

---

# SÉCURITÉ DES TECHNOLOGIES INTERNET (STI)

Dans ce rapport, vous découvrirez la vie d'une application web,  
ses problèmes et la manière dont ils ont été réglés.

## Projet 2

---

## 1. TABLE DES MATIÈRES

1.	Table des matières	1
2.	Introduction	3
2.1.	Environnement de test	3
2.1.1.	Technologie	3
2.1.2.	L'application en bref	3
3.	Décrire le système	4
3.1.	Quels sont les objectifs du système	4
3.2.	Hypothèses de sécurité	4
3.3.	Quelles sont les exigences de sécurité du système	4
3.4.	Comment est constitué le système	4
3.5.	Rôles des utilisateurs	4
3.6.	DFD	5
3.7.	Identifier ses biens	5
3.8.	Définir le périmètre de sécurisation	5
4.	Identifier les sources de menaces	6
5.	Identifier les scénarios d'attaques	7
5.1.	Scénario de menace 1 : Credentials spoofing	7
5.1.1.	Impact business	7
5.1.2.	Source de la menace	7
5.1.3.	Motivation	7
5.1.4.	Actif ciblé	7
5.1.5.	Scénarios d'attaques :	7
5.1.6.	Contre mesure	7
5.2.	Scénario de menace 2 : Database tampering	8
5.2.1.	Impact business	8
5.2.2.	Source de la menace	8
5.2.3.	Motivation	8
5.2.4.	Actif ciblé	8
5.2.5.	Scénarios d'attaques	8
5.2.6.	Contre mesure	8
5.3.	Scénario de menace 3 : Information disclosure	9
5.3.1.	Impact business	9

---

5.3.2.	Source de la menace	9
5.3.3.	Motivation	9
5.3.4.	Actif ciblé	9
5.3.5.	Scénarios d'attaques	9
5.3.6.	Contre mesure	9
5.4.	Scénario de menace 4 : DoS	10
5.4.1.	Impact businessse	10
5.4.2.	Source de la menace	10
5.4.3.	Motivation	10
5.4.4.	Actif ciblé	10
5.4.5.	Scénarios d'attaques	10
5.4.6.	Contre mesure	10
5.5.	Démonstration des attaques	11
5.5.1.	Sniffing (récupération des identifiants d'un utilisateur)	11
5.5.2.	Code injection (XSS : vol de token de session)	11
5.5.3.	Répudiation (Insertions, suppressions, modifications de données par injection SQL)	12
5.5.4.	Horizontal privilege escalation (Suppression de messages)	13
5.5.5.	CSRF (Suppression d'utilisateur)	14
5.5.6.	Sniffing (Récupération de données confidentielle)	15
5.5.7.	Code injection (déni de service par injection HTML, redirection de page)	16
6.	Identifier les contre-mesures	17
6.1.	Sanitarisation des entrées utilisateur	17
6.2.	Injection SQL : Prepared Statement	17
6.3.	Répudiation	17
6.4.	Token anti CSRF	17
7.	Conclusion	18
8.	Table des illustrations	19
9.	Annexe :	19

---

## 2. INTRODUCTION

Ce travail faisant suite au **projet 1**, dont le but était de réaliser une application de messagerie Web pour entreprise, consiste à la reprendre afin d'y effectuer une analyse de menace complètes. Il sera, par la suite possible d'y apporter les aspects sécuritaires manquant.

Dans le cadre de ce projet, nous allons uniquement sécuriser l'**application**. Les risques encourus par le serveur web ainsi que la machine ne seront, quant à eux, pas corrigé. Eventuellement, nous proposeront des recommandations quant à la sécurité de l'infrastructure complètes plus tard dans le rapport.

Notre équipe, composée de **Dejvid Muaremi** et **Loïc Schürch** réaliseront l'audit de sécurité de l'application réalisée par **Dejvid Muaremi** et **Romain Silvestri**.

### 2.1. ENVIRONNEMENT DE TEST

#### 2.1.1. TECHNOLOGIE

- HTML
- PHP
- Javascript
- Docker
- SQLite

#### 2.1.2. L'APPLICATION EN BREF

Lorsque l'utilisateur souhaite utiliser l'application, une authentification simple sera nécessaire afin d'y accéder. Une fois la page de login passée, celui-ci pourra recevoir et envoyer des emails tout ce qu'il y a de plus ordinaire ainsi que de changer son mot de passe lui permettant d'accéder à l'application.

Un administrateur aura, en plus de la gestion de sa boîte mail, la possibilité de gérer les utilisateurs, ils seront responsables de l'ajout, la modification, et la suppression des utilisateurs et administrateurs.

### 3. DÉCRIRE LE SYSTÈME

#### 3.1. QUELS SONT LES OBJECTIFS DU SYSTÈME

Comme indiqué précédemment, le but premier de l'application est d'offrir aux utilisateurs une interface web permettant d'échanger des emails professionnels au sein de l'entreprise.

#### 3.2. HYPOTHÈSES DE SÉCURITÉ

Nous partons de l'hypothèse audacieuse que réseau interne et les administrateurs sont dignes de confiance mais aussi le système d'exploitation et le serveur web qui contiennent l'application.

#### 3.3. QUELLES SONT LES EXIGENCES DE SÉCURITÉ DU SYSTÈME

- Les fonctionnalités réservées aux collaborateurs doivent être accessibles uniquement par le personnel de l'entreprise.
- Les fonctionnalités réservées aux administrateurs doivent être accessibles uniquement par les administrateurs.
- Les emails transmis doivent être fiables, intègres et non répudiables par l'expéditeur et le destinataire.
- Le site Web doit être disponible autant que possible, dans le cadre du projet nous proposons une disponibilité best effort.
- Les données des utilisateurs doivent être protégées.

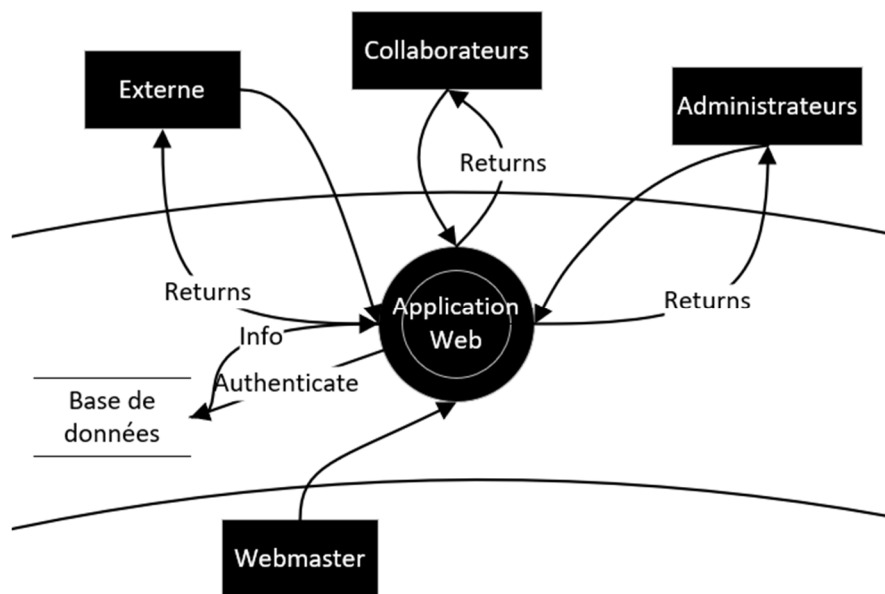
#### 3.4. COMMENT EST CONSTITUÉ LE SYSTÈME

Dans le cadre du projet, nous utilisons une image docker qui contient le service pour lancer un serveur PHP et qui met en place une version locale de l'application web ainsi qu'une base de données MySQLite contenant les utilisateurs et les emails.

#### 3.5. RÔLES DES UTILISATEURS

- Administrateurs de l'application. (Qui gère l'application)
- Collaborateur de l'entreprise. (Qui utilise l'application)
- Webmaster. (Qui construit le site web)
- Externe. (Attaquant potentiel)

### 3.6. DFD



### 3.7. IDENTIFIER SES BIENS

- Les données des utilisateurs.
- Les emails échangés qui peuvent potentiellement contenir des données sensibles de l'entreprise.

### 3.8. DÉFINIR LE PÉRIMÈTRE DE SÉCURISATION

Dans le cadre du projet, nous allons nous contenter de l'applications web, il sera éventuellement possible de commenter la sécurité de l'infrastructure lorsque ceci est nécessaire ou lorsque la sécurité de l'application est intrinsèquement liée à celle-ci.

---

#### 4. IDENTIFIER LES SOURCES DE MENACES

- Hackers, script-kiddies
  - Motivation : s'amuser, gloire
  - Cible : n'importe quel élément / actif
  - Potentialité : haute
- Cybercrime (spam, maliciels)
  - Motivation : financières
  - Cible : vol de crédenciales du client, spam des clients, modification d'informations
  - Potentialité : basse
- Utilisateurs malins
  - Motivation : accès aux fonctionnalités d'administration
  - Cible : escalade de privilège verticale/horizontale
  - Potentialité : haute
- Concurrent / espionnage industriel
  - Motivation : vol de données confidentiel, entacher la réputation de l'entreprise, leak de mails sensibles
  - Cible : Contenu de la base de données
  - Potentialité : moyenne

---

## 5. IDENTIFIER LES SCÉNARIOS D'ATTAQUES

### 5.1. SCÉNARIO DE MENACE 1 : CREDENTIALS SPOOFING

#### 5.1.1. IMPACT BUSINESS

Moyen : perte de confiance, perte de réputation

#### 5.1.2. SOURCE DE LA MENACE

Hackers, script kiddies, concurrence, espionnage industriel

#### 5.1.3. MOTIVATION

Curiosité, challenge, financière

#### 5.1.4. ACTIF CIBLÉ

Table des utilisateurs de la base de données

#### 5.1.5. SCÉNARIOS D'ATTAQUES :

Sniffing (récupération des identifiants d'un utilisateur)

Code injection (récupération des mots de passe par injection SQL + collision SHA1)

Brute force

Code injection (XSS : vol de token de session)

#### 5.1.6. CONTRE MESURE

Mise en place de HTTPS, politique de mot de passe, protection contre la brute force, validation des saisie utilisateur.



---

## 5.2. SCÉNARIO DE MENACE 2 : DATABASE TAMPERING

### 5.2.1. IMPACT BUSINESS

Haut : corruption des données, perte de temps et d'argent pour la restauration

### 5.2.2. SOURCE DE LA MENACE

Hacker, crime organisé, concurrence, espionnage industriel

### 5.2.3. MOTIVATION

Challenge, gloire, financière

### 5.2.4. ACTIF CIBLÉ

Contenu de la base de données

### 5.2.5. SCÉNARIOS D'ATTAQUES

Répudiation (Insertions, suppressions, modifications de données par injection SQL)

Horizontal privilege escalation (Suppression de messages)

CSRF (Suppression d'utilisateur)

### 5.2.6. CONTRE MESURE

Utilisation de prepared statement, demande de confirmation avant une suppression, utilisation de token anti CSRF, utilisation du referrer.

---

### 5.3. SCÉNARIO DE MENACE 3 : INFORMATION DISCLOSURE

#### 5.3.1. IMPACT BUSINESS

Très haut : leak de message confidentiel, perte de l'image et de confiance envers l'entreprise.

#### 5.3.2. SOURCE DE LA MENACE

Espionnage industriel, concurrence, crime organisé

#### 5.3.3. MOTIVATION

Financière, élimination de la concurrence

#### 5.3.4. ACTIF CIBLÉ

Table des emails dans la base de données

#### 5.3.5. SCÉNARIOS D'ATTAQUES

Sniffing (Récupération de données confidentielle)

Code injection (XSS)

#### 5.3.6. CONTRE MESURE

Chiffrement des données, mise en place de HTTPS, validation des saisie utilisateur, prepared statement.

---

## 5.4. SCÉNARIO DE MENACE 4 : DoS

### 5.4.1. IMPACT BUSINESSSE

Faible : Interruption temporaire du service de messagerie pour un nombre d'utilisateur limité, jusqu'à la suppression du message fautif.

### 5.4.2. SOURCE DE LA MENACE

Interne à l'entreprise, hacker, script kiddies, smart user

### 5.4.3. MOTIVATION

S'amuser, embêter un collègue

### 5.4.4. ACTIF CIBLÉ

Interface web de l'application

### 5.4.5. SCÉNARIOS D'ATTAQUES

Code injection (déni de service par injection HTML, redirection de page)

Code injection (XSS redirection de page)

### 5.4.6. CONTRE MESURE

Sanitarisation des saisies utilisateur.

## 5.5. DÉMONSTRATION DES ATTAQUES

### 5.5.1. SNIFFING (RÉCUPÉRATION DES IDENTIFIANTS D'UN UTILISATEUR)

Lors du login d'un utilisateur, il est possible de récupérer les credentials d'un utilisateur avec un simple sniffeur de réseau tel que Wireshark.

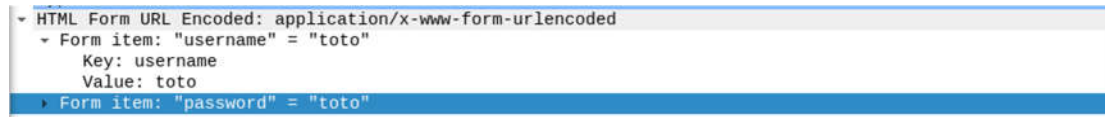


FIGURE 1: RÉCUPÉRATION DES CRÉDENTIALS

### 5.5.2. CODE INJECTION (XSS : VOL DE TOKEN DE SESSION)

En envoyant le bon payload dans un email, il est possible d'exécuter du script, par conséquent, il est possible de faire pratiquement tout ce que l'on veut avec l'application.

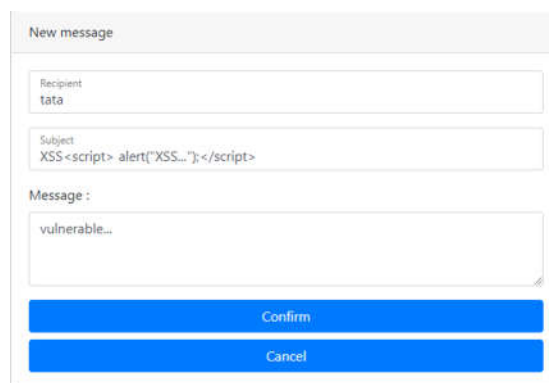


FIGURE 2: PAYLOAD XSS

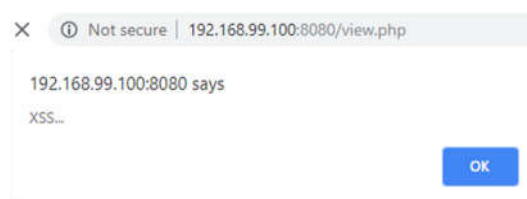


FIGURE 3: DEMONSTRATION XSS

### 5.5.3. RÉPUDIATION (INSERTIONS, SUPPRESSIONS, MODIFICATIONS DE DONNÉES PAR INJECTION SQL)

Il est possible sur certaines fonctionnalités de l'application d'insérer des données dans la base de données par injection SQL, comme on peut le voir sur l'image suivante.

New message

Recipient  
tata

Subject  
, 'sql injection...'; '2020-06-10 10:10:10'); INSERT INTO messageSent (sender, recei

Message :  
hacked!

Confirm

Cancel

FIGURE 4: PAYLOAD INJECTION SQL

	← T →	sender	receiver	idMessage
<input type="checkbox"/>	Edit Delete	user	user	1
<input type="checkbox"/>	Edit Delete	user	user	2
<input type="checkbox"/>	Edit Delete	tata	user	5
<input type="checkbox"/>	Edit Delete	user	user	6
<input type="checkbox"/>	Edit Delete	user	user	7
<input type="checkbox"/>	Edit Delete	admin	admin	8
<input type="checkbox"/>	Edit Delete	tata	toto	11
<input type="checkbox"/>	Edit Delete	tata	tata	12
<input type="checkbox"/>	Edit Delete	attacker	tata	99

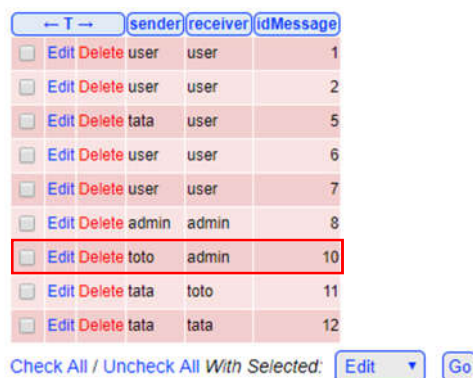
Check All / Uncheck All With Selected: Edit Go

FIGURE 5: DEMONSTRATION INJECTION SQL

L'injection utilisée est la suivante : ', 'hacked', '2020-06-10 10:10:10'); INSERT INTO messageSent (sender, receiver, idMessage) VALUES ('attacker', 'tata', '99');

#### 5.5.4. HORIZONTAL PRIVILEGE ESCALATION (SUPPRESSION DE MESSAGES)

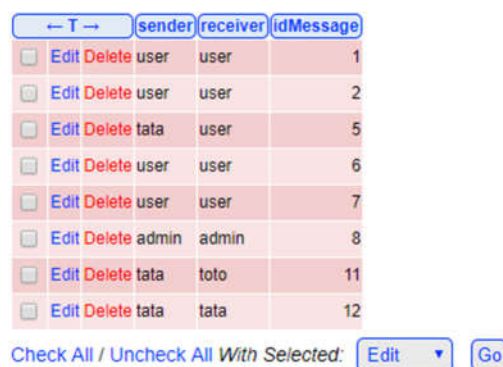
Comme on peut le voir sur les images ci-dessous, il est possible de supprimer un email quelconque en utilisant l'url suivant : </validate-delete-message.php?id=XXX> ou XXX représente l'id du message que l'on souhaite détruire.



		← T →	sender	receiver	idMessage
<input type="checkbox"/>	Edit Delete		user	user	1
<input type="checkbox"/>	Edit Delete		user	user	2
<input type="checkbox"/>	Edit Delete		tata	user	5
<input type="checkbox"/>	Edit Delete		user	user	6
<input type="checkbox"/>	Edit Delete		user	user	7
<input type="checkbox"/>	Edit Delete		admin	admin	8
<input checked="" type="checkbox"/>	Edit Delete		toto	admin	10
<input type="checkbox"/>	Edit Delete		tata	toto	11
<input type="checkbox"/>	Edit Delete		tata	tata	12

Check All / Uncheck All With Selected: Edit Go

FIGURE 6: HORIZONTAL PRIVILEGE ESCALATION - AVANT



		← T →	sender	receiver	idMessage
<input type="checkbox"/>	Edit Delete		user	user	1
<input type="checkbox"/>	Edit Delete		user	user	2
<input type="checkbox"/>	Edit Delete		tata	user	5
<input type="checkbox"/>	Edit Delete		user	user	6
<input type="checkbox"/>	Edit Delete		user	user	7
<input type="checkbox"/>	Edit Delete		admin	admin	8
<input type="checkbox"/>	Edit Delete		tata	toto	11
<input type="checkbox"/>	Edit Delete		tata	tata	12

Check All / Uncheck All With Selected: Edit Go

FIGURE 7: HORIZONTAL PRIVILEGE ESCALATION – APRÈS

### 5.5.5. CSRF (SUPPRESSION D'UTILISATEUR)

L'administrateur peut être amené à effectuer une requête illégitime suite à une requête malveillante afin de supprimer des utilisateurs en passant par l'url vulnérable suivant : </validate-delete-user.php?id=USERNAME>. Cette ressource est accessible sans devoir passer par la méthode parente si l'administrateur est déjà authentifié sur l'application. On peut en voir le résultat sur les images suivante.

Username	Role	Active	Delete	Edit	Change Password
admin	Admin	Yes	Delete	Edit	Change
user	User	Yes	Delete	Edit	Change
tata	User	Yes	Delete	Edit	Change
toto	User	Yes	Delete	Edit	Change
d	User	Yes	Delete	Edit	Change
a	User	Yes	Delete	Edit	Change
Username	Role	Active	Delete	Edit	Change Password

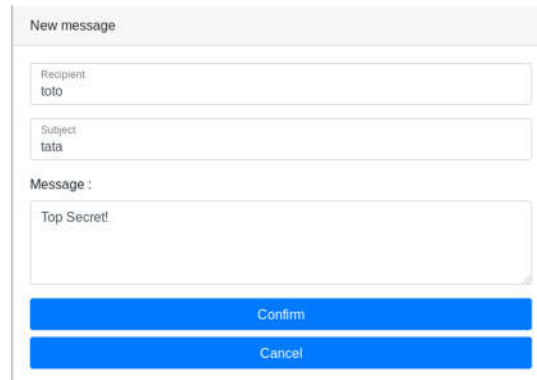
FIGURE 8: CSRF - AVANT

Username	Role	Active	Delete	Edit	Change Password
admin	Admin	Yes	Delete	Edit	Change
user	User	Yes	Delete	Edit	Change
tata	User	Yes	Delete	Edit	Change
toto	User	Yes	Delete	Edit	Change
d	User	Yes	Delete	Edit	Change
Username	Role	Active	Delete	Edit	Change Password

FIGURE 9: CSRF - APRÈS

### 5.5.6. SNIFFING (RÉCUPÉRATION DE DONNÉES CONFIDENTIELLE)

Il est possible de sniffer le contenu des messages qui sont envoyés à l'aide de Wireshark par exemple.



New message

Recipient  
toto

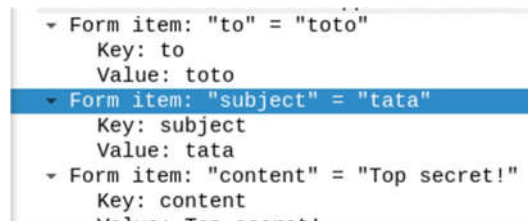
Subject  
tata

Message :  
Top Secret!

Confirm

Cancel

FIGURE 10: ENVOIE D'EMAIL CONFIDENTIEL



```
▼ Form item: "to" = "toto"
  Key: to
  Value: toto
▼ Form item: "subject" = "tata"
  Key: subject
  Value: tata
▼ Form item: "content" = "Top secret!"
  Key: content
  Value: Top secret!
```

FIGURE 11: LECTURE D'EMAIL CONFIDENTIEL



### 5.5.7. CODE INJECTION (DÉNI DE SERVICE PAR INJECTION HTML, REDIRECTION DE PAGE)

Il y a un moyen très simple de provoquer un déni de service, pour un utilisateur, il suffit d'injecter une redirection de page HTML dans un mail, de cette manière, lorsqu'un utilisateur se connecte il se fait automatiquement redirigé vers une page d'erreur 404.

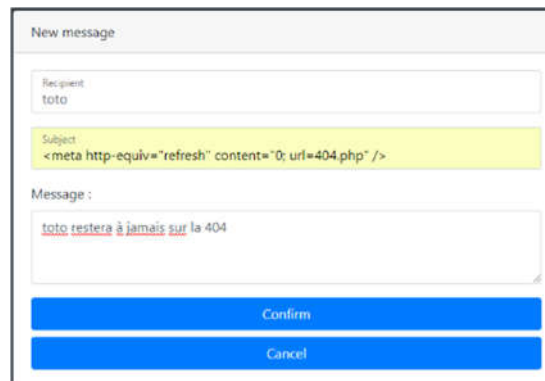


FIGURE 12: PAYLOAD DOS

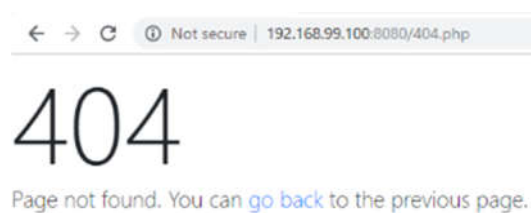


FIGURE 13: DEMONSTRATION DOS

## 6. IDENTIFIER LES CONTRE-MESURES

### 6.1. SANITARISATION DES ENTRÉES UTILISATEUR

Afin de nettoyer les entrées utilisateur, nous utilisons 3 méthodes prévues à cet effet, chacune d'entre elles va s'occuper d'un certain type de caractère que vous retrouverez ci-dessous.

trim: ''

stripslashes: \'

htmlspecialchars: '&', '"', "'", '<', '>'

Ces méthodes doivent être appelées avant d'utiliser les entrées utilisateur. Comme on peut le voir ci-dessous, les injections précédemment testées ne sont plus possibles.

Reception Date	Sender	Subject	Answer	Delete	Details
2019-01-12 05:31:02	user	','hacked','2020-06-10 10:10:10'); INSERT INTO messageSent (sender, receiver, idMessage) VALUES ('attacker', 'tata', '99');	Answer	Delete	Details
2019-01-12 05:30:08	user	<script>alert(1)</script>	Answer	Delete	Details
2019-01-12 05:29:45	user	<meta http-equiv="refresh" content="0; url=404.php" />	Answer	Delete	Details
2018-10-29 13:55:03	user	user	Answer	Delete	Details
Reception Date	Sender	Subject	Answer	Delete	Details

### 6.2. INJECTION SQL : PREPARED STATEMENT

Les prepared statements permettent de prévenir les injections SQL car les valeurs des paramètres sont correctement échappées. C'est à ce jour, l'une des meilleures façons d'écrire du code SQL tout en évitant les injections dans un projet PHP.

### 6.3. RÉPUDIATION

Nous avons corrigé une erreur de logique qui permettait à quiconque de supprimer les mails qui ne leur appartiennent pas. Pour se faire, nous avons ajouté une gestion des utilisateurs à la méthode de suppression des mails.

### 6.4. TOKEN ANTI CSRF

Les pages vulnérables (delete message et delete user) sont maintenant protégées par un token anti-csrf. Celui-ci est créé après la création de la session de l'utilisateur. Lorsqu'une demande de suppression d'un message ou utilisateur est faite, on vérifie que le bon token soit fourni pour autoriser l'action.

---

## 7. CONCLUSION

Cette deuxième partie du projet était très intéressante, grâce à elle, nous avons pu mettre en pratique ce que nous avons appris lors du cours et nous avons trouvé assez amusant la recherche des failles sur chaque page du site.

Finalement, nous avons corrigé la majorité des failles de l'application web, il reste cependant d'autre faille que l'on n'a pas traitée dans notre solution.

Parmi les problèmes non traités, on retrouve entre autres le fait qu'il est relativement facile, pour une personne mal intentionnée, de se connecter sur la page MySQLiteAdmin afin d'y faire ce que l'on souhaite. Il y a aussi les problèmes récurrents liés à l'architecture de notre application, comme le fait que notre site ne met pas en place une connexion sécurisée HTTPS/TLS/SSL, par conséquent toutes les données transitent en claire et il est possible de les voir avec un sniffer de réseau.

D'un autre côté, nous n'avons pas mis en place de politique de mots de passe, nous avons conclu que le fait de forcer un utilisateur à utiliser certains caractères, ainsi que le nombre de fois ou ceux-ci apparaissent donnait des indications supplémentaires aux attaquants et il serait mieux de sensibiliser les utilisateurs quant à la manière de créer un mot de passe solide.

Notre application ne met pas non plus de défense particulière face au brute force, nous avons pensé qu'il serait préférable d'analyser les logs de connexions afin de sonner une alerte si celui-ci était détecté.

Une autre méthode afin de sécuriser l'application, que nous n'avons pas exploré, serait de rendre le site accessible uniquement depuis l'intranet de l'entreprise ainsi les utilisateurs auraient l'obligation de passer par un VPN et tant que celui-ci sera protégé, nous pourrions réduire les risques liés à une attaque externe.

---

## 8. TABLE DES ILLUSTRATIONS

Figure 1: Récupération des credentials	11
Figure 2: Payload XSS	11
Figure 3: Demonstration XSS	11
Figure 4: Payload injection SQL	12
Figure 5: Demonstration injection SQL	12
Figure 6: Horizontal privilege escalation - avant	13
Figure 7: Horizontal privilege escalation – Après	13
Figure 8: CSRF - avant	14
Figure 9: CSRF - Après	14
Figure 10: Envoie d'email confidentiel	15
Figure 11: Lecture d'email confidentiel	15
Figure 12: Payload DoS	16
Figure 13: Demonstration DoS	16

## 9. ANNEXE :

Manuel d'utilisateur	STI-muaremi_schurch-PRO2_manuel.pdf
Présentation du projet :	STI-muaremi_schurch-PRO2_presentation.pptx
Site web sécurisé :	site.zip