crypt = require("crypto-js/sha256");

const debut="gy-WcckczVp&!EXxXVt8Lt-Pn?@RRX#^Hf9+ZEhbs%sMxb2X8s!J?NR=^q&9p*vwA!9G7"+
                    +"6h5@mXwbU%B+@HZH9^qCj*J^xJpy5sWLm*qQc_TKUrJEyP%r^t7@=w46bdr",
        fin="PH%^L2d4$jKV6+PtT^X*JQGRFGeCJB8E^7a9AQCwwK8ufPBk=MfLF=U*B#LP=jx_PNYmK"+
                    +"w!CAzk9+E=AUd2tBZpDm5faF?v?&6VP^Yap$EJEE!M-qGwQD=h+=QU*93Gk";

                    function crypter(password,callback) {
                      password = ""+debut+password+fin;
                      let hash = crypt(password);
                      hash = hash.toString();
                      callback(hash)
                    }
```

Figure 49 : Constants et méthodes utilisés pour sécuriser le mot de passe

VI. Conclusion

Nous avons implémenté un système de gestion d'identité unique (SSO), et nous gérons aussi l'accès aux objets connectés pour les propriétaires. Ceci grâce à standard JWT et des options de connectivité flexibles de Web.

Conclusion Générale

Au cours de ce travail, nous avons essayé d'identifier les entités du système pour les affecter les permissions appropriées. C'est l'encadrement nécessaire pour limiter l'accès avant toute implémentation d'un politique de contrôle d'accès.

L'approche de la solution est basée sur le modèle d'authentification unique (SSO), elle fait centraliser l'entrée du système par le fournisseur d'identité, en suivant les mécanismes de sécurités requis. Ce modèle a été choisi après que nous avons vu les différents problèmes et attaques liés à l'architecture étudié.

Suite aux tests effectués, nous avons validé la sureté des services collaboratifs et la communication en temps réel entre les différentes entités de la plateforme IoT, nous avons réalisé le système de la communication en temps réel entre quatre entités, en suivant la série de bonnes pratiques d'OWASP, avec une bonne restriction des accès.

Améliorations

- Pour sécuriser les serveurs STUN et TURN nous proposons d'implémenter un serveur CoTurn [W22] qui est Open-source et supporter l'utilisation du protocole OAuth, et permet de récupérer les clés de vérification à partir d'une base de données.
- La protection des données sensible par un algorithme fort comme $blowfish$ [W23], et la gestion de traçabilité par faire un classement des actions non valides.
- Implémentation d'une politique de contrôle d'accès qui permet de gérer des conférences avec plusieurs utilisateurs et plusieurs objets connectés.

Références bibliographiques et webographies

Références bibliographies

[1] Nisha.V.K, Liyamol.A, Asha.A,"An Overview of Cryptographic Solutions to Web Security", Anna University of Technology, Coimbatore, India, 2010, p01.

[2] [3] [4] [5] Scholte.T, "Amélioration de la sécurité par la conception des logiciels web", Institut des sciences et technologies, Paris, France, 2012. pp. 25-31.

[6] [7] Pimentel.V, Nickerson B.G, "Communicating and Displaying Real-Time Data with WebSocket", Publié par IEEE Computer Society, 2012. p46-47.

[8] Wang.W.P, Mei.L, "A design of Multimedia Conferring System Based on WebRTC Technology", Université de Boston IEEE, USA, 2017. p148-149.

[9] [10] [11] [12] Bryan.S, Vincent.L, "Web Application Security A Beginner's Guide", McGraw-Hill Compagnies, USA, 2012. p11-19.

[13] [14] Claire. L. B, "Identités numériques", Institut Mines-Télécom, Paris, France, 2016. pp. 30-46.

[15] La norme ISO/IEC 24760-1 :2011, p20.

[16] Guillaume.H, "IAM - Gestion des identités et des accès : concepts et états de l'art", Creative Commons CC-BY-NC-ND, France ,2013. p05.

[17] [18] J. Vincent, "Identité numérique en contexte Télécom", Université de Caen Basse-Normandie, France, 2014. pp. 08-15.

[19] Claire.L.B, Op. Cit. p. 32.

[20] [21] [22] [23] [24] : J. Vincent, Op. Cit. P24-29. p. 37

[25] [26] Hany F.A, Madini O.A, "XACML for Building Access Control Policies in Internet of Things", 2018. p254.

[27] Bryan.S, Vincent.L, Op. Cit. pp. 150-151.

[28] [29] Guinard.D, Trifa.V, "Building the Web of Things", MANNING, Shelter Island USA, 2016. p08-09.

Références webographies :

[W1] : " Security Testing - Quick Guide"
https://www.tutorialspoint.com/security_testing/security_testing_quick_guide.htm (consulter le 17/04/2019)

[W2] : OWASP Top10 – 2017, The Ten Most Critical Web Application Security Risks, 2017.
https://www.owasp.org/index.php/Top_10-2017_Top_10 (consulter le 16/06/2019)

[W3] : D. Hardt, "The OAuth 2.0 Authorization Framework", RFC 6749, 2012.
https://tools.ietf.org/html/rfc6749 (consulter le 22/07/2019).

[W4] : Fossati.T " Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things", RFC 7925
https://tools.ietf.org/html/rfc7925 (consulter le 15/07/2019)

[W5] : Kis.Z et Nimura.K, "Web of Things (WoT) Scripting API", W3C Working Draft. 2018.
https://www.w3.org/TR/wot-scripting-api/ (consulter le 25/07/2019)

[W6] : Käbisch.S et Kamiya.T "Web of Things (WoT) Thing Description", W3C Working Draft, 2018.
https://www.w3.org/2019/wot/td (consulter le 25/07/2019)

[W7] : Reshetova.E et McCool.M "Web of Things (WoT) Security and Privacy Considerations", W3C Working Draft, 2018.
https://www.w3.org/TR/wot-security/ (consulter le 25/07/2019)

[W8] : Rosenberg.J , Mahy.R, Wing.D ,"Session Traversal Utilities for NAT (STUN)", RFC 5389, 2008.
https://tools.ietf.org/html/rfc5389 (consulter le 05/06/2019)

[W9] : Mahy.R et Matthews.P "Traversal Using Relays around NAT (TURN)", RFC 5766, 2010.
https://tools.ietf.org/html/rfc5766 (consulter 05/06/2019)

[W10] : Schneider.C "Cross-Site WebSocket Hijacking"

https://www.christian-schneider.net/CrossSiteWebSocketHijacking.html (consulter le 07/08/2019)

[W11] : Jones.M et Hardt.D "The OAuth 2.0 Authorization Framework: Bearer Token Usage", 2012.
https://tools.ietf.org/html/rfc6750 (consulter le 04/08/2019)

[W12] : Stein.S "Views", Microsoft Docs,2017.

https://docs.microsoft.com/en-us/sql/relational-databases/views/views (consulter le 02/08/2019).

[W13] : NodeJS.org (consulter le 25/05/2019)

[W14] : NpmJS.com (consulter le 25/05/2019)

[W15] : ExpressJS.com (consulter le 25/05/2019)

[W16] : EJS.co (consulter le 25/05/2019)

[W17] : Socket.IO (consulter le 25/05/2019)

[W18] : Raspbian.org (consulter le 27/06/2019)

[W19] : "Tornado Web Server"
https://www.tornadoweb.org (consulter le 02/08/2019)

[W20] : Nebra.M,"Adoptez une architecture MVC en PHP"
https://openclassrooms.com/fr/courses/4670706-adoptez-une-architecture-mvc-en-php (consulter le 30/07/2019)

[W21] : getpostman.com (consulter le 03/08/2019)

[W22] : "coturn TURN server project"
https://github.com/coturn/coturn (consulter le 27/07/2019)

[W23] : Pereira.R et Adams.R "The ESP CBC-Mode Cipher Algorithms", RFC 2451, 1998
https://www.ietf.org/rfc/rfc2451 (consulter le 15/06/2019)