CSRF-ЗАЩИТА

ВВЕДЕНИЕ

Laravel позволяет легко защитить ваше приложение от атак с подделкой межсайтовых запросов (CSRF). Подделка межсайтовых запросов — тип атаки на сайты, при котором несанкционированные команды выполняются от имени аутентифицированного пользователя.

Laravel автоматически генерирует CSRF-токен для каждой активной пользовательской сессии в приложении. Этот токен используется для проверки того, что именно авторизованный пользователь делает запрос в приложение.

При определении каждой HTML-формы вы должны включать в неё скрытое поле CSRF-токена, чтобы посредник CSRF-защиты мог проверить запрос. Вы можете использовать хелпер `csrf_field` для генерирования поля токена:

```
<form method="POST" action="/profile">
    {{ csrf_field() }}
    ...
</form>
```

Посредник `VerifyCsrfToken`, входящий в группу посредников `web`, автоматически проверяет совпадение токена в данных запроса с токеном, хранящимся в сессии.

CSRF токены и JavaScript

При создании приложений, работающих на JavaScript, удобно, чтобы собственная JavaScript HTTP-библиотека автоматически прикрепляла CSRF-токен к каждому исходящему запросу. По

умолчанию файл resources/assets/js/bootstrap.js регистрирует значение мета-тега csrf-token в HTTP-библиотеке Axios. Если вы не используете эту библиотеку, то вам нужно будет вручную настраивать это поведение для своего приложения.

ИСКЛЮЧЕНИЕ URI ИЗ CSRF-ЗАЩИТЫ

Иногда бывает необходимо исключить набор URI из-под CSRF-защиты. Например, если вы используете Stripe для обработки платежей и применяете их систему веб-хуков, то вам надо исключить роут вашего обработчика веб-хуков Stripe из-под CSRF-защиты, так как Stripe не будет знать, какой CSRF-токен надо послать в ваш роут.

Обычно такие роуты помещаются вне группы посредников web, которую RouteServiceProvider применяет ко всем роутам в файле routes/web.php. Но вы также можете исключить роуты, добавив их URI в свойство $except посредника VerifyCsrfToken:

```
<?php

namespace App\Http\Middleware;

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

class VerifyCsrfToken extends BaseVerifier
{
    /**
     * URI, которые надо исключить из CSRF-проверки.
     *
```

```
     * @var array
     */
    protected $except = [
        'stripe/*',
    ];
}
```

X-CSRF-TOKEN

Помимо проверки CSRF-токена как POST-параметра, посредник VerifyCsrfToken будет также проверять заголовок запроса X-CSRF-TOKEN. Например, вы можете хранить токен в HTML-теге meta:

```
<meta name="csrf-token" content="{{ csrf_token() }}">
```

После создания тега meta вы можете указать библиотеке, такой как jQuery, автоматически добавлять токен в заголовки всех запросов. Это обеспечивает простую, удобную CSRF-защиту для ваших приложений на базе AJAX:

```
$.ajaxSetup({
    headers: {
        'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
    }
});
```

{tip} По умолчанию файл resources/assets/js/bootstrap.js регистрирует значение мета-тега csrf-token в HTTP-библиотеке Axios. Если вы не используете эту библиотеку, то вам нужно будет вручную настраивать это поведение для своего приложения.

X-XSRF-TOKEN

Laravel хранит текущий CSRF-токен в cookie `XSRF-TOKEN`, которую включается в каждый отклик, генерируемый фреймворком. Вы можете использовать значение cookie, чтобы задать заголовок запроса `X-XSRF-TOKEN`.

Этот cookie в основном посылается для удобства, потому что некоторые JavaScript-фреймворки, такие как Angular, автоматически помещают его значение в заголовок `X-XSRF-TOKEN`.