STI - Projet

Modélisation de menaces

WeChat

Matthieu Chatelan & Loan Lassalle

10 janvier 2018

Table des matières

Table des figures

1 Introduction

Le but de ce document est de fournir une analyse de menaces auquels notre programme WeChat est exposé. Premièrement, une description du système est donnée fournissant ainsi une meilleure compréhension des différents composants de ce dernier, les différents rôles mis à disposition ou encore les biens qui seront à protéger.

Ensuite, une analyse des différentes sources de menaces auquels notre programme sera exposé. Les différentes sources potentielles sont énumérées ainsi que les compétences requises pour effectuer chacunes de ces attaques.

Finalement, plusieurs scénarios d'attaques seront donnés ainsi que les contremesures appropriées à mettre en place afin de les contrer.

2 Description du système

2.1 Objectifs du système

Les objectifs fixés du système étaient de développer une application web permettant d'envoyer et de reçevoir des messages textes entre collaborateurs au sein d'une entreprise. Le protocole de transmission SMTP ne devait pas être utilisé pour cette application.

Cette application serait par exemple utilisée entre employés d'une entreprise de développement de logiciels.

2.2 Exigences du système

Lors du développement de l'application, plusieurs contraintes ont été fixées par le client dans le cahier des charges. Dans les sous sections ci-dessous, ces dernières sont décrites et expliquées.

2.2.1 Exigences technologiques

Pour développer l'application, les différentes technologies suivantes ou contraintes ont dû être prises en compte :

- PHP
- SQLite
- Doit fonctionner sur l'environnement CentOS fournis

2.2.2 Exigences fonctionnelles

L'application devait être en mesure proposer deux rôles différents soit **collaborateur**, soit **administrateur**. De plus, un mécanisme d'authentification simple (utilisateur - mot de passe) doit permettre d'accéder aux fonctionnalités. Pour pouvoir se connecter, un utilisateur doit être définit comme étant "actif".

Une navigation aisée via des liens ou des boutons devait être mise en place.

Un **collaborateur** doit pouvoir effectuer les actions suivantes :

1. Lire les messages reçus sous forme de liste triée par ordre de date de réception avec plusieurs informations tels que :
   - la date de réception
   - l'expéditeur
   - le sujet
   - un bouton permettant de répondre au message (avec le sujet directement définit)
   - un bouton permettant de supprimer le message
   - un bouton permettant d'ouvrir un message et de voir le contenu du corps
2. Ecrire des messages à l'attention d'un autre utilisateur. Les informations suivantes doivent être fournies :
   - destinataire (unique)
   - sujet
   - corps du message
3. Changer le mot de passe de l'utilisateur connecté.

Un **administrateur** doit pouvoir effectuer les actions suivantes :

1. Avoir les mêmes actions qu'un collaborateur
2. Ajouter, modifier ou supprimer un utilisateur, représenté par les attributs suivants :
   - Un login (non modifiable)
   - Un mot de passe (modifiable)
   - Une validité actif/inactif (modifiable)
   - Un rôle (modifiable)

2.2.3 Exigences sécuritaires

Une authentification était demandée. Cette dernière ne devait pas permettre l'accès à toute autre page que celle de login lorsque l'utilisateur n'est pas authentifié.

Lors du développement initial, aucunes autre mesures de sécurité n'étaient demandées. De ce fait, aucunes protections contre des attaques XSS, CSRF, SQL Injection ou autre n'ont été mises en places.

Néanmoins, les mots de passes n'étaient pas stockés en clair dans la base de donnée, mais sous forme de hash salé pour empêcher une attaque par rainbow table sur ces derniers.

2.3 Constitution du système

Le système développé comprenait plusieurs acteurs et processus ainsi qu'une base de donnée centrale. Dans le diagramme de flux de données de l'application (Figure 1), un acteur est représenté par un rectangle alors qu'un processus est représenté par un cercle.

On notera que tous les processus qui se situent au dessus de la base de données dans le diagramme (5 au total) sont des processus que les deux acteurs peuvent utiliser. En revanche, les 3 processus au dessous de la base de données sont des processus réservés aux administrateurs.

Lellipse traitillée en orange représente la frontière de confiance de l'application. Cela signifie que nous considérons que les communications entres processus et la base de données sont sécurisés. Toutes entrées utilisateur doivent être contrôlés avant de pénétrer cette zone de confiance.

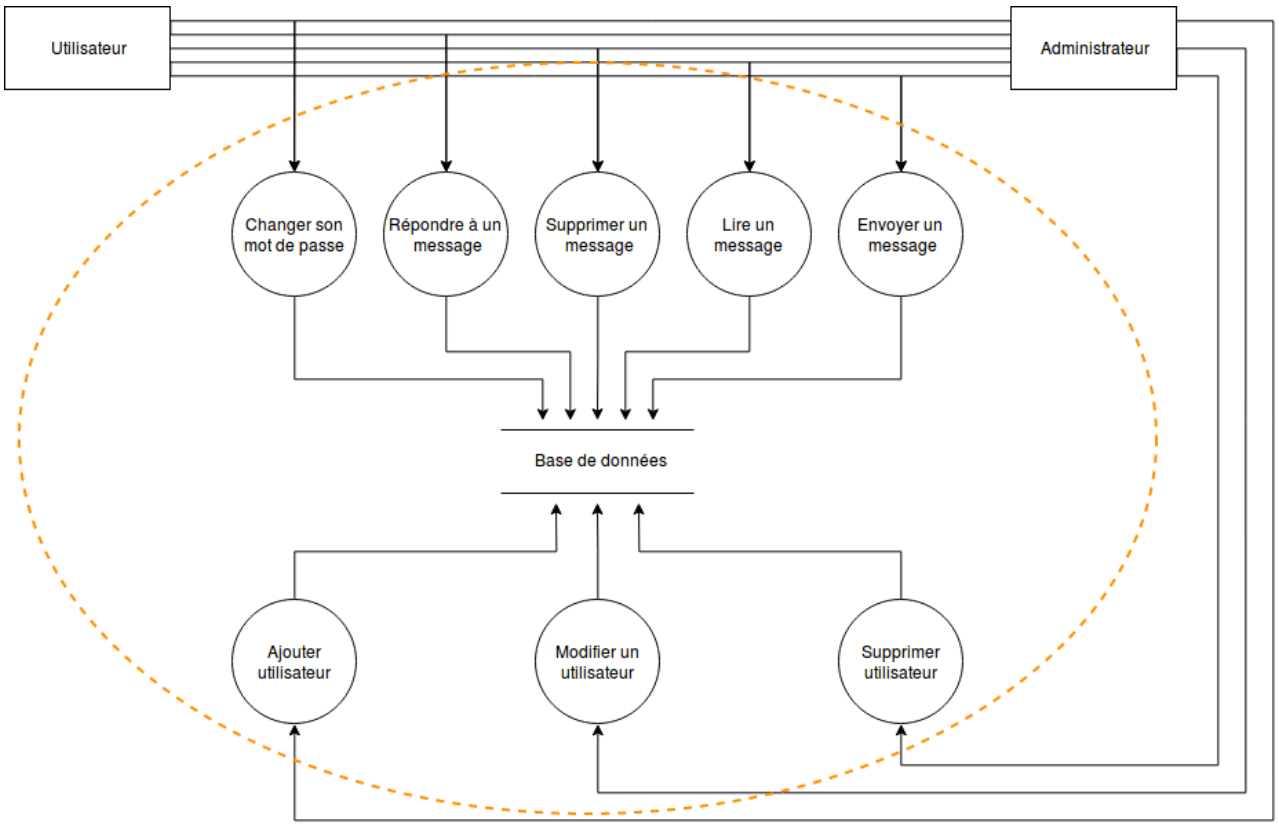

FIGURE 1 – Diagramme de flux de données de l'application

2.4 Biens

Les biens à protéger sont multiples dans une telle application. Ci-dessous, une liste non exhaustive de ces derniers

- Les messages
- Les comptes utilisateurs (leurs mots de passes)
- Laccès à l'administration des comptes

3 Sources de menaces

Initiateur	Motivations	Cible(s)	Potentialité
Utilisateur	Fun, Revanche, Curiosité	Tout éléments accessibles	Haute
Administrateur	Revanche, Curiosité	Tout éléments accessibles	Moyenne
Concurrent	Secrets d'entreprise	Base de données	Moyenne
Hackers	Gloire, Argent, Destruction	Tout éléments accessibles	Faible
Cyber-criminel	Vol d'informations, Spam, DDoS	Base de données	Faible

Parmi les différents initiateurs décrits ci-dessus, les utilisateurs représentent la plus grande menace et la menace la plus probable d'arriver. En effet, comme le programme est destiné à un usage interne à l'entreprise, a priori seuls les employés y auront accès.

N'y avoir accès que en interne ne signifie pas que ce n'est pas accessible de l'extérieur. Un attaquant externe (hacker, concurrent, cybercriminel) pourra tout à fait utiliser de l'ingénierie sociale pour inciter un utilisateur à lancer un programme malveillant afin de rendre l'accès à distance possible par le biais d'un reverse shell par exemple.

4 Scénarios d'attaques

Dans cette section, plusieurs scénarios d'attaque sont décrits. Grâce à ces scénarios, il est possible d'imaginer les différents points vulnérables de l'application et ainsi permet de mettre en place des contremesures appropriées.

#	**Impacte**	**Source de menace**	**Bien ciblé**
4.0.1	Perte de confidentialité	Utilisateur, Concurrent, Hacker, Cyber-criminel	BDD
4.0.2	Perte de disponibilité	Utilisateur, Concurrent, Hacker, Cyber-criminel	Tout élément actif
4.0.3	Perte d'intégrité	Hacker, Concurrent	BDD

Les différents scénarios d'attaques et leurs sources de menaces dépendent de comment l'application de chat est déployée et de comment elle est utilisée par les employés. En effet, si l'application est accessible par l'extérieur de l'entreprise, il sera facile pour des intervenant externes tels que des cyber-criminels ou des hackers d'accèder directement aux biens.

En revanche, si l'application est déployée sur un réseau local non accessible par l'extérieur, un attaquant externe devra s'introduire dans le réseau avant de pouvoir attaquer l'application et accèder aux biens. De l'ingénierie sociale peut être utilisée pour inciter un employer à visiter un lien ou à installer une application vulnérable.

4.0.1 Scénario d'attaque

Dans ce scénario d'attaque, les différents acteurs listés en tant que source de menace tenteront d'accèder à des informations qui ne leurs sont pas destinées causant ainsi une perte de confidentialité.

Pour parvenir à ses fins, un attaquant pourra essayer de multiples vecteurs d'attaque tels que :

- Manipulation de l'URL
- Injection SQL dans les divers champs de saisie disponibles
- Utilisation d'une faille de XSS stocké afin de récupérer des cookie de session

Sans des connaissances poussées en informatique, on pouvait très rapidement remarquer que lors de la lecture d'un message, l'id de ce dernier était donnée dans l'url. Etant donné qu'aucunes vérification n'était faite pour s'assurer que la personne lisant le message était bien le destinataire de ce dernier, il était possible d'écrire un script python qui testait de manière incrémentale l'accès à tous les message en incrémentant l'id de ce dernier. Si en réponse une erreur 404 est obtenue, on sait que le message n'existe pas. En revance, si un code 200 est retourné, on sait qu'un message existe et on peut automatiquement l'extraire et le stocker dans un fichier ou le parser afin de tenter de trouver des informations confidentielles tels que des adresses mail ou des mots de passes.

Avec peu de recherches sur internet et quelques tentatives, un attaquant avec de faibles connaissances sera en mesure de récupérer des données confidentielles tels que des mails ou encore des mots de passes en utilisant une injection SQL. Une fois les mots de passes récupérés, une étape de cassage hors ligne est nécessaire étant donné que mots de passes sont stocké sous forme hachée et salée.

Avec une attaque plus élaborée, cet employé pourrait récupérer des cookie de session en se

servant d'une attaque par XSS. Pour récupérer des informations sur les conversations d'un utilisateur, une solution serait d'envoyer un message avec du code javascript en tant que sujet du message. Ce dernier sera stocké sur le serveur et lorsque l'utilisateur cible se connectera à sa boîte de réception, le code malicieux sera exécuté car les sujets sont affichés. Tous les destinataires ou sujets des messages reçus affichés par la victime peuvent être envoyés à l'attaquant (tant que notre message est toujours visible dans la liste et donc que le script est exécuté).

L'exemple de code ci-dessous à insérer dans le champ sujet du message permet de récupérer facilement tout le contenu html de la page après un délai de une seconde (pour laisser le temps aux autres informations de s'afficher dans le navigateur de la cible). Ici l'exemple ne transmet pas les informations à l'attaquand mais les affiche à la cible.

```
<script>
  setTimeout(() => {alert(document.documentElement.innerHTML);}, 1000);
</script>
```

4.0.2 Scénario d'attaque

Ce scénario concerne la destruction d'informations par des tiers tels que des hackers ou encore des concurrents. Plusieurs approches sont possibles :

- Manipulation de l'URL
- Attaques XSS réfléchi ou stocké
- Attaques CSRF
- Injections SQL

4.0.3 Scénario d'attaque

5 Contremesures

tableau récap et montrer new implémentation

#	**Vulnérabilité**	**Contremesure**
5.1	xxx	yyy
5.2	xxx	yyy
5.3	xxx	yyy
5.4	xxx	yyy
5.5	xxx	yyy

5.1 Contremesure

blabla pour fix il faut faire ca :

```
1  mettre du super code ici
```

comme ca plus possible de hack :)

5.2 Contremesure

5.3 Contremesure

5.4 Contremesure

5.5 Contremesure