Rapport d'étude de menaces

STI Projet 2

Date: 20 janvier 2022
Le Ray Quentin, Pellissier David

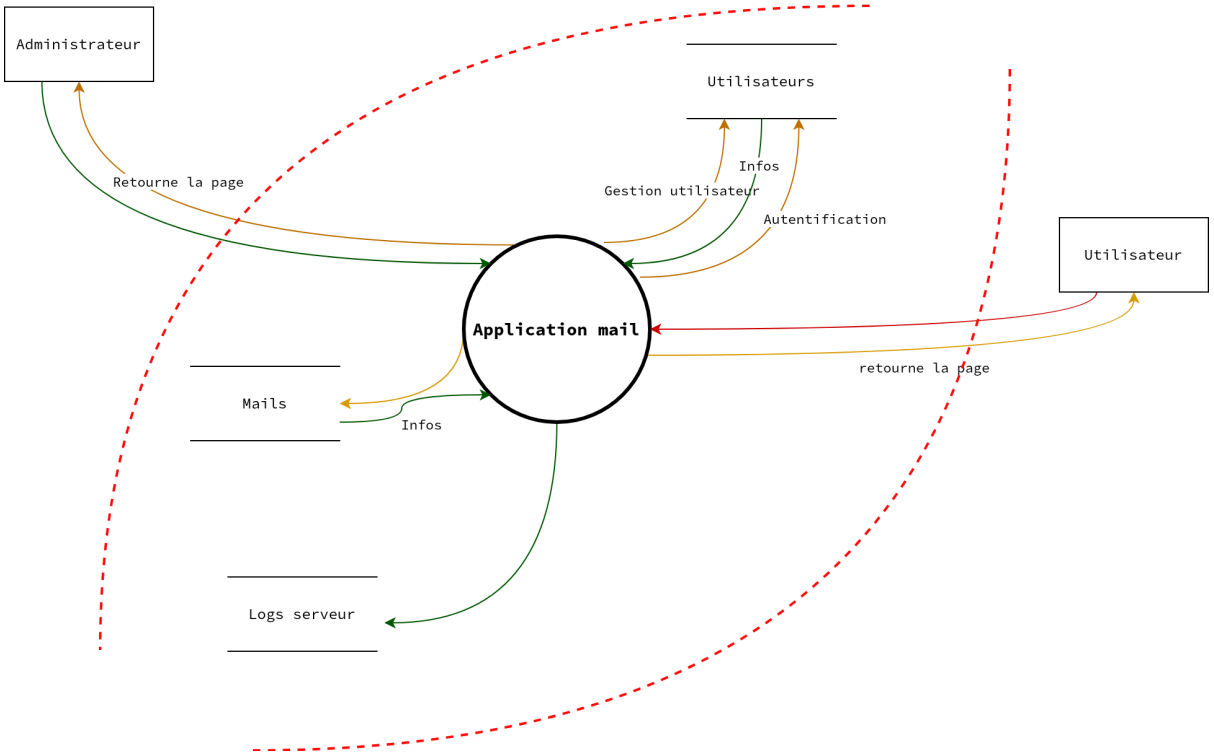
Table des matières

Introduction

Ce document a pour but d'analyser les menaces de l'application de messagerie développée dans le Projet 1 de STI, ainsi que de décrire leurs contre-mesures.
Cette étude de menace sera utilisée afin de sécuriser le code de l'application. La première version de l'application peut se trouver ici: [RebeccaTevaearai/STI_projet1]

Description du système

DFD



Identifier ses biens

- Logins utilisateur
- Mails envoyés
- Logs
- Disponibilité de l'infrastructure

Périmètre de sécurisation

La sécurisation ne se fera que sur la partie applicative du site.
Les améliorations de sécurité touchant à la configuration et version du serveur (Nginx et PHP) ainsi qu'à l'infrastructure (HTTPS, Load balancing) seront listées mais pas implémentées à l'issue de la sécurisation du site.

Nous partons du principe que l'application est exposée à Internet et n'importe qui peut y accéder.

Sources de menaces

- Espionnage industriel de la part d'un concurrent.
- Mauvaise utilisation d'un employé.
- Employé mécontent
- Cybercriminel
- Pirate informatique et Script kiddies

Scénarios d'attaques

Eléments du système attaqué

- Page de login
- Pages de messagerie
- Pages d'administration
- Base de données
    - Utilisateurs
    - Mails
    - Logs du serveur
- Serveur web - disponibilité

Motivations

- Perte d'argent ou de productivité de l'entreprise
- Vol de données et espionnage
- Rançon ou destruction de données
- Pirater l'application pour le divertissement

Scénarios d'attaque et contres-mesures (STRIDE)

Usurpation (Spoofing)

Agent externe:

Scénario	Contres-mesures
Utiliser un mot de passe volé pour se connecter à l'application	1.Multi Factor Authentication 2.Obliger l'utilisateur à changer de mot de passe périodiquement
Trouver le mot de passe d'un utilisateur par attaque de force brute (bruteforce)	1.Bloquer ou mettre un timeout après des requêtes de login répétitives 2.Utiliser une politique de mots de passes stricte

	3.Saler les hashs pour éviter l'utilisation de Rainbow Tables si le hash vient à être volé.
Récupérer les informations de connexion d'un compte par une attaque phishing	1.Filtrer les mails dangereux 2.Avertir l'utilisateur lorsqu'il clique sur un lien suspect
Via une faille XSS, voler le cookie de session d'un utilisateur en l'envoyant sur un serveur distant	1.Protection XSS: utiliser htmlspecialchars() pour encoder les données avant de les afficher. 2.Configurer le CORS pour empêcher les requêtes externes 3.Configurer le flag "HttpOnly" du cookie pour empêcher de le récupérer via Javascript 4. Configurer le CSP pour empêcher l'inclusion de script externes 5. Configurer le flag "Secure" du cookie pour qu'il ne puisse être envoyé que par HTTPS

Base de données:

Se connecter à la base de données locale avec le login par défaut	Changer les credentials par défaut
À cause d'un fichier de config accessible, on peut récupérer le mot de passe de la BDD	Conserver la config dans un dossier protégé, non servi par le serveur

Altération de données (Tampering)

Bases de données:

La base de données des mails ne vérifie pas suffisamment les authentifications et il est possible d'envoyer des mails sans avoir de compte sur l'application	Vérifier à chaque requête les autorisations de l'utilisateur
Par injection SQL, il est possible de modifier ou supprimer des mails de la base de données	Protection contre les injections SQL: Préparer les requêtes avec les fonctions PHP prepare() et bindValue().
Modifier/Supprimer un compte utilisateur par injection SQL	Protection contre les injections SQL

Processus:

Modifier le contenu d'une variable de session à partir d'une injection PHP	Éviter d'utiliser des "eval" ou autres fonctions permettant des appels système.

Flux de données:

Modifier l'affichage d'un mail via une XSS stockée dans le but d'exécuter du code JavaScript malveillant	Protections XSS listées dans la partie Spoofing

Répudiation

Agent externe:

Un utilisateur utilise une CSRF afin d'envoyer un mail important de la part d'un autre utilisateur, dans le cadre de social engineering	1.Token CSRF 2.Configurer le CORS

Base de données:

Modifier le mot de passe d'un autre utilisateur	1.Vérifier l'autorisation de l'utilisateur connecté avant de faire la requête 2.Envoyer un message d'avertissement par un autre service (SMS)

Processus:

L'application utilise une librairie JavaScript externe, qui peut être contrôlée par un utilisateur	Stocker les fichiers de librairie sur le serveur

Divulgation d'information (information disclosure)

Base de données:

La base de données n'est pas protégée par un mot de passe et on peut y lire toutes les données sans se connecter	1.Mettre un mot de passe administrateur à la base de données 2.Whitelister les adresses IP autorisées à accéder à la BDD

Processus:

Il pourrait rester des commentaires dans le code source donnant des informations confidentielles.	Supprimer les commentaires non essentiels et TOUS les commentaires de débug
Trop d'informations retournées dans les requêtes au serveur	Retourner uniquement le minimum nécessaire au bon fonctionnement du front-end

Flux de données:

Le protocole utilisé n'est pas chiffré (HTTP) ce qui permet de voir en clair les requêtes effectuées	1.Utiliser HTTPS et renouveler son certificat avant son expiration. 2.Interdire les requêtes HTTP et rediriger vers HTTPS

Déni de service

Processus:

Un gros volume de requêtes peut être envoyé sur la page de login afin de rendre inutilisable le service (DDoS).	1.Système anti DDoS 2.Améliorer l'infrastructure en ajoutant des serveurs et du load balancing
Faire crasher le service en utilisant une faille liée à la version de l'application serveur	Mettre à jour l'infrastructure (php, serveur apache, etc.)

Base de données:

Un utilisateur effectue un gros volume de requêtes sur la base de données pour la rendre indisponible voir la faire planter.	Limiter le nombre de requêtes qu'un utilisateur peut effectuer sur la base de données par minute.

Elévation de privilèges

Processus

Bypass des sessions PHP car vieille version utilisée.	Mettre à jour la version de PHP utilisée et mettre en place une gestion des sessions authentifiées via cookie chiffré/hashé.
Grâce à une attaque CSRF contre l'admin, attribuer les droits admin à un utilisateur	Protections CSRF données dans la partie Repudiation

Base de données

Modification de son rôle grâce à une injection SQL	Protections SQL déjà décrites dans ce document

Conclusion

Au terme de ce projet, nous avons pu sécuriser l'application selon les scénarios d'attaque étudiés.

Toutes les améliorations effectuées sont détaillées dans le fichier RapportSecurisation.md. Nous y avons également précisé celles que nous n'avons pas implémentées car sortant du cadre de ce projet, c'est-à-dire tout ce qui touche à la config du serveur et de l'infrastructure.

Nous avons également amélioré et factorisé le code lorsque nous le jugions nécessaire, tout en prenant soin de ne pas trop modifier le code du projet 1.