I. Cross-Site Scripting XSS

RK00

Universidad
Rey Juan Carlos

Universidad Rey Juan Carlos

ÍNDICE

Universidad Rey Juan Carlos

OBJETIVO DE LA CLASE

Universidad Rey Juan Carlos

Introducción a XSS

¿Qué es el XSS?

Es una vulnerabilidad que permite inyectar código JavaScript en páginas WEB

¿Para que sirve el XSS?

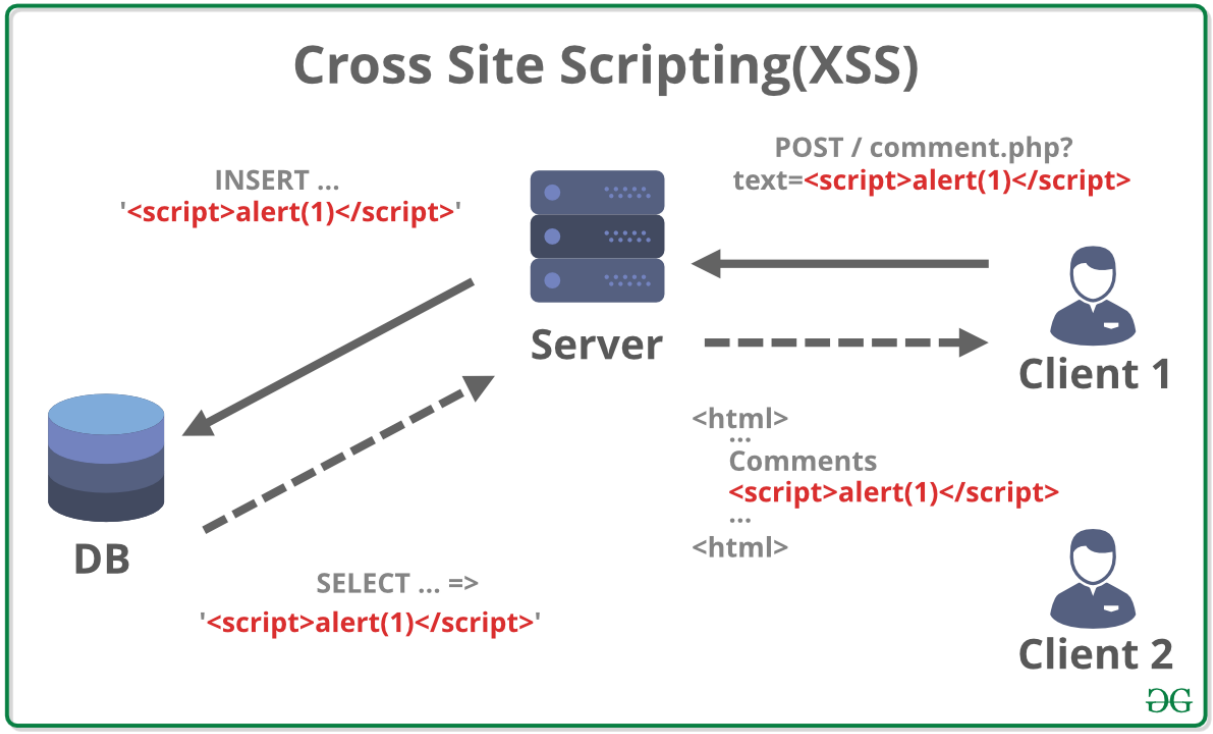
Robar cookies y suplantar, redirigir, mostrar contenido falso

Ejemplo básico

<script>alert(1)</script>

XSS

¿Cómo funciona?



TIPOS DE XSSS

REFLECTED XSS

STORED XSS

DOM-BASED XSS

Universidad Rey Juan Carlos

Reflected XSS

- El script viene de la petición actual
- Se ejecuta cuando un usuario accede a una url manipulada

Es la forma más simple de XSS. Ocurre cuando una aplicación recibe datos de una petición HTTP y lo devuelve directamente en la respuesta, sin sanearlo

Ejemplo: https://insecure-website.com/status?message= -> https://insecure-website.com/status?message=All+is+well

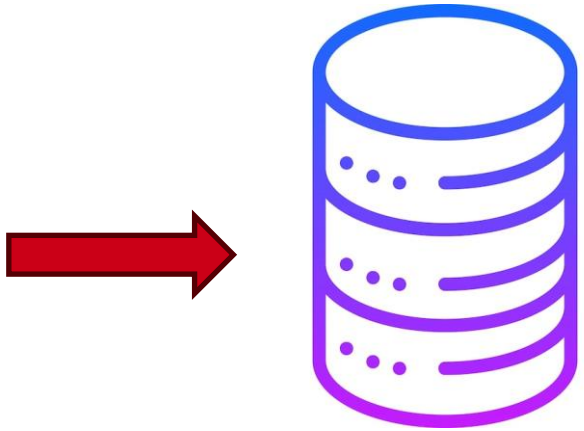
Tenemos de respuesta en la web lo siguiente: Status: All is well

Si modificamos el payload: https://insecure-website.com/status?message=<script>alert(1)</script>

La respuesta será: Status: <script>Malicioso</script>

Stored XSS

El script se guarda en la base de datos



Si el servidor renderiza la pagina y utiliza la base de datos para cargar contenido, esto afectara a todos los usuarios que quieran cargar la pagina. Por ello es el MAS PERSISTENTE Y PELIGROSO.

Aplicación recibe datos -> Los almacena -> Lo muestra más tarde en la respuesta HTTP

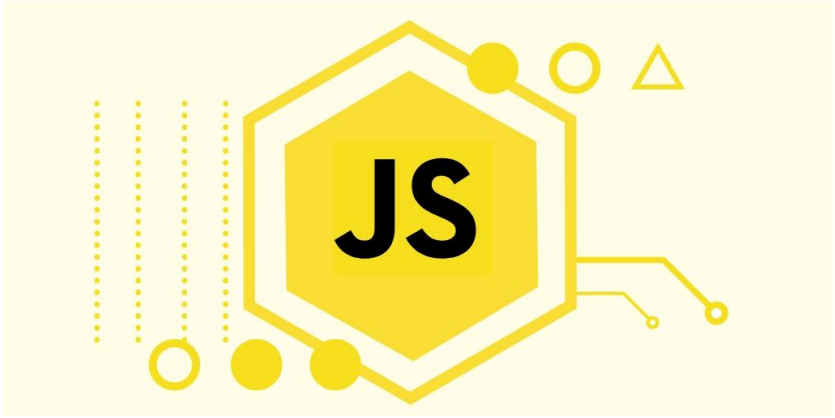
Ejemplo más típico comentarios de una web: Hola, esto es un comentario , si modificamos esto por un payload típico: <script>alert(1)</script>

Cuando un usuario visualice el contenido de la pagina, el script se ejecutará en su navegador

Universidad Rey Juan Carlos

DOM-BASED XSS

- La vulnerabilidad está en el código JS del cliente y no del servidor
- Requiere analizar el código fuente para detectarlo



Código JavaScript es el encargado de procesar la petición, normalmente escribiéndolo en el DOM sin validación.

Ejemplo: var search = document.getElementbyId('search').value;

Var result = document.getElementbyId('result');

Results.innerHTML = 'you search:' + search;

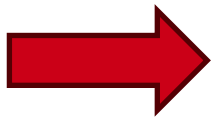
DOM-BASED XSS

Ejemplos comunes de JS vulnerable, para analizar en el código fuente

- window.location.search (parámetros de URL)
- window.location.hash (fragmento de URL)

```
<script>
  function doSearchQuery(query) {
    document.getElementById('searchMessage').innerHTML = query;
  }
  var query = (new URLSearchParams(window.location.search)).get('search');
  if(query) {
    doSearchQuery(query);
  }
  </script>
```

Ejemplo de código js que procesa el parámetro de búsqueda



Si controlamos el parámetro de búsqueda:

```
GET /?search='
```

Podremos ejecutar un payload que ejecute js
<img src=1 onerror=alert(1)>

Universidad Rey Juan Carlos

DOM-BASED XSS

Hay muchas formas de explotar un DOM-based XSS

Cual es la clave para encontrar estas vulnerabilidades

¡¡¡ANALIZAR LOS FICHEROS JS DE LA PAGINA!!!

Universidad Rey Juan Carlos

Angular JS

Si una pagina utiliza Angular JS, será posible ejecutar JS sin usar etiquetas.
Para comprobar si se esta usando Angular JS -> miramos si en HTML se está usando el atributo ng-app.

Si esto sucede, el HTML se procesará por angular y este ejecutar JS.

Con las llaves podemos ejecutar JS: {{1+1}} o Podemos ejecutar un alert de la siguiente forma: $on.constructor('alert(1)')()

XSS context

Cuando estamos probando un reflected o un stored xss, debemos tener en cuenta lo siguiente:

- La localización donde aparece el input en el response
- Cualquier validación del input, o cualquier otro proceso que haga la página con los datos

XSS between HTML tags
Cuando el contexto del XSS es texto entre tags de HTML, se necesita introducir nuevas tags de HTML para poder ejecutar JS.

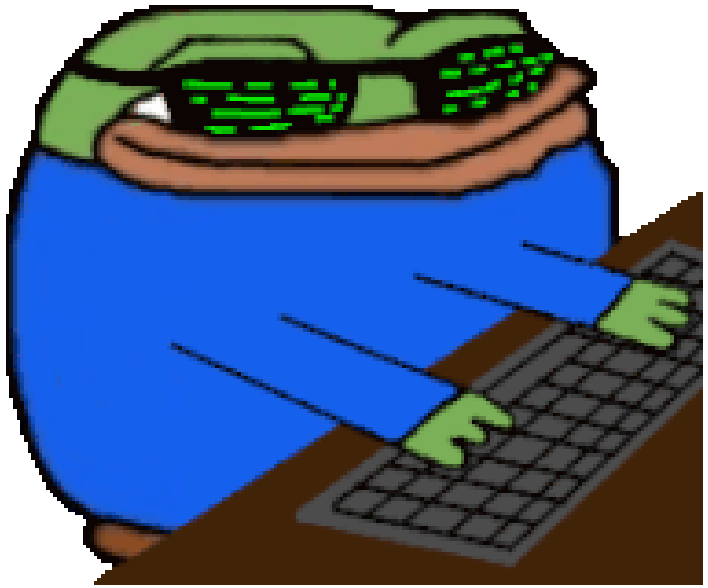

```
<script>
...
var input = 'controllable data here';
...
</script>
```

Se tiene que cerrar la tag script, en el contexto en el que nos encontramos

```
</script><img src=1 onerror=alert(document.domain)>
```

Universidad Rey Juan Carlos

Demostración practica

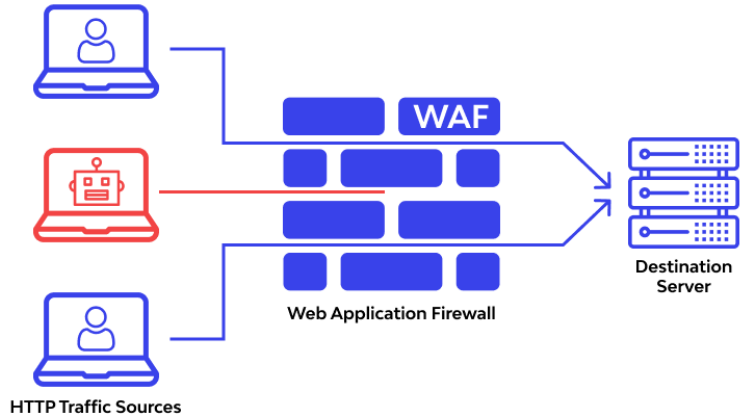


Lab: Reflected XSS into HTML context with nothing encoded | Web Security Academy
Lab: Stored XSS into HTML context with nothing encoded | Web Security Academy
Lab: DOM XSS in document.write sink using source location.search | Web Security Academy

Técnicas de bypass

Escapar de cadenas de texto en JS

Bypass de WAF



Escapar de cadenas de texto en JS

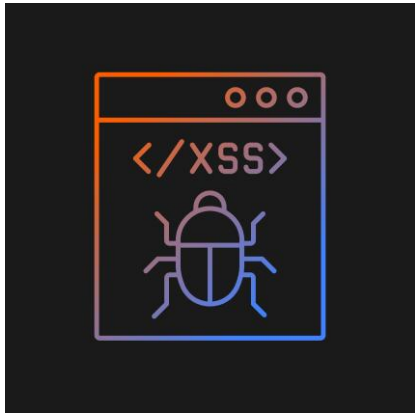
Cuando el contexto de XSS este dentro de cadenas de texto, es posible romper la logica de estas y ejecutar JS.

Ejemplo:

```
var searchTerms = ' ';
                document.write('<img src="/resources/images/tracker.gif?searchTerms='+encodeURIComponent(searchTerms)+'">');
```

Nosotros al controlar la variable searchTerms, que al final es el input que nosotros metemos en un buscador. Dentro de ese contexto, si conseguimos que la función: encodeURIComponent, tenga un input como –alert(), conseguiremos explotar XSS:
Payload : '-alert(1)//

Explicación del payload
‘: cerramos la comilla de la izquierda -> var searchTerms = ‘’ ‘;
-alert(1): llama a la función alert
//: comenta todo lo demás para que el rsto de código no de errores de sintaxis -> //‘;



Con / puedes escapar caracteres

Universidad Rey Juan Carlos

Bypass de WAF

Metodología:

Fuzzing de tags: <§§> -> En el intuder de Burpsuite -> Usamos el diccionario de tags -> Cross-Site Scripting (XSS) Cheat Sheet - 2025 Edition | Web Security Academy

Fuzzing de eventos: comprobamos que eventos no están baneadas para esa tag-> <svg><animatetransform%20§§=1>
Usamos el diccionario de all events -> Cross-Site Scripting (XSS) Cheat Sheet - 2025 Edition | Web Security Academy

Esto dará lugar a encontrar un XSS que el WAF no pille. Ejemplo: <svg><animatetransform onbegin=alert(1)>

Universidad Rey Juan Carlos

Material recomendado

Cheat sheets

- PortSwigger XSS Cheat Sheet
- OWASP XSS Filter Evasion

Herramientas

- Burp Suite
- Browser DevTools para debug

Laboratorios

- PortSwigger Web Security Academy

Universidad Rey Juan Carlos

CHALLENGE

2. File Upload

Astharot15

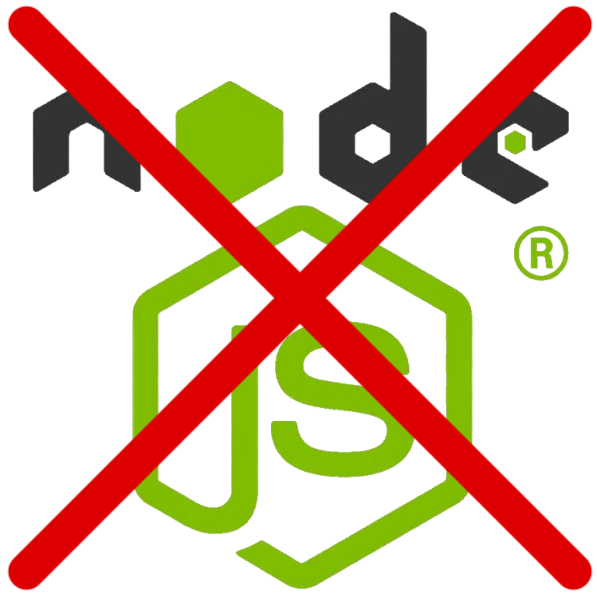
Universidad
Rey Juan Carlos

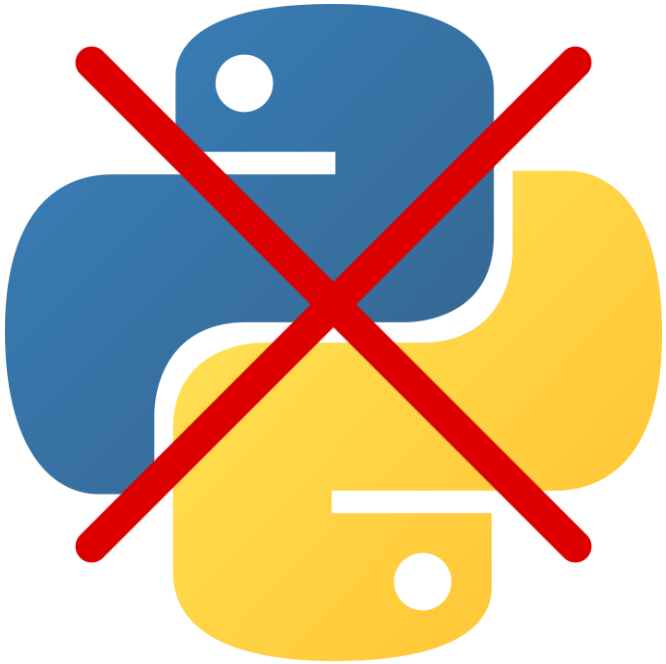
Universidad Rey Juan Carlos

ÍNDICE

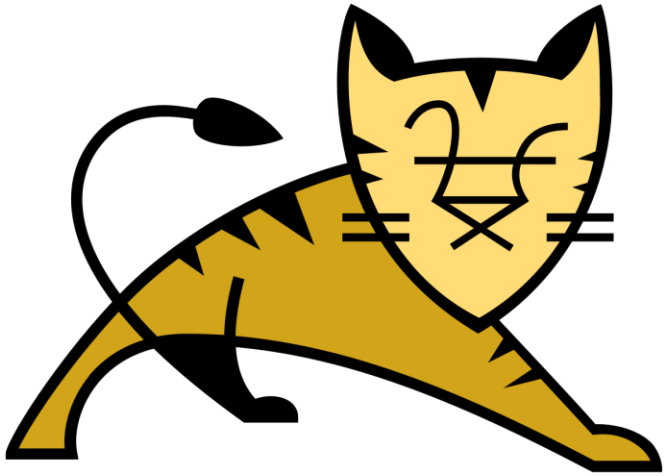
1. Arquitectura Web
2. File Upload Feature
3. Webshells
4. Validaciones y bypass

Arquitectura Web

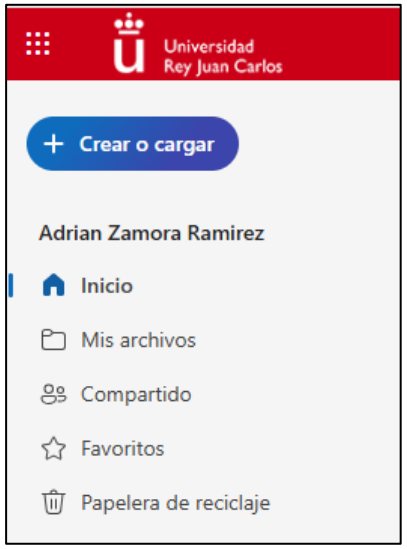


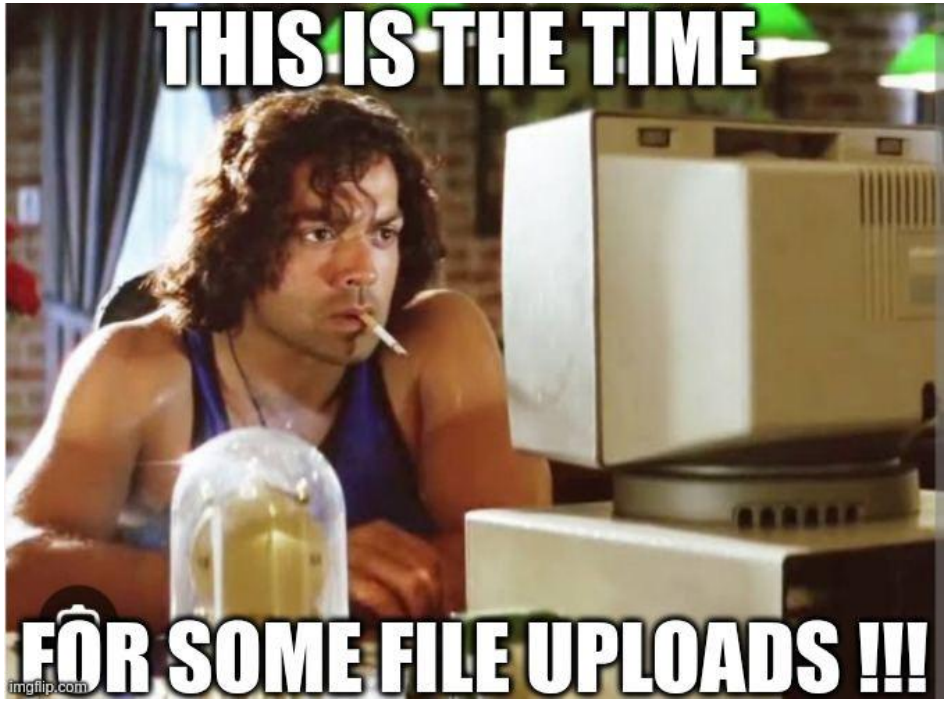


Arquitectura Web



File Upload Feature





Universidad Rey Juan Carlos

Webshells

Requisitos

- Subida de archivos arbitraria
- Acceso a la ubicación del archivo
- Tecnología corriendo por detrás

Problemas

- Nombres aleatorios
- No se renderiza el archivo
- Blacklists / Whitelists

Universidad Rey Juan Carlos

Validaciones y Bypass

Front-End

- Se valida en el html
- Usar BurpSuite
- Inspect

Validaciones y Bypass

Extensión

- Contiene la extensión
- Termina por la extensión
- Varias extensiones
- Null Byte
- Regex insegura

Universidad Rey Juan Carlos

Validaciones y Bypass

Content-Type

- Se añade de forma automática
- Se puede modificar en la petición

Universidad Rey Juan Carlos

Validaciones y Bypass

Magic Bytes

- Validación por magic bytes (vistos en forense)
- Suele validarse con funciones mimetype

Universidad Rey Juan Carlos

CHALLENGE

I. Modulo III WEB

Alumnos Ciberseguridad

Universidad
Rey Juan Carlos