Qué es y cómo funciona un ataque Cross-Site Scripting

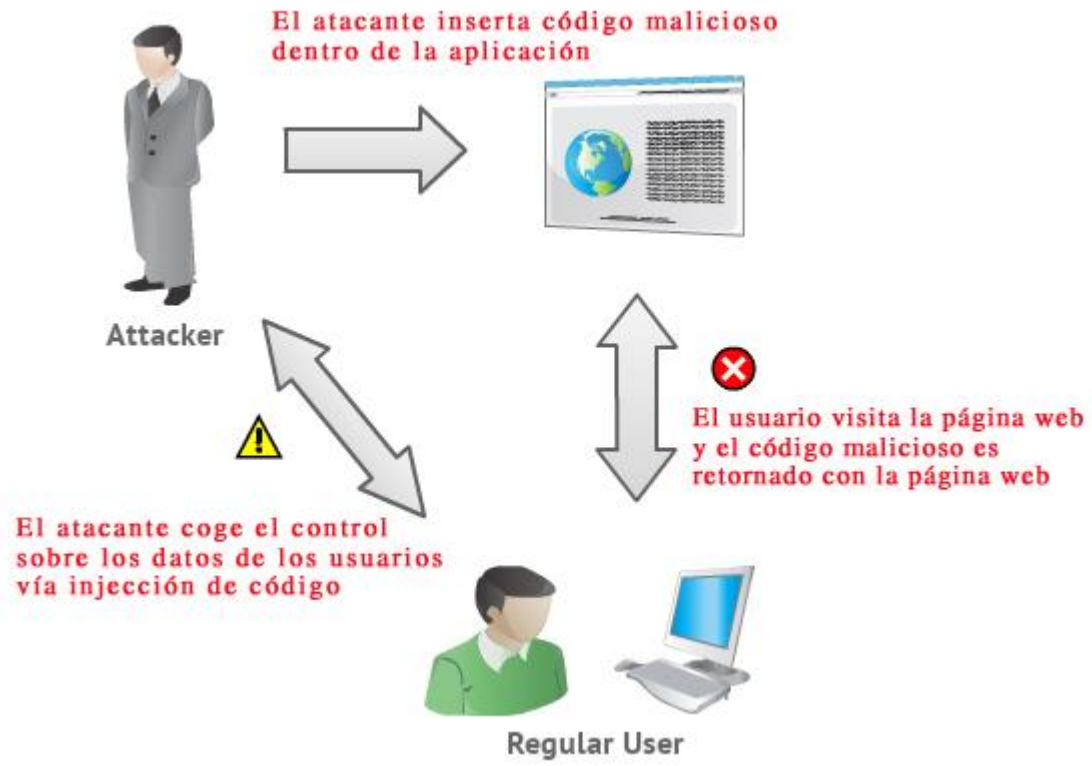
Cardenal Gardoki, 1
48008 BILBAO (Vizcaya)
Teléfono: 902 012 199
www.hostalia.com

Todo sitio web lleva detrás de sí miles de líneas de código que son las encargadas de crear tanto la parte visual de la aplicación como toda la funcionalidad que ofrecen los portales a los clientes.

A la hora de programar todas esas líneas de código, es fundamental asegurarse de que **no dejamos ninguna puerta abierta por la que podamos sufrir algún tipo de ataque**, problema que puede provocar desde cambiar la información de la web hasta el envío continuo de mensajes de spam e incluso, en el peor de los casos, obtener datos críticos que puedan poner en peligro la información de nuestra empresa o nuestros clientes.

En la actualidad nos podemos encontrar con varios tipos de vulnerabilidades relacionadas con la programación utilizada en la web, pero en este documento nos centraremos en el denominado "Cross site scripting", también conocido por las siglas XSS.

Qué es Cross Site Scripting



El XSS se trata de una vulnerabilidad que muchos desarrolladores web dejan pasar, bien por falta de planificación o por desconocimiento. Esta vulnerabilidad suele aparecer por la falta de mecanismos en el filtrado de los campos de entrada que dispone la web, permitiendo el envío de datos e incluso la ejecución de scripts completos.

El código malicioso utilizado en este tipo de ataques está compuesto por cadena de datos: **scripts completos contenidos en enlaces o ejecutados desde formularios vulnerables**.

Por ejemplo, si un usuario escribiera en una caja de texto el siguiente script, donde no se controle la entrada de datos estaríamos consiguiendo que apareciera una ventana con un mensaje de bienvenida.

```
<script>alert("hola");</script>
```

Clasificación de los ataques Cross Site Scripting

Los diversos tipos de ataques que nos podemos encontrar de Cross Site Scripting pueden ser catalogados en dos grandes grupos: XSS persistente o directo, y XSS reflejado o indirecto.

1.- XSS persistente o directo

Este tipo de ataque consiste en embeber código HTML peligroso en sitios que lo permitan por medio de etiquetas <script> o <iframe>. Es la más grave de todas ya que el código se queda implantado en la web de manera interna y es ejecutado al abrir la aplicación web.

Dentro de este tipo nos encontramos el subtipo "Local" que aparece por un mal uso del DOM (Modelo de objetos del documento) con JavaScript, que permite la apertura de nuevas páginas con código malicioso JavaScript incrustado, afectando el código de la primera página en el sistema local. Estos códigos son ejecutados del lado del cliente, por lo que los filtros utilizados en el servidor no funcionan para este tipo de vulnerabilidades.

2.- XSS reflejado o indirecto

Es el tipo de ataque XSS más habitual y consiste en editar los valores que se pasan mediante URL, cambiando el tipo de dato pasado del usuario a la web, haciendo que ese código insertado se ejecute en dicho sitio.

Métodos de inyección de código utilizado en los ataques

A la hora de lanzar un ataque de este tipo, los atacantes pueden utilizar varios tipos de inyección de código distinto. Veamos a continuación cuáles son los más utilizados.

1.- Inyección en un formulario

Se trata del ataque más sencillo. Consiste en inyectar código en un formulario que después al enviarlo al servidor, será incluido en el código fuente de alguna página. Una vez insertado en el código fuente, cada vez que se cargue la página se ejecutará el código insertado en ella.

2.- Inyección por medio de elementos

En este tipo de sistema de inyección de código se utiliza cualquier elemento que viaje entre el navegador y la aplicación, como pueden ser los atributos usados en las etiquetas HTML utilizadas en el diseño de la página.

3.- Inyección por medio de recursos

Aparte de los elementos en la url y los formularios, hay otras formas en la que se puede actuar como son las cabeceras HTTP. Estas cabeceras son mensajes con los que se comunican el navegador y el servidor. Aquí entran en juego las cookies, las sesiones, campo referer…

Consejos para evitar los ataques

Por suerte, prevenir estos ataques es sencillo, y una vez que aprendamos cómo hacerlo es algo que siempre deberíamos tener presente para evitarlos.

A continuación veremos algunos consejos prácticos que podemos llevar a cabo para proteger nuestras aplicaciones web mediante la validación de entrada de datos, asegurándonos que cada uno de los campos que utilicemos no contengan código que pueda afectar nuestra aplicación o a nuestros usuarios.

1.- Limitar los caracteres de entrada

Lo primero que debemos hacer es limitar los caracteres que un usuario puede introducir en los campos de texto. Por ejemplo, si tenemos un campo para introducir el nombre del usuario, no vamos a dejarlo abierto para que se puedan introducir un número indefinido de caracteres, sino que lo vamos a limitar por ejemplo a 20 o 30 caracteres. Para limitar el número de caracteres, podemos utilizar la variable “maxlength” que nos proporciona el estándar HTML.

2.- Sanear los datos

Cuando hablamos de sanear los datos, nos estamos refiriendo a quedarnos únicamente con la información que nos interesa, eliminando las etiquetas HTML que pueden ser incluidas en una caja de texto. Por ejemplo, si estamos almacenando el nombre de una persona, de poco nos sirve que el usuario lo introduzca en negrita, ya que lo único que nos interesa es su nombre.

Para lograr esta limpieza, podemos utilizar la función “strip_tags”. Por ejemplo:

```
$comentario = strip_tags($_POST['comentario']);
```

3.- Escapar los datos

Para proteger los datos y mostrarlos tal y como el usuario los introdujo, deberíamos “escapar” los datos al presentarlos al usuario. Es decir, caracteres que deben ser representados por entidades HTML si se desea preservar su significado (por ejemplo las comillas dobles hay que transformarlas en " que es como se representa en HTML). Con esto evitamos que el navegador lo ejecute y evalúe el código.

Para lograr esto, podemos utilizar la función “htmlspecialchars”. Por ejemplo:

```
echo "Mostrando resultados de: ".htmlspecialchars($_GET['busqueda']);
```

Pero aparte de tomar medidas de seguridad para impedir ataques XSS en nuestros sitios, debemos evitar almacenar información relevante de nuestros visitantes en las Cookies, y además encriptar las sesiones de usuario para que en caso de que las capturen no puedan acceder a la información almacenada en ella.