Final Engagement

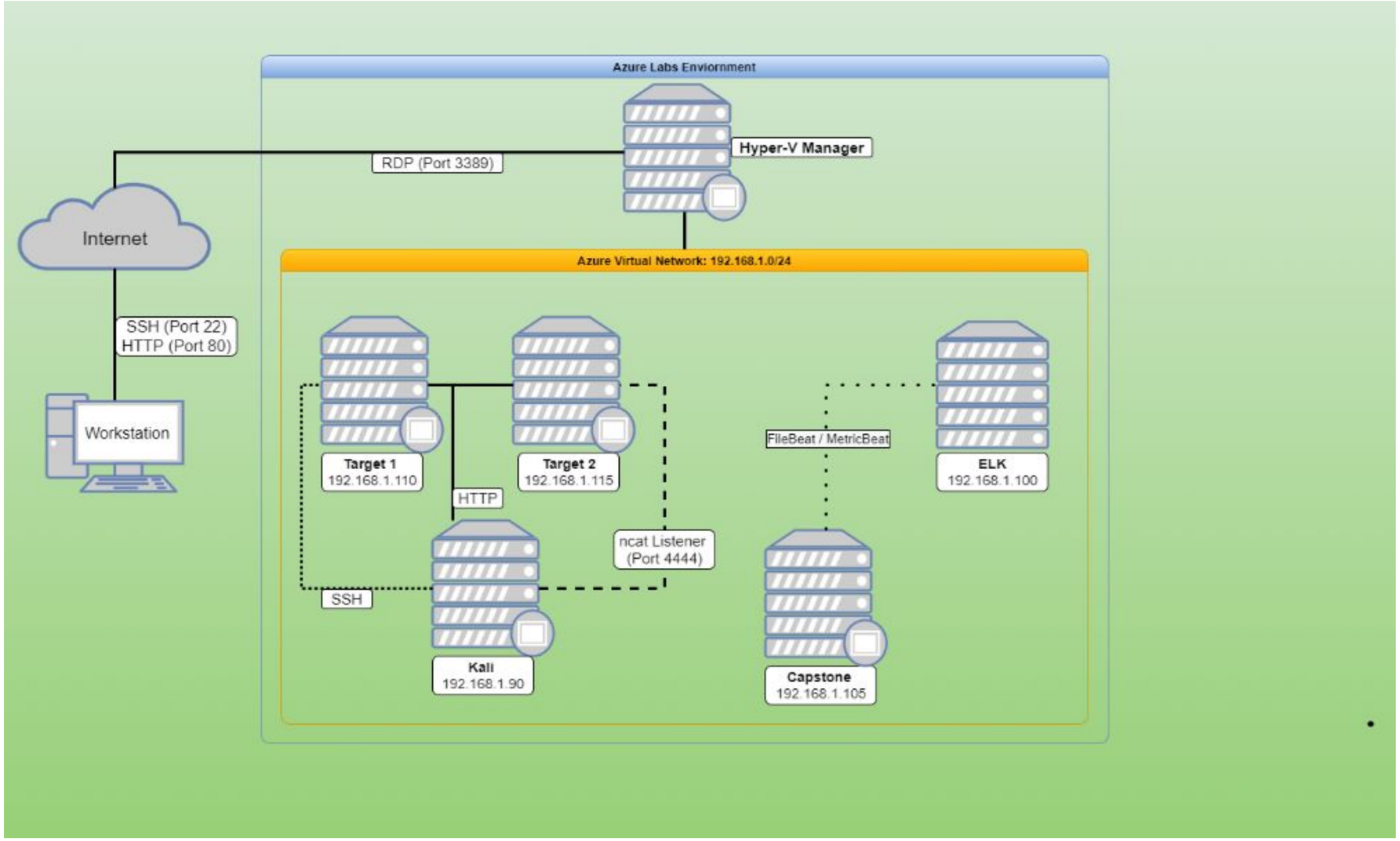
Attack, Defense & Analysis of a Vulnerable Network

Table of Contents

This document contains the following resources:

Network Topology & Critical Vulnerabilities

Network Topology



Network
Address Range: **192.168.1.0/24**
Netmask: **255.255.225.0**
Gateway: **192.168.1.1**

Machines
IPv4: **192.168.1.1**
OS: **Windows 10**
Hostname: **Hyper-V Manager (ML-REFVM-684427)**

IPv4: **192.168.1.90**
OS: **Linux**
Hostname: **Kali**

IPv4: **192.168.1.100**
OS: **Linux**
Hostname: **ELK Server**

IPv4: **192.168.1.105**
OS: **Linux**
Hostname: **Capstone**

IPv4: **192.168.1.110**
OS: **Linux 3.2 – 4.9**
Hostname: **Target 1**

Critical Vulnerabilities: Target 1

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

Vulnerability	Description	Impact
Brute Force	**Created Alert:** WHEN count() GROUPED OVER top 5 'http.response.status_code' IS ABOVE 400 FOR THE LAST 5 minutes	This Alert is High Reliability because it will reduce the number of false positives and alert when a high number of HTTP Error's are present after 5 minutes
Code Injection attacks (XSS or CRLF)	**Created Alert:** WHEN sum() of http.request.bytes OVER all documents IS ABOVE 3500 FOR THE LAST 1 minute	This is Medium Reliability because some it may generate false positives. There is the possibility that a large, non-malicious HTTP request may trigger the alarm.
Virus or Malware	**Created Alert:** WHEN max() OF system.process.cpu.total.pct OVER all documents IS ABOVE 0.5 FOR THE LAST 5 minutes	This is Low Reliability because this alert will generate a lot of false positives. CPU can spike for several different reasons.

Alerts Implemented

Excessive HTTP Errors

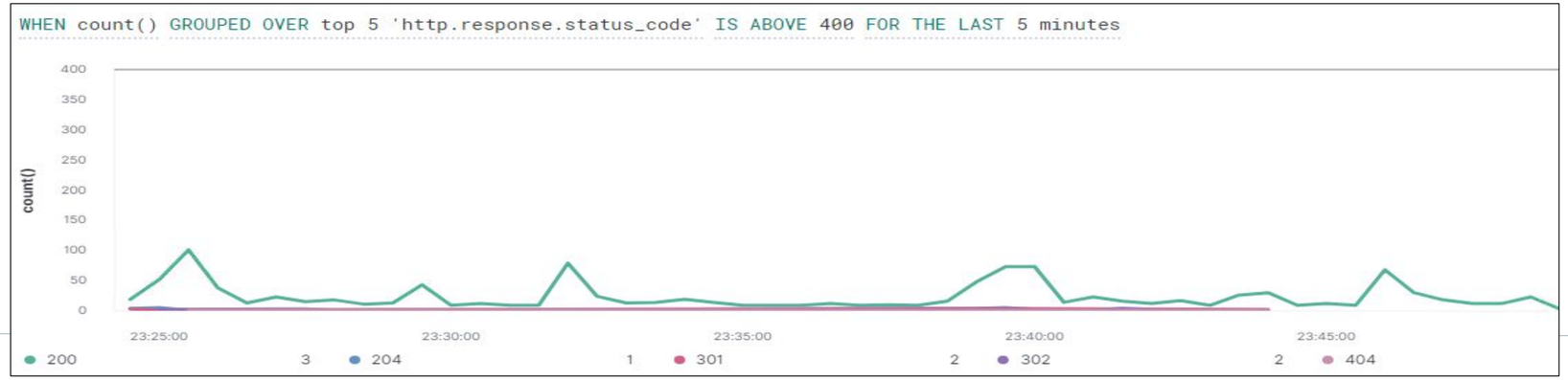
Alert 1 is implemented as follows:

- **Metric**: Packetbeat ('http.response.status_code’)

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

- **Vulnerability Mitigated**: Brute Force Attacks

- **Reliability**: This Alert is High Reliability because it will reduce the number of false positives and alert when a high number of HTTP Errors are present after 5 minutes



HTTP Request Size Monitor

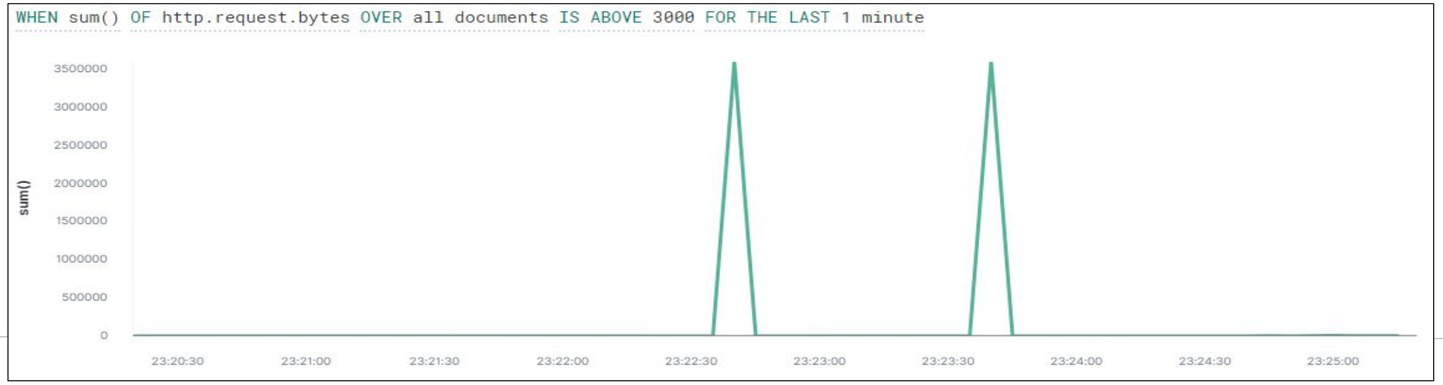
Alert 2 is implemented as follows:

- **Metric:** Packetbeat (http.request.bytes)

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

- **Vulnerability Mitigated**: Code Injection attacks (XSS or CRLF)

- **Reliability**: This is Medium Reliability because some it may generate false positives. There is the possibility that a large, non-malicious HTTP request may trigger the alarm.



CPU Usage Monitor

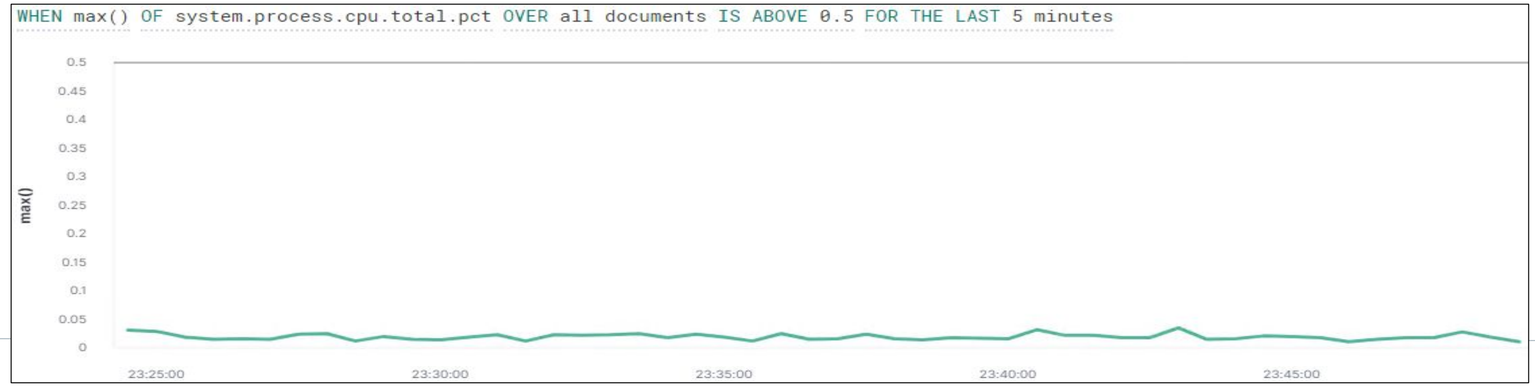
Alert 3 is implemented as follows:

- **Metric**: Metricbeat (system.process.cpu.total.pct)

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

- **Vulnerability Mitigated**: Virus or Malware

- **Reliability**: This is Low Reliability because this alert will generate a lot of false positives. CPU can spike for several different reasons.



Hardening

Hardening Against BRUTE FORCE Vulnerability on Target 1

Vulnerability 1: Brute Force Attacks

Patch: WordPress Hardening

- Update WordPress and other software: apt-get upgrade weekly
- Popular WordPress Plugins: Loginizer, WP Limit Login Attempts, Brute Force Login Protection: These plugins help protect websites from malicious attacks
- Implement a strong password policy: 12 characters/2 special characters and password reset every 60 days
- Create alerts when when a user has met the "failed login" threshold and lock out the user
- Implement a Multi-Factor Authentication password reset policy
- Establish rule to block all known VPN Traffic
- Disable unused features such as WordPress REST API

Why It Works:

- Updating software weekly typically will automatically fix errors and update patches
- Plugins help secure the website based on their intended purpose.
- Strong password policies help reduce BruteForce Attacks
- Alerts help notify Cyber Professionals when there is unusual activity like a large amount of HTTP Requests
- Multi-Factor Authentication is a useful tool to ensure attackers can not gain access to another user's account
- Blocking VPN Traffic is one way monitor and identify traffic
- Disabling REST API prevents enumeration of users

Hardening Against Code Injection Attacks on Target 1

Vulnerability 2: Code Injection attacks (XSS or CRLF)

Patch: Code Injection/DDOS Hardening

- Update software: apt-get upgrade weekly
- Create Ansible Playbook to include Code Injection Plugins:
  - XXXtrike
  - BruteXSS Terminal
  - Brute XSS GUI
  - XSS Scanner Online
  - XSSer
  - xsscrapy
- Restrict PHP and EXE
- Establish HTTP Request Limit Rules
  - Max URL Length
  - Max length of a query string
  - Max size of a request

Why It Works:

- Updating software weekly typically will automatically fix errors and update patches
- Creating Ansible Playbook with Industry Standard plugins improve automation/efficiency and improve security for Code Injection Attacks
- Restricting PHP and EXE helps reduce Injection Attacks on the front end
- Establishing HTTP Request Limit Rules automatically drops traffic when threshold has been met

Hardening Against Virus or Malware Vulnerability on Target 1

Vulnerability 3: Virus or Malware

Patch: Virus and Malware Hardening

- Update software: apt-get upgrade weekly
- Update and install industry standard Anti-Virus Software
- Implement and Monitor Network using Intrusion Detection System (IDS)
  - SNORT
  - Kibana
  - Wireshark
  - Nessus

Why It Works:

- Updating software weekly typically will automatically fix errors and update patches
- Installing Anti-Virus Software is critical to any Network Security. Installing Anti-Virus Software is an Industry Standard Practice.
- Create Ansible Playbook to automate and update Virus Protection
- IDS is critical for network security because it enables you to detect and respond to malicious traffic

Implementing Patches

Implementing Patches with Ansible

Playbook Overview: Brute Force

Playbook Name: ansible-vault-brute-force.sh

Executable File | 20 lines (17 sloc) | 374 Bytes

```
1  vault_path=$1
2
3  dictionary=$(echo {p,a,s}{p,a,s}{p,a,s}{p,a,s})
4
5  words=($dictionary)
6  for pass in "${words[@]}"
7  do
8    echo -n "Trying $pass "
9    echo $pass > vaultpass
10
11   ansible-vault decrypt $vault_path --vault-password-file vaultpass &>/dev/null
12   if [ $? -eq 0 ]; then
13     echo 'success'
14     echo "Password is: $pass"
15     break
16   else
17     echo "fail"
18   fi
19 done
20 rm vaultpass
```

Implementing Patches with Ansible

Playbook Overview: Code Injection Attacks (XSS or CRLF)

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted web sites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page.

XSS Vulnerability Scanner Tool's are simple tools to help mitigate Code Injection Attacks:

- ➔ XSStrike
- ➔ BruteXSS Terminal
- ➔ BruteXSS GUI
- ➔ XSS Scanner Online
- ➔ XSSer
- ➔ xsscrapy

Implementing Patches with Ansible

Playbook Overview: Virus or Malware

There are several trusted Industry Standard Playbooks that can be used to automate and secure the network against Viruses and Malware.

For example this is a picture of Splunk SOAR Playbook: Crowdstrike Malware Triage

The Playbook

This playbook walks through the steps that are performed automatically by Phantom to triage file hashes ingested from Crowdstrike.