

Protection CSRF

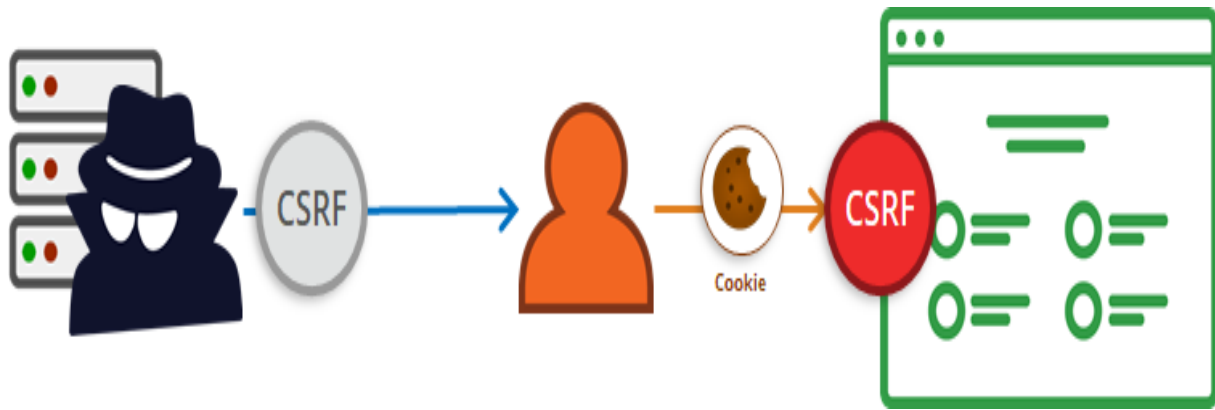
Sécuriser les formulaires avec Laravel

1. Objectifs

- Connaître les méthodes de sécurisation d'un formulaire avec Laravel.
- Être capable de sécuriser un formulaire laravel.

2. Présentation

- Un formulaire est un vecteur d'attaque potentiel pour une personne malveillante.
- Laravel propose par défaut un mécanisme de défense face aux attaques de type **CSRF** (Cross-Site Request Forgery).
- Dans ce formulaire on va voir comment protéger très facilement une application Laravel contre les attaques CSRF (Cross Site Request Forgery)
- La forme complète de CSRF est la falsification de requêtes intersites. Il s'agit d'un type d'attaque en ligne dans lequel l'attaquant envoie des demandes en tant qu'utilisateur autorisé à un système en obtenant des informations d'accès d'un utilisateur particulier de ce système et effectue différents types d'activités malveillantes en utilisant l'identité de cet utilisateur. L'impact de cette attaque dépend des privilèges de la victime sur le système.
- Si la victime est un utilisateur normal, cela n'affectera que les données personnelles de la victime. Mais si la victime est l'administrateur du système, l'attaquant peut endommager l'ensemble du système.



3. Fonctionnement du CSRF

- La falsification de requêtes intersites (CSRF) est un type d'attaque effectué par l'attaquant pour envoyer des requêtes à un système avec l'aide d'un utilisateur autorisé auquel le système fait confiance.
- Laravel fournit une protection contre les attaques CSRF en générant un jeton **CSRF**. Ce jeton CSRF est généré automatiquement pour chaque utilisateur. Ce jeton n'est rien d'autre qu'une chaîne aléatoire gérée par l'application Laravel pour vérifier les demandes des utilisateurs.
- Cette protection de jeton CSRF peut être appliquée à n'importe quel formulaire HTML dans l'application Laravel en spécifiant un champ de formulaire caché du jeton CSRF.
- Une des manières efficaces de se prémunir contre les attaques **CSRF** est d'accompagner chaque formulaire à protéger d'un jeton (en anglais "token") d'identification de session.
- Pour ce faire, lorsqu'un visiteur se connecte sur votre site, vous enregistrez en session une chaîne de caractère aléatoire.
- Cette même chaîne est récupérée en session et passée au formulaire, sous la forme d'un champ de type "**hidden**".
- Lorsque le formulaire est posté, votre système de validation doit vérifier que le champ caché contient la même valeur que celle enregistrée en session.

- Si ça n'est pas le cas, ça signifie que quelqu'un a essayé de soumettre le formulaire depuis un autre endroit que votre site. Il faut alors annuler le traitement de ce formulaire.

Laravel inclut un **plug-in CSRF** intégré, qui génère des jetons pour chaque session utilisateur active. Ces jetons vérifient que les opérations ou requêtes sont envoyées par l'utilisateur authentifié concerné.

4. La mise en œuvre

- CSRF est implémenté dans les formulaires HTML déclarés dans les applications Web. Vous devez inclure un jeton CSRF valide masqué dans le formulaire, afin que le middleware de protection CSRF de Laravel puisse valider la demande.
- La protection **CSRF** peut être implémentée dans Laravel à l'aide de n'importe quel formulaire HTML avec une forme cachée de jeton CSRF et la demande de l'utilisateur est validée à l'aide du middleware CSRF **VerifyCsrfToken**.
- L'une des options suivantes peut être utilisée pour générer un jeton **CSRF**.

1. @csrf

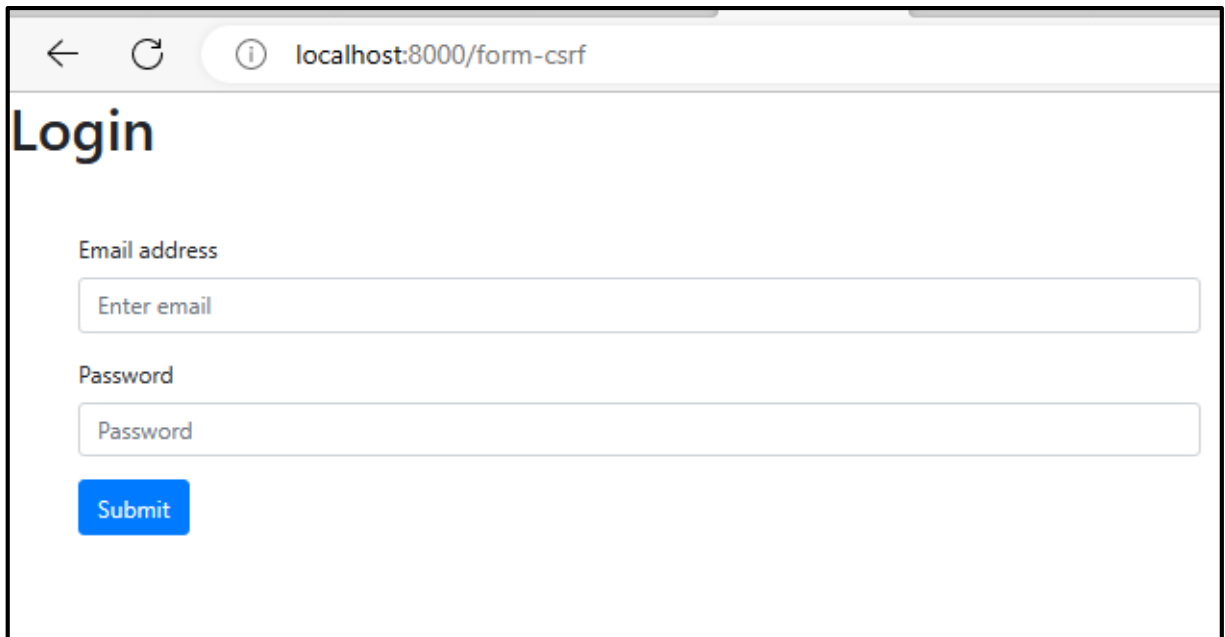
- C'est une directive de Blade pour générer un champ de jeton qui sera utilisé pour la vérification. Il génère un champ de saisie masqué.
- Lorsque la directive blade **@csrf** est utilisée, une balise *input* de type='hidden' est ajoutée au formulaire.
- La valeur sera le jeton CSRF qui sera envoyé dans le cadre des données du formulaire lorsque le formulaire est soumis.

▪ Activité

- Créez un fichier de vue Laravel nommé **form-csrf.blade.php** avec le code HTML suivant où la directive @csrf est utilisée pour générer le jeton CSRF.

```
<form action="{{route('details')}}" method="POST" >
  @csrf
  <label>Email address</label>
  <input name="email" type="email">
  <label>Password</label>
  <input type="password" name="password" >
```

```
<button type="submit">Submit</button>
</form>
```



The screenshot shows a web browser window with the address bar displaying 'localhost:8000/form-csrf'. The page has a title 'Login'. Below the title, there are two input fields: 'Email address' with a placeholder 'Enter email' and 'Password' with a placeholder 'Password'. A blue 'Submit' button is located below the password field.

- Dans le fichier **web.php** pour charger le fichier de vue dans le navigateur. Lorsque l'utilisateur donnera **form-csrf** après l'URL de base (http://127.0.0.1:8000/form-csrf), il recherchera le fichier **form-csrf.blade.php** dans le dossier de vue du projet Laravel.

```
Route::get('/form-csrf', function () {return
view('form-csrf');});
```

- Si vous inspectez la page après l'exécution, vous obtiendrez la sortie comme ci-dessous. Ici, un champ masqué avec la valeur est généré automatiquement par la directive **@csrf**.

```

▼ <div class="container mt-5">
  ▼ <form action>
    <input type="hidden" name="_token" value="3bv3Jkk1BnsafdLNyxUe0gXCreX99ztI120f5QNB"> == $0
    ▶ <div class="row">...</div>
    ▶ <div class="row">...</div>
    <input type="submit" name="send" value="Submit" class="btn btn-primary btn-block">
  </form>
</div>

```

2. csrf_token ()

- Cette fonction peut être utilisée dans la balise **meta** et le champ de saisie masqué du formulaire HTML. Il génère une chaîne aléatoire en tant que jeton CSRF.
- Mettez à la place de **@csrf** dans la vue Laravel nommé **form-csrf.blade.php** la fonction `csrf_token ()` est utilisée pour générer le jeton CSRF. Cette fonction est utilisée comme valeur de l'attribut `value` du champ masqué et est utilisée avec deux accolades.

▪ Syntaxe

```

<form method = "POST">
  <input type = "hidden" name = "_ token" value =
  "{{csrf_token ()}}">
  .....
  .....
</form>

```

- Si vous inspectez la page après l'exécution, vous obtiendrez la même sortie de l'exemple précédent.
 - Veuillez noter que le bouton d'envoi inclut des fonctionnalités dans la section fonction de rappel. Il est montré ci-dessous:

```

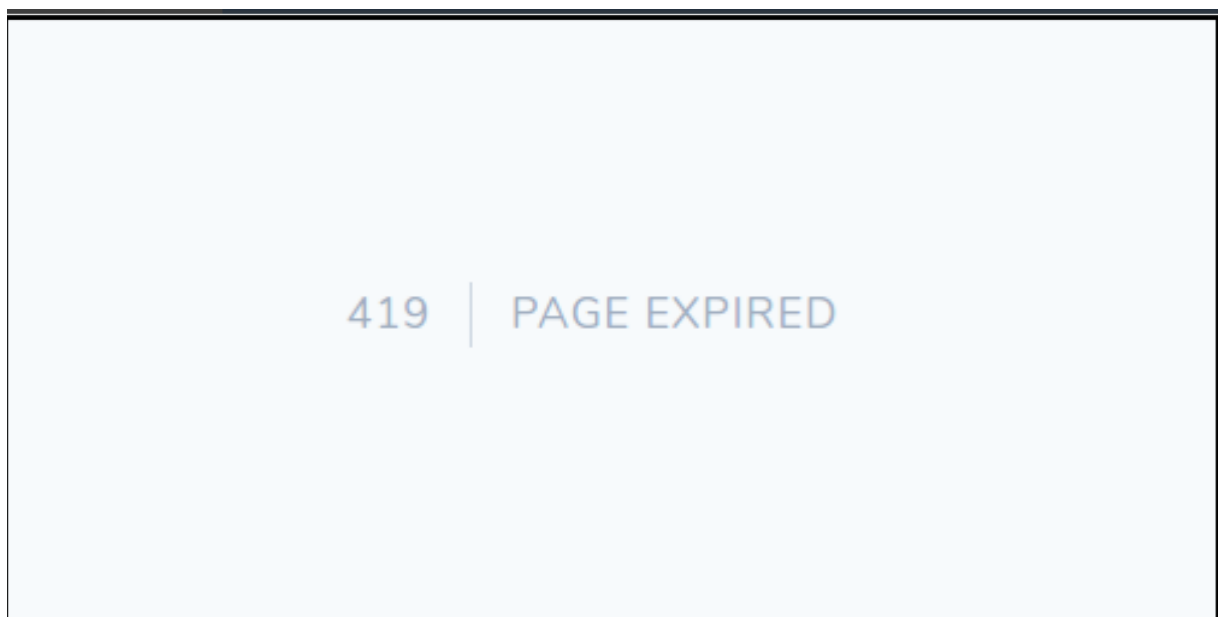
Route::post('/details', function (Request $req) {
    return $req->all();
})->name('details');

```

- La sortie obtenue renverra JSON avec un jeton comme indiqué ci-dessous:

```
localhost:8000/details
{"_token":"hfliQ8jp09Hd77hhw9rzPYCB73xu2W8AQsb1s0kV","email":null,"password":null}
```

Si le jeton CSRF n'est pas présent dans la demande de formulaire envoyée ou s'il semble invalide, Laravel envoie un message d'erreur "Page Expired" avec un code d'état 419.



3. Exclure les URI de la protection CSRF

Parfois, vous souhaitez peut-être exclure un ensemble d'URI de la protection CSRF. En règle générale, vous devez placer ces types de routes en dehors du groupe web de middleware **App\Providers\RouteServiceProvider** qui s'applique à toutes les routes du fichier **routes/web.php**. Cependant, vous pouvez également exclure les routes en ajoutant leurs URI à la propriété **\$except** du middleware **VerifyCsrfToken** :

```
<?php
namespace App\Http\Middleware;
```

```
use Illuminate\Foundation\Http\Middleware\VerifyCsrfToken as
Middleware;

class VerifyCsrfToken extends Middleware
{
    /**
     * The URIs that should be excluded from CSRF
    verification.
     *
     * @var array<int, string>
     */
    protected $except = [
        '/details'
    ];
}
```

Maintenant essayez de poster à nouveau le formulaire en supprimant [@csrf](#).