Capstone Engagement

Attack, Defense & Analysis of a Vulnerable Network

Table of Contents

This document contains the following resources:

Network Topology & Critical Vulnerabilities

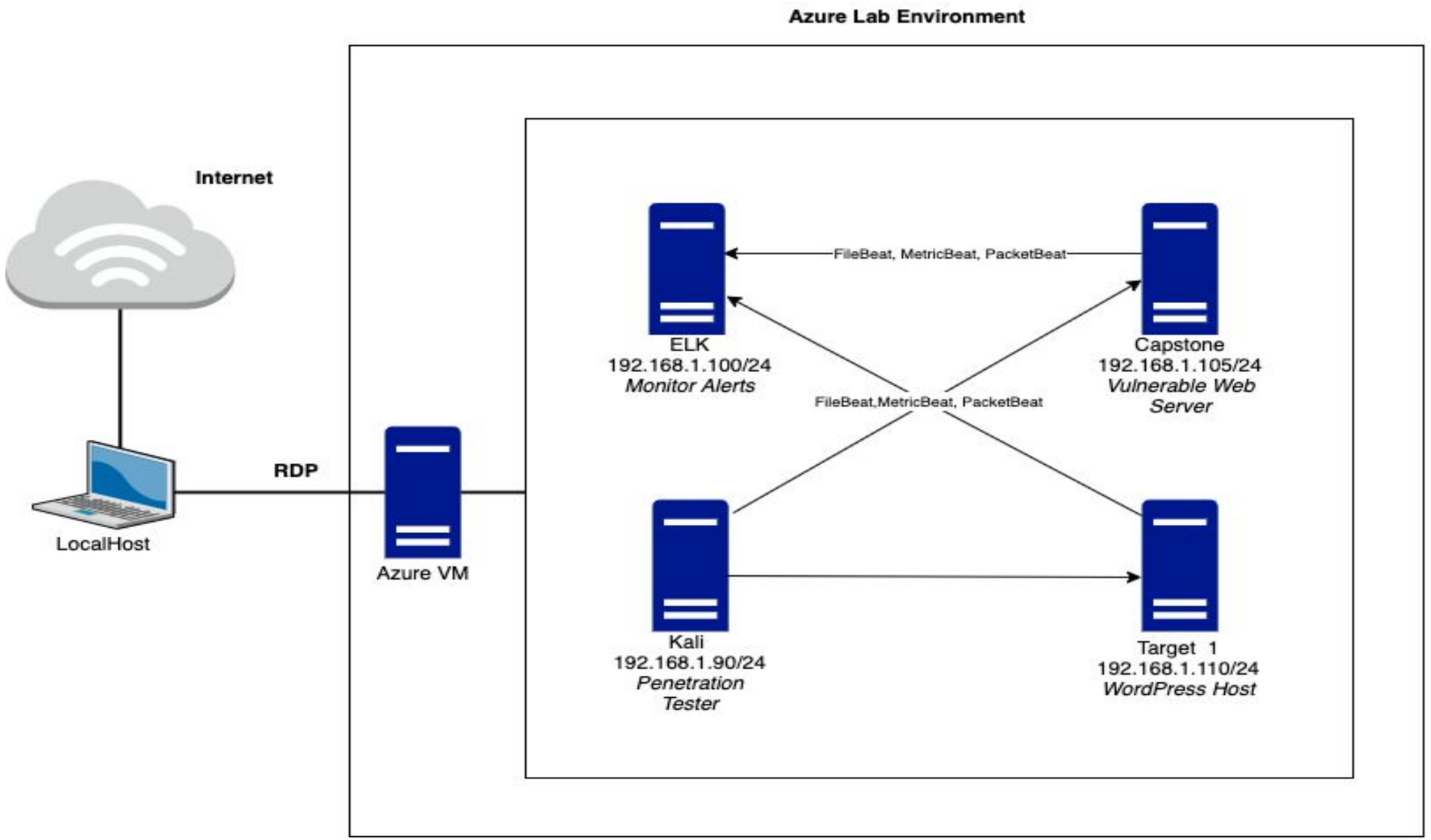
Alerts Implemented

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: Linux
Hostname: Kali

IPv4: 192.168.1.100
OS: Linux
Hostname: ELK

IPv4: 192.168.1.105
OS: Linux
Hostname: Capstone

IPv4: 192.168.1.110
OS: Linux
Hostname: Target 1

Critical Vulnerabilities: Target 1

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

Vulnerability	Description	Impact
Enumerate Wordpress	To acquire users, Wordpress version, and other useful information	The aim is to gain useful information about the target. This is not a direct threat to systems or data.
Broken Access Control Root password to the MYSQL database	Root password was in plaintext in wp.config file.	Hackers just need to access the wp.config file and gain access to the database
Privilege Escalation with Python	Limited users can acquire root privileges with Python	Gaining root access gives hacker access to data, create backdoor, and plenty of malicious acts.

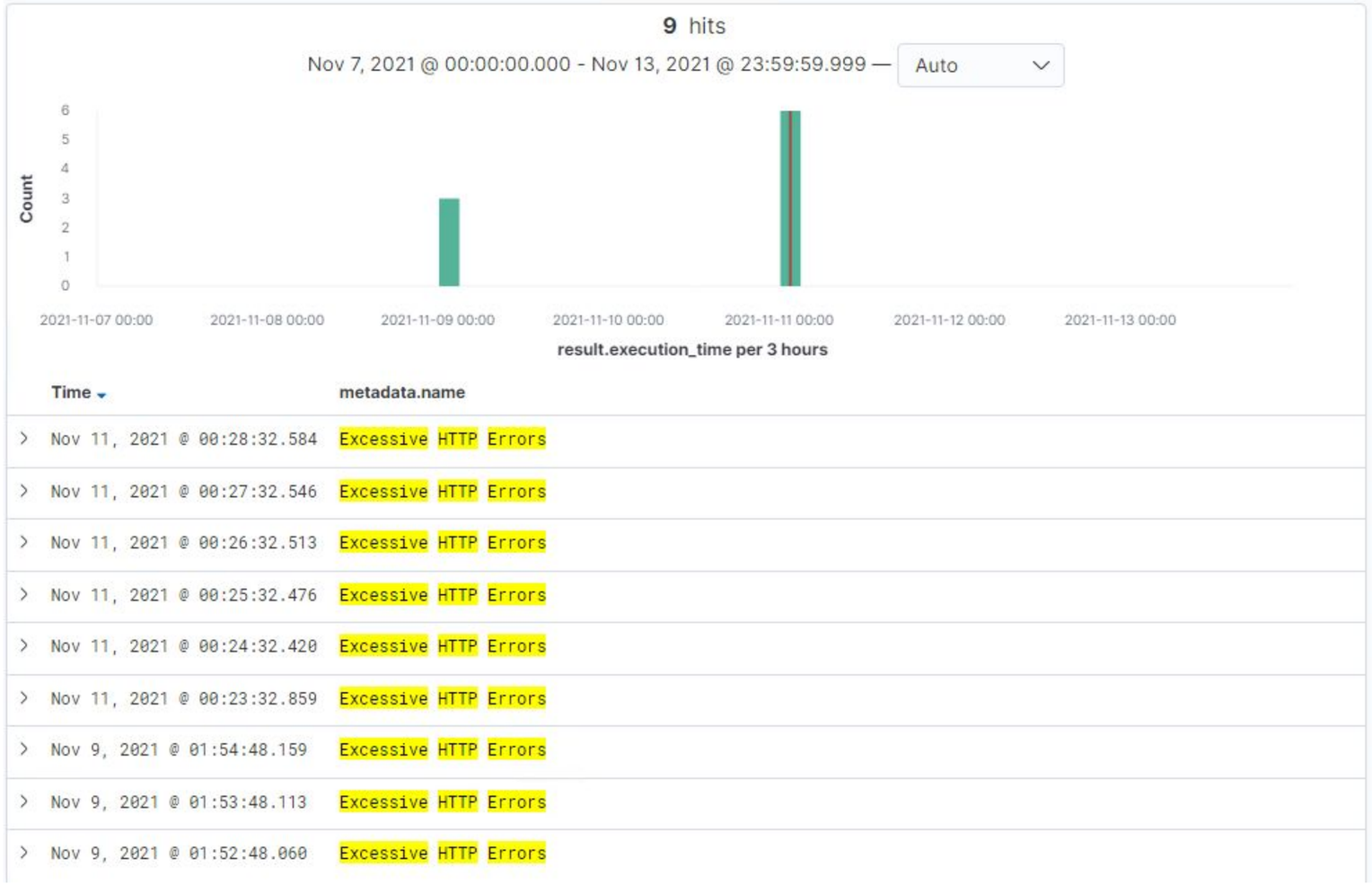
Alerts Implemented

Excessive HTTP Errors

This alert monitors packetbeat and has a threshold of 400 per 5 minutes.

Its purpose is to alert when there is unauthorized access, which could alert of brute force attacks.

The alert worked, although there were only 9 hits which mostly occurred during vulnerability scanning. Such an alert can be useful in determining if someone is attempting to conduct reconnaissance.



HTTP Request Size Monitor

This alert monitors packetbeat and has a threshold of 3500 per minute.

Its purpose is to ensure websites remain up and running. For example, this can alert of a DOS attack.

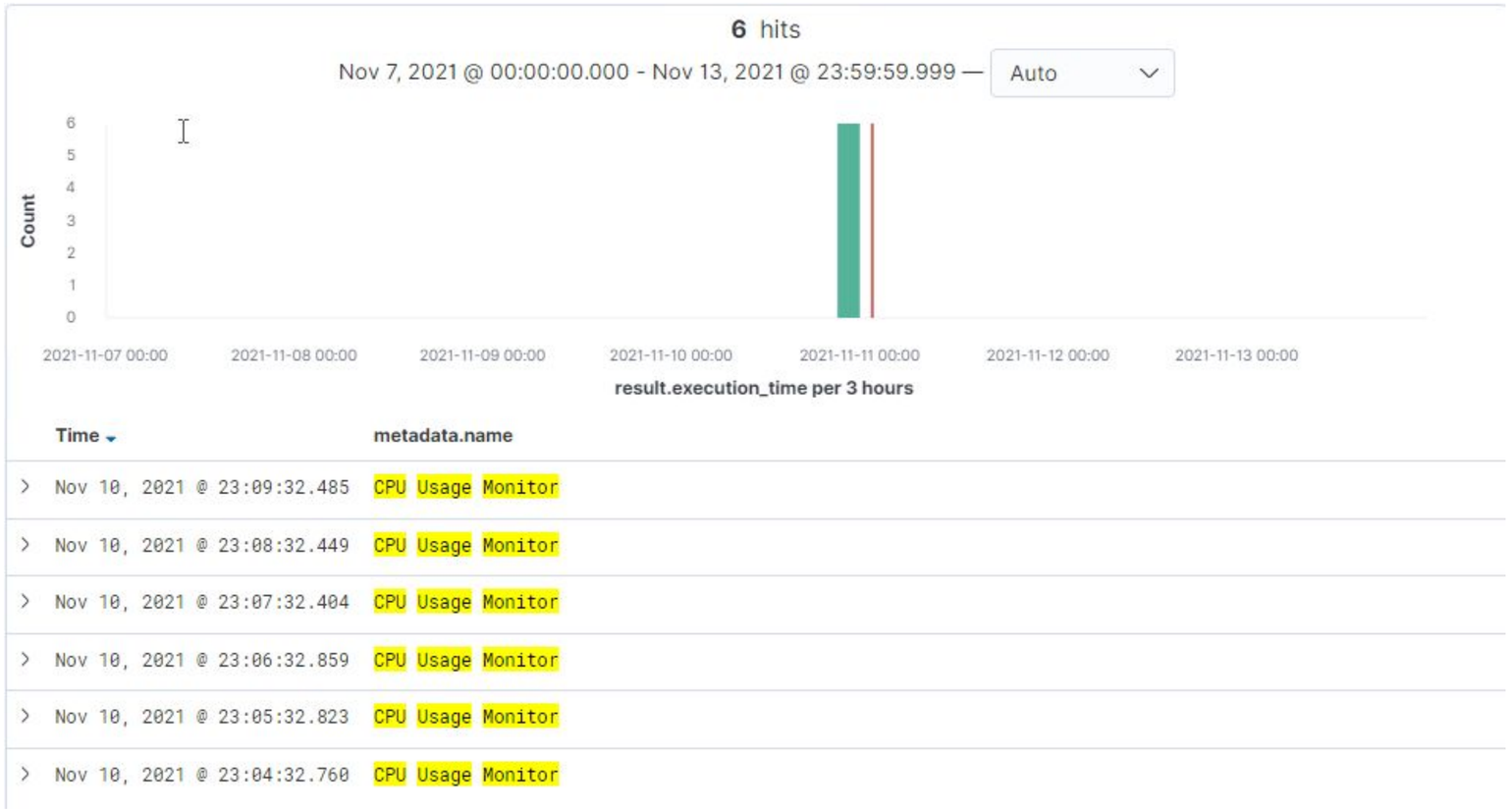
This alert was highly reliable and comprised 93.9% of all alerts.

CPU Usage Monitor

This alert monitors metricbeat and has a threshold of .5 per 5 minutes.

It is used to detect abnormally high CPU usage, which can be an indicator of malicious activity.

This report worked, although there were only 6 hits, which all occurred during vulnerability scanning. Such an alert can be useful in determining if someone is attempting to conduct reconnaissance.



Hardening

Hardening Against Excessive HTTP Errors on Target 1

We propose the following as viable solutions to better protect Target 1:

- **IPS.** Configure an IPS that would block traffic from suspicious IPs
- **Default Deny.** Any ports that are not business critical should be disabled
  - In cases where ports are required to be open, implement steps to filter traffic
- **Account Management.** Have a lockout policy for failed login attempts
  - **Example.** Accounts will automatically be locked if there are five failed attempts in less than three minutes

Hardening Against HTTP Request Size on Target 1

We propose the following as viable solutions to better protect Target 1:

- **Harden Network Perimeter.** Inbound firewall rules to reduce the likelihood of ICMP flood attacks
- **Increase Resilience.** Using a load balancer and failover servers to increase system resiliency by diverting malicious HTTP traffic away from critical infrastructure (sinkholes).
- **Draft Crisis Plan.** Have a documented Business Continuity/Disaster Recovery plan in place for DDoS attacks. This should include circuit diagrams with appropriate telecom and ISP contacts.

Hardening Against CPU Usage on Target 1

We propose the following as viable solutions to better protect Target 1:

- **IDS (Intrusion Detection System).** Have sensors generate alerts in response to a host machine's CPU usage exceeding a set threshold.
  - Set threshold to 50% CPU usage for five minutes
  - **WHEN max() OF system.process.cpu.total.pct OVER all documents IS ABOVE 0.5 FOR THE LAST 5 minutes**

Hardening Against Weak Passwords on Target 1

We propose the following as viable solutions to better protect Target 1:

- **Implement Password Best Practices.**
  - Set Minimum password length to at least a value of 8
  - Enforce password history between 1 - 24
  - Set Maximum password age to expire passwords between 60 and 90 days
  - Configure Minimum password age to to one day
  - The Passwords must meet complexity requirements:
    - Passwords may not contain the user's account name or full name
    - Password contains 3 of the following: uppercase characters, lowercase characters, base 10 digits, non-alphanumeric characters, or unicode characters.
- **Navigate to /etc/pam.d/ and edit the common-password file**
  - minlen=8 (minimum length)
  - dcredit=-1 (require one lower case character)
  - ucredit=-1 (require one upper case character)

Hardening Wordpress on Target 1

- **Reduce wpscan surface area.**
  - Disable your WordPress RSS feed
  - Block Access to xmlrpc.php file

```
location ^~ /xmlrpc.php {
    deny all;
    error_page 403 =404 / ;
}
```

  - Block access to install.php and upgrade.php
  - Remove version query parameters from all enqueued CSS and JS files
- **Deny access.** Adding this code to the htaccess file will deny access to wp.config file.

```
<files wp-config.php>
order allow, deny
deny from all
</files>
```

Hardening Against Privilege Escalation on Target 1

We propose the following as viable solutions to better protect Target 1:

- **Enforce Least Privilege.** Users and groups only have access to what they need:
  - Files, folders, and directories that are not related to a user's role should not be accessible or viewable.
  - Privileged users should not be able to increase their own network access.
  - Implement narrow focused admin accounts: desktop-admin, server-admin, o365-admin, etc.
- **Secure Remote Access.** Implement MFA and VPN to reduce the potential that login attempts are compromised.
- **Password Management.** Implement a solution that vaults privileged user credentials as well enforces a check-in & check-out policy for said credentials.

Implementing Patches

Implementing Patches with Ansible

Playbook Overview

Ansible script to patch WordPress Enumeration - adding this code to the function.php file can block enumeration

```
if (!is_admin()) {
// default URL format
 if (preg_match('/author=([0-9]*)/i' ' , $_SERVER['QUERY_STRING'']))
 die(); add_filter("redirect_canonical' ', 'shapeSpace_check_enum',
 0, 2);
  }
 function shapeSpace_check_enum($redirect, $request) {
  // permalink URL format
  if (preg_match('/\?author=([0-9]*)(\/*)/i' , $request)) die();
   else return $redirect;
   }
```

https://www.getastra.com/blog/cms/wordpress-security/stop-user-enumeration/