# **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**

*TODO: Fill out the information below.*

The following machines were identified on the network:

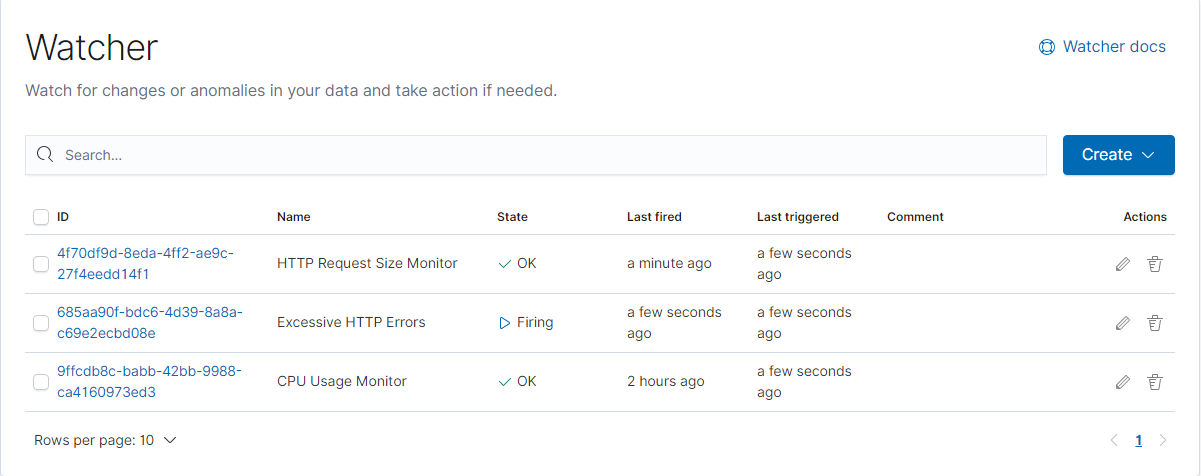
* Target 1
  + **Operating System**: Linux
  + **Purpose**: Machine being attacked.
  + **IP Address**: 192.168.1.110
* Kali
  + **Operating System**: Linux
  + **Purpose**: Attack Machine
  + **IP Address**: 192.168.1.90
* Capstone
  + **Operating System:** Linux
  + **Purpose:**
  + **IP Address:** 192.168.1.105
* ELK
  + **Operating System:** Linux
  + **Purpose:** Storing log files
  + **IP Address:** 192.168.1.100

### **Description of Targets**

*TODO: Answer the questions below.*

The target of this attack was: Target 1 192.168.1.110

Target 1 is an Apache web server and has SSH enabled, so ports 80 and 22 are possible ports of entry for attackers. As such, the following alerts have been implemented:



### **Monitoring the Targets**

Traffic to these services should be carefully monitored. To this end, we have implemented the alerts below:

#### **Excessive HTTP Errors**

*TODO: Replace Alert 1 with the name of the alert.*

Excessive HTTP Errors is implemented as follows:

* **Metric**: packetbeat-\*
* **Threshold**: WHEN count() OVER all documents IS ABOVE 1000 FOR THE LAST 5 minutes
* **Vulnerability Mitigated**: Brute force attacks.
* **Reliability**: TODO: Does this alert generate lots of false positives/false negatives? Rate as low, medium, or high reliability. Does not generate a lot of false positives/negatives because of the high threshold. High reliability.

#### **HTTP Request Size Monitor**

Alert 2 is implemented as follows:

* **Metric**: packetbeat-\*
* **Threshold**: WHEN sum() of http.request.bytes OVER all documents IS ABOVE 3500 FOR THE LAST 1 minute
* **Vulnerability Mitigated**: Prevents unauthorized uploads and mitigates reverse shell scripts from being uploaded.
* **Reliability**: TODO: Does this alert generate lots of false positives/false negatives? Rate as low, medium, or high reliability. Because the threshold is 3500, there would not be a lot of false positives/negatives. Therefore it would have a high reliability.

#### **CPU Usage Monitor**

Alert 3 is implemented as follows:

* **Metric**: metricbeat-\*
* **Threshold**: WHEN max() OF system.process.cpu.total.pct OVER all documents IS ABOVE 0.5 FOR THE LAST 5 minutes
* **Vulnerability Mitigated**: Denial of service attacks.
* **Reliability**: TODO: Does this alert generate lots of false positives/false negatives? Rate as low, medium, or high reliability. Medium reliability due to false positives if there is a lot of legitimate traffic on the webserver

*TODO Note: Explain at least 3 alerts. Add more if time allows.*

### **Suggestions for Going Further (Optional)**

*TODO*:

* Each alert above pertains to a specific vulnerability/exploit. Recall that alerts only detect malicious behavior, but do not stop it. For each vulnerability/exploit identified by the alerts above, suggest a patch. E.g., implementing a blocklist is an effective tactic against brute-force attacks. It is not necessary to explain *how* to implement each patch.

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
  + **Patch**: Creating a Password Policy
  + **Why It Works**: Creating a strict password policy could be an easy solution to brute force attacks, granted it would not 100% prevent them, but would surely decrease the possibility of them happening.
* Vulnerability 2
  + **Patch**: Creating Password Protected Directories (Installing Gnome EncFS Manager)
  + **Why It Works**: Creating password protected directories to contain particular sensitive information, where limited users are allowed access, could be a plausible workaround to forbid general users from accessing said information.
* Vulnerability 3
  + **Patch**: Restricting Python Root Access
  + **Why It Works**: Upon inspection of the sudoers file, we can see that the user Steven has special privileges to run Python as a root user. This is rather odd, as inspecting Steven’s user ID reveals that he does belong to any special user groups. Restricting Steven’s Python abilities to that of an average user closes this vulnerability.