Final Engagement

Attack, Defense & Analysis of a Vulnerable Network

By: Asfandiyar Qamar, Shay Rabbers, Haris Mian, Joseph Kays, Jonas Halberg, Kaich Ogul

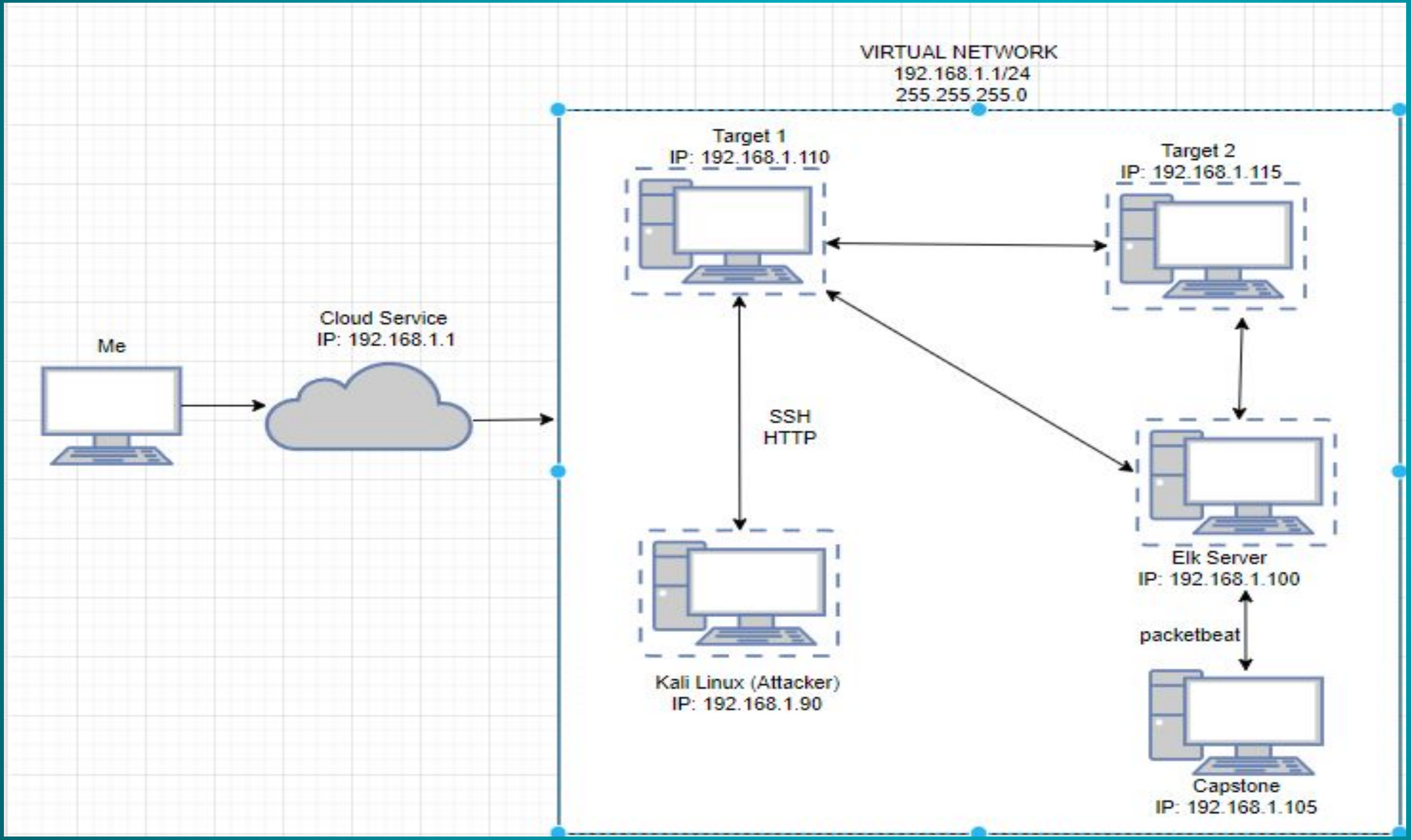
Table of Contents

This document contains the following resources:

- **Network Topology & Critical Vulnerabilities**
- **Alerts Implemented**
- **Hardening**
- **Implementing Patches**

Network Topology
& Critical Vulnerabilities

Network Topology



Network
Address Range:
192.168.1.1/24
Netmask:255.255.255.0
Gateway: 192.168.1.1

Machines
IPv4: 192.168.1.110
OS: Linux
Hostname: Target 1

IPv4: 192.168.1.115
OS: Linux
Hostname: Target 2

IPv4: 192.168.1.90
OS: Kali Linux
Hostname: Kali

IPv4: 192.168.1.100
OS: Linux
Hostname: Elk

IPv4: 192.168.1.105
OS: Linux
Hostname: Capstone

Critical Vulnerabilities: Target 1

Our assessment uncovered the following critical vulnerabilities in Target 1.

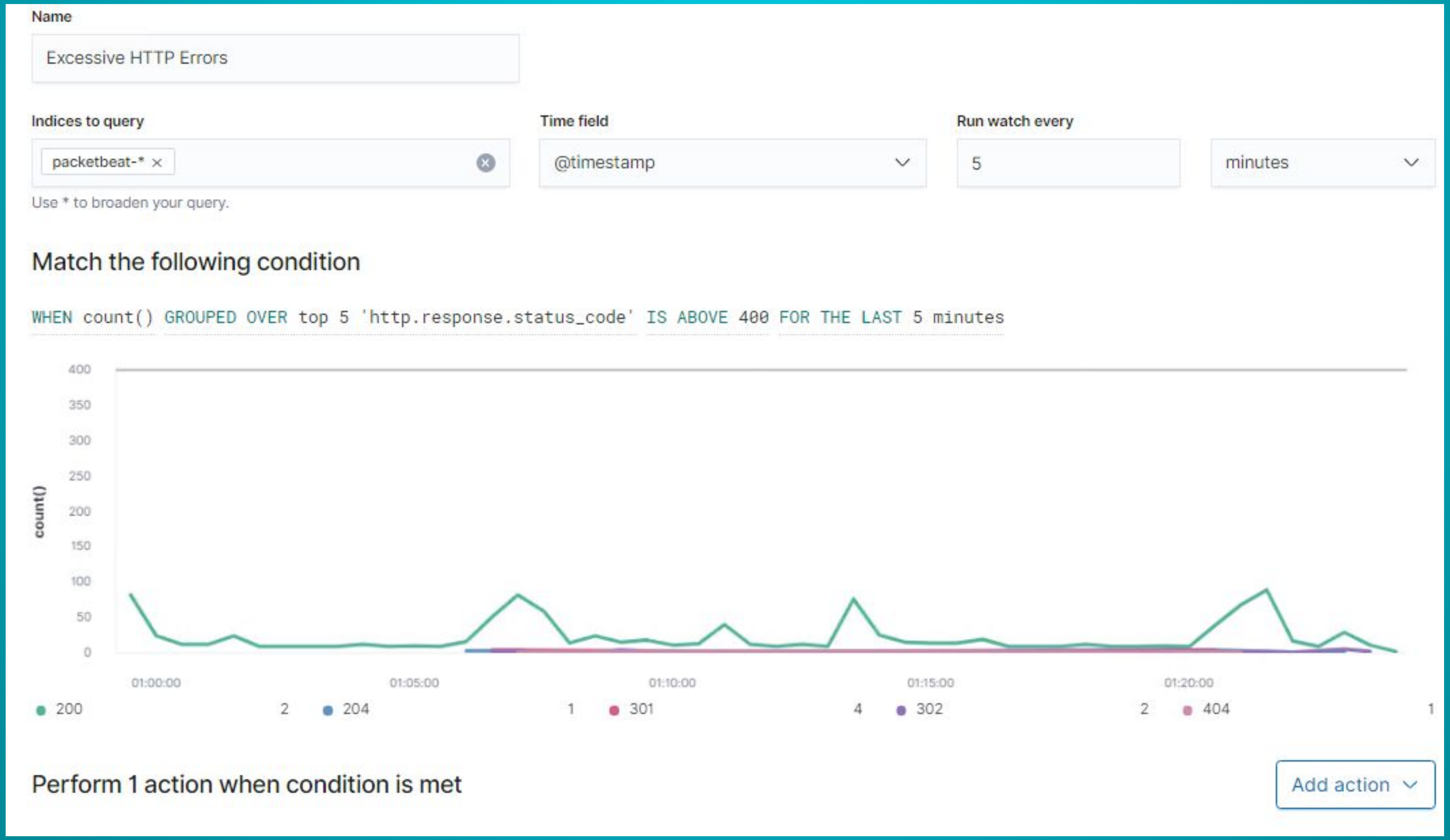
Vulnerability	Description	Impact
Allowed SSH	The target machine allowed remote access through port 22	Path to gain access was identified
User enumeration	The system allowed user enumeration through WPScan	Discovered all publicly available usernames: michael and steven
Weak and Unsalted Passwords	User micheal's password was easy to guess, and Brute Force attack also revealed their password easily. User steven's password was easily cracked using JohnTheRipper.	User michael's password was *michael* and user steven's was *pink84*
Misconfiguration of Privileges and No Security on File Access	Plain text passwords contained in wp-config.php to MYSQL database and ability to run Python commands	Used username *root* and password *R@v3nSecurity* to log into the MySQL database

Alerts Implemented

Excessive HTTP Errors

Summarize the following:

- **Metric =** WHEN counts () GROUPED OVER top 5 ‘http.response.status_code’
- **Threshold =** IS ABOVE 400 for the LAST 5 minutes



Name

Excessive HTTP Errors

Indices to query

packetbeat-* ×

Use * to broaden your query.

Time field

@timestamp

Run watch every

5

minutes

Match the following condition

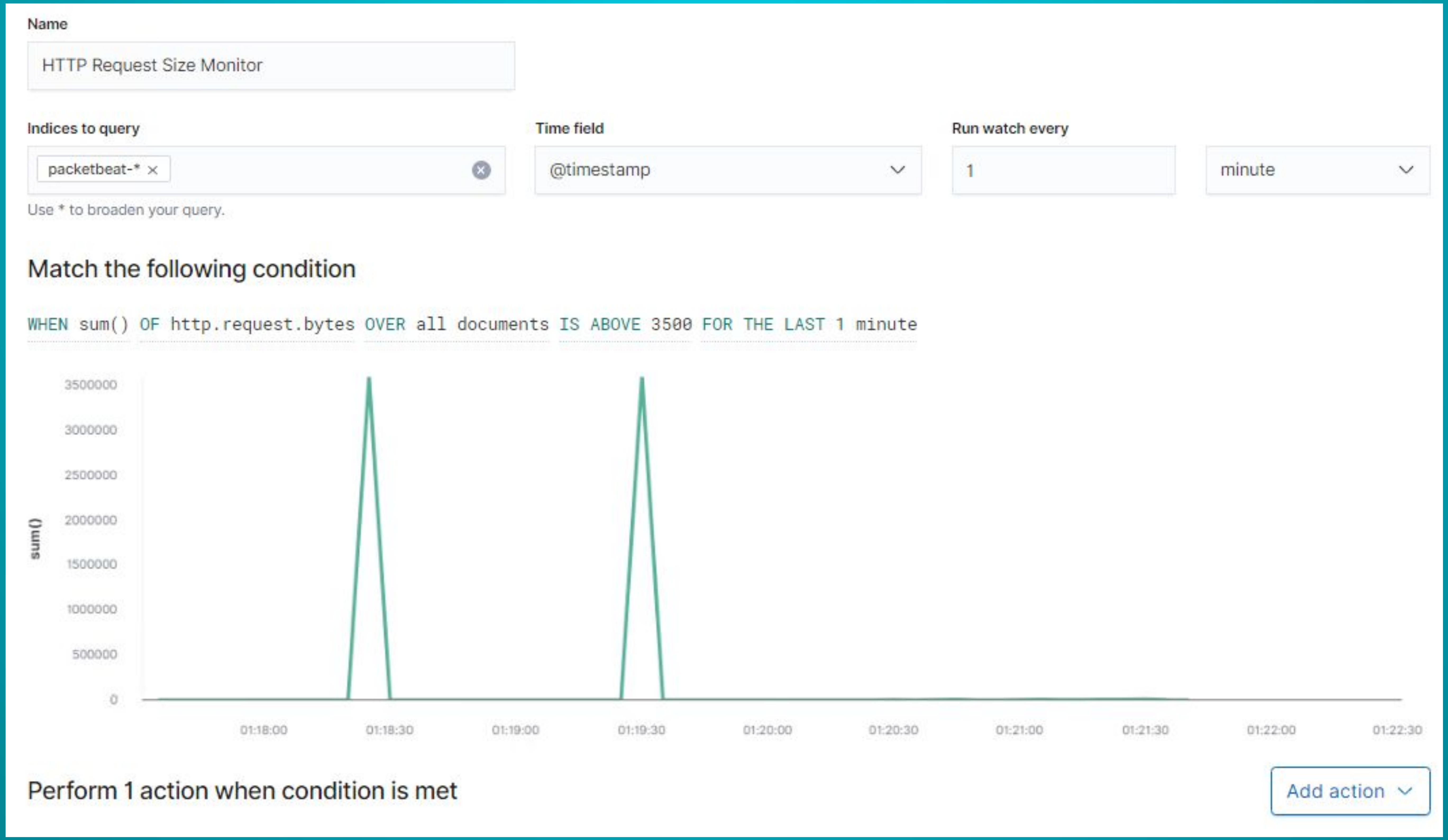
```
WHEN count() GROUPED OVER top 5 'http.response.status_code' IS ABOVE 400 FOR THE LAST 5 minutes
```

Perform 1 action when condition is met

HTTP Request Size Monitor

Summarize the following:

- **Metric =** WHEN sum () of ‘http.request.bytes’ OVER all Documents
- **Threshold =** IS ABOVE 3500 FOR THE LAST 1 minute



Name

HTTP Request Size Monitor

Indices to query: packetbeat-*

Use * to broaden your query.

Time field: @timestamp

Run watch every: 1 minute

Match the following condition

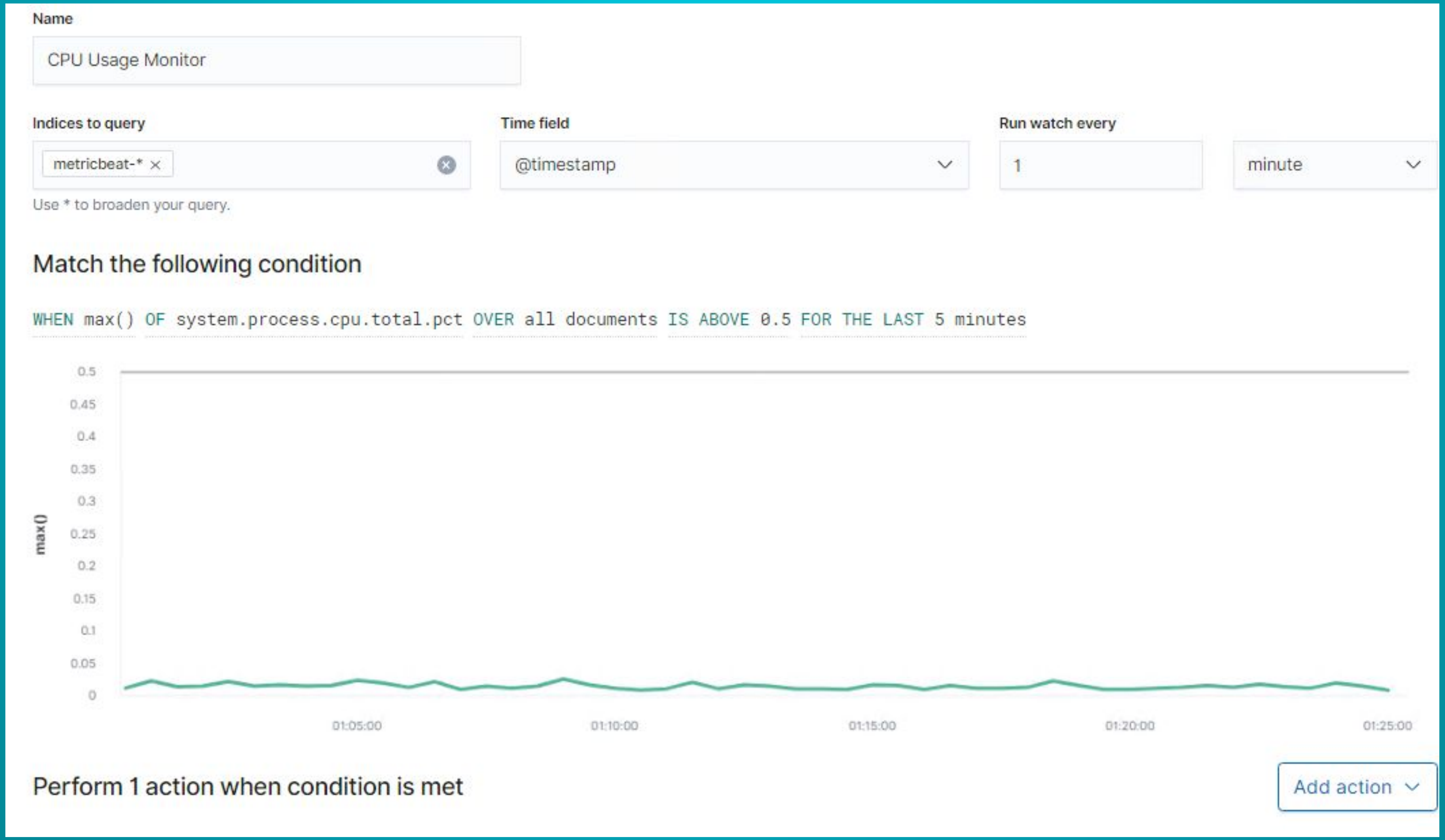
```
WHEN sum() OF http.request.bytes OVER all documents IS ABOVE 3500 FOR THE LAST 1 minute
```

Perform 1 action when condition is met

CPU Usage Monitor

Summarize the following:

- **Metric =** When max () OF ‘system.process.cpu.total.pct’ OVER all documents
- **Threshold =** IS ABOVE 0.5 FOR THE LAST 5 minutes



Name

CPU Usage Monitor

Indices to query: metricbeat-*

Use * to broaden your query.

Time field: @timestamp

Run watch every: 1 minute

Match the following condition

```
WHEN max() OF system.process.cpu.total.pct OVER all documents IS ABOVE 0.5 FOR THE LAST 5 minutes
```

Perform 1 action when condition is met

Hardening

Hardening Against SSH connection

- Change ssh to a different port than the default port to keep intruders guessing
    - locate and edit the sshd_config file > /etc/ssh/sshd_config
    - Change #Port 22 to any other port, save and close the file
    - disable and then enable ssh service

- Disable SSH service
    - systemctl stop ssh
    - systemctl disable ssh

Hardening Against Weak Password

- Implement a complex password policy and require users to change passwords every six months

- Implement 2FA or MFA on all accounts

- Implement controls on amount of invalid login attempts

Hardening Against User Enumeration

- Block any access to specific files in the WordPress root folder.
- Block WPScan from enumeration WordPress plugin version
- Block access to Install.php and Upgrade.php files to anyone.

Hardening Against MySQL Database Access

- Encrypt all files containing credentials and hashes

- Only allow certain admin users access to these files

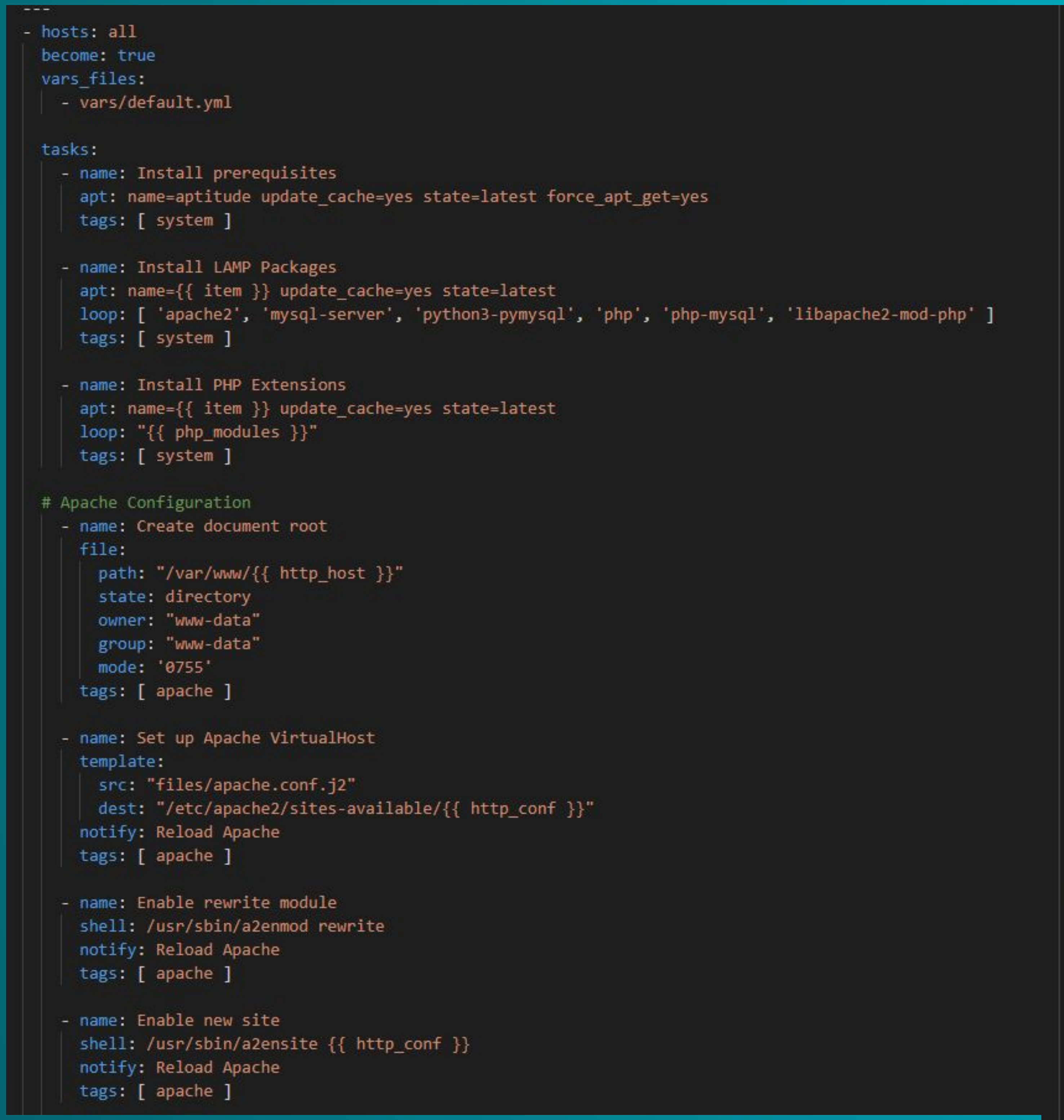
- Connect wordpress to an FTP service and gain access to the htaccess file. Edit the file to deny access to the wp-config.php file

- Ensure WordPress is alway updated to the latest version

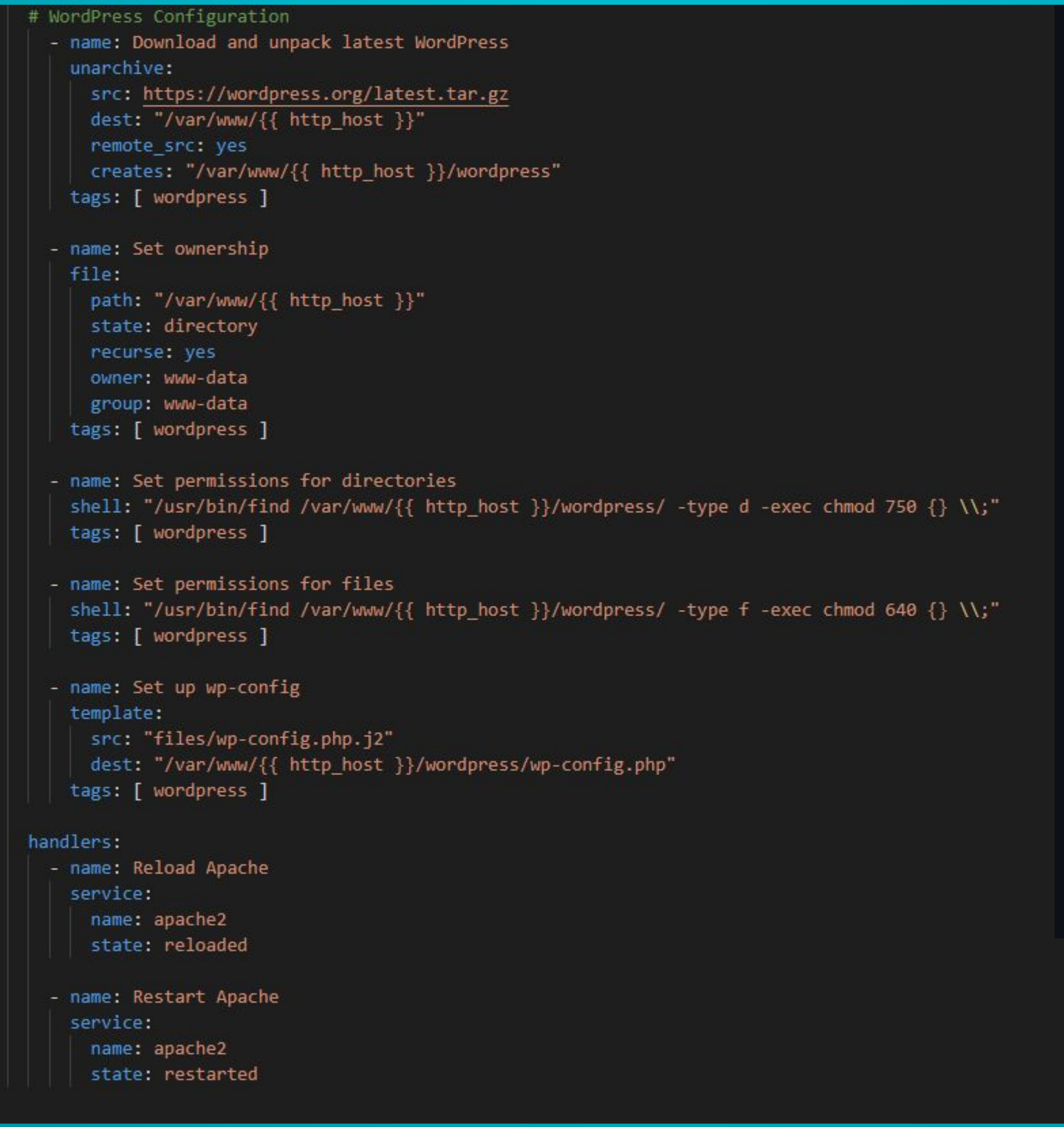
Implementing Patches

Implementing Patches with Ansible

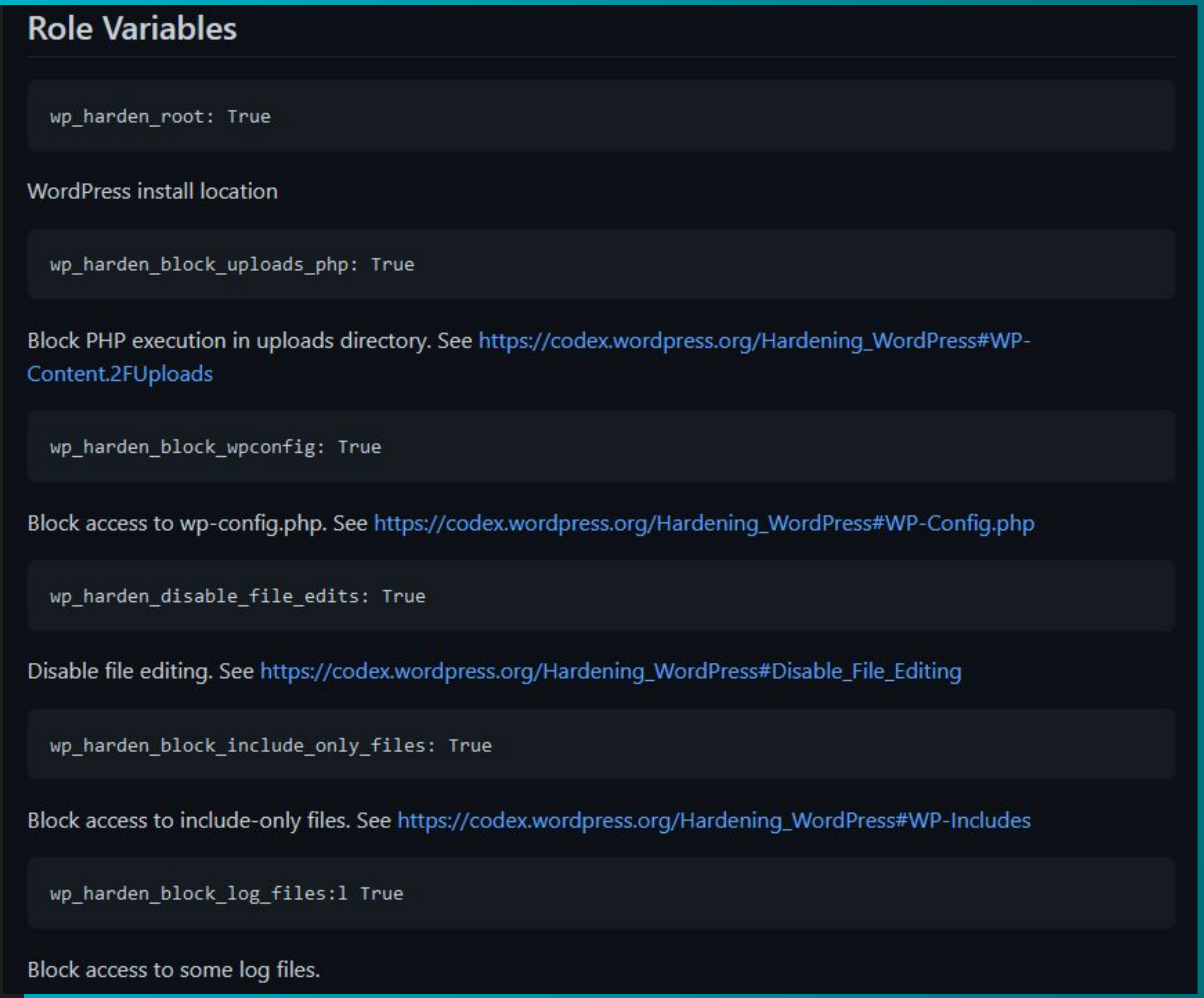
This Ansible Playbook implements hardening measures and update measures to the WordPress configuration files. It also assigns permissions and roles to correct users.



```yaml
---
- hosts: all
  become: true
  vars_files:
    - vars/default.yml

  tasks:
    - name: Install prerequisites
      apt: name=aptitude update_cache=yes state=latest force_apt_get=yes
      tags: [ system ]

    - name: Install LAMP Packages
      apt: name={{ item }} update_cache=yes state=latest
      loop: [ 'apache2', 'mysql-server', 'python3-pymysql', 'php', 'php-mysql', 'libapache2-mod-php' ]
      tags: [ system ]

    - name: Install PHP Extensions
      apt: name={{ item }} update_cache=yes state=latest
      loop: "{{ php_modules }}"
      tags: [ system ]

  # Apache Configuration
    - name: Create document root
      file:
        path: "/var/www/{{ http_host }}"
        state: directory
        owner: "www-data"
        group: "www-data"
        mode: '0755'
      tags: [ apache ]

    - name: Set up Apache VirtualHost
      template:
        src: "files/apache.conf.j2"
        dest: "/etc/apache2/sites-available/{{ http_conf }}"
      notify: Reload Apache
      tags: [ apache ]

    - name: Enable rewrite module
      shell: /usr/sbin/a2enmod rewrite
      notify: Reload Apache
      tags: [ apache ]

    - name: Enable new site
      shell: /usr/sbin/a2ensite {{ http_conf }}
      notify: Reload Apache
      tags: [ apache ]
```



```yaml
  # WordPress Configuration
    - name: Download and unpack latest WordPress
      unarchive:
        src: https://wordpress.org/latest.tar.gz
        dest: "/var/www/{{ http_host }}"
        remote_src: yes
        creates: "/var/www/{{ http_host }}/wordpress"
      tags: [ wordpress ]

    - name: Set ownership
      file:
        path: "/var/www/{{ http_host }}"
        state: directory
        recurse: yes
        owner: www-data
        group: www-data
      tags: [ wordpress ]

    - name: Set permissions for directories
      shell: "/usr/bin/find /var/www/{{ http_host }}/wordpress/ -type d -exec chmod 750 {} \\;"
      tags: [ wordpress ]

    - name: Set permissions for files
      shell: "/usr/bin/find /var/www/{{ http_host }}/wordpress/ -type f -exec chmod 640 {} \\;"
      tags: [ wordpress ]

    - name: Set up wp-config
      template:
        src: "files/wp-config.php.j2"
        dest: "/var/www/{{ http_host }}/wordpress/wp-config.php"
      tags: [ wordpress ]

  handlers:
    - name: Reload Apache
      service:
        name: apache2
        state: reloaded

    - name: Restart Apache
      service:
        name: apache2
        state: restarted
```



Role Variables

```
wp_harden_root: True
```

WordPress install location

```
wp_harden_block_uploads_php: True
```

Block PHP execution in uploads directory. See https://codex.wordpress.org/Hardening_WordPress#WP-Content.2FUploads

```
wp_harden_block_wpconfig: True
```

Block access to wp-config.php. See https://codex.wordpress.org/Hardening_WordPress#WP-Config.php

```
wp_harden_disable_file_edits: True
```

Disable file editing. See https://codex.wordpress.org/Hardening_WordPress#Disable_File_Editing

```
wp_harden_block_include_only_files: True
```

Block access to include-only files. See https://codex.wordpress.org/Hardening_WordPress#WP-Includes

```
wp_harden_block_log_files:l True
```

Block access to some log files.