Informe Semana 1 – Pentesting Web (Básico - Intermedio)

Integrantes: Alexis Chacón, Nayeli Leiva, Henry Ludeña

Fecha: 09/09/2025

Entorno: DVWA en Docker, WordPress vulnerable con plugins desactualizados y OWASP Juice Shop

Resumen:

Los laboratorios de esta semana se basan en conocer técnicas de pentesting básico con herramientas como Burp Suite, cURL, WPScan, y wfuzz. De esta manera se logra aplicar los conceptos de escaneo, análisis y modificación de datos en el sector web. Además, conocer como instalar entornos aislados para poder poner en práctica las habilidades de pentesting de forma segura y ética.

Metodología:

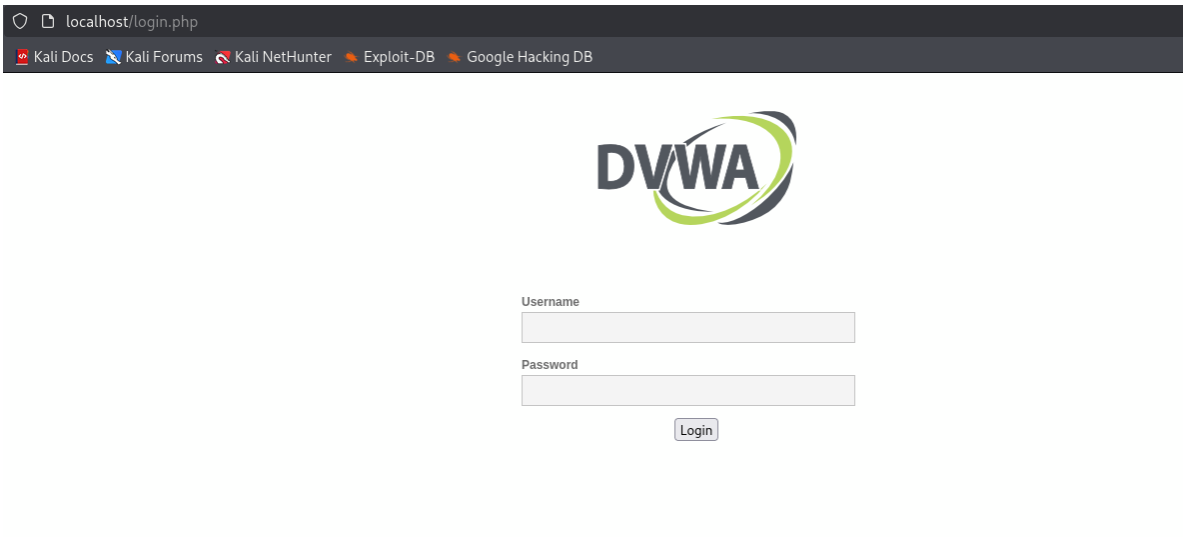
- **Despliegue de máquinas:** Corrida de máquinas Docker de DVWA y Juice Shop, y crear Docker compose para WordPress.
- **Enumeración:** Listado de plugins y usuarios de WordPress.
- **Modificación de tráfico:** Interceptación y modificación de requests HTTP; inserción de payloads
- **Fuzzing de directorios:** Ataques de fuerza bruta para enumeración de directorios web

Entorno de Pruebas:

1. Docker DVWA:

Para esta instalación se corre el comando (docker run --rm -it -p 8080:80 vulnerables/web-dvwa).

Una vez instalado se ingresa al puerto declarado (8080) y se ingresa con las credenciales por defecto (Admin/password). Luego, se selecciona Create / Reset Database para crear una nueva base de datos. Al hacerlo se debe volver a iniciar sesión con las mismas credenciales, a partir de ahí se puede escoger el nivel de dificultad (Low, Medium, Hard, Impossible).



2. WordPress

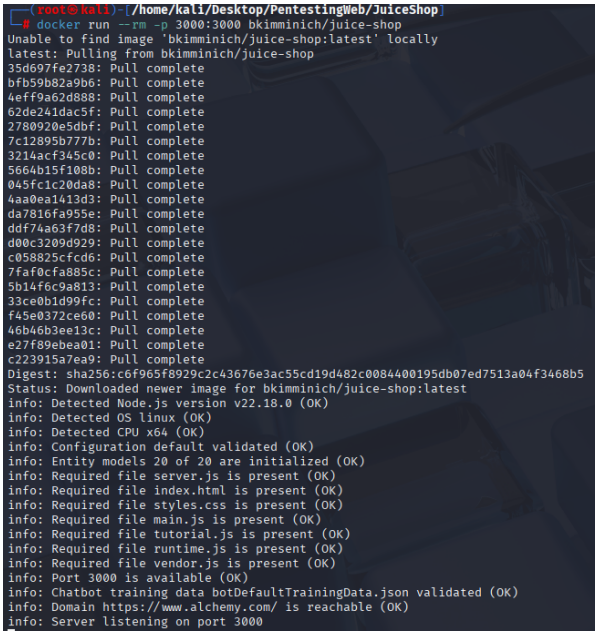
Para descargar una versión antigua de WordPress en este caso se usa un archivo .yml para automatización de descarga, la imagen de MariaDB y de WordPress en una versión antigua (5.8 que es de 2021).

```
┌──(kali㉿kali)-[~/Desktop/PentestingWeb]
└─$ cat docker-compose.yml
version: '3.3'
services:
  db:
    image: mariadb:10.5
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wp
      MYSQL_PASSWORD: wp
    ports:
      - "3307:3306"

  wordpress:
    image: wordpress:5.8-apache   #versión antigua
    restart: always
    ports:
      - "8081:80"
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wp
      WORDPRESS_DB_PASSWORD: wp
      WORDPRESS_DB_NAME: wordpress
    depends_on:
      - db
```

Posteriormente se procede a la instalación con “Docker compose up”, y se accede a la página mediante el puerto indicado (8081 en este caso).

3. Juice Shop

Se procede la instalación de Juice Shop mediante su contenedor Docker, para ello se usa el comando: docker run --rm -p 3000:3000 bkimminich/juice-shop (--rm sirve para eliminar archivos efímeros al detener el contenedor).

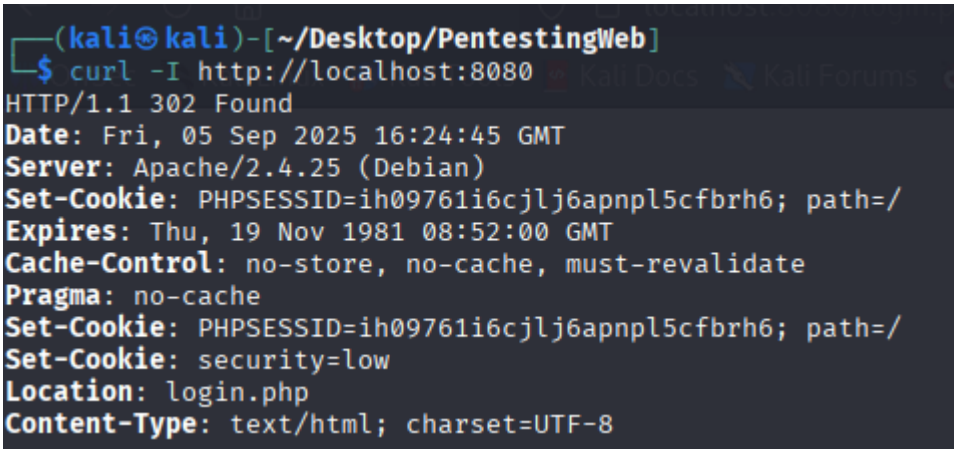


Laboratorios:

Fingerprinting y análisis con wget/curl:

Análisis con curl

Después de levantar el contenedor Docker de DVWA, se analiza la cabecera con el comando curl -I http://localhost:8080 y este devuelve la siguiente información:

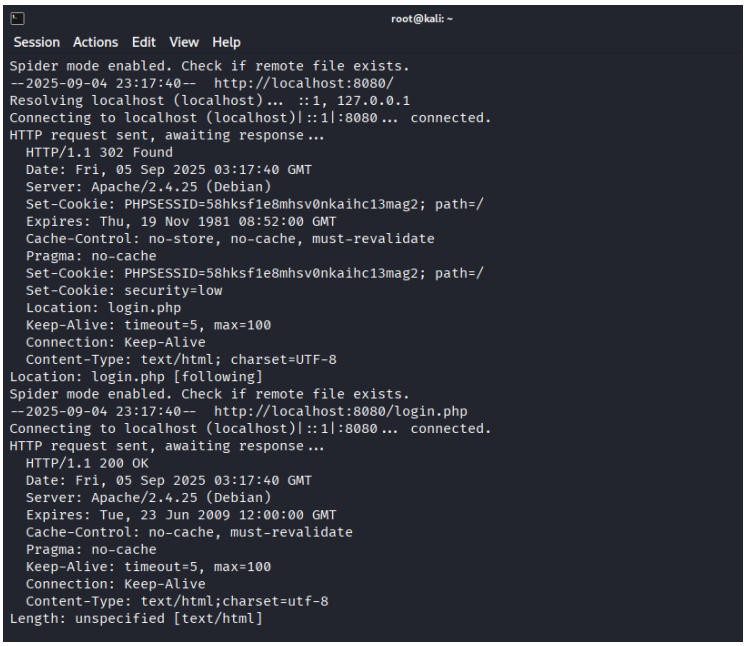


En esta parte hay información relevante del sistema, por ejemplo:

- **HTTP/1.1:** Protocolo usado para compatibilidad con dispositivos legacy.
- **Apache/2.4.25:** Es la versión del servidor lo cual es útil para buscar vulnerabilidades con ese nombre
    - Por ejemplo según la página oficial de Apache en esta versión se encontró una vulnerabilidad de DoS Attack (CVE-2016-216) que causa que el servidor se crashee incluso si los requests posteriores son válidos.
- **Set-cookie:** confirma que la aplicación hace uso de sesiones, para las cookies suelen existir flags de seguridad como HttpOnly o Secure, en este caso al no existir puede dar cabida a hijacking.
    - La cookie security=low igualmente carece de una flag de seguridad
- **security=low**, indica el nivel de seguridad 'bajo'
- **pragma: no cache,** esto sirve para que las paginas de login eviten que el navegador guarde copias
- **Login.php**: Indica donde está la página del login, lo cual podría ser útil para ataques de fuerza bruta.
- **Codigo de estado: 302 Found ->** Indica una redirección temporal.

Análisis con wget:

- Ahora el comando cambia pero el objetivo sigue siendo el mismo: wget --server-response --spider http://localhost:8080



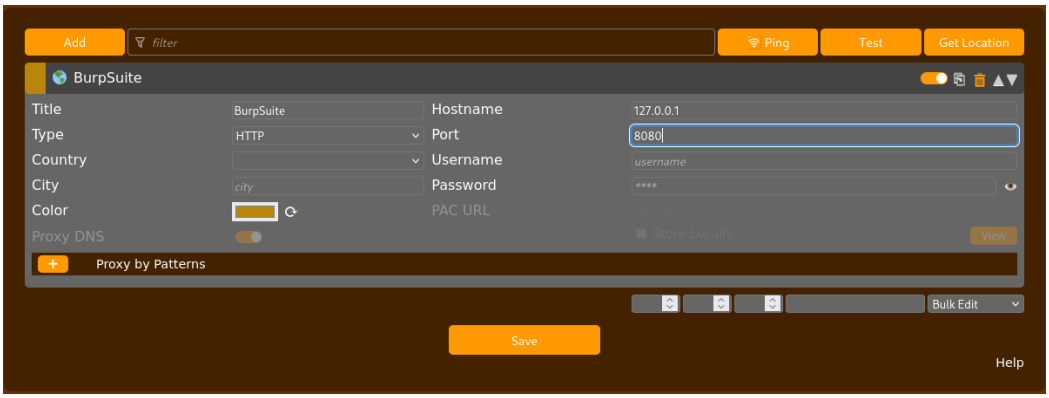
- Wget: Para hacer peticiones HTTP y realizar descargas
- --spider es para decirle a wget que no descargue nada
- --server-response para mostrar el header que envía el servidor

- Principal diferencia entre Curl y Wget

- Usa curl para inspección, pruebas rápidas, APIs y manipulación de requests.
- Usa wget para descargar archivos/sitios completos y almacenar evidencias.

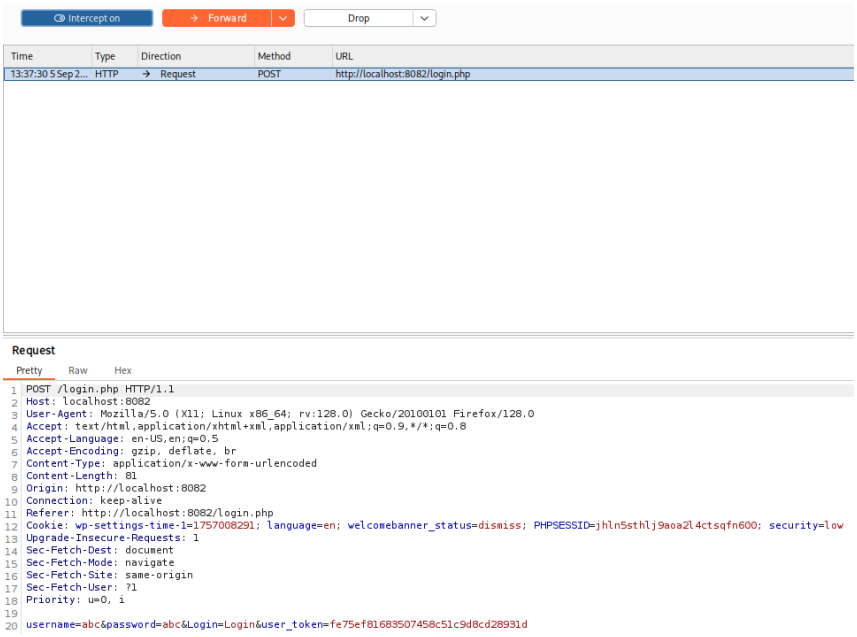
Interceptación y modificación de tráfico con BurpSuite:

Este laboratorio ayudó a comprender el concepto de servidor proxy como intermediario entre el endpoint y el servidor. Para su realización, primero se hizo uso de la extensión FoxyProxy, esta parte ayudó a comprender el uso de una extensión de Firefox: “FoxyProxy” la cual ayuda como intermediario (servidor proxy) haciendo que Burp Suite logre escuchar en una IP específica (127.0.0.1) con un puerto específico (8080).



Luego se activó el proxy en Burp Suite para interceptar tráfico. Una vez preparado Burp para interceptar tráfico se ingresa las credenciales de login, en este caso van a ser unas credenciales incorrectas: abc para usuario y abc para contraseña.

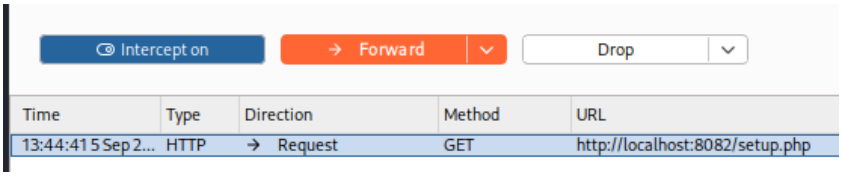
Burp Suite las intercepta y al ser HTTP, es decir, no tiene una capa de seguridad, entonces permite ver los datos ingresados a pesar de que es POST. La información que se observa es el tipo de petición (POST), la URL, la longitud, el motor de búsqueda, etc. Y en la parte inferior están las credenciales: user, password y token



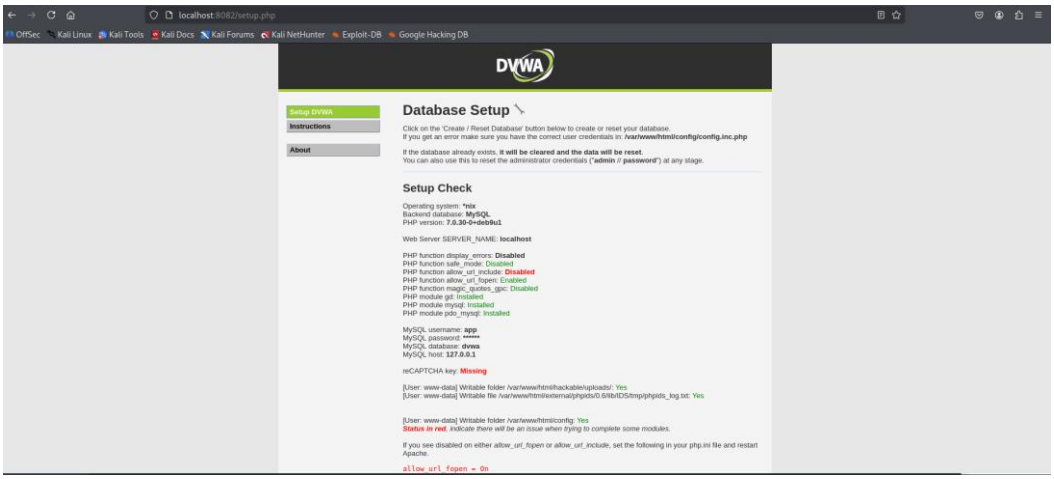
Estos datos pueden ser modificados fácilmente, por ejemplo, en este caso se va a cambiar las credenciales user y password para que sean las correctas (admin y password)

```
POST /login.php HTTP/1.1
Host: localhost:8082
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 81
Origin: http://localhost:8082
Connection: keep-alive
Referer: http://localhost:8082/login.php
Cookie: wp-settings-time-1=1757008291; language=en; welcomebanner_status=dismiss; PHPSESS
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i

username=admin&password=password&Login=Login&user_token=fe75ef81683507458c51c9d8cd28931d
```

De esta manera si hace clic en Forward el mensaje modificado se envía al servidor, luego se hace otra petición que es que una vez que se ha conectado, hace la petición de la página de inicio que en este caso se llama setup.php, se continua con Forward.



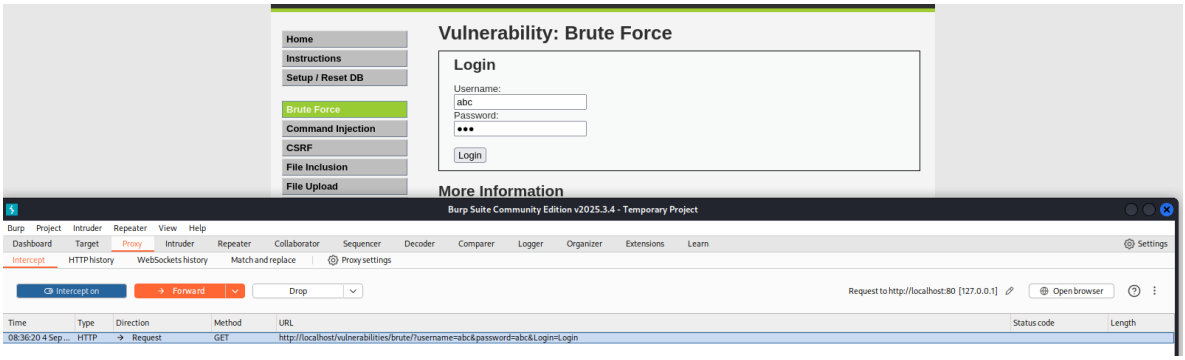
Y entonces, se tiene acceso a la página de inicio como si se hubiese ingresado las credenciales correctas.



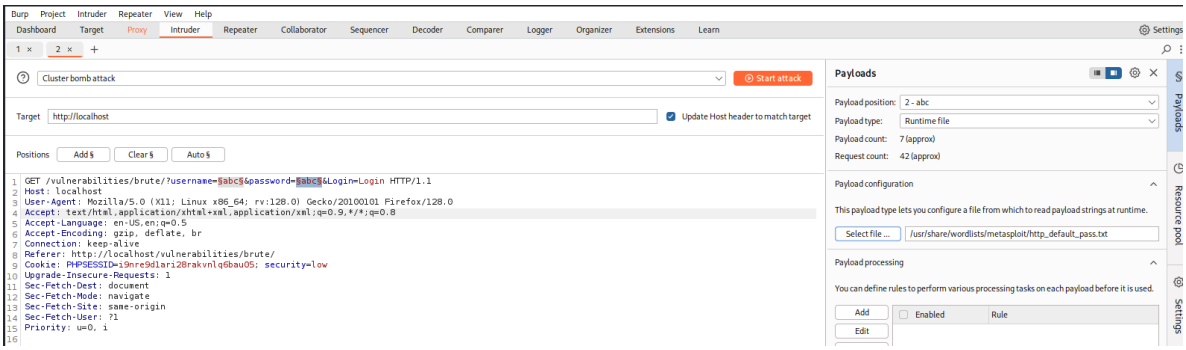
Fuerza Bruta con BurpSuite:

Para este bloque se aprendió el uso de BurpSuite para ataques de Fuerza Bruta mediante la máquina de DVWA en la sección de “Brute Force”.

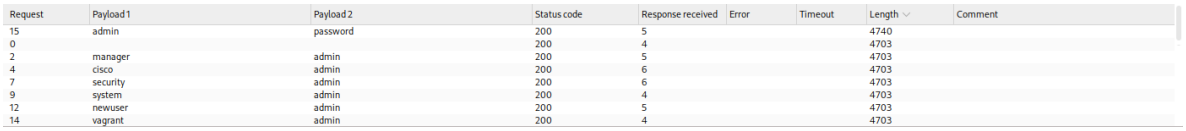
Como este es el inicio, se escogió el nivel más sencillo (Low), y se procedió con Burp Suite, dentro de esta herramienta se inició el proxy, y se procedió a capturar los paquetes enviados, en este caso al enviar el usuario y contraseña.



Una vez capturado el paquete se procede a usar la opción de Intruder, que es para analizar los paquetes y donde se va a aplicar el payload requerido que en este caso es Fuerza Bruta. Para ello se les coloca “Markers” a las áreas requeridas (usuario y contraseña) que es donde se va a aplicar el payload, y se escoge “Clear Bomb Attack” que es para que cada campo tenga su propio payload y no todos el mismo, posteriormente en la sección de Payloads se les agrega un diccionario a cada uno de estos campos (user y pass).



Una vez el payload se procede con el ataque, lo que hace este ataque de fuerza bruta es realizar combinaciones entre las palabras del diccionario de usuarios con el de contraseñas para intentar dar con la correcta. Lo importante aquí es la longitud que indica la respuesta del servidor, muchas son similares (4702 y 4703) pero cuando una varía más de lo normal quiere decir que es un posible candidato a intentar, en este caso quienes más varían son “admin” y “password” que son las credenciales buscadas.



Request	Payload 1	Payload 2	Status code	Response received	Error	Timeout	Length	Comment
15	admin	password	200	5			4740	
0			200	4			4703	
2	manager	admin	200	5			4703	
4	cisco	admin	200	6			4703	
7	security	admin	200	6			4703	
9	system	admin	200	4			4703	
12	newuser	admin	200	5			4703	
14	vagrant	admin	200	4			4703	

Lo interesante es que en nivel “Medium” puede servir la misma técnica, pues la única diferencia es que en “Low” cada intento no tiempo intermedio, en cambio en “Medium” es de 2 segundos, por lo cual solo aumenta la demora en obtener las credenciales.

Fuzzing de directorios y parámetros con Wfuzz:

Para este laboratorio primero se uso igualmente DVWA, y una vez activo se usó el comando: wfuzz -c -z file,/usr/share/wordlists/dirb/common.txt --hc 404 http://localhost:8080/FUZZ. Dentro de este comando hay varios parámetros útiles a destacar:

- -c: Coloca un color a cada resultado, por ejemplo a los 400’s les coloca un rojo, a los 300’s un azul y a los 200’s un verde, para identificar mejor.
- -z file,/usr/share/wordlists/dirb/common.txt: define la ruta del payload a usar que en este caso es un archivo (file) y se indica la ruta del archivo, que en este contexto es un diccionario (common.txt)
- --hc 404: No enlista las respuestas que den esta respuesta debido a que la mayoría de requests lo van a devolver, entonces solo se filtra lo útil.
- FUZZ: es el marcador de posición donde se colocan cada una de las palabras del diccionario

El resultado fue el siguiente:

Aquí se destacan varias Responses por parte del servidor, cada una con un significado propio:

- 403: los archivos existen, pero no son accesibles, por ejemplo .htpasswd puede contener hashes de las contraseñas
- 301: Indica que el recurso solicitado se ha movido a otra URL permanentemente, es decir, cuando se coloque este enlace, el motor redirige al nuevo enlace
- 302: Recurso movido temporalmente a otra URL
- 200: El enlace existe y se puede ingresar

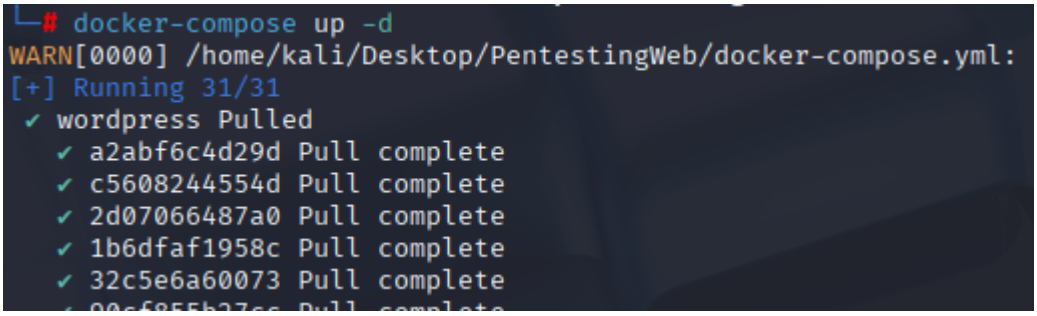
Escaneo de WordPress con WPScan:

- Dependencias:
  - Docker
  - Docker-compose
- Docker-compose.yml

```yaml
version: '3.3'
services:
  db:
    image: mariadb:10.5
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wp
      MYSQL_PASSWORD: wp
    ports:
      - "3307:3306"

  wordpress:
    image: wordpress:5.8-apache   #versión antigua
    restart: always
    ports:
      - "8081:80"
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wp
      WORDPRESS_DB_PASSWORD: wp
      WORDPRESS_DB_NAME: wordpress
    depends_on:
      - db
```

Aquí se descarga WordPress mediante Docker con una versión antigua (5.8 en este caso), se especifica el puerto en 8081

- Descarga de WordPress:



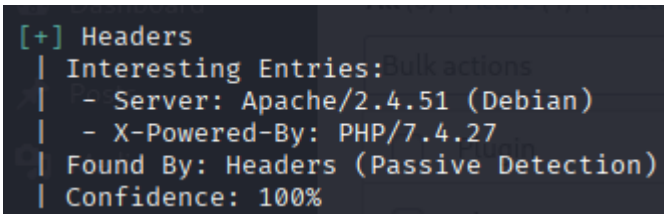
- Plugins instalados:
    - Contact Form 7 4.0.1
    - Duplicator 1.1.0
    - Gutenberg 8.4.0
    - Performance Lab 1.2.0

Una vez instalado WordPress y sus plugins desactualizados se procede a usar WPScan. El primer comando es para identificar los usuarios en la plataforma, para ello se usa: wpscan --url http://localhost:8081/ -e u

- --url -> especifico la URL (en este caso localhost)
- -e u -> se escoge el objetivo, en este caso es buscar usuarios (u)

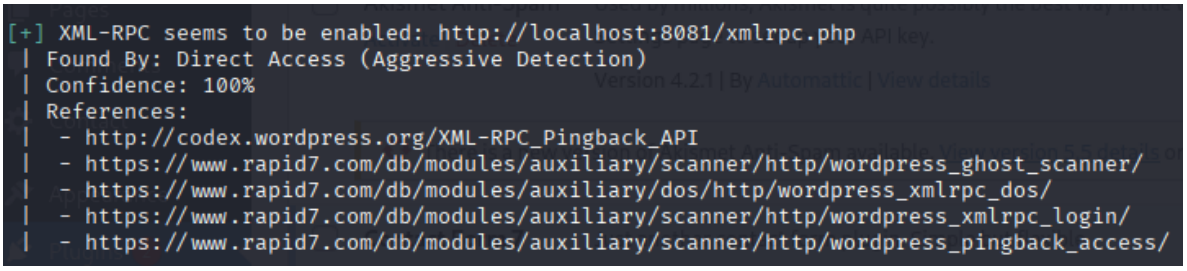
Lo que devuelve WPScan es lo siguiente:

1. Headers



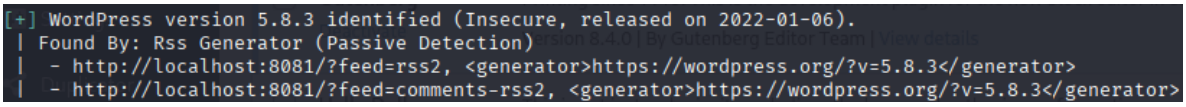
Se identifica la versión del servidor y la versión PHP que usa

2. XML-RPC



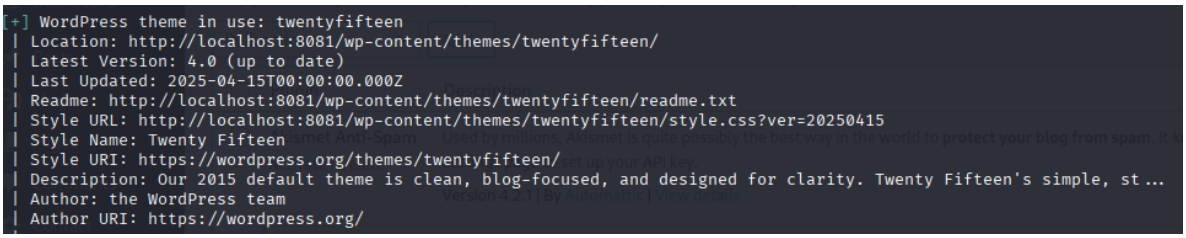
Esta parte es muy importante, porque el archivo XML-RPC es un archivo que permite realizar ataques de fuerza bruta, por lo cual se puede el conocimiento previo de Burp Suite para intentar obtener las credenciales si estas son débiles.

3. Versión de WordPress



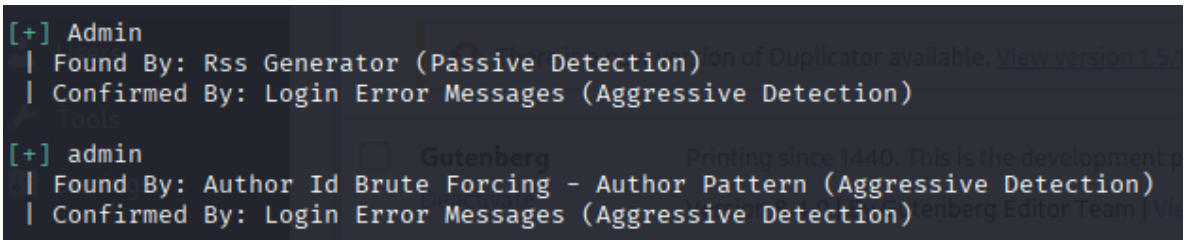
Se indica la versión de WordPress además de indicar que esta versión es insegura, para estos casos se puede buscar en GitHub algún exploit para estas versiones antiguas

4. Template usado



Brinda información del template que usa el usuario al momento, la URL donde se encuentra alojado, su versión, descripción, autor, entre otros.

5. Usuarios



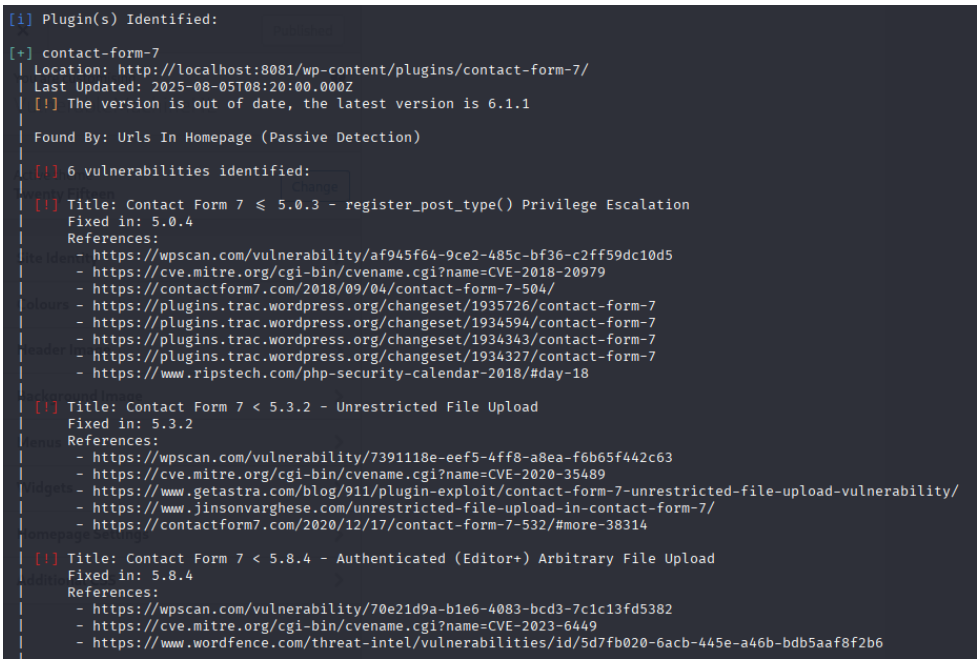
Y por último, los usuarios, esto es de ayuda porque si se combina con XML-RPC mencionado previamente facilita el ataque de fuerza bruta al ya conocer el 50% de las credenciales.

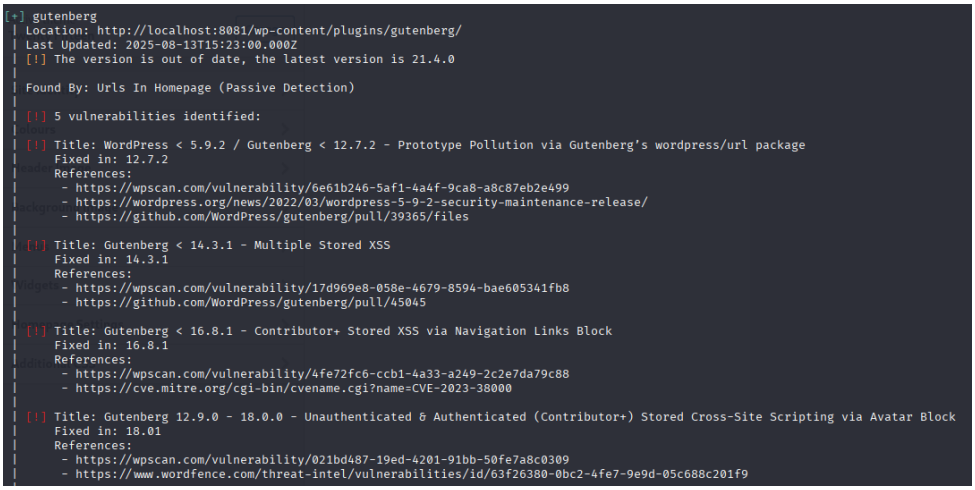
Además WPScan tiene una API que permite ver posibles vulnerabilidades, es lo que se va a probar ahora para los plugins, este tema se hubiese podido probar igualmente similar al comando anterior pero cambiando “u” por “p”, pero el escaneo devolvería solo los plugins y si están actualizados, no sus posibles vulnerabilidades.

```
[i] Plugin(s) Identified:

[+] contact-form-7
 | Location: http://localhost:8081/wp-content/plugins/contact-form-7/
 | Last Updated: 2025-08-05T08:20:00.000Z
 | [!] The version is out of date, the latest version is 6.1.1
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | Version: 4.0.1 (100% confidence)
 | Found By: Query Parameter (Passive Detection)
 |  - http://localhost:8081/wp-content/plugins/contact-form-7/includes/css/styles.css?ver=4.0.1
 |  - http://localhost:8081/wp-content/plugins/contact-form-7/includes/js/scripts.js?ver=4.0.1
 | Confirmed By:
 |  Readme - Stable Tag (Aggressive Detection)
 |   - http://localhost:8081/wp-content/plugins/contact-form-7/readme.txt
 |  Readme - ChangeLog Section (Aggressive Detection)
 |   - http://localhost:8081/wp-content/plugins/contact-form-7/readme.txt

[+] gutenberg
 | Location: http://localhost:8081/wp-content/plugins/gutenberg/
 | Last Updated: 2025-08-13T15:23:00.000Z
 | [!] The version is out of date, the latest version is 21.4.0
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | Version: 8.4.0 (100% confidence)
 | Found By: Readme - Stable Tag (Aggressive Detection)
 |  - http://localhost:8081/wp-content/plugins/gutenberg/readme.txt
 | Confirmed By: Change Log (Aggressive Detection)
 |  - http://localhost:8081/wp-content/plugins/gutenberg/changelog.txt, Match: '= 8.4.0'
```

En cambio, con la API, esto permite una consulta a la base de datos de vulnerabilidades de WPScan lo cual daría un informe más detallado, indicando vulnerabilidades desde la propia versión de WordPress hasta sus plugins instalados. El comando es: wpscan --url http://localhost:8081/ -e p --api-token='KURgKWWQ7MsfhkM4sEZnaWrPKylmluUhs0SfRLf0aS4'

```
[+] WordPress version 5.8.3 identified (Insecure, released on 2022-01-06).
 | Found By: Rss Generator (Passive Detection)
 |  - http://localhost:8081/?feed=rss2, <generator>https://wordpress.org/?v=5.8.3</generator>
 |  - http://localhost:8081/?feed=comments-rss2, <generator>https://wordpress.org/?v=5.8.3</generator>
 |
 | [!] 33 vulnerabilities identified:
 |
 | [!] Title: WordPress < 5.9.2 - Prototype Pollution in jQuery
 |     Fixed in: 5.8.4
 |     References:
 |      - https://wpscan.com/vulnerability/1ac912c1-5e29-41ac-8f76-a062de254c09
 |      - https://wordpress.org/news/2022/03/wordpress-5-9-2-security-maintenance-release/
 |
 | [!] Title: WordPress < 5.9.2 / Gutenberg < 12.7.2 - Prototype Pollution via Gutenberg's wordpress/url package
 |     Fixed in: 5.8.4
 |     References:
 |      - https://wpscan.com/vulnerability/6e61b246-5af1-4a4f-9ca8-a8c87eb2e499
 |      - https://wordpress.org/news/2022/03/wordpress-5-9-2-security-maintenance-release/
 |      - https://github.com/WordPress/gutenberg/pull/39365/files
 |
 | [!] Title: WP < 6.0.2 - Reflected Cross-Site Scripting
 |     Fixed in: 5.8.5
 |     References:
 |      - https://wpscan.com/vulnerability/622893b0-c2c4-4ee7-9fa1-4cecef6e36be
 |      - https://wordpress.org/news/2022/08/wordpress-6-0-2-security-and-maintenance-release/
 |
 | [!] Title: WP < 6.0.2 - Authenticated Stored Cross-Site Scripting
 |     Fixed in: 5.8.5
 |     References:
 |      - https://wpscan.com/vulnerability/3b1573d4-06b4-442b-bad5-872753118ee0
 |      - https://wordpress.org/news/2022/08/wordpress-6-0-2-security-and-maintenance-release/
```





Algunas de las vulnerabilidades más notorias son:

1. Contact Form 7 < 5.9.2 - Reflected Cross-Site Scripting
    - El plugin Contact Form 7 no sanitiza los parámetros dados por el usuario, lo cual termina mostrando los resultados de los comandos dentro de la página
2. WordPress < 6.4.3 - Deserialization of Untrusted Data
    - WordPress no desinfecta las opciones al instalarse y actualizarse antes de serializarlas, lo que podría permitir que usuarios con altos privilegios, como el administrador, realicen un ataque de inyección de objetos PHP.
3. WP < 6.0.3 - SQLi in WP_Date_Query
    - WordPress no desinfecta adecuadamente la relación pasada a WP_Date_Query, lo que podría provocar una inyección de SQL
        - WP_Date_Query = Ayudante que permite filtrar resultados según la columna DATE.

Conclusión:

A lo largo de semana se logró poner en práctica diferentes técnicas de pentesting, desde escaneo hasta modificación de datos, pasando por el análisis del objetivo. Esto mediante el uso de herramientas de web hacking como Burp Suite o WPScan, además de librerías como cURL o wfuzz, de esta manera se puso en práctica los conceptos teóricos como métodos HTTP (POST, GET, …), encabezados HTTP, scripts web en PHP, proxy, estados HTTP, entre otros. A partir de estos conceptos igualmente se aprendieron técnicas como escaneo pasivo y agresivo, ofuscación, modificación de peticiones HTTP, ataques de fuerza bruta y enumeración.

Recursos:

- FoxyProxy: https://www.youtube.com/watch?v=wY0EBp2OeHo
- BruteForce: https://www.youtube.com/watch?v=pSBD9cgwgk0&t=84s
- Old Plugins WordPress: https://www.youtube.com/watch?v=hCI0Hgwhs8Q
- Apache Vulnerabilities: https://httpd.apache.org/security/vulnerabilities_24.html