Laravel - CSRF Protection

Topics : Laravel
Written on December 25, 2023

Cross-Site Request Forgery (CSRF) is a security vulnerability that allows an attacker to trick a user's browser into making unintended requests on a web application. Laravel includes built-in CSRF protection to guard against this type of attack.

1. Understanding CSRF Protection:

CSRF protection involves generating and validating tokens to ensure that requests made to your application are legitimate and not forged by malicious users.

2. CSRF Token in Forms:

Laravel automatically generates a CSRF token for each active user session. You can include this token in your HTML forms using the `@csrf` Blade directive.

```
<form method="POST" action="/example">
    @csrf
    <!-- Your form fields go here -->
    <button type="submit">Submit</button>
</form>
```

3. CSRF Token in AJAX Requests:

When making AJAX requests, you can include the CSRF token in the request headers. Laravel includes a `csrf_token` cookie that you can read and include in your AJAX requests.

```
// Example using Axios
axios.defaults.headers.common['X-CSRF-TOKEN'] =
document.head.querySelector('meta[name="csrf-token"]').content;

axios.post('/example', {
    // Your data here
})
    .then(response => {
        // Handle the response
    })
    .catch(error => {
        // Handle errors
```

```
    });
```

4. CSRF Middleware:

Laravel includes a middleware called `VerifyCsrfToken` that automatically checks the CSRF token for all incoming POST, PUT, PATCH, and DELETE requests. This middleware is included by default in the `web` middleware group.

You can find the middleware in the `app/Http/Middleware/VerifyCsrfToken.php` file.

5. Excluding URIs from CSRF Protection:

If you need to exclude specific URIs from CSRF protection, you can add them to the `$except` property in the `VerifyCsrfToken` middleware.

```
protected $except = [
    'example/*', // Exclude all routes starting with 'example/'
];
```

6. CSRF Token Verification in Controllers:

Laravel automatically verifies the CSRF token for you in incoming requests. If the token is not present or not valid, Laravel will throw a `TokenMismatchException`. You can catch this exception and handle it as needed.

```
use Illuminate\Http\Request;

public function example(Request $request)
{
    // Laravel automatically verifies the CSRF token

    // Your controller logic here
}
```

7. Configuring the CSRF Token Name and Cookie Name:

You can customize the CSRF token name and cookie name in the `config/session.php` file.

```
'session' => 'your_session_name',
'token' => 'your_token_name',
```

8. CSRF Protection in APIs:

If you are building an API and don't need CSRF protection, you can remove the `VerifyCsrfToken` middleware from the `web` middleware group and use a different middleware group for API routes.

```
// app/Http/Kernel.php

protected $middlewareGroups = [
```

```
    'web' => [
        // ...
        // Comment out or remove VerifyCsrfToken middleware
        // \App\Http\Middleware\VerifyCsrfToken::class,
    ],

    'api' => [
        'throttle:60,1',
        'bindings',
    ],
];
```

9. CSRF Protection in Stateless APIs:

If you are building a stateless API and don't want to use CSRF tokens, you can create a middleware to disable CSRF protection for specific routes.

```
// Example middleware to disable CSRF protection
public function handle($request, Closure $next)
{
    config(['session.driver' => 'array']);

    return $next($request);
}
```

10. Testing CSRF Protection:

To test CSRF protection in your application, you can create tests using Laravel's testing features, such as the `actingAs` method to simulate a logged-in user.

```
public function testExample()
{
    $response = $this->actingAs($user)
                     ->post('/example', ['name' => 'John Doe']);

    $response->assertStatus(200);
}
```