

Cross-Site Scripting XSS

XSS Cross-Site Scripting (o "Inyección de scripts entre sitios", en español). Es un tipo de vulnerabilidad de seguridad en aplicaciones web, donde un atacante puede insertar código malicioso (como JavaScript) en una página web, que luego se ejecutará en el navegador de un usuario que visite esa página.

Los ataques XSS ocurren cuando una aplicación web no valida correctamente las entradas de los usuarios, permitiendo que un atacante inserte código malicioso en una página web que se servirá a otros usuarios. El código malicioso puede ser diseñado para robar información personal, redirigir a los usuarios a sitios web maliciosos, mostrar anuncios no deseados, o incluso para tomar control del navegador del usuario.

Existen dos tipos principales de ataques XSS:

Reflejado: El ataque XSS reflejado ocurre cuando el código malicioso se ejecuta en la página web después de que un usuario hace una solicitud a la aplicación web. El código malicioso se "refleja" de vuelta al usuario a través de la respuesta de la aplicación web.

Almacenado: El ataque XSS almacenado ocurre cuando el código malicioso se almacena en la base de datos de la aplicación web y se ejecuta cada vez que se solicita la página web afectada.

Los desarrolladores pueden prevenir ataques XSS mediante la validación y filtrado de las entradas de los usuarios, utilizando bibliotecas de seguridad como Content Security Policy (CSP) y asegurándose de que todas las entradas de los usuarios se escapen de forma adecuada antes de ser utilizadas en una página web.

Sanitización de XSS en PHP

Aquí hay un ejemplo de cómo mitigar el riesgo de XSS en PHP utilizando la función `htmlspecialchars()` para escapar las salidas de usuario:

```
<?php
$params = "<a href='test'>Test</a>";
$valid = htmlspecialchars($params, ENT_QUOTES, 'UTF-8');

echo $valid // <a href='test'>Test</a>
```

?>

En este ejemplo, estamos utilizando la función `htmlspecialchars()` para escapar los caracteres especiales HTML en la entrada del usuario. La función toma tres argumentos: la cadena de entrada a escapar, la opción `ENT_QUOTES` para escapar tanto comillas dobles como comillas simples, y la codificación de caracteres 'UTF-8'.

Sanitización XSS en Laravel

Este método fue probado para Laravel 7, y consiste en crear un middleware para sanitizar las entradas.

El middleware proporciona un mecanismo conveniente para inspeccionar y filtrar las solicitudes HTTP que ingresan a su aplicación.

Para crear un middleware se debe aplicar el siguiente comando en la raíz del proyecto de Laravel:

```
php artisan make:middleware XssSanitizer
```

Editar el archivo `app/Http/Middleware/XssSanitizer.php` para que quede de la siguiente manera:

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

use Closure;
use Illuminate\Http\Request;

class XssSanitizer
{
    /**
     * Handle an incoming request.
     *
     * @param \Illuminate\Http\Request $request
     * @param \Closure $next
     * @return mixed
     */
    public function handle(Request $request, Closure $next)
    {
        $input = $request->all();
```

```

        array_walk_recursive($input, function(&$input) {
            $input = strip_tags($input);
        });
        $request->merge($input);
        return $next($request);
    }
}

```

Ahora es necesario agregar la ruta de XssSanitizer.php al vector \$routeMiddleware ubicado en app/Http/Kernel.php:

```

protected $routeMiddleware = [

    'auth' => \App\Http\Middleware\Authenticate::class,

    ....

    'XssSanitizer' => \App\Http\Middleware\XssSanitizer::class,
];

```

Una vez realizado esto, ya se puede utilizar XssSanitizer middleware en las rutas para realizar la sanitización:

```

Route::group(['middleware' => ['XssSanitizer']], function () {

    Route::get('/', function () {
        return view('welcome');
    });

    Route::get('/formulario',function () {
        return view('formulario');
    });

    Route::get('form-get', function (Illuminate\Http\Request $request)
    {
        return $request->input('buscar');
    }->name('form-get');

});

```

Revision #5

Created 28 febrero 2023 15:05:45 by Vladimir Urquiola

Updated 14 abril 2023 14:45:30 by Vladimir Urquiola