

# **Final Engagement**

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

# Table of Contents

---

This document contains the following resources:



**Network Topology & Critical Vulnerabilities**



**Alerts Implemented**



**Hardening**



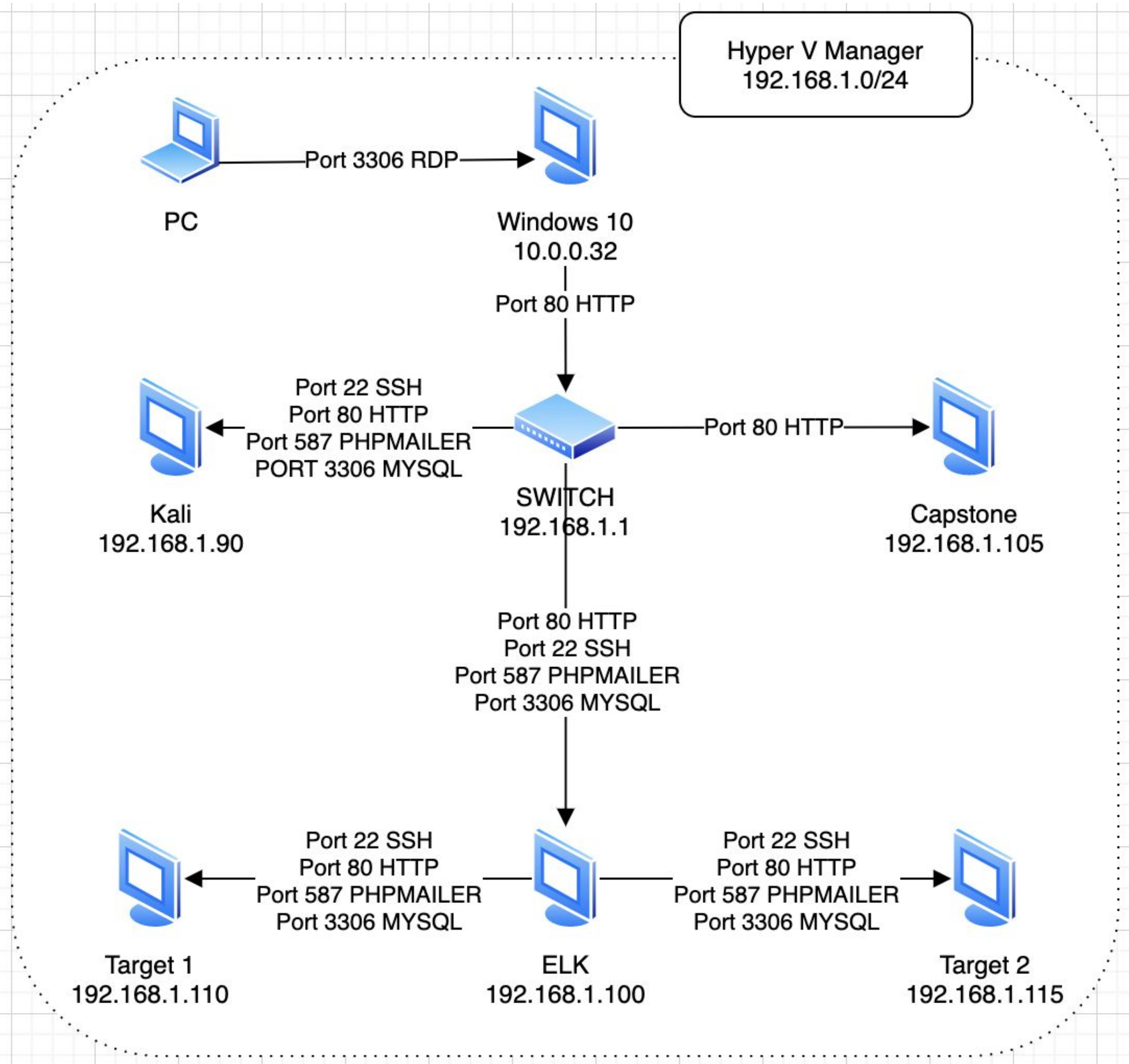
**Implementing Patches**

# 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: TARGET1

IPv4: 192.168.1.115  
OS: Linux  
Hostname: TARGET2

# Critical Vulnerabilities: Target 1

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

Vulnerability	Description	Impact
WordPress Site Enumeration	Use of WordPress Security Scanner (wpscan) against server (192.168.1.110) reveals sensitive information	The WordPress Security Scanner reveals the usernames: “michael” and “steven”
Brute Force Attack (michael)	By using Metasploit’s ssh_login module, user michael’s password can be cracked on the target	Malicious attackers can ssh into the TARGET1 server as user michael
wp-config.php readability	wp-config.php reveals sensitive information about root password for MySQL database	Vulnerable user michael has access to viewing wp-config.php which reveals root password for MySQL
John the Ripper (steven)	Using John on a hashed password for Steven found in MySQL	User steven’s hashed password cracked and obtained
Privilege escalation with Python	Exploit steven’s sudo permissions on python to gain root access of system: sudo python -c ‘import pty;pty.spawn(“/bin/bash”)’	Steven’s sudo permissions with Python can be leveraged to gain root access of the system

# Critical Vulnerabilities: Target 2

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

Vulnerability	Description	Impact
Wordpress Scan	<ul style="list-style-type: none"><li>● WPScan Tool to scan a server's Wordpress database for any vulnerabilities.</li><li>● Can also be done using an nmap script.</li></ul>	<ul style="list-style-type: none"><li>● Hidden folders such as "manual" and "vendor" were revealed in the scan.</li><li>● Accessing "vendor" folder and the VERSION page leads to a single flag.</li></ul>
PHPMailer 5.2.16	<ul style="list-style-type: none"><li>● Exploitdb search revealed an RCE vulnerability for PHPMailer versions below 5.2.18</li></ul>	<ul style="list-style-type: none"><li>● Uploading a unique PHP script into the server and calling on it will send a shell to the attacker machine.</li></ul>
MySQL Root Access	<ul style="list-style-type: none"><li>● Access to server's MySQL database as 'root' allows the attacker to administer the database.</li></ul>	<ul style="list-style-type: none"><li>● Attacker can create tables and functions that can allow the attacker to take control from MySQL.</li></ul>



# Alerts Implemented



# Excessive HTTP Errors

- Description: Suspicious number of HTTP 400+ error responses
- Purpose: Identifies brute force attacks and other similar attempts which require a large number of HTTP requests
- Threshold: Fires when the top five HTTP response status codes are above 400 for the last 5 minutes

Current status for 'Excessive HTTP Errors'

Execution history

Action statuses

Last one hour

Trigger time	State
2020-07-22T04:41:20+00:00	✓ OK
2020-07-22T04:40:20+00:00	✓ OK
2020-07-22T04:39:20+00:00	✓ OK
2020-07-22T04:38:20+00:00	✓ OK
2020-07-22T04:37:20+00:00	✓ OK
2020-07-22T04:36:20+00:00	✓ OK
2020-07-22T04:35:20+00:00	✓ OK
2020-07-22T04:34:20+00:00	✓ OK
2020-07-22T04:33:20+00:00	✓ OK
2020-07-22T04:32:20+00:00	✓ OK

Rows per page: 10

Executed on Wed Jul 22 2020 04:41:20 GMT+0000

Actions

Name	State
logging_1	✓ OK

JSON

```
{  "watch_id": "40606738-8f72-4d0d-8272-bc3a1b3378b4",  "node": "FNfCktQkTMGDGHxIwpIOug",  "state": "execution_not_needed",  "status": {    "state": {      "active": true,      "timestamp": "2020-07-16T00:34:19.341Z"    },    "last_checked": "2020-07-22T04:41:20.355Z",    "last_met_condition": "2020-07-18T15:30:08.188Z",    "actions": {      "logging_1": {        "ack": {          "timestamp": "2020-07-18T15:31:08.225Z",          "state": "awaits_successful_execution"        },        "last_execution": {          "timestamp": "2020-07-18T15:30:08.188Z",          "successful": true        },        "last_successful_execution": {          "timestamp": "2020-07-18T15:30:08.188Z",          "successful": true        }      }    }  }
```

Name

Excessive HTTP Errors

Indices to query

packetbeat-\*

Time field

@timestamp

Run watch every

5

minutes

Match the following condition

WHEN count() GROUPED OVER top 5 'http.response.status\_code' IS ABOVE 400 FOR THE LAST 5 minutes



# HTTP Request Size Monitor

- Description: An HTTP request contains a larger-than-average amount of data
- Purpose: Identifies malicious file transfers
- Threshold: Fires when the sum of bytes in incoming HTTP requests is above 3,500 for the last minute

Current status for 'HTTP Size Request Monitor'

Execution history

Action statuses

Last one hour

Trigger time	State
2020-07-22T04:41:20+00:00	✓ OK
2020-07-22T04:40:20+00:00	✓ OK
2020-07-22T04:39:20+00:00	✓ OK
2020-07-22T04:38:20+00:00	✓ OK
2020-07-22T04:37:20+00:00	✓ OK
2020-07-22T04:36:20+00:00	✓ OK
2020-07-22T04:35:20+00:00	✓ OK
2020-07-22T04:34:20+00:00	✓ OK
2020-07-22T04:33:20+00:00	✓ OK
2020-07-22T04:32:20+00:00	✓ OK

Rows per page: 10

Executed on Wed Jul 22 2020 04:41:20 GMT+0000

Actions

Name	State
logging_1	✓ OK

JSON

```
{
  "watch_id": "ce5b078a-d2cf-4577-89f1-1d65683f1a30",
  "node": "FNfCktQkTMGDGHxIwpIOug",
  "state": "execution_not_needed",
  "status": {
    "state": {
      "active": true,
      "timestamp": "2020-07-15T23:26:21.043Z"
    },
    "last_checked": "2020-07-22T04:41:20.355Z",
    "actions": {
      "logging_1": {
        "ack": {
          "timestamp": "2020-07-15T23:26:21.043Z",
          "state": "awaits_successful_execution"
        }
      }
    },
    "execution_state": "execution_not_needed",
    "version": -1
  },
  "trigger_event": {
    "type": "schedule",
    "triggered_time": "2020-07-22T04:41:20.355Z",
    "schedule": {
      "scheduled_time": "2020-07-22T04:41:20.347Z"
    }
  }
}
```

Edit HTTP Request Size Monitor

Send an alert when your specified condition is met. Your watch will run every 1 minute.

Name

HTTP Request Size Monitor

Indices to query

metricbeat-\*

Time field

@timestamp

Run watch every

1

minute

Use \* to broaden your query.

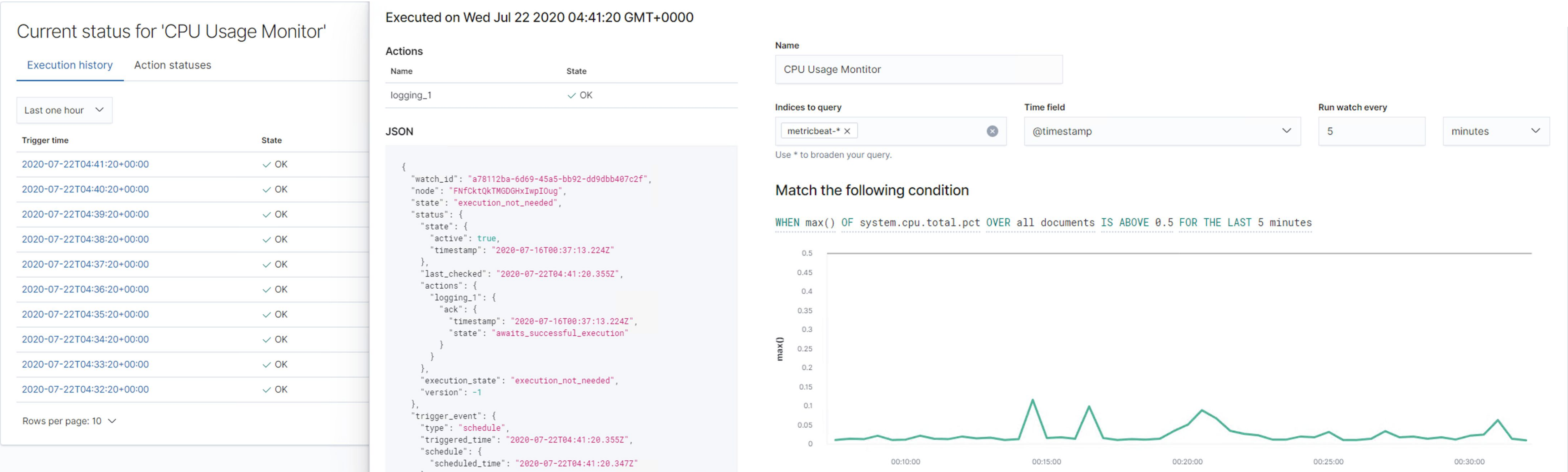
Match the following condition

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

sum()

# CPU Usage Monitor

- Description: Dangerously high CPU usage
- Purpose: Identifies a machine about to go offline or when the CPU is being used for nefarious purposes
- Threshold: Fires when system processes exceed 0.5 (50%) CPU total percentage for the last 5 minutes





# Hardening





# Hardening Against WPSCAN on Target 1

---

- An excellent way to mitigate potential exploits is to constantly update one's software
- The latest version of WordPress (5.4.2) claims to patch critical security flaws and will prevent WPscan from finding vulnerabilities
- Alternatively, by installing Varnish v3 or v4 will prevent user enumeration with minor edits to its sub vcl\_recv section
- To install or update WordPress:  
apt-get update  
apt-get install wordpress
- To install Varnish:  
apt-get update  
apt-get install varnish

# Hardening Against Bruteforce on Target 1

---

- By being able to bruteforce Michael's password, it allowed us access to the remainder of the system
- Thus by simply creating a more complex password, it would not only increase the time to bruteforce but could possibly give system administrators enough time to respond and react accordingly
- An example of a more complex password would be 10 characters in length, and contain random characters
- Alternatively limiting failed login attempts or implementing two factor authentication would also work

# Hardening Against wp-config.php on Target 1

---

- By being able to access and view wp-config.php from Michael's account, we were able to determine Steven's wordpress hash
- To prevent this, one could limit Michael's access to sensitive data or limit file permissions to sensitive data, such as hashes, to only administrator level accounts



# Hardening Against John the Ripper on Target 1

---

- As aforementioned, creating a more complex password in terms of length and the characters it contains would mitigate John the Ripper attacks by drastically increasing the time it would take to crack

# Hardening Against Python Exploit on Target 1

---

- The sole reason we were able to achieve root access is due to Steven's root permissions for running Python
- Disabling Steven's root access to Python would have prevented us from being able to gain root access

# Hardening Against Enumeration on Target 2

---

- Using nmap we were able to view directories on the target and were able to access sensitive data
- Firewall rules or an IDS would have prevented a successful nmap scan and would have kept the directories protected against enumeration



# Hardening Against PHPMailer on Target 2

---

- The CVE-2016-10033 vulnerability allowed us to execute arbitrary code due to the target running an outdated version of PHPMailer (5.2.16)
- To mitigate this, updating PHPMailer to its current version would remove the possibility of exploit as recent versions of PHPMailer (6.1.6) do not allow arbitrary code execution
- Alternatively, products such as Fortinet's IPS would have prevented this exploit as the malicious actor could not have used a vulnerability exploit
- To install or update PHPMailer:  
apt-get update  
apt-get install libphp-phpmailer

# Hardening Against MySql UDF Exploit on Target 2

---

- By exploiting an outdated MySQL program (5.5.60) we were able to use UDF to escalate ourselves to root and find the final flag
- Current versions of MySQL (8.0) prevent the ability to exploit the UDF library and escalate ourselves to root
- To install or update MySQL:

apt-get install update

apt-get install -y wget

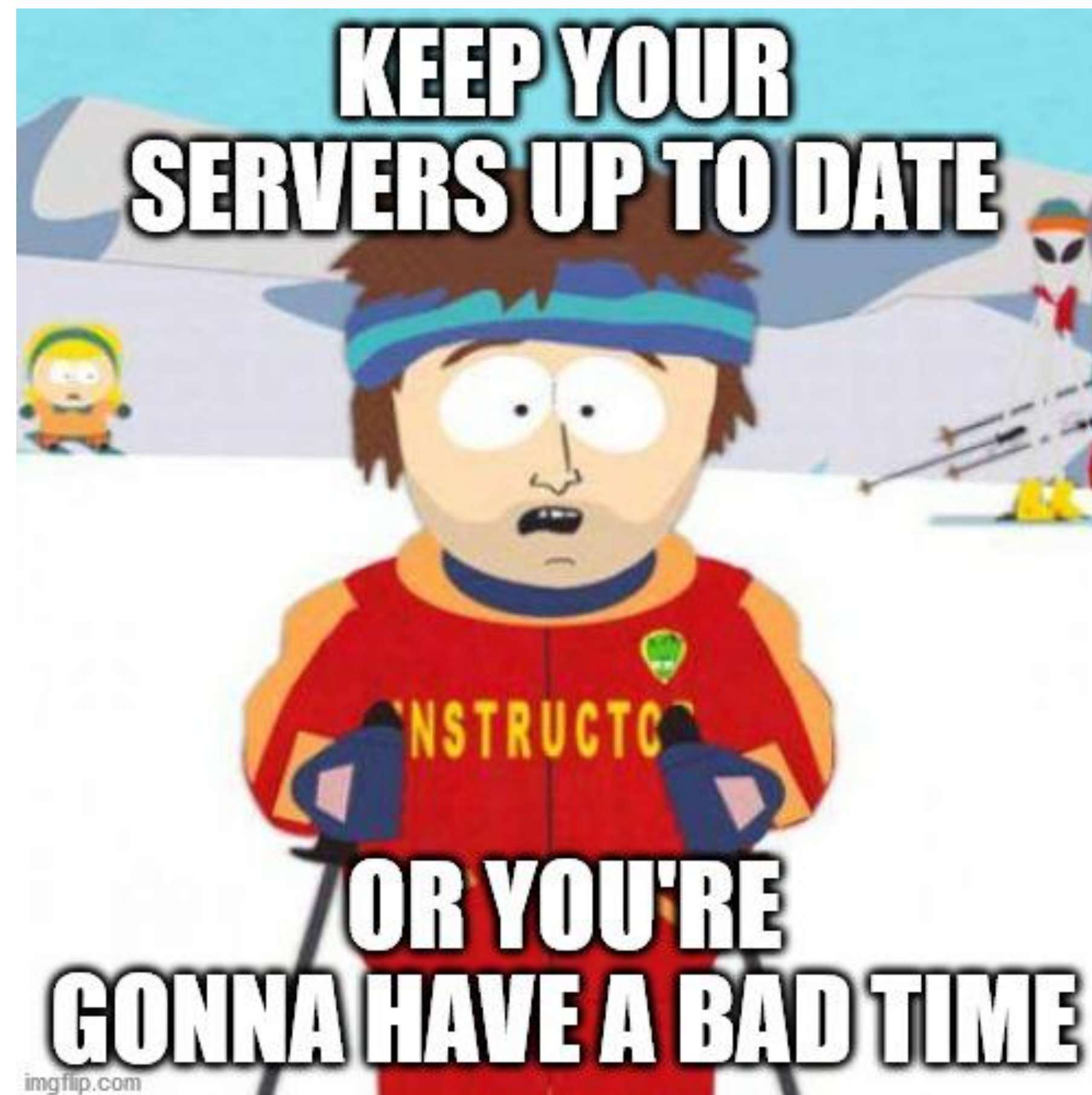
wget [https://dev.mysql.com/get/mysql-apt-config\\_0.8.15-1\\_all.deb](https://dev.mysql.com/get/mysql-apt-config_0.8.15-1_all.deb)

dpkg -i mysql-apt-config\_0.15-1\_all.deb

apt-get install mysql-community-server

systemctl enable --now mysql

**If there's anything to take away from this blue team presentation, it's to UPDATE YOUR SOFTWARE TO PREVENT EXPLOITS**





[Start of Network Analysis]