

# Blue Team: Summary of Operations

## Table of Contents

- Network Topology
- Description of Targets
- Monitoring the Targets
- Patterns of Traffic & Behavior
- Suggestions for Going Further

## Network Topology

The following machines were identified on the network:

- **Network**

1. Address Range: 192.168.1.0/24
2. Netmask: 255.255.255.0
3. Gateway: 192.168.1.1
4. Cloud Provider: Azure

- **Machines**

- IPv4: 192.168.1.90
  - OS: Kali Linux 5.4.0
  - Hostname: Kali
  - Purpose: Penetration Testing
- IPv4: 192.168.1.110
  - OS: Linux
  - Hostname: Target 1
  - Purpose: Target Machine with WordPress Vulnerabilities
- IPv4: 192.168.1.100
  - OS: Linux
  - Hostname: Elk

- Purpose: Metricbeat, Filebeat, Packetbeat, Watcher Log Collection
- IPv4: 192.168.1.105
  - OS: Linux Ubuntu
  - Hostname: Capstone
  - Purpose: Kibana

## Description of Targets

- The two VMs that are on the network were vulnerable to attacks. Target 1 (192.168.1.110)
- Target 1 is an Apache web server and has SSH enabled, so ports 80 and 22 are ports of entry for attackers. As such, the following alerts have been implemented:

## Monitoring the Targets

This scan would be able to identify which ports of entry for attacks:

- Port 22/TCP Open SSH OpenSSH 6.7p1 Debian 5+deb8u4
- Port 80/TCP Open HTTP Apache HTTPd 2.4.10 (Debian)

## Excessive HTTP Errors

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

- **Metric:** This monitoring rule watches the http.response.status\_code from metricbeat
- **Threshold:** Is above 400
- **Vulnerability Mitigated:** Enumeration/Brute Force
- **Syntax:** WHEN count() GROUPED OVER top 5 'http.response.status\_code' IS ABOVE 400 FOR THE LAST 5 minutes
- **Reliability:** The alert is extremely dependable. This type of alert measures error codes 400 and above. This alert also filters out any regular/effective responses.

The 400+ codes are client and server errors which have more reason to have that concern. It also raises alarms when the error codes go off at a high rate.

Current status for 'Excessive HTTP Errors'

[Deactivate](#)

[Delete](#)

[Execution history](#)

[Action statuses](#)

Last one hour ▾

Trigger time	State	Comment
2022-05-04T02:50:13+00:00	✓ OK	
2022-05-04T02:49:13+00:00	✓ OK	
2022-05-04T02:48:13+00:00	✓ OK	
2022-05-04T02:47:13+00:00	✓ OK	
2022-05-04T02:46:13+00:00	✓ OK	
2022-05-04T02:45:13+00:00	✓ OK	
2022-05-04T02:44:13+00:00	✓ OK	
2022-05-04T02:43:13+00:00	✓ OK	
2022-05-04T02:42:13+00:00	✓ OK	
2022-05-04T02:41:13+00:00	✓ OK	

Rows per page: 10 ▾

< 1 2 3 4 5 ... 43 >

## • HTTP Request Size Monitor

HTTP Request Size Monitor is implemented as follows:

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

- **Metric:** This monitoring rule watches the http.request.bytes from metricbeat
- **Threshold:** Is above 3500
- **Vulnerability Mitigated:** Code injection in HTTP requests (XSS and CRLF) or DDOS
- **Syntax:** WHEN sum() OF http.request.bytes OVER all documents IS ABOVE 3500 FOR THE LAST 1 minute
- **Reliability:** This is a helpful and extremely dependable alert as it can point to suspicious activity and identify the beginning of a DoS attack. It's a very uncomplicated way of identifying large data transactions in a short duration without the risk of excessive false positives

## Current status for 'HTTP Request Size Monitor '

[Deactivate](#)[Delete](#)[Execution history](#)[Action statuses](#)

Last one hour ▾

Trigger time	State	Comment
2022-05-04T02:50:13+00:00	✓ OK	
2022-05-04T02:49:13+00:00	✓ OK	
2022-05-04T02:48:13+00:00	✓ OK	
2022-05-04T02:47:13+00:00	✓ OK	
2022-05-04T02:46:13+00:00	✓ OK	
2022-05-04T02:45:13+00:00	✓ OK	
2022-05-04T02:44:13+00:00	✓ OK	
2022-05-04T02:43:13+00:00	✓ OK	
2022-05-04T02:42:13+00:00	✓ OK	
2022-05-04T02:41:13+00:00	✓ OK	

Rows per page: 10 ▾

&lt; 1 2 3 4 5 ... 43 &gt;

## CPU Usage Monitor

CPU Usage Monitor is implemented as follows:

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

- **Metric:** This monitoring rule watches the system.process.cpu.total.pct from metricbeat
- **Threshold:** Is above 0.5
- **Syntax:** WHEN max() OF system.process.cpu.total.pct OVER all documents IS ABOVE 0.5 FOR THE LAST 5 minutes
- **Vulnerability Mitigated:** Malicious software, Denial of Service attacks, programs (malware or viruses) running taking up resources
- **Reliability:** This alert is low reliability because there are several things that could happen with a machine that would cause high CPU usage and be unrelated to a DDoS attack.


## Current status for 'CPU Usage Monitor'

[Deactivate](#) [Delete](#)

[Execution history](#) [Action statuses](#)

Last one hour 

Trigger time	State	Comment
2022-05-04T02:51:13+00:00	✓ OK	
2022-05-04T02:50:13+00:00	✓ OK	
2022-05-04T02:49:13+00:00	✓ OK	
2022-05-04T02:48:13+00:00	✓ OK	
2022-05-04T02:47:13+00:00	✓ OK	
2022-05-04T02:46:13+00:00	✓ OK	
2022-05-04T02:45:13+00:00	✓ OK	
2022-05-04T02:44:13+00:00	✓ OK	
2022-05-04T02:43:13+00:00	✓ OK	
2022-05-04T02:42:13+00:00	✓ OK	

Rows per page: 10 

< 1 2 3 4 5 ... 43 >

## Suggestions for Going Further (Optional)

The logs and alerts generated during the assessment suggest that this network is susceptible to several active threats, identified by the alerts above. In addition to watching for occurrences of such threats, the network should be hardened against them. The Blue Team suggests that IT implement the fixes below to protect the network:

- Vulnerability 1-Hardening Against Vulnerable Ports 22 and 80 on Target 1
  - **Patch:** Close port 22 and use port 443 with HTTPS instead of 80.
  - **Why It Works:** Port 22 will prevent open SSH access to the machine. Using port 443 will provide a layer of security using SSL instead of the open port.
    - Port 80 and 22 can be shut down with:
      - `sudo ufw deny PORT 80`
      - `sudo ufw deny PORT 22`
      - `sudo ufw allow PORT 443`
      - Each command should be run one at a time and checked status with `sudo ufw status verbose`
- Vulnerability 2- Hardening Against Weak/Insecure Passwords on Target 1
  - **Patch:** Users should change passwords to a best-practices format involving at least 16 characters, special characters, numbers, and symbols.

1-hour lockouts should be implemented after 5 unsuccessful attempts within 15 minutes. Multi-factor authentication should also be used.

- **Why It Works:** Complex passwords are difficult to crack with brute force and lockouts will prevent multiple attempts. Additionally, notification alerts could be generated to further protect the accounts
  - How to install it (include commands) <https://ostechnix.com/how-to-setpassword-policies-in-linux/>
- 
- Vulnerability 3- - Hardening Against Python Privilege Escalation on Target 1
    - **Patch:** Python privileges should be removed for users vulnerable to ssh as well as users who are not authorized for root privileges.
    - **Why It Works:** Removing the python sudo privileges will eliminate the potential for circumventing access restrictions
    - vi /etc/sudoers
      - Delete this line: steven ALL=(ALL) NOPASSWD: /usr/bin/python
- 
- Vulnerability 4 - Hardening Against Enumerate WordPress Site on Target 1
    - **Patch:** Deploy the Ansible Playbook that updates the WordPress site to a patched version with the Stop User Enumeration plug-in and adjust the firewall to block similar behaviors of enumerating traffic
    - **Why It Works:** Updated versions of WordPress won't allow enumeration with appropriate plugins
      - Run the ansible-playbook discussed in the link (<https://github.com/cybergurl95/Final-Project/blob/main/WPandApache.yml>) and make sure the Stop User Enumeration plug-in is installed and enabled
      - <https://wordpress.org/plugins/stop-user-enumeration/>
- 
- Vulnerability 5 - Hardening Against Apache 2.4.1 CVE-2016-4975 on Target 1
    - **Patch:** Regularly update Apache server to the latest stable version
    - **Why It Works:** Apache tends to have substantial susceptibilities with every version of any update. To stay at the forefront of these threats, it is crucial to maintain a coherent methodology for modernizing the versions of Apache

- Run the ansible-playbook found here (<https://github.com/cybergurl95/Final-Project-/blob/main/WPandApache.yml>) which provides the automation for updating both the WordPress and Apache files to the latest and least-known vulnerabilities
- This should be run on a persistent basis