Final Engagement

Attack, Defense & Analysis of a Vulnerable Network

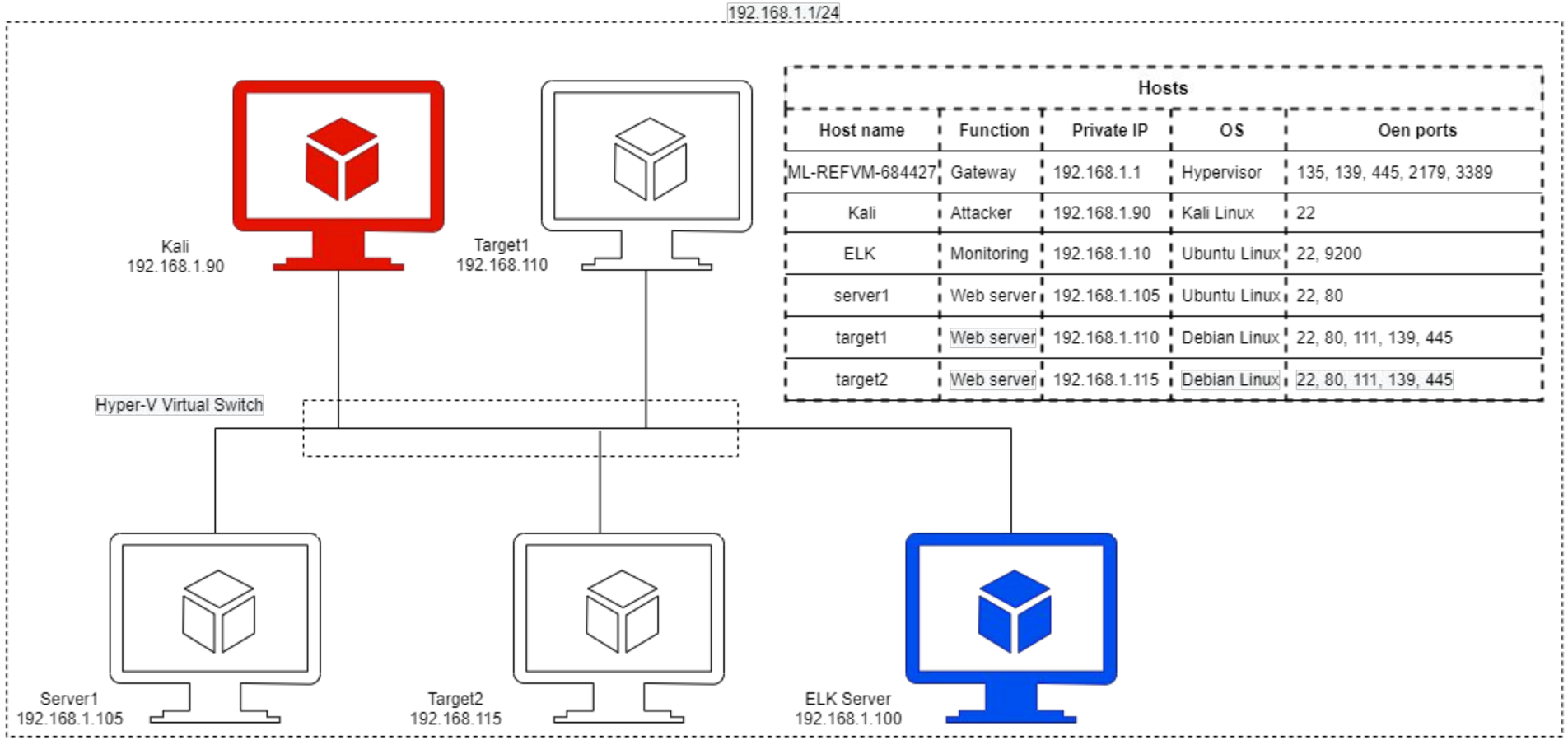
Team members:
Ben McCall, Billy Binedell, Farzan Akbaridoust, Kai Ye, Luke Gonzaga, Scottie Fuhrmann

Table of Contents

This document contains the following resources:

Network Topology
& Critical Vulnerabilities

Network Topology



Hosts

Host name	Function	Private IP	OS	Oen ports
ML-REFVM-684427	Gateway	192.168.1.1	Hypervisor	135, 139, 445, 2179, 3389
Kali	Attacker	192.168.1.90	Kali Linux	22
ELK	Monitoring	192.168.1.10	Ubuntu Linux	22, 9200
server1	Web server	192.168.1.105	Ubuntu Linux	22, 80
target1	Web server	192.168.1.110	Debian Linux	22, 80, 111, 139, 445
target2	Web server	192.168.1.115	Debian Linux	22, 80, 111, 139, 445

Network
Address Range: 192.168.1.1/24
Netmask: 255.255.255.0
Gateway: 192.168.1.1

Machines
IPv4: 192.168.1.90
OS: Kali Linux 5.4.0
Hostname: Kali

IPv4: 192.168.1.110
OS: Debian 3.16.0-6
Hostname: Target 1

IPv4: 192.168.1.100
OS: Ubuntu 18.04 LTS
Hostname: Elk

IPv4: 192.168.1.115
OS: Debian 3.16.0-6
Hostname: Target 2

IPv4: 192.168.1.105
OS: Ubuntu 18.04 LTS
Hostname: Capstone

Critical Vulnerabilities: Target 1

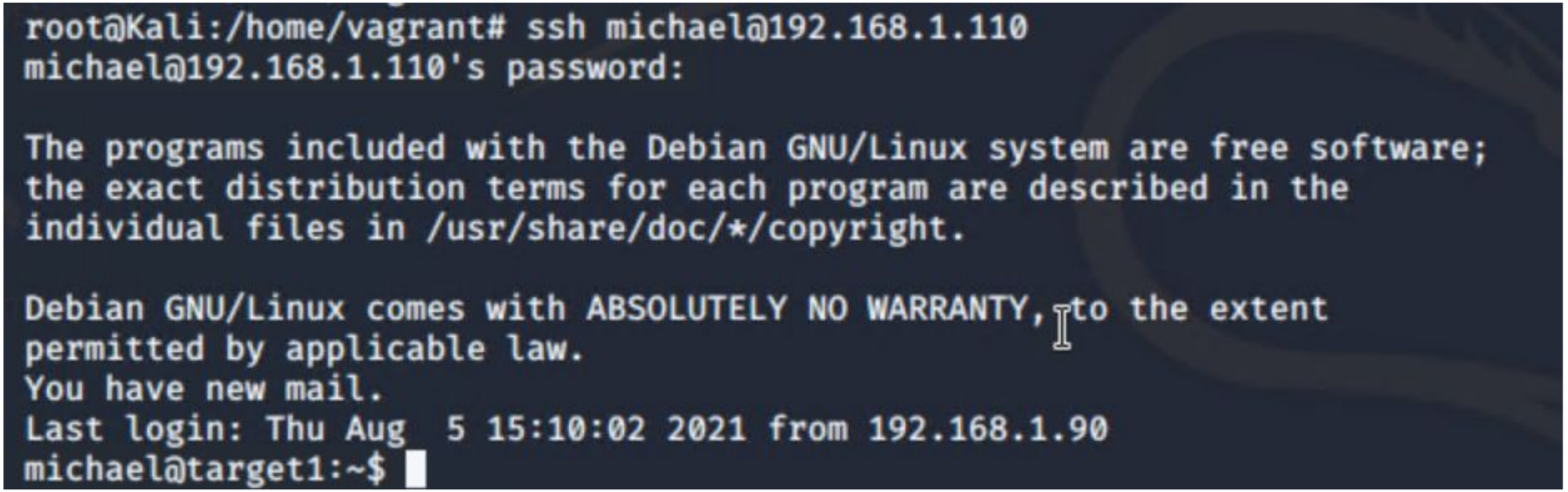
Our assessment uncovered the following critical vulnerabilities in **Target 1**.

Vulnerability	Description	Impact
Weak Passwords & SSH	The first user ‘michael’ had set up his password as ‘michael’. Therefore we could ssh into michael@192.168.1.110 and view the contents of /var/www/html	The malicious actor can easily guess the first users password without need of brute forcing.
Admin login credentials stored on public server file	The wp-config.php located in /var/www/html contains the admin credentials needed to login to the wordpress MySQL database.	The actor now has access to the credentials to access the wordpress database. This php file should not have been accessible from michael’s account.
Accessible hash files	The hashed passwords of both users ‘michael’ and ‘steven’ were found in the database under wp_users	These hashes can be cracked by the actor through use of john the ripper, the actor now has access to user steven’s account.
Root escalation	When logged into user steven’s account via “ssh steven@192.168.1.110” privilege escalation to the root user was made possible through “sudo python -c ‘import pty;pty.spawn(“/bin/bash)’ ”	The malicious actor now logged in as the second user, using this method to escalate to root privileges and then able to exfiltrate the desired information.

Exploits Used

Exploitation: Weak Password (SSH)

- No strong password policy has been enforced for Raven Security, as seen by Michael’s password being “michael” could be guessed or brute forced.



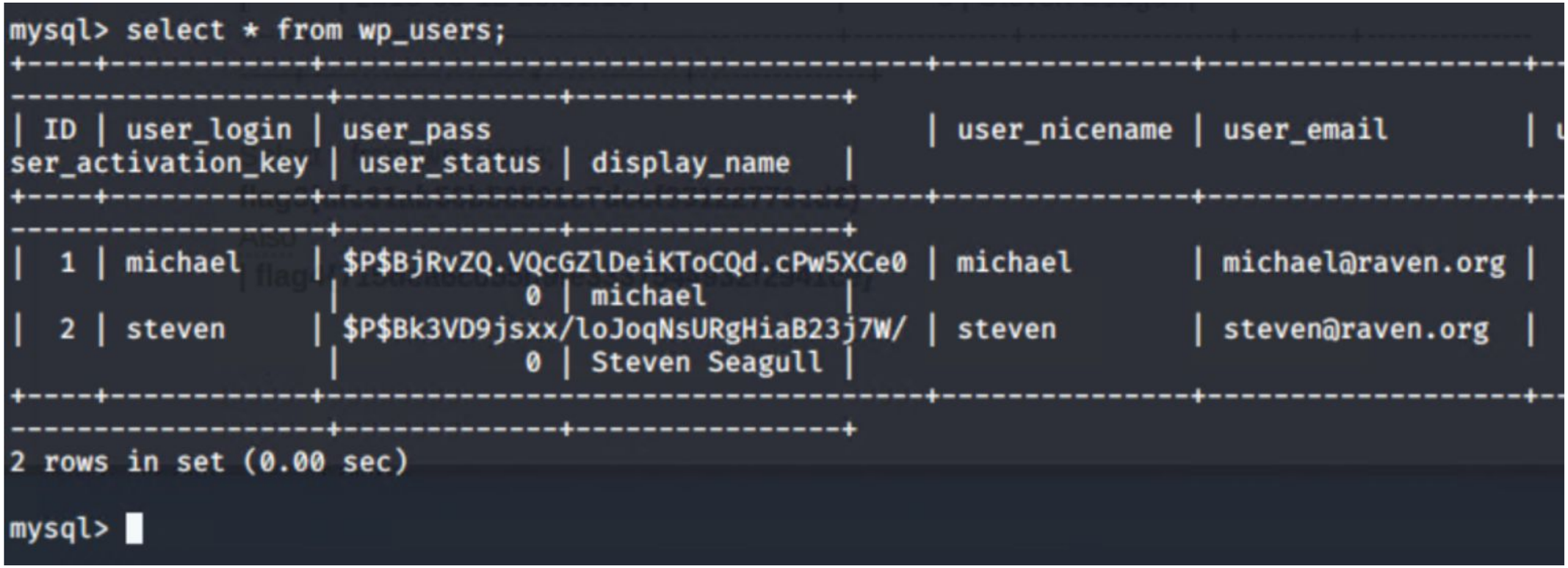
Exploitation: Access to Application Configuration Files

- User michael privileges are sufficient to access /var/www/html/wordpress/wp-config.php
- To get the wordpress database - root username, password and database name

```
// ** 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');
```

Exploitation: Access to Database Containing P/W Hashes

- Reveal user names and password hashes inside the wordpress database
- mysql> select * from wp_users



- John the ripper promptly solved the password for user steven because the hashes were not salted

Exploitation: Root Escalation

- The Root Escalation exploit occurred by using a Python script to gain root access in Steven’s account.
- The command used was sudo python -c ‘import pty;pty.spawn(“/bin/bash”);’, which uses the pty library’s spawn method to create a pseudo-shell with sudo privilege.

```
$ id
uid=1001(steven) gid=1001(steven) groups=1001(steven)
$ sudo python -c 'import pty;pty.spawn("/bin/bash")'
root@target1:/var/www/html/wordpress# id
uid=0(root) gid=0(root) groups=0(root)
root@target1:/var/www/html/wordpress#
```

Avoiding Detection

Stealth Exploitation Monitoring Overview

Elasticsearch Watcher threshold alerts created for:

- HTTP Errors (Response) using Packetbeat indices

  WHEN count() GROUPED OVER top 5 ‘http.response.status_code’ IS ABOVE 400 FOR THE LAST 5 minutes

- CPU Usage using Packetbeat indices

  WHEN max() of system.process.cpu.total.pct OVER all documents IS ABOVE 0.5 FOR THE LAST 5 minutes

- HTTP Request Size using Metricbeat indices

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

- Directory Access using Packetbeat indices

  WHEN count() GROUPED OVER top 5 ‘url.path’ IS ABOVE 10 FOR THE LAST 30 seconds

Monitoring Issues

- Excessive false positive alerts
- Limitations of threshold alerts versus advanced watches (using JSON)

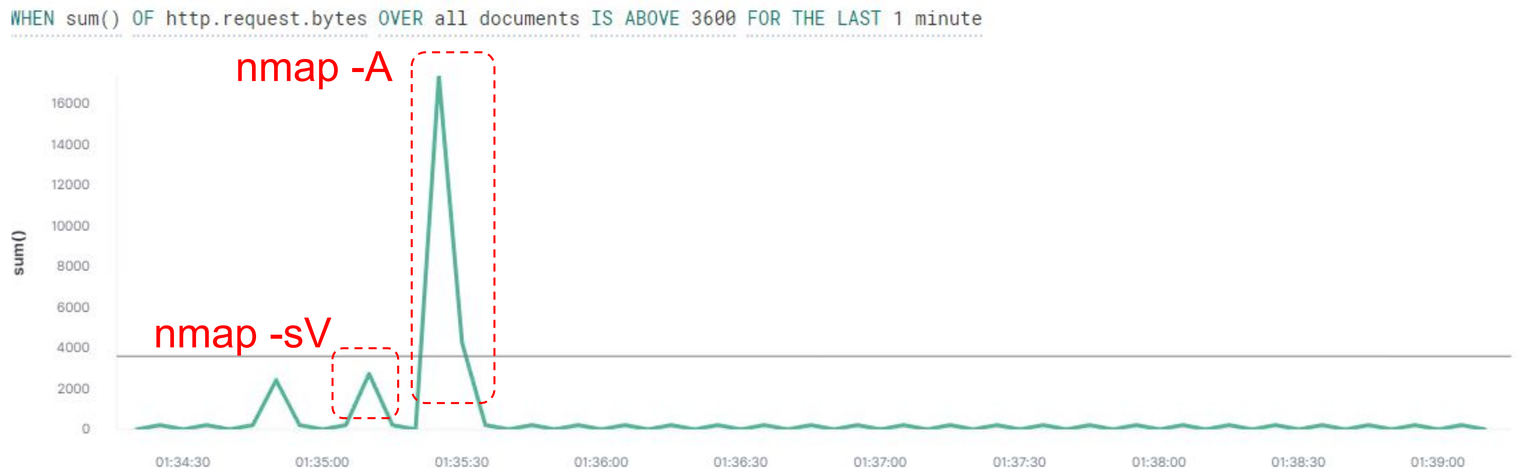
Stealth Exploitation: Network Mapper (Nmap)

Alerts Triggered

- **nmap -A** was detected by the **HTTP Request Size** alert

Alert Bypass Suggestion

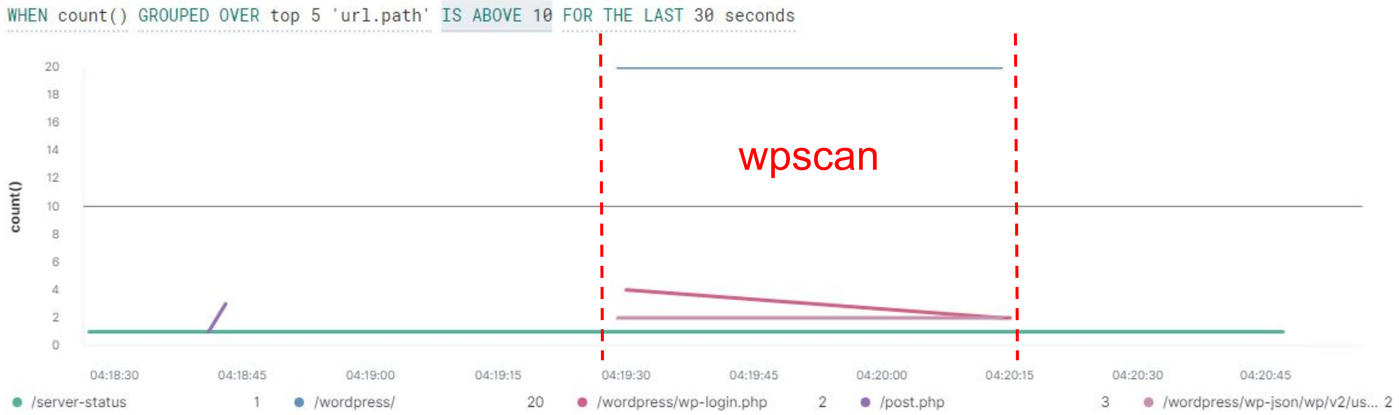
- **nmap -sV** did not trigger the **HTTP Request Size** alert



Stealth Exploitation: WordPress Security Scanner (wpscan)

Alerts Triggered

- **wpscan** was detected by the **Directory Access** alert



Alert Bypass Suggestion

- **wpscan --stealthy** uses random user agents and passive detection, however this option does not enumerate the users
- Create a noisy environment before, during and after the **wpscan** execution

Stealth Exploitation of Root Escalation

Alerts Triggered

- No Watcher threshold alerts were triggered by this activity

Alert Bypass Suggestion

- Python script that was used to gain escalated privileges was not detected
- The approaches to remain less detectable are:
  - Exploit silently via a reverse shell (Meterpreter or Netcat)
  - If Python exploit is used, ensure that required tasks are performed quickly and then log out