Final Engagement

Attack, Defense & Analysis of a Vulnerable Network

Maria, Karen, Ryan and Ed

Table of Contents

This document contains the following resources:

Network Topology & Critical Vulnerabilities

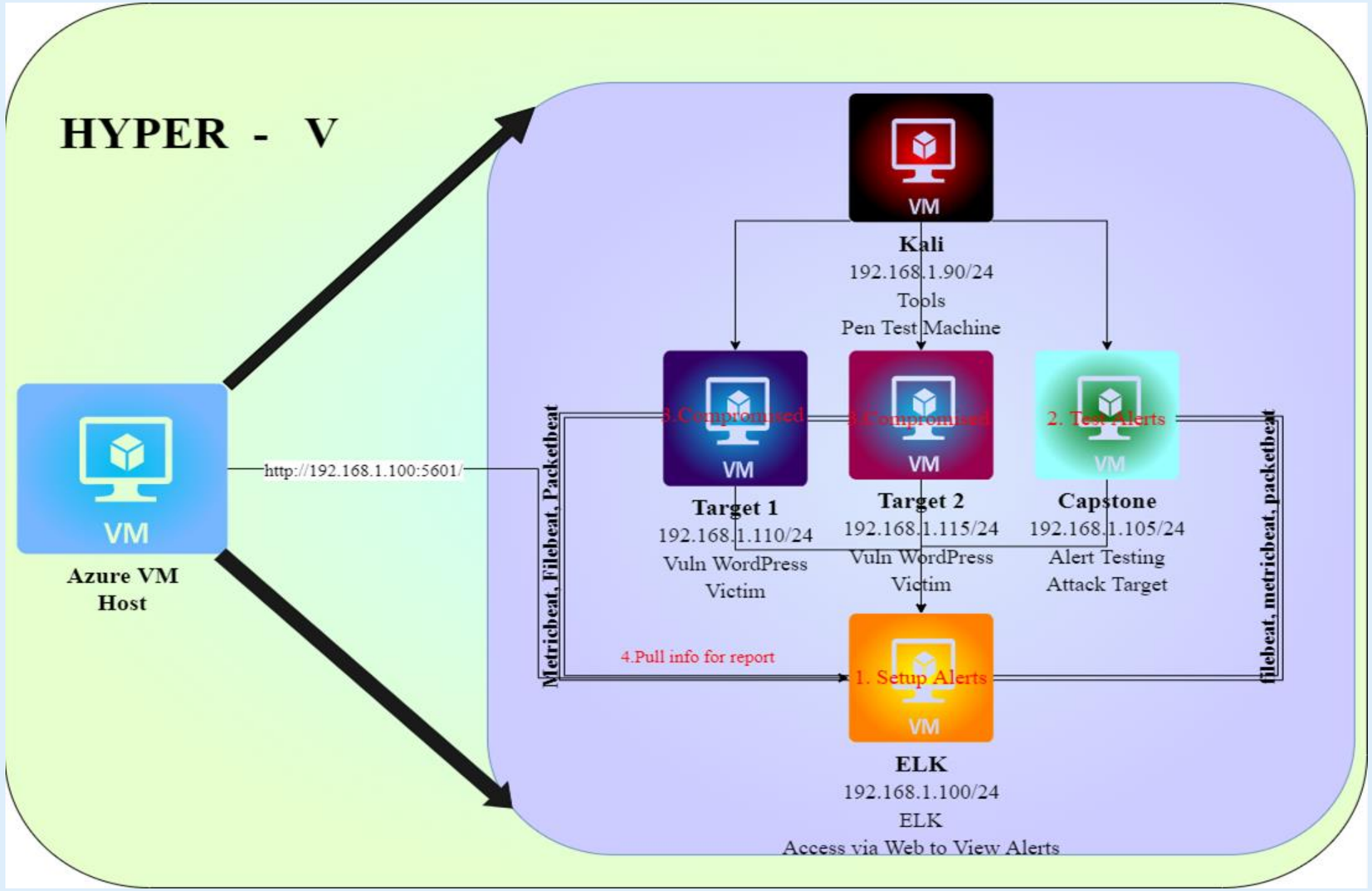
Alerts Implemented

Hardening

Implementing Patches

Network Topology & Critical Vulnerabilities

Network Topology



Network
Address Range:**192.168.1.0/24**
Netmask:**255.255.255.0**
Gateway:**192.168.1.1**

Machines
IPv4:**192.168.1.90**
OS:**Debian**
Hostname:**Kali**

IPv4:**192.168.1.100**
OS:**Ubuntu**
Hostname:**ELK**

IPv4:**192.168.1.105**
OS:**Ubuntu**
Hostname:**Capstone**

IPv4:**192.168.1.110**
OS:**Debian**
Hostname:**Target 1**

IPv4:**192.168.1.115**
OS:**Debian**
Hostname:**Target 2**

Critical Vulnerabilities: Target 1 and Target 2

Our assessment uncovered the following critical vulnerabilities in **Target 1 and Target 2**.

Vulnerability	Description	Impact
Unprotected Ports	Attackers can access system through ports 22 and 80	unauthorized system access; compromise a network;
Weak Passwords	Attackers can guess or easily crack passwords	Unauthorized system access; root escalation; Access/compromise to sensitive information;
Clear Text Credentials	SQL database credentials found in wp-admin file	Access to confidential Data; Compromise data; install malware; elevate privileges
Downrev software including susceptibility to LFI and XSS	Debian 3.16; WordPress software 4.8.19; mysql 5.5.60	Malware installation and data breach

Alerts Implemented

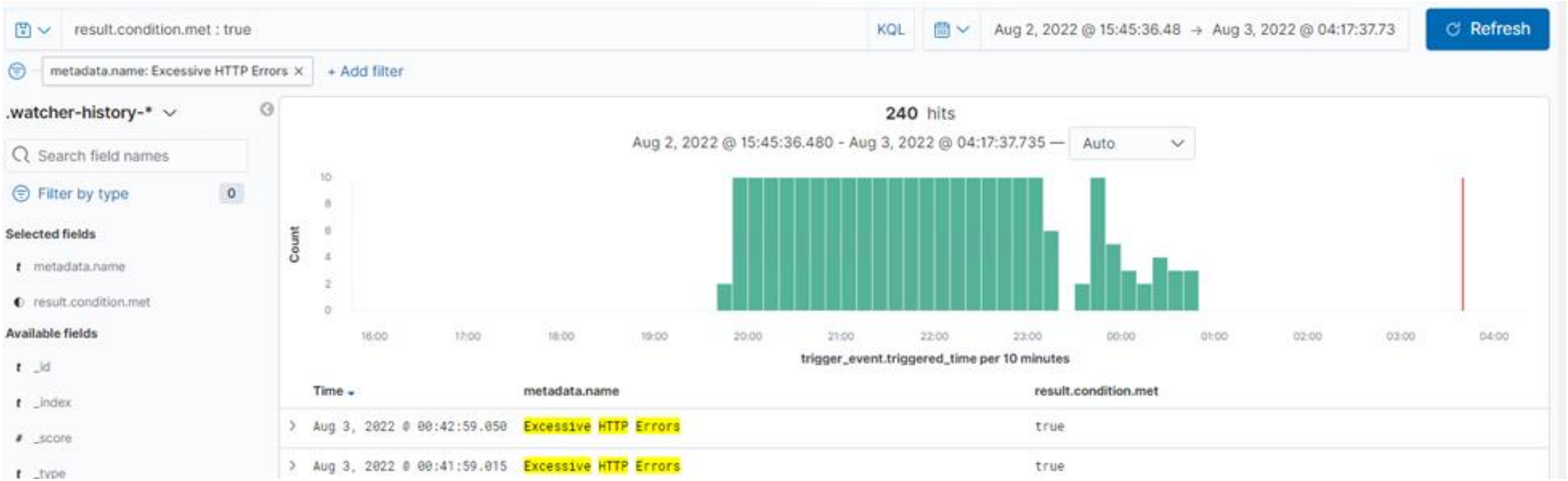
Alert 1 - Excessive HTTP Errors

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

Metric: packetbeat

Threshold: 400

Vulnerability Mitigated: This caught the hydra brute force attack on Target1



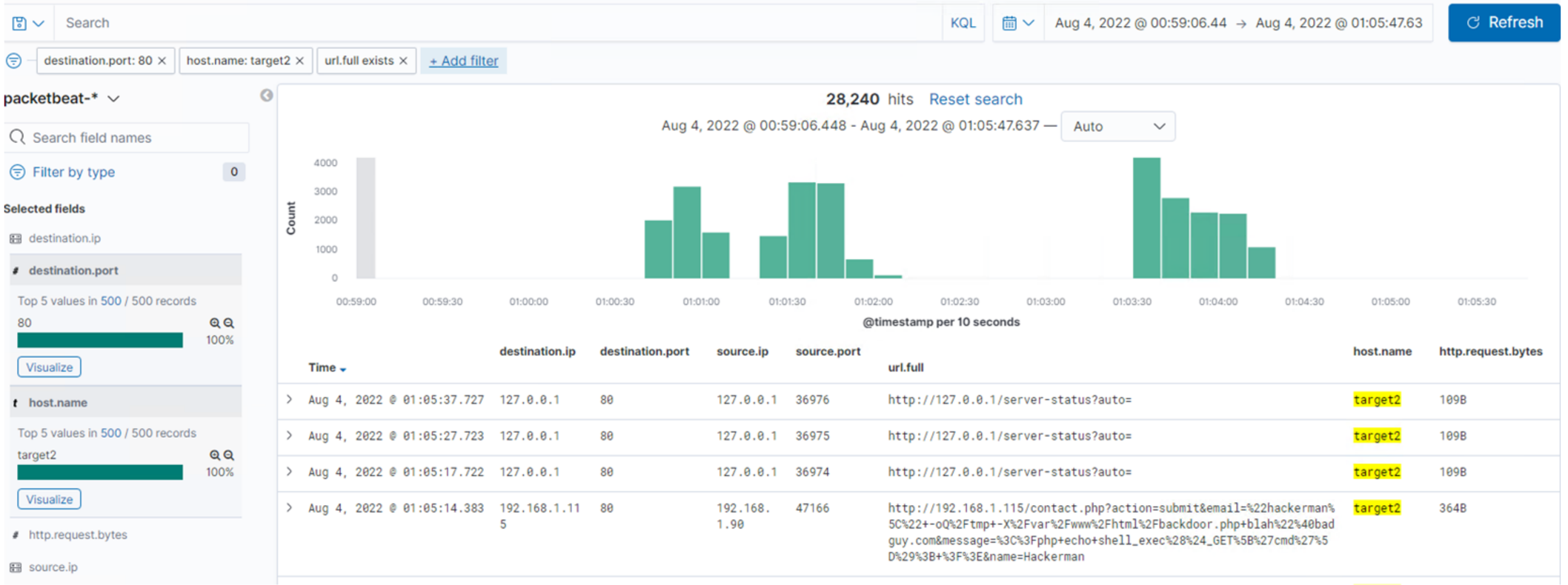
Alert 2 - HTTP Request Size Monitor

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

Metric: packetbeat

Threshold: 3500 bytes

Vulnerability Mitigated: This caught the upload of the exploit.sh malware to Target2



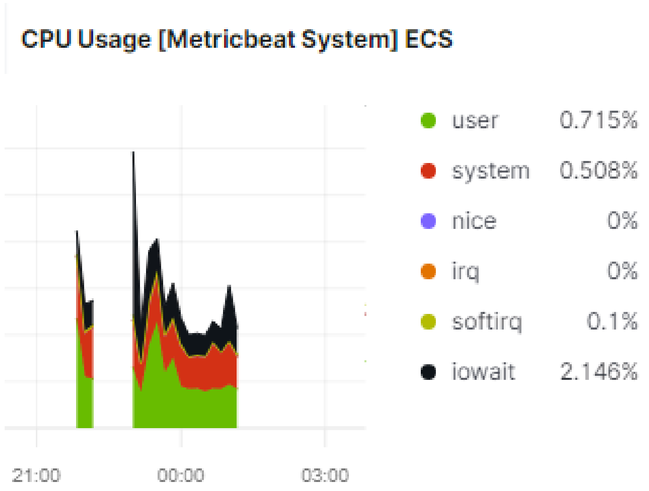
Alert 3 - CPU Usage Monitor

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

Metric: metricbeat

Threshold: CPU load > 50%

Vulnerability Mitigated: This would trigger when the web server's CPU usage is high, potentially catching a DDoS attack or other malware.



Hardening

Hardening Against Unprotected Ports

- Restrict ssh access to allowed users
  - not allowing a remote user to elevate to root will expose less data in a breach
    - nano /etc/ssh/sshd_config
      - change #PermitRootLogin from "yes" to "no"
  - modify user access to only allow users wanted to ssh
    - nano /etc/ssh/sshd_config
      - locate user michael and remove

Hardening Against Weak Passwords

- Add password policy
- A password policy will force users to pick passwords that are hard to guess. Also by adding an expiration date we limit the window an exposed password can be vulnerable
  - Install password checking library
    - sudo apt install libpam-pwquality
  - Modify common-password file
    - nano /etc/pam.d/common-password
    - goto line with password [success=2 default=ignore] pam_unix.so obscure sha512
    - add the following to the end to change policy
      - minlen=10 (makes password length 10)
      - ucredit=-1 uppercase letter required
      - dcredit=-1 lowercase letter required
      - ocredit=-1 other character required
  - Modify password expiration date
    - sudo nano /etc/login.defs
    - modify the following options to your business needs
      - PASS_MAX_DAYS 100
      - PASS_MIN_DAYS 0
      - PASS_WARN_AGE 7

Hardening Against Clear Text Credentials

- Storing the sql passwords for wordpress in a secret manager will prevent an attacker from reading the credentials from a config file
  - download and install https://www.vaultproject.io/docs/what-is-vault/
  - follow quickstart instructions
- Creating a sql user with only read access to tables required by wordpress site prevents an attacker with credentials from viewing user table
  - login to root
    - mysql -u root -p
  - create read-only user
    - CREATE USER ‘$user@’127.0.0.1' IDENTIFIED BY ‘$password‘;
  - give access to only tables wanted
    - GRANT SELECT, SHOW VIEW ON $database_name.<table name/s> TO $user@’127.0.0.1' IDENTIFIED BY ‘$password‘ REQUIRE SSL;
    - FLUSH PRIVILEGES;

Hardening Against Downrev Software

- Updating the wordpress installation with prevent this vulnerability
  - Backup your existing database and WordPress directory
    - mkdir -p /backup/21072016
    - $ mysqldump -u user -p wp_database > /backup/21072016/wp_database.sql
    - $ tar -zcvf /backup/21072016/app.tar.gz /var/www/sites/<wp directory>
  - Download the latest version from here
    - $ cd /tmp
    - $ wget http://wordpress.org/latest.zip
    - $ unzip latest.zip
  - Execute below commands
    - $ cd /var/www/sites/<wp directory>/app
    - $ cp -avr /tmp/wordpress/* .
    - $ rm -rf /tmp/wordpress /tmp/latest.zip
  - Open a browser and run upgrade script
    - ex. http://<target domain>/wp-admin/upgrade.php

Hardening Against Downrev Software (cont.)

- Update Debian to prevent attack using this and other vulnerabilities to break into machine.
  - Get latest source
    - http://httpredir.debian.org/debian stable main contrib (geographic redirect for closest mirror)
  - update aptitude
    - aptitude update
  - update debian
    - aptitude full-upgrade
- Update mysql to prevent attack using this and other vulnerabilities to break into machine.
  - configure slow shutdown
    - mysql -u root -p --execute="SET GLOBAL innodb_fast_shutdown=0"
  - do shutdown
    - mysqladmin -u root -p shutdown
  - download version (https://dev.mysql.com/downloads/)
  - Start the MySQL server, using the existing data directory. For example:
    - mysqld_safe --user=mysql --datadir=/path/to/existing-datadir &
  - Run mysql_upgrade. For example:
    - mysql_upgrade -u root -p
  - Shut down and restart the MySQL server to ensure that any changes made to the system tables take effect. For example:
    - mysqladmin -u root -p shutdown

Implementing Patches

Implementing Patches with Ansible

Playbook Overview

- **Strategize how we can automate patches and updates.**
- **Provide proper validation on all patches and updates.**
- **Notifications on recent updates**

How to approach our Ansible Playbook

Automate patches and updates: Anatomy on our playbook

Our Red Team has advised our team to apply an automated function to update patches and services.

To prioritize the Ansible playbook, we must decide on the essential working functionality :

1. Initializing the ansible-galaxy role list: This will allow you to have reusable automation components by grouping and encapsulation replated automation: configuration files, templates, and tasks.
2. Our plan is to develop a playbook with the following tasks:

Perform OS Patching / Running Patching Pre-check /Show Services updated and successfully installed.

1. Our goal in this outline for the Ansible Playbook will provide us the flexibility to ADD or REMOVE services as needed.

Providing proper validation on all patches and updates.

When constructing the ansible playbook, the configuration will be documented for (Standard Operating Procedure) SOP. The configuration file will include - Automate patching, updating WordPress and provide the STATE of each service.

```
- - -
- name: Perform OS Patching
  hosts: “{{ Servers }}”
  become: yes
  vars:
    service_list:
                    -ngix
                    -https
                    -mysql
  - name:  Patching PreCheck
    include_role:
     name:  linux_os_patching
      task_from: PreCheck.yml
```

Continue: **Providing proper validation on all patches and updates.**

The most essential source in creating the PreCheck.yml

The updates to WordPress . The PreCheck.yml will remotely update to the latest version.

```
- name: Install WordPress
  remote_user: ubuntu
  hosts: all
  become: true
  become_user: root
  gather_facts: true
  tasks:
    - name: Download and Extract WordPress
      unarchive:
        src: https://wordpress.org/latest.tar.gz
        dest: /var/www/
        remote_src: yes
```

Notifications on recent updates

```
 name: Running Patching Pre-check
 debug:
   msg: "Pre-Check started"
- name: Get Service List
  service_facts:
- name: Show service
  debug:
                msg: "{{
ansible_facts.services }}"
- name: Filter Running Services
  set_facts:
                running_services:
"{{ ansible_facts | json_quert('services.* |[?state == 'running'].name'}
}}"
- name: Show Running Services
  debug:
                msg: "{{
running_services }}"
- name: Check service if running
  set_fact:
                any_service_running:
true
  when: running_services is
search(items)
  with_items:
                - "{{ service_list
}}"
```