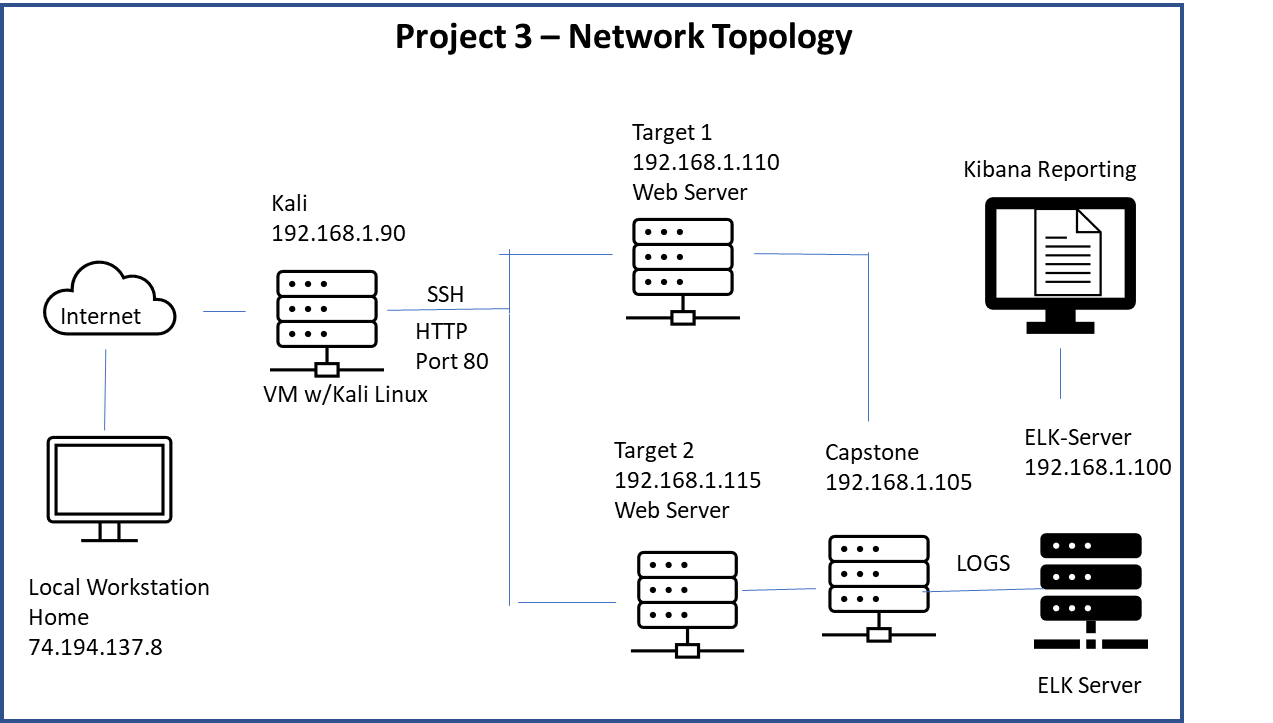
# Blue Team: Summary of Operations (JD Haynes)

## Table of Contents

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

## Network Topology

The following machines were identified on the network:



**Target System 1**

* Operating System: Linux Ubuntu
* Purpose: Web Server
* IP Address: 192.168.1.110
* MAC Address: 00:15:5d:00:04:10

**Target System 2**

* Operating System: Linux Ubuntu
* Purpose: Web Server
* IP Address: 192.168.1.115
* MAC Address: 00:15:5d:00:04:11

**System 3**

* Operating System: ?
* Purpose: Capstone
* IP Address: 192.168.1.105
* MAC Address: 00:15:5d:00:04:0f

**System 4**

* Operating System: ?
* Purpose: ELK Server
* IP Address: 192.168.1.100
* MAC Address: 4c:eb:42:d2:d5:d7

**System 5**

* Operating System: ?
* Purpose: ?
* IP Address: 192.168.1.1
* MAC Address: 00:15:5d:00:04:03

## Description of Targets

* Two VMs on the network were vulnerable to attack: Target 1 at IP address 192.168.1.110 and Target 2 at IP address 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 – List of potentially vulnerable services**

* HTTP - Apache
* SSH
* Wordpress – SQL
* Ports 22, 80, 111, 139, 445 open

**Target 2 – List of potentially vulnerable services**

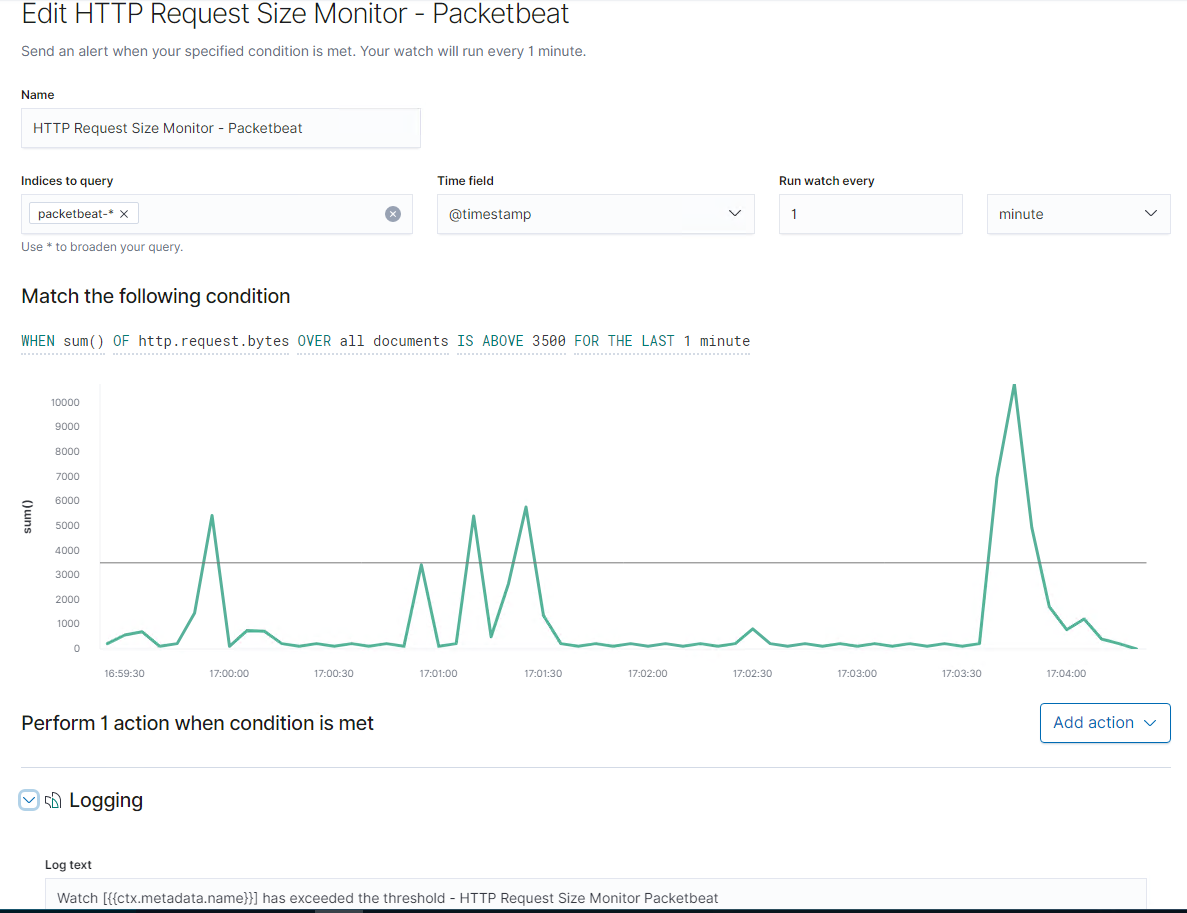
* HTTP - Apache
* SSH
* Wordpress – SQL
* Ports 22, 80, 111, 139, 445 open

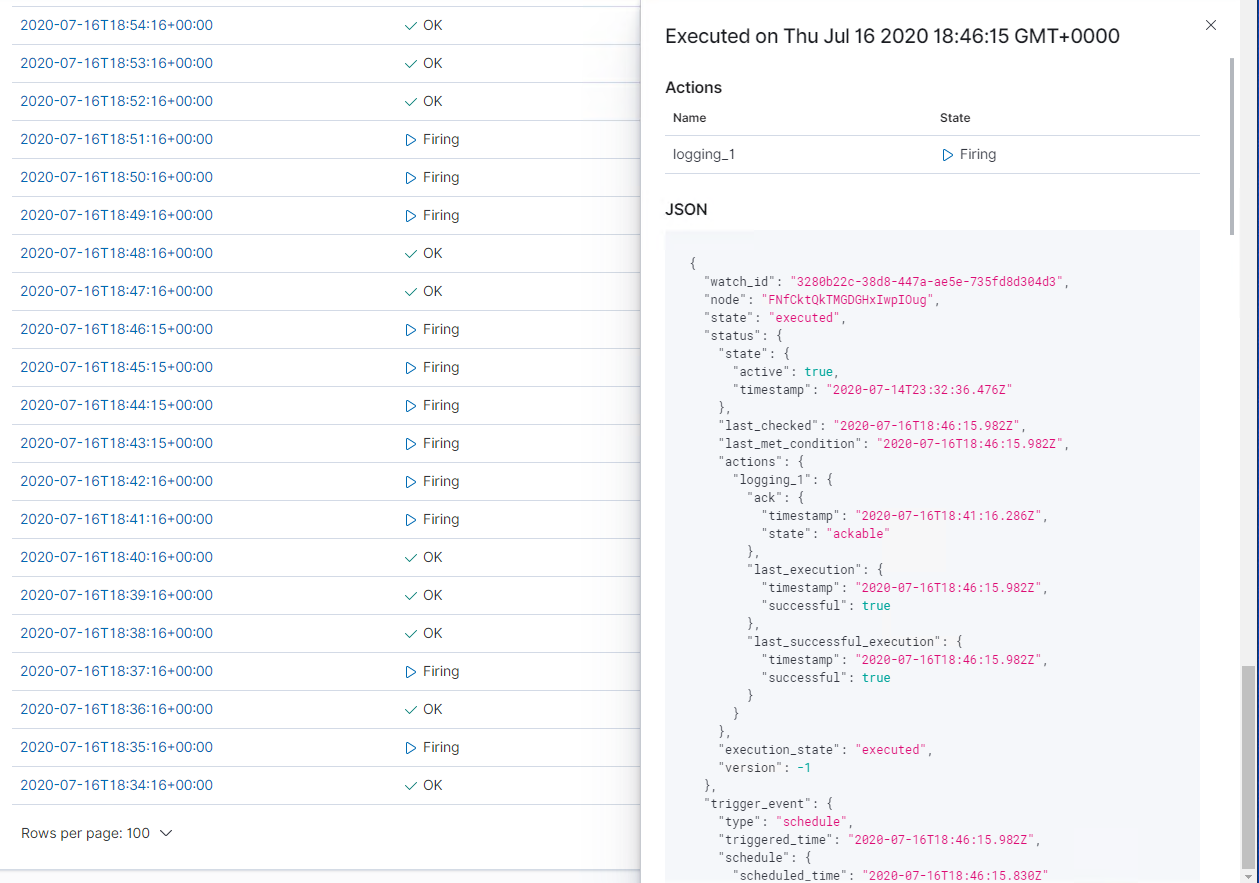
Traffic to these services should be carefully monitored. To this end, we have implemented the alerts below: (Note: Add at least three alerts. You can add more if time allows.)

**Alert 1 – HTTP Request Size Monitor**

HTTP Request Size Monitor is implemented as follows:

* Metric: Measures bytes of http requests per minute using Packetbeat
* Threshold: WHEN sum() OF http.request.bytes OVER all documents IS ABOVE 3500 FOR THE LAST 1 minute
* Vulnerability Mitigated: Ongoing file uploads or downloads which could include malware exploits and payloads.
* Reliability: Medium reliability – can generate alerts when the requests are legitimate.

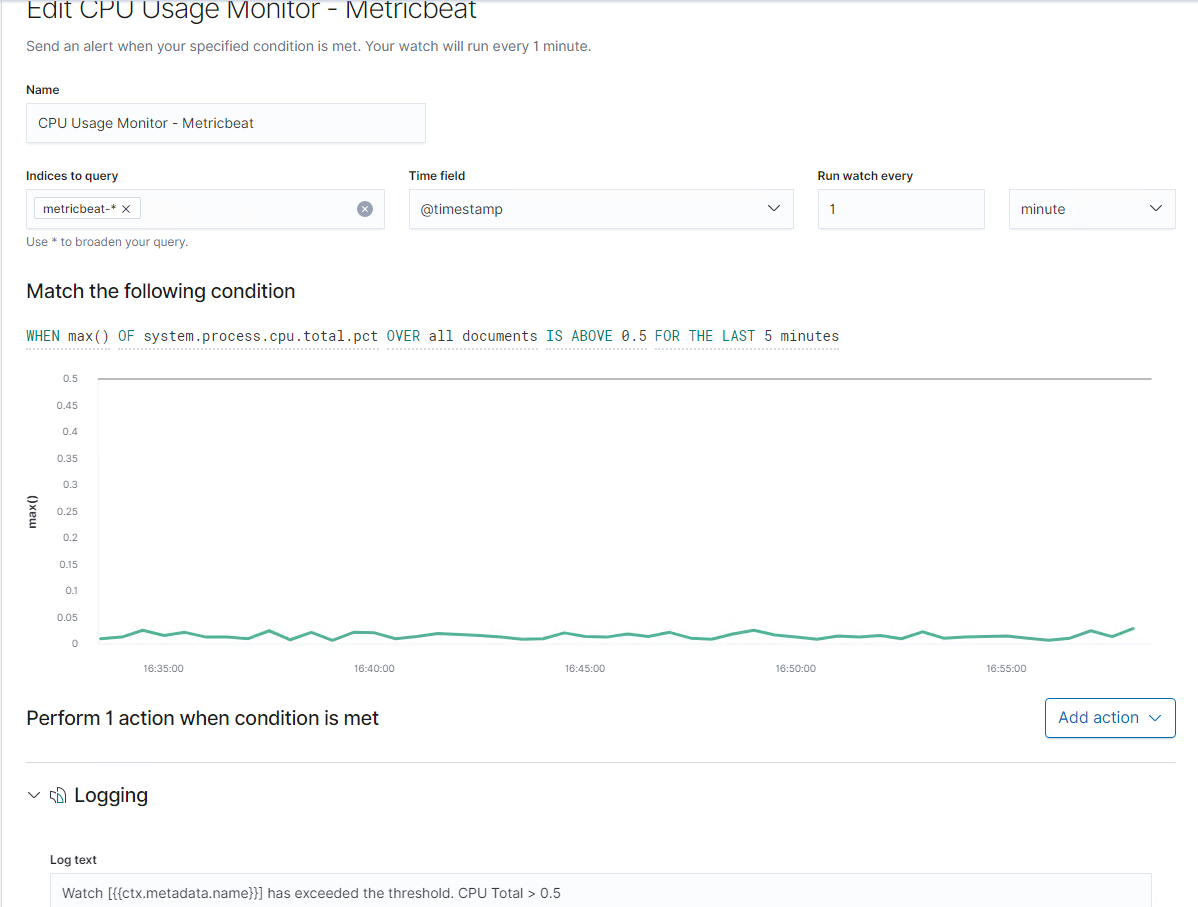




**Alert 2 – CPU Usage Monitor**

CPU Usage Monitor is implemented as follows:

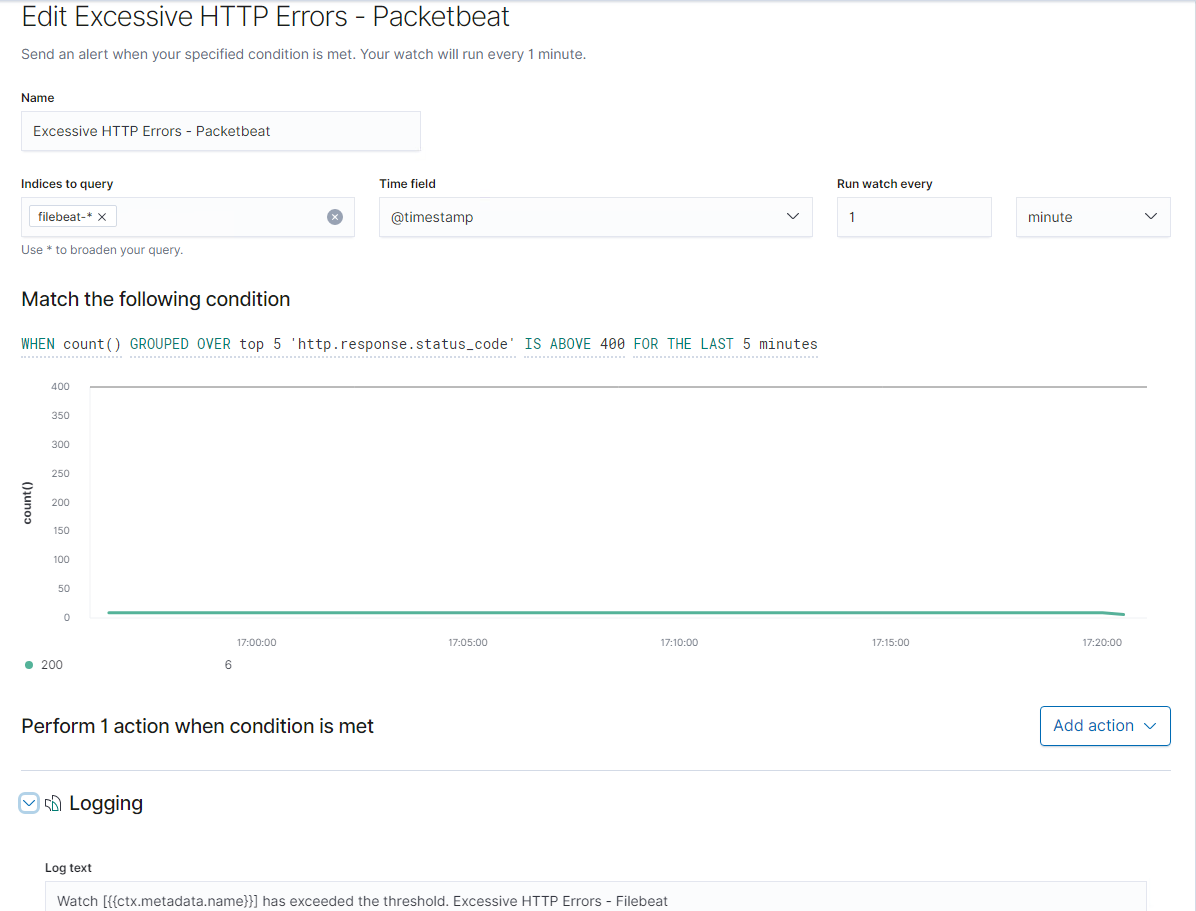
* Metric: Monitor the CPU usage or volume of work over time
* Threshold: WHEN max() OF system.process.cpu.total.pct OVER all documents IS ABOVE 0.5 FOR THE LAST 5 minutes
* Vulnerability Mitigated: can indicate whether cluster is sized correctly but can also be triggered when malware is scanning a system, info is being sent back and forth between the system and the C2, or when the system is working as part of a botnet.
* Reliability: High reliability, does not generate a lot of false positives. Did not fire since the highest measurement was 0.028.

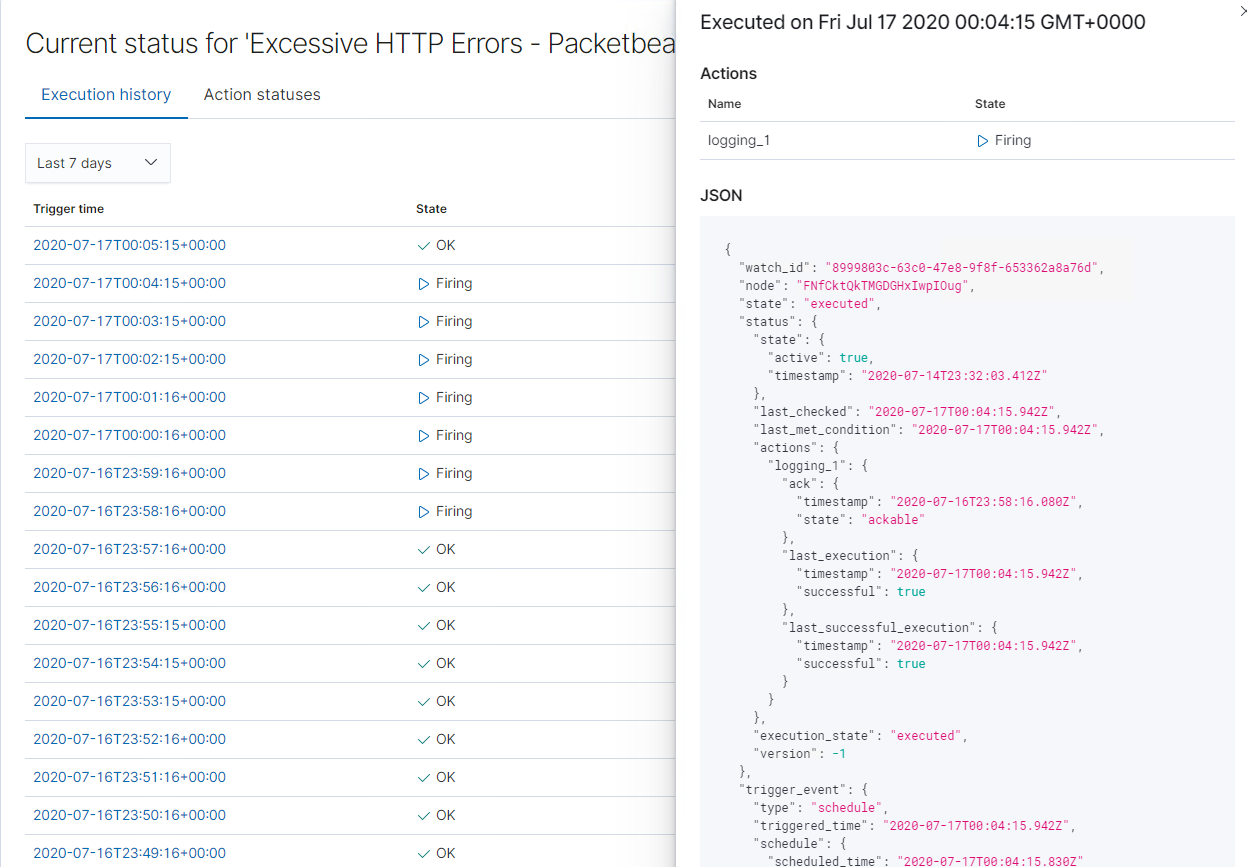


**Alert 3 – Excessive HTTP Errors**

Excessive HTTP Errors is implemented as follows:

* Metric: Measures when an excessive number of HTTP error status codes such as 400/Bad Request, 401/Unatuthorized, 403/Forbidden, 429/Too Many Requests, etc. occur
* Threshold: WHEN count() GROUPED OVER top 5 ‘http.response.status\_code’ IS ABOVE 400 FOR THE LAST 5 minutes
* Vulnerability Mitigated: Attempts by bad actors to access system from black-listed Ips, unauthenticated clients, high number of requests that could lead to DOS attack,
* Reliability: High Reliability, does not generate many false positives or false negatives. While it can catch HTTP errors that are high volume, it does not catch lower volume messages such as redirect error codes (302, 307, 308) that can be used to send users to a malicious URL.





**Alert 4 – SSH Logins**

An SSH login alert was not configured, but this would have been a good alert to set up – to monitor SSH logins since port 22 is open.

## Suggestions for Going Further

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 in addition **to patching and upgrading all software/applications - including Apache, Wordpress, PHP, SSH, SQL, and the OS - to the most current version**, IT should also implement the fixes below to protect the network:

**Vulnerability 1 – HTTP**

* Patch 1: Suppress the web server vendor name and version (Apache). This can be done by modifying the site configuration file in Apache or the PHP.INI file in PHP.
* Why It Works: It is more difficult for an attacker to determine the web server vendor and version which reduces the chances they will know which attack to apply
* Patch 2: Sanitize the website input for all fields by using the frameworks filtering functions and also don’t return HTML tags to the client.
* Why It Works: Sanitizing the input can prevent hackers from accessing the system through SQL injection and defend against HMTL injection by XSS.
* Patch 3: Convert from HTTP to HTTPS.
* Why It Works: HTTPS is more secure and less vulnerable to hackers.

**Vulnerability 2 - SSH**

* Patch 1: Set a custom SSH port by modifying the SSH main config file to set the port to something other than port 22
* Why It Works: Port 22 is widely known and used by hackers to launch attacks by SSH
* Patch 2: Use TCP wrappers by modifying the /etc/hosts.allow and the /etc/hosts.deny files. For example if you want to allow access from your static home IP, add this to the end of the /etc/hosts.allow file: <sshd: xx.xx.xx.xx> with the x’s representing your static home IP.
* Why It Works: Essentially whitelists which IP addresses can use SSH to access the system.
* Patch 3: Filter the SSH port to your firewall by setting up which ports can go in to and come out of the system. A custom SSH port can be set up here as well.
* Why It Works: Reduces which incoming and outgoing ports can be used by attackers.
* Patch 4: Disable root login from port 22 by setting the “PermitRootLogin” variable to “no” in /etc/ssh/sshd\_config
* Why It Works: Prevents hackers from gaining root access through SSH, especially when passwords are weak.
* Patch 5: Set a custom SSH port by modifying the SSH main config file to set the port to something other than port 22
* Why It Works: Port 22 is widely known and used by hackers to launch attacks by SSH
* Patch 6: Strong password/passphrase policy that includes 12 character minimum, 1 upper and 1 lower case character, 1 number, 1 symbol.
* Why It Works: Makes it more difficult for attackers to successfully use password cracking tools
* Patch 7: Limit max authentication attempts by changing “MaxAuthTries” so a low values such as “3” in the /etc/ssh/sshd\_config file.
* Why It Works: Helps prevent password hacking via guessing or password dictionary tools.

**Vulnerability 3 - Wordpress**

* Patch 1: Strong password/passphrase policy that includes 12 character minimum, 1 upper and 1 lower case character, 1 number, 1 symbol.
* Why It Works: Makes it more difficult for attackers to successfully use password cracking tools
* Patch 2: Upgrade all WordPress plugins and themes to the latest versions. This is best done by enabling automatic updates following the detailed instructions here: <https://kinsta.com/blog/wordpress-automatic-updates/>
* Why It Works: WordPress uses many 3rd party plugins and themes that are vulnerable to attackers – keeping these patched and upgraded helps prevent successful attacks.
* Patch 3: Change and hide the WordPress login URL by changing the wp-admin to a different name using a plugin such as WPS Hide login or Perfmatters..
* Why It Works: The domain.com/wp-admin is known as an easy access to hackers.
* Patch 4: Limit login attempts with a plugin such as Cerber Limit Login Attempts or Login LockDown.
* Why It Works: It can lock out the hacker before a successful login occurs.
* Patch 5: Harden the wp-config.php file by moving it out of the default directory
* Why It Works: This can make it harder for a hacker to locate the wp-config.php file which is commonly used for exploits.
* Patch 6: Change permissions on files in the root directory from 644 to 440 or 400 using the FTP client.
* Why It Works: 644 means the files are readable and writeable by the file owner and readable by everyone else. Setting the permissions to 440 or 400 will prevent other users the server from reading the files in the root directory.
* Patch 7: Hide your WordPress version by adding the code below to the WordPress theme’s “functions.php” file:

function wp\_version\_remove\_version() {

return '';

}

add\_filter('the\_generator', 'wp\_version\_remove\_version');

* Why It Works: Attackers can not discover version specific vulnerabilities as easily.