



Tecnologías Web

Seguridad en PHP

XSS, SQLI, CMDI Y LFI



Índice

- INTRODUCCIÓN
- CROSS SITE SCRIPTING
 - NO PERSISTENTE
 - PERSISTENTE
- CMD INJECTION
- SQL INJECTION
- LOCAL FILE INCLUSION
- CONCLUSION

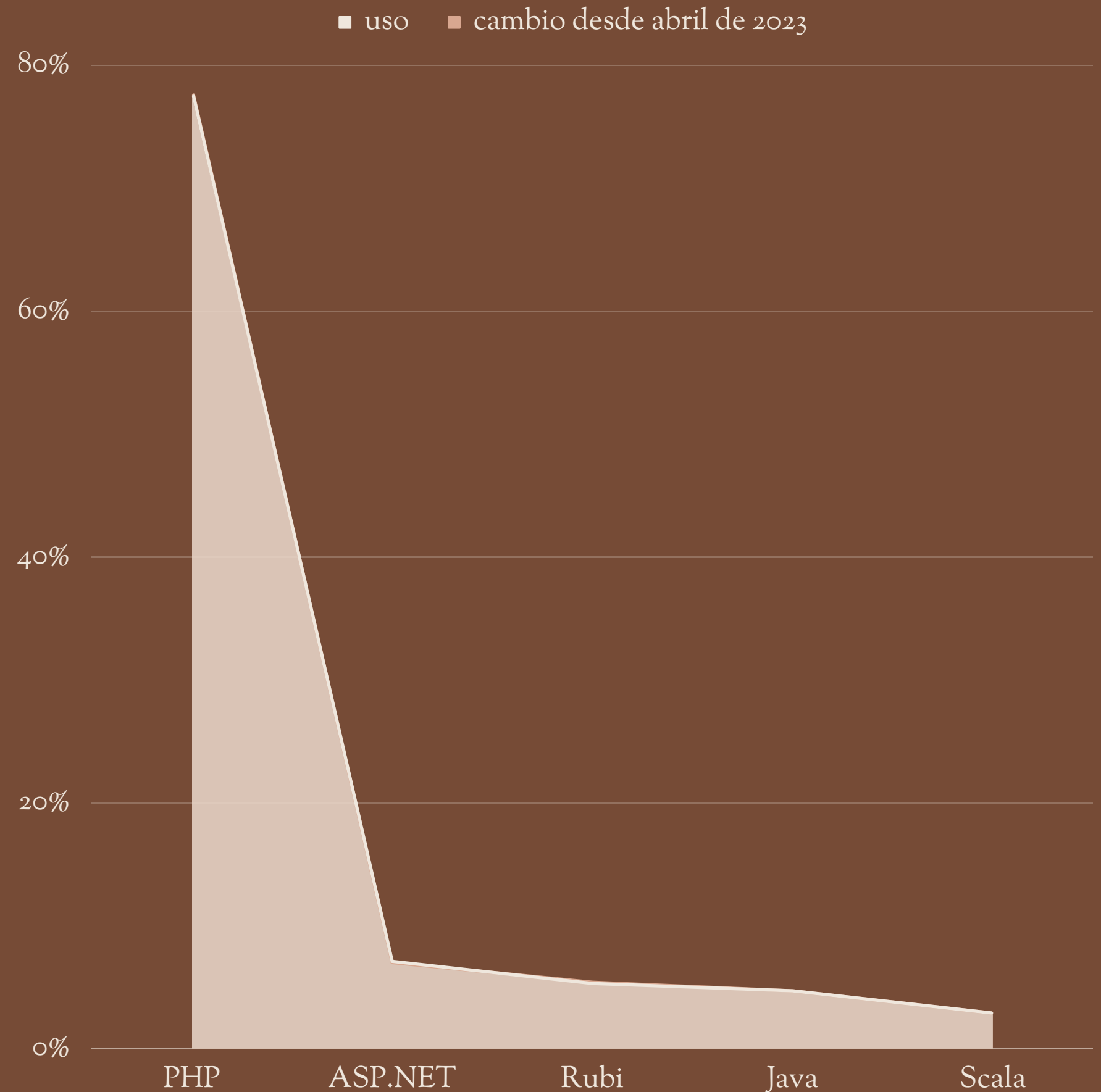




Introducción

“La seguridad no está en el lenguaje utilizado sino en la capacidad de implementarla del programador.”

LENGUAJES DE PROGRAMACIÓN MÁS POPULARES PARA EL BACKEND



Fuente: <https://w3techs.com/>



Orientado al desarrollo de BD

Fácil de aprender y usar

Amplia comunidad de desarrollo

Ventajas

Actualizaciones regulares

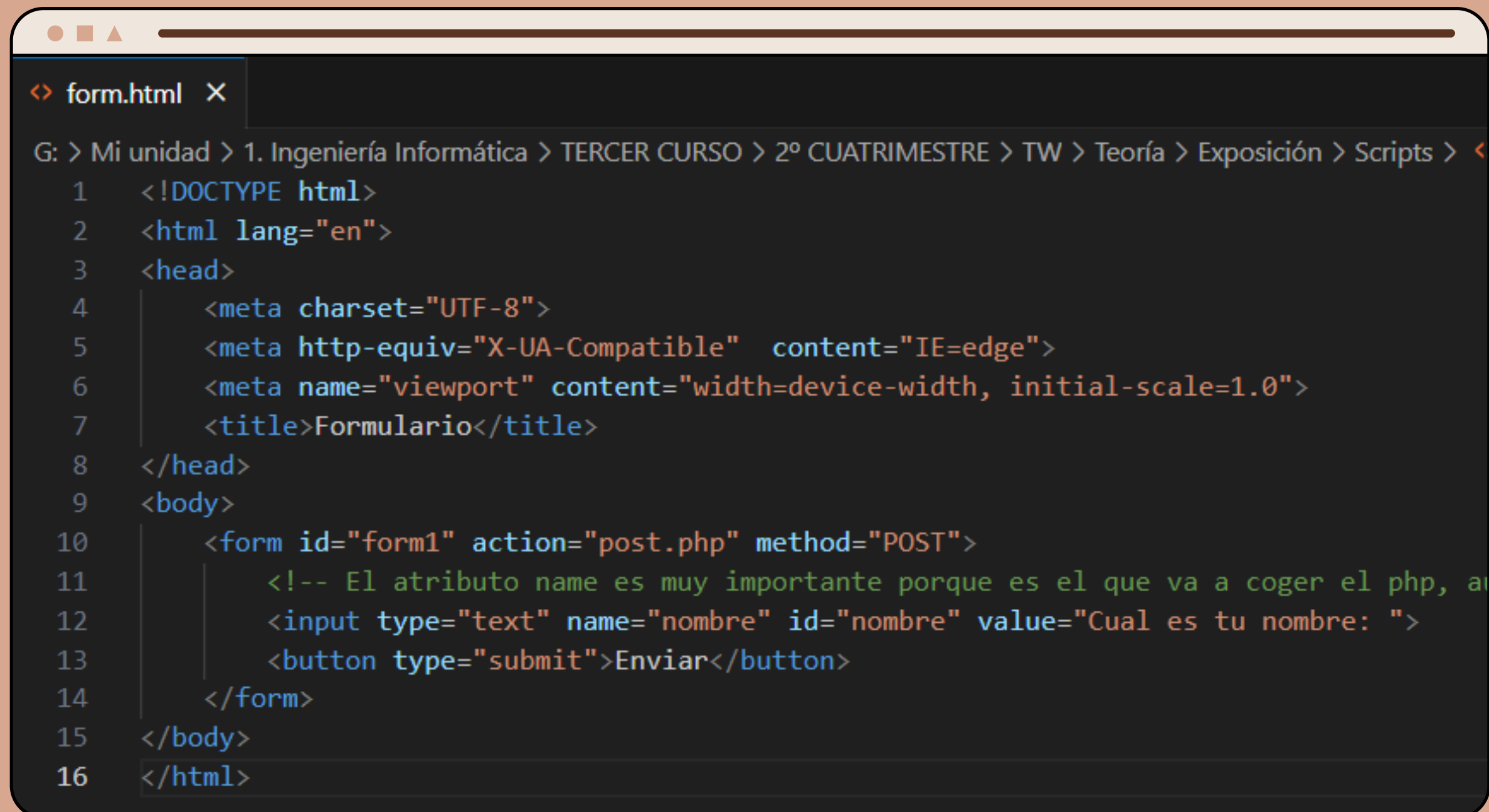
Eficiente

Versátil



XSS

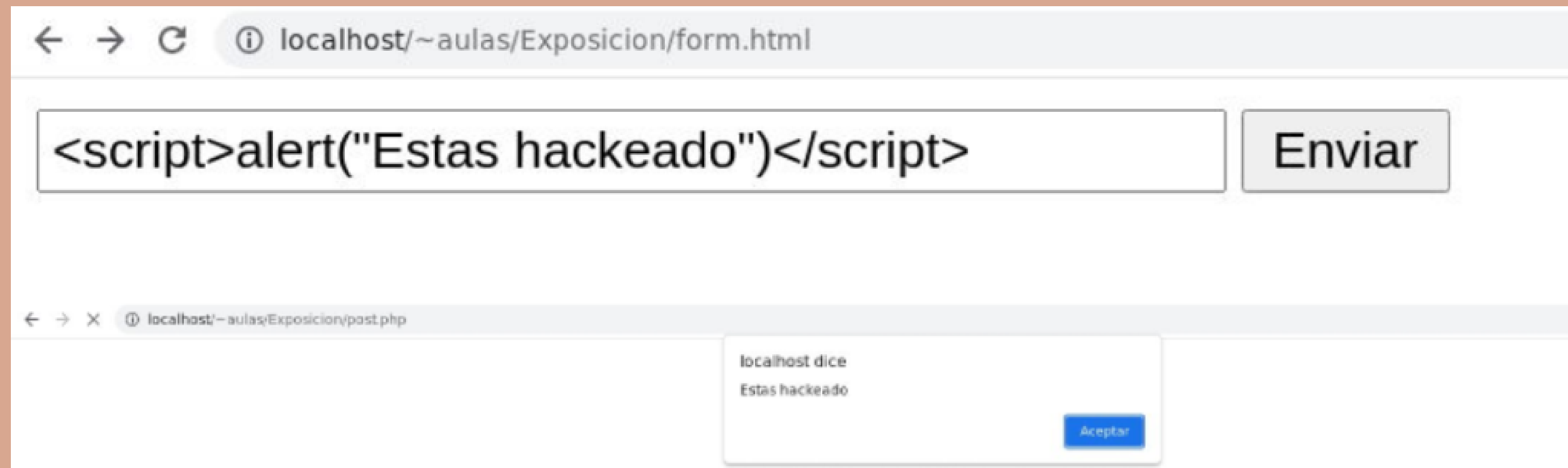
CROSS SITE SCRIPTING



```
<> form.html X
G: > Mi unidad > 1. Ingeniería Informática > TERCER CURSO > 2º CUATRIMESTRE > TW > Teoría > Exposición > Scripts > <
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Formulario</title>
8  </head>
9  <body>
10     <form id="form1" action="post.php" method="POST">
11         <!-- El atributo name es muy importante porque es el que va a coger el php, al
12         <input type="text" name="nombre" id="nombre" value="Cual es tu nombre: ">
13         <button type="submit">Enviar</button>
14     </form>
15 </body>
16 </html>
```

XSS

CROSS SITE SCRIPTING





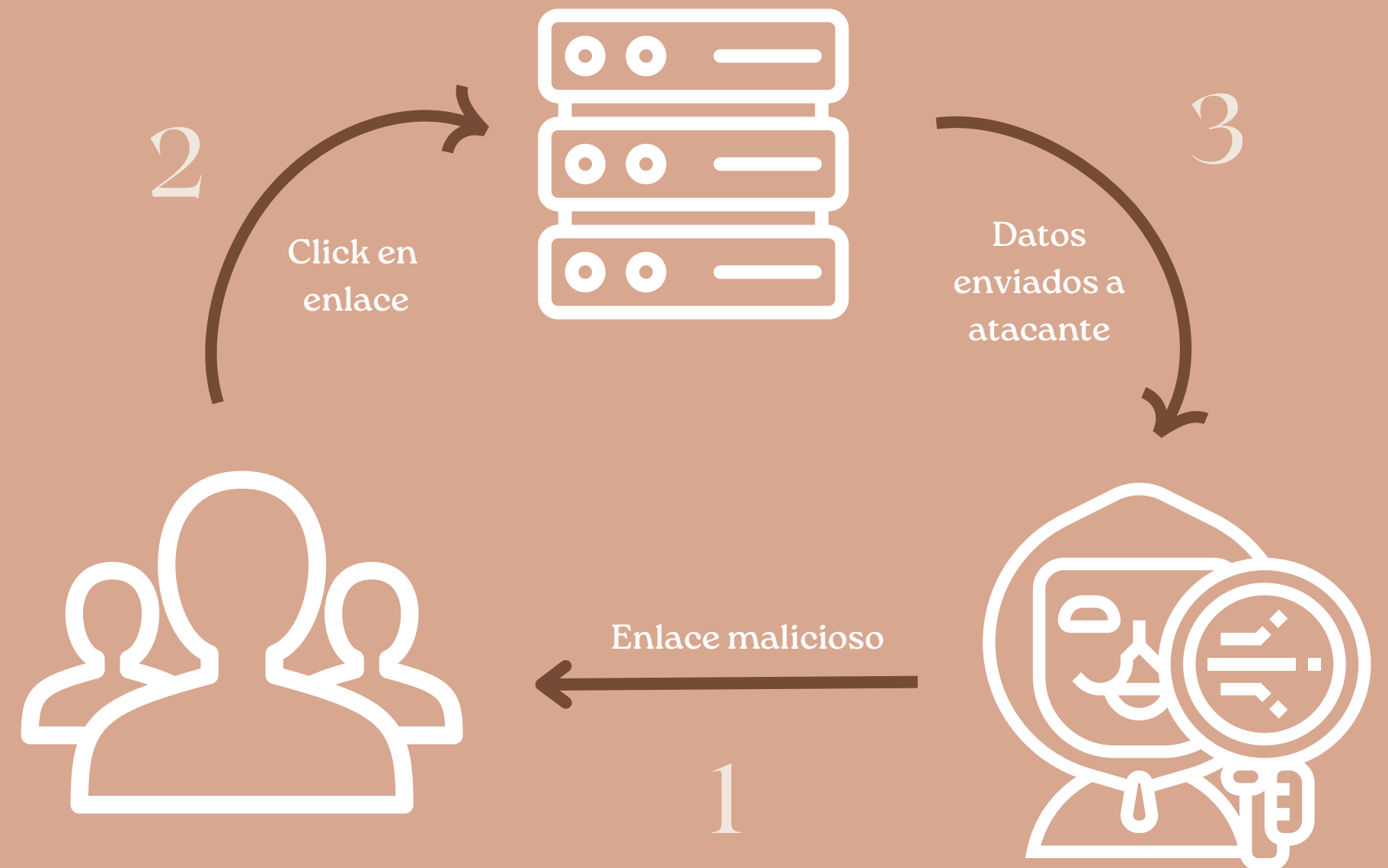
Ejemplos reales

	CASO 1	CASO 2	CASO 3
AÑO	2005	2008	2013
ENTIDAD AFECTADA	MySpace (red social)	Web de campaña presidencial de Barack Obama	Yahoo Mail
INFORMACIÓN	"Samy worm" +1 millón de perfiles afectados	Robo de información de cuentas de afiliados	Aprovecharon un zero-day para robar cookies y acceder a correos electrónicos

XSS No Persistente

Reflejado

A pesar de que el código que se inyecta es en la página web, los que se ven afectados son los usuarios finales de la web, pudiendo robarse sus cookies, redirigir a otras web, o ejecutar código malicioso para ganar acceso a la máquina del usuario.



XSS

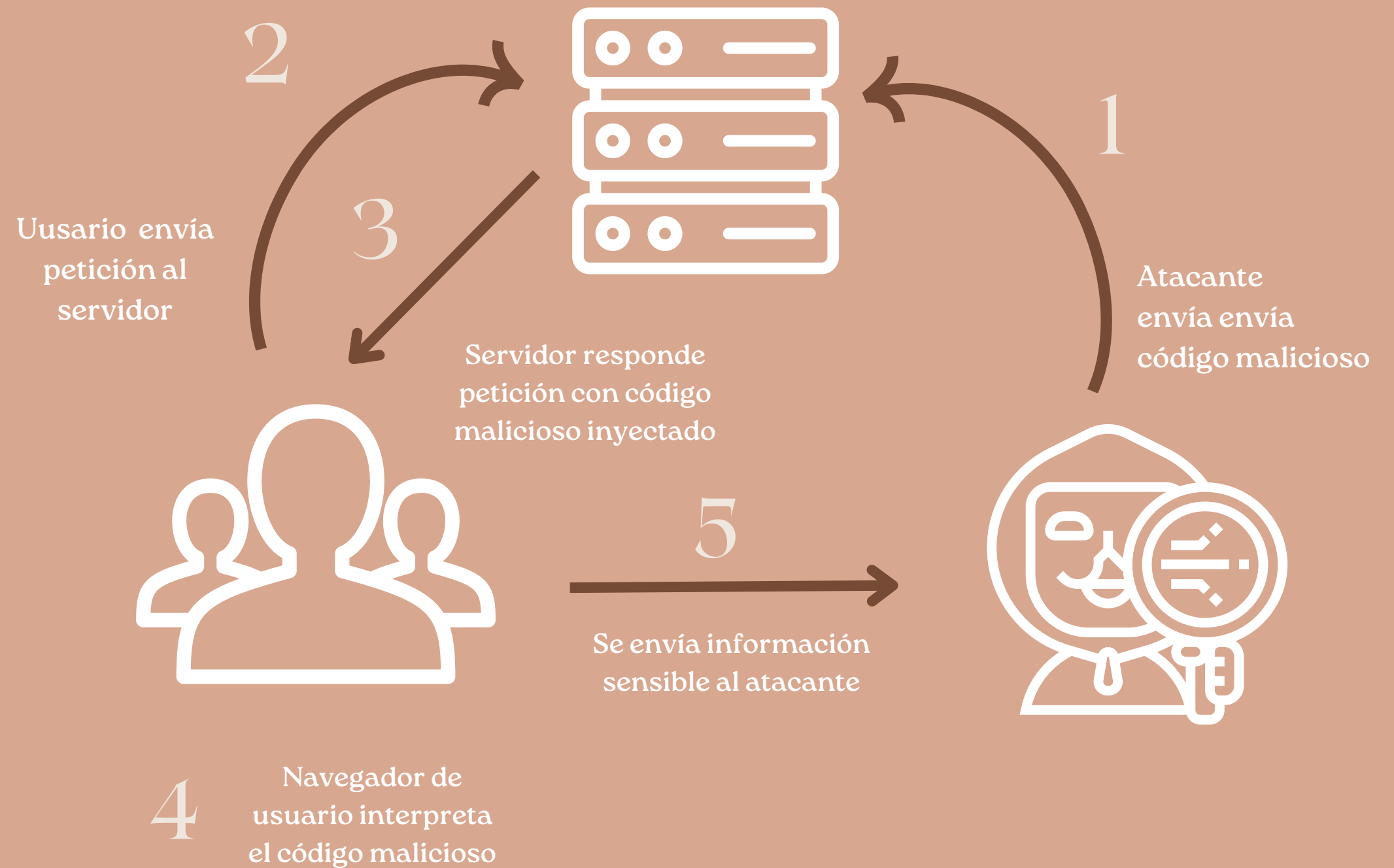
Persistente

Almacenado

Puede suceder en varias partes del servidor:

- Comentarios de usuarios
- Publicaciones en foros
- Etc.

Es un problema que debemos de solucionar de cara al proyecto final de prácticas para nuestro foro de quejas.





Prevención de XSS

```
if ($input['email']) {  
    if (si la dirección de correo e  
        ar($email, FILTER_VALIDATE_E  
        dirección de correo electróni  
        ección de correo electróni
```

VALIDACIÓN DE DATOS

Asegurarnos de que los datos que se van a enviar por medio del formulario se rijan por un control específico, por ejemplo por expresiones regulares.

```
trada  
r($data, FILTER_SANITIZE_  
: " . $dataSanitized;
```


SANITIZACIÓN DE DATOS

Hacer que los datos cumplan con unas condiciones específicas y que ejecuten un resultado preciso de los datos



XSS

EJEMPLO DE VALIDACIÓN DE DATOS


 datos_validados.php

```
1  <?php
2  $email = $_POST['email'];
3
4  // Verificar si la dirección de correo es válida
5  if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
6      echo("La dirección de correo electrónico es válida");
7  } else {
8      echo("La dirección de correo electrónico no es válida");
9  }
10 ?>
```



XSS

EJEMPLO DE SANITIZACIÓN DE DATOS

 datos_sanitizados.php

```
1  <?php
2  $data = $_POST['data'];
3
4  // Limpiar la cadena de entrada
5  $dataSanitized = filter_var($data, FILTER_SANITIZE_STRING);
6
7  echo "Los datos saneados son: " . $dataSanitized;
8  ?>
9
```

CMD INJECTION

```
PING www.google.com ([REDACTED]) 56(84) bytes of data.  
64 bytes from mil04s25-in-f4.1e100.net ([REDACTED]): icmp_seq=1 ttl=51 time=67.  
64 bytes from mil04s25-in-f4.1e100.net ([REDACTED]): icmp_seq=2 ttl=51 time=68.  
64 bytes from mil04s25-in-f4.1e100.net ([REDACTED]): icmp_seq=3 ttl=51 time=77.  
64 bytes from mil04s25-in-f4.1e100.net ([REDACTED]): icmp_seq=4 ttl=51 time=67.  
  
--- www.google.com ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3005ms  
rtt min/avg/max/mdev = 67.696/70.334/77.673/4.247 ms  
www-data
```

Comando ejecutado

Máquina Linux



Ejemplos reales

	CASO 1	CASO 2	CASO 3
AÑO	2010	2012	2014
ENTIDAD AFECTADA	Apache	GoDaddy	sistemas Unix, en particular a los sistemas que utilizaban la Bash Shell
INFORMACIÓN	Vulnerabilidad que permitía inyección de comandos	Permitió a los atacantes redirigir el tráfico de los sitios web de sus clientes	Ataque "Shellshock", que permitía ejecutar comandos arbitrarios en estos sistemas



Protección frente a CMD Injection

- VALIDAR ENTRADA DE USUARIO
- ESCAPAR ENTRADA DE USUARIO
- USAR FUNCIONES DE ALTO NIVEL
- MINIMIZAR PERMISOS DEL USUARIO
- USAR DIRECTIVAS DE SEGURIDAD
- PRUEBAS DE SEGURIDAD PERIÓDICAS

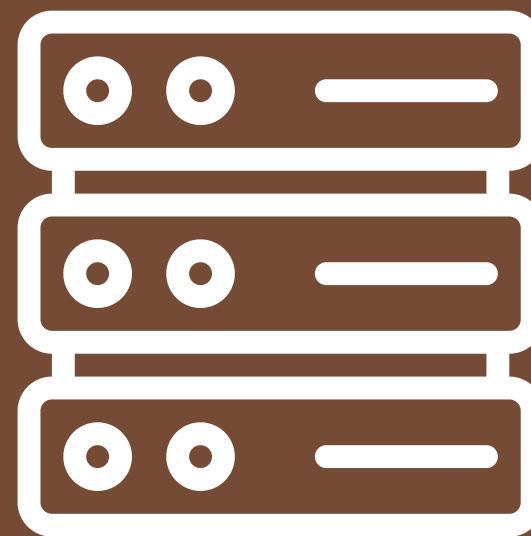




SQL INJECTION



Atacante



Web API Server



SQL Database



Ejemplos reales

	CASO 1	CASO 2	CASO 3
AÑO	2008	2011	2012
ENTIDAD AFECTADA	Heartland Payment Systems	Yahoo	Sony Pictures
INFORMACIÓN	La mayor brecha de seguridad de tarjetas de crédito en la historia hasta esa fecha.	Exposición de 450,000 nombres de usuario y contraseñas.	Grupo de hackers LulzSec Exposición de datos de alrededor de un millón de usuarios.



Funciones utilizadas:

- `str_ireplace()`
- `trim()`
- `stripslashes()`

Prevención de SQL Injection

SCRIPT ÚTIL

```
<?php

function limpiar_cadena($cadena){

    // También sirve para evitar ataques XSS:
    $cadena=str_ireplace("<script>", "", $cadena);
    $cadena=str_ireplace("</script>", "", $cadena);
    $cadena=str_ireplace("<script src", "", $cadena);
    $cadena=str_ireplace("<script type=", "", $cadena);

    $cadena=str_ireplace("SELECT * FROM", "", $cadena);
    $cadena=str_ireplace("DELETE FROM", "", $cadena);
    $cadena=str_ireplace("INSERT INTO", "", $cadena);
    $cadena=str_ireplace("DROP TABLE", "", $cadena);
    $cadena=str_ireplace("DROP DATABASE", "", $cadena);
    $cadena=str_ireplace("TRUNCATE TABLE", "", $cadena);
    $cadena=str_ireplace("SHOW TABLES", "", $cadena);
    $cadena=str_ireplace("SHOW DATABASES", "", $cadena);

    $cadena=str_ireplace("<?php", "", $cadena);
    $cadena=str_ireplace(">", "", $cadena);
    $cadena=str_ireplace("^", "", $cadena);
    $cadena=str_ireplace("<", "", $cadena);
    $cadena=str_ireplace("[", "", $cadena);
    $cadena=str_ireplace("]", "", $cadena);
    $cadena=str_ireplace("==", "", $cadena);
    $cadena=str_ireplace("; ", "", $cadena);
    $cadena=str_ireplace("::", "", $cadena);

    $cadena=trim($cadena);
    $cadena=stripslashes($cadena);

    return $cadena;
}

?>
```




Funciones utilizadas:

• `str_ireplace()` →

Equivalente a `str_replace()`
pero insensible a mayúsculas y
minúsculas

• `trim()` →

Limpiar espacios en blanco

• `stripslashes()` →

Quitar barras invertidas de strings
con comillas escapadas

Prevención de SQL Injection

SCRIPT ÚTIL

```
<?php

function limpiar_cadena($cadena){

    // También sirve para evitar ataques XSS:
    $cadena=str_ireplace("<script>", "", $cadena);
    $cadena=str_ireplace("</script>", "", $cadena);
    $cadena=str_ireplace("<script src", "", $cadena);
    $cadena=str_ireplace("<script type=", "", $cadena);

    $cadena=str_ireplace("SELECT * FROM", "", $cadena);
    $cadena=str_ireplace("DELETE FROM", "", $cadena);
    $cadena=str_ireplace("INSERT INTO", "", $cadena);
    $cadena=str_ireplace("DROP TABLE", "", $cadena);
    $cadena=str_ireplace("DROP DATABASE", "", $cadena);
    $cadena=str_ireplace("TRUNCATE TABLE", "", $cadena);
    $cadena=str_ireplace("SHOW TABLES", "", $cadena);
    $cadena=str_ireplace("SHOW DATABASES", "", $cadena);

    $cadena=str_ireplace("<?php", "", $cadena);
    $cadena=str_ireplace(">", "", $cadena);
    $cadena=str_ireplace("^", "", $cadena);
    $cadena=str_ireplace("<", "", $cadena);
    $cadena=str_ireplace("[", "", $cadena);
    $cadena=str_ireplace("]", "", $cadena);
    $cadena=str_ireplace("==", "", $cadena);
    $cadena=str_ireplace("; ", "", $cadena);
    $cadena=str_ireplace("::", "", $cadena);

    $cadena=trim($cadena);
    $cadena=stripslashes($cadena);

    return $cadena;
}

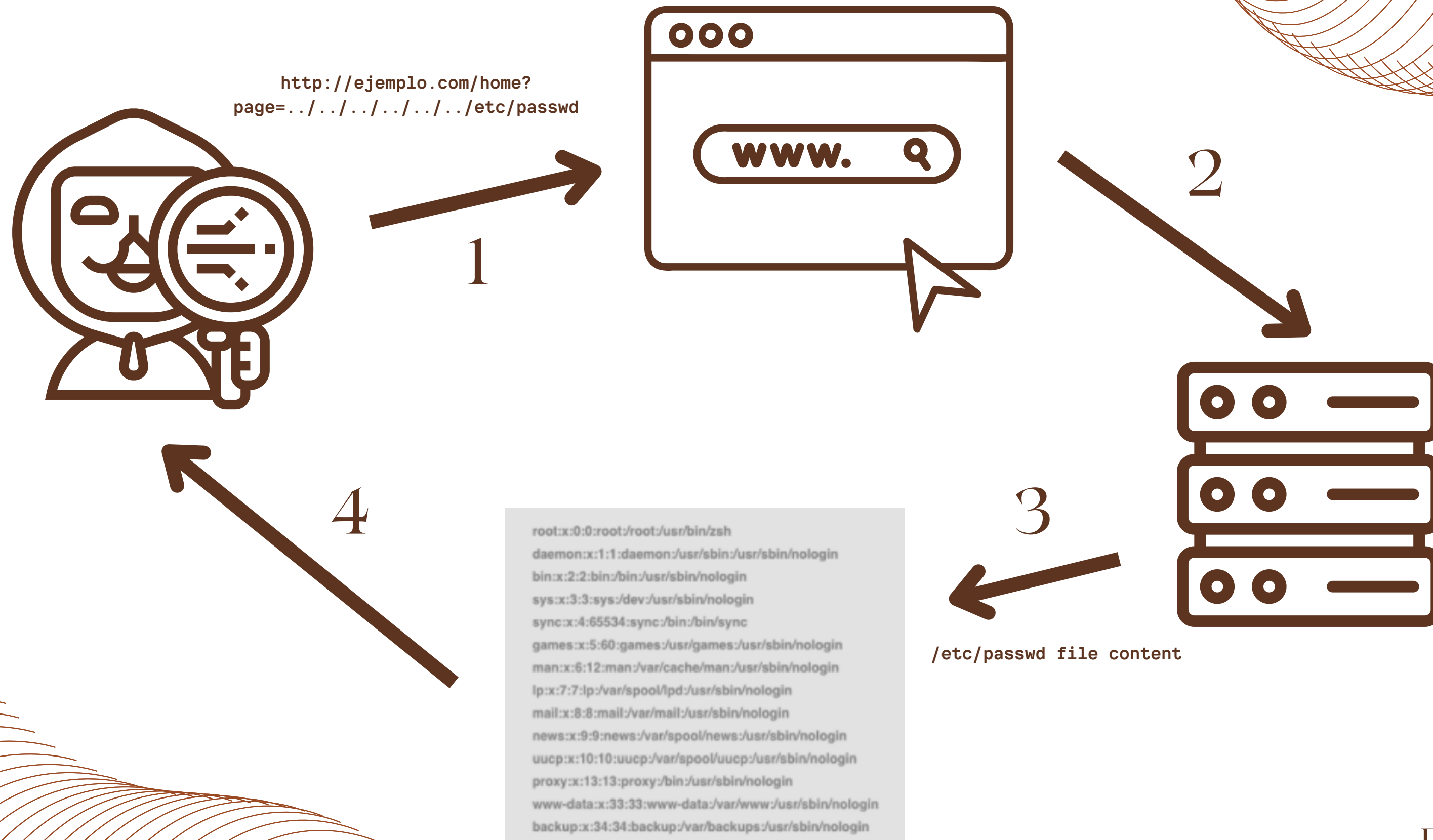
?>
```



Tecnologías Web

LFI

LOCAL FILE INCLUSION



* Técnica
Path Traversal

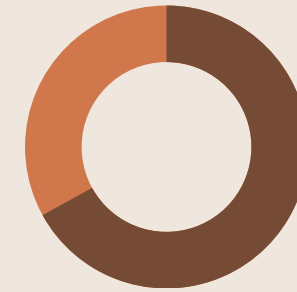


Ejemplos reales

	CASO 1	CASO 2	CASO 3
AÑO	2010	2011	2012
ENTIDAD AFECTADA	WEB PHP.NET	WORDPRESS	AGENCIA ESPACIAL EUROPEA
INFORMACIÓN	Acceso a archivos sensibles del servidor	Modificar código de ficheros para cargar y ejecutar código malicioso	Obtener acceso a la base de datos

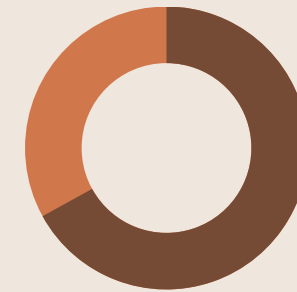


Prevención de LFI



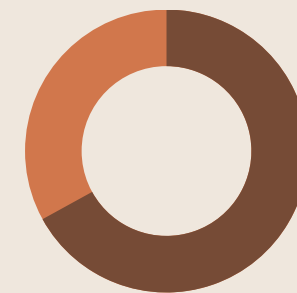
EVITAR POR COMPLETO PASAR
NOMBRES DE ARCHIVO EN LA
ENTRADA DEL USUARIO.

Evita manipular cookies, entre otras cosas



CREAR UNA WHITELIST DE
ARCHIVOS SEGUROS

Si se requiere usar nombres de archivo de la
entrada del usuario



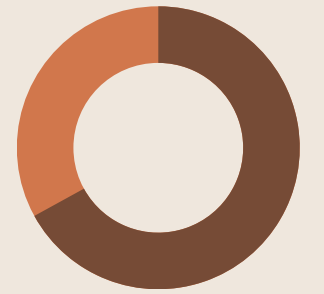
USAR STR_REPLACE()

```
<?php
    $file = str_replace('../', '', $_GET['file']);
    if(isset($file))
    {
        include("/var/www/html/$file");
    }
?>
```



USAR STR_REPLACE()

Como hacíamos para evitar SQL Injection



```
<?php

    $file = str_replace('../', '', $_GET['file']);

    if(isset($file))
    {

        include("/var/www/html/$file");

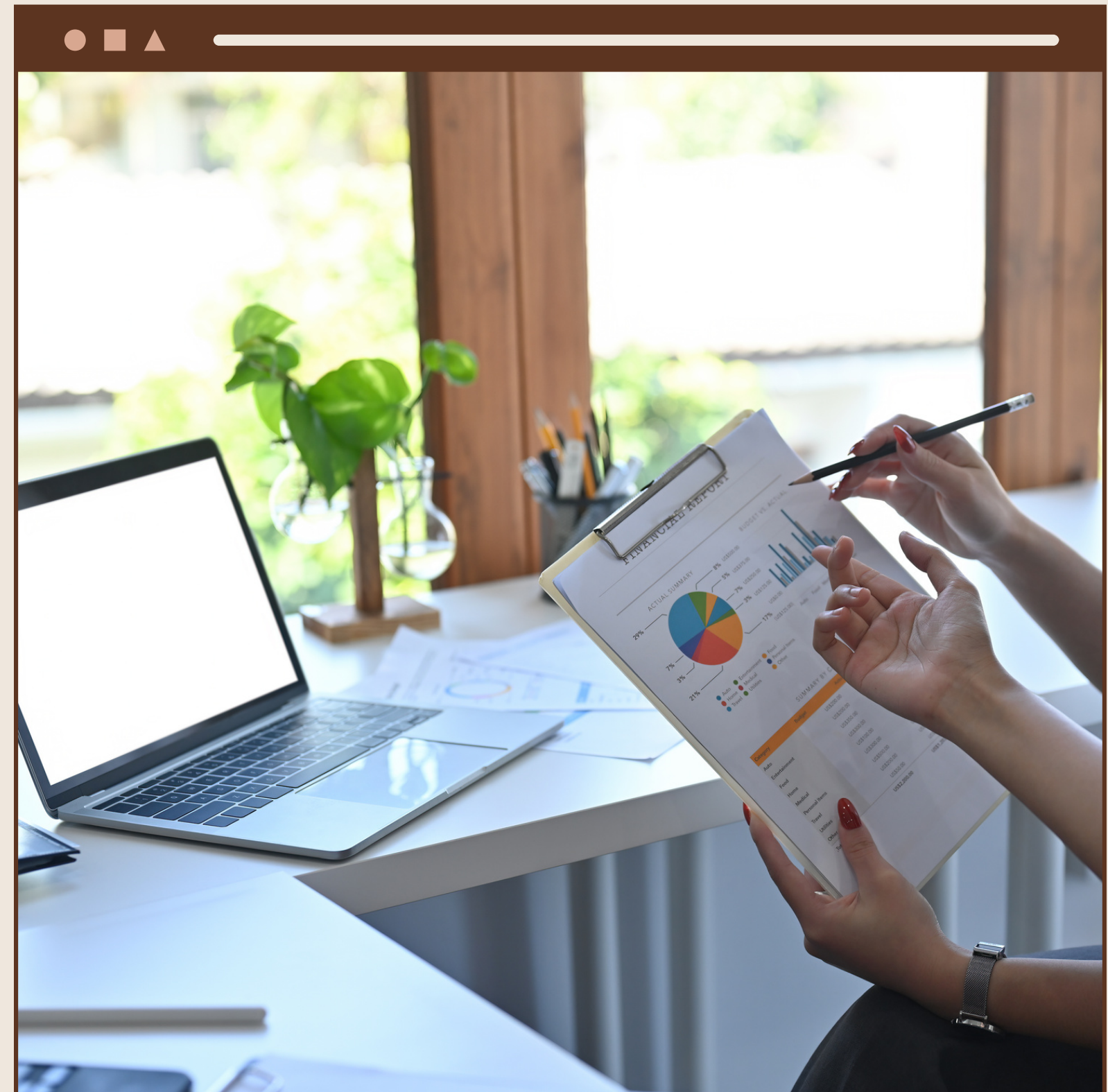
    }

?>
```

CONCLUSIONES

Garantizar la seguridad de un sitio web es un **proceso continuo** que requiere una comprensión de las posibles vulnerabilidades y cómo mitigarlas.

Las **buenas prácticas de codificación**, como el uso de funciones de seguridad adecuadas y la realización regular de pruebas de seguridad son esenciales para mantener una defensa robusta contra las amenazas de seguridad cibernética.





Tecnologías Web

¿Alguna pregunta?