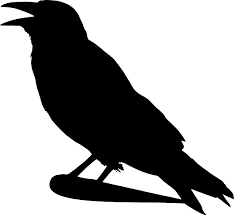
**RAVEN PENETRATION TEST REPORT**



***By: Andrew Parker***

**Executive Summary**

The importance of these findings is to show the vulnerabilities within the Raven server. The findings, which can be shown below, describes in detail the number of exploited vulnerabilities that were discovered during the penetration test. Some of these vulnerabilities are more critical than others, meaning these high critical vulnerabilities can significantly impact the business. These vulnerabilities can cause lost/stolen data, website defacement, denial-of-service, and can damage reputation. One of the most critical vulnerabilities found was an open SSH port. Knowing that this port is open to any user is extremely dangerous and can allow attackers to brute-force usernames and logins. Once a login is found, the attacker can begin to search for any sensitive data from within the server. This type of vulnerability should immediately be addressed since the amount of potential attacks that can come from logging in through SSH are essentially endless.

In addition to port 22 (SSH) being open, another critical vulnerability that should immediately be addressed in order to protect the business is how exposed the web server is. The exposed web server means that directories within the web server are front-facing/accessible by anybody. An attacker can use different scanning tools to enumerate all directories within a web server to further expand their attack surface. Someone could identify a directory for a vulnerable program that can have a known exploit. Another critical vulnerability discovered was poor security access controls. Any user (sudo or not) has access to sensitive files that are used for configuration or have certain information stored that should not be exposed to just anybody. This means the principle of least privileged should be heavily followed. Giving somebody more access than is needed significantly increases risk and increases the odds of someone getting exposed to an attacker.

The final critical vulnerability that should immediately be addressed is weak password usage. The users had non-complex passwords that were easily able to be brute-forced within minutes. This kind of vulnerability is an obvious threat to any business since using basic passwords increases the odds of an attacker being able to gain access and even worse, root access. Simple and commonly used passwords enable intruders to easily gain access and control of a computing device.

By strengthening the complexity of the users’ passwords and creating stronger encryption, can also significantly reduce risk for the company.

Vulnerabilities Discovered:

Apache 2.4.10 is out of date: There are numerous exploits

SQL password hardcoded in config.php file

Wordpress is Vulnerable

Usernames are present in WordPress Directories

Passwords were very weak

One of the users had sudo privileges to run python

Direcotires are listable / front facing

SSH can be brute forced

Mysql config file is readable to all users which contains login credentials

NO IDS/IPS

**Attack Narrative**

**Reconnaissance**

When running nmap, a few critical ports were discovered to be open which were ports 22 and 80 (http/SSH). The command that was used to return these results was “nmap -A 172.16.125.131”. This ran an aggressive scan on the target IP address to display all open ports on the target machine. The scan also gave me the version of the server being used which was “Apache 2.4.10” which is an outdated version of Apache that has known vulnerabilities. The most critical discovery on the target machine was SSH being open. SSH provides a secure encrypted connection between two hosts over an insecure network. This can be used for terminal access, file transfers, and for tunneling other applications.

**Enumeration and Vulnerability Analysis**

|  |  |  |  |
| --- | --- | --- | --- |
| **IP Address** | **Operating System** | **Vulnerabilities** | **Risk (Low/Med/High)** |
| 172.16.125.131 | Linux 3.X|4.X | Port 22 (SSH) open   * Can be used to gain access to server with a known username. | HIGH |
| 172.16.125.131 | Apache 2.4.10 | Exposed Web Server:   * Config.php files accessible to all users. * Directories are visible/front-facing. | HIGH |
| 172.16.125.131 | Linux 3.X|4.X | User has Sudo privileges and can run Python which can cause a privilege escalation attack. | HIGH |
| 172.16.125.131 | Linux 3.4X|4.X | Passwords for users are very weak and can easily be brute-forced. | MEDIUM |
| 172.16.125.131 | Linus 3.4X|4.X | Apache 2.4.10 version is outdated. | MEDIUM |
| 172.16.125.131 | Linux 3.4X|4.X | Linux version 3.X|4.X is out of date | LOW |

**Web Server Analysis**

To begin attacking the web server, I decided to run ‘DirBuster’. DirBuster is a multi-threaded java application designed to brute force directories and file names on web applications/servers. After running DirBuster with a pre-set wordlists, we were able to see the directories that were present on the web server. One directory that stood out was WordPress. WordPress is a content management system based on PHP and MySQL that is usually used with the MySQL or MariaDB database servers. Because WordPress is so widely used it has many known vulnerabilities and tools that can be used to exploit these vulnerabilities. One of theses tools is ‘WpScan’.

WPSccan is a tool used specifically designed to find vulnerabilities in WordPress hosts. With WPScan you can also find out if any weak passwords, users, and security configuration issues are present. By running an enumeration WPScan using the command: ‘wpscan –url <http://172.16.125.131/wordpress/> -e’, we can enumerate and see what we can use for further exploitation. As a result of the scan, we were able to extract two usernames that are most likely admins who manage the site. The usernames were ‘michael’ and ‘steve’. Because port 22 (SSH) is open and we now have usernames for the webserver, we can attempt a brute-force attack in order gain access.

**Network Analysis**

Now that we have two valid usernames, we can attempt to login through SSH by brute-forcing the password. First, we will try to brute-force ‘Michael’s password by using ‘Hydra’. Hydra is a password cracking tool that can be used to brute-force usernames and/or password

credentials. In order to run ‘Hydra’ you must use a password list that is either a custom list or a pre-set list. I decided to first use a custom list that had common passwords to test and see how weak the password might be. The command I used was: ‘hydra -l Michael -P password\_list.txt ssh://172.16.125.131’, which allowed hydra to run attempt every password in my custom list to see if any were successful. The brute-force was successful and the results for the password was simply ‘michael’. Now that the password has been cracked, we can ssh into our target server. The command used to login was: ‘ssh [michael@172.16.125.131](mailto:michael@172.16.125.131)’ along with the password ‘michael’ which was a success. We were now logged into the webserver as ‘michael’ and now it’s time to escalate our privileges.

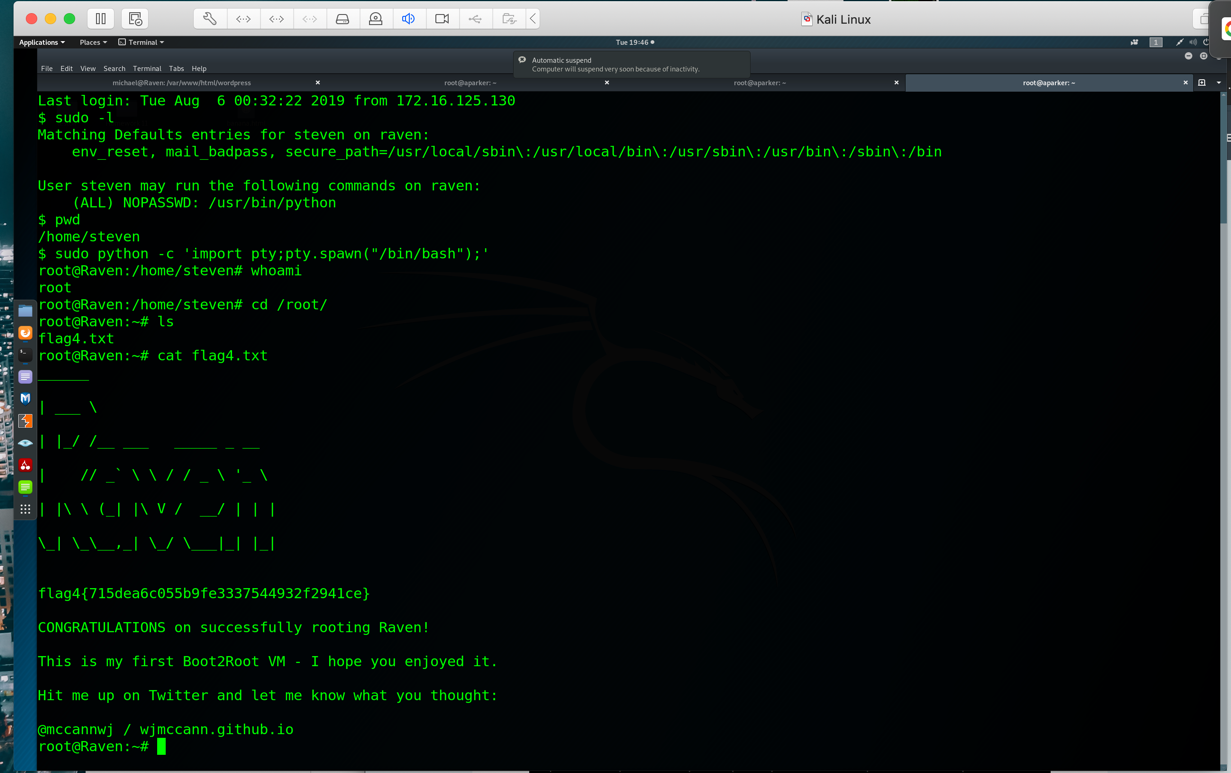
**Post-Exploitation Exploration and Privilege Escalation**

Now that I successfully logged into the target machine, I am going to first see if I can escalate my privileges. I ran the command ‘sudo -l’ to list and see what the user had sudo access for, but ‘michael’ had no sudo privileges. Although we are unable to escalate privileges with the user ‘michael’ we can still look through any files and directories that he may have access to that can assist in our reconnaissance. After doing some digging, I was able to locate a wordpress directory in the following location: ‘/var/www/html/wordpress’ which had a ‘wp-config.php’ file. I used cat to open the file and even though ‘michael’ was not a sudo user, I was able to view the file, meaning any user can access this sensitive document. Once I catted the file to view the configurations, I immediately noticed that written in plain text was the password for the MySQL database. This password was: ‘R@v3nSecurity’. I immediately tested the password by attempting to login to the SQL server using: ‘mysql -u root -p’ and entered the password for a successful login.

Once the MySQL login was successful I wanted to see what databases were available. I did this using the SQL command: ‘show databases;’ and immediately noticed a database called ‘wp\_users’. I used the command ‘use wp\_users;’ in order to see if we can maybe get a sudo username or password. To see the table in ‘wp\_users,’ I ran ‘SELECT \* FROM wp\_users’. The results gave us both usernames (Michael and Steve) and their hashed passwords. Now that we have a hashed password for ‘Steve’ we can attempt to crack the hashed password and login as ‘Steve’ in order to see if he has root access.

Once the hash for ‘steve’ was displayed, I immediately stored the hash in a file called ‘steve\_hash.txt’. In order to crack the hash, I decided to use the application John the Ripper. John the Ripper is a fast password cracker, and is primarily used to detect weak Unix passwords. I ran the command: ‘John steve\_hash.txt’ to run the application. Once finished, I noticed the results said ‘1 cracked password’. In order to view the cracked password I used: ‘john -show steve\_hash.txt’ which resulted in the cracked password: ‘pink84’.

Now that we have the credentials for steve, I immediately attempted to SSH into the target machine as ‘Steve’ which was a success. I then ran ‘sudo -l’ in order to see if Steve had any sudo privileges that we can escalate and become root. As a result ‘Steve’ does have sudo privileges for running Python. Because ‘Steve’ has sudo Python privileges, we can escalate our privileges by spawning a bin/bash shell which will elevate our privileges to root. In order to do this I ran ‘sudo python -c ‘import pty;pty.spawn(“/bin/bash”);’ Once the command was ran I can see that I became ‘root@Raven’. I ran a quick ‘pwd’ command to verify and see that we are in fact ‘root’.



**Conclusion and Recommendations**

Based on the results documented above, we recommend the client take the following steps to remediate the vulnerabilities identified on the target machine.

**Web Server**

* Update Apache 2.4.10 to newest version.
* WordPress has many known vulnerabilities. In order to reduce risk it is best to add security plugins and keeping WordPress version up-to-date.
* Take out any plain text usernames/password hashes in config/html files.
* Have users use more complex passwords.
* Create a rule to hide or change format of files to stop against enumeration attacks like ‘DirBuster’.
* Update Linux to most recent version.

**Network Services**

* Have users use more complex passwords.
* Follow the principle of least privilege by limiting any user’s sudo access on web server.
* Limit access to any sensitive files or databases (e.g. config.php)
* Use a more advanced encryption method like AES
* Create a rule for which user’s can SSH into machine
* Do not show any passwords in plain text in any files (e.g. config.php MySQL)

**Hardening the Server**

* Set an access control on who is allowed certain sudo privileges.
* Set access control on MySQL database and only limit to certain database admins
* Use complex passwords for MySQL database access.
* Store no plain text passwords in any files.
* Setup IDS/IPS and create rules to whitelist necessary IP’s.

**RAVEN WALKTHROUGH**

I started this project off by installing the Raven virtual machine and running an ‘ifconfig’ to see what the IP is. This turned out to be 172.16.125.131 which will now be our target.

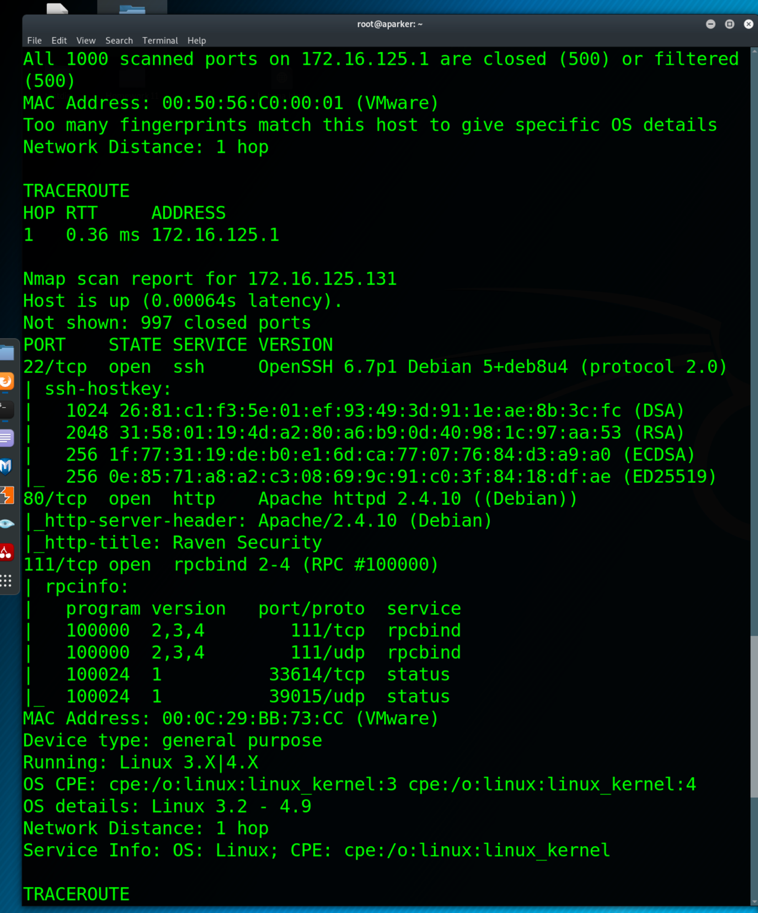
1. By running an aggressive nmap scan, you can see below that the following ports are open on the target IP: 172.16.125.131

22/tcp Open – SSH Version: OpenSSH 6 . 7p1 Debian

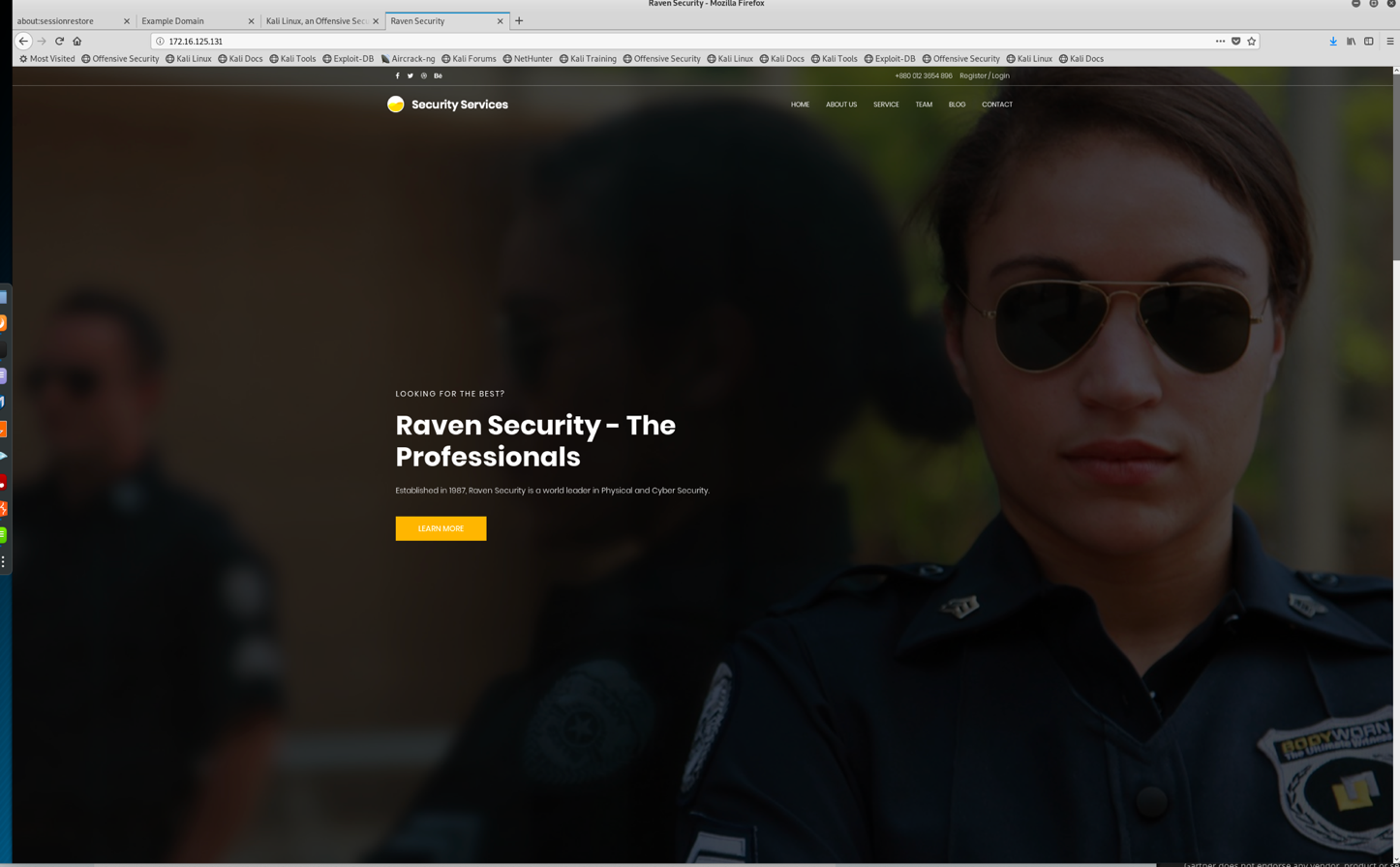
80/tcp Open – HTTP Version: Apache httpd 2.4.10 ((Debian))

111/tcp Open-rpcbind 2-4

**Command used: nmap -A 172.16.125.131**



After seeing what ports were open, the first port I decided to start with port 80. Upon visiting the website, you get the following home page shown below. Because we can see in our above scan that the server is running Apache 2.4.10, which by doing some research, I can tell that this is not the most updated version



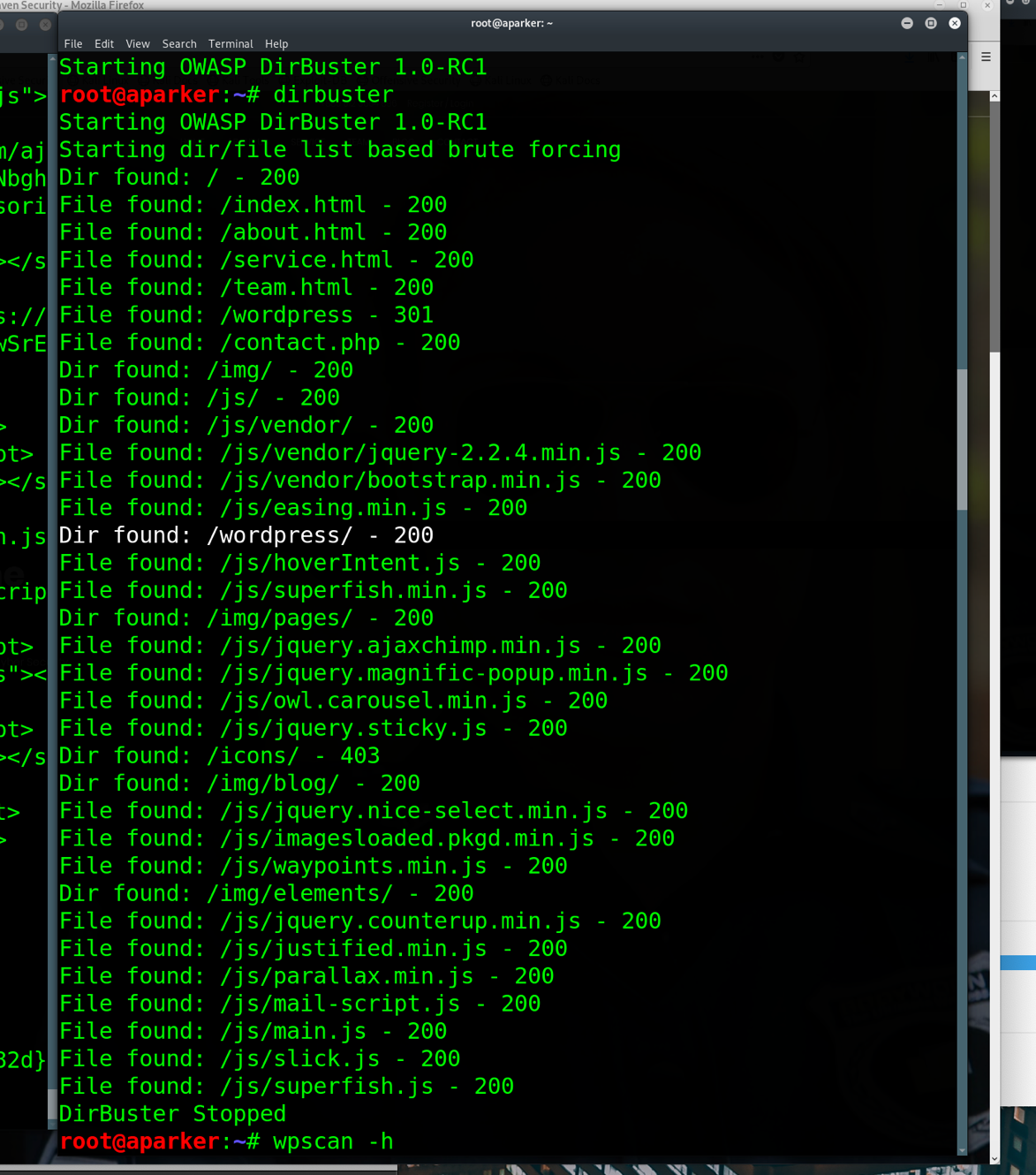
1. To see if the site was vulnerable for attacks, I decided to use Dirbuster. DirBuster is a designed to brute force directories and files names on web/application servers. Web servers usually have pages and applications that are hidden within that can be used to for further enumeration, or reconnaissance. Once DirBuster was ran, I used the target IP 172.16.125.131 as the input and used a pre-set DirBuster wordlist for Apache and ran the application.

**Command used: Dirbuster**

**Wordlist: /usr/share/dirbuster/wordlists/apache-user-enum-2.0.txt**

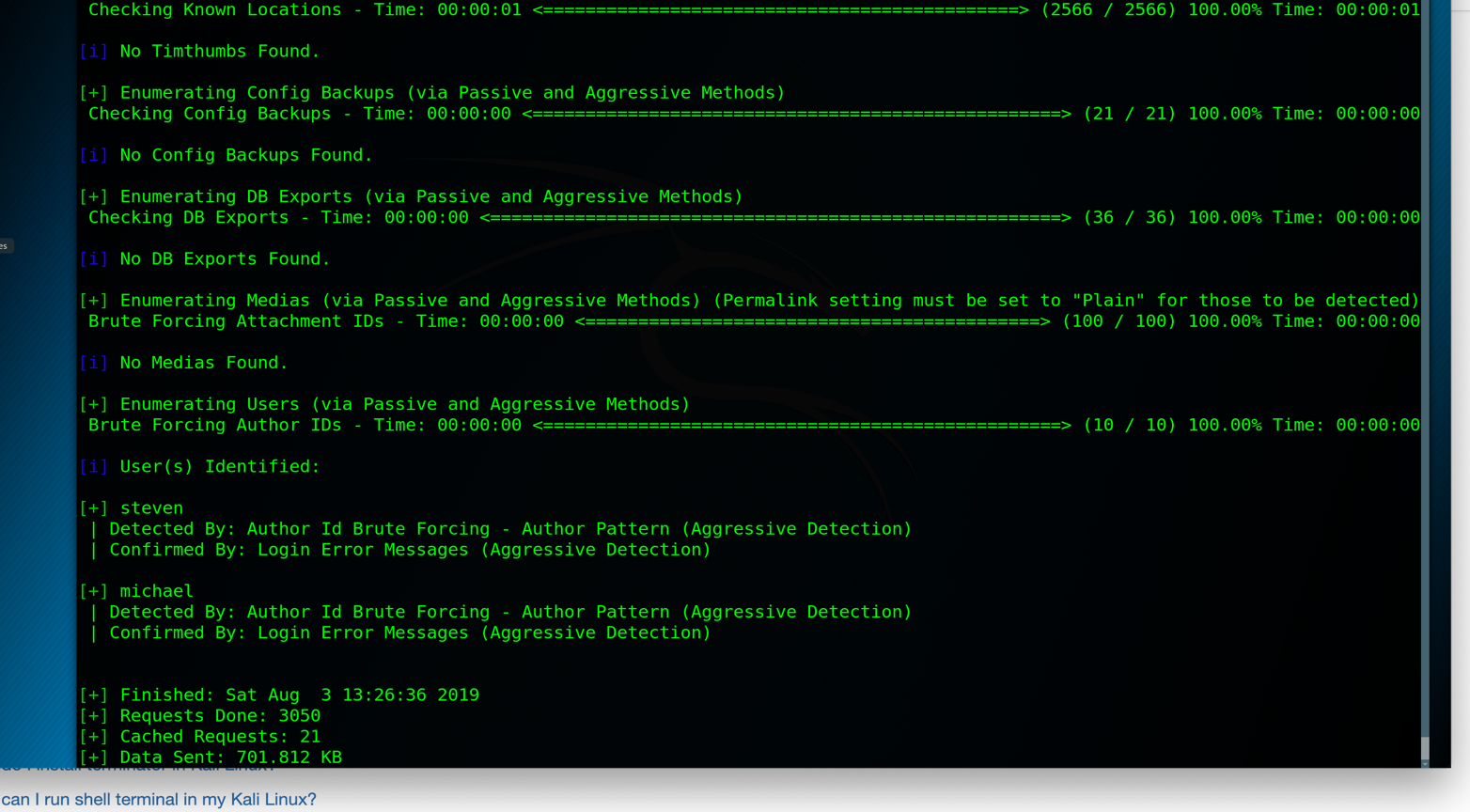


The DirBuster output can be seen below. As you can see there were many directories found; However one that stands out is a ‘wordpress’ directory. This is something that can further be of interest to use in order to find more flags. Because Wordpress is so widely used for blogs, websites, etc. it is constantly being attacked and exploited with known vulnerabilities.



1. Now that we know WordPress is being used, we can run WPscan. WPScan is a tool used specifically designed to find vulnerabilities in WordPress hosts. With WPscan, you can also find out if any weak passwords, users, and security configurations issues are present. I ran an enumeration WPscan in order to take a closer look and gain more information and reconnaissance.

**Command Used: wpscan –url <http://172.16.125.131/wordpress/> -e**



1. Once wpscan was run, the picture above were the results I obtained. I noticed that two users were identified as a result of my scan and this gave me two usernames that can be used to login to Wordpress. The next step now would be to try and brute force one of these users in order to further explore files from within the target machine. Because earlier we established that port 22 (SSH) was open, we could try to SSH into the machine using ‘michael’ as the username. To do this, I decided to use the Hydra application to attempt a brute force attack.

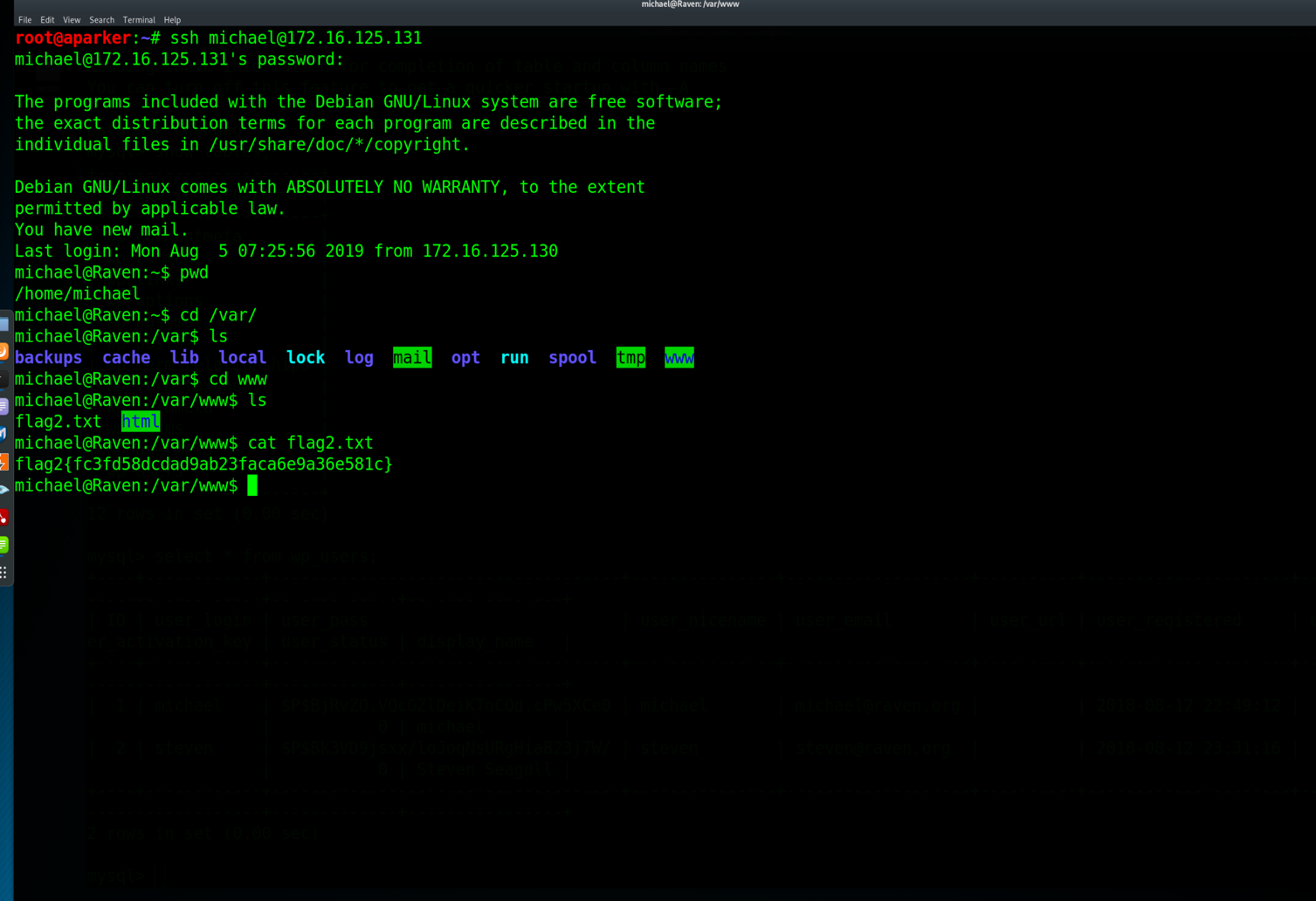


The image above was the results of running Hydra in order to brute force. I created a custom wordlist called password\_list.txt which had a list of basic passwords that can be used to try and brute force. The results showed that ‘michael’ was the correct password for that user. This now gave us a username and password that can be used to ssh into the Wordpress page.

**Command Used: hydra -l Michael -P password\_list.txt ssh://172.16.125.131**

1. Now that we have the credentials needed to SSH into the web server, we can start searching for any useful files. Upon logging in, I navigated to the /var/www directory and found flag2. We have now found two flags. After finding the flag, I tried to see if the user ‘michael’ has root access, but he did not.

**Command used: Sudo -l**

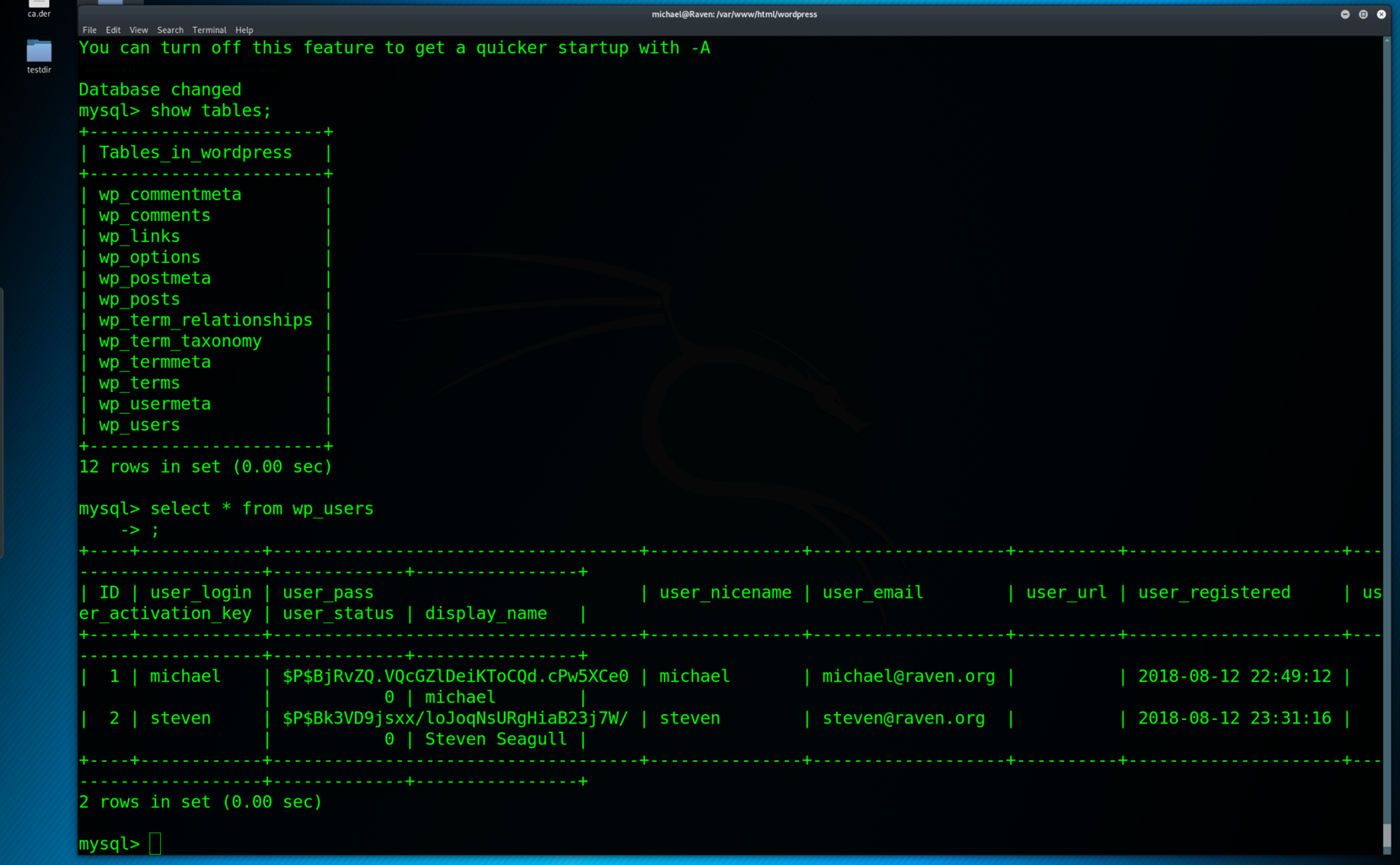


1. Because we know that Wordpress was installed in the application, we can check to see if there are any database credentials in the wp-config.php. Upon opening up the file we can find the MySQL database password which is: ‘R@v3nSecurity’.

Command Used: **mysql -u root –p**

Password: **R@v3nSecurity**

From here we can show what database are available and try and view some kind of user database.

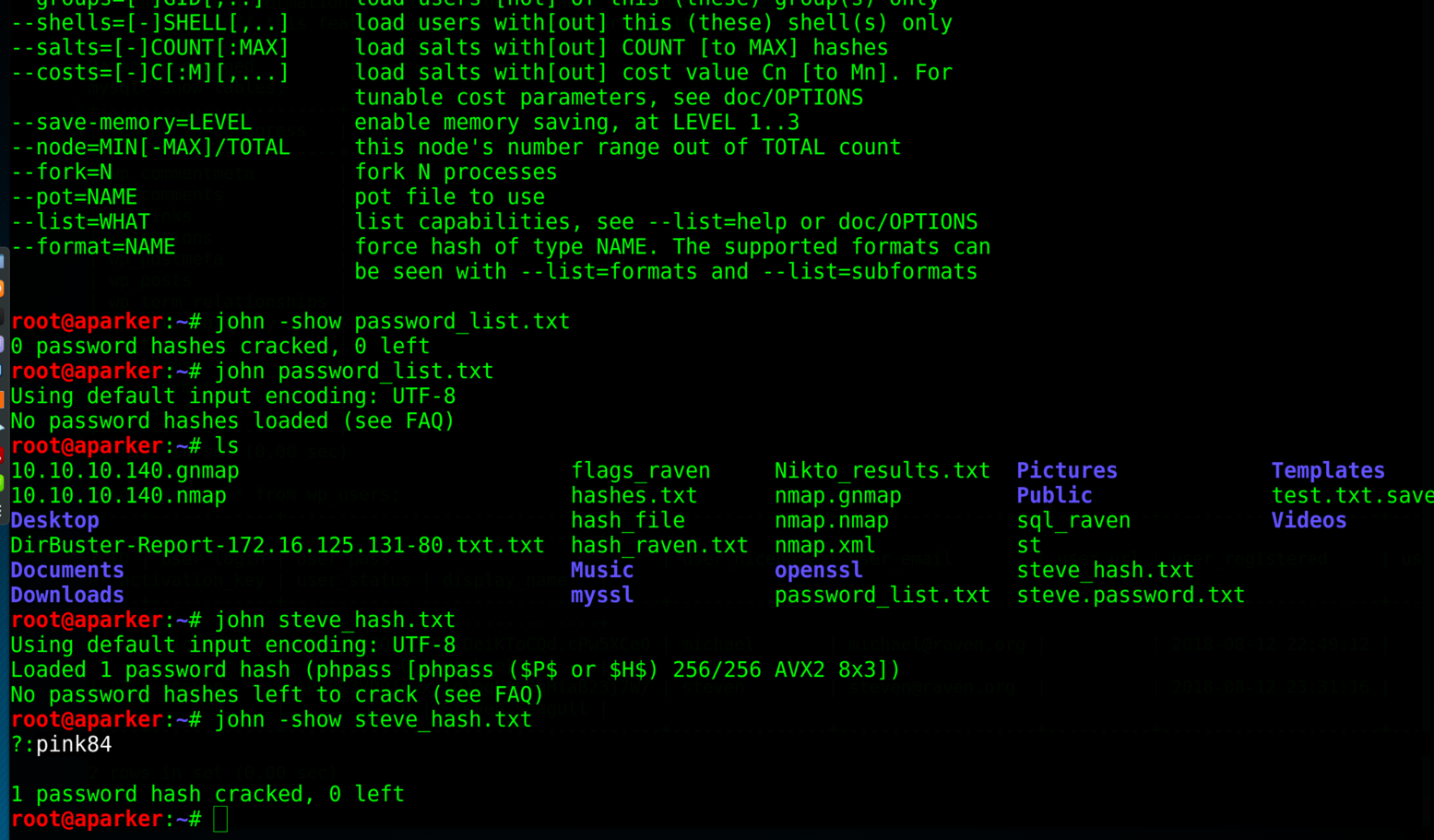


1. As you can see in the above picture, we were able to go through the database and find the two users ‘michael’ and ‘steven’ with both password hashes shown. Once I had the hash for the user ‘steven’, I copy and pasted that hash in a file so I can use John the Ripper to crack it.

**Commands used: show databases;**

**use wp\_users;**

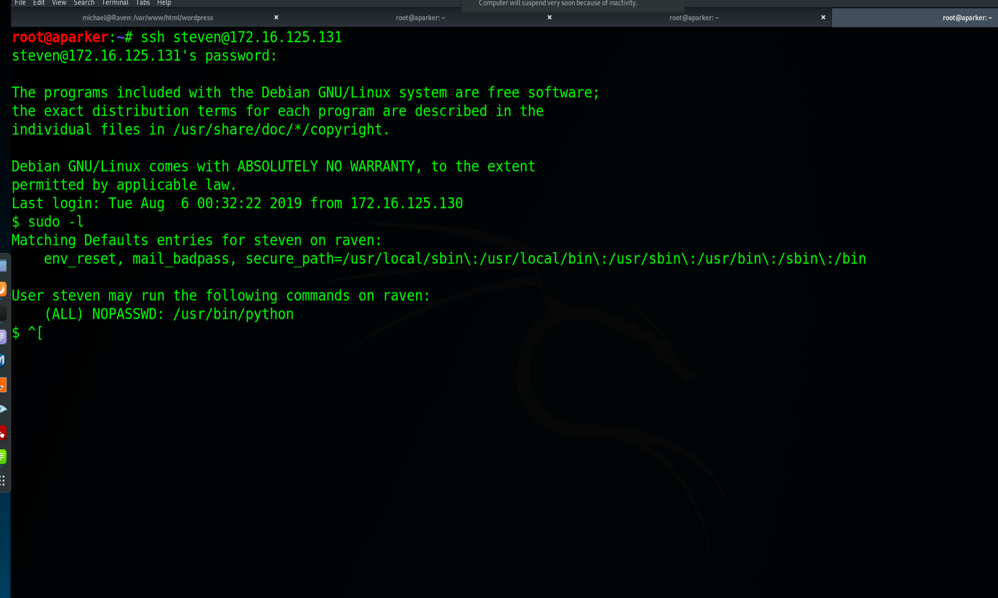
**select \* from wp\_users;**



1. After running the hash through John the Ripper, we were able to get a password which is ‘pink84’. Now that we have this password, we can ssh into the target machine as Steve and see if we can escalate privileges to root the machine,

**Command used: john steve\_hash.txt**

**john -show steve\_hash.txt**



1. We can see that our cracked password for Steven was successful. I then ran a sudo -l command to see what we can use with Sudo. I found that we can use Python with sudo, which can be seen in the above image. As sudo is used to execute commands with root user, we can run the sudo python command to take root access of the machine. This can also be seen in the image above that we now have root access.

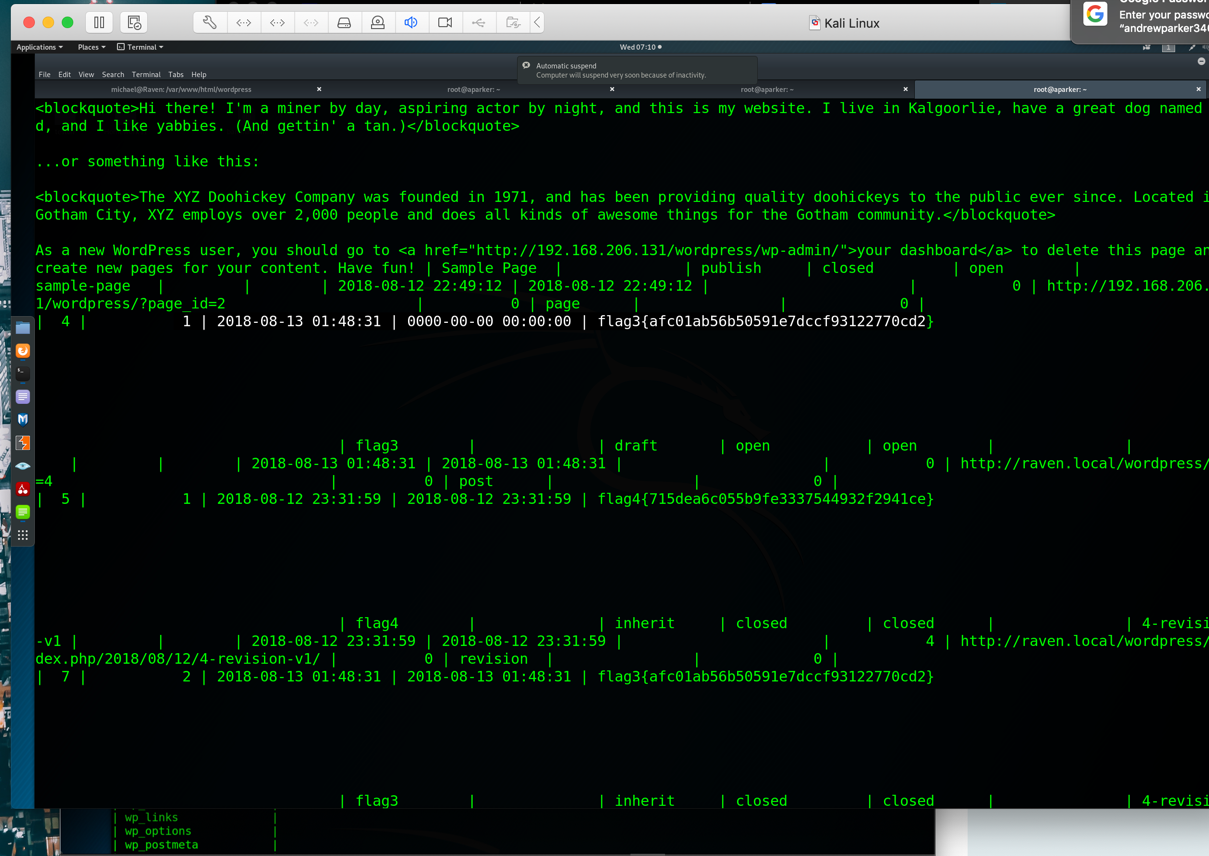
**Command Used: sudo python -c ‘import pty;pty.spawn(“/bin/bash”);’**



1. Now that we have root access, we can move to the /root/ folder to try and find the last two flags. As we can see from the image above, flag4 was in the root directory. There is still one flag remaining that we still need to find.

**Command used: cd /root/**

**Cat flag4.txt**



1. Now that we have one flag left and are have root access, we can start digging through directories in order to find the last flag. After having no luck, I decided to go back to the mysql database to see if there are any more flags present. I went through the table ‘wp\_posts’ and was able to find the final flag (flag3.txt).

**Command used: mysql -u root -p**

**Show databases;**

**Use wordpresss;**

**Select \* from wp\_posts;**

We now have all 4 flags and officially rooted Raven:

**Flag 1: b0bbcb33e11b80be759c4e844862482d**

**Flag 2: fc3fd58dcdad9ab23faca6e9a36e581c**

**Flag 3: afc01ab56b50591e7dccf93122770cd2**

**Flag 4: 715dea6c055b9fe3337544932f2941ce**