

Análisis y Securización de las principales vulnerabilidades presentes en aplicaciones web utilizando OWASP

Proyecto de final de curso realizado por Benjamin Gordo
2º Curso de Administración de Sistemas Informáticos en Red
Salesianos Centro Don Bosco de Villamuriel de Cerrato

Contenido

Índice de ilustraciones	5
Introducción	8
Objetivos	9
Planificación	10
1. Tecnologías utilizadas	11
1.1 Docker	11
¿Qué es Docker?	11
Ventajas del uso de los contenedores Docker	11
Instalación de Docker Engine	12
Instalar Docker Compose	14
1.2 Node JS Y Node Goat	15
¿Qué es Node JS?	15
¿Qué es Node Goat?	15
Instalación Node Goat	16
2. Marco teórico sobre OWASP	18
¿Qué es OWASP y su Top 10?	18
Diferencias entre OWASP top ten 2010 al 2013	20
Diferencias entre OWASP top ten 2013 al 2017	22
¿OWASP 2021?	23
3. Explicación y explotación de las vulnerabilidades	24
A1 - Inyección de Código	24
Cómo prevenirlo	24
Vulnerabilidades presentes en Node Goat	25
A1.1 - Inyección para acceder a archivos	25
A1.2 - Ataque DOS	27
A1.3 - Inyección de comandos	28
A1.4 – Usuarios de la aplicación	30
A2 - Pérdida de autenticación y gestión de sesiones.	31
Cómo prevenirlo	31
Vulnerabilidad presente en Node Goat	32
Gestión de sesiones.	32

Autenticación.	33
A3 - Secuencia de comandos en sitios cruzados (XSS)	37
Cómo prevenirlo	37
Explotación de la vulnerabilidad en Node Goat	38
A4 - Referencia directa a objetos inseguros (DOR)	41
Cómo prevenirlo	41
Explotación de la vulnerabilidad en Node Goat	41
A5 - Configuración de seguridad incorrecta	42
¿Cómo prevenirlo?	42
Explotación de la vulnerabilidad en Node Goat	43
A6 – Exposición de datos sensibles	45
Cómo prevenirlo	45
Explotación de la vulnerabilidad en Node Goat	45
A7 – Ausencia de control de acceso a las funciones.	47
Cómo prevenirlo	47
Explotación de la vulnerabilidad en Node Goat	47
A8 – Falsificación de peticiones en sitios cruzados (CSRF).	49
Cómo prevenirlo.	49
Explotación de la vulnerabilidad en Node Goat	50
A9 – Uso de componentes con vulnerabilidades conocidas	52
Cómo prevenirlo.	52
Explotación de la vulnerabilidad en Node Goat	52
A10 – Redirecciones y reenvíos no validados	54
Cómo prevenirlo.	54
Explotación de la vulnerabilidad en Node Goat	54
4. Securitización de vulnerabilidades	55
A1.1 - Inyección para acceder a archivos	55
A3 - Secuencia de comandos en sitios cruzados (XSS)	56
A4 - Referencia directa a objetos inseguros (DOR)	57
A6 – Exposición de datos sensibles	58
A8 – Falsificación de peticiones en sitios cruzados (CSRF).	60
A9 - Uso de componentes con vulnerabilidades conocidas	62
A10 – Redirecciones y reenvíos no validados	63

Índice de ilustraciones

Ilustración 1	Tiempo de dedicación	10
Ilustración 2	Actualizar repositorios	12
Ilustración 3	Permitir el uso de HTTPS en repositorios	12
Ilustración 4	Modificación del repositorio de Docker	13
Ilustración 5	Repositorio ya corregido de Docker	13
Ilustración 6	Actualización completa de los repositorios de Kali Linux	13
Ilustración 7	Instalación de Docker	13
Ilustración 8	Uso de la imagen Hello-World	13
Ilustración 9	Ejemplo de archivo YAML	14
Ilustración 10	Dar permisos de ejecución al archivo Docker-compose	14
Ilustración 11	Versión de Docker-compose ya instalado	14
Ilustración 12	Secciones de NodeGoat	15
Ilustración 13	Clonación del repositorio de GitHub	16
Ilustración 14	Configuración e instalación de Node Goat	16
Ilustración 15	Poner en marcha NodeGoat	16
Ilustración 16	Login de NodeGoat	17
Ilustración 17	Top 10 OWASP 2007-2017	18
Ilustración 18	Interfaz de OWASP WebGoat	19
Ilustración 19	Interfaz OWASP ZAP	19
Ilustración 20	Actualización del año 2010 al 2013	21
Ilustración 21	Actualización del año 2013 al 2017	23
Ilustración 22	Inyección de código malicioso 1	25
Ilustración 23	Archivos del directorio de la aplicación web	25
Ilustración 24	Inyección de código malicioso 2	25
Ilustración 25	Archivo /etc/passwd de la aplicación web	26
Ilustración 26	Inyección de código malicioso 3	26
Ilustración 27	Mensaje de error	26
Ilustración 28	Directorio home de los usuarios	27
Ilustración 29	Inyección de código por URL	27
Ilustración 30	Uso de recursos de la imagen	27
Ilustración 31	Crasheo de la aplicación	27
Ilustración 32	Sección contributions	28
Ilustración 33	Análisis de la cabecera de petición	28
Ilustración 34	Enviar a Repeater	28
Ilustración 35	Cabecera de petición	29
Ilustración 36	Código malicioso a inyectar	29
Ilustración 37	Análisis de tráfico de red	29
Ilustración 38	inyección a través de URL	30
Ilustración 39	Resultado de la inyección	30
Ilustración 40	Acceso al perfil de la víctima	33
Ilustración 41	Acceder al BurpSuite	33
Ilustración 42	Prueba de login con el usuario admin	33
Ilustración 43	Análisis de la petición de login	34
Ilustración 44	Enviar cabecera a Intruder	34
Ilustración 45	Cabecera en Intruder	34
Ilustración 46	Añadir diccionario	35
Ilustración 47	Comenzar el ataque	35
Ilustración 48	Análisis del tamaño de las cabeceras de respuesta	35
Ilustración 49	Cabecera de respuesta de la contraseña Admin_123	36
Ilustración 50	Página de inicio del administrador	36

Ilustración 51 Sección Profile para actualizar el perfil del usuario	38
Ilustración 52 Código malicioso para saber la cookie de sesión	38
Ilustración 53 Cookie de sesión del usuario	39
Ilustración 54 Código malicioso para redireccionar a otro sitio web	39
Ilustración 55 Página de inicio del administrador	39
Ilustración 56 Acceso del usuario como administrador	40
Ilustración 57 Sitio web a la que será redireccionado el administrador	40
Ilustración 58 URL de la aplicación web a modificar	41
Ilustración 59 Información del id de usuario 2	41
Ilustración 60 Información del id de usuario 3	41
Ilustración 61 Cabecera de petición al servidor	43
Ilustración 62 Id de sesión visible	43
Ilustración 63 Solicitud de recurso no existente	43
Ilustración 64 Respuesta al solicitar el recurso no disponible	44
Ilustración 65 Versión del servidor web	44
Ilustración 66 Directorio de la aplicación web	44
Ilustración 67 Login de inicio con user1	45
Ilustración 68 Filtro HTTP	46
Ilustración 69 Filtro con login para localizar paquete	46
Ilustración 70 Credenciales del usuario	46
Ilustración 71 Id de sesión del usuario	46
Ilustración 72 Inicio de DirBuster	47
Ilustración 73 Configuración de DirBuster	47
Ilustración 74 Directorios obtenidos por el ataque	48
Ilustración 75 Modificación de URL	48
Ilustración 76 Acceso al directorio benefits	48
Ilustración 77 Perfil de la víctima	50
Ilustración 78 Aplicación web con el código malicioso	50
Ilustración 79 Servidor web malicioso	50
Ilustración 80 Información de la víctima ya modificada	51
Ilustración 81 Archivo package.json	52
Ilustración 82 Análisis de ZAP	53
Ilustración 83 Vulnerabilidad de Bootstrap versión anticuada	53
Ilustración 84 Vulnerabilidad de JQuery versión anticuada	53
Ilustración 85 Redireccionamiento que realiza la aplicación por defecto	54
Ilustración 86 URL modificada	54
Ilustración 87 Aplicación usando la función eval	55
Ilustración 88 Aplicación usando la función parseInt	55
Ilustración 89 Inyección de código	55
Ilustración 90 Error al inyectar código no válido	55
Ilustración 91 Inicio de la API Swig	56
Ilustración 92 Activar auto-escapado	56
Ilustración 93 Inyección de código XSS	56
Ilustración 94 Inyección de código sin éxito	56
Ilustración 95 Acceso a través de URL	57
Ilustración 96 Acceso a través de la autenticación del usuario	57
Ilustración 97 Datos usuario autenticado	57
Ilustración 98 Intento de explotación 1	57
Ilustración 99 Intento de explotación	57
Ilustración 100 Creación de certificado ssl	58
Ilustración 101 levantar servidor HTTPS	58
Ilustración 102 Dirección de la aplicación web	58
Ilustración 103 Candado de seguridad https	58
Ilustración 104 Resultado del uso de filtros	59

Ilustración 105 Archivo package.json que contiene las dependencias	60
Ilustración 106 Código de creación de tokens	60
Ilustración 107 Campo oculto en formulario	60
Ilustración 108 Aplicación web maliciosa	60
Ilustración 109 Error de generación de token	61
Ilustración 110 Campo oculto en formulario	61
Ilustración 111 Librerías vulnerables	62
Ilustración 112 Ruta de librerías	62
Ilustración 113 Análisis con vulnerabilidad corregida	62
Ilustración 114 Direccion vulnerable	63
Ilustración 115 Vulnerabilidad resuelta	63
Ilustración 116 Dirección de la página web	63

Introducción

Con el uso masivo de internet y de las aplicaciones web por millones de usuarios en todo el mundo, donde cada usuario contiene cientos de datos sensibles (tarjetas de crédito, teléfonos, fotos, mensajes, correos electrónicos , etc) una de las principales preocupaciones tanto de las propias empresas como de los usuarios (no de todos) es la seguridad al usar las aplicaciones web y todo lo que las rodea para así, poder mantener todos estos datos privados alejados de aquellos usuarios que no tengan permiso. Para evitar estas acciones hacia los usuarios como el robo, modificación de dichos datos, etc se deben de realizar diferentes técnicas y acciones en las aplicaciones para así evitar la intrusión y las acciones de los usuarios “maliciosos”.

Por todo esto, este proyecto consiste en realizar un análisis del OWASP Top Ten el cual es utilizado por desarrolladores para aplicar políticas seguras y técnicas de seguridad en las aplicaciones web. Todo este análisis de vulnerabilidades se llevará a cabo en la aplicación web NodeGoat la cual está creada con unos de los lenguajes más utilizados en el Back-End, NodeJS. Esta aplicación contiene todas las vulnerabilidades descritas en el OWASP Top Ten del 2013. Para evitar problemas de instalación como falta de dependencias, NodeGoat se instalará en Docker por las ventajas de su uso.

Las pruebas y la máquina anfitriona donde está instalado todo lo necesario estará en una máquina virtual con la distro Kali Linux especializada en la seguridad informática. En la realización de la explotación se utilizará documentación de la propia aplicación ya que contiene un tutorial para principiantes y de varias páginas web para realizar el análisis. En este análisis se utilizarán diferentes herramientas de seguridad como BurpSuite, OWASP ZAP, etc y diferentes técnicas.

Una vez se haya realizado la explotación de todas las vulnerabilidades, explicado el funcionamiento y el cómo corregirlas, se realizará la corrección de algunas de las vulnerabilidades específicas de la aplicación mediante APIs, modificación de código, etc.

El motivo de la realización de proyecto se basa en el propio interés en el ámbito de la seguridad informática obtenido antes de comenzar el grado superior y apoyado en la especialización del profesorado en este campo, también por comenzar este ámbito tecnológico que se encuentra actualmente en auge y tener unos conocimientos diferentes a los demás graduados que hayan realizado el grado superior de Administración de Sistemas Informáticos en Red y así mejorar mi perfil laboral para lograr mayor interés en las empresas.

El proyecto se divide en diferentes partes:

- **Punto 1. Tecnologías utilizadas.** En este primer capítulo se explicará aquellas tecnologías utilizadas para la realización de este proyecto, en las que se encuentra Docker y NodeGoat.
- **Punto 2. OWASP.** Se explicará la organización OWASP, sus funciones y las diferencias de OWASP Top Ten.
- **Punto 3. Explotación de vulnerabilidades.** Una vez explicada la organización OWASP y su top ten, se explicará las vulnerabilidades y se realizará su explotación.
- **Punto 4. Securitización de las vulnerabilidades.** Para completar este proyecto, se realizará la corrección de algunas de las vulnerabilidades explotadas en el punto anterior.
- **Punto 5. Referencias.** Se expondrán todas aquellas fuentes de información utilizadas en este proyecto.

Herramientas utilizadas:

- **Burp Suite:** Es una herramienta utilizada en la realización de pruebas de seguridad para aplicaciones web. Contiene un servidor proxy que permite inspeccionar y modificar el tráfico haciendo intermediario entre la aplicación y el usuario.
- **OWASP ZAP:** Framework desarrollado en Java que analiza el tráfico HTTP/HTTPS. Esta herramienta puede ser usada tanto por desarrolladores o analistas de seguridad, ya que también permite escanear las vulnerabilidades de una aplicación web y comprobar cómo de segura es.
- **Wireshark:** es un analizador de protocolos utilizado para realizar análisis y solucionar problemas en redes de comunicaciones.
- **DirBuster:** Es una aplicación Java diseñada para obtener a través de la fuerza bruta nombres de directorios y archivos en aplicación web.

Objetivos

Al realizar este análisis se espera cumplir diferentes objetivos, estos se dividen en:

Principales:

- Realización del análisis completo.
- Saber por qué ocurren las vulnerabilidades, explotación y corregirlas.

Secundarios:

- Tener un primer contacto con Docker.
- Diferenciar perfil académico y laboral.

Planificación

31 de marzo – Haber realizado las vulnerabilidades en la aplicación web:

- A ver realizado el top ten de vulnerabilidades de en la aplicación y tener un primer conocimiento sobre las mismas

6 de abril – explicación de Docker, OWASP top ten 2013 y 2017 y NodeJS:

- Buscar información de cómo funciona Docker y como se instala
- Buscar información de la organización OWASP y de sus tops ten más actuales
- Información de que es NodeJS y de la aplicación Node Goat

31 de abril – Explicación de la gran mayoría de las vulnerabilidades:

- Tener el Top 10 de vulnerabilidades explicado en el proyecto con sus correspondientes capturas, información de estas y como prevenirlas. También tener mayor conocimiento sobre su funcionamiento.

16 de mayo – Corrección de vulnerabilidades:

- Corrección de vulnerabilidades para mejorar la seguridad de la aplicación y entender completamente por qué ocurren.

23 de mayo – Finalización de memoria:

- Completar la memoria y maquetación.

Porcentajes de dedicación

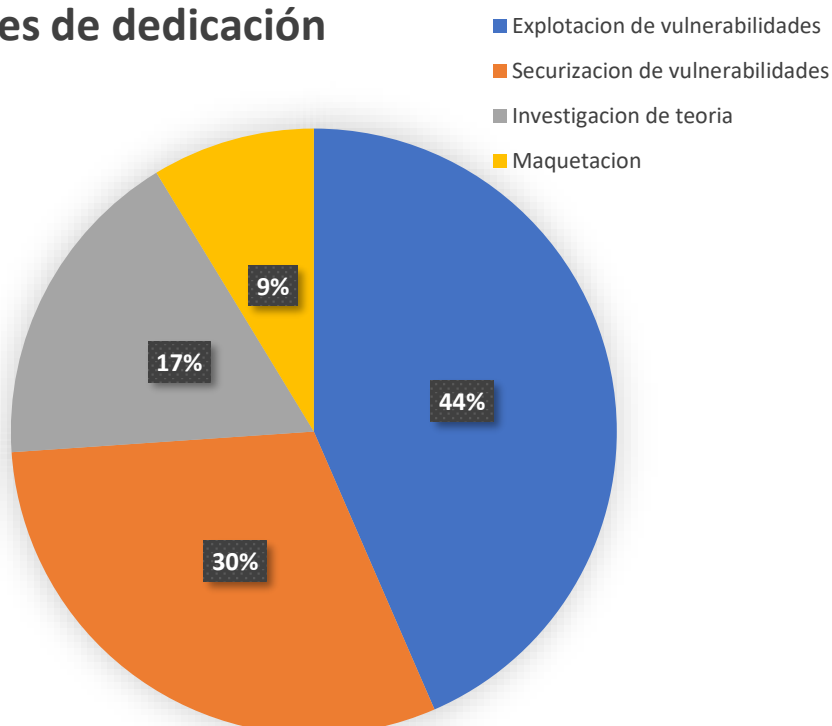


Ilustración 1 Tiempo de dedicación

1. Tecnologías utilizadas

1.1 Docker

A continuación, se explicará qué es Docker, algunas de sus características y la instalación de este.

¿Qué es Docker?

Docker es contenedor de software, en cada contenedor se “empaqueta” un software en el cual se incluye todo lo necesario para su correcto funcionamiento, incluyendo bibliotecas, herramientas de sistema, código y tiempo de ejecución.

Para entender mejor la definición de Docker, se podría usar la analogía de un carguero; el carguero es el propio sistema donde corre Docker y los containers que transporta contiene los diferentes softwares.

*“**Docker** es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos” Wikipedia.*

Al usar Docker al desarrollador se le facilita la tarea, ya que puede mover ese contenedor a otra máquina sin ningún problema y sin la necesidad de hacer nada más, ni preocuparse de la versión del software ni de sus librerías. También permite un rápido despliegue ya que no contiene un hypervisor¹ y corre directamente en los recursos de la máquina anfitriona sin la necesidad de un sistema operativo.

Ventajas del uso de los contenedores Docker

- Modularidad: Se puede tomar una parte de la aplicación, para actualizarla o repararla sin necesidad de usar toda la aplicación entera.
- Control de versiones y capas: Cada imagen de Docker se compone de diferentes capas, al modificar la imagen se forma una capa nueva, y cada modificación está en un registro con los cambios realizados permitiendo así, un completo control de las imágenes.
- Restauración: Permite restaurar a una capa anterior de la imagen.
- Implementación rápida: Permite la construcción rápida y fácil de contenedores ya que, al crear un contenedor para cada proceso este puede compartir los recursos similares con otras aplicaciones.

¹ **Hypervisor:** Un hypervisor de máquina virtual es una capa que permite una virtualización del hardware y que aísla el sistema operativo y los recursos.

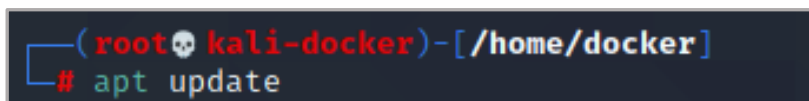
Además, el uso de contenedores tiene también sus ventajas respecto a las máquinas virtuales:

- Son más livianos que las máquinas virtuales.
- No es necesario instalar un sistema operativo por contenedor.
- Menos uso de recursos de la máquina anfitriona.
- Mayor cantidad de contenedores por equipo físico.
- Mejor portabilidad.

Instalación de Docker Engine

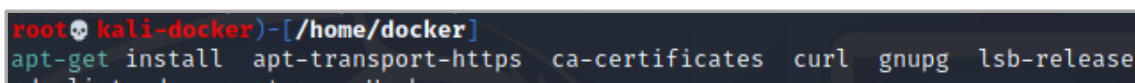
Para empezar a usar Docker primero deberemos de instalar Docker-engine, para ello podemos seguir los pasos que tienen Docker en su [página web oficial](#).

En primer lugar, deberemos actualizar los repositorios de nuestra máquina anfitriona y añadir unos nuevos repositorios para poder permitir el uso de repositorios HTTPS.



```
(root@kali-docker)-[/home/docker]
# apt update
```

Ilustración 2 Actualizar repositorios



```
(root@kali-docker)-[/home/docker]
apt-get install apt-transport-https ca-certificates curl gnupg lsb-release
```

Ilustración 3 Permitir el uso de HTTPS en repositorios

Para agregar la clave GPG² oficial de Docker. Para esto usaremos el siguiente comando:

```
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg-dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

Añadiremos a nuestra máquina los repositorios de Docker para tenerlo actualizado y poder descargarlo de los repositorios de Docker, para ello usaremos el siguiente comando.

```
echo \
"deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-
keyring.gpg] https://download.docker.com/linux/debian \
$(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
```

² **GNU Privacy Guard (GnuPG o GPG)** es una herramienta de cifrado y firmas digitales, pero con la principal diferencia que es software libre licenciado bajo la GPL.

En la ruta /etc/apt/sources.list.d/docker.list, estará el repositorio de Docker Engine la cual deberemos sustituir la parte Kali-rolling por buster.

```
list.d/docker.list *  
[root@kali ~]# sed -i 's/kali-rolling/buster/g' /etc/apt/sources.list.d/docker.list
```

Ilustración 4 Modificación del repositorio de Docker

```
s.list.d/docker.list *  
[root@kali ~]# cat /etc/apt/sources.list.d/docker.list  
deb https://download.docker.com/linux/debian buster stable
```

Ilustración 5 Repositorio ya corregido de Docker

Una vez configurado los repositorios actualizaremos la lista de repositorios de nuestra máquina e instalaremos Docker engine.

```
(root@kali-docker)~# apt-get update  
Obj:1 http://kali.download/kali kali-rolling InRelease  
Des:2 https://download.docker.com/linux/debian buster InRelease  
Des:3 https://download.docker.com/linux/debian buster/stable InRelease
```

Ilustración 6 Actualización completa de los repositorios de Kali Linux

```
(root@kali-docker)~# sudo apt-get install docker-ce docker-ce-cli containerd.io  
Leyendo lista de paquetes... Hecho
```

Ilustración 7 Instalación de Docker

Para comprobar que la instalación se ha hecho correctamente usaremos una imagen de prueba llamada hello-world para comprobar que funciona correctamente.

```
(root@kali-docker)~# sudo docker run hello-world  
Unable to find image 'hello-world:latest' locally  
latest: Pulling from library/hello-world  
b8dfde127a29: Pull complete  
Digest: sha256:308866a43596e83578c7dfa15e27a73011bdd402185a84c5cd7f32a88b5011  
Status: Downloaded newer image for hello-world:latest  
  
Hello from Docker!  
This message shows that your installation appears to be working correctly.  
  
To generate this message, Docker took the following steps:  
1. The Docker client contacted the Docker daemon.  
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
   (amd64)  
3. The Docker daemon created a new container from that image which runs the  
   executable that produces the output you are currently reading.  
4. The Docker daemon streamed that output to the Docker client, which sent it  
   to your terminal.  
  
To try something more ambitious, you can run an Ubuntu container with:  
$ docker run -it ubuntu bash
```

Ilustración 8 Uso de la imagen Hello-World

Instalar Docker Compose

Para complementar la instalación de Docker, también instalaremos Docker-Compose en nuestra máquina anfitriona. Docker compose es una herramienta que nos que nos facilitará la instalación de las aplicaciones, sus dependencias y su configuración a través de archivos con formato YAML.

```
version: "3.9" # optional since v1.27.0
services:
  web:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - ./code
      - logvolume01:/var/log
    links:
      - redis
  redis:
    image: redis
    volumes:
      logvolume01: {}
```

Ilustración 9 Ejemplo de archivo YAML

Para instalar Docker Compose al igual que en la instalación de Docker Engine usaremos las instrucciones de su [página web](#).

En primer lugar, para empezar con la instalación tendremos que descargar del repositorio de GitHub Docker compose, para ello utilizaremos este comando en la terminal.

```
sudo curl -L
https://github.com/docker/compose/releases/download/1.28.6/docker-
compose-\$\(uname -s\)-\$\(uname -m\) -o /usr/local/bin/docker-compose
```

Una vez descargado e instalado Docker Compose le daremos permisos de ejecución al archivo.

```
(root@kali-docker)-[/home/docker]
# sudo chmod +x /usr/local/bin/docker-compose
```

Ilustración 10 Dar permisos de ejecución al archivo Docker-compose

Para poder comprobar la versión de Docker Compose instalado y que lo hemos instalado bien usaremos:

```
(root@kali-docker)-[/home/docker]
# docker-compose --version
docker-compose version 1.28.5, build c4eb3a1f
```

Ilustración 11 Versión de Docker-compose ya instalado

1.2 Node JS Y Node Goat

En los siguientes apartados se explicará qué es NodeGoat y que es NodeJS.

¿Qué es Node JS?

NodeJS es un entorno de ejecución de JavaScript utilizado en el Back-end y que está construido sobre el motor v8 de Chrome el cual, es uno de los motores más utilizados y avanzados a nivel de JavaScript. NodeJS es liviano y eficiente ya que trabaja con entradas y salidas, está orientado a eventos y es asíncrono³.

¿Qué es Node Goat?

NodeGoat es una aplicación web desarrollada con Node JS que contiene las vulnerabilidades OWASP top 10 del año 2013 y que contiene una base de datos sobre MongoDB⁴ la cual almacena todos los datos de la aplicación. NodeGoat emula una página web sobre los bonos, fondos e inversiones de los clientes en la bolsa. Tiene un panel con diferentes secciones, donde cada sección contiene las diferentes vulnerabilidades.

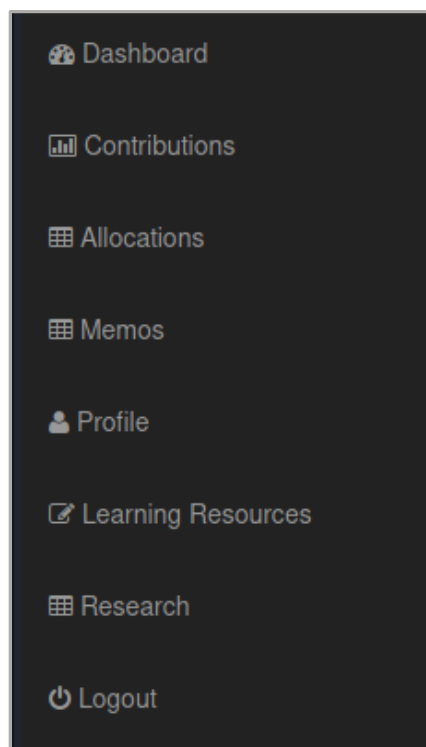


Ilustración 12 Secciones de NodeGoat

³ Que no tiene lugar en completa correspondencia temporal con otro proceso o causa

⁴ MongoDB es una base de datos distribuida, basada en documentos y de uso general que ha sido diseñada para desarrolladores de aplicaciones modernas y para la era de la nube

Instalación Node Goat

Para comenzar a utilizar Node Goat se puede hacer de varias maneras:

- Instalándolo directamente en tu máquina y todas sus dependencias
- Utilizando Docker
- A través del PaaS⁵ Heroku⁶.

Todas estas opciones están indicadas en su repositorio de [GitHub](#), en este análisis usaremos Docker por su rapidez y por sus ventajas comentadas anteriormente.

En primer lugar, clonamos el repositorio y lo guardaremos en un directorio (en mi caso descargas).

```
(root@kali-docker)-[/home/docker/Descargas]
# git clone https://github.com/OWASP/NodeGoat.git
Clonando en 'NodeGoat' ...
remote: Enumerating objects: 6279, done.
remote: Total 6279 (delta 0), reused 0 (delta 0), pack-reused 6279
Recibiendo objetos: 100% (6279/6279), 8.79 MiB | 10.36 MiB/s, listo.
Resolviendo deltas: 100% (1844/1844), listo.
```

Ilustración 13 Clonación del repositorio de GitHub

Usaremos Docker-compose build para instalar NodeGoat en Docker y que se configure.

```
(root@kali-docker)-[/home/docker/Descargas/NodeGoat]
# docker-compose build
mongo uses an image, skipping
Building web
Sending build context to Docker daemon 2.63MB
Step 1/14 : FROM node:12-alpine
```

Ilustración 14 Configuración e instalación de Node Goat

Una vez instalado y configurado usaremos el comando Docker-compose up para iniciar la máquina. También usaremos esta orden cada vez que queramos iniciar NodeGoat.

```
(root@kali-docker)-[/home/docker/Descargas/NodeGoat]
# docker-compose up
Creating network "nodegoat_default" with the default driver
Pulling mongo (mongo:latest) ...
latest: Pulling from library/mongo
```

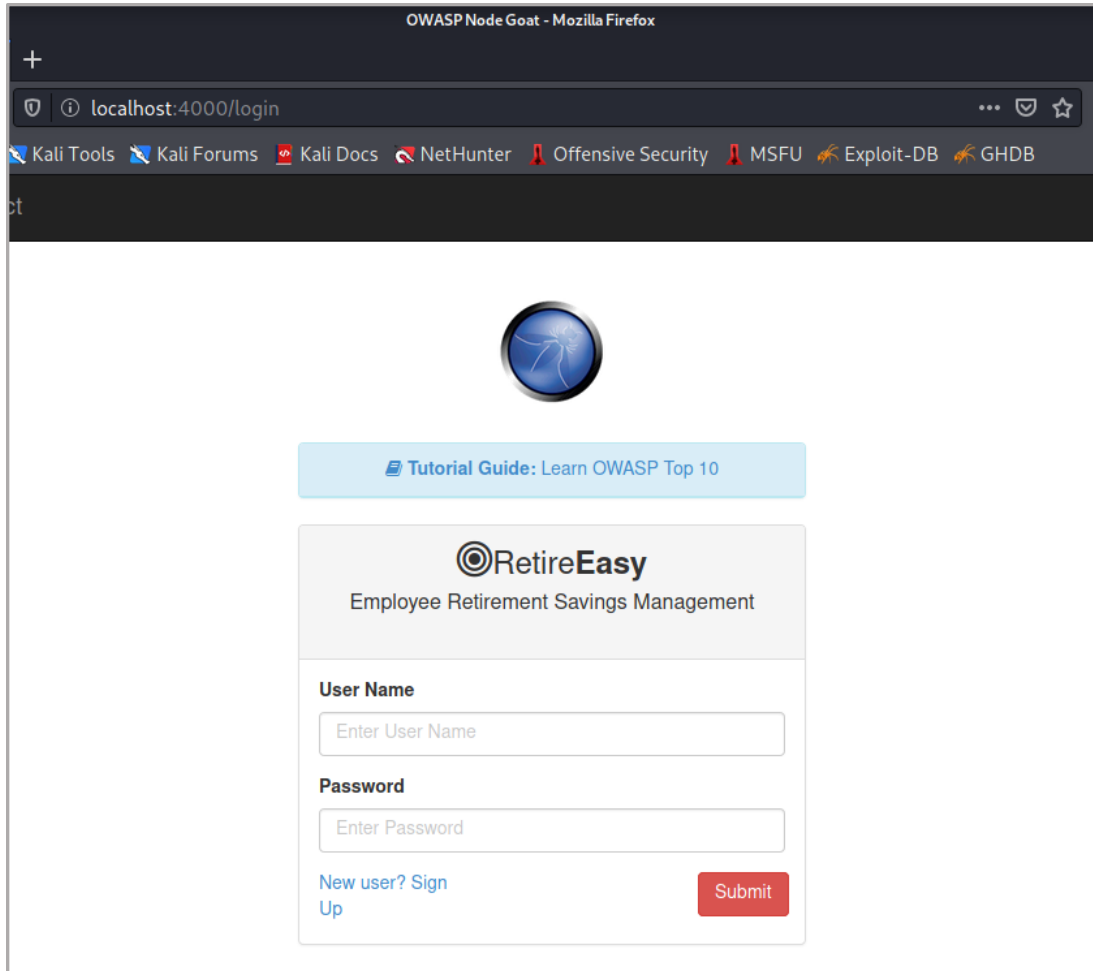
Ilustración 15 Poner en marcha NodeGoat

⁵ Plataforma como servicio (PaaS) es un entorno de desarrollo e implementación completo en la nube, con recursos que permiten entregar todo, desde aplicaciones sencillas basadas en la nube hasta aplicaciones empresariales sofisticadas habilitadas para la nube.

⁶ Heroku es uno de los PaaS () más utilizados en la actualidad en entornos empresariales por su fuerte enfoque en resolver el despliegue de una aplicación.

Para poder acceder a la aplicación web NodeGoat, en la barra de búsquedas tendremos que poner la dirección <http://localhost:4000>. En el login podremos usar diferentes usuarios para poder acceder y explorar la aplicación:

- Administrador: admin|Admin123
- Usuario1: user1|User1_123
- Usuario2: user2|User2_123




OWASP Node Goat - Mozilla Firefox

localhost:4000/login

Kali Tools Kali Forums Kali Docs NetHunter Offensive Security MSFU Exploit-DB GHDB

ct



[Tutorial Guide: Learn OWASP Top 10](#)

RetireEasy
Employee Retirement Savings Management

User Name

Password

New user? [Sign Up](#)

Ilustración 16 Login de NodeGoat

2. Marco teórico sobre OWASP

En los siguientes apartados se explica que es OWASP, su top ten, de que se compone y diferencias entre los más actuales tops ten y explotación de las vulnerabilidades en la aplicación Node Goat.

¿Qué es OWASP y su Top 10?

OWASP (Open Web Application Source Project) o proyecto abierto de seguridad en aplicaciones web es una comunidad Open Source⁷ fundada el 1 de diciembre del año 2003 la cual se dedica a que las organizaciones desarrollen aplicaciones seguras en la que se puedan confiar nuestros datos.

Cada 4 años, OWASP saca su Top 10, el cual expone las vulnerabilidades más comunes y con riesgos potenciales que ocurren en las aplicaciones web expuestas por los especialistas en seguridad y la comunidad. Para poder navegar seguro y sin preocupaciones de ser posibles víctimas de ciberdelincuentes, se aconseja que todas aquellos usuarios o empresas cuyo sector sea el desarrollo de aplicaciones, adopten estas reglas expuestas en el top ten para así garantizar su uso seguro.

OWAPS TOP 10 - 2007		OWAPS TOP 10 - 2010		OWAPS TOP 10 - 2013		OWAPS TOP 10 - 2017
A1 – Secuencia de Comandos en Sitios Cruzados (XSS)	▲ 1	A1 – Inyección	→ 0	A1 – Inyección	→ 0	A1 – Inyección
A2 – Inyección	▼ -1	A2 – Secuencia de Comandos en Sitios Cruzados (XSS)	▲ 1	A2 – Pérdida de Autenticación y Gestión de Sesiones	→ 0	A2 – Pérdida de Autenticación
A3 – Ejecución Maliciosa de Ficheros	▲ 4	A3 – Pérdida de Autenticación y Gestión de Sesiones	▼ -1	A3 – Secuencia de Comandos en Sitios Cruzados (XSS)	▲ 3	A3 – Exposición de Datos Sensibles
A4 – Referencia Directa Insegura a Objetos	▲ 1	A4 – Referencia Directa Insegura a Objetos	→ 0	A4 – Referencia Directa Insegura a Objetos	(*)	A4 – XML External Entities (XEE)
A5 – Falsificación de Peticiones en Sitios Cruzados (CSRF)	→ 0	A5 – Falsificación de Peticiones en Sitios Cruzados (CSRF)	▲ 1	A5 – Configuración de Seguridad Incorrecta	(**)	A5 – Pérdida de Control de Acceso
A6 – Filtrado de Información y Manejo Inapropiado de Errores	(*)	A6 – Defectuosa Configuración de Seguridad	(*)	A6 – Exposición de Datos Sensibles	▼ -1	A6 – Configuración de Seguridad Incorrecta
A7 – Pérdida de Autenticación y Gestión de Sesiones	▲ 1	A7 – Almacenamiento Criptográfico Inseguro	(*)	A7 – Ausencia de Control de Acceso a las Funciones	↓ -4	A7 – Secuencia de Comandos en Sitios Cruzados (XSS)
A8 – Almacenamiento Criptográfico Inseguro	▲ 2	A8 – Falla de Restricción de Acceso a URL	↓ -3	A8 – Falsificación de Peticiones en Sitios Cruzados (CSRF)	(*)	A8 – Deserialización insegura
A9 – Comunicaciones Inseguras	(*)	A9 – Protección Insuficiente en la Capa de Transporte	(*)	A9 – Uso de Componentes con Vulnerabilidades Conocidas	→ 0	A9 – Uso de Componentes con Vulnerabilidades Conocidas
A10 – Falla de Restricción de Acceso a URL	(*)	A10 – Redirecciones y reenvíos no validados	→ 0	A10 – Redirecciones y reenvíos no validados	(*)	A10 – Registro y monitorización insuficiente

Ilustración 17 Top 10 OWASP 2007-2017

⁷ Open Source: El software open source es un código diseñado de manera que sea accesible al público (<https://www.redhat.com/es/topics/open-source/what-is-open-source>)

La organización OWASP también proporciona recursos gratuitos que tratan sobre las vulnerabilidades en los entornos de las aplicaciones web, tanto el cómo corregirlas y el por qué se producen. Entre los recursos que ofrece se puede dividir en:

- **Materiales de educación:** OWASP Top 10, guías sobre el desarrollo seguro en las aplicaciones web, *cheat sheets*, etc.
- **Software:**
 - o **WebGoat:** aplicación insegura que permite a los desarrolladores explotar las diferentes vulnerabilidades en una aplicación web y así comprender su funcionamiento y cómo corregirlas.

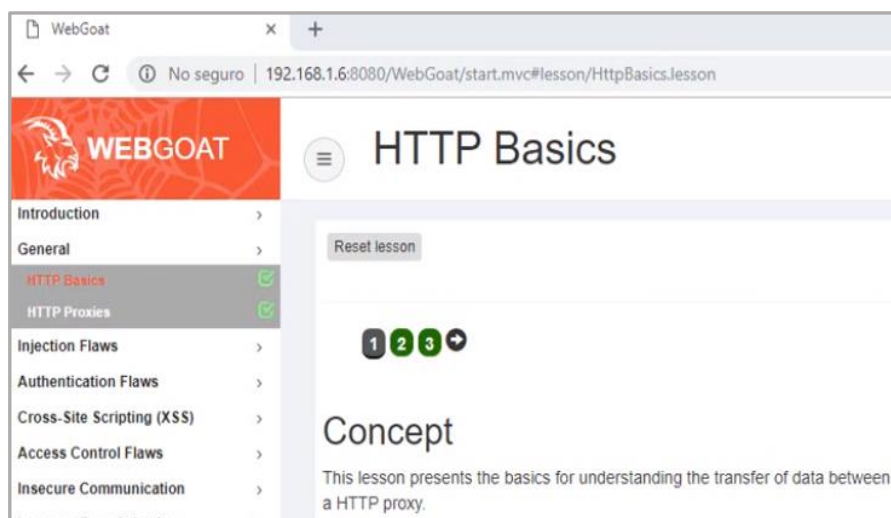


Ilustración 18 Interfaz de OWASP WebGoat

- o **OWASP-ZAP:** Framework desarrollado en Java que analiza el tráfico HTTP/HTTPS, el cual puede ser usado por un desarrollador o analista de seguridad, ya que también permite escanear las vulnerabilidades de una aplicación web.

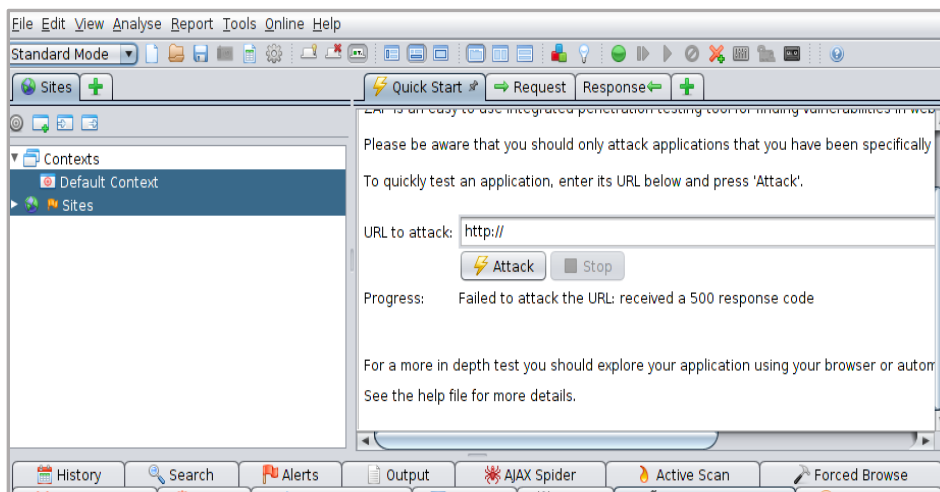


Ilustración 19 Interfaz OWASP ZAP

Diferencias entre OWASP top ten 2010 al 2013

Con el avance de los años y de las nuevas tecnologías que se utilizan en el ámbito de las aplicaciones web, se cambia tanto el modo de ataque, como las vulnerabilidades a explotar y las defensas. Los cambios son los siguientes:

1. Basados en los datos proporcionados por la comunidad, la Pérdida de autenticación y gestiones de sesiones asciende en el ranking. Se piensa que no es por el crecimiento de ataques sino por el mayor esfuerzo de detección de estas. Se intercambia las posiciones del A2 y A3.
2. La falsificación de peticiones en sitios cruzados (CSRF) disminuye la prevalencia, por lo tanto, desciende la posición A5 a la A8. Esto se debe a que durante los anteriores años ha estado en Top Ten por lo cual ha motivado a los desarrolladores y organización a combatir esta vulnerabilidad.
3. Ampliación de la falla de Restricción de Acceso a la URL para ampliar el significado: la falla A8-2010: Falla de Restricción de Acceso a URL es A7-2013: Ausencia de Control de Acceso a las Funciones, esto se hace para no solo cubrir los accesos a través de la URL, sino también de los otros métodos posibles.
4. Fusión y ampliación de A7-2010: Almacenamiento criptográfico Inseguro y A9-2010: Redirecciones y reenvíos no validados para la creación de A6-2013: Exposición de datos sensibles, esta nueva vulnerabilidad abarca la protección de datos sensibles desde que son provistos por el usuario, transmitidos, almacenados por la aplicación y enviado al navegador nuevamente.
5. Adición de A9-2013: Uso de componentes con vulnerabilidades conocidas: Esta vulnerabilidad estaba presente en A6-2010: Defectuosa configuración de seguridad, pero en el OWASP top ten 2017 posee su propia categoría debido al crecimiento del uso de componentes de terceros incrementado el riesgo de la utilización de componentes con vulnerabilidades conocidas.

OWASP Top 10 – 2010 (Previo)	OWASP Top 10 – 2013 (Nuevo)
A1 – Inyección	A1 – Inyección
A3 – Pérdida de Autenticación y Gestión de Sesiones	A2 – Pérdida de Autenticación y Gestión de Sesiones
A2 – Secuencia de Comandos en Sitios Cruzados (XSS)	A3 – Secuencia de Comandos en Sitios Cruzados (XSS)
A4 – Referencia Directa Insegura a Objetos	A4 – Referencia Directa Insegura a Objetos
A6 – Defectuosa Configuración de Seguridad	A5 – Configuración de Seguridad Incorrecta
A7 – Almacenamiento Criptográfico Inseguro – Fusionada A9→	A6 – Exposición de Datos Sensibles
A8 – Falla de Restricción de Acceso a URL – Ampliada en →	A7 – Ausencia de Control de Acceso a las Funciones
A5 – Falsificación de Peticiones en Sitios Cruzados (CSRF)	A8 – Falsificación de Peticiones en Sitios Cruzados (CSRF)
<dentro de A6: – Defectuosa Configuración de Seguridad>	A9 – Uso de Componentes con Vulnerabilidades Conocidas
A10 – Redirecciones y reenvíos no validados	A10 – Redirecciones y reenvíos no validados
A9 – Protección Insuficiente en la Capa de Transporte	Fusionada con 2010-A7 en la nueva 2013-A6

Ilustración 20 Actualización del año 2010 al 2013

Diferencias entre OWASP top ten 2013 al 2017

En esos años hubo un gran cambio en el sector de las aplicaciones web, por ellos OWASP tuvo que mejorar su metodología, la obtención de datos, trabajando más con la comunidad y reordenando los riesgos. Todo esto ocurrido por el gran cambio de la tecnología y de la arquitectura de las aplicaciones.

1. Los microservicios escritos por NodeJS y Spring Boot reemplazan a las aplicaciones tradicionales. Estos microservicios vienen con su propia seguridad, incluyendo el establecimiento de confianza entre los microservicios y la comunicación entre sí.
2. Las aplicaciones de una sola página, escrita con frameworks⁸ de JavaScript (Angular, ReactJS), permiten la creación de interfaces para los usuarios, estas funcionalidades tradicionalmente iban al Back-End lo cual trae nuevos desafíos de seguridad.
3. JavaScript es el lenguaje n°1 en el uso de las aplicaciones web, ya sea por uso en el Front-End⁹ con sus framework Bootstrap, Angular, ReactJS y en el Back-End con NodeJS.

Por todos estos avances OWASP ha actualizado su Top Ten con las siguientes actualizaciones:

- Nuevos riesgos respaldados por la comunidad y por los datos obtenidos por análisis:
 1. A4-2017: Entidades Externas XML (XEE) es una de las nuevas categorías la cual ha sido respaldada principalmente por el uso de las herramientas de análisis estático de código (SAST¹⁰). Esta vulnerabilidad es producida por un analizador XML mal configurado.
 2. A8.2017: Deserialización Insegura, esta vulnerabilidad permite la ejecución de código remoto o la manipulación de objetos sensibles en la plataforma afecta por esta.
 3. A10-2017: Registro y Monitoreo Insuficientes, con la falta de estos aspectos se puede atrasar o no llegar a detectar las actividades maliciosas o de sustracción de datos, y así no llegar a realizar una respuesta adecuada a los incidentes y a la investigación de estos.

⁹ Es la parte que ve el usuario y en la que se incluyen la línea de diseño y los elementos gráficos de la página.

¹⁰ Las herramientas SAST son aquellas que están diseñadas para analizar el código fuente o versiones compiladas de código para ayudar a encontrar fallas de seguridad.

- También con la actualización se fusionan y se retiran vulnerabilidades, esto ocurre ya sea porque se ha erradicado total o parcialmente el riesgo.

1. A4-2013: Referencia Directa Insegura a Objetos y A7-2013: Ausencia de control de acceso a las funciones se fusionan en A5-2017: Pérdida de control de acceso.
2. A8-2013: Falsificación de Peticiones en Sitios cruzados (CSRF) debido al uso de los framework y que algunos incluyen defensas contra CSRF solo se encontró en el 5% de las aplicaciones.
3. A10-2013: Redirecciones y reenvíos no validados, este riesgo se encuentra aproximadamente en el 8% de las aplicaciones el cual fue superada por A4-2017: Entidad Externa de XML (XEE)

OWASP Top 10 2013	±	OWASP Top 10 2017
A1 – Inyección	➔	A1:2017 – Inyección
A2 – Pérdida de Autenticación y Gestión de Sesiones	➔	A2:2017 – Pérdida de Autenticación y Gestión de Sesiones
A3 – Secuencia de Comandos en Sitios Cruzados (XSS)	➔	A3:2017 – Exposición de Datos Sensibles
A4 – Referencia Directa Insegura a Objetos [Unido+A7]	U	A4:2017 – Entidad Externa de XML (XEE) [NUEVO]
A5 – Configuración de Seguridad Incorrecta	➔	A5:2017 – Pérdida de Control de Acceso [Unido]
A6 – Exposición de Datos Sensibles	➔	A6:2017 – Configuración de Seguridad Incorrecta
A7 – Ausencia de Control de Acceso a las Funciones [Unido+A4]	U	A7:2017 – Secuencia de Comandos en Sitios Cruzados (XSS)
A8 – Falsificación de Peticiones en Sitios Cruzados (CSRF)	⊗	A8:2017 – Deserialización Insegura [NUEVO, Comunidad]
A9 – Uso de Componentes con Vulnerabilidades Conocidas	➔	A9:2017 – Uso de Componentes con Vulnerabilidades Conocidas
A10 – Redirecciones y reenvíos no validados	⊗	A10:2017 – Registro y Monitoreo Insuficientes [NUEVO, Comunidad]

Ilustración 21 Actualización del año 2013 al 2017

¿OWASP 2021?

A la fecha de la realización de este proyecto, todavía no se ha expuesto ninguna fecha de publicación respecto al OWASP Top Ten 2021. La posible fecha de publicación podría ser de octubre a diciembre del año 2021 según las fechas de salida de los anteriores Top Ten.

3. Explicación y explotación de las vulnerabilidades

En este capítulo se comenzará a realizar las vulnerabilidades de OWASP Top Ten 2013 disponibles en NodeGoat explicando su realización, por qué ocurren y cómo se corrigen.

A1 - Inyección de Código

Este ataque es fácil de reconocer y puede conllevar grandes riesgos. Estas vulnerabilidades se encuentran a menudo en consultas SQL, NoSQL, LDAP (protocolo ligero de acceso a directorios), comandos de sistema, etc. y ocurren cuando se envían datos no validados a un intérprete como parte de una consulta para “engañarle” y así ejecute el código malicioso mostrando datos o realizando acciones sin autorización. Esta vulnerabilidad puede ocasionar pérdida de datos, pérdida de acceso y denegación de acceso.

Cómo prevenirlo

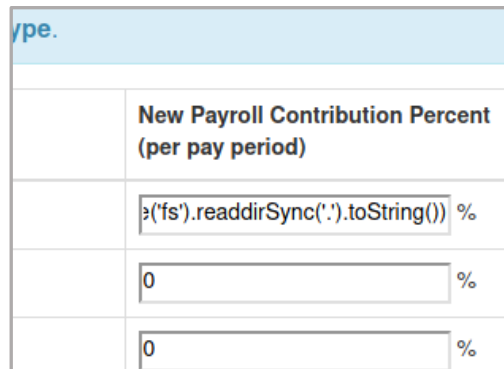
- Revisar los datos de entrada antes de ejecutarlos, para ello se pueden llevar a cabo diferentes acciones:
 - Uso de una API segura que evite el uso completo de un intérprete.
 - Utilizar la función LIMIT y otros controles SQL para evitar la fuga masiva de datos en caso de ataque SQL con éxito.
 - Pruebas y escaneo con herramientas especializadas.
 - Uso de listas blancas que contengan las estructuras y caracteres permitidos.
 - Siempre que la aplicación web pueda aceptar entrada de los usuarios, estas deberán pasar por un tratamiento de los datos:
 - Filtrar: En base a patrones o expresiones regulares, determinar qué tipo de datos acepta el sistema y cuáles no.
 - Escapar: Aceptar todo tipo de datos, pero escapándose correctamente.
 - Sanear: Aceptar los datos eliminando el contenido inapropiado.
 - Transformar: Aceptar todo tipo de datos, pero transformándolos a un tipo de datos aceptado y validado.

Vulnerabilidades presentes en Node Goat

A1.1 - Inyección para acceder a archivos

Esta vulnerabilidad ocurre en la pestaña de *Contributions*. Se produce en los campos de porcentajes. A través de la inyección se pueden obtener los archivos y directorios a los que el usuario con el que se ejecuta la aplicación tiene acceso.

Con el código `res.end(require('fs').readdirSync('.').toString())`, esto produce que con el módulo `fs` utilizado para trabajar con ficheros del sistema se obtengan los archivos que contiene el directorio donde corre la aplicación.



The screenshot shows a web form titled "New Payroll Contribution Percent (per pay period)". It contains two input fields for percentages. The first field has the malicious payload `require('fs').readdirSync('.').toString() %` injected into it. The second field contains the value "0".

Ilustración 22 Inyección de código malicioso 1

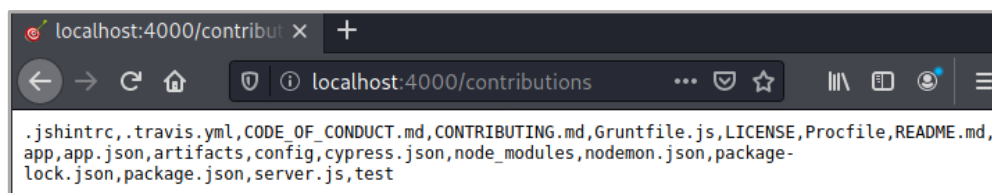
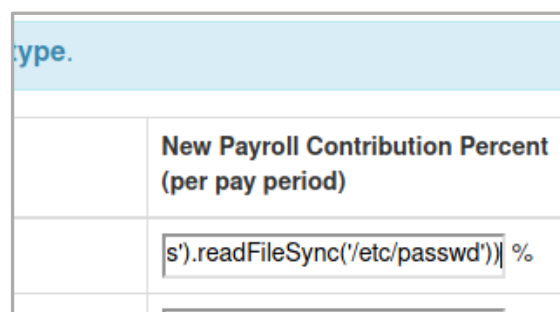


Ilustración 23 Archivos del directorio de la aplicación web

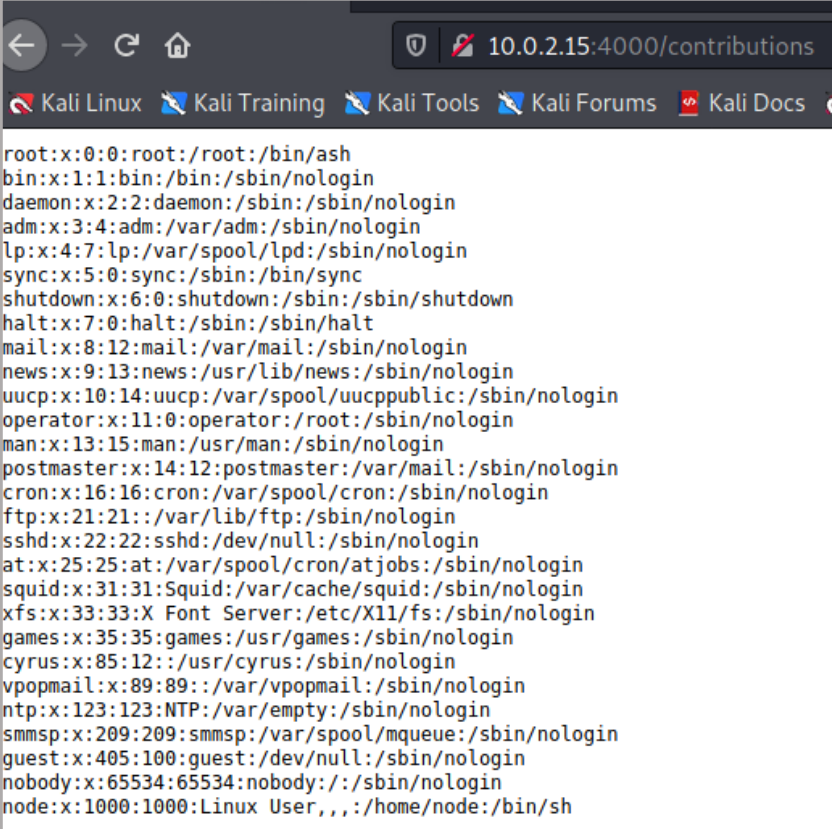
Se puede obtener también los archivos de otros directorios, por ejemplo `/etc/passwd` que contiene todos los usuarios del sistema. Para ello utilizaremos la siguiente inyección: `res.end (require ('fs'). readFileSync ('/etc/passwd'))`



The screenshot shows the same web form as in Illustration 22. The first input field now contains the malicious payload `require('fs').readFileSync('/etc/passwd') %`.

Ilustración 24 Inyección de código malicioso 2

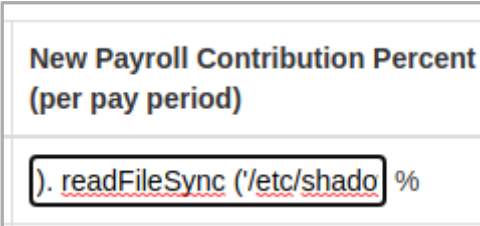
Como podemos observar la inyección tiene éxito y podemos observar todos los usuarios del sistema.



```
root:x:0:0:root:/root:/bin/ash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/mail:/sbin/nologin
news:x:9:13:news:/usr/lib/news:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucppublic:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
man:x:13:15:man:/usr/man:/sbin/nologin
postmaster:x:14:12:postmaster:/var/mail:/sbin/nologin
cron:x:16:16:cron:/var/spool/cron:/sbin/nologin
ftp:x:21:21::/var/lib/ftp:/sbin/nologin
sshd:x:22:22:sshd:/dev/null:/sbin/nologin
at:x:25:25:at:/var/spool/cron/atjobs:/sbin/nologin
squid:x:31:31:Squid:/var/cache/squid:/sbin/nologin
xfs:x:33:33:X Font Server:/etc/X11/fs:/sbin/nologin
games:x:35:35:games:/usr/games:/sbin/nologin
cyrus:x:85:12:/usr/cyrus:/sbin/nologin
vpopmail:x:89:89:/var/vpopmail:/sbin/nologin
ntp:x:123:123:NTP:/var/empty:/sbin/nologin
smmsp:x:209:209:smmsp:/var/spool/mqueue:/sbin/nologin
guest:x:405:100:guest:/dev/null:/sbin/nologin
nobody:x:65534:65534:nobody:/:/sbin/nologin
node:x:1000:1000:Linux User,,,:/home/node:/bin/sh
```

Ilustración 25 Archivo /etc/passwd de la aplicación web

En cambio, si solicitamos el archivo /etc/shadow que contiene las contraseñas no podemos acceder al archivo ya que el usuario que usamos en la aplicación no tiene los permisos necesarios. Para ello utilizaremos la siguiente inyección: `res.end (require ('fs'). readFileSync ('/etc/shadow'))`



```
. readFileSync ('/etc/shadow') %
```

Ilustración 26 Inyección de código malicioso 3



```
Oops..
Error: EACCES: permission denied, open '/etc/shadow'
```

Ilustración 27 Mensaje de error

También podremos saber los usuarios del sistema y sus documentos consultando el directorio `/home`. Para ello utilizaremos la siguiente inyección: `res.end(require('fs').readdirSync('/home/').toString())`

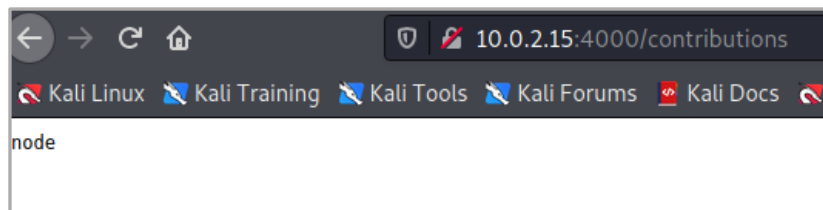


Ilustración 28 Directorio home de los usuarios

A1.2 - Ataque DOS

También permite en la pestaña de *Allocations* modificar la URL para crear un bucle con `while` y así saturar la aplicación web realizando un ataque DoS.

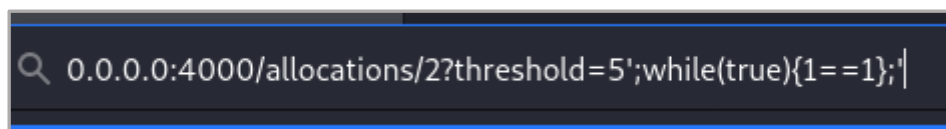


Ilustración 29 Inyección de código por URL

Al correr la aplicación en Docker utilizaremos el comando `Docker stats` para ver el uso del sistema y así poder comprobar que se está realizando el ataque mostrando el uso excesivo de los recursos.

Archivo Acciones Editar Vista Ayuda					
CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	
5021cdb6f7e3	nodegoat_web_1	0.00%	63.12MiB / 7.773GiB	0.79%	
fdd8310cc9d1	nodegoat_mongo_1	93.70%	191.5MiB / 7.773GiB	2.41%	

Ilustración 30 Uso de recursos de la imagen

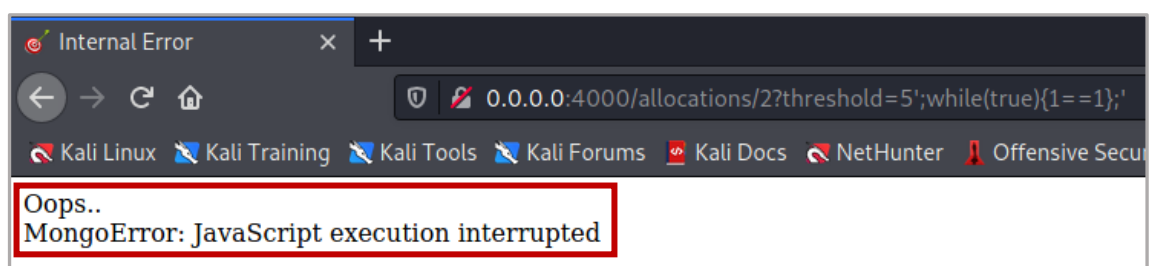


Ilustración 31 Crasheo de la aplicación

A1.3 - Inyección de comandos

Para poder realizar este ataque utilizaremos BurpSuite para poder modificar la petición y realizar la inyección de código. Para ello en primer lugar iremos a la pestaña de *Contributions* para enviar datos al servidor, capturar la cabecera de petición con BurpSuite y así poder modificarla.

This screen allows you to change the payroll percentages deducted from your paycheck for each type.

Contribution Type	Payroll Contribution Percent (per pay period)	New Payroll Contribution Percent (per pay period)
Employee Pre-Tax	0 %	<input type="text" value="0"/> %
Roth Contribution	0 %	<input type="text" value="0"/> %
Employee After Tax	5 %	<input type="text" value="1"/> %

Ilustración 32 Sección contributions

Una vez tengamos la cabecera de petición la enviaremos a *Repeater* para modificar la cabecera y volver a enviarla.

Request to http://localhost:4000 [127.0.0.1]

Forward

Drop

Intercept is on

Action

Open Browser

Pretty

Raw

ln

Actions

1 POST /contributions HTTP/1.1

2 Host: localhost:4000

3 Content-Length: 26

4 Cache-Control: max-age=0

5 Upgrade-Insecure-Requests: 1

6 Origin: http://localhost:4000

7 Content-Type: application/x-www-form-urlencoded

8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.36

9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8

10 Sec-Fetch-Site: same-origin

11 Sec-Fetch-Mode: navigate

12 Sec-Fetch-User: ?1

13 Sec-Fetch-Dest: document

14 Referer: http://localhost:4000/contributions

15 Accept-Encoding: gzip, deflate

16 Accept-Language: es-ES,es;q=0.9

17 Cookie: connect.sid=s%3ABK7nI-zK_SuKH0y3zyyQJdX2cvYUi9mJ.6po4SAfV1M%2FEjIsc

18 Connection: close

19

20 preTax=0&roth=0&afterTax=1

Ilustración 33 Análisis de la cabecera de petición

	Scan	
UKH0	Send to Intruder	Ctrl-I
	Send to Repeater	Ctrl-R
	Send to Sequencer	
	Send to Comparer	

Ilustración 34 Enviar a Repeater

Una vez tengamos la cabecera en *Repeater*, modificaremos el parámetro *ofertas* cambiando el valor dado anteriormente por el código a inyectar. En este caso usaremos nslookup para hacer que la aplicación envíe una petición de DNS a as.com resolviendo así su nombre e IP.

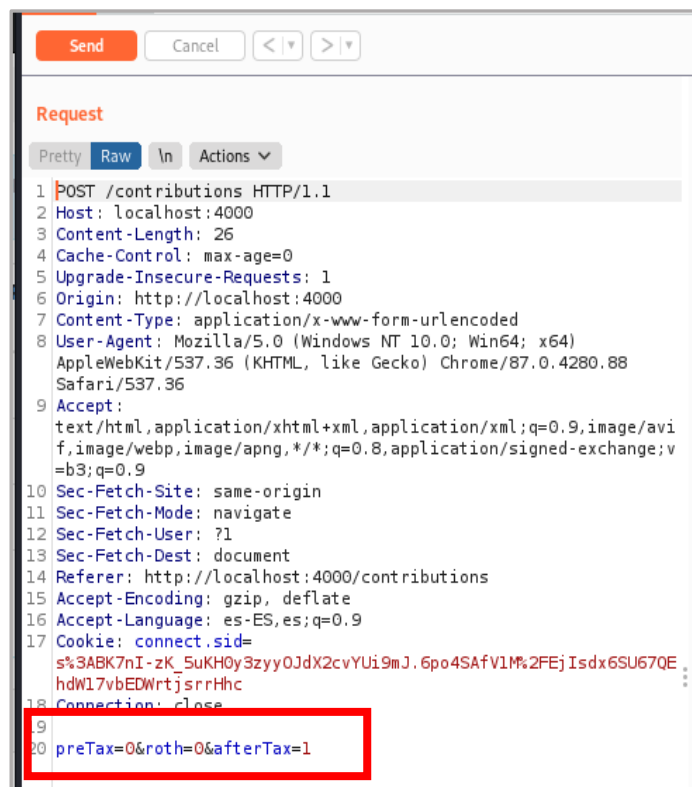


Ilustración 35 Cabecera de petición

A través del módulo `child_process` que nos permite ejecutar funcionalidades del sistema operativo usaremos el siguiente código para inyectar: `require('child_process').exec('nslookup+as.com');`



Ilustración 36 Código malicioso a inyectar

Para comprobar que este código malicioso funciona, utilizaremos un analizador de tráfico (Wireshark en este caso) para comprobar el tráfico que llega a la aplicación web. Una vez enviada la cabecera si analizamos las peticiones que se muestran en Wireshark podemos observar que se realiza una consulta de la dirección 10.0.2.15 (aplicación web) hacia la dirección 212.166.210.80 (servidor DNS) que resuelve el nombre de as.com realizando así la ejecución del comando nslookup

que un filtro de visualización ... <Ctrl-/>					
Time	Source	Destination	Protocol	Length	Info
1 0.000000000	10.0.2.15	212.166.210.80	DNS	66	Standard query 0xad46 A as.com
2 0.000238105	10.0.2.15	212.166.210.80	DNS	66	Standard query 0xae3e AAAA as.com
3 0.131740323	212.166.210.80	10.0.2.15	DNS	146	Standard query response 0xae3e AAAA
4 0.131740383	212.166.210.80	10.0.2.15	DNS	98	Standard query response 0xad46 A as.

Ilustración 37 Análisis de tráfico de red

A1.4 – Usuarios de la aplicación

Al realizar esta inyección podremos observar todos los usuarios registrados en la aplicación web y sus porcentajes. En primer lugar, iremos a la pestaña de *allocations* y modificaremos la URL para añadir `1' || 'a' == 'a`

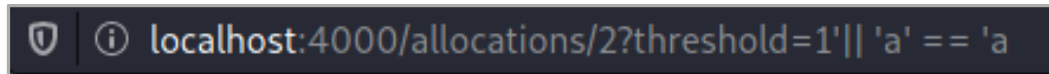


Ilustración 38 inyección a través de URL

Una vez modificada la URL se realizará la inyección y nos saldrán todos los porcentajes de los usuarios de la aplicación.

Asset Allocations for Node Goat Admin
Domestic Stocks : 6 %
Funds: 40 %
Bonds: 54 %
Asset Allocations for John Doe
Domestic Stocks : 25 %
Funds: 32 %
Bonds: 43 %
Asset Allocations for Will Smith
Domestic Stocks : 40 %
Funds: 34 %
Bonds: 26 %

Ilustración 39 Resultado de la inyección

A2 - Pérdida de autenticación y gestión de sesiones.

En este tipo de ataques, el atacante puede situarse tanto dentro de la organización o ser un agente externo. Este riesgo ocurre cuando los desarrolladores de aplicaciones web realizan un esquema de cómo se realizará la autenticación y la gestión de sesiones, el cómo serán implementadas y si son adecuadas o no, lo cual permitiría a los atacantes que comprometan las contraseñas, claves o tokens de sesión incluso otro tipo de fallos que permita al atacante suplantar las identidades de los usuarios.

Esta tarea es compleja, se debe realizar correctamente, por la cual suelen tener fallas de cierre de sesión, tiempo de desconexión, funciones de recordar contraseña, preguntas de recuperación, etc. Todos estos riesgos son difíciles de identificar ya que cada aplicación web es única.

Cómo prevenirlo

- Evitar la vulnerabilidad de XSS para evitar el robo de sesiones.
- Cumplir con todos los requisitos de autenticación (sección v2) y gestión de sesiones (sección v3) definidos en el Application Security Verification Standard (ASVS) de OWASP.
 - Medidas de autenticación:
 - Contraseñas seguras: más de 12 caracteres, utilizar símbolos, números, mayúsculas y minúsculas.
 - Factores de doble autenticación como captchas, mensajes a través de SMS o correo, etc.
 - Ciclo de vida de autenticación: las contraseñas tengan un tiempo de validez.
 - Requisitos del almacenamiento de las credenciales: que las contraseñas estén cifradas o hasheadas¹¹ y no en texto claro.
 - Intentos de logueo
 - Medidas de gestión de sesiones:
 - Verificar que la id de sesión no se muestra en la URL.
 - Verificar que el id de sesión se genera con cada autenticación del usuario.
 - Verificar que al cierre de sesión o la expiración invalide el token de sesión impidiendo así que retrocediendo se pueda acceder a la página.

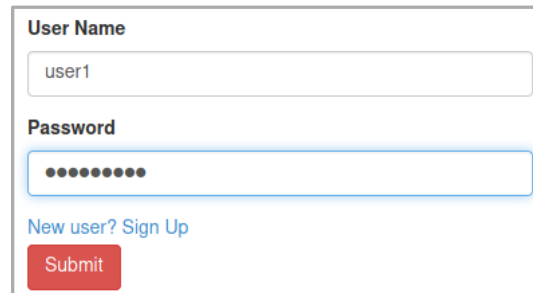
¹¹ Un hash es una operación criptográfica que a través de una entrada genera una salida o resumen único e irrepetible ej: entrada-owasp[salida-e0aca2fe8231010480c521fa93bc7ee6

Vulnerabilidad presente en Node Goat

Gestión de sesiones.

En esta prueba comprobaremos si al cerrar la sesión abierta y dando al botón de retroceder del navegador, podremos o no acceder a la sesión anterior.

El usuario 1 accedera al perfil de su aplicación y cerrará sesión.



A login form with two input fields. The first field is labeled 'User Name' and contains the text 'user1'. The second field is labeled 'Password' and contains ten dots. Below the password field is a link that says 'New user? Sign Up' in blue text. At the bottom of the form is a red button labeled 'Submit'.

Ilustración 43 Usuario 1 accede a la aplicación

Una vez el usuario haya consultado la página cerrará sesión ya abandona el ordenador.

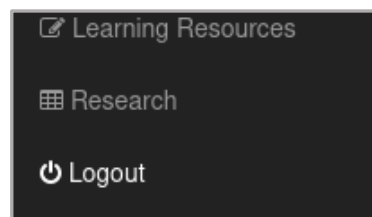


Ilustración 44 El usuario cierra sesión

El atacante llega al ordenador y ve el login, da al botón de retroceder...

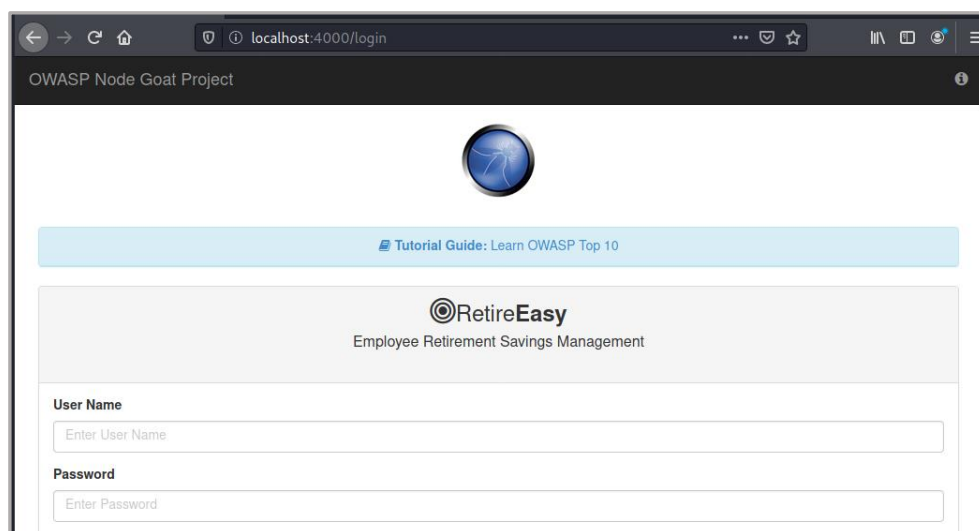


Ilustración 45 Página de login de la aplicación

Y el atacante logra obtener acceso al perfil del anterior usuario autenticado en la aplicación.

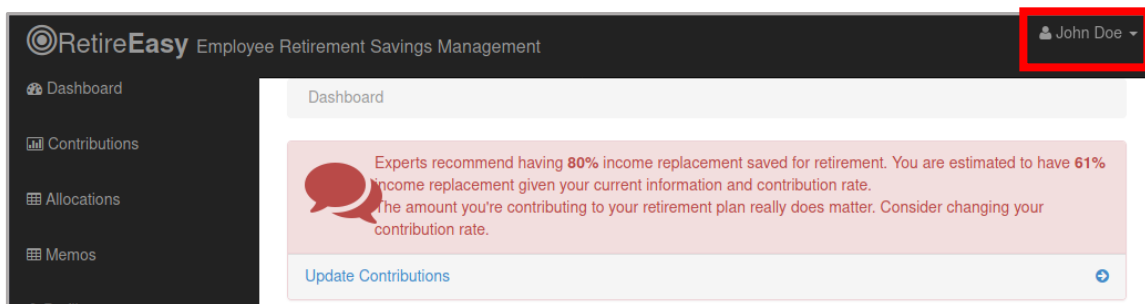


Ilustración 40 Acceso al perfil de la víctima

Autenticación.

En este ataque podemos comprobar dos riesgos:

- El uso de usuarios y contraseñas conocidos.
- Ninguna función de intentos de logeo.

Utilizaremos el usuario admin, el cual, a través de la fuerza bruta, obtendremos su contraseña. Este ataque es hacia la contraseña, pero también se puede realizar al usuario con la técnica de [Password Spraying](#).

Para ello utilizaremos BurpSuite, nos iremos a la pestaña de *Proxy* ▯ *Intercept*, una vez ahí daremos a *Open Browser* el cual nos abrirá un navegador configurado automáticamente donde pondremos la dirección a atacar.

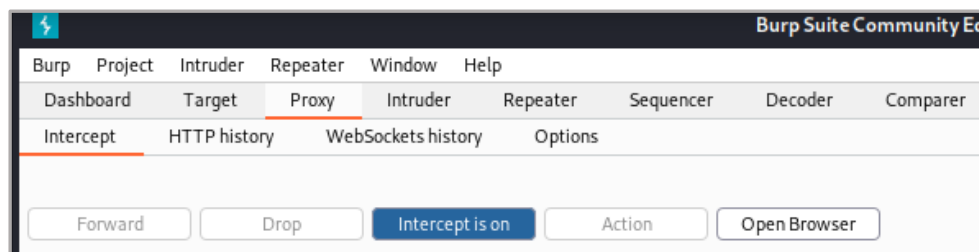


Ilustración 41 Acceder al BurpSuite

Una vez en login de la aplicación haremos un intento de logeo solo con el usuario para así a través de BurpSuite podré analizar la cabecera de petición.

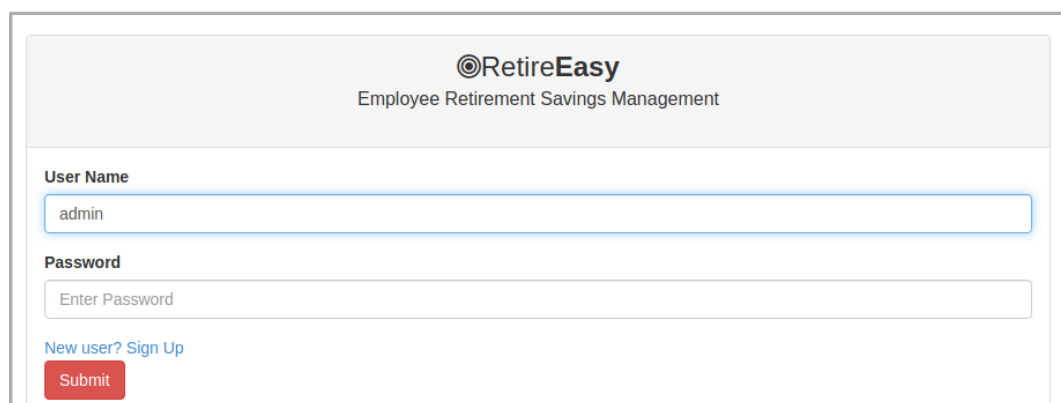


Ilustración 42 Prueba de login con el usuario admin

Al hacer la petición, nos saldrá en BurpSuite con los campos que se van a enviar a la aplicación. El campo que nos importa es el último, que contiene el usuario admin y la contraseña.

```
5 Upgrade-Insecure-Requests: 1
6 Origin: http://localhost:4000
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: http://localhost:4000/login
15 Accept-Encoding: gzip, deflate
16 Accept-Language: es-ES,es;q=0.9
17 Cookie: connect.sid=s%3AAXdNwu7jcojIGdpKdKGzkKSuJyE35DwZ.%2F8%2BEav1Npd5YoX4qA0MJJIJbnNxQfUUKYXpwNkS9Arw8
18 Connection: close
19
20 userName=admin&password=&_csrf=
```

Ilustración 43 Análisis de la petición de login

En el menú de arriba, seleccionamos *Action* y *Send to Intruder*.

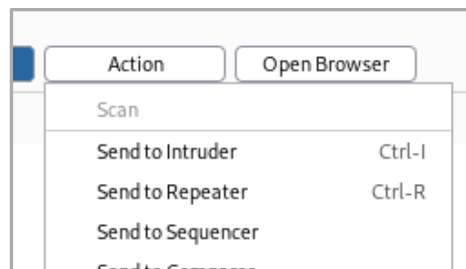


Ilustración 44 Enviar cabecera a Intruder

En el menú de *Intruder*, seleccionaremos la pestaña de *Payloads*.

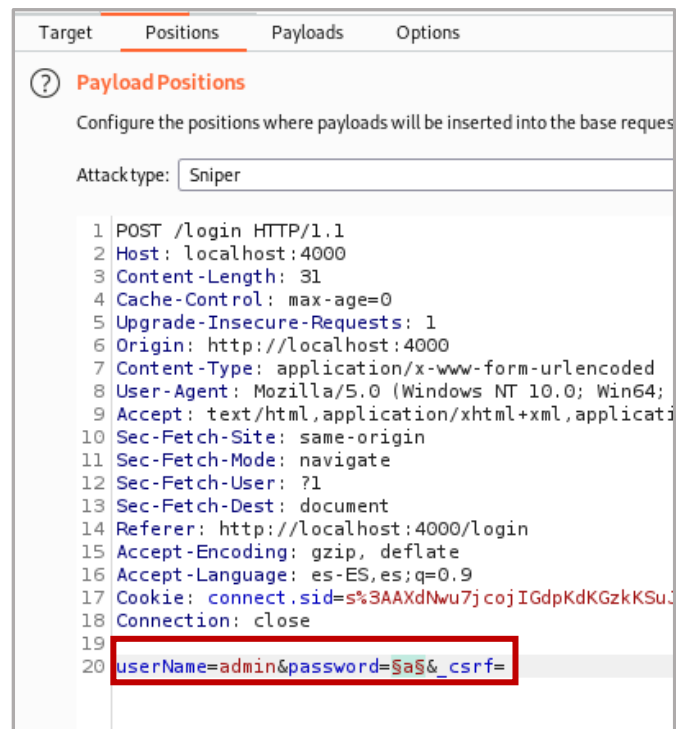


Ilustración 45 Cabecera en Intruder

Dentro de *Payloads* tendremos que añadir el diccionario que usaremos para realizar el ataque a la contraseña por fuerza bruta.

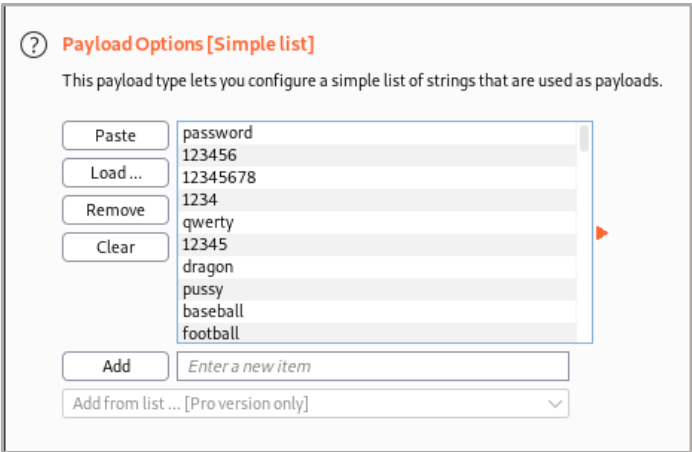


Ilustración 46 Añadir diccionario

Una vez añadido daremos al botón de *Start Attack* para comenzar así el ataque.



Ilustración 47 Comenzar el ataque

Una vez comenzado el ataque, nos saldrá una ventana con los intentos que está realizando, para poder comprobar cuál es la contraseña, nos tendremos que fijar en los tamaños de las cabeceras de respuesta. En esta captura podemos ver que la contraseña Admin_123 tiene diferente tamaño a las demás.

20	skinhead	200	<input type="checkbox"/>	<input type="checkbox"/>	7715
21	skilled	200	<input type="checkbox"/>	<input type="checkbox"/>	7715
22	shakira	200	<input type="checkbox"/>	<input type="checkbox"/>	7715
23	shaft	200	<input type="checkbox"/>	<input type="checkbox"/>	7715
24	shadow12	200	<input type="checkbox"/>	<input type="checkbox"/>	7715
25	Admin_123	302	<input type="checkbox"/>	<input type="checkbox"/>	258
26	tang	200	<input type="checkbox"/>	<input type="checkbox"/>	7715
27	1234qwer	200	<input type="checkbox"/>	<input type="checkbox"/>	7715
28	alfred	200	<input type="checkbox"/>	<input type="checkbox"/>	7715
29	ball	200	<input type="checkbox"/>	<input type="checkbox"/>	7715
30	98765432	200	<input type="checkbox"/>	<input type="checkbox"/>	7715

Ilustración 48 Análisis del tamaño de las cabeceras de respuesta

Si seleccionamos la cabecera de la contraseña Admin_123, podemos observar que es correcta y que nos está redirigiendo a la página principal.

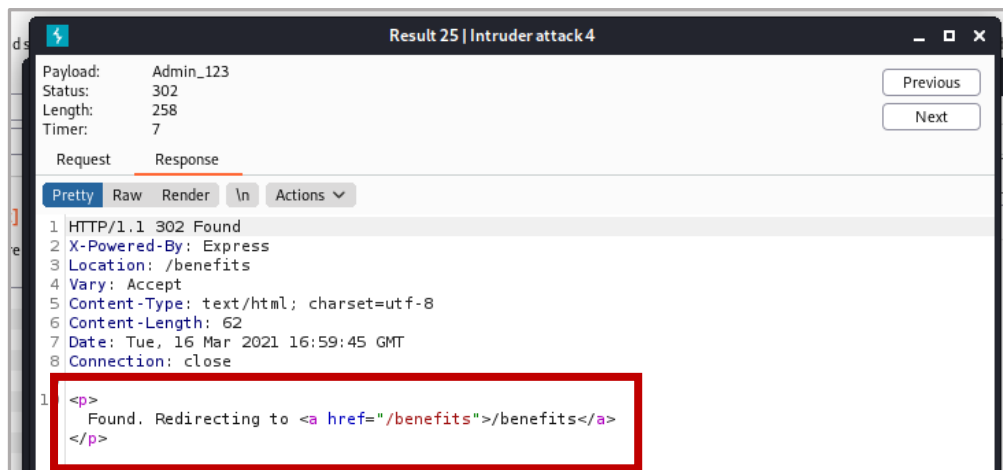


Ilustración 49 Cabecera de respuesta de la contraseña Admin_123

Si nos dirigimos al login y ponemos como usuario admin y como contraseña utilizamos la que hemos conseguido con el ataque de fuerza bruta, podremos ver que tenemos acceso como administrador a la aplicación web.

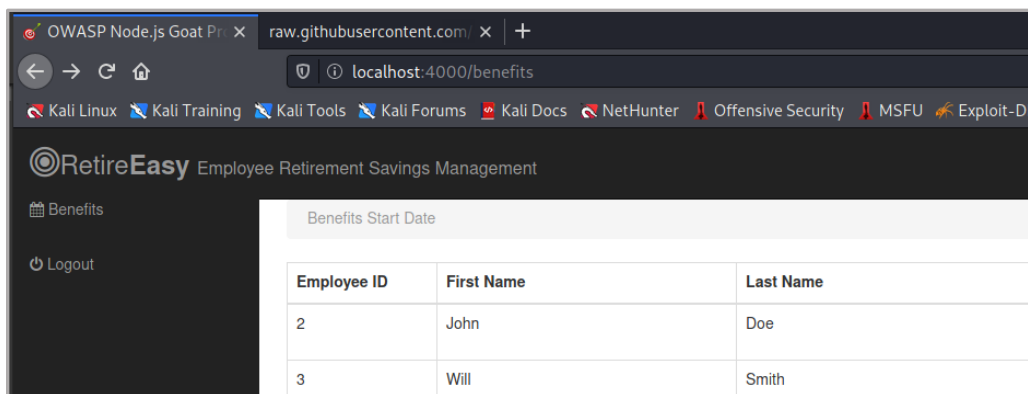


Ilustración 50 Página de inicio del administrador

A3 - Secuencia de comandos en sitios cruzados (XSS)

La vulnerabilidad XSS (Cross Site Scripting) se produce cuando una aplicación web toma datos no confiables y los envía al navegador sin ninguna validación, codificación apropiada o sanitización. El ataque XSS permite a los atacantes ejecutar una serie de comandos en el navegador de la víctima con lo que pueden llegar a secuestrar la sesión, destruir o modificar sitios web y dirigir al usuario hacia otro lugar malicioso.

Hay varios tipos de ataques XSS:

- **Reflejado:** Son aquellos cuyo ataque se refleja en el servidor web como un mensaje de alerta. Estos son enviados a la víctima para que cuando haga clic en algún lugar o solo navegue por la página, esta acción ejecute el código malicioso y así refleje en el navegador de la víctima.
- **Almacenado:** Son aquellos ataques en los cuales, el código inyectado se almacena en el servidor. Cualquier víctima que solicita la información ejecuta el código malicioso.
- **Basado en DOM:** Este riesgo se produce en el lado del cliente y modifica el entorno DOM en el navegador. La respuesta HTTP no cambia, pero el código que se ejecuta en el navegador (lo que ve la víctima) si esta modificado.

Cómo prevenirlo

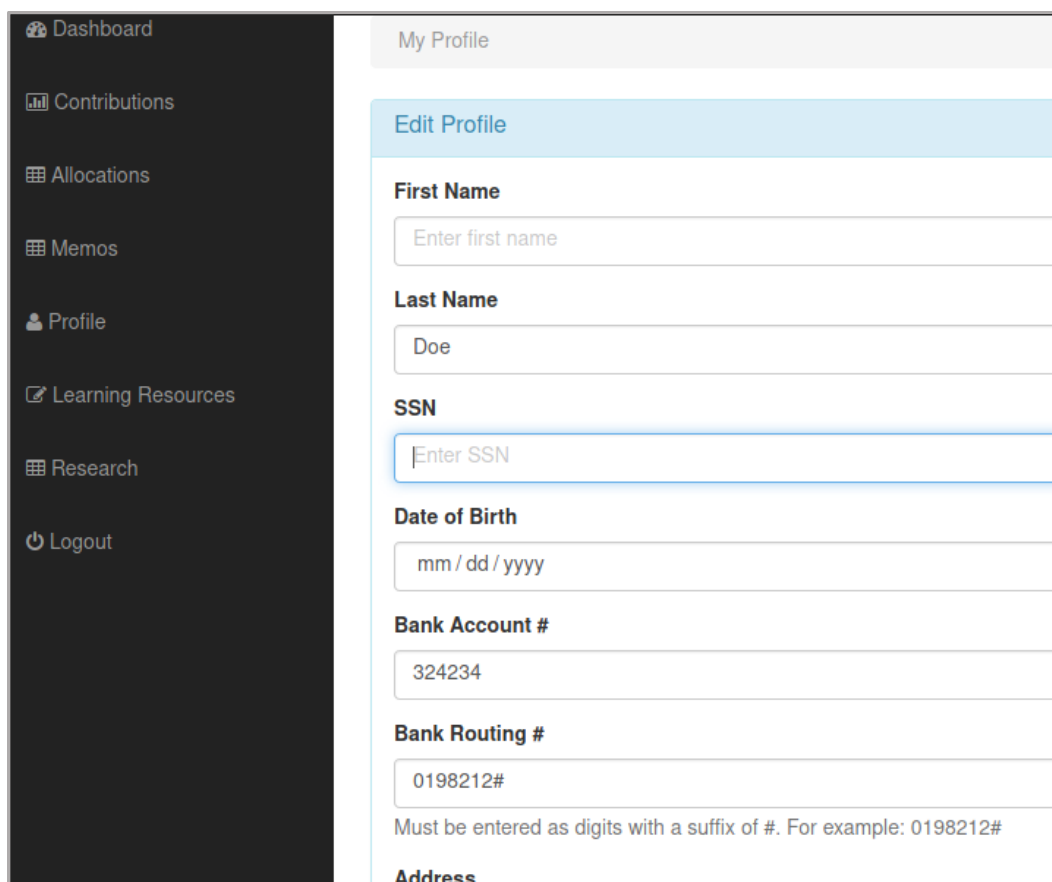
Para poder prevenir esta vulnerabilidad se aconseja mantener los datos no confiables separados del contenido activo del navegador.

- El uso de framework que escape¹² automáticamente el XSS.
- Codificación de JavaScript antes de insertar datos que no son de confianza en valores de datos de JavaScript
- Codificación a HTML antes de insertar los datos que no son de confianza en un elemento HTML.
- Codificación a CSS y validación antes de insertar datos que no sean de confianza en valores de propiedad de estilo HTML
- Evitar el uso de JavaScript en la URL

¹² Escapar: convertir o marcar caracteres clave de los datos para evitar que se interprete en un contexto peligroso y el navegador no ejecute.

Explotación de la vulnerabilidad en Node Goat

NodeGoat es vulnerable frente ataques XSS, para esto accederemos como usuarios normales a la aplicación web. Una vez dentro de la aplicación, seleccionamos *Profile* para poder modificar el perfil.



Dashboard

Contributions

Allocations

Memos

Profile

Learning Resources

Research

Logout

My Profile

Edit Profile

First Name

Enter first name

Last Name

Doe

SSN

Enter SSN

Date of Birth

mm / dd / yyyy

Bank Account #

324234

Bank Routing #

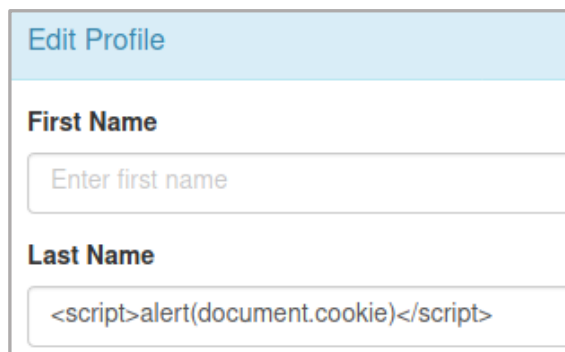
0198212#

Must be entered as digits with a suffix of #. For example: 0198212#

Address

Ilustración 51 Sección Profile para actualizar el perfil del usuario

Donde haremos el ataque XSS almacenado será en el campo del apellido, esto solo afectará al propio usuario y a todos aquellos que puedes obtener su apellido, ya que el navegador ejecuta el código malicioso que está como apellido. Primero probaremos con la obtención de la cookie de sesión.



Edit Profile

First Name

Enter first name

Last Name

`<script>alert(document.cookie)</script>`

Ilustración 52 Código malicioso para saber la cookie de sesión

Una vez acceda un usuario que pueda ver el apellido del usuario 1, le saldrá una alerta con su cookie de sesión. Con esta respuesta los atacantes pueden enviarla a un servidor malicioso y poder almacenar el ID de sesión para así realizar un secuestro de sesión o *Hijacking*.

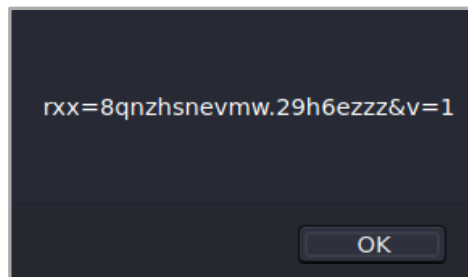


Ilustración 53 Cookie de sesión del usuario

También a través del siguiente código malicioso podemos hacer que la aplicación no funcione correctamente impidiendo así a los usuarios que puedan observar el apellido, los redireccione a otra página impidiendo así el acceso a la aplicación.

Edit Profile

First Name

Last Name

Ilustración 54 Código malicioso para redireccionar a otro sitio web

Una vez confirmado la actualización del perfil realizando así el ataque pondremos la siguiente situación para explicar el funcionamiento del Código malicioso mencionado anteriormente:

Cuando el administrador accede a la aplicación, su página inicial contiene los usuarios incluyendo el apellido del usuario que contiene el código malicioso.

Benefits Start Date			
Employee ID	First Name	Last Name	Benefits Start Date
2	John	Doe	01 / 10 / 2030
3	Will	Smith	11 / 30 / 2025

Ilustración 55 Página de inicio del administrador

Entonces una vez inyectamos el código malicioso, cuando el administrador acceda, se ejecutará el código redireccionando a otro sitio web impidiéndole así el acceso a la aplicación.

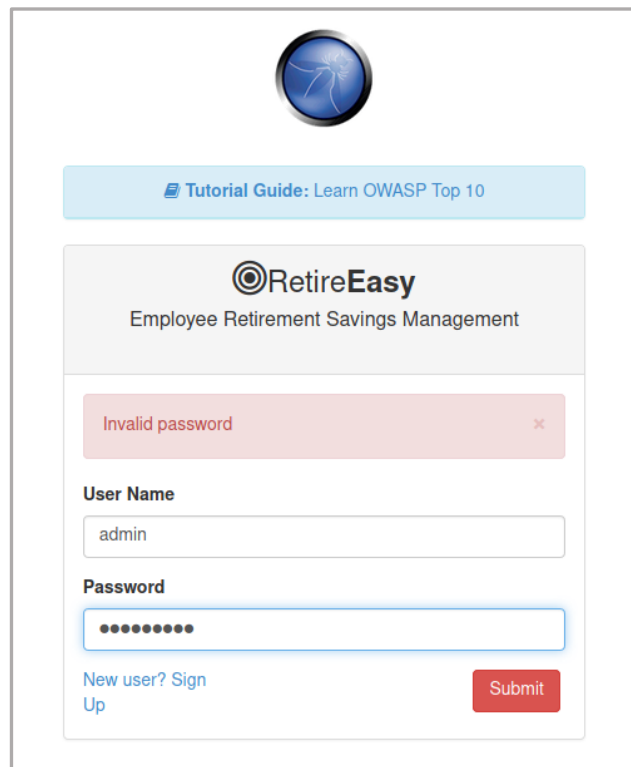


Ilustración 56 Acceso del usuario como administrador

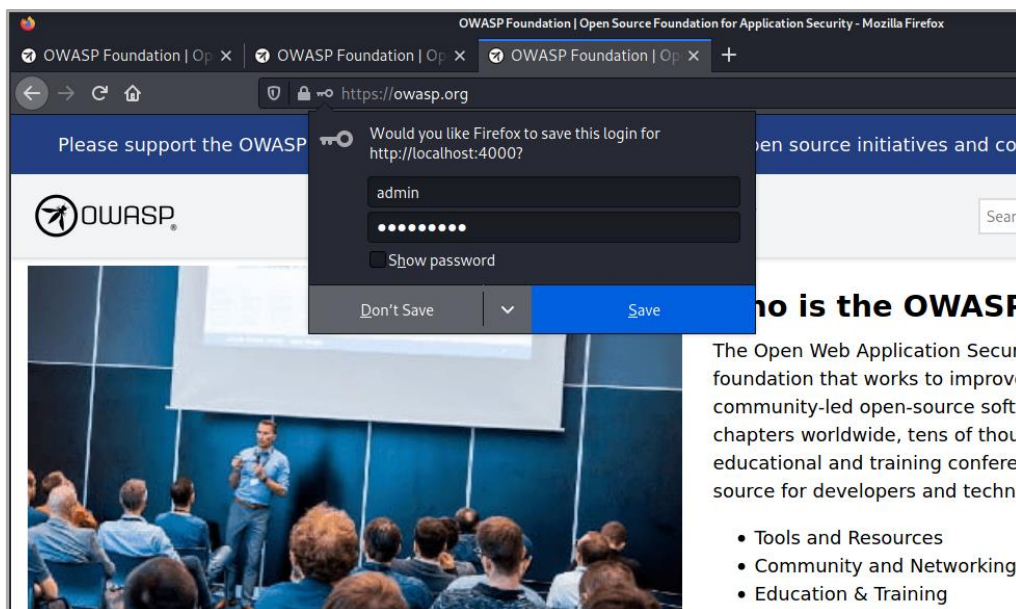


Ilustración 57 Sitio web a la que será redireccionado el administrador

A4 - Referencia directa a objetos inseguros (DOR)

La referencia directa a objetos inseguros (DOR) se produce cuando para acceder a algunos recursos como archivos, directorio o bases de datos no se realiza ninguna verificación de control de acceso, permitiendo así los atacantes modificar estas referencias para acceder a los datos no autorizados.

Cómo prevenirlo

- Prueba y análisis de código: comprobar si la aplicación verifica bien.
- Utilizar referencias de objetos indirectas por usuario o sesión.
- Comprobar el acceso de los usuarios y sus permisos.

Explotación de la vulnerabilidad en Node Goat

Accederemos a la aplicación web como un usuario normal. Una vez autenticados, accederemos a la pestaña de *allocations*. Al acceder al menú nos fijamos en la URL la cual contiene un número que podemos entender que es el id del usuario.

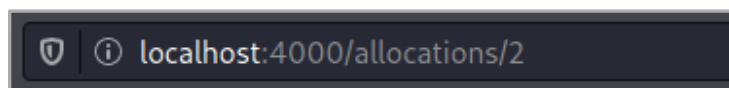


Ilustración 58 URL de la aplicación web a modificar

Si modificamos el número 2 por el número 3, podemos observar que nos direcciona a la información de otro usuario.

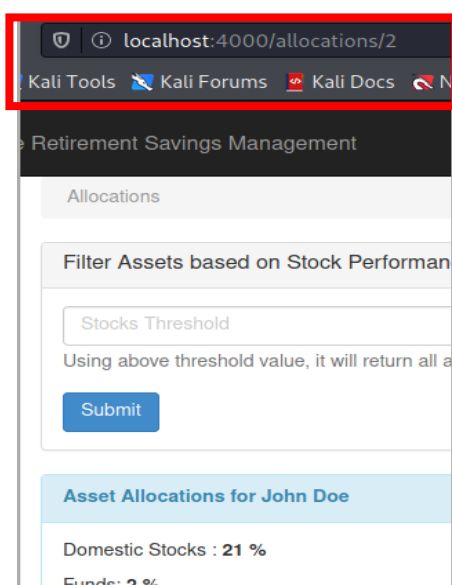


Ilustración 59 Información del id de usuario 2

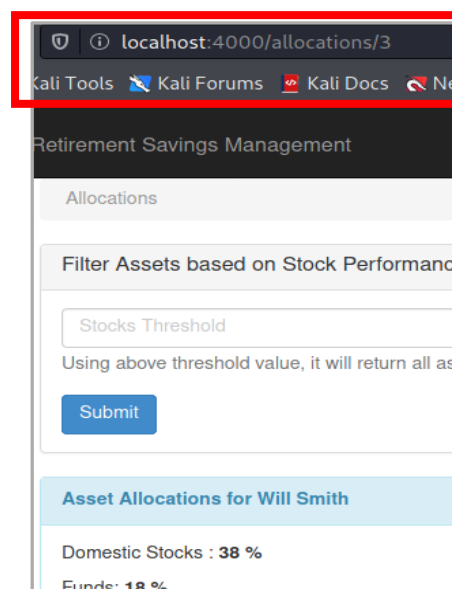


Ilustración 60 Información del id de usuario

A5 - Configuración de seguridad incorrecta

La configuración de seguridad incorrecta ocurre a cualquier nivel de la aplicación; plataforma, servidor, servidor web, base de datos, framework, redes, etc. A través de esta vulnerabilidad, los atacantes pueden tener acceso a lugares o funciones no autorizadas. Esto puede ocurrir cuando:

- Usas software sin actualizar (S.O, aplicaciones, DBMS¹³, etc).
- Características habilitadas innecesarias: acceder a la bd por defecto como administrador, puertos abiertos sin uso, cuentas por defecto, etc.
- Contraseñas por defecto.
- Cuando el manejo de errores revela el funcionamiento de la aplicación o información.

¿Cómo prevenirlo?

- Tener un control de versiones de las librerías, softwares, etc.
- Realizar auditorías y escaneos para detectar fallos de configuración o parches omitidos.
- Tener la aplicación planeada y estructurada correctamente para proporcionar una separación segura y efectiva entre los componentes de la aplicación.
- Deshabilitar la opción *Directory Listing*.

¹³ DBMS (**D**atabase **M**anagement **S**ystem) o sistema gestor de base de datos es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos

Explotación de la vulnerabilidad en Node Goat

Al analizar las cabeceras podemos ver que nos indica la cookie de sesión de usuario.

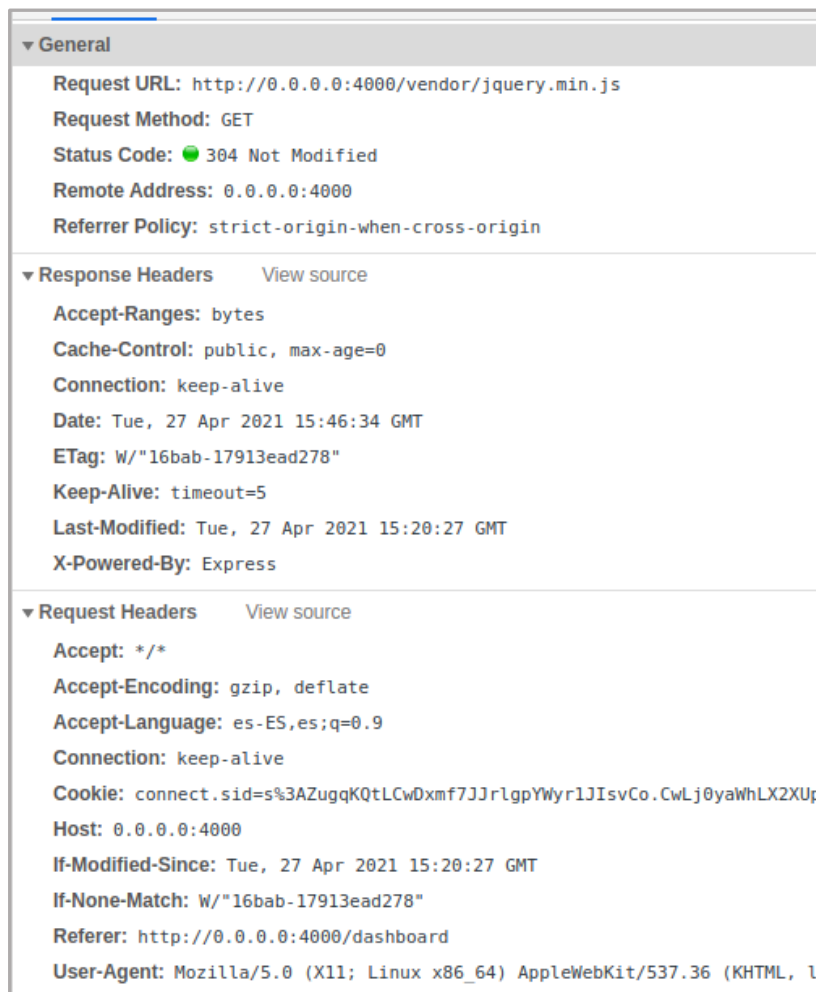


Ilustración 61 Cabecera de petición al servidor

```
Cookie: rxx=8qzhsnevmw.29h6ezzz&v=1; connect.sid=s%3AQcpuwXyblp-IkUfPCid  
Pl.qiL4qTBE2JliQfYz6WDbj%2BjEdAtQBE0cbj4AI4xn%2Fck
```

Ilustración 62 Id de sesión visible

Como podemos ver se muestra la cookie en la cabecera de la petición, la cual puede ser “robada” usando técnicas XSS. También si accedemos al directorio */Tutorial* y consultamos un recurso que no esté disponible obtener archivos necesarios para la aplicación.

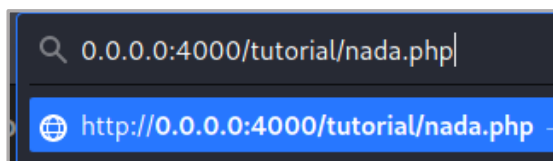


Ilustración 63 Solicitud de recurso no existente

```

1 <!doctype HTML>
2 <html>
3
4 <head>
5   <title>Internal Error</title>
6 </head>
7
8 <body>
9
10   Oops..
11   <br>Error: Cannot find module 'php'
12 Require stack:
13 - /home/node/app/node_modules/express/lib/view.js
14 - /home/node/app/node_modules/express/lib/application.js
15 - /home/node/app/node_modules/express/lib/express.js
16 - /home/node/app/node_modules/express/index.js
17 - /home/node/app/server.js
18 </body>
19
20 </html>

```

Ilustración 64 Respuesta al solicitar el recurso no disponible

Otros ejemplos:

- Mostrar la versión del servidor que alberga la aplicación.

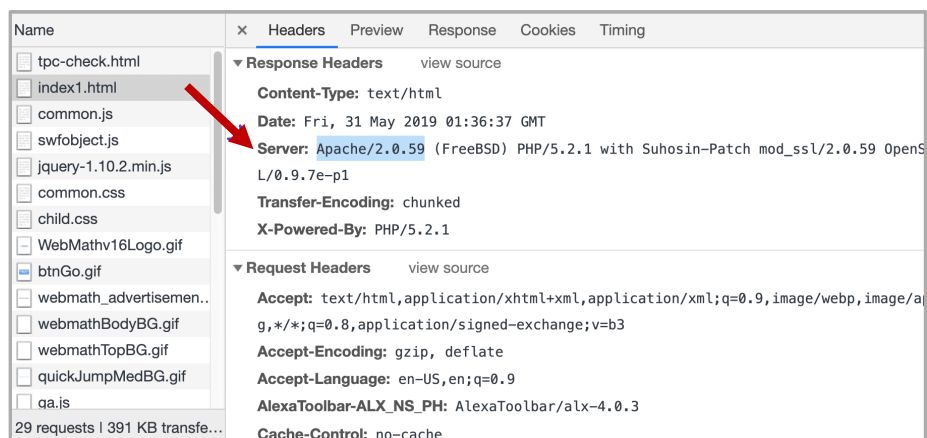


Ilustración 65 Versión del servidor web

- Mostrar los documentos el directorio al solicitar un recurso no disponible.

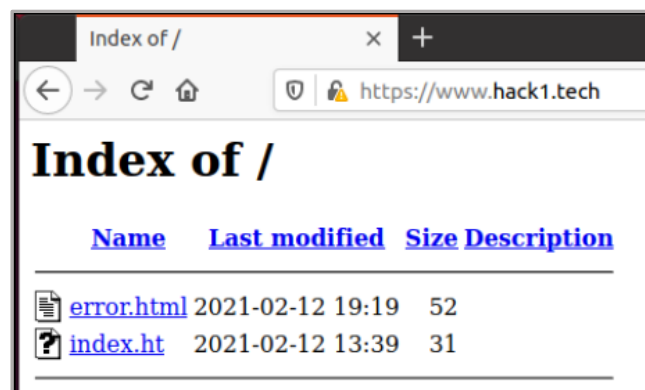


Ilustración 66 Directorio de la aplicación web

A6 – Exposición de datos sensibles

Muchas de las aplicaciones web contienen esta vulnerabilidad que simplemente es que no usan funciones de cifrado y los datos viajan en texto claro, permitiendo a posibles atacantes poder robar o modificar tales datos para llevar a cabo actos criminales como fraudes, robo de identidad, alteración de los datos, etc a través de técnicas, como por ejemplo *man in the Middle*.

Cómo prevenirlo

- Al conectarse a otra máquina usar software de cifrado como SSH.
- Usar páginas que tengan el certificado HTTPS (candado verde).
- Cifrar datos sensibles almacenados o durante el tráfico de red.
- No almacenar datos sensibles innecesariamente.
- Uso de algoritmos de cifrado seguros y contraseñas seguras.
- Deshabilitar el autocompletado de los formularios y cacheado de páginas.

Explotación de la vulnerabilidad en Node Goat

Para explotar la vulnerabilidad A6: Exposición de datos sensibles en Node Goat analizaremos el tráfico de red cuando un usuario se autentifica. En primer lugar, usaremos un analizador de tráfico, Wireshark en este caso y usaremos un filtro HTTP para los paquetes que se intercambien entre el servidor web y el navegador.

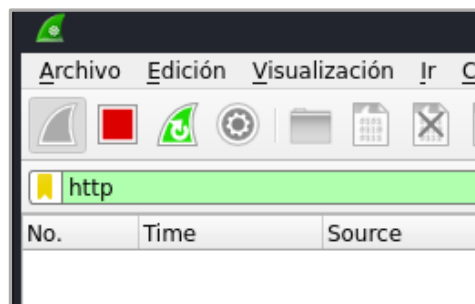


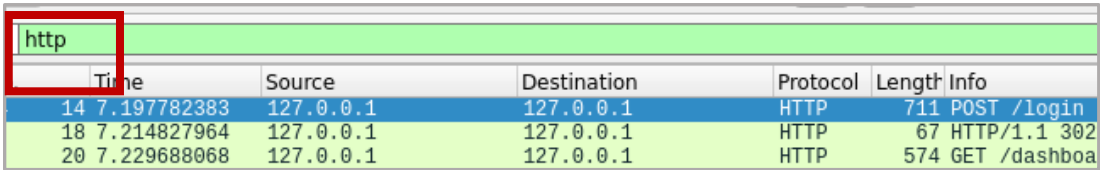
Ilustración 74 Fijar filtro para paquetes HTTP

Una vez estemos analizando el tráfico iniciaremos sesión en la aplicación.

A screenshot of the 'RetireEasy' login page. The page has a light gray header with the 'RetireEasy' logo and the text 'Employee Retirement Savings Management'. Below the header is a white box containing the login form. The form has two sections: 'User Name' with a text input field containing 'user1', and 'Password' with a password input field represented by a series of dots.

Ilustración 67 Login de inicio con user1

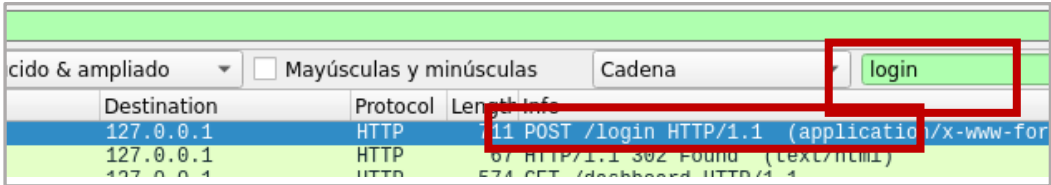
Una vez hayamos iniciado login comenzará la comunicación entre el navegador y el servidor web.



	Time	Source	Destination	Protocol	Length	Info
14	7.197782383	127.0.0.1	127.0.0.1	HTTP	711	POST /login
18	7.214827964	127.0.0.1	127.0.0.1	HTTP	67	HTTP/1.1 302
20	7.229688068	127.0.0.1	127.0.0.1	HTTP	574	GET /dashboa

Ilustración 68 Filtro HTTP

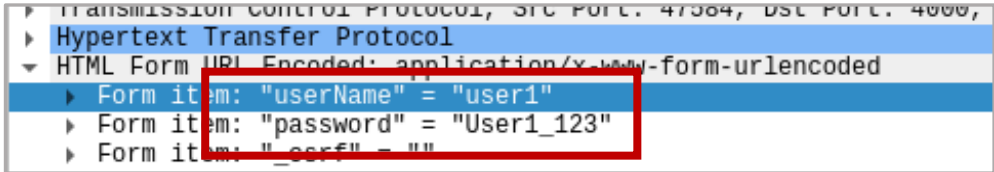
Para localizar el paquete que contenga las credenciales podemos usar otro filtro para facilitar la búsqueda.



	Destination	Protocol	Length	Info
14	127.0.0.1	HTTP	711	POST /login HTTP/1.1 (application/x-www-form-urlencoded)
18	127.0.0.1	HTTP	67	HTTP/1.1 302 Found (text/html)
20	127.0.0.1	HTTP	574	GET /dashboard HTTP/1.1

Ilustración 69 Filtro con login para localizar paquete

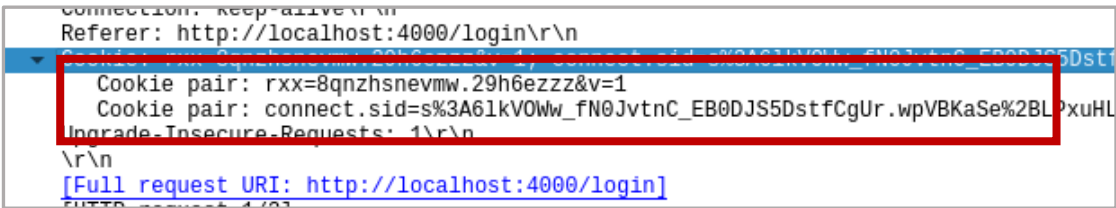
Una vez localizado el paquete correspondiente, si lo analizamos podemos ver que tanto como el usuario como las contraseña están es texto claro.



Transmission Control Protocol, Src Port: 47584, Dst Port: 4000
Hypertext Transfer Protocol
HTML Form URL Encoded: application/x-www-form-urlencoded
Form item: "userName" = "user1"
Form item: "password" = "User1_123"
Form item: "_csrf" = ""

Ilustración 70 Credenciales del usuario

Al igual que con la vulnerabilidad A5: Configuración incorrecta, también se muestra el id de sesión de la víctima



Connection: keep-alive
Referer: http://localhost:4000/login\r\n
Cookie pair: rxx=8qzhsnevwmw.29h6ezzz&v=1
Cookie pair: connect.sid=s%3A61kVOWw_fN0JvtnC_EB0DJS5DstfCgUr.wpVBKaSe%2BLP_xuHL
Upgrade-Insecure-Requests: 1\r\n
[Full request URI: http://localhost:4000/login]

Ilustración 71 Id de sesión del usuario

A7 – Ausencia de control de acceso a las funciones.

Gran parte de las aplicaciones web actuales verifican los permisos que tienen los usuarios, pero también hay aplicaciones que permiten por una mala gestión, conceder acceso y permisos a usuarios sin autorización

Cómo prevenirlo

- Definir las políticas de control de acceso a los usuarios a la aplicación.
- Registro de intentos de acceso
- Controles de usuario y grupos. El cliente que acceda pertenezca a un grupo que no tenga permisos en el sistema.

Explotación de la vulnerabilidad en Node Goat

Para realizar el ataque a la aplicación atacaremos a la URL por fuerza bruta para obtener los directorios de la aplicación. En este caso usaremos DirBuster.

Una vez iniciemos DirBuster, tendremos que poner la dirección de la aplicación web y un diccionario para realizar la fuerza bruta.

```
(docker@kali-docker)-[~/Escritorio]
$ dirbuster
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Starting OWASP DirBuster 1.0-RC1
```

Ilustración 72 Inicio de DirBuster

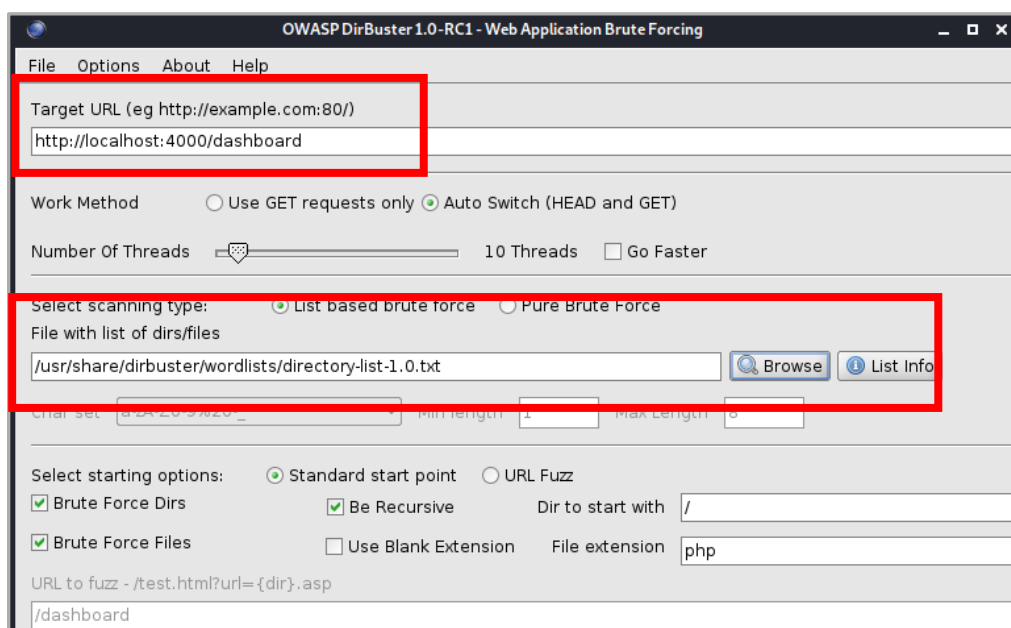
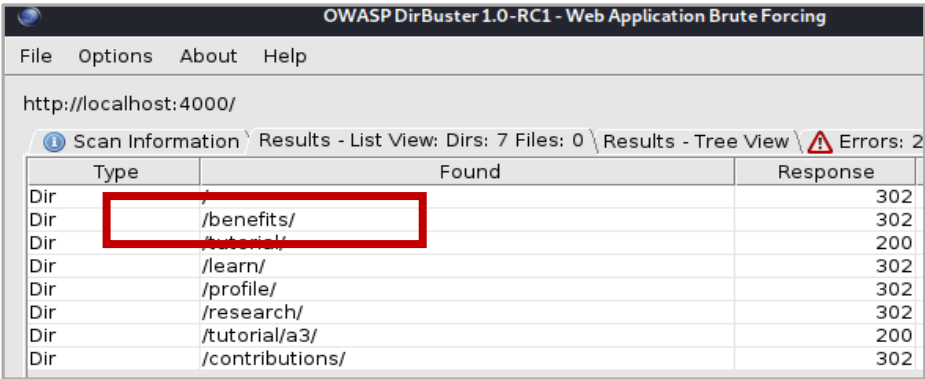


Ilustración 73 Configuración de DirBuster

Una vez realizado el ataque obtendremos los directorios de la aplicación. Como podemos ver existe el directorio *benefits* que no está disponible cuando accede un usuario normal, este directorio solo está disponible si accedes a la aplicación como administrador.



Type	Found	Response
Dir	/	302
Dir	/benefits/	302
Dir	/tutorial/	200
Dir	/learn/	302
Dir	/profile/	302
Dir	/research/	302
Dir	/tutorial/a3/	200
Dir	/contributions/	302

Ilustración 74 Directorios obtenidos por el ataque

Para terminar el ataque nos logueamos a la aplicación como un usuario normal y modificaremos la URL añadiendo *benefits* para realizar así el cambio de directorio y comprobar si la aplicación contiene la vulnerabilidad.

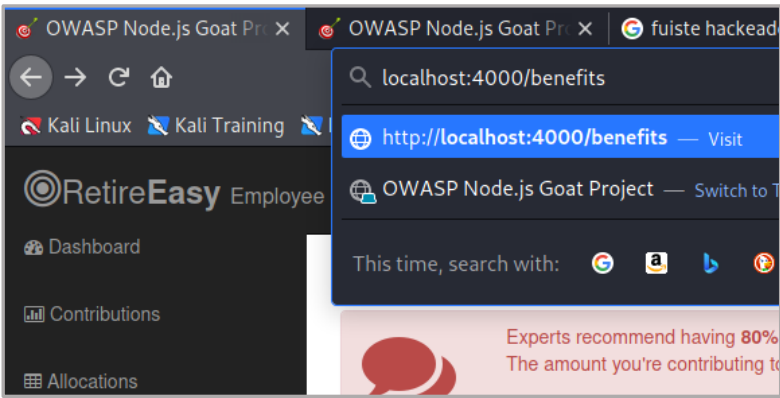
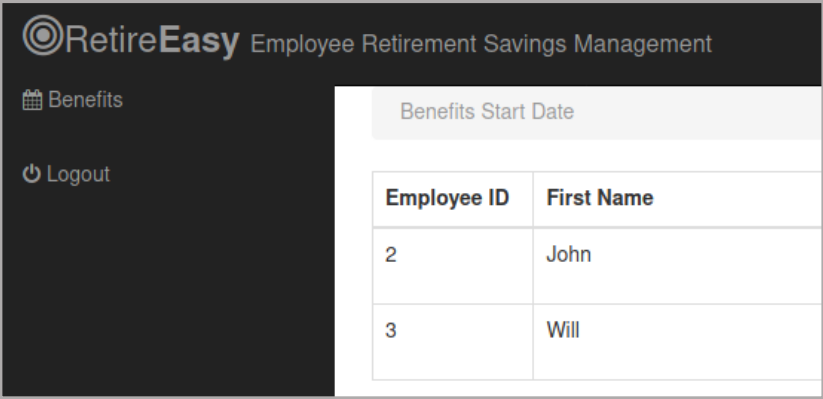


Ilustración 75 Modificación de URL

Como podemos observar obtenemos acceso al directorio *benefits* como administrador.



Benefits Start Date	
Employee ID	First Name
2	John
3	Will

Ilustración 76 Acceso al directorio benefits

A8 – Falsificación de peticiones en sitios cruzados (CSRF).

El ataque CSRF trata de que cuando una víctima está autenticada en un sistema el cual puede modificar contenido, el atacante envía una dirección hacia una aplicación comprometida. Una vez accede la víctima a la aplicación comprometida esta enviara una petición HTTP falsificada al servidor permitiendo así modificar datos de usuario de la víctima.

Esto puede ocurrir cuando los usuarios no cierran correctamente sesión y sigue activa mientras visitas otras páginas web donde puede haber código malicioso (ilustración 49), el cual se ejecutará automáticamente o al realizar una acción.

Cómo prevenirlo.

El usuario para prevenir este ataque puede realizar estas acciones:

- Configurar el navegador para que no recuerde usuarios ni contraseñas.
- Navegar con el Modo Incógnito.
- Cerrar sesión
- No utilizar la función de mantener la sesión abierta de la aplicación web.
- Utilizar diferentes navegadores, uno para ocio y otro para gestiones.
- Utilizar extensiones para bloquear la ejecución scripts (Script Defender).

Para el desarrollador de aplicaciones puede realizar las siguientes acciones:

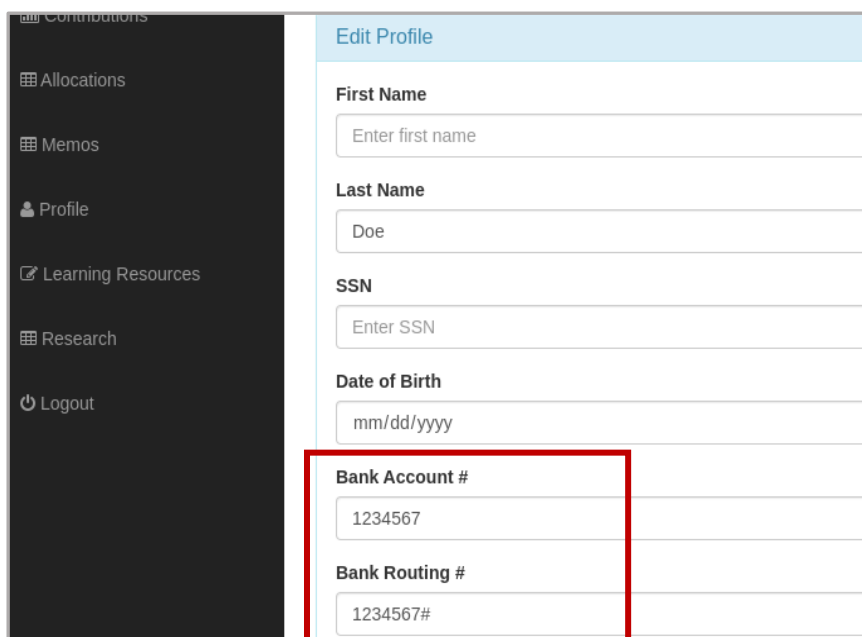
- Incluir en cada solicitud HTTP un token no predecible y único por cada sesión. Se aconseja que este token este en un campo oculto para evitar así la inclusión de URL.
- Requerir que el usuario devuelva a autenticar o usar otras pruebas de autenticación como CAPTCHA.
- También existen herramientas las cuales permite implementar el token automáticamente como por ejemplo CSRFGuard¹⁴, ESAPI¹⁵, [Csurf](#), etc.

¹⁴ CSRFGuard es una biblioteca que implementa una variante del patrón de token del sincronizador para mitigar el riesgo de ataques de falsificación de solicitudes entre sitios (CSRF)

¹⁵ ESAPI es una biblioteca de control de seguridad de aplicaciones web de código abierto y gratuita que facilita a los programadores la escritura de aplicaciones de menor riesgo.

Explotación de la vulnerabilidad en Node Goat

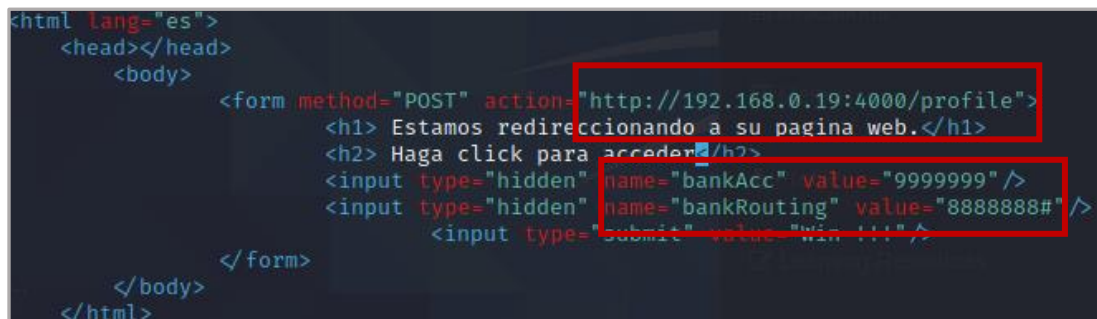
El usuario está autenticado en la aplicación y se le enviará una dirección para que acceda a ella y realizar el ataque. El código malicioso le modificara *Bank Account* y *Bank Routing*.



Edit Profile	
First Name	<input type="text" value="Enter first name"/>
Last Name	<input type="text" value="Doe"/>
SSN	<input type="text" value="Enter SSN"/>
Date of Birth	<input type="text" value="mm/dd/yyyy"/>
Bank Account #	<input type="text" value="1234567"/>
Bank Routing #	<input type="text" value="1234567#"/>

Ilustración 77 Perfil de la víctima

El servidor malicioso tendrá una aplicación web con un código malicioso.



```
<html lang="es">
  <head></head>
  <body>
    <form method="POST" action="http://192.168.0.19:4000/profile">
      <h1> Estamos redireccionando a su pagina web.</h1>
      <h2> Haga click para acceder.</h2>
      <input type="hidden" name="bankAcc" value="9999999" />
      <input type="hidden" name="bankRouting" value="8888888#" />
      <input type="submit" value="win !!!" />
    </form>
  </body>
</html>
```

Ilustración 78 Aplicación web con el código malicioso

Una vez acceda la víctima a la aplicación maliciosa y de click al botón de Acceder, se ejecutará el código malicioso que enviará una solicitud HTTP falsificada a la aplicación web.

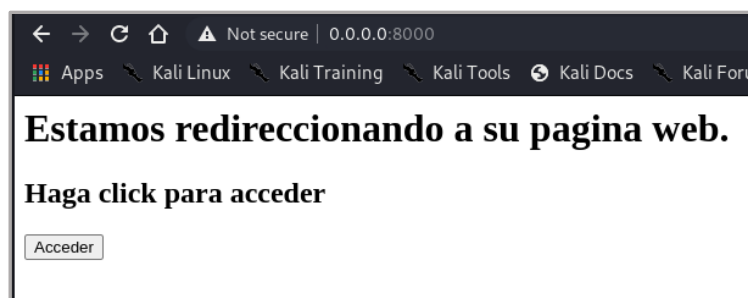


Ilustración 79 Servidor web malicioso

Una vez realizado el ataque, el perfil del usuario está modificado.

RetireEasy

Employee Retirement Savings Management

Dashboard

Contributions

Allocations

Memos

Profile

Learning Resources

Research

Logout

My Profile

Profile updated successfully.

Edit Profile

First Name

Enter first name

Last Name

Enter last name

SSN

Enter SSN

Date of Birth

mm/dd/yyyy

Bank Account #

9999999

Bank Routing #

8888888#

Must be entered as digits with a suffix of 0001-9999

Ilustración 80 Información de la víctima ya modificada

A9 – Uso de componentes con vulnerabilidades conocidas

Algunos frameworks, librerías y otros componentes se ejecutan con los mismos permisos que la propia aplicación web. Si uno de estos componentes contiene una vulnerabilidad y consigue explotarse, puede poner en riesgo a la aplicación web permitiendo así obtener acceso datos sensibles o al funcionamiento del servidor.

Cómo prevenirlo.

- Remover dependencias, funcionalidades, componentes, archivos y documentación innecesaria y no utilizada.
- Uso de herramientas o software para el mantenimiento de versiones de los componentes de la aplicación web.
- Monitorizar base de datos sobre vulnerabilidades como [CVE](#) sobre los componentes utilizados.
- Realizar escaneos automatizados para análisis de vulnerabilidades como OWASP ZAP, Nessus, Nikto, etc.
- Obtener los componentes de páginas oficiales y no de páginas no confiables.

Explotación de la vulnerabilidad en Node Goat

En Node Goat la vulnerabilidad trataba sobre el uso de una versión insegura de la biblioteca Marked. Lo que ocurría con esta librería era que en la sección de *Memos* sea vulnerable a XSS, pero al parecer en NodeGoat actualizaron Marked a una nueva versión 0.3.9 corrigiendo así la vulnerabilidad que tenía NodeGoat (Información obtenida de un post de [GitHub](#) y del fichero *package.json* de dependencias disponible en su [GitHub](#).)

```
version": "2.0.0",
"description": "A tool to learn OWASP Top 10 for node.js developers",
"main": "server.js",
"dependencies": {
  "bcrypt-nodejs": "0.0.3",
  "body-parser": "^1.15.1",
  "consolidate": "^0.14.1",
  "csurf": "^1.8.3",
  "dont-sniff-mimetype": "^1.0.0",
  "express": "^4.13.4",
  "express-session": "^1.13.0",
  "forever": "^2.0.0",
  "helmet": "^2.0.0",
  "marked": "0.3.9",
  "mongodb": "^2.1.18",
  "needle": "2.2.4",
}
```

Ilustración 81 Archivo package.json

Tras analizar con OWASP ZAP descubrimos que NodeGoat usa librerías no actualizadas de JQuery y Bootstrap las cuales contienen vulnerabilidades conocidas.

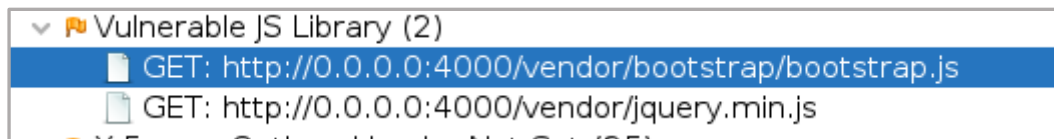


Ilustración 82 Análisis de ZAP

Si queremos más información sobre las vulnerabilidades obtenidas, podemos obtenerla en base de datos como <https://cve.mitre.org/> o <https://www.cvedetails.com/>

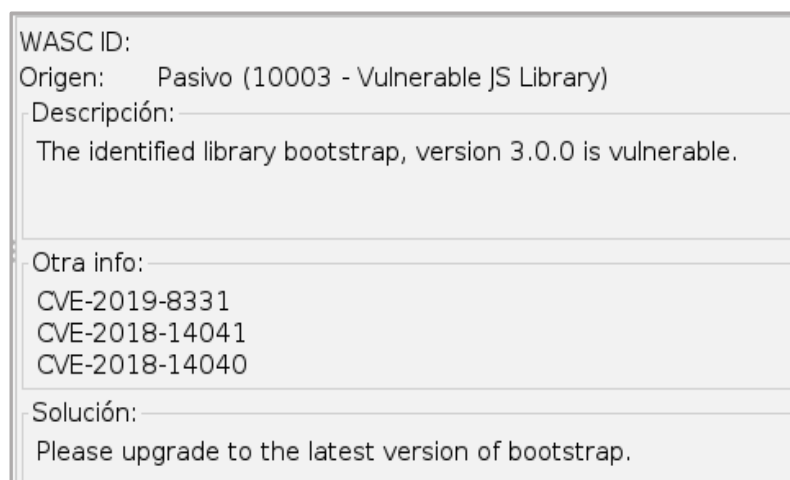


Ilustración 83 Vulnerabilidad de Bootstrap versión anticuada



Ilustración 84 Vulnerabilidad de JQuery versión anticuada

A10 – Redirecciones y reenvíos no validados

Las aplicaciones web habitualmente redirigen y reenvían a usuarios a otras páginas web o sitios. Cuando la entrada para reenviar y redireccionar no está validada, se puede redireccionar a los usuarios a un sitio web o página que no es de confianza permitiendo llevar a cabo un ataque de *phishing*, robo de credenciales de usuario, ataque CSRF, etc. Dado que el nombre de servidor no se modifica puede dar más confianza a los usuarios.

Cómo prevenirlo.

- Evitar el uso de direccionamiento y reenvíos
- No usar parámetros de redireccionamiento que pueda modificar el usuario
- Obligar a todos los redireccionamientos a pasar primero por una página notificando a los usuarios que están abandonando su sitio y dar clic en un enlace para confirmar.

Explotación de la vulnerabilidad en Node Goat

Daremos botón derecho sobre *Learning Sources* y copiaremos el link. Una vez obtenido la dirección, le modificaremos el redireccionamiento por el sitio web atacante

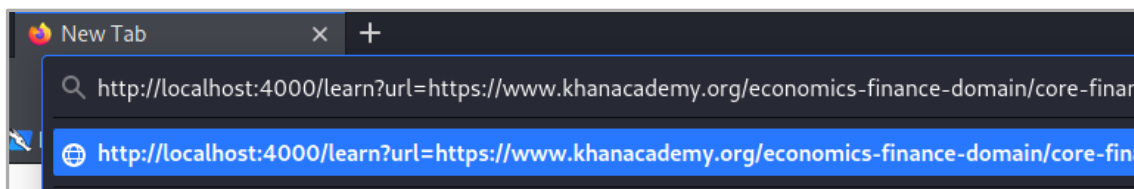


Ilustración 85 Redireccionamiento que realiza la aplicación por defecto

Una vez modificada la URL enviaremos la dirección con la ayuda de técnicas de “ingeniería social”¹⁶ modificada a la víctima.

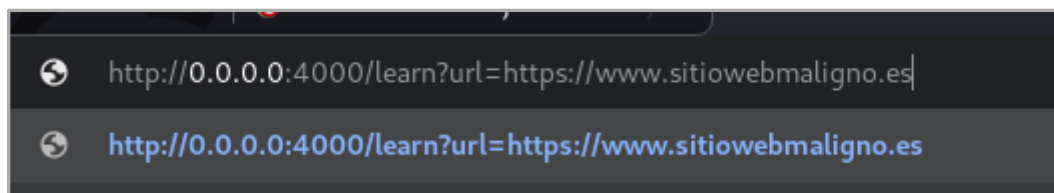


Ilustración 86 URL modificada

¹⁶ La ingeniería social es la práctica de obtener información confidencial a través de la manipulación de usuarios legítimos

4. Securitización de vulnerabilidades

Para finalizar este proyecto, se realizará la resolución de algunas de las vulnerabilidades comentadas en el punto 3- Explicación y explotación de las vulnerabilidades.

A1.1 - Inyección para acceder a archivos

Al utilizar la función *eval* para evaluar la entrada del usuario, puede llegar a producir una entrada a la inyección de código ya que no evalúa la cadena sino que la devuelve. Para solucionar esto se utiliza la función *parseInt* que permite devolver un número entero ya que es lo que necesita la aplicación para funcionar.

```
const preTax = eval(req.body.preTax);
const afterTax = eval(req.body.afterTax);
const roth = eval(req.body.roth);
```

Ilustración 87 Aplicación usando la función eval

```
const preTax = parseInt(req.body.preTax);
const afterTax = parseInt(req.body.afterTax);
const roth = parseInt(req.body.roth);
```

Ilustración 88 Aplicación usando la función parseInt

Ahora realizamos la inyección de código para comprobar que el uso de la función no permite la inyección de código.

	New Payroll Contribution Percent (per pay period)
	<input type="text" value="readdirSync ('.'). toString ()) %"/>

Ilustración 89 Inyección de código

Al inyectar el código y confirmar nos saldrá un error de que el valor dado no es válido solución la vulnerabilidad de inyección de código.

Invalid contribution percentages

Ilustración 90 Error al inyectar código no válido

A3 - Secuencia de comandos en sitios cruzados (XSS)

Podremos solucionar la falla de XSS o Cross-Site-Scripting habilitando la API [Swig](#). Esta permite el auto-escapeado de entradas HTML no seguras. Para ello debemos iniciar la dependencia y activar el auto-escapeado.


```
const consolidate = require("consolidate");  
const swig = require("swig");  
const helmet = require("helmet");  
const MongoClient = require("mongodb").Mongo
```

Ilustración 91 Inicio de la API Swig

```
swig.setDefaults({  
  autoescape: true  
});
```

Ilustración 92 Activar auto-escapeado

Si volvemos a realizar la explotación de la vulnerabilidad podemos observar que cuando se actualice el perfil, no se ejecutará el código ya que será “escapeado” evitando así la ejecución del código y será interpretado como texto.

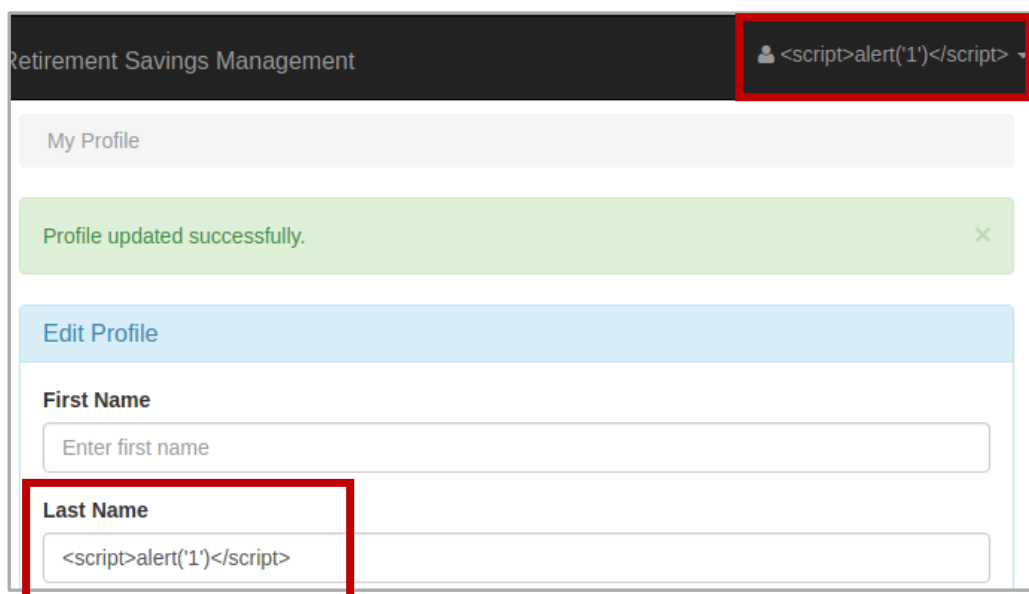


Last Name

`<script>alert('1')</script>`

SSN

Ilustración 93 Inyección de código XSS



Retirement Savings Management

Profile updated successfully.

Edit Profile

First Name

Enter first name

Last Name

`<script>alert('1')</script>`

Ilustración 94 Inyección de código sin éxito

A4 - Referencia directa a objetos inseguros (DOR)

Para evitar que otros usuarios puedan acceder a recursos a los que no tienen permiso deberemos sustituir el modo de acceso a estos, para ello iremos a la ruta *Nodegoat/app/router* y modificaremos el archivo *allocations.js*. La modificación consta de en vez de obtener la identificación a través de la URL, obtenerla por la autenticación de sesión del usuario.

```
const {
  userId
} = req.params;
const {
  threshold
} = req.query
allocationsDAO.getByIdAndThreshold
```

Ilustración 95 Acceso a través de URL

```
const {
  userId
} = req.session;
const {
  threshold
} = req.query
allocationsDAO.getByIdAndThreshold
```

Ilustración 96 Acceso a través de la autenticación del usuario

Una vez modificado el parámetro, si realizamos la explotación de la vulnerabilidad podemos observar que, al modificar el parámetro de la URL, no se muestra la información del usuario.

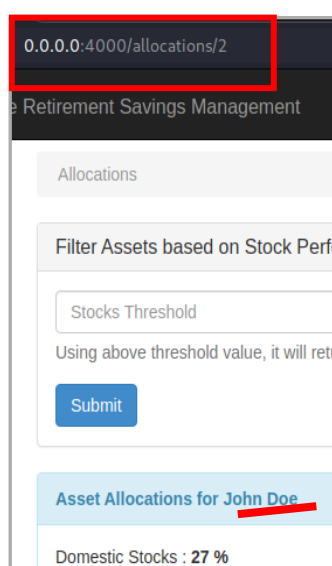


Ilustración 97 Datos usuario autenticado

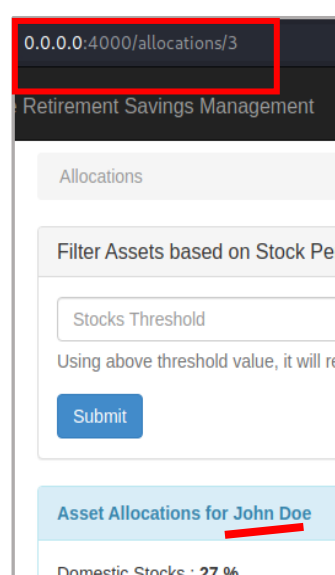


Ilustración 98 Intento de explotación 1

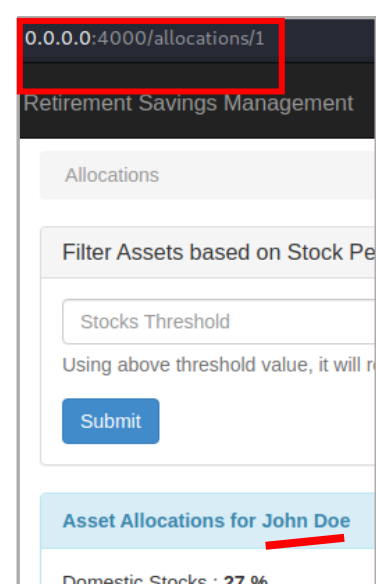


Ilustración 99 Intento de explotación

A6 – Exposición de datos sensibles

Para evitar que mediante un analizador de tráfico se pueda ver la información que viaja hacia la aplicación web, debemos utilizar el protocolo HTTPS en nuestra aplicación, que permitirá cifrar los datos que estén en texto claro. Para ello en primer lugar crearemos un certificado con openssl.

```
docker@kali-docker)~[/Descargas/NodeGoat]
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout artifacts/cert/ssl.key -out artifacts/cert/ssl.crt
```

Ilustración 100 Creación de certificado ssl

Una vez creado el certificado deberemos especificar las rutas donde está el certificado y habilitar el uso del protocolo HTTPS.

```
const https = require('https');
const fs = require('fs');

const options = {
  key: fs.readFileSync('artifacts/cert/ssl.key'),
  cert: fs.readFileSync('artifacts/cert/ssl.crt')
};
```

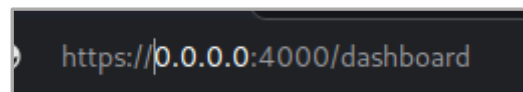
Ilustración 108 Ruta de certificado ssl

También levantaremos un servidor HTTPS que utilice el certificado creado anteriormente. En este caso solo se podrá acceder a la aplicación por HTTPS.

```
https.createServer(options, app).listen(port, () => {
  console.log(`Express http server listening on port ${port}`);
});
```

Ilustración 101 levantar servidor HTTPS

Una vez guardado los cambios y levantada la aplicación, en nuestro navegador pondremos la URL <https://0.0.0.0:4000> para poder acceder a la aplicación.



https://0.0.0.0:4000/dashboard

Ilustración 102 Dirección de la aplicación web

Una vez dentro de la aplicación nos saldrá el candado asegurando de que nuestra aplicación maneja la información cifrándola.

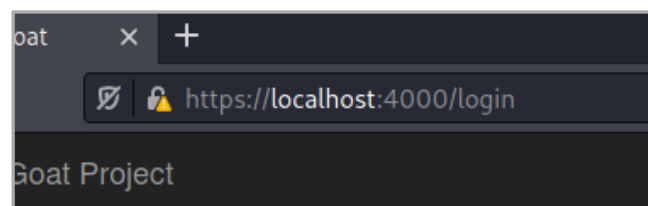


Ilustración 103 Candado de seguridad https

Para comprobar que la información viaja cifrada, nos logueamos en la aplicación web y a la vez analizaremos el tráfico de red que va hacia la aplicación. Si realizamos los mismos pasos en que la explotación vista anteriormente podemos ver que al utilizar filtros no obtenemos ningún resultado corrigiendo así la vulnerabilidad A6 – Exposición de datos sensibles.

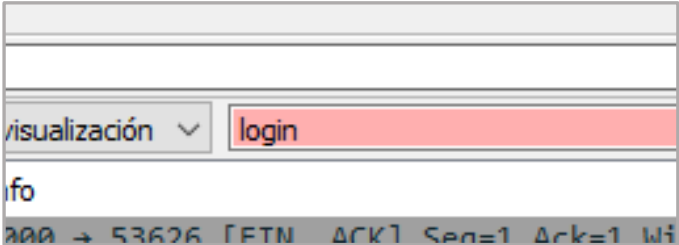


Ilustración 104 Resultado del uso de filtros

A8 – Falsificación de peticiones en sitios cruzados (CSRF).

Para poder corregir la vulnerabilidad de CSRF deberemos añadir en el archivo de descarga de dependencias la librería Csurf. Este middleware produce un token con cada petición POST y GET que es enviada a la aplicación web identificando así al usuario.

```
"description": "A tool to learn OWAS",
"main": "server.js",
"dependencies": {
  "bcrypt-nodejs": "0.0.3",
  "body-parser": "^1.15.1",
  "consolidate": "0.14.1",
  "csrf": "^1.8.3",
  "dotenv": "1.0.0",
  "express": "^4.13.4",
  "express-session": "^1.13.0",
  "forever": "^2.0.0",
```

Ilustración 105 Archivo package.json que contiene las dependencias

Una vez tengamos la librería iremos al archivo *server.js* donde agregaremos el siguiente código para poder usar Csurf y crear los tokens.

```
var csrf = require('csrf');
app.use(csrf());
app.use((req, res, next) => {
  res.locals.csrfToken = req.csrfToken();
  next();
});
```

Ilustración 106 Código de creación de tokens

También en todos los formularios de la aplicación donde deberemos agregar un campo invisible donde irá el CSRF token.

```
</div>
<input type="hidden" name="_csrf" value="{{csrfToken}}" />
```

Ilustración 107 Campo oculto en formulario

Si utilizamos la aplicación maliciosa usada anteriormente en la explotación de la vulnerabilidad, nos saldrá un error de token no válido solucionando así la vulnerabilidad.

Estamos redireccionando a su pagina web.

Haga click para acceder

Acceder

Ilustración 108 Aplicación web maliciosa

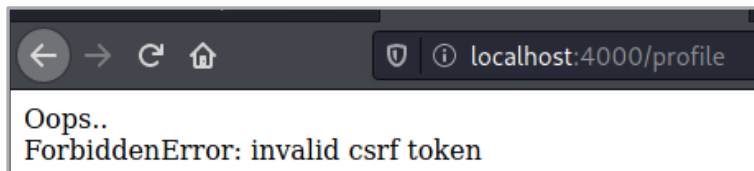


Ilustración 109 Error de generación de token

En cambio, si un usuario autenticado en la aplicación modifica su perfil podrá realizar la actualización ya que los formularios contienen un campo oculto que contiene el token generado por Csurf que será enviado al servidor validando al usuario.

```
<input type="text" class="form-control" id="bankRouting" name="bankRouting" value=""
placeholder="Enter bank routing number">
<p class="help-block">Must be entered as digits with a suffix of #. For example:
0198212# </p>
</div>
<div class="form-group">
  <label for="address">Address</label>
  <input type="text" class="form-control" id="address" name="address" value=""
placeholder="Enter address">
</div>
<div class="form-group">...</div>
<input type="hidden" name="_csrf" value="sDRCGRgb-oXK_w-BYwhsmivDu2wdf9Ed32B0"> == $
<button type="submit" class="btn btn-default" name="submit">Submit</button>
```

Ilustración 110 Campo oculto en formulario

A9 - Uso de componentes con vulnerabilidades conocidas

Utilizando un analizador de vulnerabilidades como puede ser OWASP ZAP, podemos ver en los resultados de la aplicación que hay varias librerías que no están actualizadas y estas contienen vulnerabilidades, estas son, Bootstrap y jQuery.

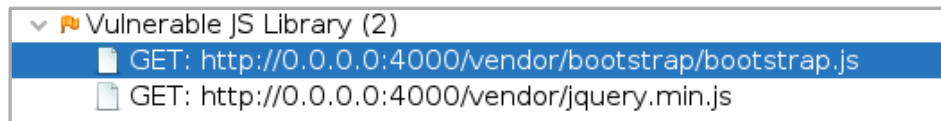


Ilustración 111 Librerías vulnerables

Para resolver esta vulnerabilidad, descargamos las nuevas versiones de las librerías de Bootstrap JS y jQuery. Una vez descargadas, para poder reemplazar las antiguas librerías, iremos a la ruta `/NodeGoat/app/assets/vendor` y sustituiremos el archivo `jquery.min.js` y para poder actualizar la librería de Bootstrap, iremos a la ruta `/NodeGoat/app/assets/vendor/Bootstrap` la cual contiene los archivos que sustituiremos para actualizar la versión.

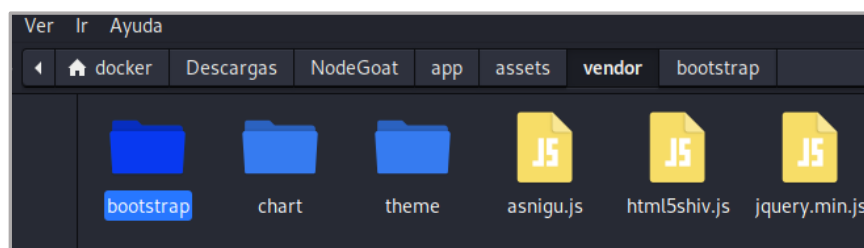


Ilustración 112 Ruta de librerías

Una vez hayamos actualizado las dos librerías con vulnerabilidades, si realizamos otro análisis con OWASP podremos observar que la vulnerabilidad A9- Uso de componentes con vulnerabilidades conocidas estará solucionada haciendo nuestra aplicación más segura.

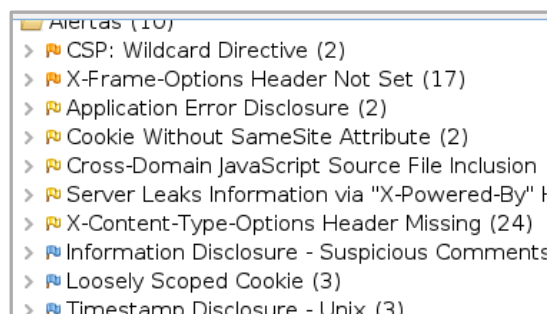


Ilustración 113 Análisis con vulnerabilidad corregida

Es posible que al actualizar algunas librerías antiguas se tenga que realizar tareas de comprobación de las aplicaciones a nivel de diseño y de funcionamiento.

A10 – Redirecciones y reenvíos no validados

Para securizar esta vulnerabilidad, deberemos modificar la redirección que hace poniendo solo la dirección de la página web a la que será redirigido el usuario. Iremos a la ruta NodeGoat/app/views/layout.html y en la línea que corresponde al enlace modificaremos la dirección.

```
' href="/profile"><i class="fa fa-user"></i> Profile</a>  
target="_blank" href="/learn?url=https://www.khanacademy.org/economics-f  
<" href="/research"><i class="fa fa-table"></i> Research</a>
```

Ilustración 114 Direccion vulnerable

```
' href="/profile"><i class="fa fa-user"></i> Profile</a>  
target="_blank" href="https://www.khanacademy.org/economics-finance-domain/core-f  
' href="/research"><i class="fa fa-table"></i> Research</a>
```

Ilustración 115 Vulnerabilidad resuelta

Una vez modificada si volvemos a la aplicación y realizamos la explotación de la vulnerabilidad copiando el enlace, podemos observar que ya no saldrá la página de nuestra aplicación, sino solo la dirección destino evitando así la modificación de la dirección permitiendo así engañar a posibles usuarios.

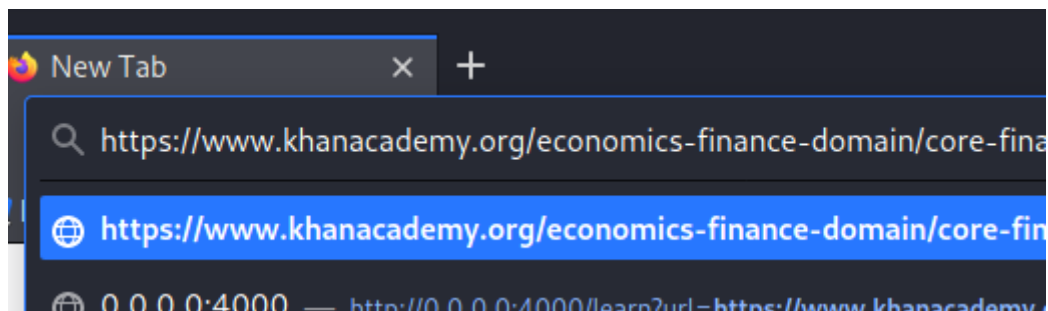


Ilustración 116 Dirección de la página web

5. Referencias

Docker

¿Qué es Docker? ¿Qué son los contenedores? y ¿Por qué no usar VMs? (2017). Platzi. <https://platzi.com/contributions/guia-del-curso-de-docker/>

Docker. (2021). Página oficial de Docker. <https://docs.docker.com/get-started/overview/>

Instalación de Docker Engine. (2021). Docker página oficial. <https://docs.docker.com/engine/install/debian/>

Instalación de Docker Compose. (2021). Docker página oficial. <https://docs.docker.com/compose/install/>

OWASP

Colaboradores de Wikipedia. (2020, 17 marzo). OWASP ZAP. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/OWASP_ZAP

OWASP TOP TEN. (2013). OWASP FOUNDATION. https://owasp.org/www-pdf-archive/OWASP_Top_10_-_2013_Final_-_Espa%C3%B1ol.pdf

OWASP TOP TEN 2017. (2017). OWASP FOUNDATION. [OWASP Top 10 - 2017](https://owasp.org/www-top10/OWASP_Top_10_-_2017.pdf)

Estándar de verificación de aplicaciones web. GitHub OWASP <https://github.com/OWASP/ASVS/tree/v4.0.2#latest-stable-version---402>

Prevención de XSS. OWASP Cheat Sheet Series. https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html#rule-7-avoid-javascript-urls

Index Alphabetical - OWASP Cheat Sheet Series. (2021). Retrieved 13 May 2021, from <https://cheatsheetseries.owasp.org/Glossary.html>

Tutorial - OWASP Node Goat Project. (2022). NodeGoat. <https://nodegoat.herokuapp.com/tutorial/a7>

NodeJS

NodeJS. (2021). NodeJS página oficial. <https://nodejs.org/es/about/>

NodeJS. (2011, 26 agosto). What is require? Node.js. <https://nodejs.org/en/knowledge/getting-started/what-is-require/>

Buna, S. (2020, 7 septiembre). Node.js Child Processes: Everything you need to know. FreeCodeCamp.Org. <https://www.freecodecamp.org/news/node-js-child-processes-everything-you-need-to-know-e69498fe970a/>

Node.js File System Module. (2021). W3schools. https://www.w3schools.com/nodejs/nodejs_filesystem.asp