# 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:

* Capstone
  + **Operating System**: Ubuntu 18.04.1 LTS
  + **Purpose**: Alert Testing / Attack target
  + **IP Address**: 192.168.1.105
* ELK
  + **Operating System**: Ubuntu 18.04.4 LTS
  + **Purpose**: Network monitoring
  + **IP Address**: 192.168.1.100
* Kali
  + **Operating System**: Kali 2020.1
  + **Purpose**: Attack machine
  + **IP Address**: 192.168.1.90
* Target 1
  + **Operating System**: Debian 3.16.57-2
  + **Purpose**: Wordpress server
  + **IP Address**: 192.168.1.110

### Description of Targets

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: Excessive HTTP Errors, HTTP Request Size Monitor, CPU Usage Monitor.

### 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:

* **Metric**: When count() Grouped Over top 5 http.response.status\_code
* **Threshold**: IS ABOVE 400 FOR THE LAST 5 minutes
* **Vulnerability Mitigated**: This could help detect brute force attacks.
* **Reliability**: This alert is fairly reliable since it uses HTTP error codes that a human would have a hard time producing naturally.

#### HTTP Request Size Monitor

Alert 2 is implemented as follows:

* **Metric**: WHEN sum() of http.request.bytes OVER all documents
* **Threshold**: IS ABOVE 3500 FOR THE LAST 1 minute
* **Vulnerability Mitigated**: (D)Doss, Brute Force
* **Reliability**: This alert generates false positives nearly half time it is triggered. This makes unreliable since it is impossible to tell if the traffic is irregular without looking deeper into the logs. Looking at a graph it can be seen that there are two large spikes every 10-15 minutes, but attempting to avoid those would decrease the reliability too much in the opposite direction. Instead I think it best to expect 10-12 alerts every hour since the spike is so high (around 3.4MB). The new threshold should be around 15000.

#### CPU Usage Monitor

Alert 3 is implemented as follows:

* **Metric**: WHEN max() OF system.process.cpu.total.pct OVER all documents
* **Threshold**: IS ABOVE 0.5 FOR THE LAST 5 minutes
* **Vulnerability Mitigated**: This could help indicate unauthorized access to the server if it is doing more tasks than necessary for usual web traffic.
* **Reliability**: This alert seems to be well tuned, but may result in false negatives.

### Suggestions for Going Further

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: Wordpress
  + **Patch**: Update Wordpress regularly
    - **Why It Works**: This will help keep wordpress from being targeted by known vulnerabilities.
  + **Patch**: Disable user enumeration via functions.php
  + // block WP enum scans  
    // https://m0n.co/enum  
    if (!is\_admin()) {  
     // default URL format  
     if (preg\_match('/author=([0-9]\*)/i', $\_SERVER['QUERY\_STRING'])) die();  
     add\_filter('redirect\_canonical', 'shapeSpace\_check\_enum', 10, 2);  
    }  
    function shapeSpace\_check\_enum($redirect, $request) {  
     // permalink URL format  
     if (preg\_match('/\?author=([0-9]\*)(\/\*)/i', $request)) die();  
     else return $redirect;  
    }
    - **Why It Works**: This code essentially kills all requests to search for user/author names
* Vulnerability: SSH
  + **Patch**: Use the following settings in /etc/ssh/sshd\_config
  + PasswordAuthentication no  
      
    PermitRootLogin no
    - **Why It Works**: These settings disable password authentication in SSH. Instead users will have to use SSH keys to get access to the system. This ensures that even if user passwords are compromised they cannot be used to gain access to they system through SSH. Disabling root login through SSH ensures users are restricted by permissions once they have access to the system.
* Vulnerability: Passwords
  + **Patch**:
    - apt install libpam-pwquality libpam-cracklib
    - edit /etc/login.defs
    - PASS\_MAX\_DAYS 60  
      PASS\_MIN\_DAYS 1  
      PASS\_WARN\_AGE 7
  + **Why It Works**: This sets the maximum password age to 60 days, with a minimum age of 1 day. This ensures that passwords are changed regularly and that new passwords cannot be created rapidly.
  + **Patch** edit /etc/pam.d/common-password line 25 to:
  + password requisite pam\_pwquality.so retry=3 minlen=10 maxrepeat=3 ucredit=-1 lcredit=-1 dcredit=-1 ocredit=-1 difok=3 gecoscheck=1 reject\_username enforce\_for\_root
  + **Why It Works**: The rules above set password requirements as follows:
    - retry=3: Prompt a user 3 times before returning with error.
    - minlen=10: The password length cannot be less than this parameter
    - maxrepeat=3: Allow a maximum of 3 repeated characters
    - ucredit=-1: Require at least one uppercase character
    - lcredit=-1: Must have at least one lowercase character.
    - dcredit=-1: must have at least one digit
    - difok=3 : The number of characters in the new password that must not have been present in the old password.
    - gecoscheck=1: Words in the GECOS field of the user’s passwd entry are not contained in the new password.
    - reject\_username: Rejects the password if contains the name of the user in either straight or reversed form.
    - enforce\_for\_root: Enforce password policy for root user
* Vulnerability: Users and Permissions
  + **Patch**:
  + useradd -m -c "Admin User" admin  
    passwd admin  
    usermod -aG sudo admin  
    su admin  
    sudo vi /etc/passwd  
      
    # change root:x:0:0:root:/root:/bin/bash  
    to  
    # change root:x:0:0:root:/root:/sbin/nologin
  + **Why It Works**: The commands above create an admin user with sudo priviliges. From there the new admin user disables login as root. This makes it so that people must use an admin account to view or change sensitive files.
  + **Patch**: Generally restrict user access to files and folders by using user groups
  + **Why It Works**: By creating user groups for Wordpress, Michael, and Steven you can ensure that only specific users have access to the information they need.