Final Engagement

Attack, Defense & Analysis of a Vulnerable Network

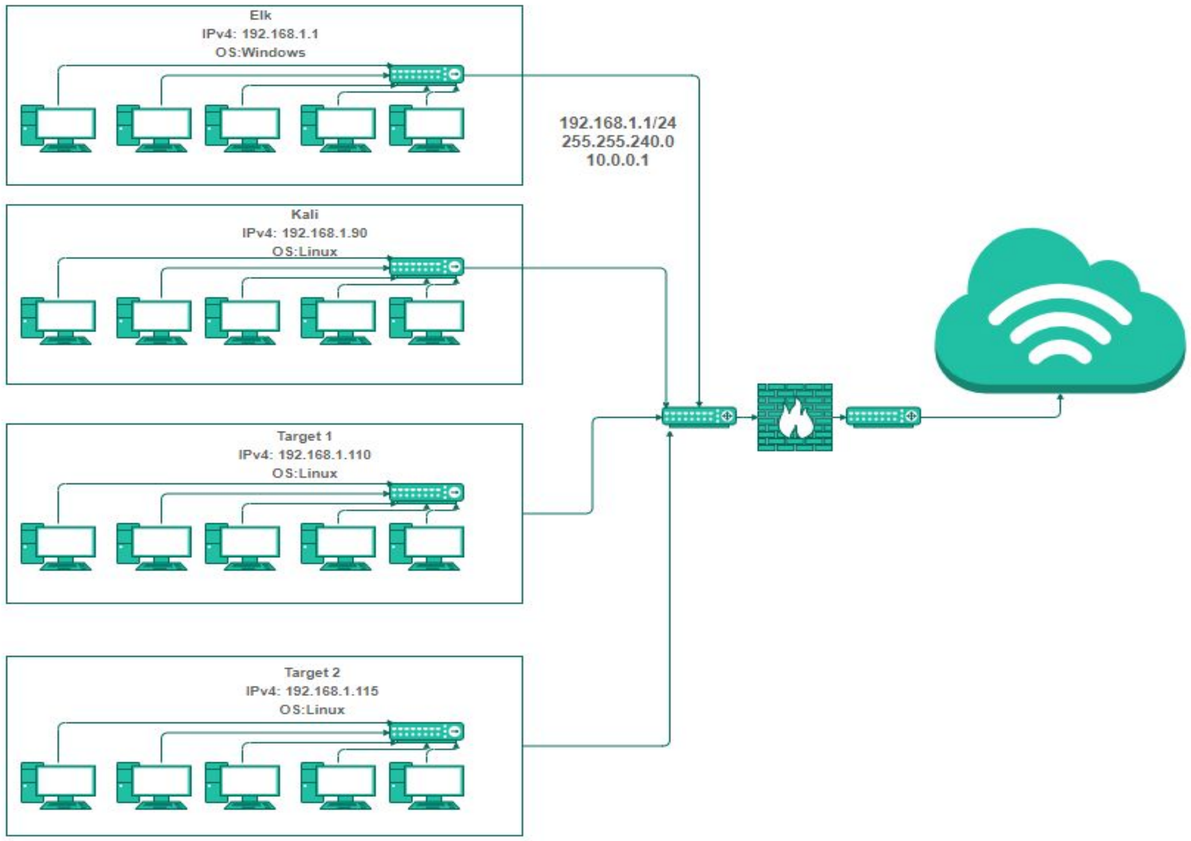
Team Name: Jurassic Cyberkings
(*chomp*)

Table of Contents

This document contains the following resources:

Network Topology & Critical Vulnerabilities

Network Topology



Network
Address Range: 192.168.1.1/225
Netmask: 255.255.240.0
Gateway: 10.0.0.1

Machines
IPv4:192.168.1.1
OS:Windows
Hostname:ELK

IPv4:192.168.1.90
OS:Linux
Hostname:Kali

IPv4:192.168.1.110
OS:Linux
Hostname:Target 1

IPv4:192.168.1.115
OS:Linux
Hostname:Target 2

Critical Vulnerabilities: Target 1

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

Vulnerability	Description	Impact
SSH	22/TCP OpenSSH	High
HTTP	80/TCP Apache httpd 2.4.10	High
netbios-ssn hey	139/TCP Samba smbd	Medium

Critical Vulnerabilities: Target 2

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

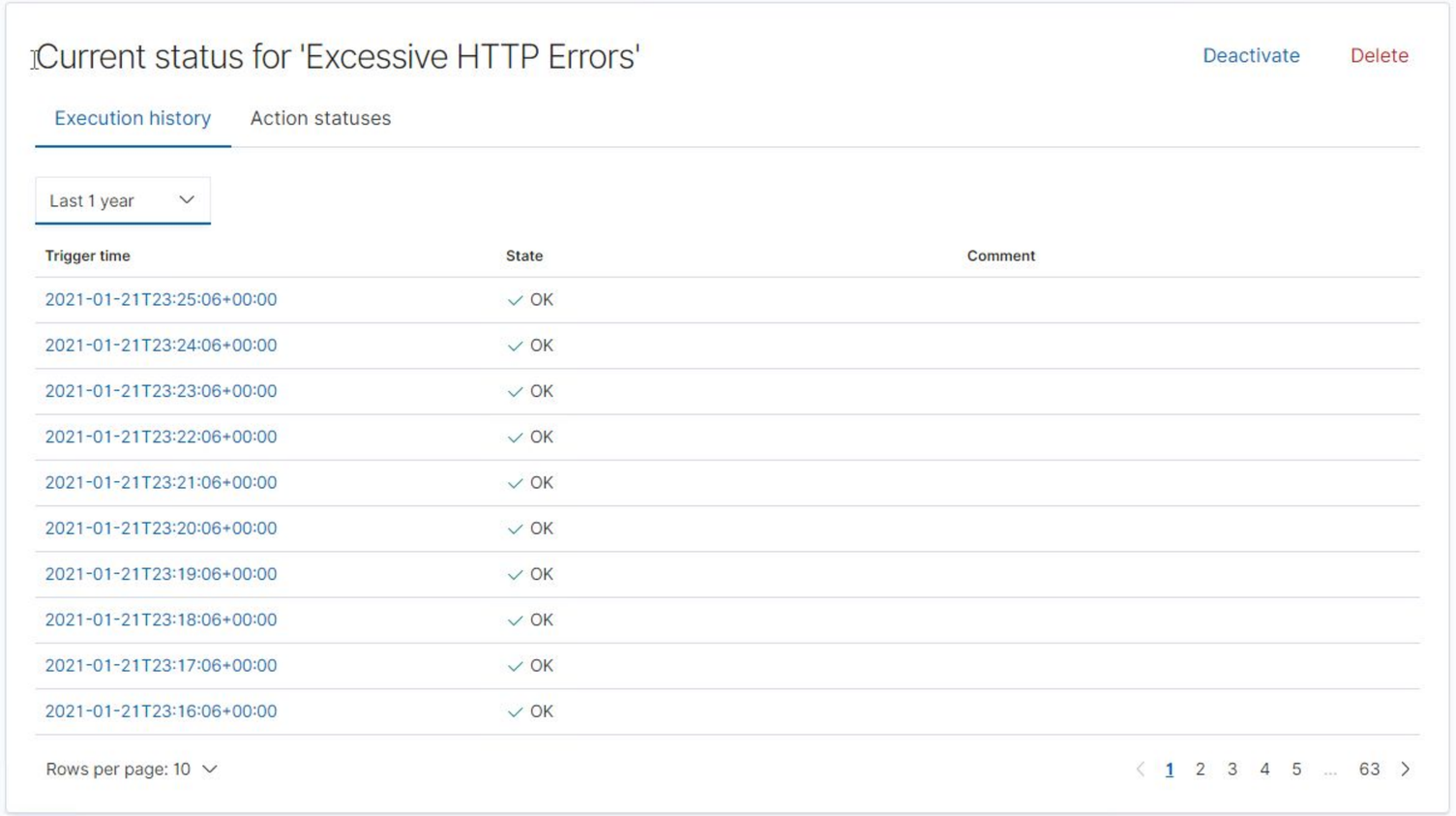
Vulnerability	Description	Impact
SSH	22/TCP OpenSSH	High
HTTP	80/TCP Apache httpd 2.4.10	High
Contact.php	Backdoor	High

Alerts Implemented

Excessive HTTP Errors

The Alert Summarized:

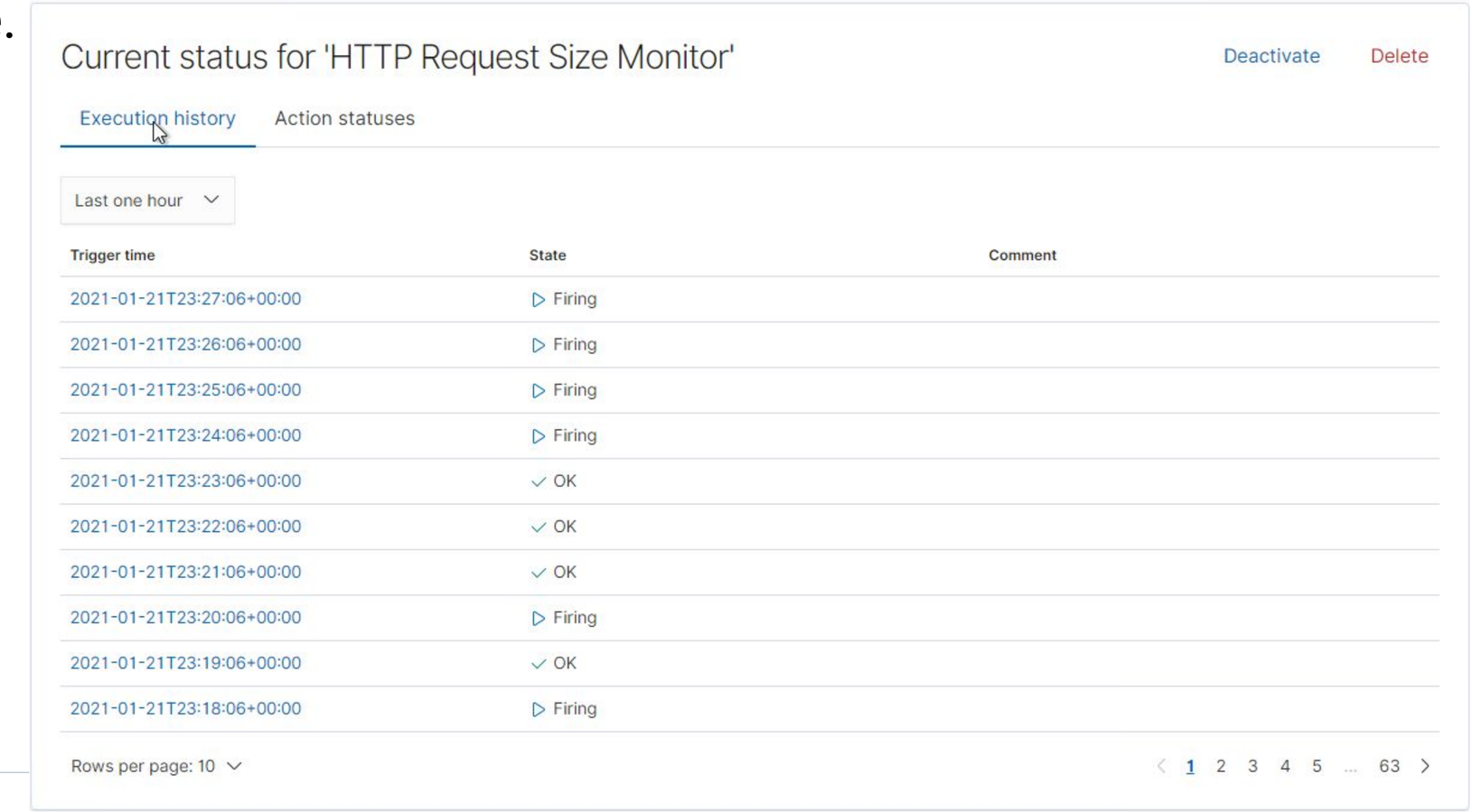
- This alert is monitoring the ‘http.response.status_code’ metric.
- The alert fires when the count is grouped over the top 5 ‘http.response.status_code’ is above 400 for the last 5 minutes.



HTTP Request Size Monitor

The Alert Summarized:

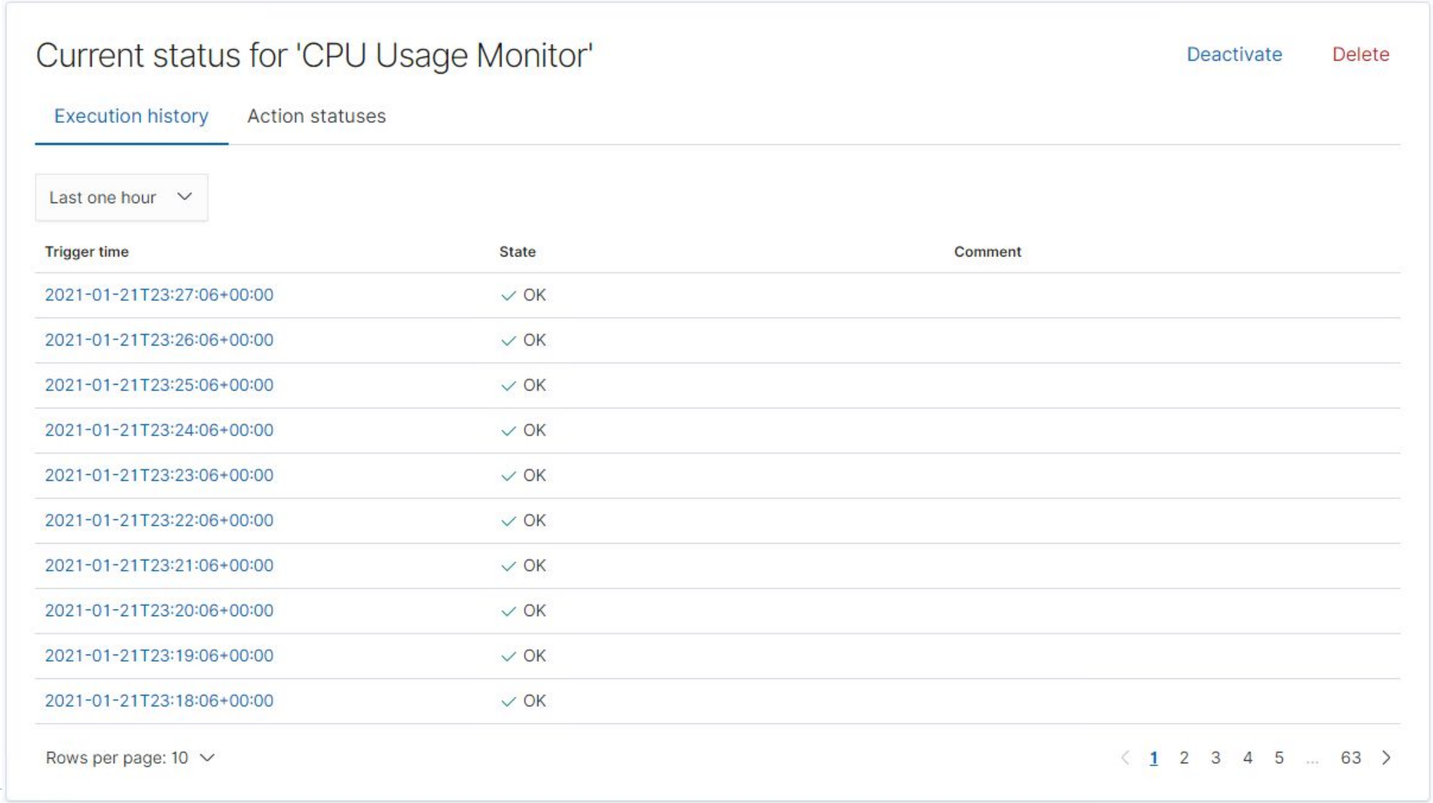
- This alert monitors the metric of ‘http.request.bytes’.
- The alert fires when the sum of the ‘http.request.bytes’ of all documents is above 3500 for the last 1 minute.



CPU Usage Monitor

The Alert Summarized:

- This alert monitors the metric of ‘system.process.cpu.total.pct’.
- The alert fires when the maximum of ‘system.process.cpu.total.pct’ of all documents is above 0.5 for the last 5 minutes.



Hardening

Hardening Against Brute Force Attacks on Target 1 & 2

Patch #1:

Invalid credentials lock-out after 30 failed attempts in 1 minute.

Why: This prevents excessive login attempts, i.e. brute forcing login credentials.

Install: To implement account lockout/timeout system, configure the ansible server to secure passwords, automate updates, basic intrusion detection, public key authorization, firewall settings, and to monitor logs.

Patch #2:

Implement two factor authentication.

Why: This makes sure that a brute force attack cannot go past the first attempt as it will not have access to the other authenticators for the login.

Install: Change the security settings for logins to use two factor authentications. Once the user tries to log in, the web site will prompt the user for the secure code that is most often sent via SMS text or email, unless the user chooses a different authentication technique.

Hardening Against DOS Attacks on Target 1&2

Patch #1:

Blacklisting IP addresses after a threshold for a week.

Why: Blacklisting Ip addresses will stop any attack from continuing successfully, yet allow regular/legitimate traffic to access the web servers.

Install: Implement via network/firewall settings, or configure ansible with the threshold and action of blocking.

Patch #2:

Install a Load Balancer on the Network

Why: Installing a load balancer will help buffer the traffic burden placed on each server and optimize network traffic and processing. This will keep accessibility maintained at all times.

Install: Using both Hardware (device included in the network) and Software (through applications) are options.

Hardening Against Excessive CPU Usage on Target 1&2

Patch #1:

Create several alerts at different thresholds of CPU Usage.

Why: This allows the monitoring of how much core activity is being utilized at any moment.

Install: there are applications (based on user/system preference) and upgrades for the SIEM suites that allow for creating alerts that will trigger alerts when CPU thresholds are surpassed.

Patch #2:

Set limits to Processing Power

Why: This can limit how much processing power each core/application can use, which would keep the CPU from getting appropriated by a single process.

Install: The Task Manager settings can be configured to set limits on CPU usage.

Hardening Against Remote Access on Target 1&2

Patch #1:

Whitelist IP Addresses

Why: This way only users verified by administrators can use remote access as identified by their ip addresses.

Install: Configure the Firewall/Network settings to make the whitelist of ip addresses. Keep this list up to date immediately upon changes of ip addresses or users occur.

Patch #2:

Implement Least Privilege Principles.

Why: This maintains that users only have privilege to access what they need to do their tasks.

Install: Maintain the user and group authorizations regularly, asap if a user is no longer in the group/company.

Hardening Against Remote Access on Target 2

Patch #1

Update the Kernel

Why: This is standard procedure for ensuring updates against recently discovered bugs or threats.

Install: Use the following command: sudo apt-get upgrade kernel

Patch #2

Install Canonical Livepatch

Why: This will update software.

Install: Use the following command(s): sudo snap install canonical-livepatch && sudo canonical-livepatch enable

Patch #3

Install KernelCare

Why: This is an “install and forget” solution. It automagically downloads and applies new kernel security patches, without rebooting the server.

Install: Use the following command(s): wget -qq -0 -- https://kernelcare/installer | bashsudo /usr/bin/kcarectl -- register <your key>

Implementing Patches

Implementing Patches with Ansible

Playbook Overview

- Prevent brute force attacks
  - > lock out after 30 failed attempts
  - > require two-factor authentication
- Prevent DOS attacks
  - > block IP addresses after lockouts
  - > balance your load
- Prevent excess CPU usage
  - > *alerting is not a function of Ansible*
- Prevent unauthorized remote access
  - > whitelist authorized IP addresses

Implementing Ansible: Prevent Brute Force Attacks

Lockout after 30 failed login attempts

Use the pam_faillock module (Pluggable Authentication Modules)

/etc/pam.d/system-auth

/etc/pam.d/password-auth

Require two-factor authentication

Duo Security is a free program for easy multifactor authentication on SSH logins

Implementing Ansible: Prevent Brute Force Attacks

```
nano Security-Playbook.yml

---
  - name: Server Security Playbook
    hosts: webservers
    become: true
    tasks:

      - name: Lockout After Failed Logins
          community.general.pamd
            name: system-auth
            type: auth
            control: required
            module_path: pam_faillock.so
            module argument: 'fail_interval=30'
            state: args_present

      - name: Require Two-Factor Authentication
          community.general.pamd
            name: common-auth
            new_type: auth
            new_control: '[success=1 default=ignore]'
            new_module_path: '/lib64/security/pam_duo.so'
            state: after
            type: auth
            module_path: pam_sss.so
            control: 'requisite'
```

**Be sure to install the community.general collection from Ansible.*

```
ansible-galaxy collection install community.general
```

Implementing Ansible: Prevent DOS Attacks

Block IP Addresses

```
- name: Block specific IP
  ansible.builtin.iptables:
    chain: INPUT
    source: 8.8.8.8
    jump: DROP
  become: yes
```

Set it to trigger when brute force lock out happens

```
# Sometimes it is desirable to let the sender know when traffic is
# being denied, rather than simply ignoring it. In these cases, use
# reject instead of deny. In addition, log rejected connections:
- community.general.ufw:
    rule: reject
    port: auth
    log: yes
```

Implementing Ansible: Prevent DOS Attacks

Balance your load

```
- name: Create a load balancer
  community.general.oneandone_load_balancer:
    auth_token: oneandone_private_api_key
    name: ansible load balancer
    description: Testing creation of load balancer with ansible
    health_check_test: TCP
    health_check_interval: 40
    persistence: true
    persistence_time: 1200
    method: ROUND_ROBIN
    datacenter: US
    rules:
     -
       protocol: TCP
       port_balancer: 80
       port_server: 80
       source: 0.0.0.0
    wait: true
    wait_timeout: 500
```

Implementing Ansible: Prevent Unauthorized Remote Access

Whitelist allowable IP Addresses

```
- name: Match on IP ranges
  ansible.builtin.iptables:
    chain: FORWARD
    src_range: 192.168.1.100-192.168.1.199
    dst_range: 10.0.0.1-10.0.0.50
    jump: ACCEPT
```

Implementing Ansible: Prevent Unauthorized Remote Access

Keep kernel updated

```
- name: update the system
  yum:
    name: "*"
    state: latest
```

```
- name: Install the latest version of Apache
  yum:
    name: httpd
    state: latest
```

**Be sure to install the yum collection from Ansible.*

```
ansible-galaxy collection install yum
```

```
- name: restart system to reboot to newest kernel
  shell: "sleep 5 && reboot"
  async: 1
  poll: 0

- name: wait for 10 seconds
  pause:
    seconds: 10

- name: wait for the system to reboot
  wait_for_connection:
    connect_timeout: 20
    sleep: 5
    delay: 5
    timeout: 60
```

The End
(*yum*)