

Práctica 2.5

XSS Reflejado

Índice

Índice.....	2
1. Introduce el código Javascript necesario para mostrar el mensaje “Hola”.....	3
2. Introduce el código Javascript necesario para mostrar la cookie del usuario.....	4
3. Introduce el código Javascript necesario para redirigir la página a Google.es usando la función “window.location”.....	5
4. Sanear los datos para evitar ataques XSS.....	6

1. Introduce el código Javascript necesario para mostrar el mensaje “Hola”.

Código JavaScript:

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

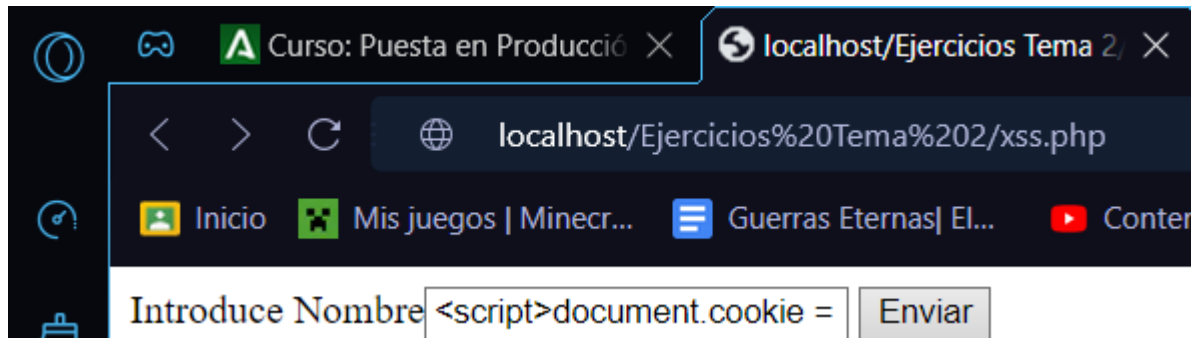
Introduce Nombre Hola



2. Introduce el código Javascript necesario para mostrar la cookie del usuario.

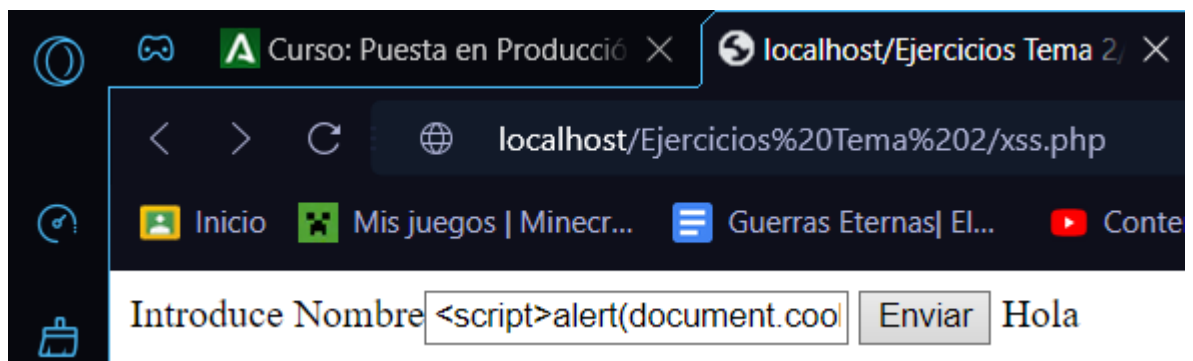
Código JavaScript:

```
<script>document.cookie = "username=John Doe";</script>
```



Código JavaScript:

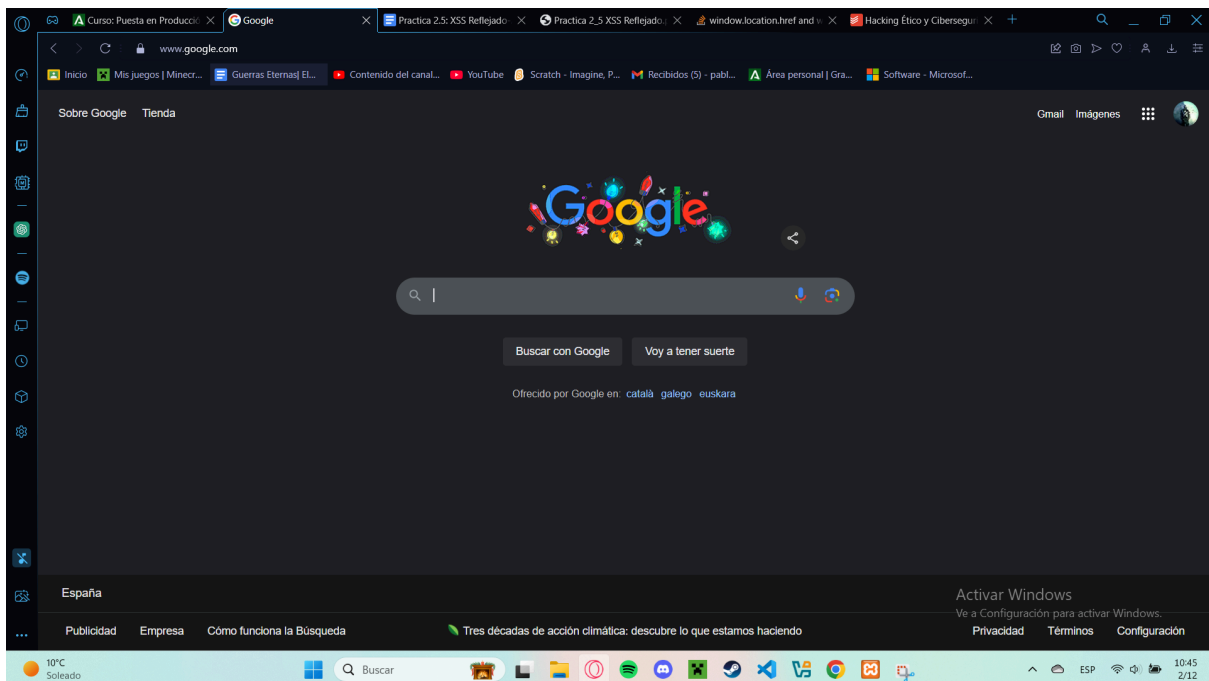
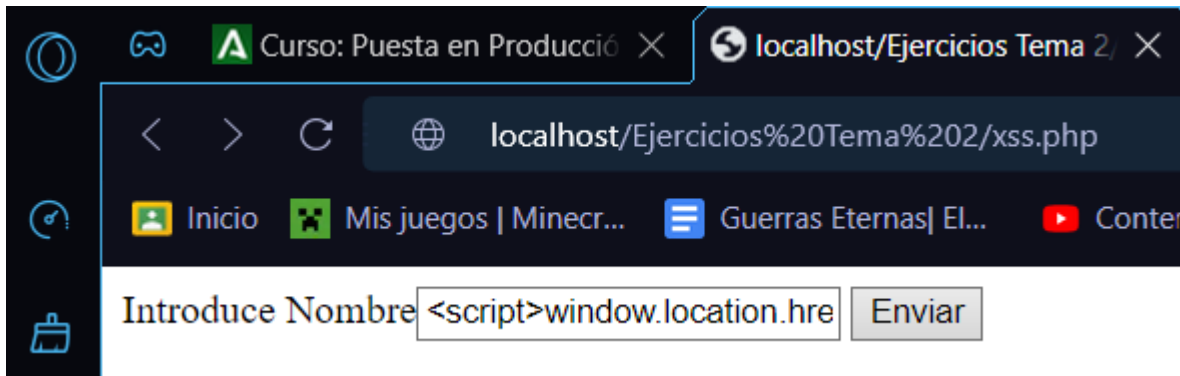
```
<script>alert(document.cookie)</script>
```



3. Introduce el código Javascript necesario para redirigir la página a Google.es usando la función "window.location".

Código JavaScript:

```
<script>window.location.href = 'http://www.google.com';</script>
```



4. Sanear los datos para evitar ataques XSS.

Una posible solución sería sanear los datos, de forma que ninguna cadena que empiece por `<script>` o similares, pueda ser ejecutada por la aplicación:

```
<?php
    if (isset($_GET["nombre"])){
        echo (str_starts_with(strtolower($_GET["nombre"]), '<script>'))
        ?
        "Se encontró la etiqueta script"
        :
        "Hola ".$_GET["nombre"];
    }
?>
```

Otra solución sería el indicar qué valor o valores son válidos en la aplicación, tal y como lo hace la siguiente función:

```
function validarNombre($valor) {
    $validacion=false;

    if (gettype($valor)=="string" &&
preg_match("/^([A-Za-zÑñÁáÉéÍíÓóÚú]+['\-]{0,1}[A-Za-zÑñÁáÉéÍíÓóÚú]+)(\s+
+([A-Za-zÑñÁáÉéÍíÓóÚú]+['\-]{0,1}[A-Za-zÑñÁáÉéÍíÓóÚú]+))*$/", $valor)){

        $validacion=true;
    }

    return $validacion;
}
```

En ambos códigos se hace un saneamiento de los datos, con la diferencia de que en el primero solo se impide el valor `<script>`, lo que significa que otros valores son válidos, y por lo tanto otro tipo de inyecciones.

Sin embargo, en el segundo código se indica el valor o valores válidos, por lo que cualquier otro valor que no cumpla con las condiciones no accederá a la aplicación. Impidiendo no solo un ataque XSS, sino otro tipo de ataques.

En conclusión, a la hora de sanear los datos de una aplicación las **Whitelist** brindan una mayor protección frente a posibles valores corruptos, que una **Blacklist**.