CROSS SITE SCRIPTING (XSS)

Cross-Site Scripting (XSS)

XSS ocurre cuando un atacante es capaz de inyectar un script, normalmente Javascript, en un formulario web principalmente por validar incorrectamente datos de usuario.

Para el navegador, el script es parte del código de la web por lo que se lo ejecuta como fiable.

Ejemplo de ataque XSS

Supongamos por ejemplo el siguiente formulario:

```
<form action="post.php" method="post">
    <input type="text" name="comentario" value="">
    <input type="submit" name="submit" value="Submit">
<form>
```

Supongamos que en post.php mostrará los datos:

```
echo $_POST['comentario'];
```

Si no se filtran los datos del formulario, el atacante puede enviar un script a través del formulario como el siguiente:

```
<script>alert("Hola")</script>
```

Por lo que se podría ejecutar código que dañara la web.

Prevenir ataques XSS

Cualquier dato debe ser validado o escapado antes de mostrarlos.

Las medidas a tomar se pueden dividir en tres: **data validation, data sanitization y output escaping.**

Data validation

Cada dato debe ser validado cuando se recibe para asegurarse que es del tipo correcto, y rechazado si no pasa ese proceso de validación.

Data sanitization

La sanitización de datos manipula los datos para asegurarse que son seguros, eliminando cualquier etiqueta html, xml o php.
La función strip_tags () elimina una cadena de las etiquetas HTML, XML y PHP.

```
// Sanitizar comentario de usuario
$comentario = strip_tags($_POST["comentario"]);
```

Output escaping

Para proteger la integridad de los datos que se devuelven, el output data, se debe escapar cualquier dato que se devuelve al usuario. Esto evita que el navegador malinterprete alguna secuencia especial de caracteres. La función htmlentities () convierte los caracteres en entidades HTML.

Por ejemplo:

```
$cadena = '<a href="https://www.google.com">Google</a>';
echo htmlentities($cadena);
```

La salida HTML será en la vista código:

```
&lt;a href=&quot;https://www.google.com&quot;&gt;Google&lt;/a&gt;
```

La salida en el navegador será:

```
<a href="https://www.google.com">Google</a>
```

Ejemplo de prevención contra ataques XSS

```
<?php
	// Validar el comentario
	$comentario = trim($_POST["comentario"]);
	if(empty($comentario)){
		exit("Debes introducir un comentario");
	}

	// Sanitizar comentario
	echo strip_tags($comentario);
	echo "<br>";

	// Escapamos la salida
	echo htmlentities($comentario);
	echo "<br>";

	// Sin Escapar o Sanitizar
	echo $comentario;
?>
```

Nos aseguramos de que no se guardan comentarios vacíos.
Se sanitizan los datos con strip_tags() eliminando cualquier posible etiqueta HTML que pudiera contener.
La función _striptags hace que no sea posible insertar enlaces en los comentarios, ya que éstos utilizan una etiqueta que será eliminada. Para que puedan insertarse se puede utilizar htmlentities en su lugar.
Por último ejecutamos el ataque XSS al no Sanitizar ni Escapar entrada del usuario por el formulario.