# Blue Team: Summary of Operations

## Table of Contents

* Network Topology
* Description of Targets
* Monitoring the Targets
* Patterns of Traffic and Behavior
* Suggestions for Going Further

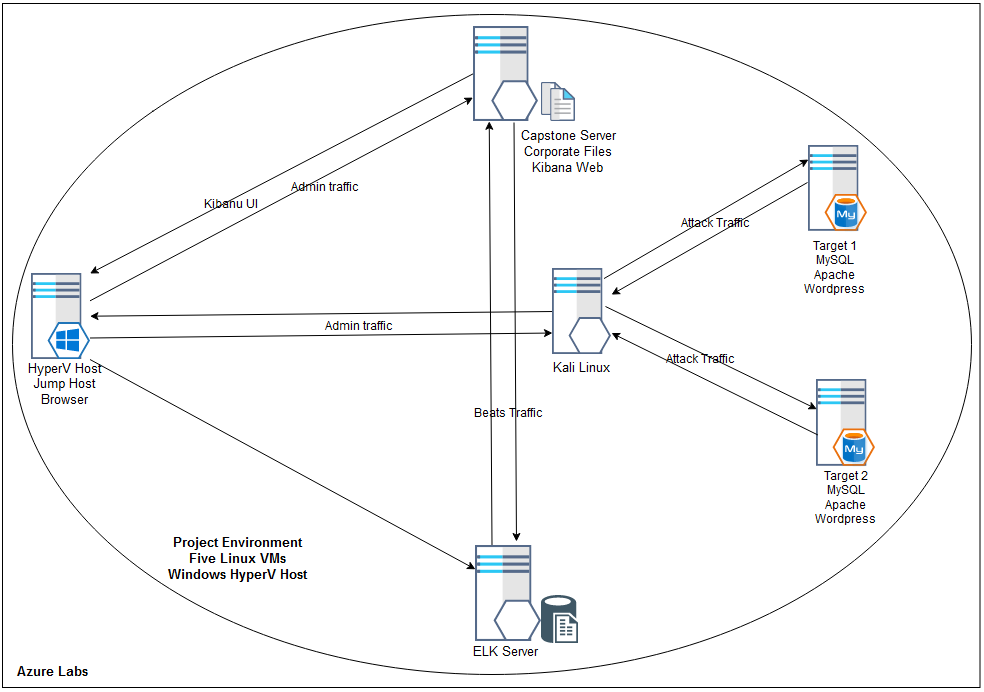
## Network Topology

NETWORK

Address Range: 192.168.1.0/24

Netmask: 255.255.255.0

Gateway: 192.168.1.1



The following machines were identified on the network:

**Capstone**

* Operating System: Linux 4.15 - 5.6
* Purpose: Corporate Web Server
* IP Address: 192.168.1.105

**ELK**

* Operating System: Linux 4.15 - 5.6
* Purpose: Monitoring Server (Elastic Stack)
* IP Address: 192.168.1.100

**Target 1**

* Operating System: Linux 3.2 - 4.9
* Purpose: Web Application Server
* IP Address: 192.168.1.110

**Target 2**

* Operating System: Linux 3.2 - 4.9
* Purpose: Web Application Server
* IP Address: 192.168.1.115

**Kali**

* Operating System: Linux 2.6.32
* Purpose: Pentest Station
* IP Address: 192.168.1.90

## Description of Targets

* Two VMs on the network were vulnerable to attack:

Target 1 192.168.1.110 and Target 2 192.168.1.115

* Each VM functions as an Apache web server and has SSH enabled, so ports 80 and 22 are possible ports of entry for attackers.

## Monitoring the Targets

This scan identifies the services below as potential points of entry:

**Target 1**

PORT  STATE SERVICE     VERSION

22/tcp  open  ssh         OpenSSH 6.7p1 Debian 5+deb8u4 (protocol 2.0)

80/tcp  open  http        Apache httpd 2.4.10 ((Debian))

111/tcp open  rpcbind     2-4 (RPC #100000)

139/tcp open  netbios-ssn Samba smbd 3.X - 4.X

445/tcp open  netbios-ssn Samba smbd 3.X - 4.X

**Target 2**

PORT  STATE SERVICE     VERSION

22/tcp  open  ssh         OpenSSH 6.7p1 Debian 5+deb8u4 (protocol 2.0)

80/tcp  open  http        Apache httpd 2.4.10 ((Debian))

111/tcp open  rpcbind     2-4 (RPC #100000)

139/tcp open  netbios-ssn Samba smbd 3.X - 4.X

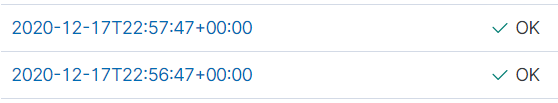
445/tcp open  netbios-ssn Samba smbd 3.X - 4.X

Traffic to these services should be carefully monitored. To this end, we have implemented the alerts below: (Note: Add at least three alerts. You can add more if time allows.)

**Alert 1: Excessive HTTP Requests**

Excessive TCP Requests is implemented as follows:

* Metric: http.response.status\_code
* Threshold: 400 in the last 5 minutes
* Vulnerability Mitigated: intensive scans
* Reliability: High, this alert generates no false positives.



**Alert 2: Excessive Login Requests**

Excessive Login Requestsis implemented as follows:

* Metric: user.name
* Threshold: 10 in last 1 minute
* Vulnerability Mitigated: bruteforce password attacks
* Reliability: High, this alert generates no false positives



**Alert 3: File Uploads**

File Uploads is implemented as follows:

* Metric: file.created
* Threshold: 1 in the last minute
* Vulnerability Mitigated: uploading malicious files
* Reliability: Low. This alert requires configuration to specific folder.



## Suggestions for Going Further

The vulnerabilities targeted from the above alerts can best be remediated by hardening the platform and patching installed software.

The logs and alerts generated during the assessment suggest that this network is susceptible to several active threats. 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:

**SSH configuration has not been secured**

* Patch: install fail2ban ‘sudo apt-get install fail2ban
* Why It Works: we configure fail2ban to harden entry point configuration which protects the server against attacks, SSHd in particular.
* First, we establish a base firewall configuration, being the default policy for the machine.
* Then we use fail2ban to set firewall rules in iptables
* fail2ban is adaptive - it will respond to current traffic patterns by invoking a policies that bans suspicious IPs for a configured time period.
* An advantage of fail2ban is that we maintain our security configuration separate to our regular system configuration

**System component configuration hardening**

**Apache Server** default configuration should be tailored with security in mind

* Harden permissions on ServerRoot Directories

mkdir /usr/local/apache (ServerRoot)  
cd /usr/local/apache   
mkdir bin conf logs   
chown 0 . bin conf logs   
chgrp 0 . bin conf logs   
chmod 755 . bin conf logs

* Harden permissions when you install httpd

cp httpd /usr/local/apache/bin   
chown 0 /usr/local/apache/bin/httpd   
chgrp 0 /usr/local/apache/bin/httpd   
chmod 511 /usr/local/apache/bin/httpd

* Server Side Includes

SSI-enabled files should have a separate extension, such as .shtml

disable the ability to run scripts and programs from SSI pages

* Check CGI scripts - CGI scripts can run essentially arbitrary commands on your system with the permissions of the web server.
* Other dynamic content Embedded scripting options which run as part of the server itself, such as mod\_php, mod\_perl, mod\_tcl, and mod\_python, run under the identity of the server itself. Scripts executed by these engines potentially can access anything the server user can, so use trusted code only here too.
* Prevent the use of .htaccess files in all directories apart from those specifically enabled by adding the below to your server.conf

<Directory "/">

AllowOverride None

</Directory>

* Protect server files by default by adding the below to your server.conf

<Directory "/">

Require all denied

</Directory>

The add appropriate Directory blocks to allow access only to specified areas

<Directory "/usr/users/\*/public\_html">

Require all granted

</Directory>

<Directory "/usr/local/httpd">

Require all granted

</Directory>

Ensure there are no Location directives that will overwrite the Directory blocks

* The Protect your UserDir in your server.conf

UserDir disabled root

* Secure and watch your log files. An attacker may gain system information from reading server logs. Understanding your server logs will provide details of normal behaviour and make it easier to identify or rule out issues during an incident

**MySQL** should be installed via a secure installation which allows us to set of security options during the installation. MySQL secure installation may be run for an existing installation also.

sudo mysql\_secure\_installation

Note: There is a choice of password validation policy levels included so ensure you are familiar and understand the selection you will make here. Were an existing account to have problems after the secure installation (such as a database user for a running service like WordPress) we can change the database user password for the user/service with this command:

mysqladmin -u USER -p password NEWPASSWORD

* Update the password plugin (Y and hit Enter)
* Select the level of password validation policy you want to enable (0 = low, 1 = medium, 2 = strong).
* Change the root password (to dismiss this option type N)
* Remove anonymous users (Y to remove)
* Disallow remote root login (Y to disable)
* Remove test database (Y to remove)

**WordPress** has numerous features that can be exploited and a raft of plugins are available that can be used to remediate these vulnerabilities.

* Second factor authentication – prevents brute force attacks on user accounts
* Installing a WordPress security plugin such as MalCare will assist administrators with a clear interface to modify important settings
* The first step is to scan and clean our website.
* Limiting Logins via a plugin like Limit Login Attempts Reloaded or by modifying functions.php
* Block PHP execution in untrusted folders; add the following to /public\_html/.htaccess

<Files \*.php>  
deny from all  
</Files>

* Disable file editor plugin via modifying wp-config
* Mandate stronger passwords (via MalCare plugin)
* Change security keys and encrypt wp-config (via MalCare plugin)
* Prevent plugin installations – via edit of wp\_config.php to add this line;

define(‘DISALLOW\_FILE\_MODS’,true);

* Automatically log out inactive users - install a plugin such as Bullet Proof Security
* Audit logging - install plugin WP Security Audit Log
* Set up a Web Application Firewall via a security plugin
* Set up alerts for suspicious activity and malware detection (via MalCare plugin)
* SSL Certificates for secure communications, especially for internet requests

**Non-current versions of component software**

* Patch: install security updates for installed components
* Why It Works: Maintaining current versions for openssl, Apache, MySQL, Wordpress, etc; will remediate the platform vulnerabilities discovered.
* Vendors release supported patches / fixes in response to reported vulnerabilities.
* Installation can be automated via docker playbook or other centralised distribution

**Lack of security focus for IT team**

* Patch: install lynis – ‘apt-get install lynis’
* Why It Works: Insecure configuration is at the core of the bulk of CWE items detailed. Lynis can be used to audit configuration of the system, and docker files. The information you gather from lynis provides focus on securing configuration.
* ‘./lynis audit system –quick –auditor “Blue Team Member”