# **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:

* KALI
  + **Operating System**: Debian Kali 5.4.0
  + **Purpose**: Attacker or Penetration Tester
  + **IP Address**:192.168.1.90
* TARGET 1
  + **Operating System**: Debian GNU/Linux 8
  + **Purpose**: WordPress Host
  + **IP Address**: 192.168.1.110
* TARGET 2
  + **Operating System**: Debian GNU/Linux 8
  + **Purpose**: WordPress Host
  + **IP Address**: 192.168.1.115
* Capstone
  + **Operating System**: Debian GNU/Linux 8
  + **Purpose**: WordPress Host
  + **IP Address**: 192.168.1.105
* ELK Stack
  + **Operating System**: Debian GNU/Linux 8
  + **Purpose**: WordPress Host
  + **IP Address**: 192.168.1.100

### 

### 

### 

### **Description of Targets**

The target of this attack was: Target 1 (IP: 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:

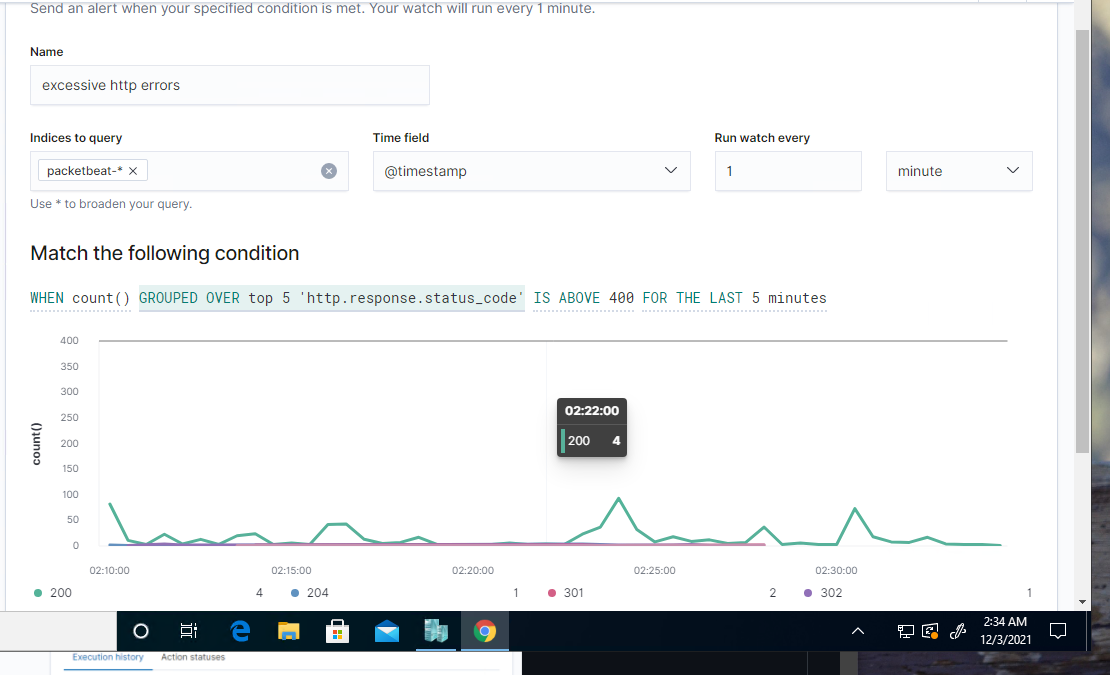
### **Monitoring the Targets**

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

#### **EXCESSIVE HTTP ERRORS**

Alert 1 is implemented as follows:

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

* **Metric**: WHEN count() GROUPED OVER top 5 'http.response.status\_code'
* **Threshold**: IS ABOVE 400 FOR THE LAST
* **Vulnerability Mitigated**: Brute Force
* **Reliability**: This alert measures any error codes 400 and above. Rate is medium reliability because of the time “5 min”. Brute forces can happen within seconds so time should be changed to make this more reliable.
* 

#### 

#### 

#### **HTTP REQUEST SIZE MONITOR**

Alert 2 is implemented as follows:

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

* **Metric**: WHEN sum() of http.request.bytes OVER all documents
* **Threshold**: IS ABOVE 3500
* **Vulnerability Mitigated**: XSS
* **Reliability**: Monitors http traffic and watches over all documents. This can be medium reliability because it may alert HTTP traffic when it is not malicious.

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### **CPU USAGE MONITOR**

Alert 3 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**: WHEN max() OF system.process.cpu.total.pct OVER all documents
* **Threshold**:IS ABOVE 0.5
* **Vulnerability Mitigated**: Malware/virus
* **Reliability**: monitors cpu usage and sees when usage is overused. This is medium reliability because it can be triggered during updates that are non malicious. Some malware or viruses may not trigger usage within the CPU and will end up under the radar.

### 

### 

### 

### 

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

* 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: Brute Force WordPress Usernames
  + **Patch**: Update wordpress and implement two-factor authentication
  + **Why It Works**: Updating wordpress will help allow for any patch work needed to be implemented. Having a two factor authentication will make it harder for hackers to gain access.
  + https://wordpress.org/support/article/brute-force-attacks/
* Vulnerability 2: HTTP Large File Request
  + **Patch**: Code Injection and DDos hardening can be implemented by having a file size limitation. Making sure these large files requests can’t happen in the first place is key.
  + **Why It Works**: When a large file request is made there can be a 404 error pop-up and show the request can’t be made.
* Vulnerability 3: CPU Usage
  + **Patch**: Updating or having a good antivirus will help to harden.
  + **Why It Works**: Antivirus software helps with detection, prevention and removal of malware.