Informe de la Explotación de la página de Fristileak desde Kali Linux y cambio de la Mac de Ubuntu

.

Asignatura

Seguridad y Auditoría en Redes

Estudiante

Andrés Felipe Asprilla Córdoba

Universidad Tecnológica del Chocó “Diego Luis Córdoba”

Ingeniería de Telecomunicaciones e Informática

Quibdó/Chocó

Informe de la Explotación de la página de Fristileak desde Kali Linux y cambio de la Mac de Ubuntu

Docente

ING Rafael Sandoval Morales

Estudiante

Andrés Felipe Asprilla Córdoba

Asignatura

Seguridad y Auditoría en Redes

Universidad Tecnológica del Chocó “Diego Luis Córdoba”

Ingeniería de Telecomunicaciones e Informática

Quibdó/Chocó

Universidad Tecnológica del Chocó
Diego Luis Córdoba

Tabla de contenido

Tabla de Ilustraciones.

Universidad Tecnológica del Chocó
Diego Luis Córdoba

Introducción

El presente informe recoge el proceso de prácticas realizadas en el marco de la asignatura de Seguridad y Auditoría en Redes, donde se trabajó en dos escenarios puntuales: la explotación del portal vulnerable Fristileaks y la modificación de la dirección MAC en un sistema Ubuntu. En el primer caso, se aplicaron técnicas de reconocimiento, enumeración y análisis sobre un sitio web que presentaba múltiples fallos de seguridad, entre ellos directorios expuestos, archivos sensibles accesibles y credenciales almacenadas en texto claro.

Esto permitió obtener acceso al sistema, manipular su base de datos y crear usuarios falsos para ingresar al portal como si fueran legítimos. En el segundo escenario, se trabajó con la configuración de red de Ubuntu para realizar cambios en la dirección MAC del adaptador, evidenciando la facilidad con la que puede modificarse la identidad de un dispositivo dentro de la red.

Ambas prácticas, aunque diferentes en su aplicación, tienen en común el objetivo de demostrar vulnerabilidades reales que pueden existir en sistemas informáticos y redes, resaltando la importancia de una correcta configuración y de la adopción de buenas prácticas de seguridad. A través de estos ejercicios se adquirió experiencia práctica en la detección y explotación de debilidades, así como en la comprensión de los riesgos que estas representan en escenarios de la vida real.

Objetivos

Objetivo general

Analizar y demostrar el proceso de explotación de vulnerabilidades en el portal Fristileaks desde Kali Linux y el cambio de dirección MAC en Ubuntu, con el fin de identificar fallos de seguridad y comprender los riesgos asociados a configuraciones débiles en aplicaciones y redes.

Objetivos específicos

- Realizar un reconocimiento inicial de los servicios activos en la máquina víctima mediante herramientas de escaneo.
- Identificar y analizar directorios ocultos y archivos sensibles dentro del servidor web para detectar posibles vulnerabilidades.
- Explorar la base de datos asociada al portal, obtener credenciales y validar el acceso mediante la creación de usuarios personalizados.
- Modificar la dirección MAC en Ubuntu para comprobar la asignación de nuevas direcciones IP y su impacto en la conectividad de la red.

Universidad Tecnológica del Chocó
Diego Luis Córdoba

Planteamiento del problema

Realizar un laboratorio práctico orientado a identificar y explotar vulnerabilidades en un entorno controlado. En la primera parte, el trabajo consiste en analizar el portal **Fristileaks**, detectando directorios y archivos expuestos, revisando si contienen credenciales en texto claro o información sensible, y utilizando esos datos para acceder al sistema. El ejercicio también consiste en comprobar el impacto de estas fallas manipulando la base de datos y creando usuarios propios que permitan ingresar de manera legítima al portal.

Luego modificar la dirección **MAC** de un sistema **Ubuntu**, aplicando comandos básicos para demostrar que la identidad de un dispositivo en la red puede alterarse con facilidad. De esta forma, se evidencia lo vulnerable que resulta la seguridad cuando depende únicamente de direcciones físicas como mecanismo de control.

Universidad Tecnológica del Chocó
Diego Luis Córdoba

Explotación de la pagina de Fristileak desde Kali Linux

Abrí la ventana de configuración de la máquina virtual y revisé las opciones de red. Verifiqué que el adaptador estuviera activado, que estuviera en modo NAT y que la casilla de “cable conectado” estuviera marcada; todo esto para asegurar que la máquina tendría salida a la red desde mi equipo y que no habría problemas de comunicación mientras trabajaba. También observé el tipo de tarjeta de red que estaba usando, para dejar constancia de la configuración antes de iniciar las pruebas.

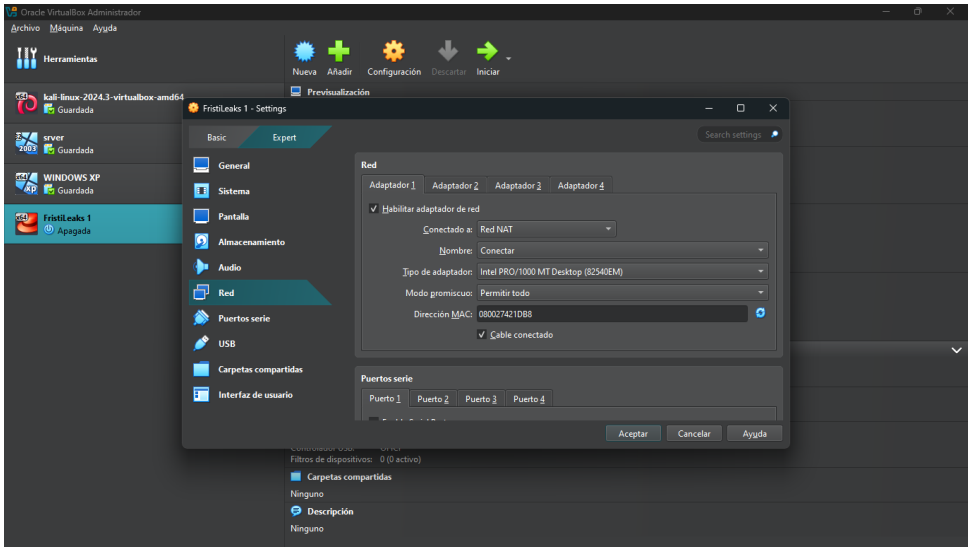

Ilustración 1Modificando la mac de la máquina de fristileaks

Volví a confirmar los mismos ajustes de red en una segunda revisión, por si acaso algún cambio no se había aplicado. Me aseguré de que la máquina virtual estuviera lista para arrancar con la configuración de red prevista y comprobé visualmente los valores (modo NAT y adaptador habilitado). Este doble chequeo lo hice para evitar perder tiempo más adelante por un fallo de conexión que no estuviera relacionado con la aplicación que iba a analizar.

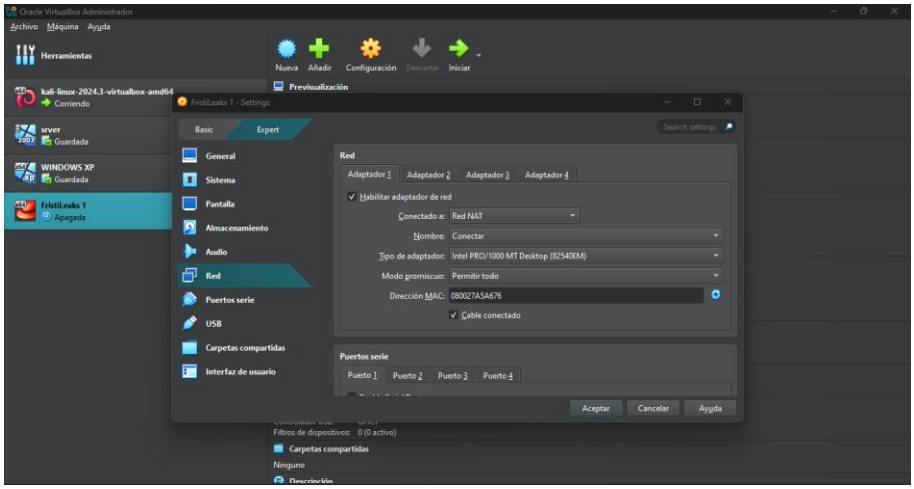

Ilustración 2Modificacion completada

Despues de modificar la mac, lo primero que realicé fue un escaneo con Nmap sobre la dirección 10.10.10.2 para identificar los puertos abiertos. El análisis mostró que la máquina tenía habilitado el puerto 135, que corresponde a un servicio de Windows. Además, se pudo ver información como la dirección MAC y el tipo de tarjeta de red virtual, lo que me permitió confirmar que se trataba de una máquina con sistema operativo Windows configurada en un entorno virtual.

Después continué con la dirección 10.10.10.5, donde también ejecuté el escaneo. En este caso, los resultados revelaron que el equipo tenía activo un servidor Apache en la versión 2.2.15, acompañado de PHP en la versión 5.3.3, lo que indicaba que era un sistema basado en CentOS. Este hallazgo fue importante porque significa que la máquina tenía un servicio web al cual podía intentar acceder más adelante.

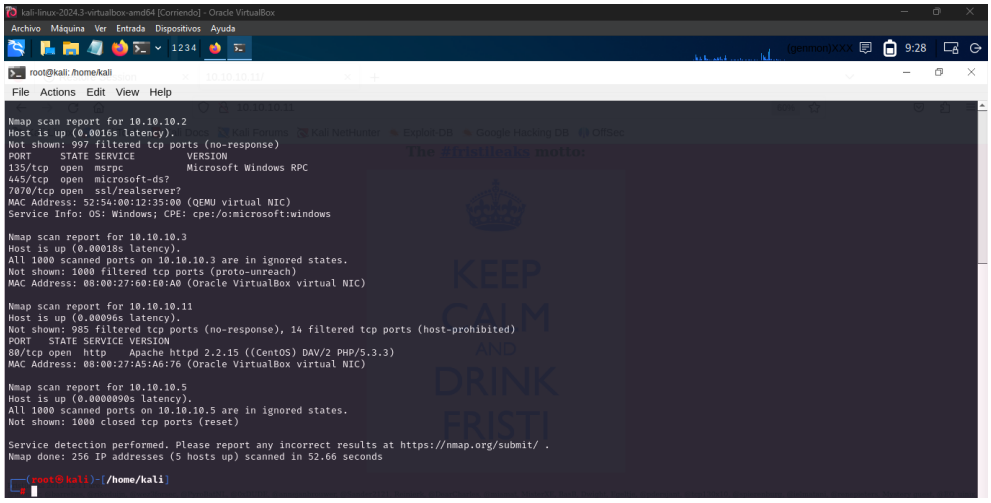

Ilustración 3Escaneo con nmap

Después de haber identificado que en la dirección **10.10.10.11** había un servidor web en funcionamiento, decidí abrir el navegador e ingresar directamente a esa IP para ver qué estaba disponible. Al hacerlo, la página cargó un sitio con fondo rosado que mostraba un mensaje con la frase *"KEEP CALM AND DRINK FRISTI"*, junto al título **#fristileaks motto**.

Este resultado me confirmó que el servidor estaba activo y mostrando contenido web. Aunque a simple vista parecía una página muy básica, sabía que detrás de este tipo de mensajes podían ocultarse pistas o vulnerabilidades que podían ser aprovechadas para seguir con la prueba.

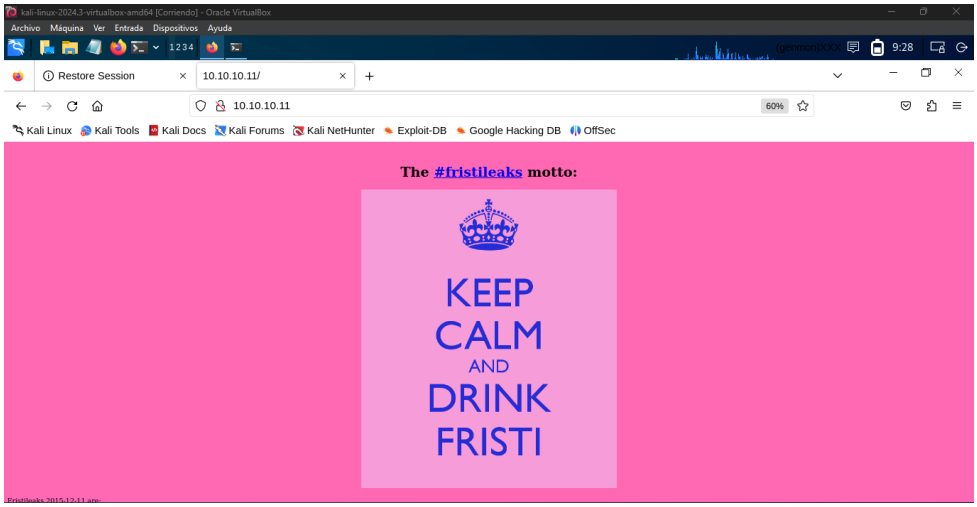

Ilustración 4KEEP CALM AND DRINK FRISTI

Después de ver que la página web en **10.10.10.11** estaba activa, decidí usar la herramienta **dirb** para buscar directorios ocultos dentro del servidor que no fueran visibles directamente desde el navegador. Para eso ejecuté el comando dirb http://10.10.10.11/ con una lista de palabras que me permitió hacer un escaneo más completo.

El resultado me mostró varias rutas interesantes: encontré que existía un directorio **/cgi-bin/**, aunque no tenía permisos de acceso, también apareció un archivo **index.html**, un **robots.txt** y un directorio llamado **/images/** que resultó ser accesible. El propio escaneo indicó que este directorio era “listable”, es decir, que se podían ver los archivos que contenía.

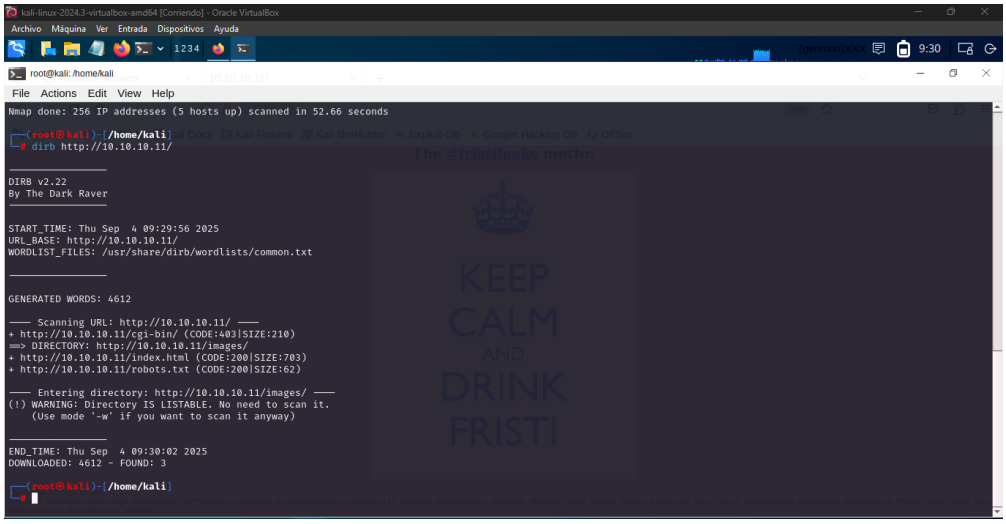

Ilustración 5herramienta dirb para buscar directorios ocultos

Después de haber utilizado **dirb** para encontrar directorios en el servidor, decidí revisar las listas de palabras que trae la herramienta en la ruta **/usr/share/dirb/wordlists**. Para eso usé los comandos cd y ls, lo que me permitió ver varios archivos de diccionarios como *big.txt*, *common.txt*, *small.txt*, entre otros.

Luego abrí el archivo **common.txt** con el comando cat common.txt para revisar su contenido. Allí encontré una lista de nombres comunes de directorios y archivos ocultos, como **.bash_history**, **.passwd**, **.ssh**, **.config**, entre otros. Estos nombres son muy útiles porque permiten a la herramienta probar automáticamente si esos directorios o archivos existen en el servidor que estoy auditando.

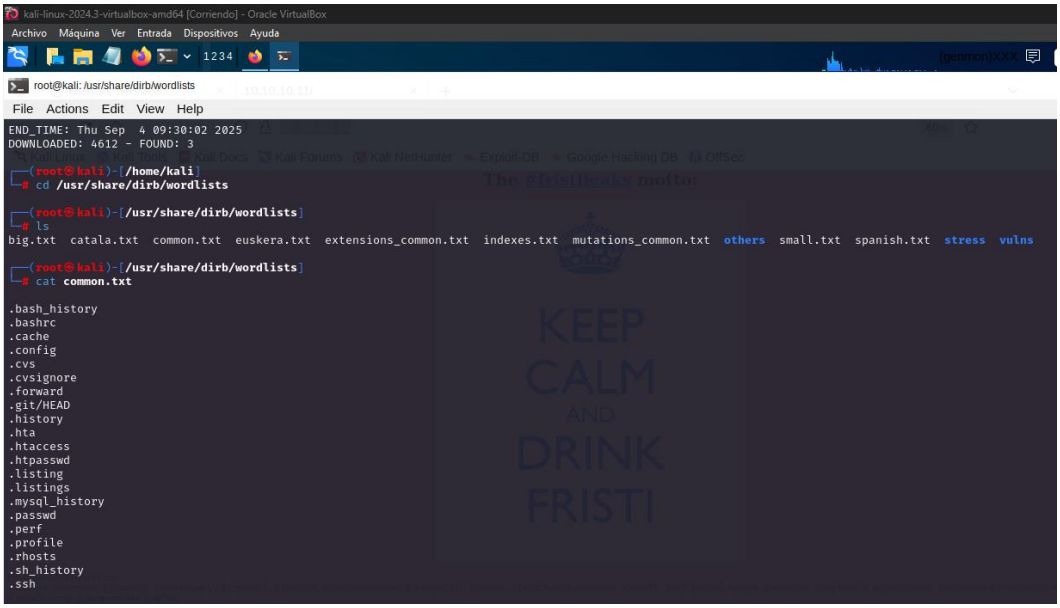

Ilustración 6common.txt

Después de identificar que existía un directorio **/cgi-bin/** en el servidor, decidí enfocarme directamente en él para ver si contenía archivos o scripts ejecutables que pudieran ser vulnerables. Para eso lancé el comando dirb http://10.10.10.11/cgi-bin/ utilizando nuevamente el diccionario **common.txt**.

El escaneo recorrió más de 4600 palabras del diccionario, pero en este caso no encontró ningún archivo ni directorio adicional dentro de **/cgi-bin/**. El resultado fue vacío, lo que me indicó que, aunque el directorio existe, aparentemente no contiene nada interesante o al menos nada accesible con el diccionario que utilicé.

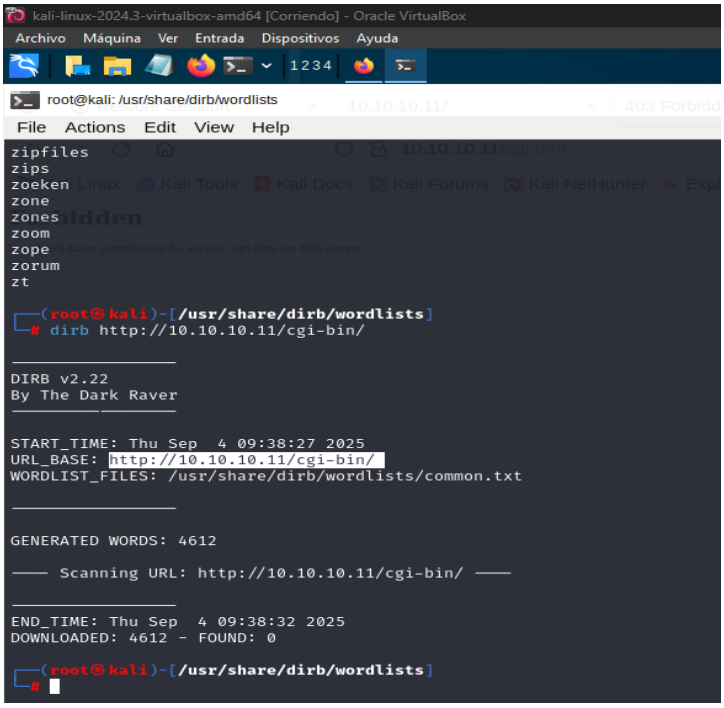

Ilustración 7directorio /cgi-bin/

Después de que el escaneo con **dirb** me indicara la existencia del directorio **/cgi-bin/**, intenté acceder directamente a él desde el navegador para confirmar si era posible visualizar algo de manera manual. Ingresé la dirección **http://10.10.10.11/cgi-bin/** y la respuesta del servidor fue un error **403 Forbidden**.

Este mensaje me indicó que, aunque el directorio existe, no tengo permisos para ver su contenido desde el navegador. Es decir, el acceso está restringido y el servidor bloquea la visualización directa de los archivos que pueda contener.

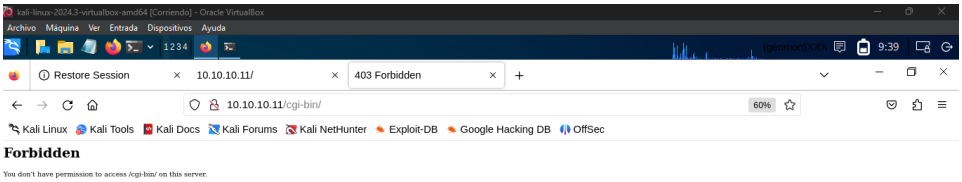

Ilustración 8dirección http://10.10.10.11/cgi-bin/

Después de haber probado con el directorio **/cgi-bin/**, decidí hacer lo mismo con la ruta **/images/** que el escaneo anterior de dirb me había mostrado como accesible. Para eso ejecuté el comando dirb http://10.10.10.11/images/ utilizando nuevamente el diccionario **common.txt**.

El escaneo analizó más de 4600 posibles nombres de archivos, pero en este caso no encontró nada adicional dentro de la carpeta de imágenes. El resultado fue vacío, lo que significa que no había ficheros detectables mediante el diccionario que estaba utilizando.

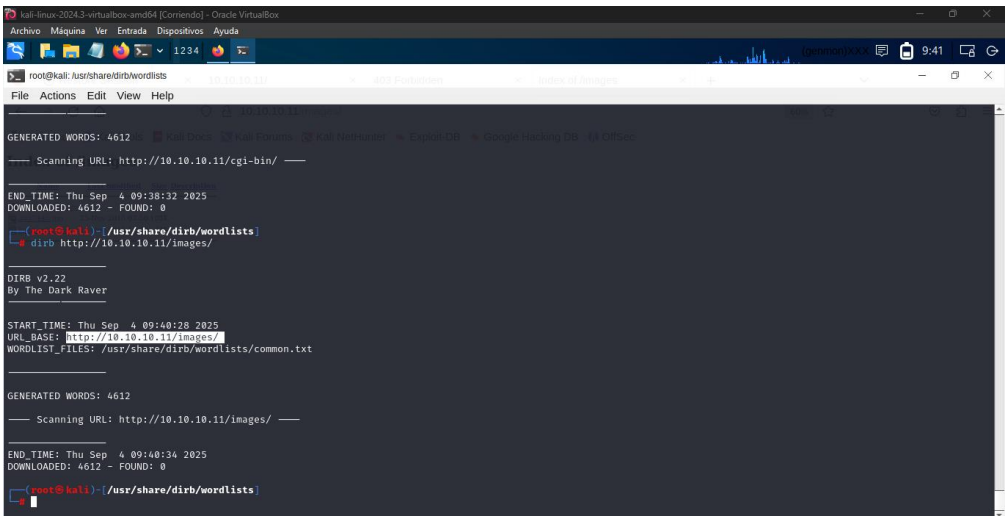

Ilustración 9ruta /images/

Luego de confirmar que el directorio **/images/** existía, decidí abrirlo directamente desde el navegador ingresando a la dirección **http://10.10.10.11/images/**. En esta ocasión sí logré visualizar el contenido de la carpeta, ya que el servidor tenía habilitado el listado de archivos.

Dentro del directorio encontré dos ficheros: una imagen llamada **3037dd40.jpg** y otra llamada **keecalms.png**. El hecho de que los archivos fueran visibles y descargables confirmaba que la carpeta no estaba protegida correctamente, lo cual representaba una posible fuga de información.

Este hallazgo fue importante porque el acceso directo a directorios como este puede revelar pistas escondidas en las imágenes o archivos que los administradores del servidor dejaron expuestos sin restricciones.

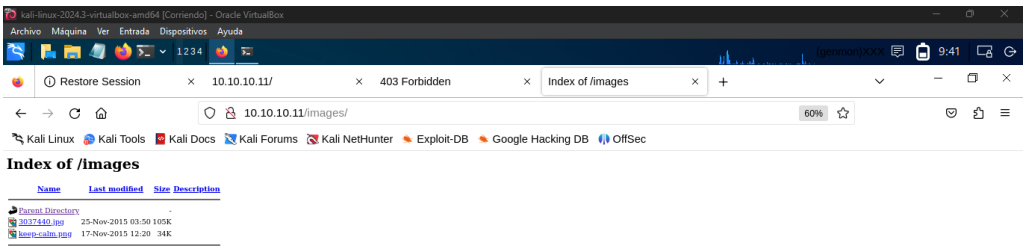

Ilustración 10dirección http://10.10.10.11/images/

Después de revisar el directorio **/images/**, decidí continuar explorando otros archivos que el escaneo inicial de **dirb** había reportado. Uno de ellos fue el archivo **index.html**, que correspondía a la página principal del servidor.

Para verificar si este archivo contenía algo más oculto en su interior, ejecuté el comando dirb http://10.10.10.11/index.html con el mismo diccionario **common.txt**. El análisis probó más de 4600 palabras, pero en este caso no encontró ningún directorio ni archivo adicional relacionado con esa ruta.

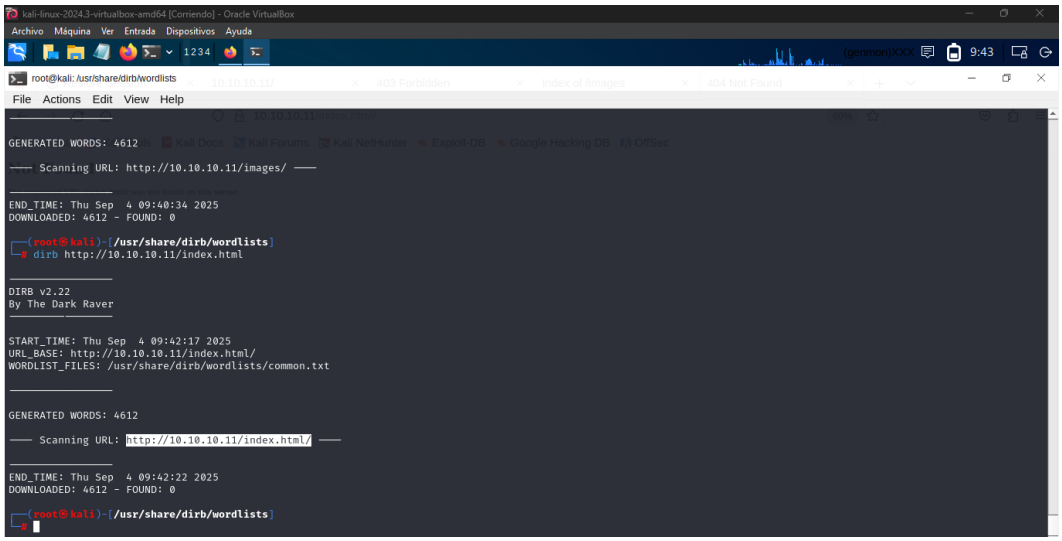

Ilustración 11archivo index.html

Luego de que el escaneo de **dirb** me mostrara la existencia de un archivo **index.html**, intenté abrirlo directamente desde el navegador escribiendo la dirección **http://10.10.10.11/index.html/**. Sin embargo, al cargar la página recibí el mensaje **Not Found**, lo que significaba que el recurso no estaba disponible o que en realidad no existía en la ruta que intenté acceder.

Este resultado me indicó que, aunque el escaneo mencionó el archivo, en la práctica no había nada allí o el servidor no lo tenía configurado para mostrarse de manera directa. De esta forma descarté esta ruta como una opción útil para seguir avanzando.

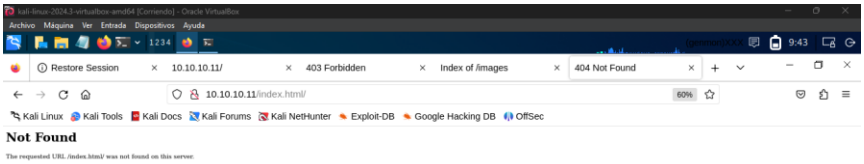

Ilustración 12dirección http://10.10.10.11/index.html/

En esta parte del proceso lo que hice fue acceder desde mi máquina Kali Linux a la dirección **10.10.10.11/fristi/**. Al cargar la página me encontré con lo que parecía ser un portal de administración llamado **"fristileaks admin portal"**.

La página mostraba una imagen en tono de burla y en la parte inferior se encontraba un formulario de inicio de sesión con campos de **usuario** y **contraseña**. En este punto identifiqué que probablemente se trataba de un sitio vulnerable o mal configurado, ya que suele ser común que en este tipo de prácticas los portales de login escondan alguna debilidad que se pueda explotar para acceder a la plataforma sin credenciales legítimas.

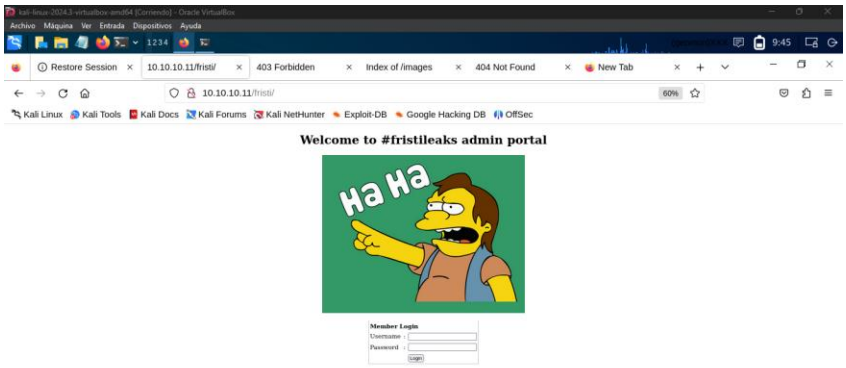

Ilustración 13dirección 10.10.10.11/fristi/.

En este paso, lo que hice fue tomar una cadena de texto en **Base64** que encontré durante el proceso de exploración de la página y la llevé a una herramienta en línea para decodificarla. Usé la plataforma *onlinepngtools.com* para convertir esa cadena y ver qué información contenía.

Al hacer la conversión, el resultado fue una imagen en formato **PNG**, y dentro de ella aparecía un texto que decía **"keKkeKKeKkEkkEk"**. Aunque en apariencia parece un mensaje sin mucho sentido, en el contexto de este reto lo interpreté como una **posible pista o contraseña** que probablemente me serviría más adelante para intentar acceder al portal de administración que había encontrado previamente.

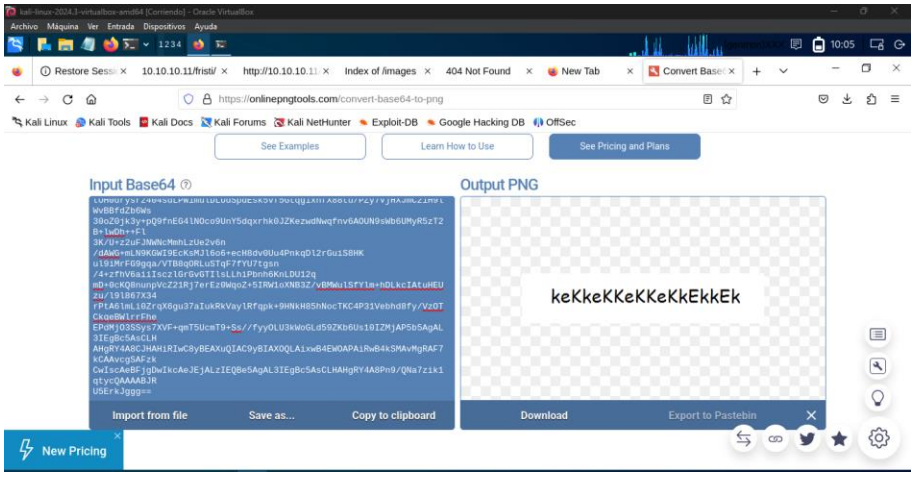

Ilustración 14Base64

Aquí lo que hice fue dar un paso más con la información que había encontrado en la imagen anterior. Después de decodificar la cadena en **Base64** y obtener la imagen con el texto extraño, utilicé una herramienta en línea llamada **Image to Text** para aplicar un proceso de **OCR (reconocimiento de caracteres)**.

De esta forma pude extraer directamente el texto que aparecía en la imagen y confirmarlo de manera más clara. El resultado fue el mismo mensaje que ya había identificado antes:

“keKkeKKeKkEkkEk”. Al pasarlo a texto plano me resultaba mucho más fácil copiarlo y utilizarlo posteriormente, sin necesidad de volver a revisar la imagen.

En este punto, lo que conseguí fue transformar una pista oculta dentro de un archivo codificado en una cadena de texto lista para usarse, lo cual reforzaba la idea de que podía tratarse de una **contraseña o clave de acceso** para el portal de administración.

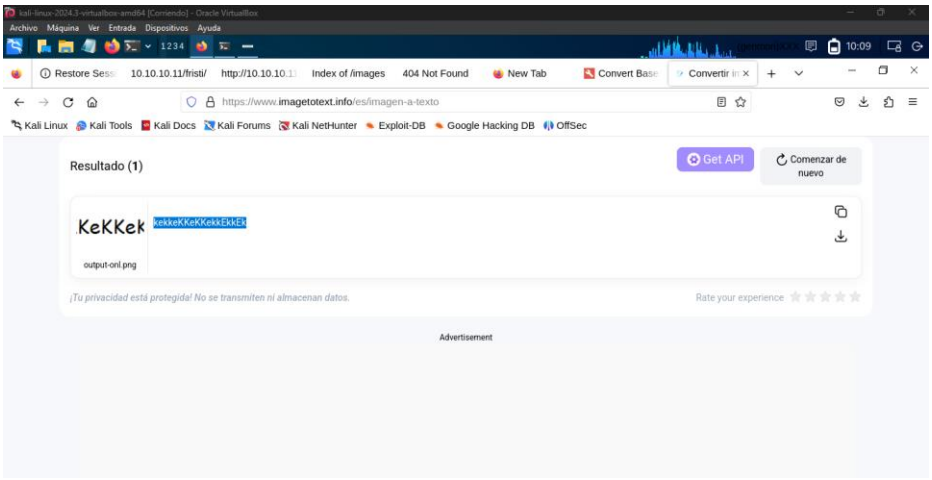

Ilustración 15Image to Text

En esta parte ya puse en práctica la pista que había encontrado anteriormente. Tomé el texto **“keKkeKKeKkEkkEk”**, que obtuve al decodificar la información en Base64 y luego procesar la imagen, y lo utilicé como **contraseña** dentro del portal de administración de la página **fristileaks**.

En el campo de **usuario** ingresé un nombre de prueba (en este caso “zeezee”) y en el campo de **contraseña** escribí la cadena que había descubierto. Con esto intenté iniciar sesión en el formulario.

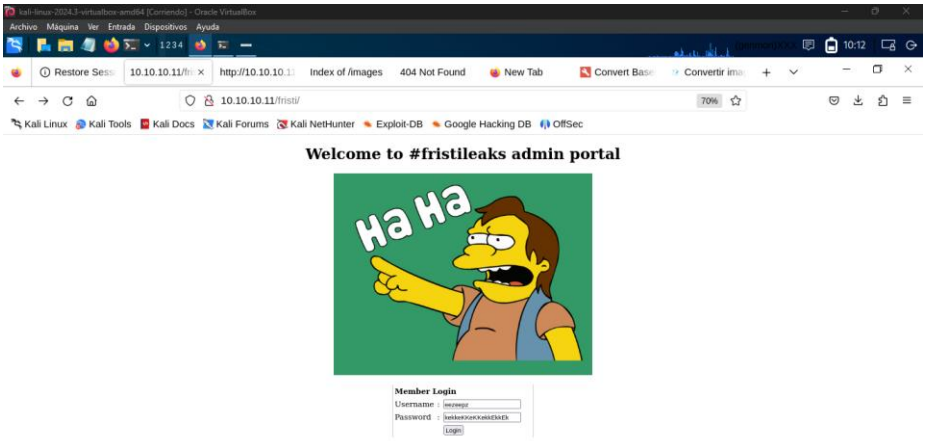

Ilustración 16Ingreso al portal

Aquí validé que la contraseña que había descubierto realmente funcionaba. Al intentar iniciar sesión en el portal de administración con el usuario de prueba y la clave **“keKkeKKeKkEkkEk”**, el sistema me redirigió a una nueva página con el mensaje **“Login successful”**.

Esto confirmó que logré acceder correctamente al portal y que la pista encontrada en la imagen codificada era la clave para vulnerar la autenticación. Además, en esta nueva página aparecía una opción llamada **“upload file”**, lo que me indicó que ahora tenía acceso a una funcionalidad que posiblemente me permitiría subir archivos al servidor.

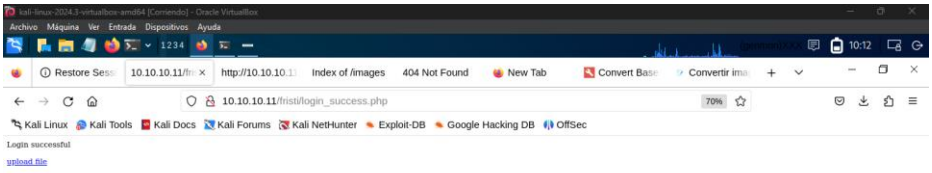

Ilustración 17Ingreso exitoso

En este punto, después de haber ingresado al portal y encontrar la opción para **subir archivos**, intenté acceder directamente al directorio **/uploads/** para verificar si lo que había subido se encontraba disponible o si podía listar los contenidos.

Sin embargo, al hacerlo, ¡el sistema me devolvió un mensaje simple con la palabra “no!”, lo que significa que este directorio estaba protegido contra la visualización directa o el listado de archivos.

Este resultado me mostró que, aunque ya tenía la posibilidad de interactuar con el portal y subir ficheros, el servidor contaba con ciertas restricciones para evitar que cualquiera pudiera ver el contenido cargado de manera tan directa. Aun así, esto representaba una señal de que debía buscar otra forma de validar si mis archivos realmente estaban siendo almacenados y cómo podía ejecutarlos para seguir avanzando en la explotación.

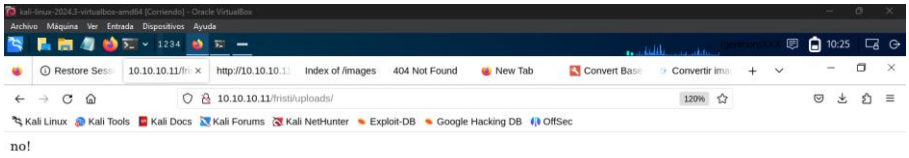

Ilustración 18directorio /uploads/

En este paso avancé preparando un archivo malicioso para poder aprovechar la opción de subida de ficheros que encontré en el portal. Para ello, desde Kali Linux accedí a la ruta **/usr/share/webshells/php**, donde se encuentran distintos ejemplos de *webshells* ya listos para usarse.

Dentro de esa carpeta localicé el archivo **php-reverse-shell.php**, que es un script diseñado para generar una **conexión inversa** entre la máquina víctima y mi máquina atacante. Lo que hice después fue copiar este archivo al escritorio y renombrarlo como **felipe.php**, con el fin de personalizarlo y evitar levantar sospechas al momento de subirlo al servidor.

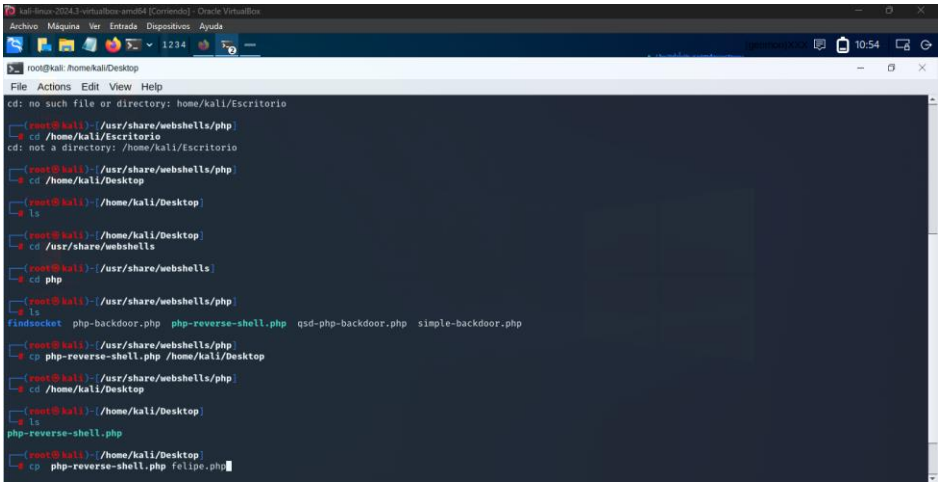

Ilustración 19la ruta /usr/share/webshells/php

Después de haber copiado y renombrado la *reverse shell* como **felipe.php**, lo que hice fue abrir el archivo con el editor de texto **nano** para comenzar a modificarlo.

La idea de este paso era ajustar la configuración interna del script, principalmente la **dirección IP** y el **puerto** que utilizaría para establecer la conexión inversa. Esto es necesario porque el archivo, tal como viene por defecto, apunta a valores genéricos, y para que funcione correctamente debía indicar la IP de mi máquina atacante y un puerto específico en el que tendría abierta la escucha.

Con esta modificación me aseguraba de que, en el momento en que lograra subir y ejecutar este archivo en el servidor vulnerable, la shell remota se conectara directamente conmigo y me diera control sobre el sistema.

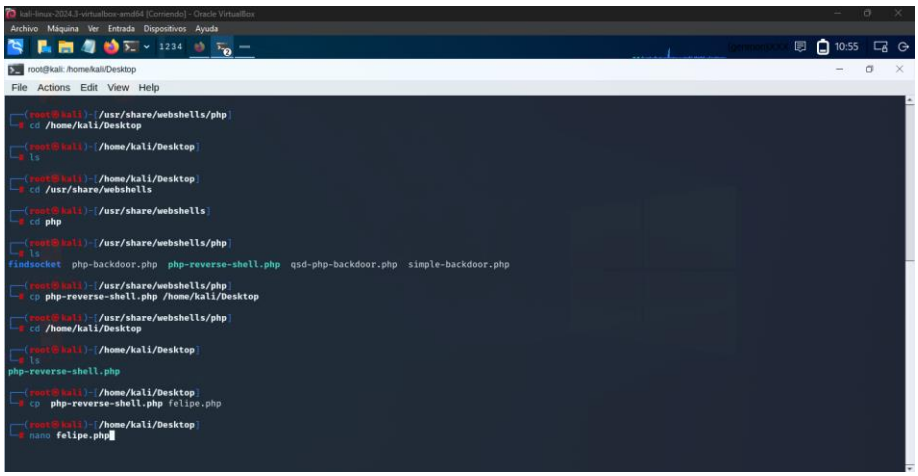

Ilustración 20renombrado la reverse shell como felipe.php

Aquí abrí el archivo **felipe.php** en el editor de texto **nano** para revisar su contenido. Este archivo es una **reverse shell en PHP**, un script que, al ejecutarse en el servidor vulnerable, intenta establecer una conexión inversa con mi máquina atacante.

Dentro del archivo se pueden ver los comentarios iniciales del código, donde se explica su uso y las licencias. Mi objetivo es localizar las secciones del script donde debía configurar los parámetros principales: la **dirección IP** de mi máquina (que actuaría como atacante) y el **puerto** en el que tendría activa la escucha.

Este paso era clave porque sin esa configuración personalizada, la reverse shell no sabría a dónde conectarse. Una vez modificados esos datos, el archivo quedaría listo para ser cargado al portal y, en caso de ejecutarse con éxito, me daría acceso remoto al sistema comprometido.

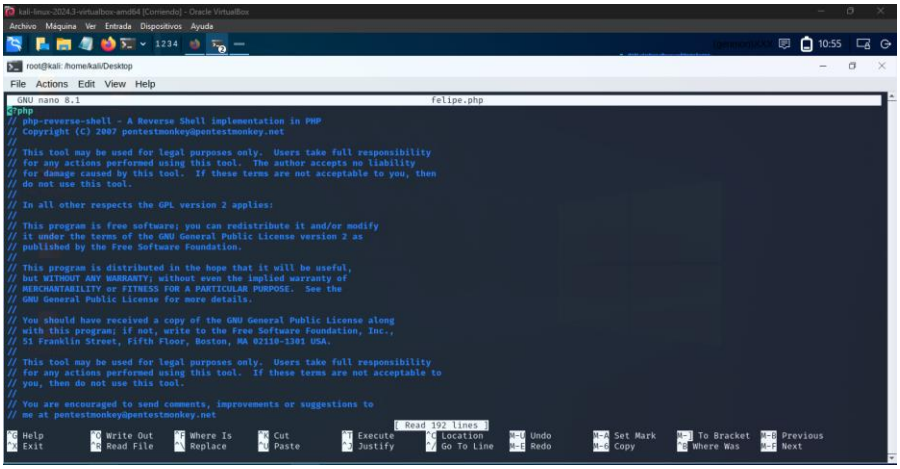

Ilustración 21la dirección IP de mi máquina

En este momento ya estaba editando el archivo **felipe.php** para configurarlo con mis propios parámetros. Dentro del código localicé las líneas donde se debía modificar la **IP** y el **puerto**.

En el campo **$ip** ingresé la dirección de mi máquina atacante (**10.10.10.5**), de manera que cuando la reverse shell se ejecutara en el servidor vulnerable, supiera exactamente a qué máquina debía conectarse. También ajusté el valor de **$port** para que coincidiera con el puerto que más adelante iba a dejar en escucha en mi equipo.

Con estos cambios me aseguré de que el archivo no usara valores genéricos, sino mi propia configuración. Así, cuando subiera y ejecutara esta shell en el portal, podría recibir la conexión inversa y comenzar a interactuar de forma remota con la máquina objetivo.

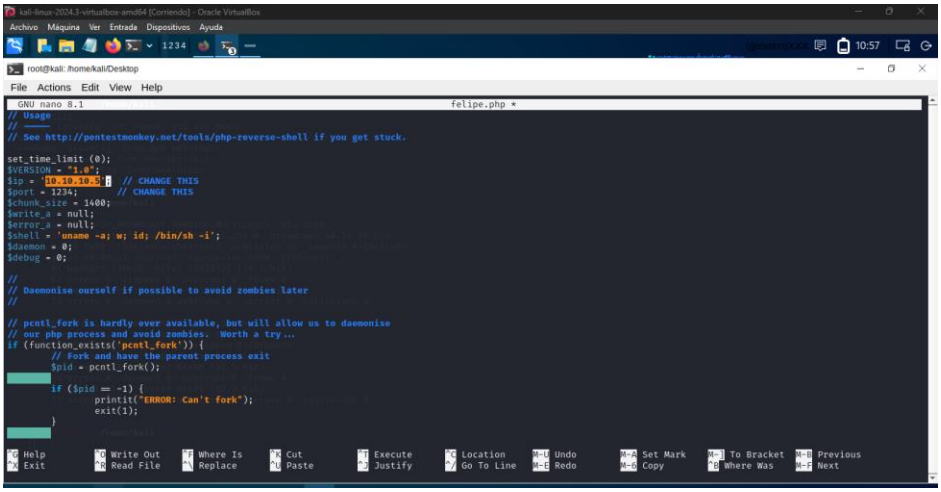

Ilustración 22editando el archivo felipe.php

En este paso lo que hice fue preparar el archivo malicioso para que pudiera ser subido sin levantar sospechas. Como el portal de administración tenía una opción de **“upload file”**, era probable que solo aceptara ciertos formatos de imagen como PNG o JPG.

Por esa razón, después de haber configurado mi **reverse shell (felipe.php)**, cambié la extensión del archivo para que se viera como una imagen, llamándolo **felipe.php.png**. De esa forma, aunque el contenido seguía siendo un script en PHP, el nombre del archivo podía engañar al sistema de validación de la subida.

En la captura se observa cómo primero tenía el archivo **felipe.php**, y luego fui creando y verificando la versión modificada con la doble extensión.**php.png**, junto con otra prueba llamada **asprilla.png**. Con esto ya estaba listo para intentar cargar el archivo en el portal y comprobar si

el servidor lo aceptaba y, posteriormente, si se podía ejecutar para establecer la conexión inversa con mi máquina.

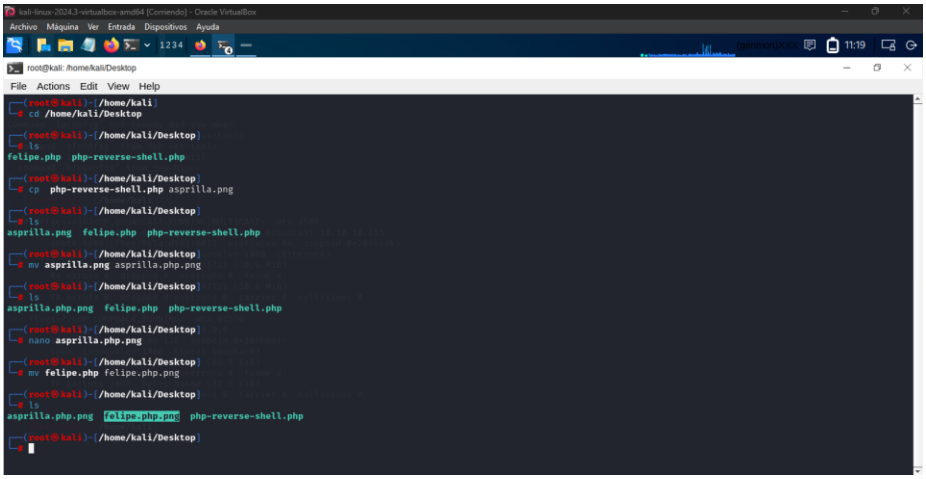

Ilustración 23preparar el archivo malicioso

Aquí intenté ejecutar el archivo malicioso que había preparado y subido al portal, el cual renombré como **felipe.php.png** para tratar de evadir los filtros de validación.

Al acceder directamente a la ruta **/fristil/uploads/felipe.php.png**, esperaba que el servidor lo interpretara como un archivo PHP y que con ello se estableciera la conexión inversa hacia mi máquina. Sin embargo, ¡en la pantalla solo apareció el mensaje “no!”, lo que indicaba que el sistema tenía algún tipo de restricción que impedía ejecutar directamente ese archivo o que detectó la extensión camuflada.

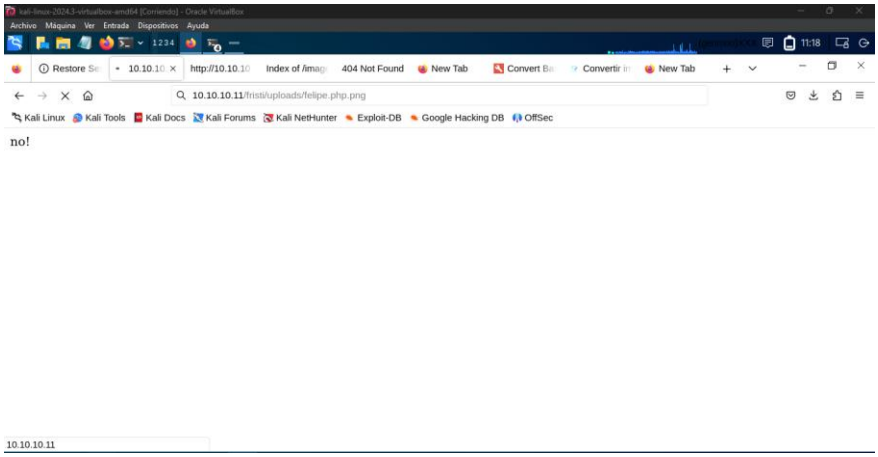

Ilustración 24ruta /fristil/uploads/felipe.php.png,

Registro de Usuarios en la página de Fristileaks

En este paso logré finalmente establecer la **conexión inversa** con la máquina víctima.

Primero, en mi máquina Kali configuré la escucha con el comando **nc -lvp 1234**, indicando que estaría esperando conexiones entrantes en el puerto 1234. Luego, tras haber preparado y subido el archivo **felipe.php**, cuando se ejecutó en el servidor, este lanzó la reverse shell hacia mi equipo.

En la parte inferior de la terminal se ve cómo la conexión fue exitosa: aparece información del sistema víctima (una máquina con Linux, kernel 3.2.0, arquitectura x86_64) y además muestra que la sesión se abrió bajo el usuario **apache**, lo cual confirma que tomé control inicial con permisos limitados.

Después de obtener acceso, probé algunos comandos básicos como **whoami** y **ping** hacia mi propia máquina (10.10.10.5) para verificar la comunicación, y todo funcionó correctamente.

Este fue un paso clave en la explotación porque a partir de aquí ya tenía acceso remoto al servidor, lo que me permitiría continuar con la escalada de privilegios y un control más completo del sistema.

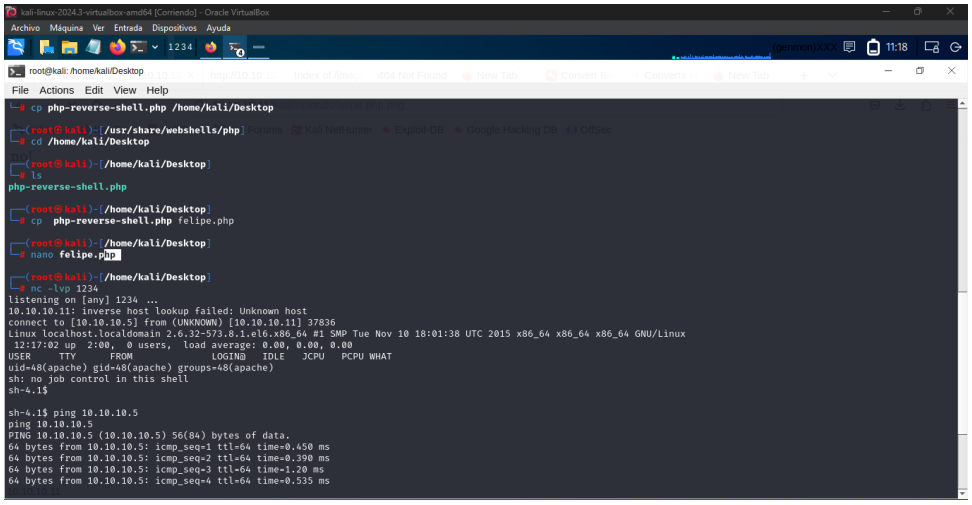

Ilustración 25nc -lvp 1234,

En esta parte del procedimiento se puede ver que logré establecer una conexión con la máquina víctima a través de un puerto específico (1234). Al ejecutarse la conexión, la terminal me muestra información del sistema comprometido, como el nombre del host, la versión del kernel y los usuarios conectados.

Una vez dentro, conseguí acceder a una **shell interactiva**, lo que me permitió empezar a ejecutar comandos directamente en el sistema remoto. Por ejemplo, utilicé el comando ls -la para listar los directorios y archivos principales de la máquina. Allí se observa la estructura típica de un sistema Linux, con carpetas como /boot, /etc, /home, /lib y otras que confirman que tengo acceso al entorno interno de la víctima.

Este paso demuestra que la vulnerabilidad fue explotada con éxito, ya que pasé de un intento de conexión a tener acceso remoto al sistema y poder interactuar con sus archivos.

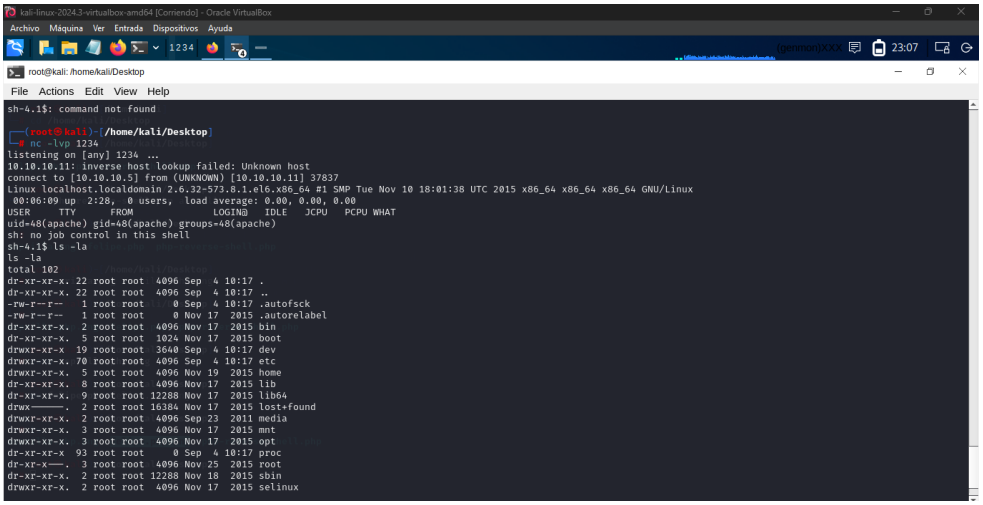

Ilustración 26shell interactiva

En esta parte de la vulneración se observa que, después de establecer la conexión inicial con la máquina víctima, mejoré la sesión obtenida para que fuera más cómoda de usar.

Primero, ejecuté un comando en Python que me permitió **convertir la shell básica en una shell interactiva**. Esto lo hice utilizando pty.spawn("/bin/bash"), lo que me dio un entorno más completo y parecido a una terminal normal. Luego, exporté algunas variables como TERM y SHELL para que la terminal se comportara correctamente y reconociera los comandos de manera más estable.

Después de eso, puse la sesión en segundo plano y la recuperé con el comando fg, ajustando la configuración del terminal con stty raw -echo. Con este paso conseguí que la shell respondiera mejor a los comandos, eliminando limitaciones de la conexión inicial.

Finalmente, utilicé reset para restablecer la terminal y dejarla lista, con un acceso mucho más funcional al sistema comprometido.

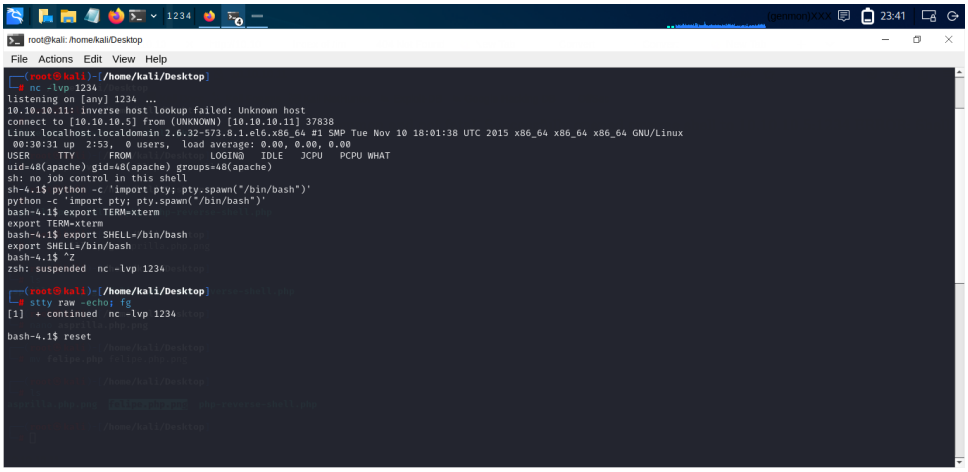

Ilustración 27pty.spawn("/bin/bash")

Aquí se muestra la continuación del acceso a la máquina víctima, ya con la **shell interactiva mejorada** que había configurado antes. En este punto, ejecuté nuevamente el comando ls -la para inspeccionar con más detalle el contenido del directorio raíz del sistema.

El resultado confirmó que tenía visibilidad sobre las carpetas principales de Linux, entre ellas /bin, /boot, /etc, /home, /lib, /usr, /var, entre otras. También se observan directorios relacionados con la seguridad y configuración del sistema, como /selinux, lo cual indica que el acceso obtenido es bastante profundo.

Este listado de directorios me permitió tener una idea más clara de la estructura interna del sistema y verificar que efectivamente tenía un control real sobre él. A partir de este punto, ya estaba en condiciones de seguir explorando archivos sensibles o buscando formas de escalar privilegios para conseguir un acceso completo como administrador (root).

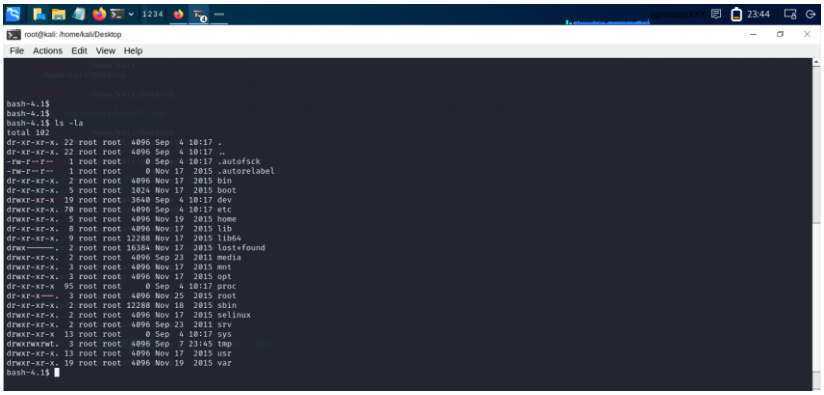

Ilustración 28listado de directorios

En esta parte de la exploración dentro del sistema comprometido, decidí moverme al directorio **/var/www**, que normalmente es donde se encuentran los archivos relacionados con los servicios web.

Al listar el contenido con ls -la, pude observar varias carpetas típicas de un servidor web, como **cgi-bin**, **html**, **error** e **icons**, que suelen contener scripts, páginas y configuraciones del sitio. Además, encontré un archivo llamado **notes.txt**, lo cual es interesante porque este tipo de archivos pueden incluir información sensible, como recordatorios, configuraciones o incluso credenciales que podrían ayudar en un proceso de escalada de privilegios.

Este hallazgo demuestra que ya no solo tengo acceso al sistema operativo en general, sino que también puedo revisar directamente los recursos del servidor web.

```
sh-4.1$ cd /var/www
cd /var/www
sh-4.1$ ls -la
ls -la
total 28
drwxr-xr-x.  6 root root 4096 Nov 17  2015 .
drwxr-xr-x. 19 root root 4096 Nov 19  2015 ..
drwxr-xr-x.  2 root root 4096 Aug 24  2015 cgi-bin
drwxr-xr-x.  3 root root 4096 Nov 17  2015 error
drwxr-xr-x.  7 root root 4096 Nov 25  2015 html
drwxr-xr-x.  3 root root 4096 Nov 17  2015 icons
-rw-r--r--   1 root root   98 Nov 17  2015 notes.txt
sh-4.1$
```

Ilustración 29directorio /var/www

En este paso, continué explorando dentro del directorio **/var/www/html**, que es donde normalmente se almacenan los archivos principales que conforman un sitio web.

Al listar el contenido con ls, encontré varias carpetas y archivos interesantes. Entre ellos destacan **beer**, **cola**, **fristi** y **sisi**, que parecen ser directorios o secciones adicionales del sitio, posiblemente usados para organizar recursos o funcionalidades específicas. También se observan archivos importantes como **index.html**, que generalmente corresponde a la página principal del sitio, y **robots.txt**, que suele contener indicaciones para los buscadores pero que, en algunos casos, puede revelar rutas ocultas o sensibles dentro del servidor.

Además, aparece la carpeta **images**, donde normalmente se almacenan recursos gráficos, pero que en algunos escenarios puede ocultar ficheros de interés si no se administra correctamente.

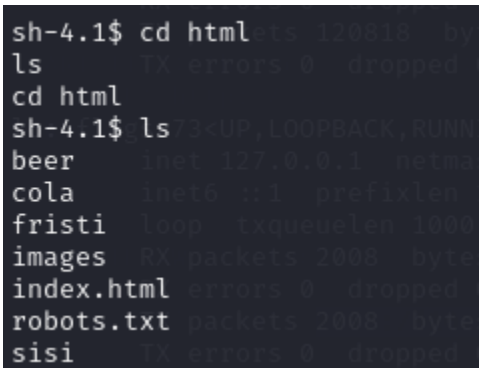

Ilustración 30Al listar el contenido con ls

En este punto ingresé al directorio **fristi**, que se encontraba dentro de la carpeta principal del sitio web. Una vez dentro, al listar los archivos, aparecieron varios ficheros de interés relacionados con el funcionamiento del portal.

Pude identificar archivos como **main_login.php**, **checklogin.php**, **login_success.php** y **logout.php**, lo cual indica que en este directorio se maneja todo el sistema de autenticación del sitio. Esto es relevante porque significa que desde aquí se controlan los accesos de usuarios, y podrían existir vulnerabilidades relacionadas con validación de credenciales o manejo de sesiones.

También encontré archivos relacionados con la subida de ficheros, como **do_upload.php** y **upload.php**, además de un directorio **uploads**. Estos son puntos especialmente sensibles, ya que los formularios de carga suelen ser una puerta de entrada para ejecutar código malicioso si no están bien protegidos.

Por otro lado, aparecieron dos ficheros con extensión. b**64** (**pic. b64** y **pic2.b64**), que probablemente contienen datos codificados en base64.

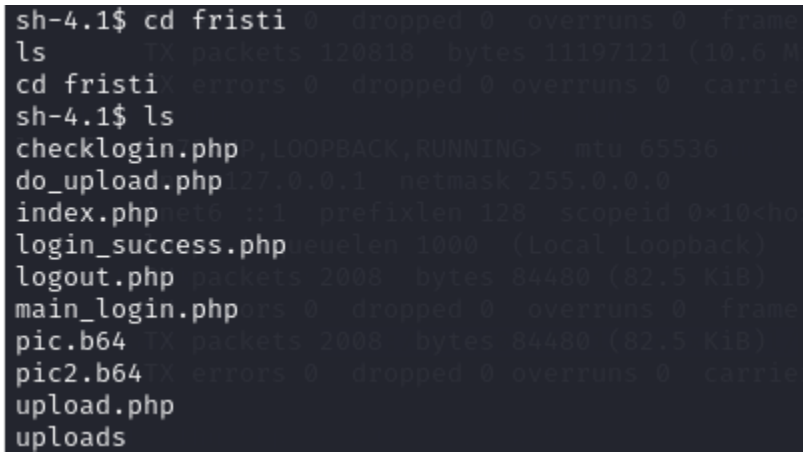

Ilustración 31directorio fristi

Aquí se muestra el momento en el que decidí moverme directamente al directorio **/var/www/html/fristi**. Este paso lo realicé para entrar en la ruta exacta donde se encontraban los archivos del sitio web que ya había detectado previamente.

El simple hecho de acceder a esta carpeta me permitió confirmar que tenía control total para navegar por el contenido del servidor web y revisar los ficheros asociados a esa sección en específico. Desde este punto ya podía analizar más a fondo los archivos que gestionaban la autenticación, la subida de ficheros y otros procesos críticos del sitio.

Ilustración 32directorio /var/www/html/fristi

En esta parte abrí el archivo **checklogin.php** para revisar su contenido. Dentro de él encontré información muy sensible, ya que el código mostraba directamente las **credenciales de acceso a la base de datos MySQL**.

El archivo contenía lo siguiente:

- **Usuario de la base de datos**: eezeepz
- **Contraseña**: AllMaal12#
- **Base de datos**: hackmeow
- **Tabla usada**: members

El código estaba diseñado para validar los datos que un usuario introduce en el formulario de inicio de sesión. Comparaba el username y el password con los registros almacenados en la tabla members. Aunque incluía algunas funciones para evitar inyecciones SQL (mysql_real_escape_string y stripslashes), la información expuesta en el propio archivo ya representaba una vulnerabilidad crítica.

```
sh-4.1$ cat checklogin.php
cat checklogin.php
<?php
ob_start();
$host="localhost"; // Host name
$username="eezeepz"; // Mysql username
$password="4ll3maal12#"; // Mysql password
$db_name="hackmenow"; // Database name
$tbl_name="members"; // Table name

// Connect to server and select databse.
mysql_connect("$host", "$username", "$password")or die("cannot connect");
mysql_select_db("$db_name")or die("cannot select DB");

// Define $myusername and $mypassword
$myusername=$_POST['myusername'];
$mypassword=$_POST['mypassword'];

// To protect MySQL injection (more detail about MySQL injection)
$myusername = stripslashes($myusername);
$mypassword = stripslashes($mypassword);
$myusername = mysql_real_escape_string($myusername);
$mypassword = mysql_real_escape_string($mypassword);

$sql="SELECT * FROM $tbl_name WHERE username='$myusername' and password='$mypassword'";
$result=mysql_query($sql);

// Mysql_num_row is counting table row
$count=mysql_num_rows($result);

// If result matched $myusername and $mypassword, table row must be 1 row

if($count==1){
```

Ilustración 33el archivo checklogin.php

Al revisar el archivo **checklogin.php** descubrí información crítica del sistema, ya que en su código estaban escritas en texto claro las credenciales de acceso a la base de datos MySQL (usuario: *eezeepz*, contraseña: *AllMaal12#*, base de datos: *hackmeow*, tabla: *members*). El archivo se encarga de validar los datos de inicio de sesión comparando el nombre de usuario y la contraseña con los registros de la base de datos, y aunque incluye funciones básicas para evitar inyecciones SQL, su nivel de protección es muy limitado.

```
// Register $myusername, $mypassword and redirect to file "login_success.php"
session_register("myusername");
session_register("mypassword");
header("location:login_success.php");
}
else {
echo "Wrong Username or Password";
}

ob_end_flush();
?>
```

Ilustración 34revisar el archivo checklogin.php

En este punto utilicé las credenciales descubiertas en el archivo **checklogin.php** para conectarme directamente al gestor de base de datos MySQL. Primero, ejecuté el comando en Python para mantener una shell interactiva más estable, y luego probé el acceso con el usuario **eezeepz** y la contraseña **AllMaal12#**. La conexión fue exitosa y logré entrar al monitor de MySQL, donde se confirmó que estaba trabajando con la versión **5.1.73** del servidor. Este paso fue clave porque demostró que las credenciales expuestas en el código eran completamente válidas, permitiéndome acceder directamente a la base de datos y abrir la posibilidad de explorar sus tablas, consultar información sensible almacenada en ellas o incluso buscar formas de escalar privilegios a nivel del sistema.

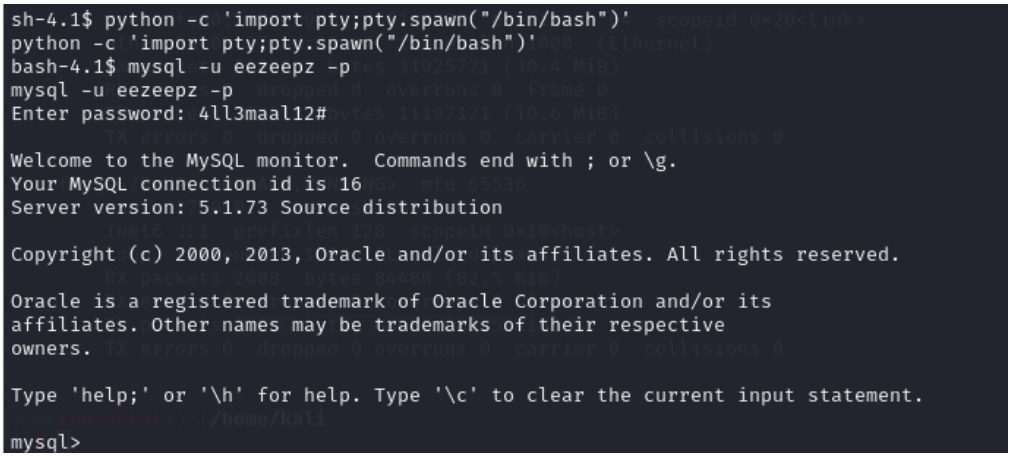

Ilustración 35credenciales descubiertas en el archivo checklogin.php

Una vez dentro de MySQL, ejecuté el comando **SHOW DATABASES;** para visualizar las bases de datos disponibles y encontré dos: **information_schema**, que es la propia del sistema, y **hackmenow**, que correspondía al sitio vulnerable. Decidí trabajar sobre **hackmenow**, por lo que utilicé el comando **USE hackmenow;** para seleccionarla y así poder interactuar directamente con sus tablas y registros. Con esto logré cambiar el contexto de trabajo hacia la

base de datos que contenía la información sensible, quedando listo para listar y explorar el contenido almacenado en ella.

Cuando ejecutaste el comando **SHOW TABLES;** dentro de la base de datos **hackmenow**, lograste visualizar todas las tablas que contenía. Esto permitió identificar la estructura interna de la base de datos y facilitó el acceso a la información almacenada en ellas, como usuarios, credenciales y otros datos relevantes. De esta forma, ya con las tablas listadas, el siguiente paso fue inspeccionar su contenido para obtener datos sensibles que podían ser aprovechados en el proceso de explotación.

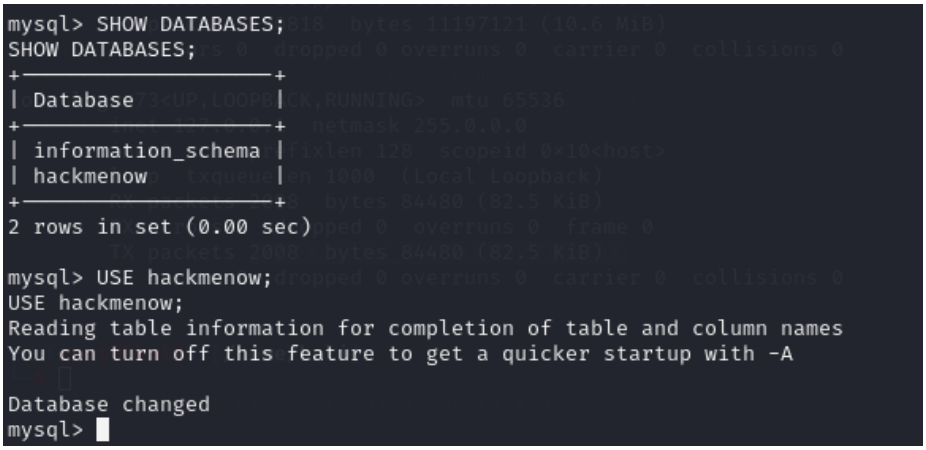

Ilustración 36SHOW DATABASES

En este punto, después de conectarme a la base de datos **hackmenow**, ejecuté el comando **SHOW TABLES;** y observé que solo existía una tabla llamada **members**. Para conocer su estructura interna, utilicé el comando **DESCRIBE members;**, lo que me mostró que dicha tabla contenía tres campos: **id**, configurado como entero y clave primaria con auto incremento, **username** de tipo varchar(65) y **password** también de tipo varchar(65). Con esta información

confirmé que la tabla almacenaba usuarios y contraseñas, lo que indicaba que era la sección más sensible y de interés para continuar con la extracción de credenciales.

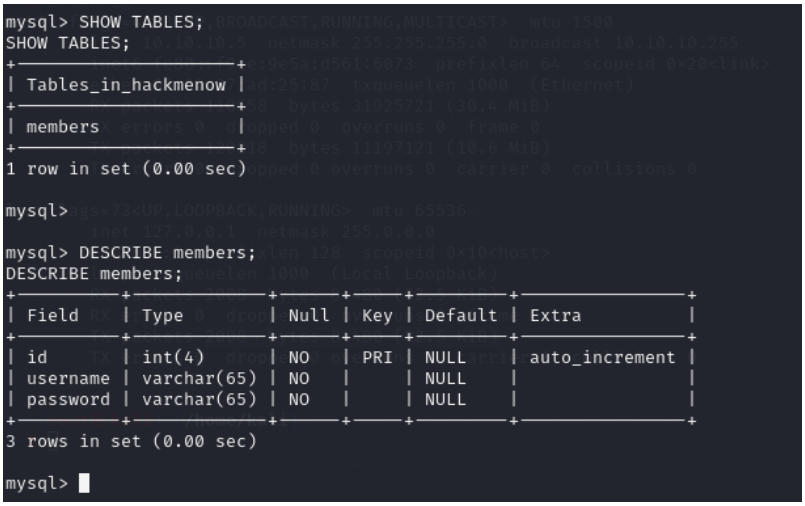

Ilustración 37SHOW TABLES

Dentro de la base de datos ejecuté el comando **SELECT * FROM members;**, lo que me permitió visualizar directamente el contenido de la tabla **members**. Allí encontré un único registro con el usuario **eezeepz** acompañado de su contraseña en texto claro **keKkeKKeKKeKkEkkEk**, lo que confirmó que la aplicación almacenaba las credenciales de forma insegura y que ya contaba con acceso válido para continuar explotando el sistema.

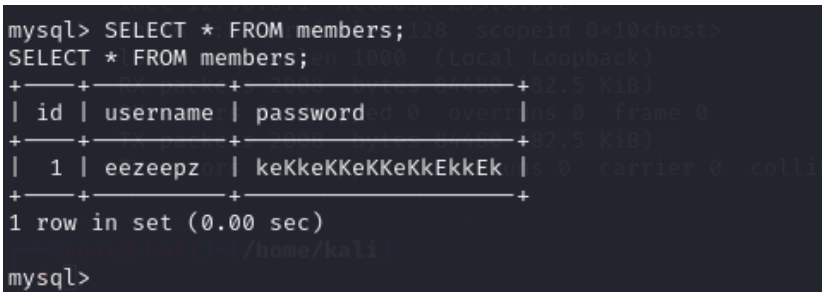

*Ilustración 38SELECT * FROM members*

Después de obtener acceso a la base de datos y comprobar la existencia de la tabla **members**, realicé la inserción de nuevos registros para añadir credenciales personalizadas. Con el comando **INSERT INTO members (username, password) VALUES ('Andres1', 'Asprilla1');** agregué un primer usuario con su respectiva contraseña, confirmando que la consulta fue exitosa al mostrar **1 row affected**. Posteriormente, ejecuté otro **INSERT** para crear un segundo usuario con los valores **('Felipe2', 'Cordoba2')**, obteniendo nuevamente una inserción correcta. Esto demostró que tenía control total para manipular los datos almacenados en el sistema, pudiendo crear accesos adicionales a la aplicación según lo requiriera.

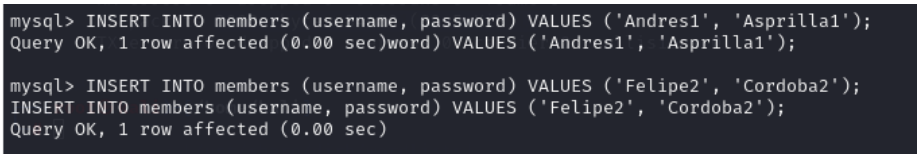

Ilustración 39la inserción de nuevos registros

Finalmente, después de insertar los nuevos usuarios en la tabla **members**, verifiqué los cambios con el comando **SELECT * FROM members;**, lo que me mostró que ahora había tres registros dentro de la base de datos. En la salida aparecía el usuario original **eezeepz** junto con su contraseña cifrada, y además los dos usuarios que yo había creado previamente: **Andres1 / Asprilla1** y **Felipe2 / Cordoba2**. Esto confirmó que la manipulación fue exitosa y que logré añadir credenciales propias dentro del sistema, lo que me permitiría acceder con diferentes combinaciones de usuarios y contraseñas según lo necesitara.

```
mysql> SELECT * FROM members;
SELECT * FROM members;
+----+----------+--------------------+
| id | username | password           |
+----+----------+--------------------+
|  1 | eezeepz  | keKkeKKeKKeKkEkkEk |
|  2 | Andres1  | Asprilla1          |
|  3 | Felipe2  | Cordoba2           |
+----+----------+--------------------+
3 rows in set (0.00 sec)

mysql>
```

*Ilustración 40SELECT * FROM members;,*

Como paso final de la vulneración, después de haber insertado en la base de datos los nuevos usuarios con sus credenciales, accedí al navegador y abrí la dirección del sitio vulnerable en la ruta **/fristigod/**, donde se encontraba el formulario de login. Allí ingresé el usuario **Andres1** y la contraseña **Asprilla1**, los mismos que había añadido previamente en la tabla *members*.

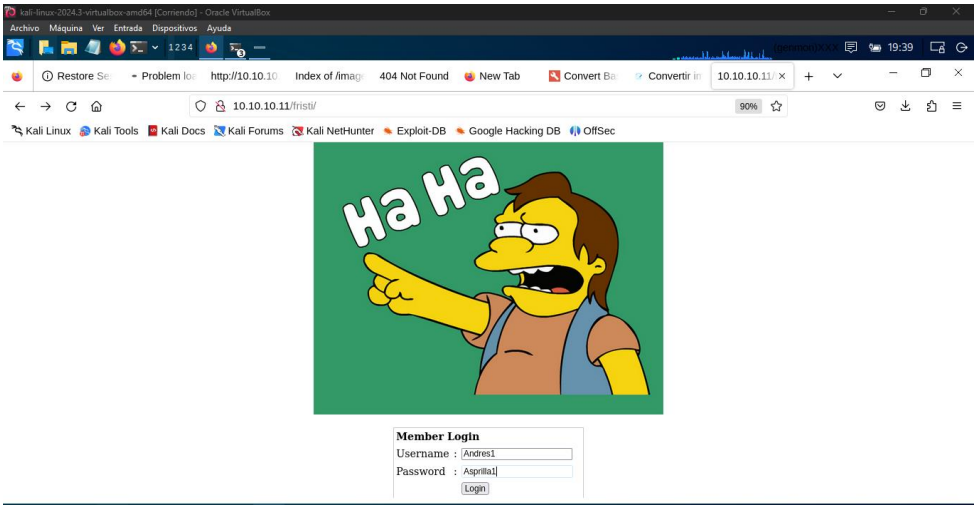

Ilustración 41Acceso con usuario Andres1

Al autenticarme, el sistema aceptó las credenciales, demostrando que el acceso creado funcionaba correctamente y que era posible entrar al sitio como un usuario válido gracias a la manipulación de la base de datos.

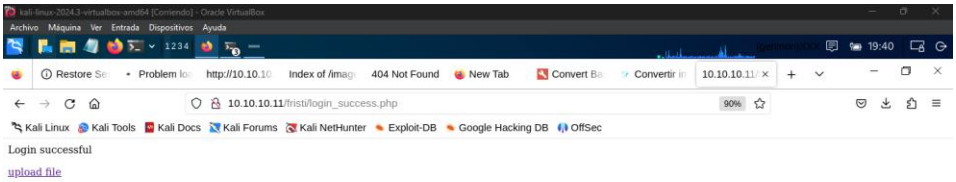

Ilustración 42Acceso exitoso de Andres1

validé también el segundo usuario que había creado en la base de datos. Accedí nuevamente al formulario de inicio de sesión del sitio vulnerable y esta vez ingresé las credenciales **Felipe2 / Cordoba2**, comprobando que el sistema aceptaba este acceso sin problemas.

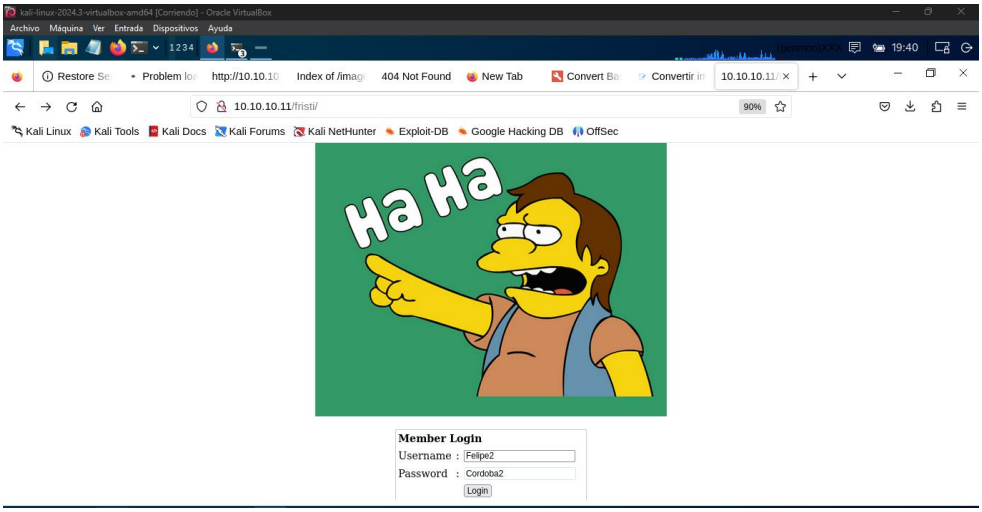

Ilustración 43Acceso con usuario Asprilla1

Esto confirmó que no solo era posible manipular los registros internos de la base de datos, sino también utilizarlos de manera efectiva para iniciar sesión en la aplicación web, consolidando así la vulnerabilidad explotada y evidenciando la falta de validaciones y medidas de seguridad en el manejo de usuarios.

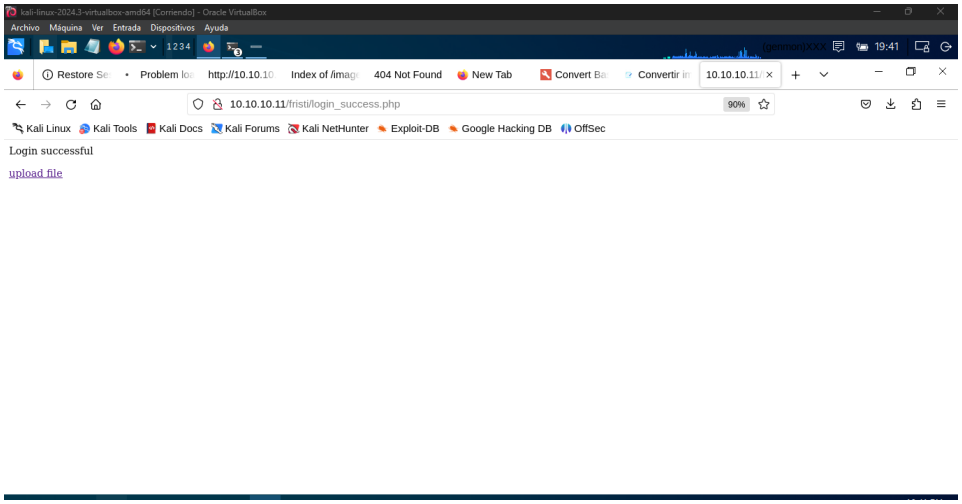

Ilustración 44Acceso exitoso con Asprilla1

Universidad Tecnológica del Chocó
Diego Luis Córdoba

Configuracion del dhcp por medio de la Mac a la máquina virtual de Ubuntu

Cambio de la Mac predeterminada por una asignada

La configuración de red está en modo Red **NAT**, lo que permite que la máquina virtual tenga acceso a internet a través del host, mientras que la **dirección MAC 0800273CCB05** identifica de manera única al adaptador de red virtual, garantizando la comunicación dentro de la red asignada.

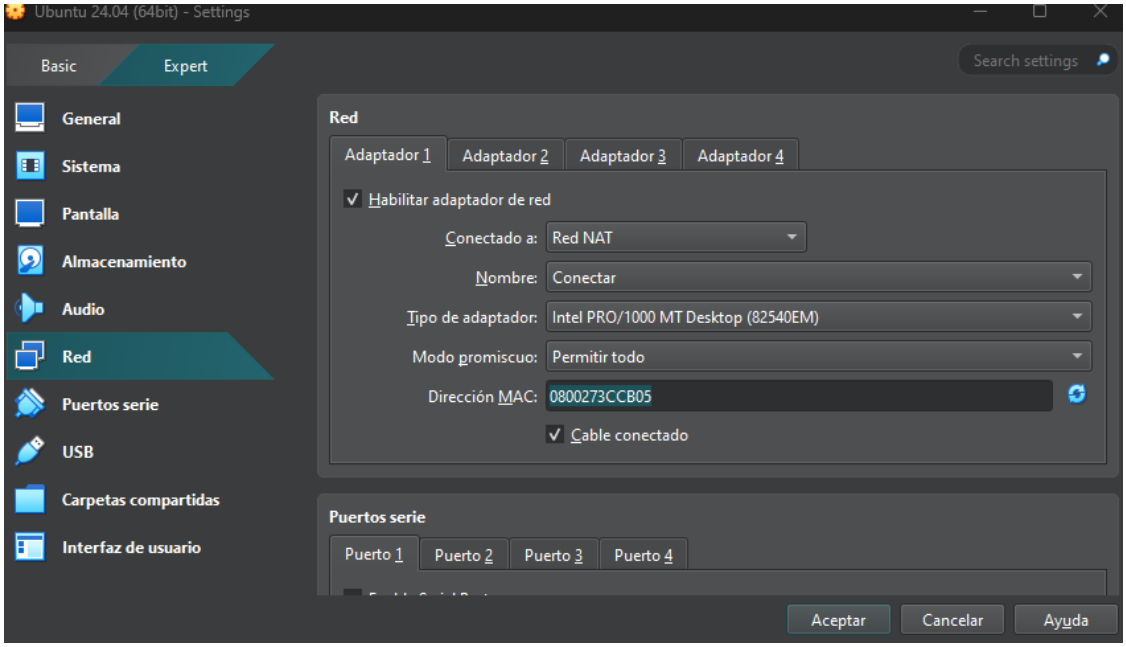

Ilustración 45Mac por defecto

Aquí se observa la configuración de red del sistema. La interfaz **enp0s3** está activa y tiene asignada la IP **10.10.10.13/24**, lo que indica que está dentro de una red privada. Además, aparece la **dirección MAC 08:00:27:3c:cb:05**, que identifica de forma única al adaptador de red y permite la comunicación dentro de la red local.

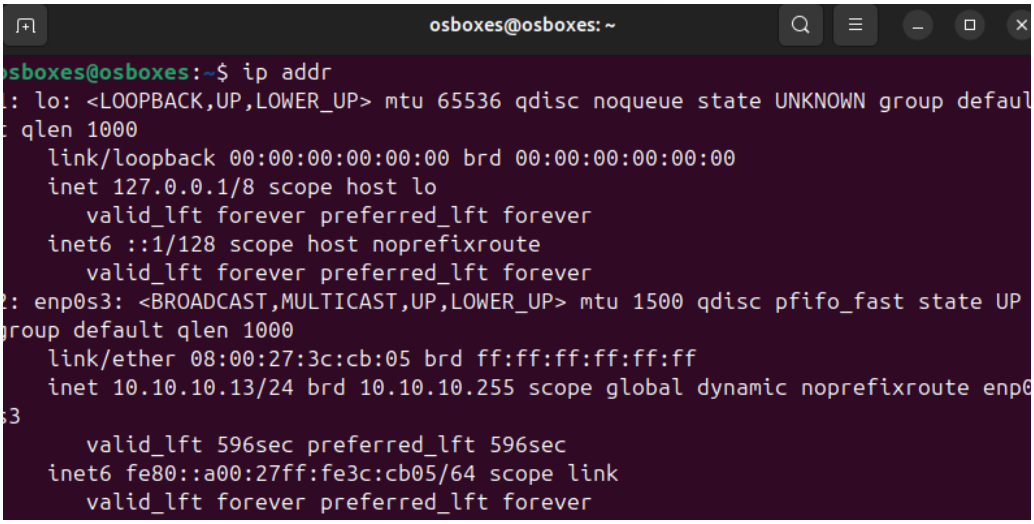

Ilustración 46Ver ip asignada por defecto

En este paso deshabilité la interfaz de red **enp0s3** utilizando el comando sudo ip link set dev enp0s3 down, lo que provocó que la tarjeta de red pasara a estado **DOWN**. De esta forma, la máquina quedó sin conectividad, ya que no se le asigna ninguna dirección IP, aunque la dirección **MAC 08:00:27:3c:cb:05** sigue apareciendo como parte del adaptador físico.

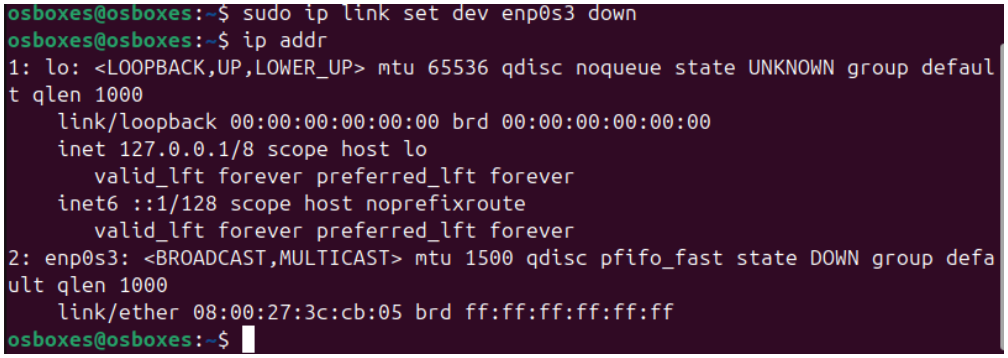

Ilustración 47Apagar la salida a la red

Lo que realicé fue cambiar la dirección MAC de mi tarjeta de red, asignándole manualmente una nueva (08:00:27:A5:A6:76) y luego volví a activar la interfaz. Al verificar con

el comando ip addr, confirmé que la tarjeta quedó con la nueva MAC y obtuvo su dirección IP (10.10.10.14), demostrando que el cambio se aplicó correctamente y la red continuó funcionando sin problemas.

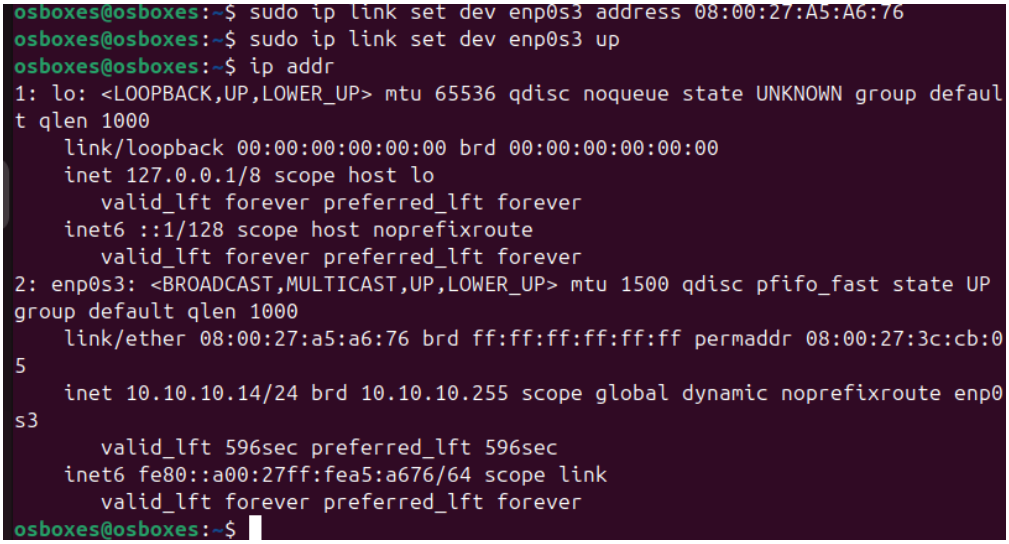

Ilustración 48configuracion de la Mac asignada por comando

Recomendaciones

- Configurar adecuadamente los permisos en directorios y archivos sensibles, evitando la exposición de información en texto claro.
- Implementar cifrado de contraseñas y buenas prácticas en la gestión de usuarios y credenciales dentro de las bases de datos.
- Restringir las funcionalidades de carga de archivos para impedir la ejecución de código malicioso en los servidores web.
- Fortalecer las configuraciones de red y monitorear cambios en direcciones MAC que puedan indicar intentos de suplantación de identidad en la red.
- Actualizar los servicios y sistemas operativos a versiones recientes para reducir riesgos de explotación de vulnerabilidades conocidas.

Conclusión

El desarrollo de la práctica permitió comprobar cómo la falta de medidas de seguridad adecuadas puede ser aprovechada en diferentes niveles de un sistema. En el caso del portal **Fristileaks**, se logró acceder a información sensible gracias a configuraciones débiles, como la exposición de directorios y credenciales en archivos de código.

Por otro lado, el ejercicio de cambio de dirección **MAC en Ubuntu** demostró lo sencillo que resulta modificar la identidad de un adaptador de red. Aunque se trata de una práctica técnica básica, pone en evidencia la debilidad de los sistemas que confían únicamente en el filtrado de direcciones físicas como medida de seguridad. Este aspecto debe complementarse con otras capas de protección en la red para garantizar un mayor nivel de confiabilidad.

La práctica combinada permitió no solo adquirir experiencia en técnicas de explotación y administración de red, sino también reflexionar sobre la necesidad de mejorar las configuraciones en sistemas y aplicaciones. Ambos escenarios demostraron que la seguridad debe ser abordada de forma integral, tanto en el software como en la red, con el fin de reducir riesgos y proteger los recursos frente a posibles atacantes.

Universidad Tecnológica del Chocó
Diego Luis Córdoba

Referencia bibliográfica

Ingeniero Rafael Sandoval