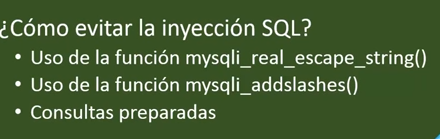
INYECCIÓN SQL

Es una vulnerabilidad presente por lo general en una aplicación web que permite a un atacante enviar peticiones modificadas para interactuar con la base de datos que usa la aplicación.



<https://www.youtube.com/watch?v=3Qw3QQg1xYU>

<https://www.youtube.com/watch?v=wbAbNOgyv9Y>

Ejercicio:

<html >

<head>

<meta http-equiv=**"Content-Type"** content=**"text/html; charset=utf-8"** />

<title>**Documento sin título**</title>

</head>

<body>

<h1> **Estaciones que existen en la población tecleada:**</h1>

<form action=**"j292.php"** method=**"post"** >

<p>**Escribe aquí una población:** <input type=**"text"** name=**"poblacion"** /></p>

<p><input type=**"submit"** name=**"submit"** value=**"ACEPTAR"** /></p>

</form>

</body>

</html>

<?php

$conexion=mysqli\_connect("localhost","root","","trenes") **or** **die**("Error en la conexion");

$registros=mysqli\_query($conexion,"select cod\_estacion,nombre

from estaciones

where poblacion='**$\_POST**[poblacion]'")

**or** **die**("Problemas en el select:".mysqli\_error());

**while** ($reg=mysqli\_fetch\_array($registros))

{

**echo** "código estacion:".$reg['cod\_estacion']."<br>";

**echo** "nombre estacion:".$reg['nombre']."<br>";

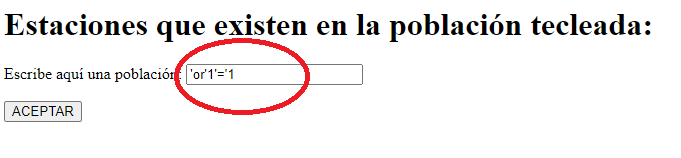
**echo** "<br>";

**echo** "<hr>";

}

mysqli\_close($conexion);

?>



Se visualizan todos los registros de la tabla estaciones.

Para evitar este tipo de inyecciones SQL, lo que debemos hacer es: utilizar la función: mysqli\_real\_escape\_string()

mysqli\_real\_escape\_string — Escapa los caracteres especiales de una cadena para usarla en una sentencia SQL, tomando en cuenta el conjunto de caracteres actual de la conexión

**mysqli\_real\_escape\_string** ( [mysqli](https://www.php.net/manual/es/class.mysqli.php) $link , string $escapestr ) : string

<?php

$conexion=mysqli\_connect("localhost","root","","trenes") **or** **die**("Error en la conexion");

$pob=mysqli\_real\_escape\_string($conexion,$\_POST['poblacion']);

$registros=mysqli\_query($conexion,"select cod\_estacion,nombre

from estaciones

where poblacion='**$pob**'")

**or** **die**("Problemas en el select:".mysqli\_error());

**while** ($reg=mysqli\_fetch\_array($registros))

{

**echo** "código estacion:".$reg['cod\_estacion']."<br>";

**echo** "nombre estacion:".$reg['nombre']."<br>";

**echo** "<br>";

**echo** "<hr>";

}

mysqli\_close($conexion);

?>

Lo que hace la función mysql\_real\_escape\_string es escapar los caracteres, haciendo que no tengan ningún significado especial, por ejemplo, un ‘ significa el final de una variable alfanumérica, pero con un \ antes (escapado), no significa el final, es solo una comilla.  Con lo cual la cadena ' or '1=’1 se convertiría en \' or \'1=\'1 y la claúsula anteriormente comentada se convertiría en WHERE poblacion='\' or \'1=\'1') con lo cual ya no se produciría el efecto indeseado y se evitaría el riesgo de uso inadecuado

Función htmlspecialchars()

 Hola<?php echo htmlspecialchars($\_POST['nombre']); ?>.

Usted tiene <?php echo (int)$\_POST['edad']; ?> años de edad.

Un ejemplo del resultado de esta secuencia de comandos puede ser:

Hola José. Usted tiene 22 años de edad.

htmlspecialchars() se asegura que todos los caracteres que son especiales en html sean codificados adecuadamente de manera que nadie pueda inyectar etiquetas HTML o Javascript en tu página web.

¿qué pasaría si en el campo nombre (me refiero al nombre de $\_POST[‘nombre’]) se introduce algo que es un programa completo?

Por ejemplo:

<script language=”JavaScript”>…….</script>

Cuando llega al navegador cliente comenzará a procesar ese código, que podría por ejemplo, cargar una página falsa con una presentación engañosa para que el visitante no sospeche que fue redireccionado a un sitio falso.

El campo edad, como sabemos que es un número, podemos convertirlo en un integer que automáticamente se deshará de cualquier carácter no numérico.

htmlspecialchars — Convierte caracteres especiales en entidades HTML

Ciertos caracteres tienen un significado especial en HTML y deben ser representados por entidades HTML si se desea preservar su significado. Esta función devuelve un string con estas conversiones realizadas.

Las traducciones realizadas son:

* ‘&’ (et) se convierte en ‘&amp;’
* ‘”‘ (comillas dobles) se convierte en ‘&quot;’ cuando ENT\_NOQUOTES no está establecido.
* “‘” (comilla simple) se convierte en ‘'’ (o &apos;) sólo cuando ENT\_QUOTES está establecido.
* ‘<‘ (menor que) se convierte en ‘&lt;’
* ‘>’ (mayor que) se convierte en ‘&gt;’

CÓDIGO SEGURO CON PHP

Ejemplo.

Una página que lee archivos de un sitio web podría ser la puerta de entrada para que un hacker lea todas las páginas del sitio web.

<HTML>

<HEAD></HEAD>

<BODY>

<?php

// riesgo de seguridad por mala codificación

**if** (**isset**($\_GET['tema'])) {

$tema = $\_GET['tema'];

$var = file($tema);

// muestra el contenido del archivo

// Pero es uno de los archivos que

// realmente queremos mostrar?

// no lo sabemos

**foreach**($var **as** $a) {

**print** $a . "<BR>";

}

}

?>

<HR><FORM METHOD=**'GET'**>**Elegir una letra de una canción**

<SELECT NAME=**"tema"**>

<OPTION VALUE=**"imagine.html"**>**Imagine**

<OPTION VALUE=**"help.html"**>**Help!**

<OPTION VALUE=**"Allmyloving.html"**>**All my loving**

<OPTION VALUE=**"lovemedo.html"**>**Love me do**

<OPTION VALUE=**"pennylane.html"**>**Penny lane**

</SELECT>

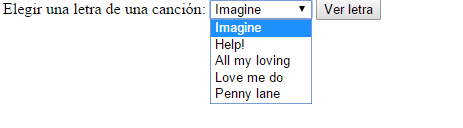
<INPUT TYPE=**"SUBMIT"** VALUE=**"Ver letra"**>

</FORM>

</BODY>

</HTML>

Al ejecutar se visualiza lo siguiente:



Cuando el cliente elige un elemento de la lista se genera esta cadena de consulta:

<http://localhost/ejercicio1.php?tema=imagine.html>

Pero, ¿qué sucede si un hacker codifica en la barra de direcciones del navegador:

<http://localhost/ejercicio1.php?tema=ejercicio2.php>

Lo que ocurre es que le entrega al hacker el contenido completo de una página de la aplicación.

La solución para evitar esta clase de brechas de seguridad es validar el nombre del archivo que se solicitar visualizar.

<HTML>

<HEAD></HEAD>

<BODY>

<?php

**if** (**isset**($\_GET['tema'])) {

esto evitaría la brecha de seguridad

**switch**($\_GET['tema']){

**case** "imagine.html":

**break**;

**case** "help.html":

**break**;

**case** "allmyloving.html":

**break**;

**case** "lovemedo.html":

**break**;

**case** "pennylane.html":

**break**;

**default**:

**die**("archivo inválido");

}

$tema = $\_GET['tema'];

$var = file($tema);

**foreach**($var **as** $a) {

**print** $a . "<BR>";

}

}

?>

<HR><FORM METHOD=**'GET'**>**Elegir una letra de una canción**

<SELECT NAME=**"tema"**>

<OPTION VALUE=**"imagine.html"**>**Imagine**

<OPTION VALUE=**"help.html"**>**Help!**

<OPTION VALUE=**"Allmyloving.html"**>**All my loving**

<OPTION VALUE=**"lovemedo.html"**>**Love me do**

<OPTION VALUE=**"pennylane.html"**>**Penny lane**

</SELECT>

<INPUT TYPE=**"SUBMIT"** VALUE=**"Ver letra"**>

</FORM>

</BODY>

</HTML>

SERVIDOR DE PRODUCCIÓN

El servidor de producción se debe configurar de modo que los mensajes de error no brinden información que pueda resultar útil a un hacer.

Las directivas siguientes del archivo de configuración deben estar en off en un servidor en producción:

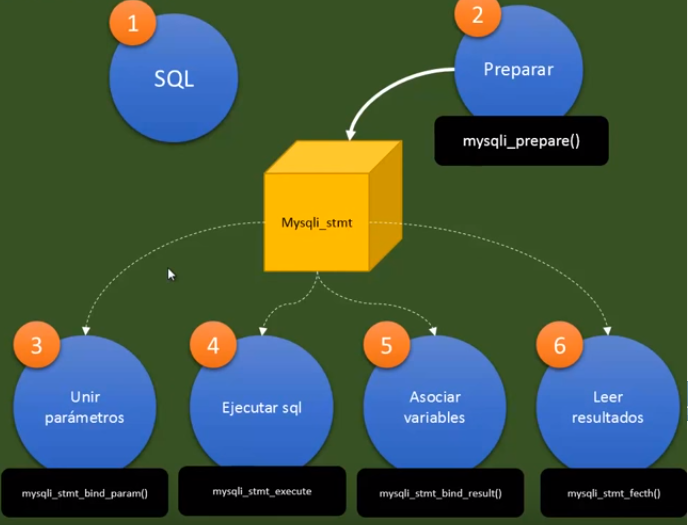
display\_errors=off

html\_errors=off

CONSULTAS PREPARADAS

<https://www.youtube.com/watch?v=UMoekwmZsRY>

<https://www.youtube.com/watch?v=4quM806MBvk>



Ejercicio:

Utilizando consultas preparadas obtener los códigos y nombres de las estaciones de Bilbao.

Ejercicio:

Utilizando consultas preparadas dar de alta a una estación.