

# STI – Rapport du Project 2

Michaël da Silva & Guillaume Schranz

## Table des matières

Introduction.....	4
Description du système.....	4
DFD .....	5
Identifier ses biens .....	6
Définir le périmètre de sécurisation.....	6
Sources de menaces .....	6
Scénarios d'attaques .....	6
STRIDE.....	6
Scénario : Saturation mémoire du serveur .....	6
Impacte sur le business .....	6
Source de la menace .....	6
Motivation de l'attaquant .....	7
Cible.....	7
Scénario d'attaque .....	7
Mesures.....	8
Scénario : Injection SQL.....	10
Impacte sur le business .....	10
Source de la menace .....	10
Motivation de l'attaquant .....	10
Cible.....	10
Scénario d'attaque .....	10
Mesures.....	13
Scénario : Mot de passe utilisateur faible.....	14
Impacte sur le business .....	14
Source de la menace .....	14
Motivation de l'attaquant .....	14
Cible.....	14
Scénario d'attaque .....	14
Mesures.....	14
Scénario : Vol d'information dans la base de données .....	15
Impact sur le business .....	15
Source de la menace .....	15
Motivation de l'attaquant .....	15

Cible .....	15
Scénario d'attaque .....	15
Mesures .....	15
Scénario : Authentification, brut force.....	16
Impacte sur le business .....	16
Source de la menace .....	16
Motivation de l'attaquant .....	16
Cible .....	16
Scénario d'attaque .....	16
Mesures .....	16
Scénario : Attaques XSS stockées.....	17
Impact sur le business .....	17
Source de la menace .....	17
Motivation de l'attaquant .....	17
Cible .....	17
Scénario d'attaque .....	17
Mesures .....	17
Scénario : Sniffing .....	18
Impacte sur le business .....	18
Source de la menace .....	18
Motivation de l'attaquant .....	18
Cible .....	18
Scénario d'attaque .....	19
Mesures .....	19
Autres scénarios courants .....	19
Conclusion .....	19

## Introduction

Ce projet fait suite au projet 1 qui était de créer un site web de messagerie simple en PHP / SQLite, qui permettait de faire communiquer plusieurs clients, le tout non sécurisé.

Ici, le but est de tester et corriger les failles de sécurité laissées dans le précédent projet afin de rendre ce site « utilisable » en production. La sécurisation de l'infrastructure tel que le serveur web (configuration Apache, modules utilisés, http, etc.) ou la machine (OS, container, etc.) ne font pas parti des objectifs de sécurisation du projet.

## Description du système

Le site de messagerie permet à des collaborateurs et à des administrateurs d'envoyer des messages entre eux. Les utilisateurs (collaborateurs et administrateur) peuvent avoir accès à leur boîte de réception privée.

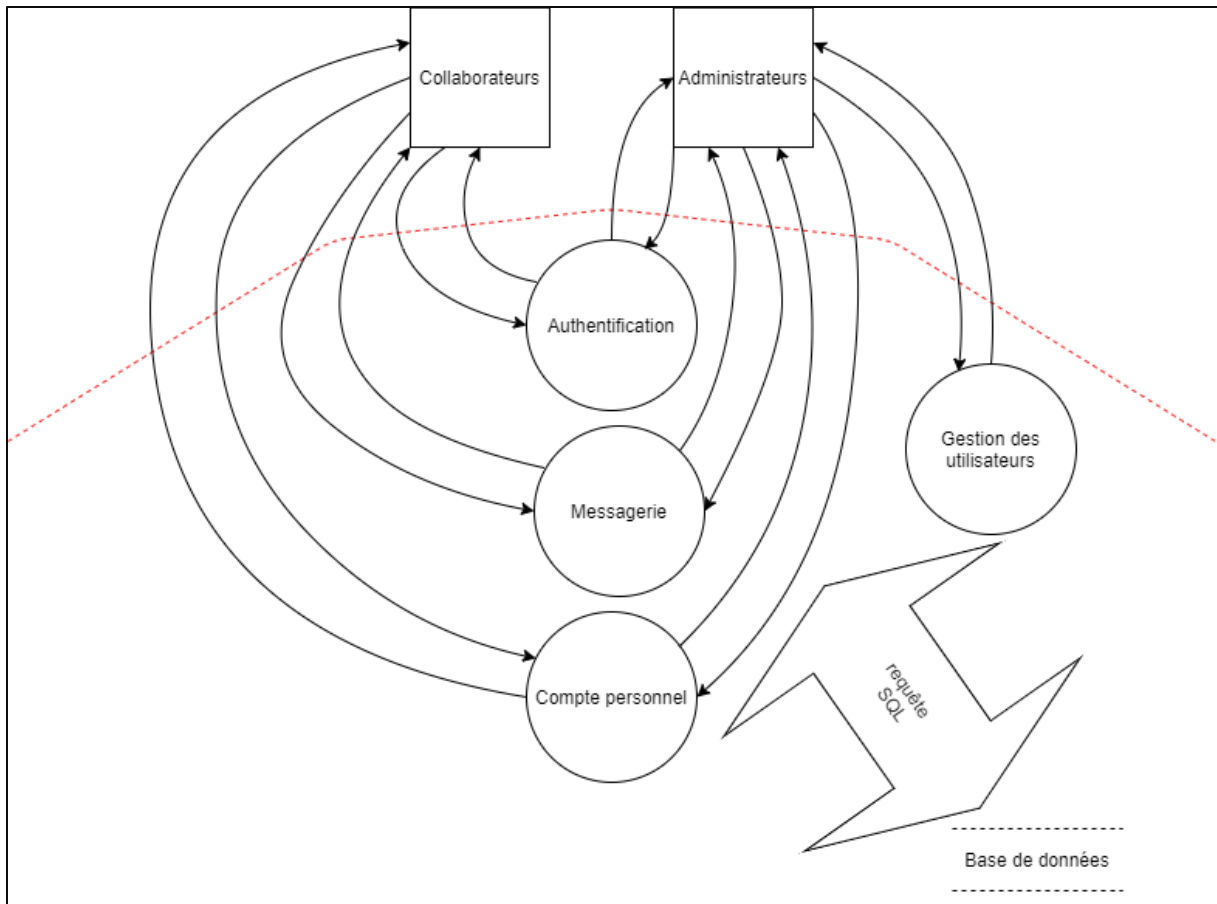
Cette boîte de messagerie affiche les messages triés par date de réception avec la date de réception, l'expéditeur et le sujet du mail. Des boutons pour répondre au message, pour le supprimer, ou bien pour l'ouvrir et découvrir ses détails sont disponibles. Le détail d'un message contient les mêmes informations que précédemment mais avec le corps du message disponible.

Les utilisateurs peuvent écrire des messages à n'importe qui inscrit sur le site web et ce message doit contenir le nom du destinataire, le sujet et le corps du message. Pour finir, un utilisateur a accès à son profil où il peut changer son mot de passe à tout moment.

L'administrateur peut effectuer les mêmes actions que les collaborateurs mais aussi : ajouter / modifier / supprimer un utilisateur qui représenté par un nom d'utilisateur (non modifiable), un mot de passe (modifiable), une validité (modifiable, permet d'accéder ou non au compte) et un rôle (modifiable, collaborateur ou administrateur)).

Ce système est constitué d'un serveur web qui contient l'application de messagerie, ainsi qu'une base de données contenant toutes les données précédentes.

## DFD



La limite de confiance se dessine entre les utilisateurs et les programmes et service de la messagerie.

Tous les processus décrits dans le DFD utilisent la base de données SQLite.

## Identifier ses biens

Dans notre application web, seul deux biens sont à protéger en tout temps :

- Les comptes utilisateurs du site
- Les emails échangés sur la plateforme

Ces données doivent être garanties au niveau de leur intégrité, disponibilité et confidentialité.

## Définir le périmètre de sécurisation

Le périmètre de sécurisation défini par le cahier des charges se limite à l'application elle-même, pas l'infrastructure qui la rend disponible.

Les versions des logiciels ou modules utilisées ne seront pas vérifiées. Toutefois, certaines attaques présentées et leur contre-mesure touchent directement à cette infrastructure car elles sont très importantes pour le bon fonctionnement et la sécurisation du serveur web et de la base de données.

## Sources de menaces

Les principales sources de menaces de notre application web sont les suivantes :

- Utilisateur du système : tous les utilisateurs inscrits sur le site
  - Attaque volontaire (admin malveillant, collaborateurs malin/curieux)
  - Attaque involontaire (mauvaise utilisation, erreurs)
- Personne malveillante : toutes les personnes extérieures au site (hackers, cybercriminels, concurrents)

## Scénarios d'attaques

Les scénarios suivants sont inspirés par les attaques présentées dans le livre « The Web Application Hackers Handbook-2<sup>nd</sup> Edition ». Nous les avons repris afin d'analyser notre système contre ces attaques.

### STRIDE

Attaque	S	T	R	I	D	E
Saturation mémoire du serveur					X	
Injection SQL		X	X			X
Mot de passe utilisateur faible	X		X			
Vol d'information dans la base de données	X					X
Authentification, brut force	X					
Attaques XSS stockées	X	X	X	X		X
Sniffing	X			X		

## Scénario : Saturation mémoire du serveur

Impacte sur le business

Élevé: perte de disponibilité (serveur down, ralentissement, DoS), perte de réputation

Source de la menace

Hacker, cybercriminel, concurrence, utilisateur

## Motivation de l'attaquant

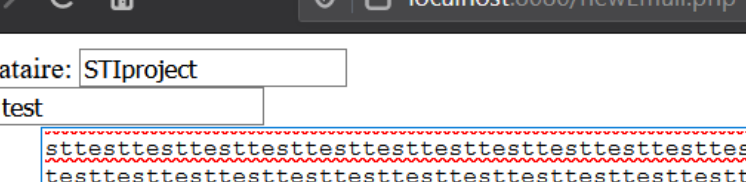
Indisponibilité du service, perte de crédibilité de la société attaquée, indirectement perte financière

## Cible

Serveur(s), service de messagerie

## Scénario d'attaque

Quand un utilisateur écrit ou répond à un email, il doit fournir le sujet et le corps du message à envoyer. Ces champs ne sont pas limités en termes de longueur de caractères. On peut très bien insérer un million de caractères dans un message.

[illegible]

localhost:8080/newEmail.php

Destinataire: STIproject

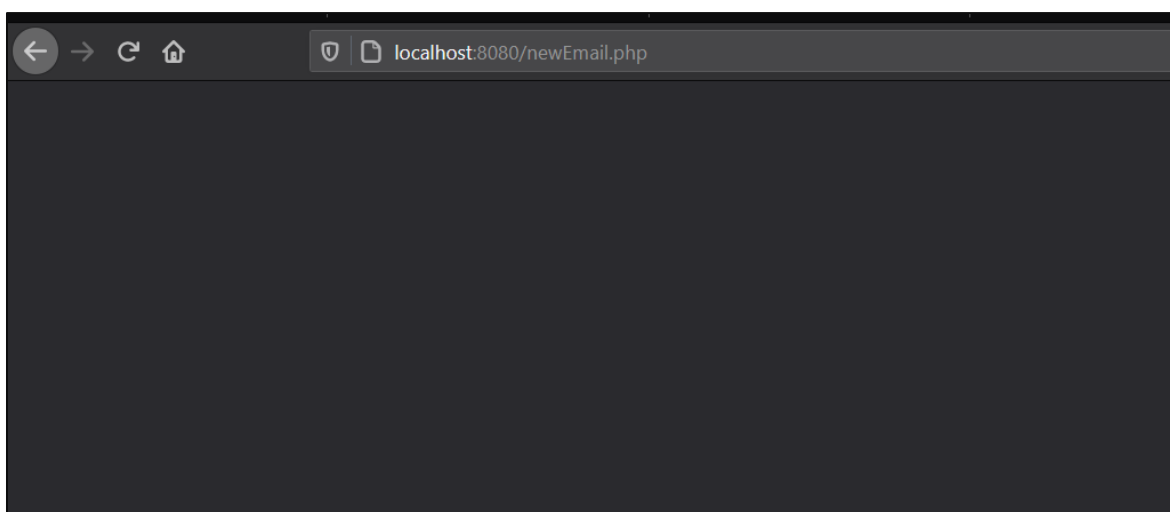
Sujet: test

Message:

Envoyer

[Retour](#)

Ici un exemple d'un texte contenant un million de caractères (texte non-complet sur l'image) que nous donnons au serveur comme corps de l'email.



Après avoir collé notre texte, le serveur ne réagit plus et finit par afficher un écran noir. Seul un redémarrage du conteneur permet de régler le problème.

Cette attaque est aussi possible sur le sujet de l'email :

Destinataire:

Sujet:

Message:

[Retour](#)

[illegible]

## Mesures

Deux modifications ont été apportées au site web : nous limitons dans le formulaire le nombre de caractères possibles, et nous ajoutons une vérification côté serveur pour vérifier que l'utilisateur transmet bien un email avec le nombre de caractère voulu.

```
Sujet: <input type="text" name="subject" maxlength="50" value="<?php echo $subject; ?>"><?php ech
Message: <textarea rows="6" cols="50" name="content" form="newEmail" maxlength="1000"><?php echo
```



```

if(!empty($_POST['subject'])) {
    if(strlen($_POST['subject']) <= 50) {
        $subject = htmlentities($_POST['subject']);
    } else {
        $subject_err = "Limite de caractères atteintes (50)!";
    }
} else {
    $subject_err = "Sujet requis !";
}

if(!empty($_POST['content'])) {
    if(strlen($_POST['content']) <= 1000) {
        $content = htmlentities($_POST['content']);
    } else {
        $content_err = "Limite de caractères atteintes (1000)!";
    }
} else {
    $content_err = "Message vide !";
}

```

Ainsi, l'utilisateur est limité en nombre de caractères pour des champs cibles.

Destinataire:

Sujet:  Limite de caractères atteintes (50)!

Message:

[Retour](#)

Inspector Console Debugger Style Editor Performance Memory Network

Search HTML

<!-- STI-Project2 2021 Groupe: Michaël da Silva & Guillaume Schranz Changement apporté: - Prepare statement caractères dans l'input de l'utilisateur (receiver, subject, content) - htmlentities contre les attaques -->

<html>
<head></head>
<body>
<form id="newEmail" method="post">
Destinataire:
<input type="text" name="receiver" maxlength="50" value="STIproject">
<br>
Sujet:
<input type="text" name="subject" value="">
Limite de caractères atteintes (50)!
<br>
Message:
<textarea rows="6" cols="50" name="content" form="newEmail" maxlength="1000">test</textarea>
<br>

The screenshot shows a web application interface for sending an email. The form has three input fields: 'Destinataire:' with the value 'STIproject', 'Sujet:' with the value 'test', and a large 'Message:' text area. Below the text area, there is a message 'Limite de caractères atteintes (1000)!', a 'Envoyer' button, and a 'Retour' link.

The browser's developer tool is open, showing the HTML source code. The code is as follows:

```
<!--
STI-Project2 2021 Groupe: Michaël da Silva & Guillaume Schranz Changement apporté: - Prepare statement contre les injections SQL - Ajout de limite de
caractères dans l'input de l'utilisateur (receiver, subject, content) - htmlentities contre les attaques XSS
-->
<html>
<head></head>
<body>
  <form id="newEmail" method="post">
    Destinataire:
    <input type="text" name="receiver" maxlength="50" value="STIproject">
    <br>
    Sujet:
    <input type="text" name="subject" maxlength="50" value="test">
    <br>
    Message:
    <textarea rows="6" cols="50" name="content" form="newEmail" maxlength="1000"></textarea>
    <br>
    Limite de caractères atteintes (1000)!
```

## Scénario : Injection SQL

Impacte sur le business

Normal à élevé, vol de login/mot de passe, adresse email, coordonnées bancaires, perte de crédibilité, ...

Source de la menace

Hacker, cybercriminel, concurrence

Motivation de l'attaquant

Revente d'information, se faire passer pour autrui, perte de crédibilité de la société victime de l'attaque.

Cible

Base de données, informations des utilisateurs

Scénario d'attaque

Un utilisateur malveillant peut tester les champs qui lui sont disponibles avec des injections SQL connus afin que le serveur SQL retourne des informations stockées, ici des emails sensibles ou des informations utilisateurs (comptes).

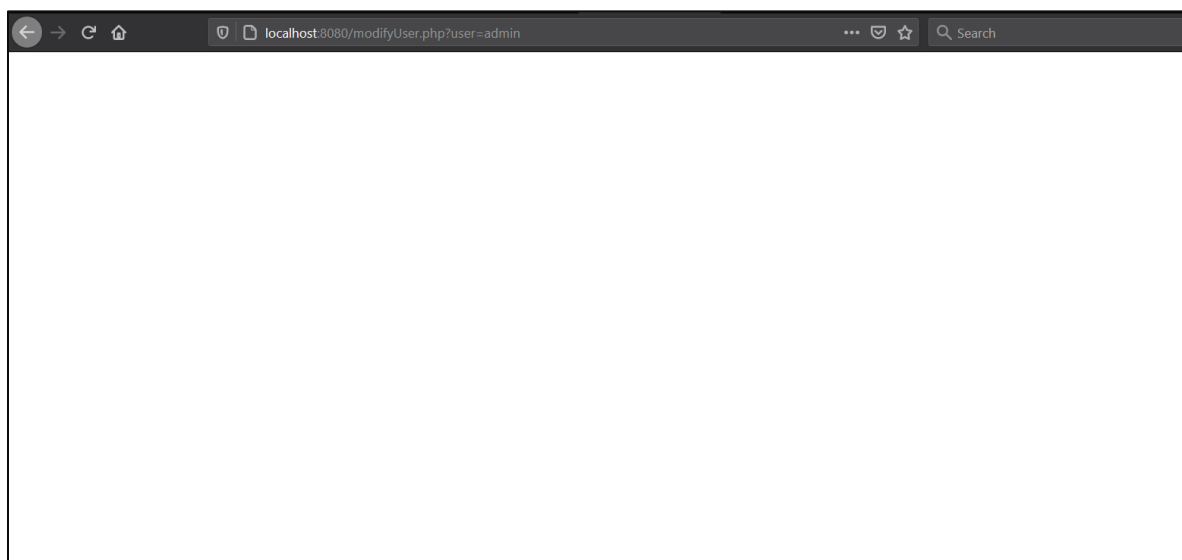
Nous avons testé plusieurs champs sur le site web et il semblerait que certains champs ne répondaient pas à nos injections. Nous avons toutefois pu exploiter un champ disponible aux utilisateurs : le champ de changement de mot de passe pour un compte, disponible pour les administrateurs du site.

The screenshot shows a web application interface for modifying a user. At the top, there are navigation links: "Home", "Profil", and "Nouveau message". The main form contains the following elements:

- Nom d'utilisateur:** A text input field.
- Password:** A password input field with 10 dots representing the masked text.
- Account Status:** Two radio buttons: ☒ "Compte inactif" and ☐ "Compte actif".
- Role:** Two radio buttons: ☐ "Collaborateur" and ☐ "Administrateur".
- Buttons:** Two blue buttons labeled "Modifier l'utilisateur" and "Annuler".

On the right side of the form, a small text editor window titled "\*Sans titre" is open. It contains the text: `Fichier Editio` and `test" --|`. This represents a SQL injection payload.

En tapant la commande sur la droite (test" --), cela permet de changer le mot de passe de tous les comptes par test.



On obtient une page blanche après le changement de mot de passe, donc aucune erreur SQL. En vérifiant la base de données, on s'aperçoit que les mots de passe de tous les utilisateurs ont été changé.

**phpLiteAdmin v1.9.6** Database 1 → account

Documentation | License | Project Site

Change Database  
[rw] Database 1 (↓)

Database 1  
[Table] account  
[Table] messages  
[Table] role

Logout

Browse Structure SQL Search Insert Export Import Rename Empty

Show : 30 row(s) starting from record # 0 as a Table

Showing rows 0 - 2, Total: 3 (Query took 0.0002 sec)  
SELECT \* FROM "account" LIMIT 0, 30

	← T →	username	password	validity	role_id
<input type="checkbox"/>	Edit Delete	admin	admin	1	2
<input type="checkbox"/>	Edit Delete	STIproject	sti	1	1
<input type="checkbox"/>	Edit Delete	test2	test	0	1

Check All / Uncheck All With Selected: Edit Go

Powered by phpLiteAdmin | This is free software. Please donate. | Page generated in 0.0803 seconds.

**phpLiteAdmin v1.9.6** Database 1 → account

Documentation | License | Project Site

Change Database  
[rw] Database 1 (↓)

Database 1  
[Table] account  
[Table] messages  
[Table] role

Logout

Browse Structure SQL Search Insert Export Import Rename Empty Drop

Show : 30 row(s) starting from record # 0 as a Table

Showing rows 0 - 2, Total: 3 (Query took 0.0001 sec)  
SELECT \* FROM "account" LIMIT 0, 30


	← T →	username	password	validity	role_id
<input type="checkbox"/>	Edit Delete	admin	test	1	2
<input type="checkbox"/>	Edit Delete	STIproject	test	1	1
<input type="checkbox"/>	Edit Delete	test2	test	0	1


Check All / Uncheck All With Selected: Edit Go

Powered by phpLiteAdmin | This is free software. Please donate. | Page generated in 0.0019 seconds.

En se déconnectant de notre compte et en se connectant à un compte en utilisant le mot de passe précédent, on peut alors accéder à n'importe quel compte disponible sur la messagerie (ici, nous avons testé le compte admin).

# STI-mail

 I'm not a robot

  
reCAPTCHA  
[Privacy](#) - [Terms](#)

Login

STI-mail Home Profil Nouveau message Gestion d'utilisateur Logout

Date	Expéditeur	Sujet	Actions
2020-10-15	michael.dasilva	test	<div>Details Repondre Supprimer</div>

On accède alors à sa boîte de messagerie.

## Mesures

Nous avons simplement modifié toutes les parties du code PHP qui utilisent la base de données en ajoutant la fonction `prepare()`. Elle permet d'obtenir une requête SQL à envoyer sans les champs qui permettent une injection SQL.

```
if(!empty($_POST['password'])){
    $db->query("UPDATE account SET password='".$_POST['password']."' WHERE username='".$_SESSION['user']."'");
}

if(empty($password_err)){
    $stmt = $db->prepare("UPDATE account SET password=? WHERE username=?");
    $hash = password_hash($password, PASSWORD_DEFAULT);
    $stmt->bindParam(1, $hash);
    $stmt->bindParam(2, $_SESSION['user']);
    $stmt->execute();
}
```

## Scénario : Mot de passe utilisateur faible

### Impacte sur le business

Normal à élevé: perte de confidentialité (données sensibles, potentiellement secrets industriels), perte d'intégrité (email forgé), perte de l'authentification (vol d'identité, si admin : suppression ou modification de compte)

### Source de la menace

Hacker, cybercriminel, concurrence

### Motivation de l'attaquant

S'amuser, se faire passer pour autrui, spam des clients, modification d'informations, perte de crédibilité de la société victime de l'attaque

### Cible

Contenu des emails, informations de l'utilisateur, si admin : dégât sur l'application web

### Scénario d'attaque

Un utilisateur malveillant peut brute-force certains comptes qui aurait des mots de passe faible car aucune politique sur la création d'un mot de passe fort n'existe sur la messagerie. Ainsi, selon le compte ciblé, l'attaquant pourrait envoyer des emails forgés à des cibles, voire modifier ou supprimer des comptes utilisateurs.

### Mesures

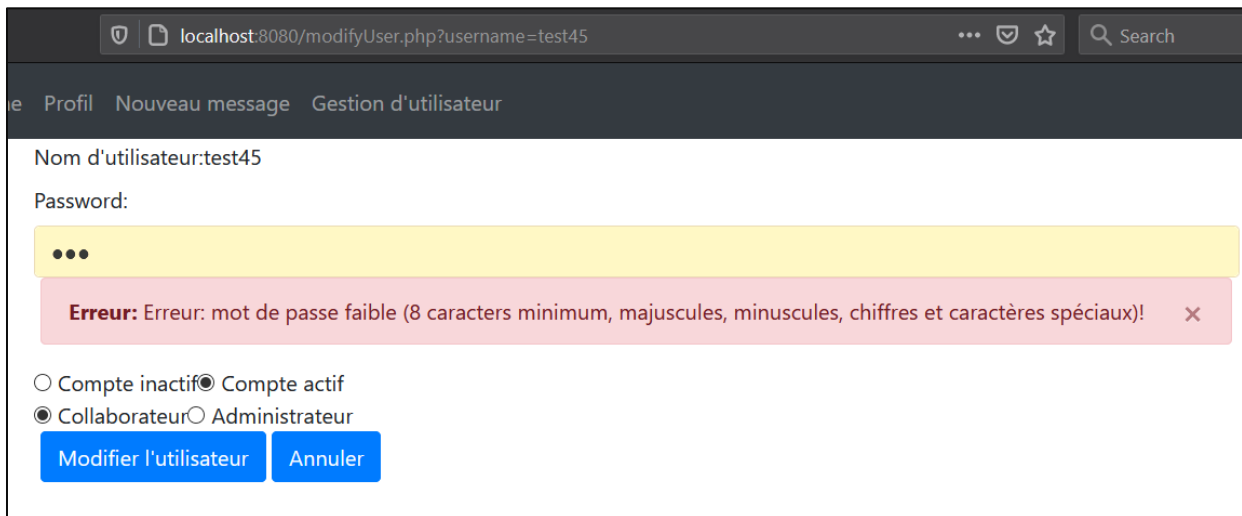
Mettre en place une fonction de vérification de mot de passe lors de la création de compte ou lors d'un changement de mot de passe.

Nous avons mis en place une fonction appelée à chaque fois qu'un mot de passe est créé ou modifié par des utilisateurs. Cette fonction est dans le fichier **checkPass.php**.

```
<?php
function check_mdp_format($mdp)
{
    $majuscule = preg_match('@[A-Z]@', $mdp);
    $minuscule = preg_match('@[a-z]@', $mdp);
    $chiffre = preg_match('@[0-9]@', $mdp);
    $specialChars = preg_match('@[^\w]@', $mdp);

    if(!$majuscule || !$minuscule || !$chiffre || !$specialChars || strlen($mdp) < 8)
        return false;
    else
        return true;
}
?>
```

Elle permet d'avoir des mots de passe d'au moins 8 caractères, avec au moins une majuscule, une minuscule, un chiffre et un caractère spécial.



localhost:8080/modifyUser.php?username=test45

Profil Nouveau message Gestion d'utilisateur

Nom d'utilisateur: test45

Password:

Erreur: mot de passe faible (8 caractères minimum, majuscules, minuscules, chiffres et caractères spéciaux)!

☐ Compte inactif
 ☒ Compte actif

☒ Collaborateur
 ☐ Administrateur

Modifier l'utilisateur Annuler

## Scénario : Vol d'information dans la base de données

### Impact sur le business

Normal à élevé, vol de login/mot de passe, adresse email, coordonnées bancaires, perte de crédibilité, ...

### Source de la menace

Hacker, cybercriminel, concurrence

### Motivation de l'attaquant

Revente d'information, se faire passer pour autrui, perte de crédibilité de la société victime de l'attaque.

### Cible

Base de données, informations des utilisateurs

### Scénario d'attaque

Un utilisateur malveillant arrive à avoir accès à la base de données et peut voir l'association user/mot de passe. Il peut ensuite se connecter en tant que la victime et effectuer diverses transactions en son nom. Il peut également utiliser les données trouvées sur d'autre site car trop peu de monde change de mot de passe sur toutes les applications sur lesquelles on se connecte.

### Mesures

Hasher les mots de passe et autres informations sensibles dans la base de données.

Nous utilisons la fonction password\_hash de php.

```
$hash = password_hash($_POST['password'], PASSWORD_DEFAULT);
```

Les mots de passe sont correctement hashé dans la DB

username	password
admin	\$2y\$10\$4LOkl0m4MdVlhZzqmW5BHe3k92bVxAVoMiHF1qe.H1JKoKrN1hRiy
STIproject	\$2y\$10\$/wQHaPpKCSnsXXzoNFNYzu12liptkc4NQWUnQE3KffAdDboNcT7.

## Scénario : Authentification, brut force

Impacte sur le business

Normal à élevé, vol d'identité. Elevé si connexion en tant qu'admin

Source de la menace

Hacker, cybercriminel, concurrence, autres utilisateurs

Motivation de l'attaquant

Se faire passer pour autrui, utiliser le compte de quelqu'un d'autre. Si admin, modifier/créer/supprimer compte utilisateurs

Cible

Comptes des utilisateurs

Scénario d'attaque

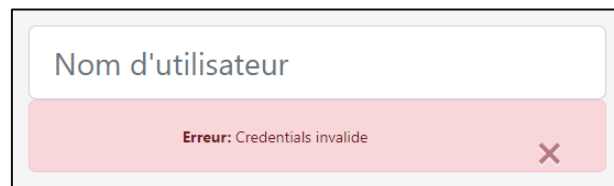
Un hacker arrive sur le site et tente de se connecter à un compte qui n'est pas le sien grâce au brute force.

Mesures

Nous avons implémenté 3 mesures distinctes.

*Indication de l'erreur*

L'indication d'une erreur est générique et ne précise plus si c'est le nom d'utilisateur ou le mot de passe qui est incorrect. Cela évite de savoir que l'utilisateur existe.

A screenshot of a web form with a light gray border. Inside, there is a white input field with the placeholder text "Nom d'utilisateur". Below the input field is a red rectangular error message box containing the text "Erreur: Credentials invalide" and a small red 'X' icon on the right.

*Mot de passe complexe*

Les mots de passe doivent avoir un niveau de complexité acceptable soit au moins :

8 caractères

- 1 majuscules
- 1 minuscule
- 1 chiffre

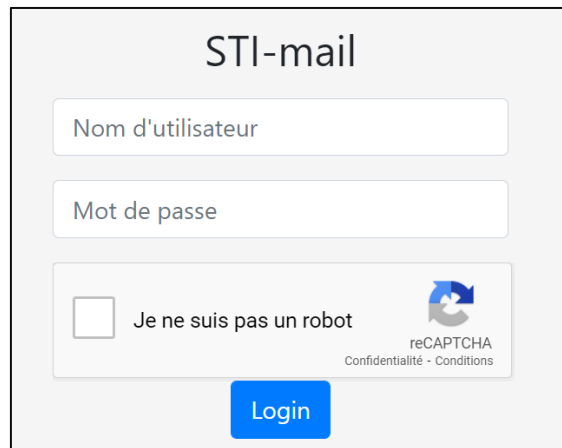
```
$majuscule = preg_match('@[A-Z]@', $mdp);  
$minuscule = preg_match('@[a-z]@', $mdp);  
$chiffre = preg_match('@[0-9]@', $mdp);  
  
if(!$majuscule || !$minuscule || !$chiffre || strlen($mdp) < 8)  
|   return false;  
else  
|   return true;
```

Cela devrait être complété de caractères spéciaux et de vérification que le mot de passe ne comporte pas de mot courant.



### ReCaptcha

Nous avons finalement mis en place un reCaptcha en utilisant l'API homonyme de Google.



## Scénario : Attaques XSS stockées

### Impact sur le business

Très élevé, permet à un hacker de tout faire en escaladant (vol d'identité/information, connexion en tant qu'admin, dump DB, rendre le site down, mise en place de malware/keylogger,... sur le site, et bien d'autres

### Source de la menace

Hacker, cybercriminel, concurrence

### Motivation de l'attaquant

Revente/vol d'information, se faire passer pour autrui, perte de crédibilité de la société victime de l'attaque,

### Cible

Informations utilisateurs, accès privilégiés, compromission du site.

### Scénario d'attaque

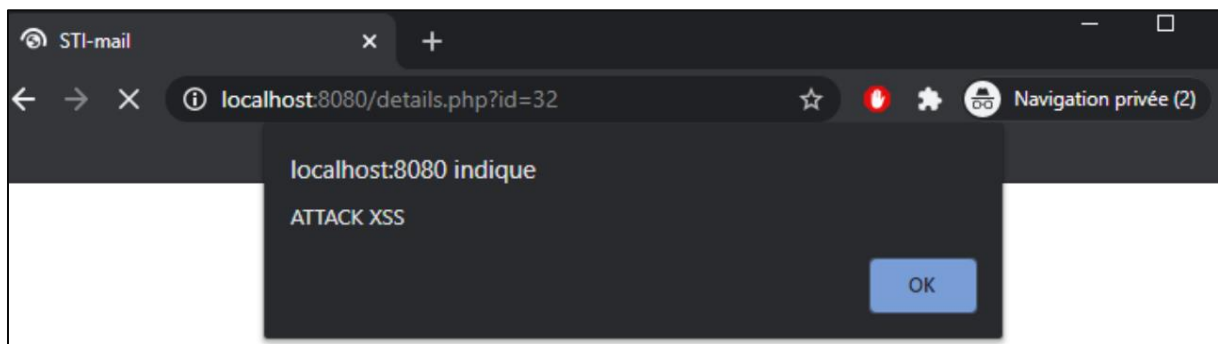
Un utilisateur utilise des champs de formulaire pour envoyer des scripts directement dans la base de données. Lors de la récupération de ses informations pour les afficher (message, nom d'utilisateur,...), un utilisateur déclenche sans le savoir un script. Les possibilités sont ensuite multiples. Keylogger, escalade, vol de cookies, ...

### Mesures

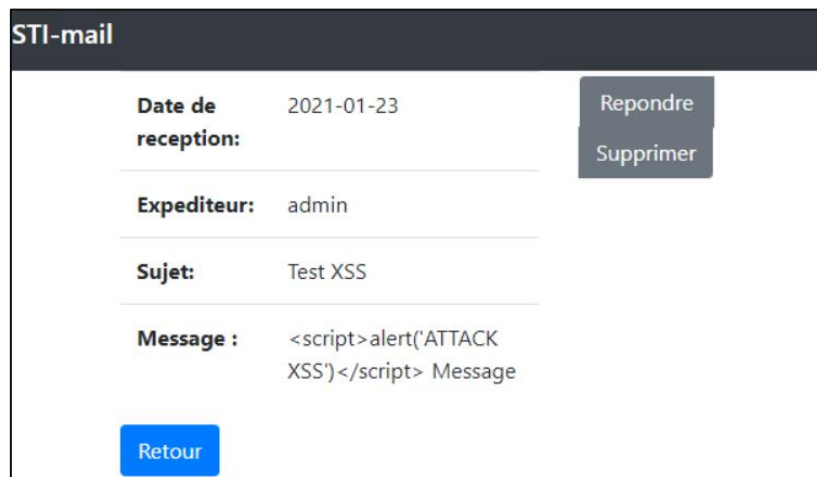
Nettoyer toutes les entrées possibles. Suppression des caractères particulier, limitations de la taille des champs.

Nous avons utilisé des htmlentities() sur toutes les entrées. Nous l'avons également fait lorsqu'une information de la base de données est affichée sur le site. La raison est qu'il est également possible de compromettre là-bas de donnée sans passer par une attaque XSS et de modifier les données qui s'y trouvent afin d'effectuer une attaque XSS stockées.

Avant implémentation, ouverture d'un mail



Après implémentation, ouverture d'un mail



Et le contenu du message correspondant

```
<th>Sujet:</th>
<td>Test XSS</td>
</tr>
<tr>
<th>Message :</th>
<td>&lt;script&gt;alert('ATTACK XSS')&lt;/script&gt;
```

## Scénario : Sniffing

Impacte sur le business

Normal à élevé, vol d'identité. Elevé si connexion en tant qu'admin

Source de la menace

Hacker, cybercriminel, concurrence, autres utilisateurs

Motivation de l'attaquant

Compromission de compte, escalade de privilège, vol de donnée

Cible

Comptes des utilisateurs

## Scénario d'attaque

Un hacker arrive à se connecter sur le réseau et sniffer les trames. Après analyse, il découvre des informations telles que des credentials, contenu de message, etc...

## Mesures

Nous n'avons pas implémenté cette fonctionnalité il faudrait effectuer un passage vers https avec tout ce que cela implique, certificat SSL, ...

## Autres scénarios courants

Attaques côté client : nous n'avons pas trouvé de potentielle attaque côté client, les contrôles se faisant côté serveur.

Mot de passe par défaut : Il s'agit de changer les mots de passe par défaut des divers systèmes que nous utilisons. Typiquement, le mot de passe pour accéder à phpliteadmin est « admin ». Il suffit de changer le mot de passe du fichier **phpliteadmin.php**.

```
//password to gain access  
$password = 'admin';
```

A noter que nous ne l'avons pas fait ici pour laisser un accès facilité lors de tests ou de corrections.

## Conclusion

Il était très intéressant de pouvoir se mettre d'une part à la place d'un développeur, d'autre part celle d'un hacker. Nous avons pu nous rendre compte par nous-même de la difficulté et du temps à investir afin de pouvoir fournir un site web avec un minimum de problème de sécurité.

Nous sommes satisfaits d'avoir pu couvrir ce qui nous semblait être les menaces les plus courantes bien qu'il reste toujours des points à traiter. Ces points auraient nécessité beaucoup plus de temps et, dans un cas réel en entreprise, cela correspondrait à un budget supplémentaire. Néanmoins, la sécurité parfaite n'existe pas et il faudra toujours garder en tête le rapport entre ce que nous pouvons investir pour la sécurité et l'impacte que cela aurait sur un business quel qu'il soit.