Final Engagement

Attack, Defense & Analysis of a Vulnerable Network

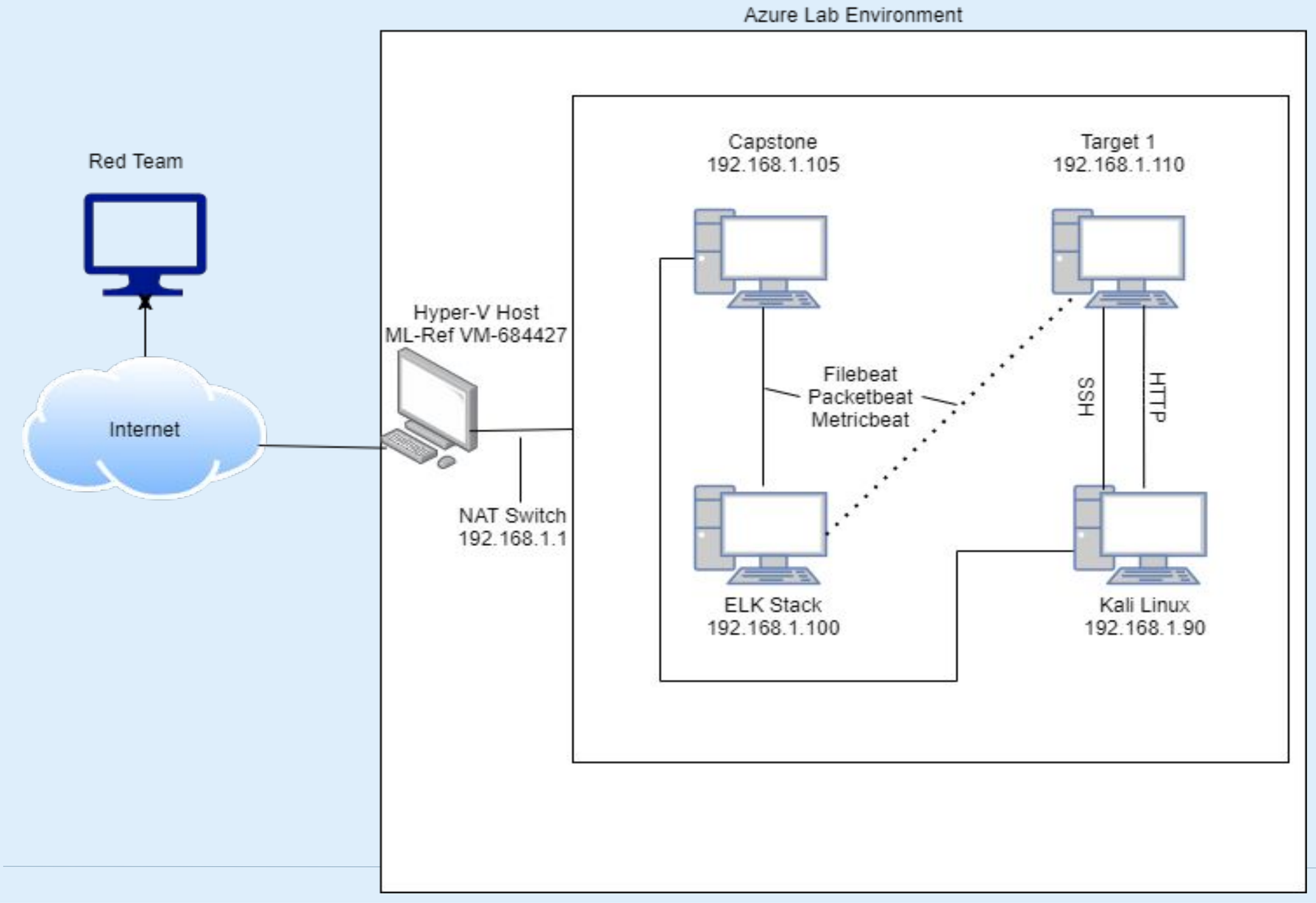
Presentation Created By: Amisha Mehta, Jordan Bow, Tim Applewhite, Nate Kahre

Table of Contents

This document contains the following resources:

Network Topology
& Critical Vulnerabilities

Network Topology



Network
Address Range:192.168.1.1/24
Netmask: 255.255.255.0
Gateway: 10.0.0.1

Machines
IPv4: 192.168.1.105/24
OS: Ubuntu 18.04.1
Hostname: Capstone

IPv4: 192.168.1.100/24
OS:Ubuntu 18.04.4
Hostname: ELK Stack

IPv4: 192.168.1.1/24
OS: Debian GNU/Linux 8
Hostname: Target 1

IPv4: 192.168.1.90/24
OS: Kali Linux
Hostname: Kali

Critical Vulnerabilities: Target 1

Our assessment uncovered the following critical vulnerabilities in **Target 1**.

Vulnerability	Description	Impact
Wordpress enumeration CVE-2017-18536	All WordPress usernames are displayed.	Usernames of the WordPress web application can be discovered. **A6:2017-Security Misconfiguration**
Weak user credentials CWE-521	Simple usernames and passwords.	Weak credentials can be easily guessed or brute forced. **A3:2017-Sensitive Data Exposure**
Misconfiguration of user privileges CWE-269	Misconfigurations allows for privilege escalation.	Root access can be gained by attackers if proper privileges are not set. **A5:2017-Broken Access Control**
Open and Unrestricted SSH access CWE-419	Open and unfiltered access to ports.	Allows attackers to gain access to sensitive data. **A6:2017-Security Misconfiguration**

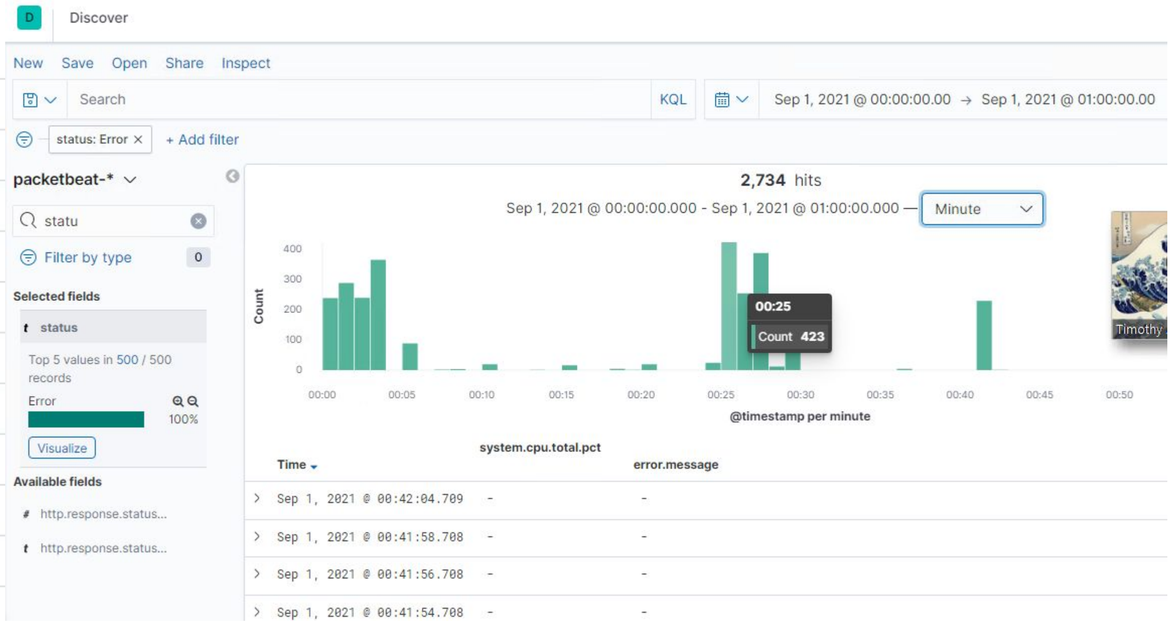
Alerts Implemented

Excessive HTTP Errors Alert

Summarize the following:

- This alert measures error codes above 400 which are client and server errors.
- The threshold is above **400** in 5 minutes.

Trigger time	State	Comment
2021-08-28T07:04:45+00:00	OK	
2021-08-28T07:03:45+00:00	OK	
2021-08-28T07:02:45+00:00	OK	
2021-08-28T07:01:45+00:00	OK	
2021-08-28T07:00:45+00:00	OK	
2021-08-28T06:59:45+00:00	OK	
2021-08-28T06:58:45+00:00	OK	
2021-08-28T06:57:45+00:00	OK	
2021-08-28T06:56:46+00:00	Firing	
2021-08-28T06:55:46+00:00	Firing	



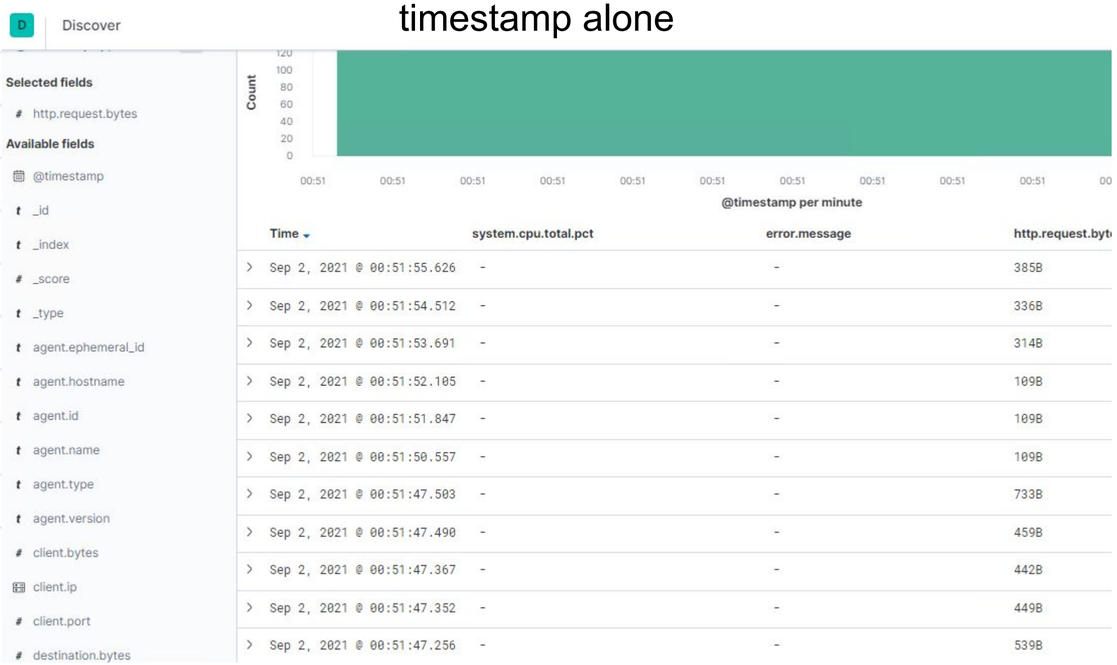
HTTP Request Size Monitor

Summarize the following:

- Which **metric** does this alert monitor? Uses packetbeat to monitor the size of HTTP requests
- What is the **threshold** it fires at? 3500 bytes per minute

3,885 Bytes on this portion of this timestamp alone

Trigger time	State	Comment
2021-08-27T15:45:29+00:00	Firing	
2021-08-27T15:44:28+00:00	OK	
2021-08-27T15:43:28+00:00	OK	
2021-08-27T15:42:28+00:00	OK	
2021-08-27T15:41:28+00:00	OK	
2021-08-27T15:40:28+00:00	OK	
2021-08-27T15:39:28+00:00	OK	
2021-08-27T15:38:28+00:00	OK	
2021-08-27T15:37:28+00:00	OK	
2021-08-27T15:36:28+00:00	OK	

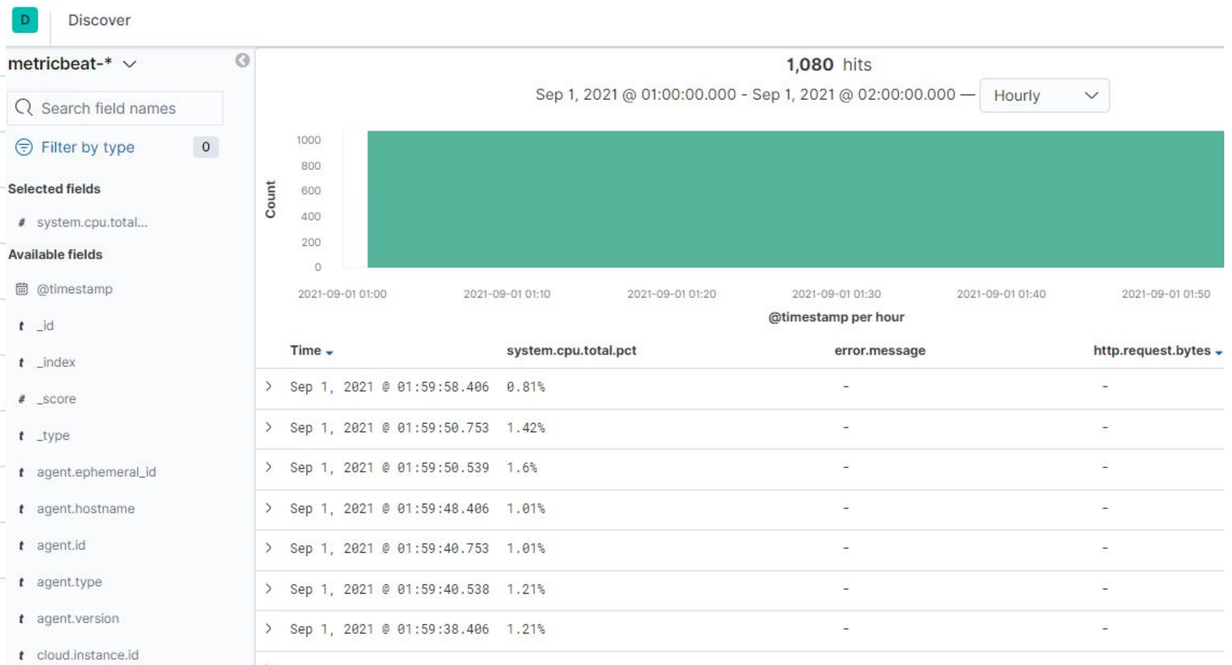


CPU Usage Monitor

Summarize the following:

- Which **metric** does this alert monitor? Uses metricbeat to monitor the cpu usage on the network
- What is the **threshold** it fires at? 50% cpu max per 5 mins

Trigger time	State	Comment
2021-08-27T15:21:29+00:00	Firing	
2021-08-27T15:20:29+00:00	Firing	
2021-08-27T15:19:28+00:00	Firing	
2021-08-27T02:24:48+00:00	OK	
2021-08-27T02:23:48+00:00	OK	
2021-08-27T02:22:48+00:00	OK	
2021-08-27T02:21:48+00:00	OK	
2021-08-27T02:20:48+00:00	OK	
2021-08-27T02:19:48+00:00	OK	
2021-08-27T02:18:48+00:00	OK	



Hardening

Hardening Against Wordpress Enumeration on Target 1

- While it may not be possible to 100% prevent Enumeration of a Wordpress site, we can certainly **reduce the attack surface** and **reduce the likelihood of a breach** by:
- **Block WPScan** from **Enumerating**.
  - We want to block all access to the **nginx configuration file** as well as the **php.ini** file by:
    - Edit **nginx.conf** to reflect **“server_tokens off;”**
    - Edit **php.ini** to reflect **expose_php = Off**
  - **Remove** WordPress **Meta Generator Tags** by adding the following to the **functions.php**:
    - **remove_action (‘wp_head’, ‘wp_generator’);**
    - **remove_action (‘opml_head’, ‘the_generator’);**
  - **Block** WordPress **Plugin Enumeration** From **WPScan**
    - Edit **nginx config**, so **WPScan** will not know what version of **Enumeration Plugin** we have:
      - **location ~* ^/wp-content/plugins/.+\.(txt|log|md)$ {**
        deny all;
        error_page 403 =404 /;
        }

Hardening Against Wordpress Enumeration on Target 1 (continued)

- While it may **not** be possible to **100%** prevent **Enumeration** of a **WordPress** site, we can certainly **reduce the attack surface** and **reduce the likelihood of a breach** by:
  - **Disabling WordPress REST API** and **WordPress XML-RPC**, if they are not being used.
  - **Configure** the web server to block requests to specific databases by the title, number, and author of the database.
  - **Preventing exposure** of the **/wp-admin** and **/wp-login.php** to the public internet through **Port 80 HTTP**.
  - For an additional layer of security, we can **block** all traffic from non-specified, **non-Whitelisted IP's** and **disable** the **Apache Server's Directory Listing** option:
    - To do this, we must edit the **/etc/apache2/apache2.conf** file with a text editor like, **nano**
    - We must add the specified **WordPress Directory** that we want to set rules for
    - Lastly, we must **Set to Allow** our known authentic business IP's, and **Set to Deny** any unrecognized IP's attempting to access the **WordPress directories**.
    - **If enabled,** we must remove the "**Indexes**" from the "**Options**" portion of the config file

Hardening Against Weak User Credentials on Target 1

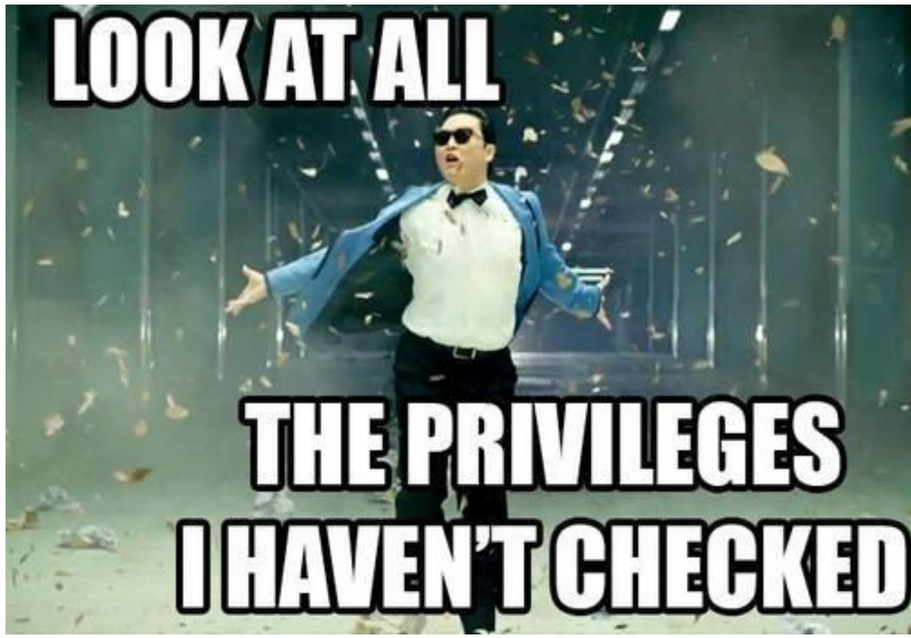
Necessary Steps:

- First, we need to **institute** and **enforce** a strong **password policy** to protect the **confidentiality** of our **data**.
  - For instance, the entire exploit was only possible because **User Michael** has his **password** set to his name, **michael**, which is easily guessable.
- Suggested changes to the credentialing policy include:
  - Creating strong, unique **passwords** (i.e. Prohibiting password **re-use**, increasing **length**, and avoiding **personal information** and/or **common words**.)
  - **Regularly changing** passwords on at least a 3 month basis
  - Implementing **incentives** and **punishments** for those subject to the policy.
- We must implement **Two-Factor Authentication,** which would offer another layer of protection for if an attacker ever did gain access to a password, even after instituting the new policy.
- Lastly, we must **salt** our password **hashes** so as to prevent **Rainbow Table** attacks.

Hardening Against Misconfiguration of User Privileges on Target 1

Necessary Steps:

- First, we need to restrict sudo privileges to prevent unauthorized access and privilege escalation
  - For instance, user “steven” had python sudo access, which allowed for the red team to escalate their privileges.
- Suggested changes
  - To solve this issue, edit the /sudoers file and remove all unnecessary users (such as steven)
  - Require all sudo users to enter their password
- Additional protections
  - Keep all systems and applications updated with the latest security fixes.
  - Regularly delete inactive user accounts.



Hardening Misconfigured ssh Access on Target 1

- Restrict port 22
  - Only open port 22 to admins
  - Close port 22 when not in use
- Verify trusted device
  - Only allow ssh access from whitelisted IP addresses
  - Two-factor authentication
  - Log and monitor ssh connections
- Limit password attempts
  - Lock out users after 5 failed logins for 5 minutes
  - Block users after multiple lockouts
  - Set an alert to go off after multiple lockouts

when you leave the company that you have been working at for years but the ssh key for the main server still works

Implementing Patches

Implementing Patches with Ansible

Overview: Ansible can be used to automate system updates quickly. Here we can see an example of SSH hardening.

- Perform full software patch.
- Add ssh networks to internal **firewalld zone**.
- Parameters tell the module to apply these rules immediately and permanently.

```
---

- hosts: all
  vars:
    allowed_ssh_networks:
      - <Network Range>
      - <VPN Range>

  tasks:
    - name: Perform full patching
      package:
        name: '*'
        state: latest

    - name: Add SSH port to internal zone
      firewalld:
        zone: internal
        service: ssh
        state: enabled
        immediate: yes
        permanent: yes

    - name: Add permitted networks to internal zone
      firewalld:
        zone: internal
        source: "{{ item }}"
        state: enabled
        immediate: yes
        permanent: yes
      with_items: "{{ allowed_ssh_networks }}"

    - name: Drop ssh from the public zone
      firewalld:
        zone: public
        service: ssh
        state: disabled
        immediate: yes
        permanent: yes

  handlers:
    - name: Reload SSH
      service:
        name: sshd
        state: reloaded
```

Implementing Patches with Ansible (continued)

```
#WordPress Configuration

  - name: Download 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 ]
```

- Updating to the latest wordpress version.
- setting permissions for wordpress directory and files.
- Add parts that would back up current wp database incase.