

**Instituto Politécnico
Nacional**



Escuela superior de computo

Tema:

XSS Cross-Site Scripting.

Asignatura:

Gobierno de TI (Computer security)

Alumno:

Julio Cesar Hernández Reyes

Docente:

Alejandro Sigfrido Cifuentes Álvarez

Grupo:

6CM2

Fecha:

05/Enero/2022

Introducción:

Los ataques XSS Cross-Site Scripting pueden ser:

→ XSS persistente o directo o almacenado. Anida código HTML por medio de etiquetas `<script>` o `<iframe>`. El

código se implanta en el sitio y se ejecuta al abrir la página web. Se elimina protegiendo todo el código del sitio.

También el uso incorrecto del DOM con JavaScript puede permitir abrir otra página web con código peligroso anidado

dañando la página principal del sitio web.

→ XSS reflejado o indirecto. Reflejado en la página. Edita los mensajes, rutas, cookies o encabezados HTTP que se pasan con URL, ingresando código dañino que se ejecuta en el sitio.

→ XSS mutante. Utiliza las propiedades de `innerHTML`, atributo utilizado en varios frameworks JavaScript, para

convertir una cadena HTML en parte del DOM, analizando las etiquetas y atributos.

Un ataque por XSS toma información de un usuario y la reenvía, sin validar, a un navegador Web. Con XSS se ejecutan

secuencias de comandos en los navegadores del usuario; además, puede secuestrar las sesiones del usuario, incluso

modificar otros sitios Web al insertar contenido dañino y atacar con Phishing (obtiene información confidencial de nombres

de usuario, contraseñas y datos de tarjetas de crédito al suplantar una comunicación válida, con secuencias peligrosas de

comandos, en el navegador del usuario. Cualquier contenido activo anidado es una fuente potencial de peligro, por ejemplo

incluir ActiveX (OLE), VBScript, Shockwave, Flash y otros.

Para proteger este tipo de ataques XSS se sugieren las siguientes precauciones:

→ Pasar la salida a través de `htmlspecialchars()`, `htmlspecialchars()` o OWASP PHP Anti-XSS, para verificarla;

además, deshabilitar `register_globals`, si es que se encuentra habilitada.

→ Validar cualquier información extraña con longitud, tipo, sintaxis o reglas de negocio.

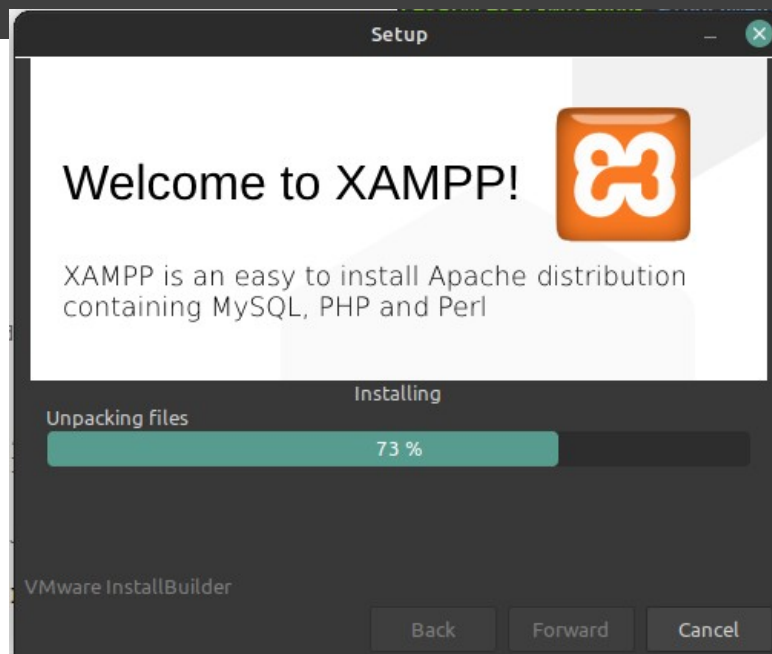
→ Decodificar y canonizar los valores de entrada con su representación interna de la aplicación antes de validarlos.

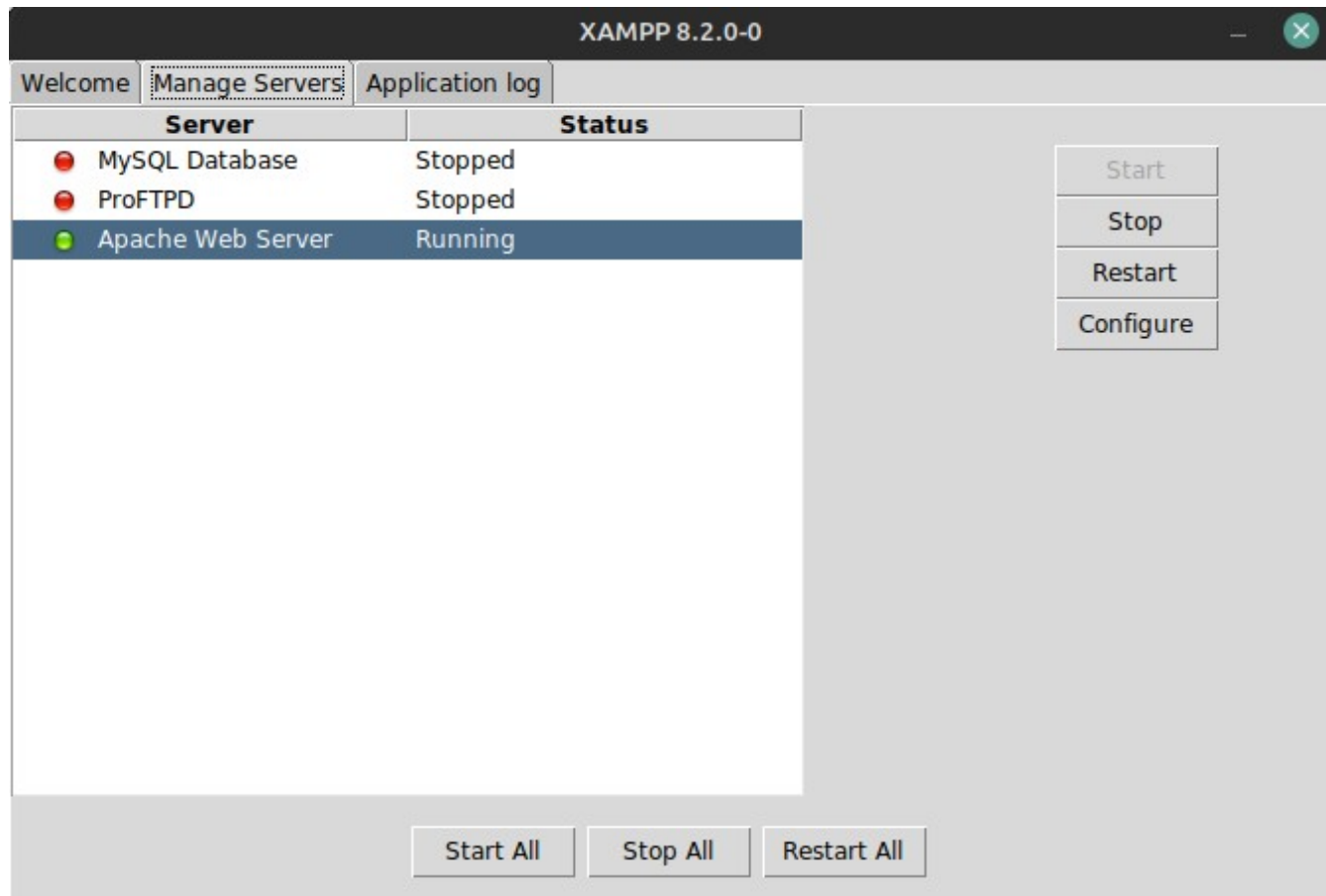
→ Especificar la codificación de la salida con UTF-8 o ISO 8859-1 (latín occidental).

DESARROLLO:

Instalación de Xampp, porque no se tenía instalado:

```
cesar@cesar-Notebook: ~/Documentos/6.Semestre/Security/XSS
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
cesar@cesar-Notebook:~/Documentos/6.Semestre/Security/XSS$ ls
ReporteXSS.odt  xampp-linux-x64-8.2.0-0-installer.run  xssl.html  xssl.php
cesar@cesar-Notebook:~/Documentos/6.Semestre/Security/XSS$ sudo chmod +x xampp-linu
x-x64-8.2.0-0-installer.run
[sudo] password for cesar:
cesar@cesar-Notebook:~/Documentos/6.Semestre/Security/XSS$ ls
ReporteXSS.odt  xampp-linux-x64-8.2.0-0-installer.run  xssl.html  xssl.php
cesar@cesar-Notebook:~/Documentos/6.Semestre/Security/XSS$ ./xampp-linux-x64-8.2.0-
0-installer.run
cesar@cesar-Notebook:~/Documentos/6.Semestre/Security/XSS$ sudo su
/usr/bin/env: 'node': No such file or directory
root@cesar-Notebook:/home/cesar/Documentos/6.Semestre/Security/XSS# ^C
root@cesar-Notebook:/home/cesar/Documentos/6.Semestre/Security/XSS# exit
exit
cesar@cesar-Notebook:~/Documentos/6.Semestre/Security/XSS$ sudo ./xampp-linux-x64-8
.2.0-0-installer.run
```







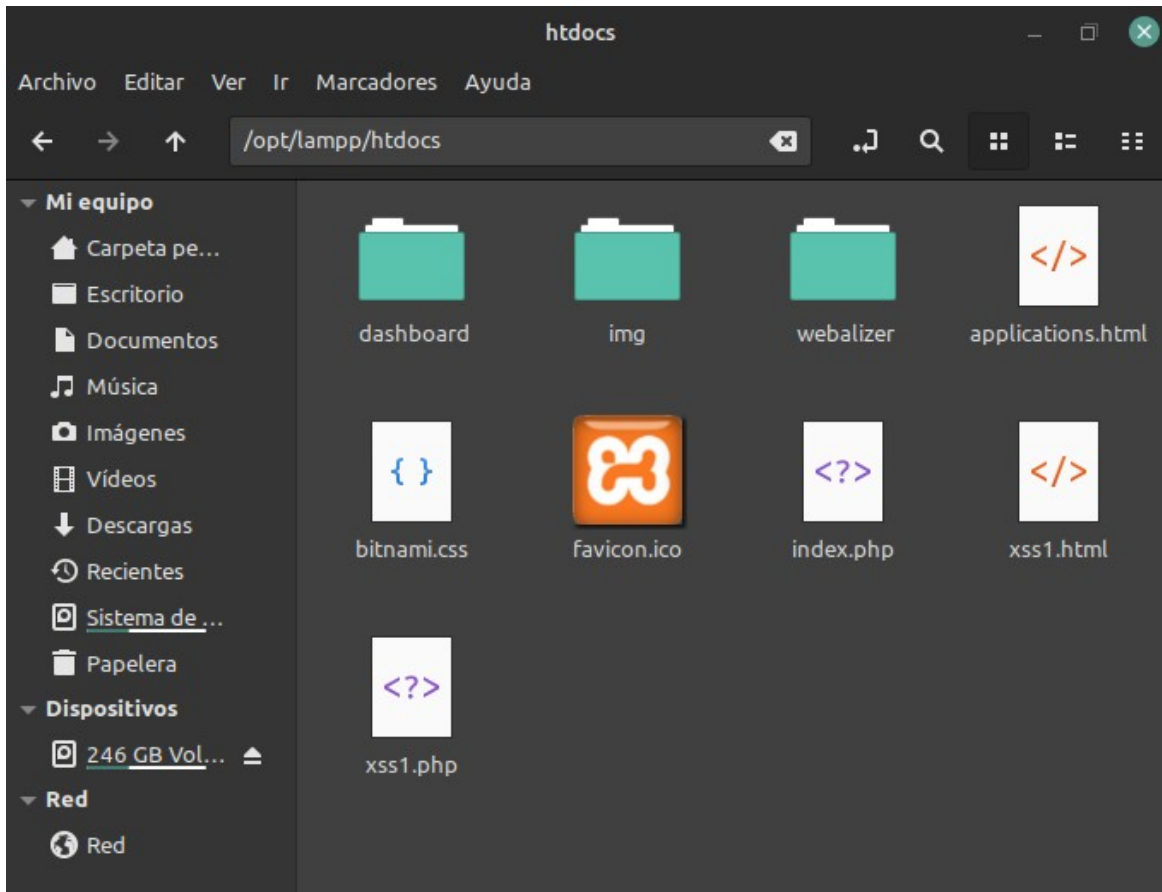
EJEMPLO 1:

Capturar los dos siguientes archivos en el bloc de notas:

```
xss1.html x xss1.php x
1 <html>
2   <head>
3     <meta charset="utf-8"/>
4   </head>
5   <body>
6     <form action="xss1.php">
7       Nombre:
8       <input type="text" name="captura"/>
9       <input type="submit" value="enviar"/>
10    </form>
11  </body>
12 </html>
13
```

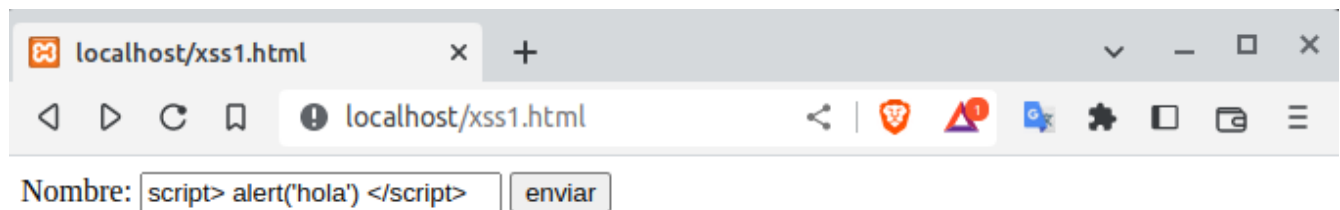
```
xss1.html x xss1.php x
1 <?php
2   if(isset($_GET['captura'])){
3     echo "Has buscado: ".$_GET["captura"];
4   }
5   ?>
6
```

Guardar los archivos en el directorio de htdocs de xampp:



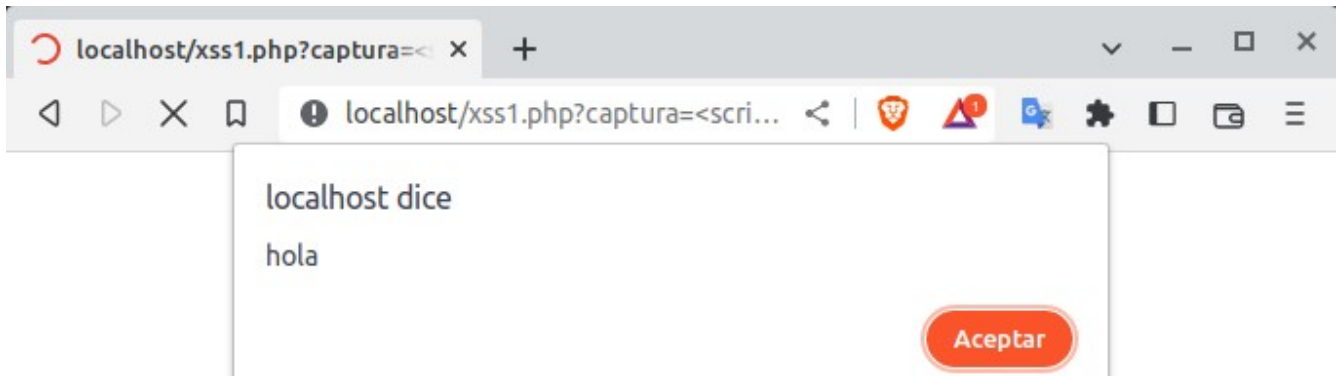
En el navegador se muestran las siguientes ventanas. En el campo de texto, ingresar la siguiente cadena y digitar en enviar:

```
<script> alert('hola') </script>
```

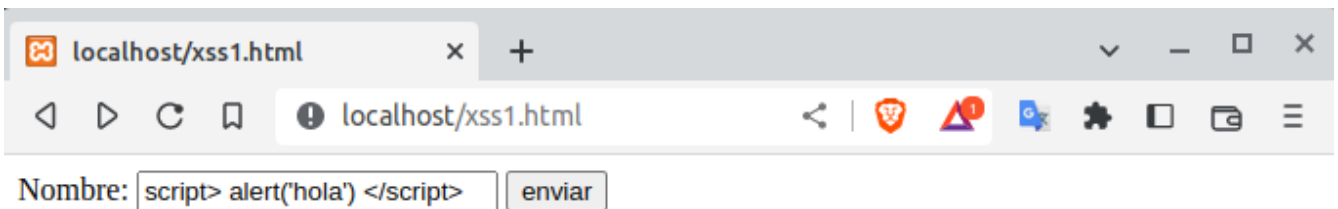


La cadena enviada no se ejecuta como un script y no se muestra porque el navegador Chrome está protegido contra este tipo de ataque XSS; otros navegadores no están protegidos contra ello. Por lo anterior, investigar y probar cuáles navegadores están o no protegidos contra XSS.

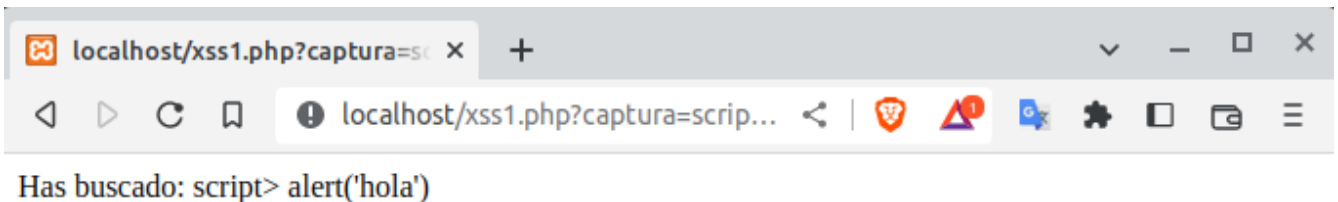
En el navegador Brave si se ejecuto el <script>:



Por otro lado, si ahora se ingresa una cadena en la que se omite el símbolo < de la etiqueta <script> o algo similar, por ejemplo la siguiente: `script> alert('hola') </script>`



Resultado:



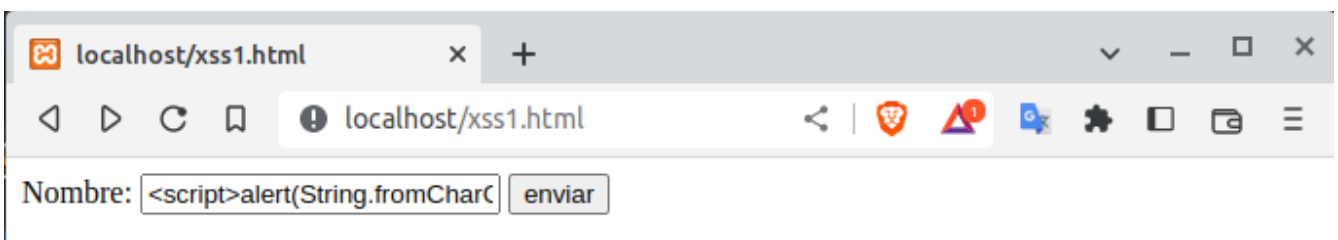
Ejercicio 1. Utilizar la función `String.fromCharCode` para cifrar un mensaje en ASCII. Por ejemplo: `String.fromCharCode(88,83,83)`

Para utilizarse en una consulta completa:

```
<script>alert(String.fromCharCode(88,83,83))</script>
```

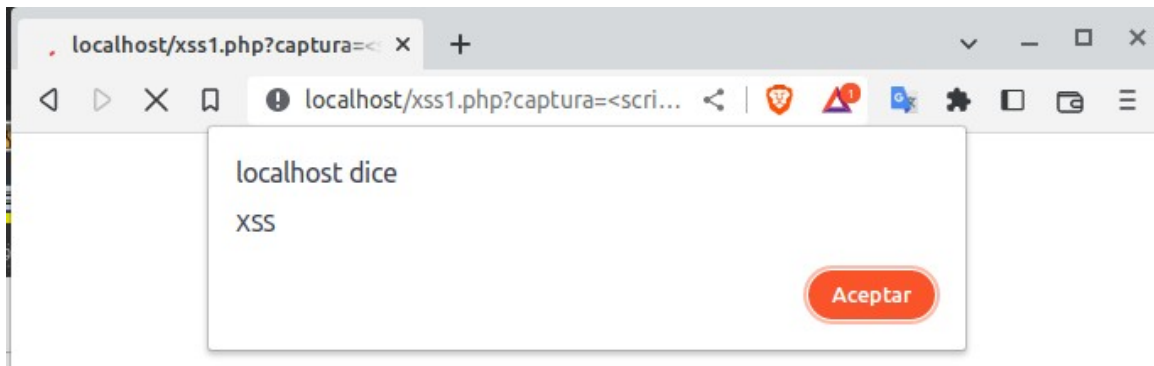
Verificar la ejecución y documentar la aplicación:

Ingresamos la cadena:



Resultado:

Se obtuvo como resultado la palabra XSS que es el tema actual:



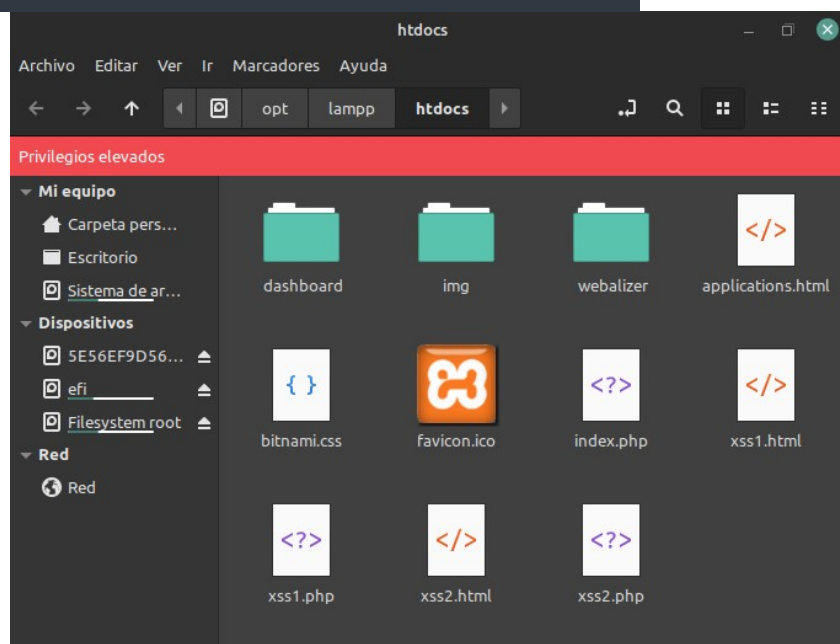
EJEMPLO 2:

Modificación de la URL. Capturar los dos siguientes archivos en el bloc de notas:

```
xss2.html xss2.php x
1 <html>
2   <head>
3     <meta charset="utf-8"/>
4   </head>
5   <body>
6     <form action="xss2.php">
7       Nombre:
8       <input type="text" name="nombre"/>
9       <input type="submit" value="Enviar"/>
10    </form>
11  </body>
12 </html>
```

```
xss2.html xss2.php x
1 <?php
2   $nombre = $_GET['nombre'];
3   echo "Hola: $nombre<br>";
4   echo "<a href='https://www.youtube.com/'>Digitar
5     aqu&iacute; para descargar</a>";
6 >
```

Se guardan en htdocs:



En el navegador se muestran las siguientes ventanas. En el campo de texto, ingresar la siguiente cadena (en una sola línea)

y digitar en Enviar:

xss_prueba.php?nombre=

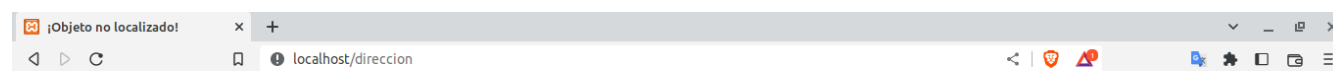
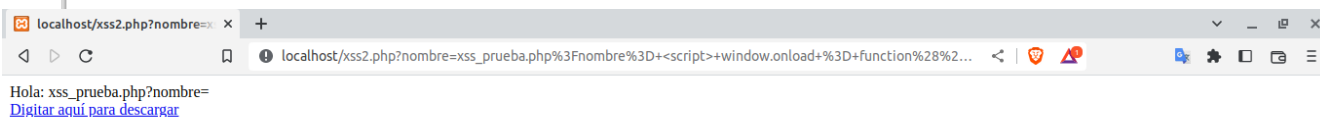
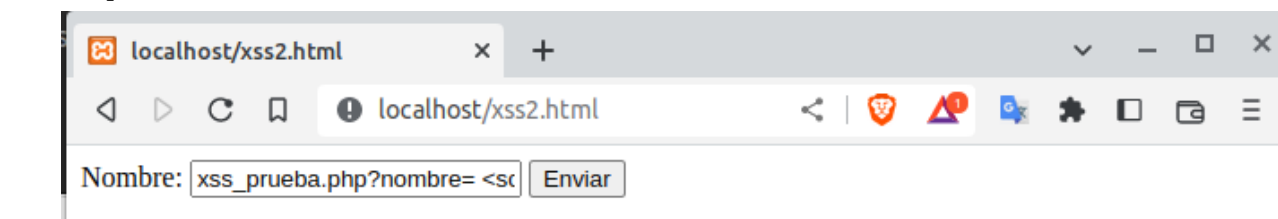
<script>

```
    window.onload = function(){
```

```
        var link = document.getElementsByTagName("a");link[0].href="direccion";
```

```
    }
```

</script>



¡Objeto no localizado!

No se ha localizado la URL solicitada en este servidor. La URL de la [página que le ha remitido](#) parece ser errónea o estar obsoleta. Por favor, informe del error al autor de [esa página](#).

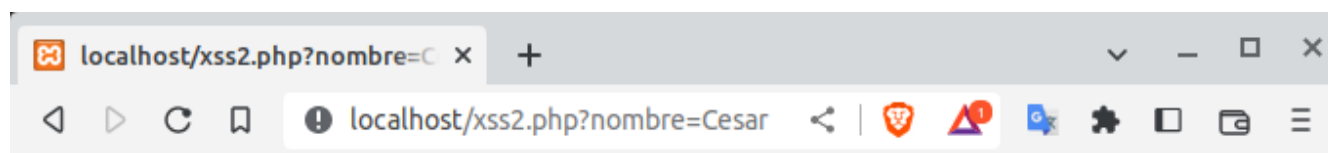
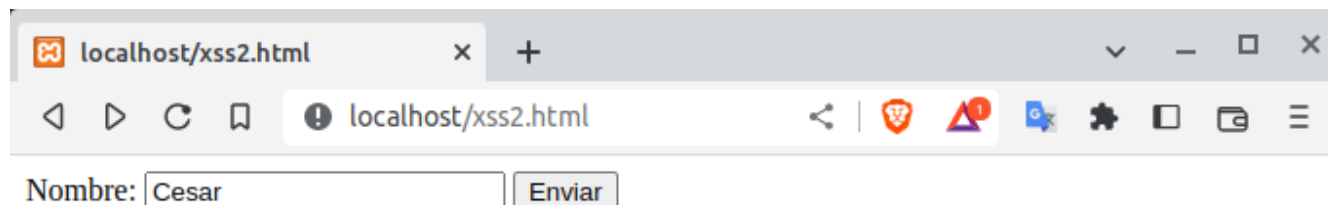
Si usted cree que esto es un error del servidor, por favor comuníquese al [administrador del portal](#).

Error 404

[localhost](#)

Apache/2.4.54 (Unix) OpenSSL/1.1.1s PHP/8.2.0 mod_perl/2.0.12 Perl/v5.34.1

Si no le cambiamos nada funciona correctamente:



Hola: Cesar

[Digitar aquí para descargar](#)

EJEMPLO 3:

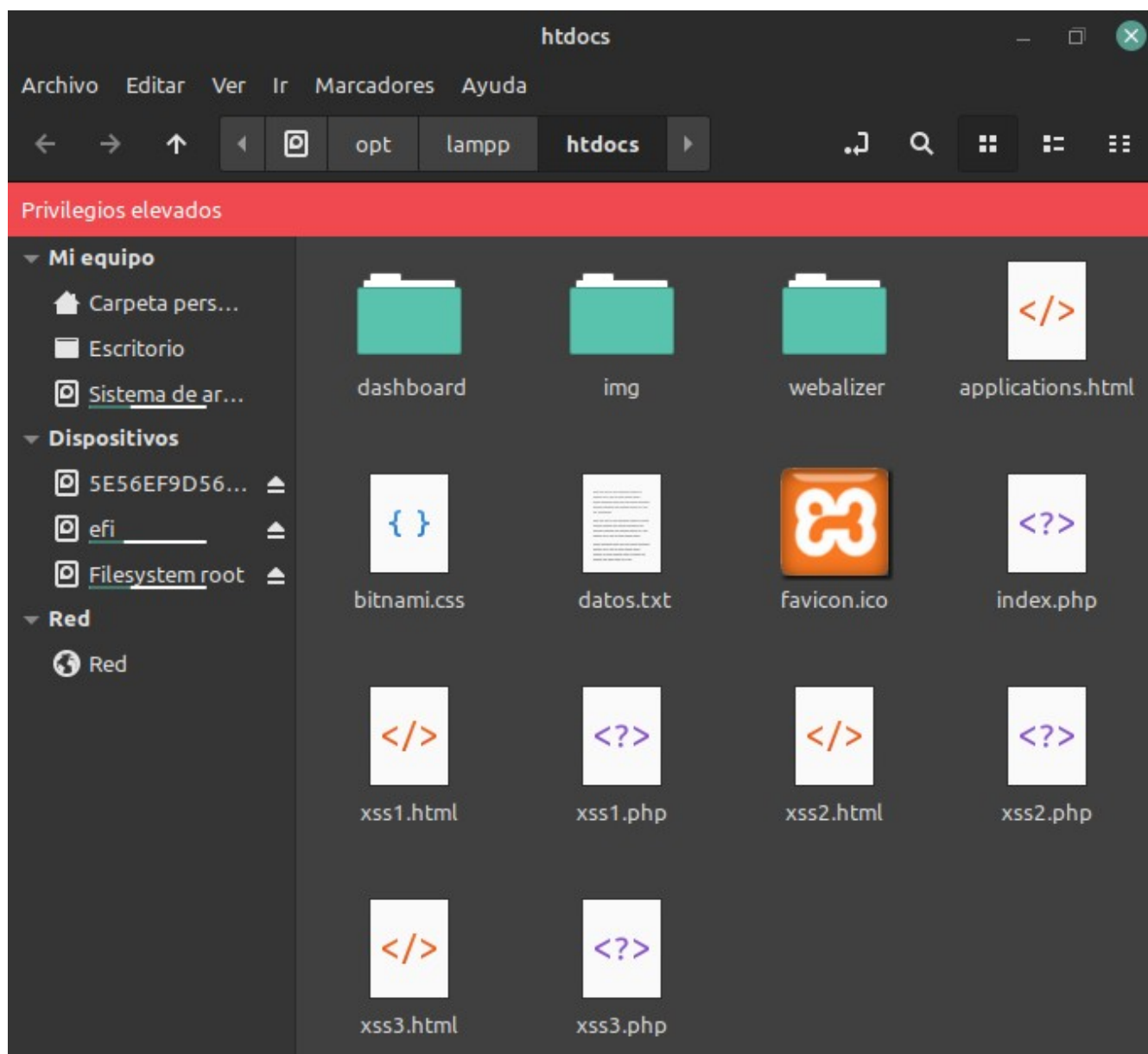
En un archivo de texto se guarda el código de un script enviado desde



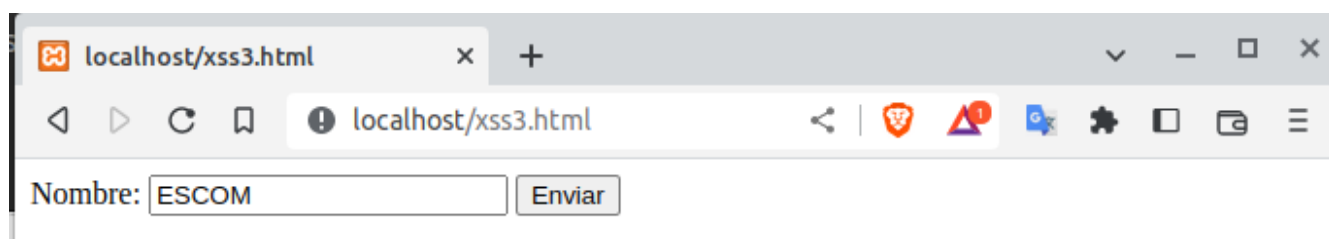
un navegador. El script se ejecutará cuando se lea el archivo. Primero, crear un archivo vacío de texto **datos.txt**. Enseguida, capturar los dos siguientes archivos en el bloc de notas:

```
~/Documentos/6.Semestre/Security/XSS/xss3.html - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
xss3.html x xss3.php x datos.txt x
1 <html>
2   <head>
3     <meta charset="utf-8"/>
4   </head>
5   <body>
6     <form action="xss3.php">
7       Nombre:
8       <input type="text" name="nombre"/>
9       <input type="submit" value="Enviar"/>
10    </form>
11  </body>
12 </html>
```

```
~/Documentos/6.Semestre/Security/XSS/xss3.php - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
xss3.html x xss3.php x datos.txt x
1 <?php
2   if (isset($_POST["nombre"])){
3     $fp = fopen('datos.txt', 'a');
4     fwrite($fp, $_POST["nombre"]);
5     fclose($fp);
6   }
7   ?>
8   <form action="" method="post">
9     <br>Enviar saludo:<input type="text" name="
10    nombre" size="50" value="" />
11    <input type="submit" />
12  </form>
13
14
15 1 Hola Instituto Politécnico Nacional !
16 2
```

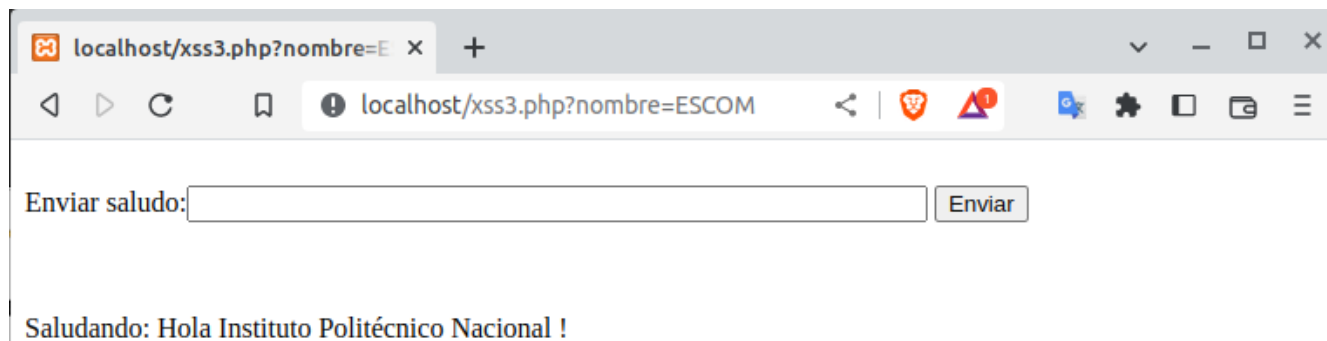


En el navegador se muestran las siguientes ventanas. En el campo de texto ingresar una cadena, por ejemplo ESCOM, y digitar en Enviar:



Al recibirse la cadena en el servidor, el archivo PHP en otro campo de texto la solicitud del ingreso de algún texto, por ejemplo Hola Instituto Politécnico Nacional !, el cual se guardará en el archivo datos.txt. Enseguida,

digitar en Enviar:



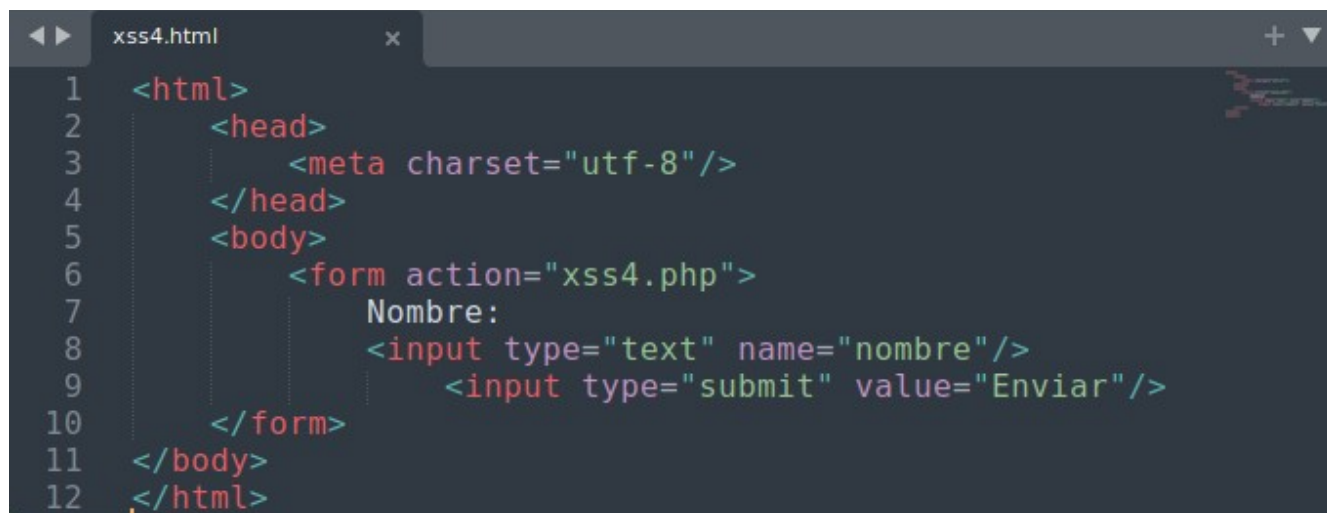
The screenshot shows a web browser window with the address bar displaying 'localhost/xss3.php?nombre=ESCOM'. Below the address bar, there is a form with the label 'Enviar saludo:' followed by a text input field and an 'Enviar' button. Below the form, the text 'Saludando: Hola Instituto Politécnico Nacional !' is displayed.

Cuando se recibe la cadena del saludo, se incluye el archivo y se envía automáticamente su contenido al navegador.

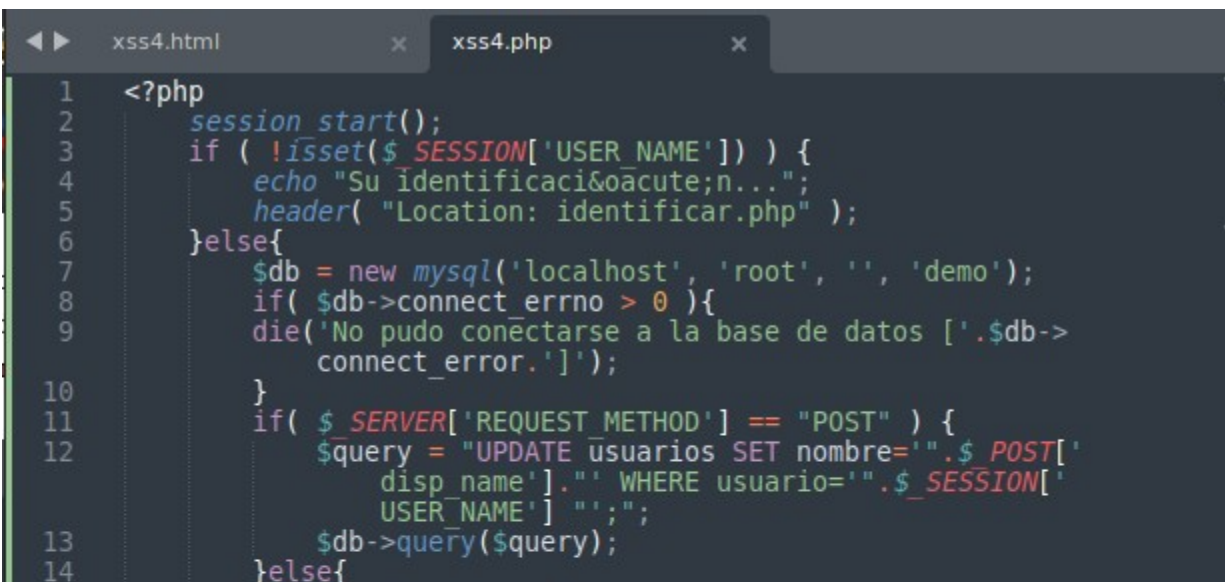
EJEMPLO 4:

La posibilidad de cambiar el nombre del usuario para que el administrador pueda abrir la lista de los usuarios.

NOTA: Crear la base de datos y el archivo xss4.html para completar la funcionalidad de este ejemplo.



```
1 <html>
2   <head>
3     <meta charset="utf-8"/>
4   </head>
5   <body>
6     <form action="xss4.php">
7       Nombre:
8       <input type="text" name="nombre"/>
9       <input type="submit" value="Enviar"/>
10    </form>
11  </body>
12 </html>
```



```
1 <?php
2   session_start();
3   if ( !isset($_SESSION['USER_NAME']) ) {
4     echo "Su identificación...";
5     header( "Location: identificar.php" );
6   }else{
7     $db = new mysqli('localhost', 'root', '', 'demo');
8     if( $db->connect_errno > 0 ){
9       die('No pudo conectarse a la base de datos ['. $db->
        connect_error. ']);
10    }
11    if( $_SERVER['REQUEST_METHOD'] == "POST" ) {
12      $query = "UPDATE usuarios SET nombre='".$_$_POST['
        disp_name']."' WHERE usuario='".$_$_SESSION['
        USER_NAME']."'";
13      $db->query($query);
14    }else{
```

```

15         if( strcmp($SESSION['USER NAME'], 'admin') == 0 ){
16             echo "Bienvenido administrador!<br /><br />";
17             echo "Usuarios:<br/>";
18             $query = "SELECT nombre FROM usuarios WHERE usuario
                        !='admin'";
19             if( !$result = $db->query($query) ){
20                 die('Error al actualizar ['. $db->error. ']');
21             }
22             while( $row = $result->fetch_assoc() ){
23                 echo $row[nombre]. '<br />';
24             }
25         }else{
26             echo "<form id='tags' name='tags'
                    action='home.php' method='POST'>";
27             echo "Actualizar el nombre:<input type='text'
                    id='disp_name' name='disp_name' value=''>";
28             echo "<input type='submit' value='Actualizar'>";
29         }
30     }
31 }
32 ?>
33

```

Ejercicio 2:

Diseñar una aplicación web que utilice la validación con el uso de la función `htmlspecialchars()`, por ejemplo:

```

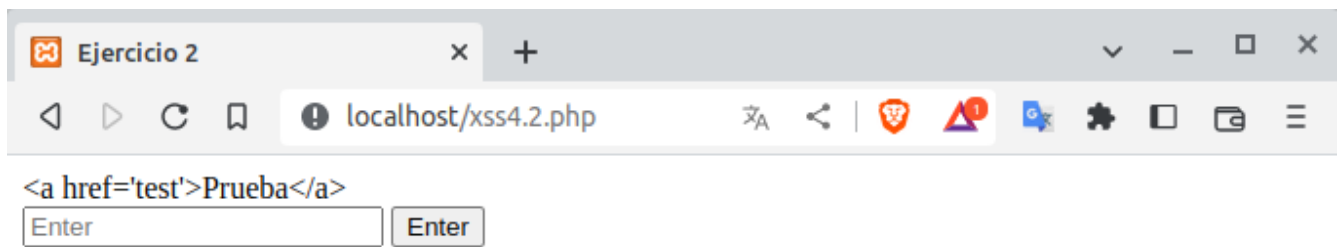
<?php
    $nuevo = htmlspecialchars("<a href='test'>Prueba</a>",
    ENT_QUOTES);
    echo $nuevo; // <a href=&#039;test&#039;&gt;Test&lt;/a&gt;
?>

```

```

xss4.2.php
1  <?php
2      // $nuevo = htmlspecialchars("<a href='test'>Prueba</a>",
3      ENT_QUOTES);
4      // echo $nuevo; // <a href=&#039;test&#039;&gt;Test&lt;/a&gt;
5
6      if(isset($_POST['btn'])){
7          // $input = htmlspecialchars($_POST['input'], ENT_QUOTES);
8          $input = htmlentities($_POST['input'], ENT_QUOTES);
9          echo $input;
10     }
11
12     ?>
13
14     <!DOCTYPE html>
15     <html lang="en">
16     <head>
17         <meta charset="UTF-8">
18         <meta http-equiv="X-UA-Compatible" content="IE=edge">
19         <meta name="viewport" content="width=device-width,
20         initial-scale=1.0">
21         <title>Ejercicio 2</title>
22     </head>
23     <body>
24         <form method="POST">
25             <input type="text" name="input" placeholder="Enter">
26             <button type="submit" name="btn">Enter</button>
27         </form>
28     </body>
29     </html>
30

```

htmlspecialchars

(PHP 3, PHP 4)

htmlspecialchars -- Convierte caracteres especiales a entidades HTML
Descripción

string htmlspecialchars (string cadena)

Ciertos caracteres tienen significados especiales en HTML, y deben ser representados por entidades HTML si se desea preservar su significado. Esta función devuelve una cadena con dichas conversiones realizadas. Esta función es útil para evitar que el texto entrado por el usuario contenga marcas HTML, como ocurre en aplicaciones de foros o libros de visita.

Actualmente, las traducciones hechas son:

'&' (ampersand) se convierte en '&';

'"' (doble comilla) se convierte en '"';

'<' (menor que) se convierte en '<';

'>' (mayor que) se convierte en '>';

Nótese que esta función no traduce nada más que lo mostrado más arriba. Para una traducción de entidades completa, vea htmlentities().

Esta función no va a traducir nada, si se requiere de una traducción se tendrá que recurrir a la función htmlentities().

Ejercicio 3:

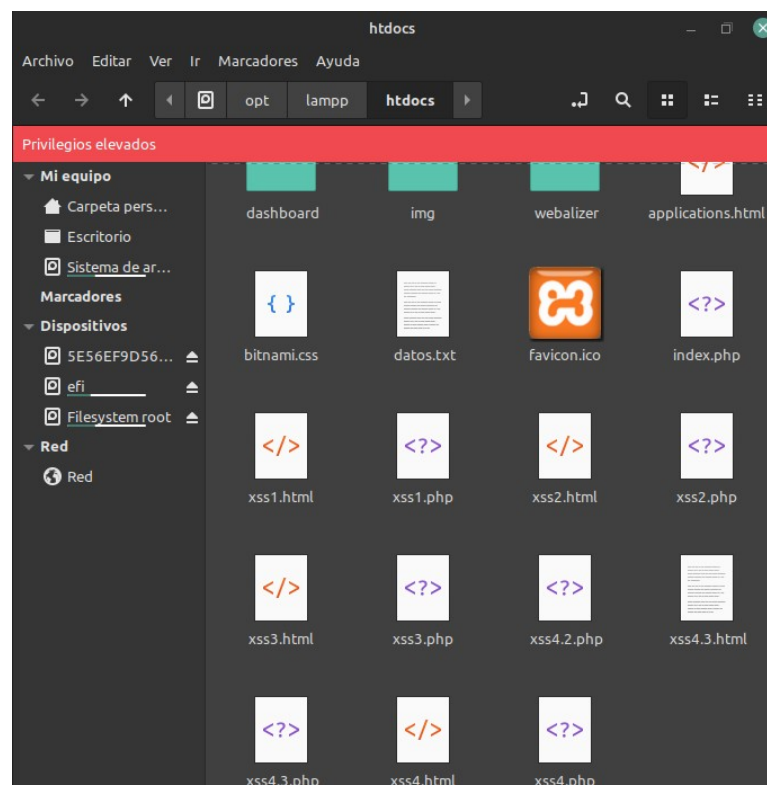
Diseñar una aplicación web que utilice la validación con el uso de la función filter_var(), por ejemplo:

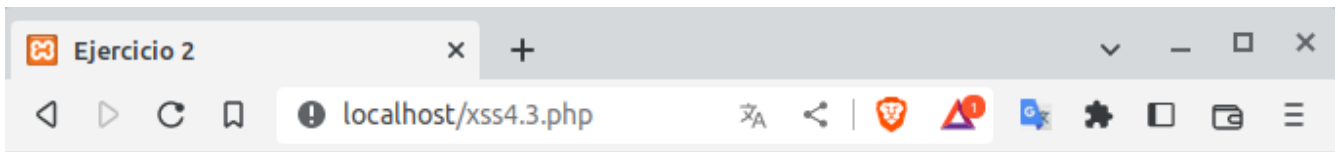
```
<?php
```

```
    var_dump( filter_var('ana@ejemplo.com', FILTER_VALIDATE_EMAIL) );  
    var_dump( filter_var('http://ejemplo.com', FILTER_VALIDATE_URL,  
        FILTER_FLAG_PATH_REQUIRED) );
```

```
?>
```

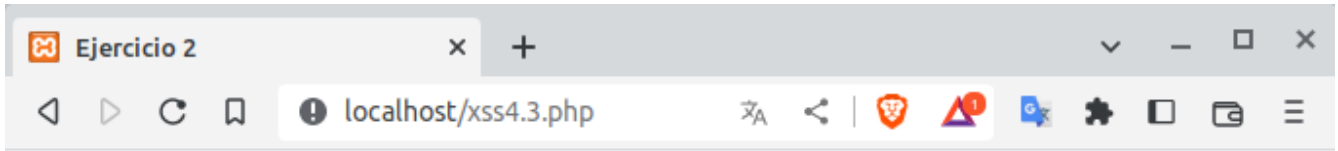
```
xss4.2.php xss4.3.php
1 <?php
2     $correo = $_POST['input'];
3     $url = $_POST['input2'];
4
5     if(isset($_POST['btn'])){
6         var_dump(filter_var($correo, FILTER_VALIDATE_EMAIL));
7     }
8     if(isset($_POST['btn2'])){
9         var_dump(filter_var($url, FILTER_VALIDATE_URL,
10             FILTER_FLAG_PATH_REQUIRED));
11     }
12
13 <!--DOCTYPE html-->
14 <html lang="en">
15 <head>
16     <meta charset="UTF-8">
17     <meta http-equiv="X-UA-Compatible" content="IE=edge">
18     <meta name="viewport" content="width=device-width,
19         initial-scale=1.0">
20     <title>Ejercicio 2</title>
21 </head>
22 <body>
23     <form method="POST">
24         <input type="text" name="input" placeholder="Correo">
25         <button type="submit" name="btn">Enter</button>
26
27         <input type="text" name="input2" placeholder="URL">
28         <button type="submit" name="btn2">Enter 2</button>
29     </form>
30 </body>
31 </html>
```





string(17) "correo@correo.com"

Esta función valida lo que se la pasa en el input, en este caso se pasó un parámetro de tipo correo en el primero, por lo cual nos indica con la función vardump(), el tipo de dato, su longitud y el contenido. De ser incorrecto el valor esperado, este solo arrojará un tipo de dato booleano siendo falso.



bool(false)