STI Projet 2

Etude de menaces

Doran Kayoumi et Delphine Scherler

20.01.2022

Introduction

Cette deuxième partie de projet a comme but d'identifier les failles de sécurité de notre application, d'analyser les menaces et finalement de sécuriser notre application. Ce document fait office d'étude de menaces.

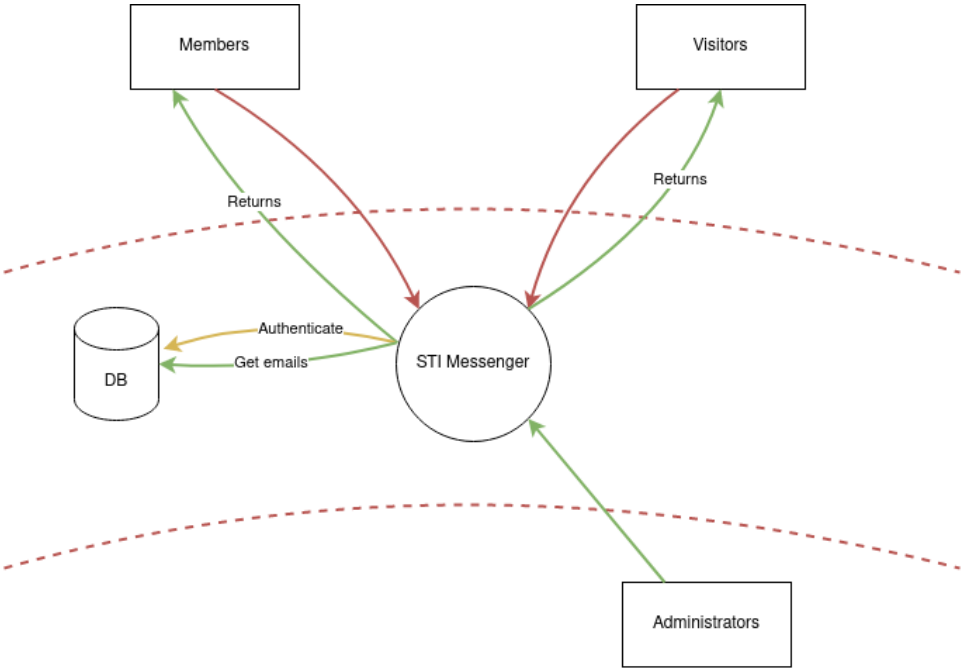
Description du système

Notre application est une application de messagerie permettant aux utilisateurs d'envoyer des messages. Il y a deux types d'utilisateurs, qui sont les utilisateurs simples et les administrateurs. L'authentification est obligatoire sur l'application pour pouvoir y faire quelconque action, en effet pour les visiteurs seul la page de login est accessible.

Les principales fonctionnalités de l'application sont les suivantes :

- Lecture des messages reçus
- Ecriture d'un nouveau message
- Changement de mot de passe
- Pour un administrateur : ajout, modification et suppression d'utilisateur

DFD (Data Flow Diagram)



Identification des biens

Les principaux biens de notre application sont les suivants :

- Noms d'utilisateur et mots de passe
- Messages

Tous deux sont stockés dans une base de données.

Périmètre de sécurisation

Dans ce projet nous n'allons sécuriser que la partie applicative. Nous n'allons pas nous occuper de la sécurisation de l'infrastructure comme le serveur web ou la machine client.

Identification des sources de menaces

Les principales sources de menaces de notre application sont :

- Menace humaine (hackers)
- Cybercrime (spam)

Identification des scénarios d'attaques

Scénario 1 : Vol des données utilisateurs de la DB

Motivation : Récupérer les mots de passe, pour ensuite les utiliser sur d'autres sites (i.e. l'utilisateur utilise le même mot de passe partout).

Cible(s) : Les données utilisateurs se trouvant dans la base de données.

Attaque(s) :

- Injection SQL

Scénario 2 : Vol des messages de la DB

Motivation : Obtenir des messages sensibles (e.g. informations bancaire, médicale, clé secrète, etc.)

Cible(s) : Les messages se trouvant dans la base de données.

Attaque(s) :

- Injection SQL

Scénario 3 : Accès aux pages d'administration

Motivation : Escalade de privilège, accès aux données utilisateurs.

Cible(s) : Le(s) comptes administrateur(s), le(s) page(s) d'administration.

Attaque(s) :

- Injection SQL
- Cross Site Scripting
- Cross-site request forgery

Scénario 4 : Enumération des utilisateurs

Motivation : Lister tous les utilisateurs ayant un compte sur l'application.

Cible(s) : Formulaire de login, les noms d'utilisateurs.

Attaque(s) :

- Timing attaque
- Injection SQL

Scénario 5 : Usurpation d'identité

Motivation : Envoi de messages spam.

Cible(s) : Les utilisateurs.

Attaque(s) :

- Injection SQL
- Cross Site Scripting
- Cross-site request forgery

Identification des contre-mesures

Sécurisation de l'authentification :

- Politique de mots de passe : le mot de passe doit contenir au minimum 8 caractères et au maximum 64.
- Hachage des mots de passe : se fait avec un algorithme bcrypt. À noter que l'algorithme Argon2 serait plus approprié au niveau sécurité, mais pas disponible avec la version de PHP que nous utilisons.

Sécurisation des champs de formulaire :

- Ajout d'un token caché dans tous les formulaires afin d'éviter des attaques CSRF.
- Validation des inputs utilisateur, avec la méthode htmlentities() pour éviter des attaques XSS.

Utilisation de prepared statements pour les requêtes de base de données.

Utilisation du bon type de requête pour les formulaires : POST vs GET. Ainsi que s'assurer de la présence de tous les paramètres requis.

Mettre le moins d'information possible dans l'URL :
À éviter : /show_message.php?sender**=<sender>**&subject=**<subject> -->** /show_message.php?**id=<id>**

Conclusion

Suivant notre étude de menaces, nous avons implémenté diverses solutions pour améliorer la sécurité de notre site web. Par exemple l'utilisation de prepared statements pour toutes les interactions avec la base de données.

Tout ce que nous avons mis en place étant fait maison, afin d'accroitre la sécurité, il serait plus intéressant d'utiliser des librairies. Par exemple utiliser un moteur de templating tel que twig et d'ORM pour les interactions avec la base de données.

Nous nous sommes principalement occupés de sécuriser le codebase du site web. Dans un deuxième temps, il faudrait mettre à jour le serveur web ainsi que la version de PHP.