

heig-vd

CREPE MAILBOX

Strategies d'attaques par Walid Koubaa & Joel Schar

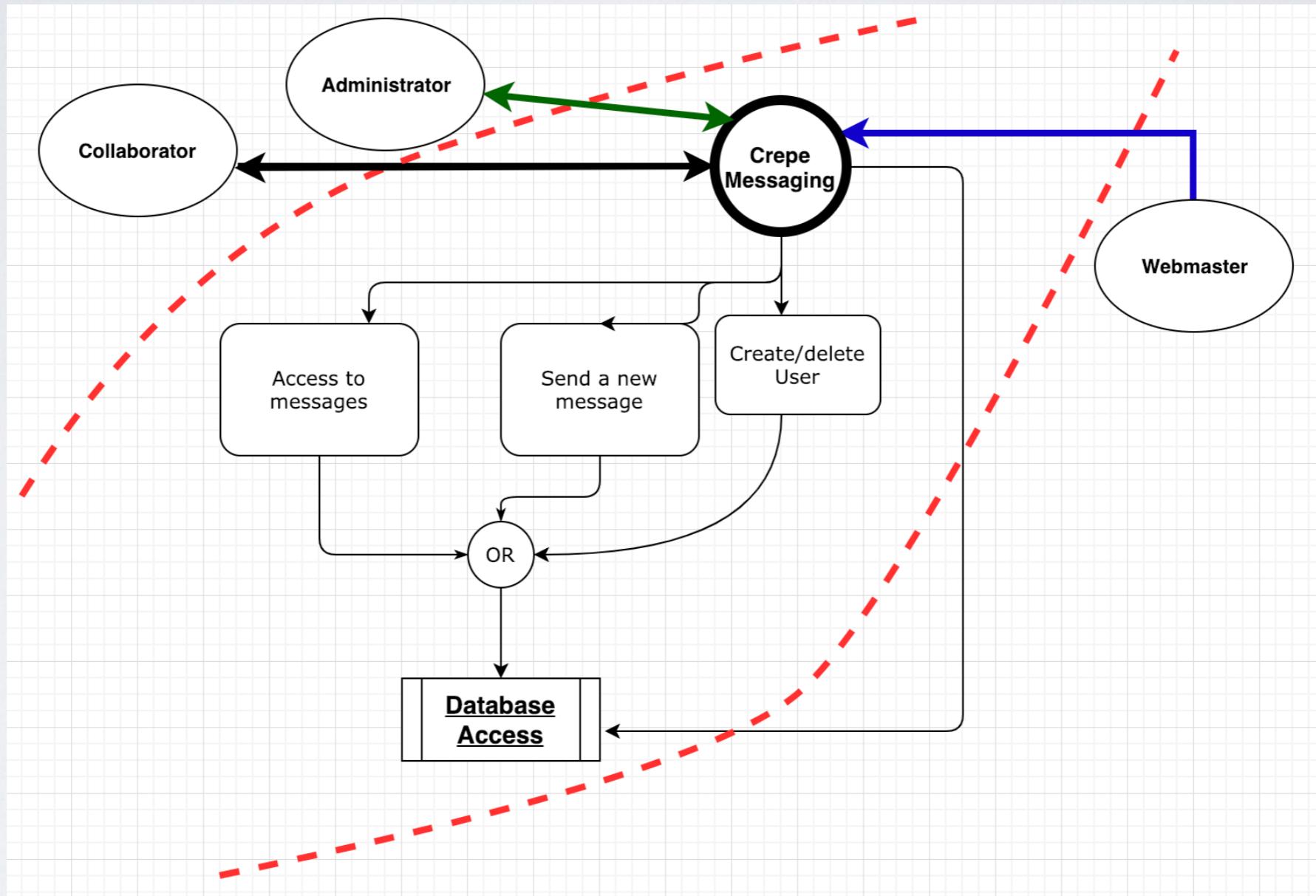
CREPE MESSAGING, C'EST QUOI ?

Une crêpe est un support comestible qui peut être remis en le lançant comme un freezbe.

Crepe Messaging vous permet d'envoyer de messages qui utilisent ce support à tous les utilisateurs inscrits sur la plateforme.

DESCRIPTION DU SYSTÈME

Data flow diagram



Objectifs du système

Permettre d'envoyer des messages entre collaborateurs et permettre aux administrateur de créer de nouveaux utilisateurs en lui spécifiant un rôle (collaborateur/administrateur).

SOURCES DE MENACES

Sur cette plateforme on peut envisager les potentielles menaces suivantes.

- 1. Un utilisateur normal qui voudrait voir les messages des autres utilisateurs.**
- 2. Un utilisateur normal qui voudrait envoyer des messages en se faisant passer pour un autre utilisateur.**
- 3. Un utilisateur qui voudrait supprimer les messages d'un autre utilisateur.**
- 4. Un utilisateur qui voudrait modifier le message d'un autre utilisateur.**
- 5. Un utilisateur qui voudrait faire passer son compte en administrateur.**
- 6. Un utilisateur qui voudrait modifier le mot de passe d'un autre compte.**
- 7. Un utilisateur qui voudrait obtenir les mots de passe des autres comptes de la plateforme.**

ATTAQUES EFFECTUÉS

A1 - Attaque par modification de l'URL

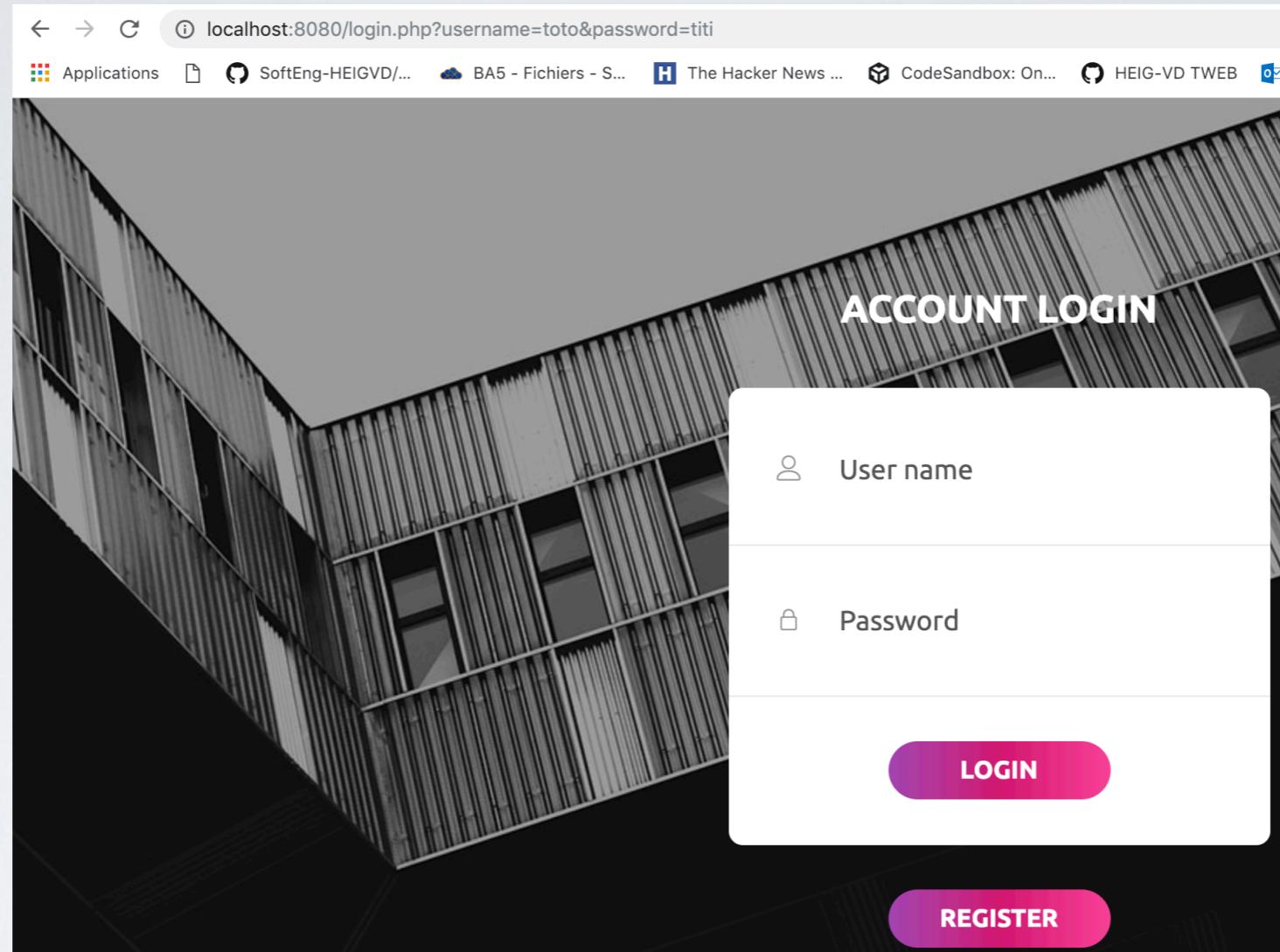
A2 - Brute force du login form - avec Burp

A3 - Id des messages directement accessibles depuis l'URL

A4 - Accès/Modification aux informations des utilisateurs
visibles seulement par l'administrateur

A5 - Modifier le mot de passe d'un utilisateur choisi 8 (Injection SQL)

AI - Attaque par modification de l'URL



Elément du système attaqué: Page de login

Motivation: L'objectif est de pouvoir se connecter automatiquement depuis l'URL via un POST

Scénario: Nous avons essayé de nous connecter directement à la manière d'un POST en spécifiant les credentials directement depuis URL.

Bilan de l'attaque: FAILURE !

A2 -Brute forceable login form - No limit of max login attempts

The screenshot shows the Burp Suite interface in Intercept mode. The request details pane displays a POST request to `/login.php` with the following headers and body:

```
POST /login.php HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:63.0) Gecko/20100101 Firefox/63.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 60
Connection: close
Cookie: PHPSESSID=1mbmoq18c0qcbcm4smfvg2uls6
Upgrade-Insecure-Requests: 1

username=admin%22+OR+1%3D1+%2F%2F&password=admin&login=login
```

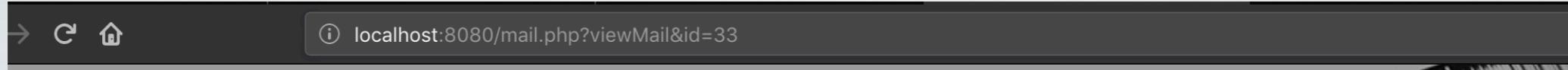
Elément du système attaqué: le formulaire de login (username et password)

Motivation: L'objectif est de bruteforcer tout les mots de passe pour tout les logins que nous spécifions dans la liste de payloads via l'outil Burp Suite.

Scénario: Avec Burp on configure le proxy de notre browser Firefox et spécifie l'adresse et le port de notre login Crepe Messaging (127.0.0.1:8080 Grace à cela nous pouvons brute forcer tout les mots de passe pour tout les logins que nous spécifions dans la liste de payloads.

Bilan de l'attaque: **SUCCESS !**

A3 - Id des messages directement accessibles depuis l'URL



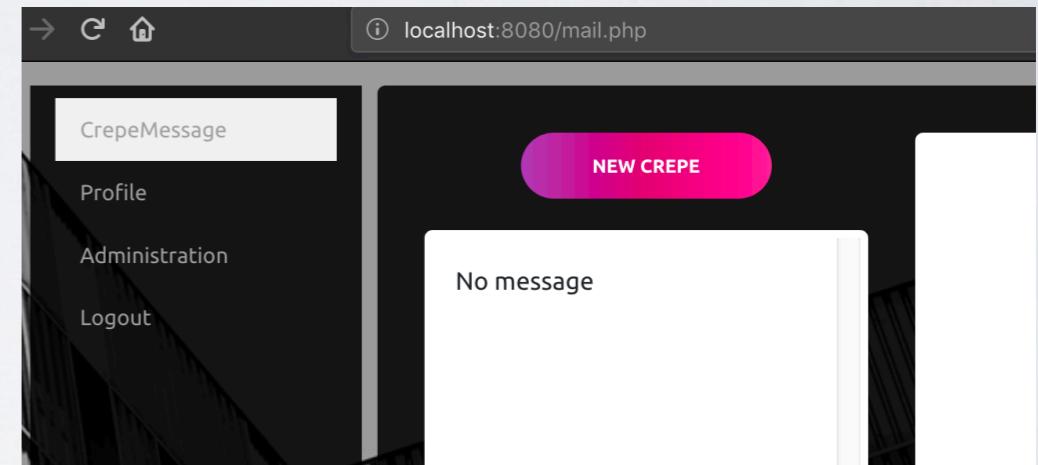
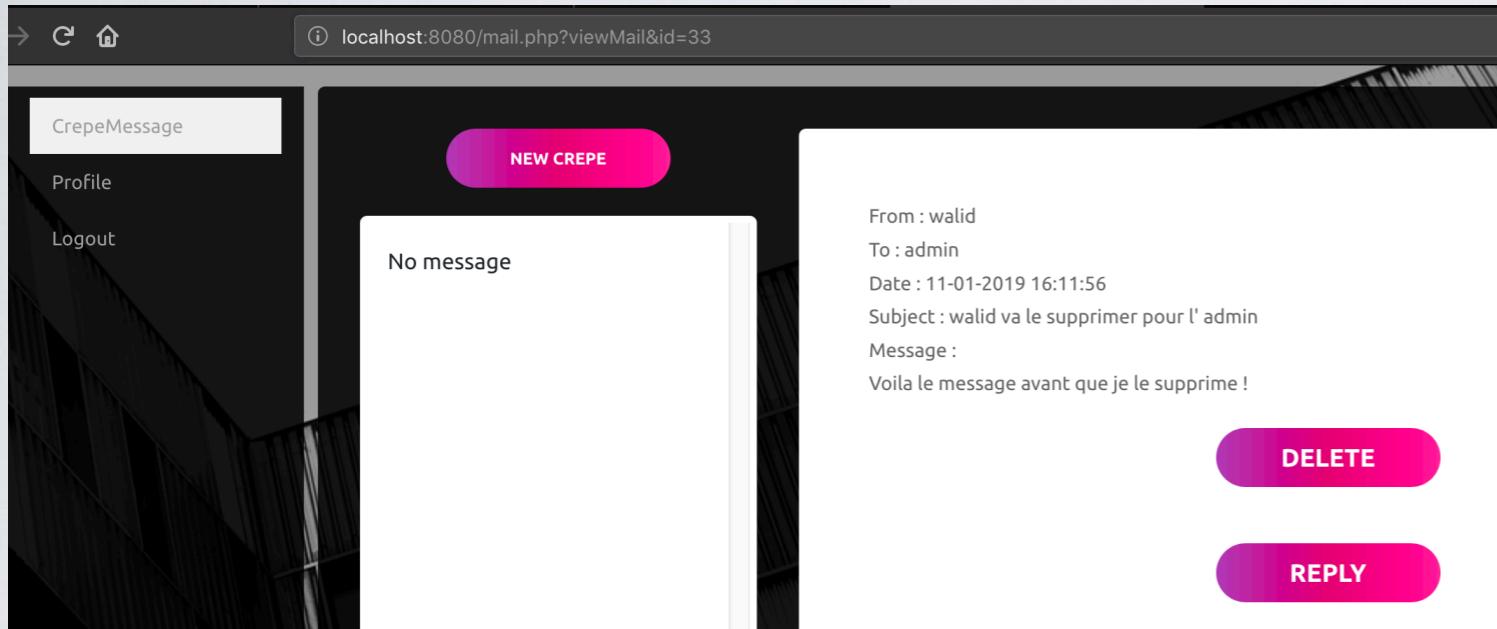
Elément du système attaqué: Transmission des paramètres par l'url.
Il n'y pas de validation du droit d'accès au message côté client.

Motivation: Permet l'accès aux messages qui ne nous sont pas destinés.

Scénario: Il suffit de se loguer avec n'importe quel utilisateur et ensuite on peut avoir accès aux messages de tout le monde. Il suffit de connaître l'id du message mais vu que ceux ci s'incrémentent à chaque nouveau message ils sont prédictives.

Bilan de l'attaque: SUCCESS !

A4 - Accès aux informations des utilisateurs visibles seulement par l'administrateur



Elément du système attaqué: Le formulaire de login (username et password)

Motivation: Permet l'accès aux messages qui ne nous sont pas destinés.

Scénario: Récupérer http://localhost:8080/admin.php?user_id=7

Il est possible de voir et de supprimer des messages dont on aurait pas légitimement accès.

Bilan de l'attaque: SUCCESS !

A5 - Modifier le mot de passe d'un utilisateur choisi

Username : cathia
Role : Collaborator

Password :

APPLY

Username : cathia
Role : Collaborator

CHANGE PASSWORD

Change successfully applied

| | | | | | |
|---|---|--------|----------------------------------|---|---|
| 1 | 7 | cathia | 1234' WHERE username='admin'; -- | 2 | 1 |
|---|---|--------|----------------------------------|---|---|

Elément du système attaqué: On attaque le champs de modification du mot de passe car celui-ci n'est pas protégé contre les injections SQL.

Motivation: On veut prendre le contrôle du compte d'un autre utilisateur.
Avoir accès à un compte administrateur

Scénario: Injection SQL sur le champs afin de modifier un mot de passe.

Bilan de l'attaque: **SUCCESS !**

CONTRE-MESURES APPORTÉES/ SUGGÉRÉES

I. Attaque 3 Faire en sorte que les messages ne soit pas incrementés mais que leur id soit hasher avec une fonction de hashage afin qu'ils ne soient pas prédictibles.

Solution : hasher les id des messages dans le code php

2. Attaque 3 Restreindre l'accès à la pages du message uniquement si l'utilisateur connecté est le destinataire du message.

Solution : On va tester quel utilisateur connecté soit bien le destinataire du message qu'il veut consulter.

```
<?php }  
else if($message->destination_id != $current_user_id){  
    ?>  
    <div class="alert-info100">  
        <p>Access restricted</p>  
    </div>  
<?php }
```

3. Attaque : Limiter le nombre d'essais consécutifs durant une période et bloquer des nouvelles tentatives pendant un certain laps de temps (30 min)

Solution : avec un timeout dans le code php au bout de 3 essais.

CONTRE-MESURES APPORTÉES

4. Attaque 2 Définir une politique de mot de passe plus restrictive et severe (exiger un mot de passe contenant au moins 8 caractères, un chiffre une lettre et un caractère spécial)

Solution proposée : hasher les id des messages dans le code php (mais dépend incrémentation de la DB table)

5. Attaque 7 Contrôler le champs entré par l'utilisateur contre les injections SQL.

Solution : Pour vérifier les entrées contre les injections, utiliser `SQLite3::escapeString`

```
function updatePassword($username, $new_password){  
    $new_password = SQLite3::escapeString($new_password);  
  
    $sql="UPDATE t_user SET password='".$new_password."' WHERE username='".$username"';  
  
    $success = true;
```

ATTACK DEMO TIME !



CONCLUSION

Nous avons effectué un rapport d'analyse des menaces et avant pris le soin d' établir des scénarios d'attaques dans le but de tester la robustesse de notre Crepe Messaging.

Celui ci était vulnérable a de nombreuses failles que nous avons pour la plupart réussit à combler grâce à une série de contre-mesures que nous avons développé.