FAILLES UNSECURE-APP

Validation des données :

- champ name : chaîne de caractères, obligatoire, 3 caractères min, 30 caractères max
- champ email : chaîne de caractères, obligatoire, doit être un email valide au bon format
- champ password : chaîne de caractère, obligatoire, min 8 caractères, max 255, doit contenir une majuscule, une minuscule, un caractère spécial, un chiffre
- champ message : chaîne de caractères, obligatoire, 250 caractères max, supprimé les espaces au début et à la fin de la chaîne de caractère
- Cryptage mot de passe md5 facilement décryptable, utilisé ***password_hash()*** et ***password_verify()*** pour la vérification des données à la connexion de l'utilisateur

Validation des données :

- Login.blade.php
  - Problème : Champ "username" - pas de minimum caractères requis et maximum - pas obligatoire

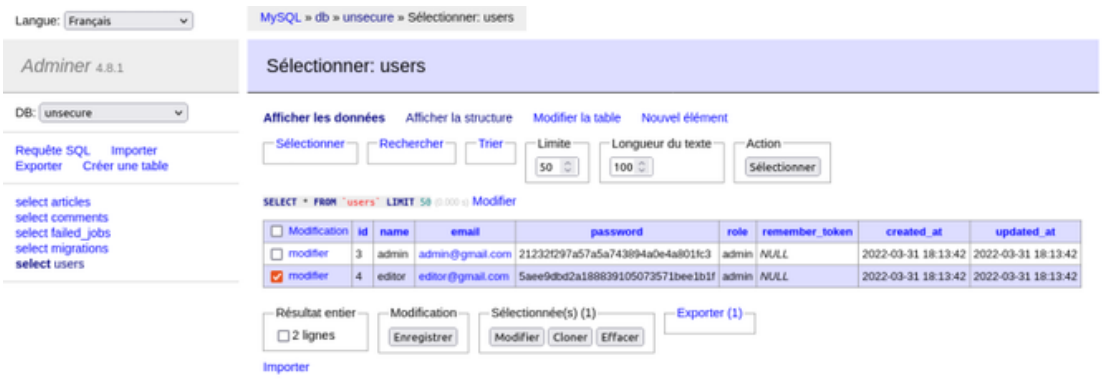
```
7     @error('login')
8         <div class="alert alert-danger">{{ $message }}</div>
9     @enderror
10    <div class="mb-3">
11      <label for="exampleFormControlInput1" class="form-label">Nom d'utilisateur</l
12      <input type="text" class="form-control" name="username" placeholder="toto" au
13    </div>
14    <div class="mb-3">
15      <label for="exampleFormControlInput1" class="form-label">Mot de passe</label>
16      <input type="text" class="form-control" name="password" placeholder="Mot de pa
17    </div>
18    <button type="submit" class="btn btn-primary">Se connecter</button>
19
20  </form>
21  @endsection
22
```

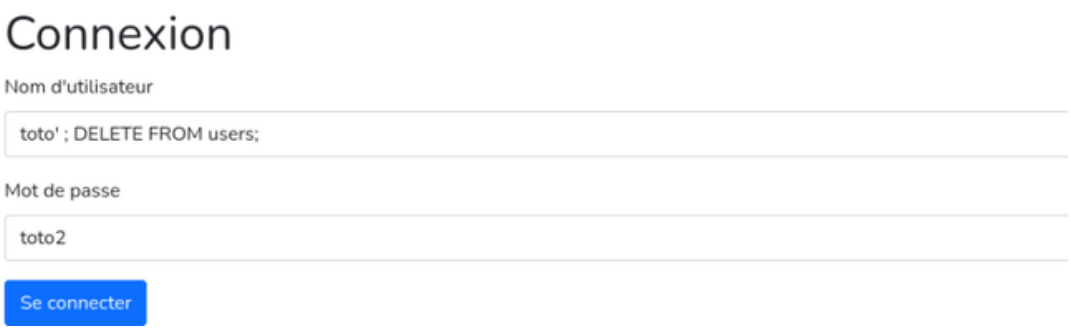
  - Solution :
    - Minimum et maximum caractère
      - min-lenght = 2 et max-lenght = 30
    - Caractères requis
      - attribut

CHEMLAL MORADE, AMALLAH SAMY, CHABERNAUD ANTOINE, HERNANDEZ SOMPARE ABRAHAM RICARDO

INJECTION SQL

Problème:





Solution:

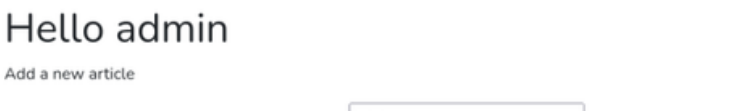
Injection SQL avec des requettes de type DELETE, INSERT, dans la base de donées

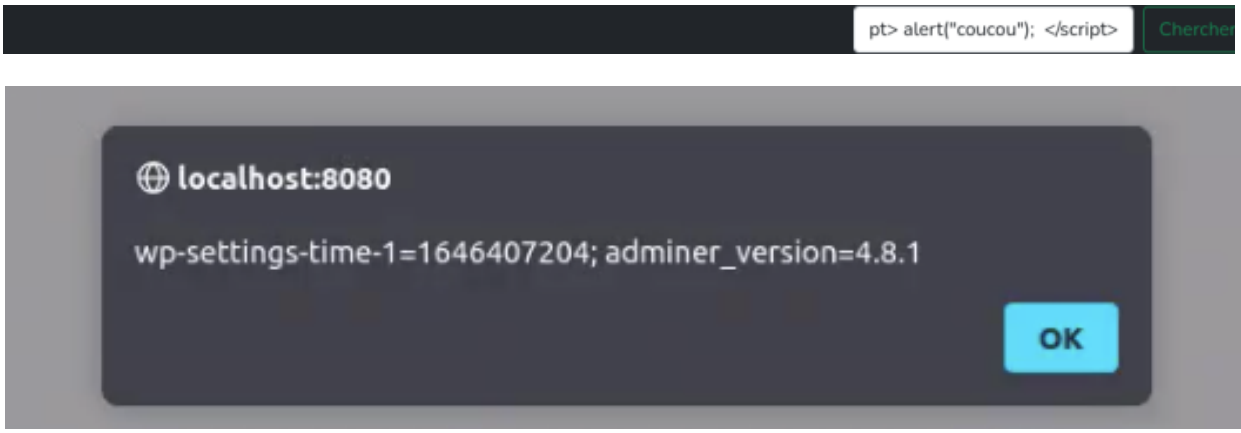
Failles XSS

Problème:

AJout de script par un **input title**

Champ recherche, formulaire ajout d'un article, formulaire ajout d'un **commentaire** : <script>alert(document.cookie) </script>





Solution:

nettoyer les données rentrées par l'utilisateur en remplaçant les **caractères spéciaux** par leur entité HTML
fonction: **htmlspecialchars()**

Bonus:

Correction: ajout de **htmlspecialchars()**

```php
public function search(Request $request)
{
  $mysql = new Mysql;

  $articles = $mysql->like('articles', '*', ['title' => $request->search]);

  if(!$articles) $articles = [];

  return view('search', [
    'articles' => $articles,
    'search' => htmlspecialchars($request->search)
  ]);
}
```

```php
  return view('admin.index', ['articles' => $articles]);
}

public function addArticle(Request $request)
{
    $article = new Article;
    $article->content = htmlspecialchars($request->content);
    $article->title = htmlspecialchars($request->title);
    $article->save();

    return redirect()->route('home');
}
```

Failles CRSF

Problème:

Formulaire d'ajout d'un article et d'un commentaire (si connecté) : absence d'un input hidden pour la gestion du **token** CSRF - risque exposition à une faille CSRF

```
<h1> Hello {{ \Auth::user()->name }}</h1>

<p> Add a new article </p>
<form method="POST" action="{{ route('admin.article.add') }}">
  <label>Title :</label>
  <input type="text" name="title" />
  <label>Content :</label>
  <textarea name="content"></textarea>
  <button type="submit">Save my new article</button>
</form>
```

```
<form method="POST" action="{{ route('article.add.comment') }}" >
  <p>Author name : </p><input type="text" name="author" />
  <p>Message : </p><textarea name="message"></textarea>
  <input type="hidden" name="article_id" value="{{ $article->id }}">
  <br>
  <button type="submit">Send my comment !</button>
</form>
```

Solution:

Solution : générer un token aléatoire qui va être stocké en session et mis en valeur d'un input hidden que l'on rajoute dans le formulaire correspondant.
Vérifier lors de la récupération des données si le **token stocké** en session correspond à celui du formulaire.

Bonus:

Surcouche gestion du token sur serveur en plus du code.
Suppression numéro de version Apache dans headers - inspecteur

```
ServerTokens Prod
ServerSignature Off
Header edit Set-Cookie ^(.*)$ $1;HttpOnly;Secure
```

Validation de données

Problème:

```
<h1> Connexion </h1>
<form method="GET" action="{{ route('login.validation') }}">
  @error('login')
      <div class="alert alert-danger">{{ $message }}</div>
  @enderror
  <div class="mb-3">
    <label for="exampleFormControlInput1" class="form-label">Nom d'utilisateur</label>
    <input type="text" class="form-control" name="username" placeholder="toto" autocomplete="off">
  </div>
  <div class="mb-3">
    <label for="exampleFormControlInput1" class="form-label">Mot de passe</label>
    <input type="text" class="form-control" name="password" placeholder="Mot de passe" autocomplete="off
  </div>
  <button type="submit" class="btn btn-primary">Se connecter</button>

</form>
```

Solution:

champ name : chaîne de caractères, obligatoire, 2 caractères min, 30 caractères max

- Formulaire : changer l'attribut type="text" pour le champ name en type="**name**"

champ password : chaîne de caractère, obligatoire, min 8 caractères, max
255, doit contenir une majuscule, une minuscule, un caractère spécial, un
chiffre

- Formulaire : changer l'attribut type="text" pour le champ password en type="password"

Bonus:

```
{
    \App\Models\Article::factory(15)->create();

    $user = new \App\Models\User;
    $user->name = 'admin';
    $user->email = 'admin@gmail.com';
    $user->password = password_hash('admin', PASSWORD_DEFAULT);
    $user->save();

    $user = new \App\Models\User;
    $user->name = 'editor';
    $user->email = 'editor@gmail.com';
    $user->password = password_hash('editor', PASSWORD_DEFAULT);
    $user->save();
}
```

Authentification

Problème:

```php
$user = new \App\Models\User;
$user->name = 'admin';
$user->email = 'admin@gmail.com';
$user->password = md5('admin');
$user->save();

$user = new \App\Models\User;
$user->name = 'editor';
$user->email = 'editor@gmail.com';
$user->password = md5('editor');
$user->save();
```

Solution:

Cryptage mot de passe md5 facilement décryptable

password_hash() avec PASSWORD_DEFAULT pour avoir bcrypt et **password_verify()** pour la vérification des données à la connexion de l'utilisateur

Bonus

```php
$user = $mysql->select('users', '*', ['name' => $request->username]);


if(isset($user[0]) && password_verify($request->password, $user[0]['password'])){
    \Auth::loginUsingId($user[0]['id']);
    return redirect()->route('admin.index')->withCookie(cookie('id', $user[0]['id'], 3600000));
}
```

Correction: ajout de password_verify()

```php
$user = new \App\Models\User;
$user->name = 'admin';
$user->email = 'admin@gmail.com';
$user->password = password_hash('admin', PASSWORD_DEFAULT);
$user->save();

$user = new \App\Models\User;
$user->name = 'editor';
$user->email = 'editor@gmail.com';
$user->password = password_hash('editor', PASSWORD_DEFAULT);
$user->save();
```

Correction: remplacement de md5 par **password_hash()** et ajouté PASSWORD_DEFAULT

Problème:

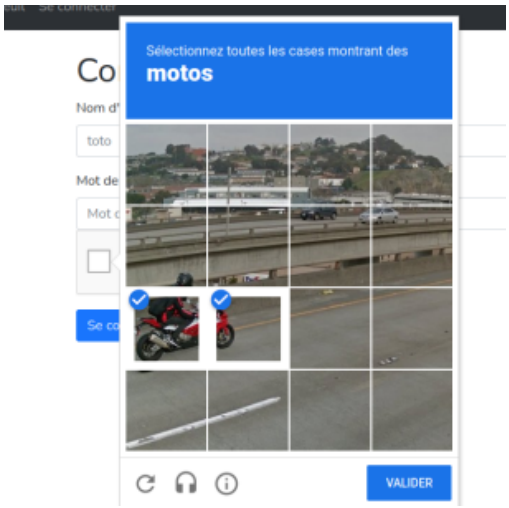
Problème robots qui font du **scraping**

Solution:

Solution: Intégration d'un **reCAPTCHA** pour éviter que des robots malveillants crée automatiquement des données dans l'application

Bonus

Correction: ajout d'un **reCAPTCHA**




```
<div class="mb-3">
  <label for="exampleFormControlInput1" class="form-label">Mot de passe</label>
  <input type="text" class="form-control" name="password" placeholder="Mot de passe" autocomplete="off">
  <?php echo '<div class="g-recaptcha" data-sitekey="'.$_ENV['V2_SECET'].'"></div>'; ?>
</div>
<button type="submit" class="btn btn-primary">Se connecter</button>
```