CS 4410 Final Project

Technical Documentation

Prepared by: Dom Olhava, Ryan Walter, Aaryn Warrior

Date of Initial Preparation: 4/14/2025

Date of Last Edit: 5/9/2025

Revision Date(s): N/A

Overview

This Final Project tasks our System Security group with securing a vulnerable virtual machine while ensuring the continuity of existing services. We will assess and mitigate risks to minimize the likelihood and impact of threats. Our techniques will include, but are not limited to, updating software, patching insecure source code, and configuring protection systems such as firewalls. The ultimate goal is to achieve a fully functional server capable of withstanding cyber attacks more effectively.

The virtual machine assigned for hardening hosts a web server at IP address **10.161.7.39**, running on a **LAMP stack**.

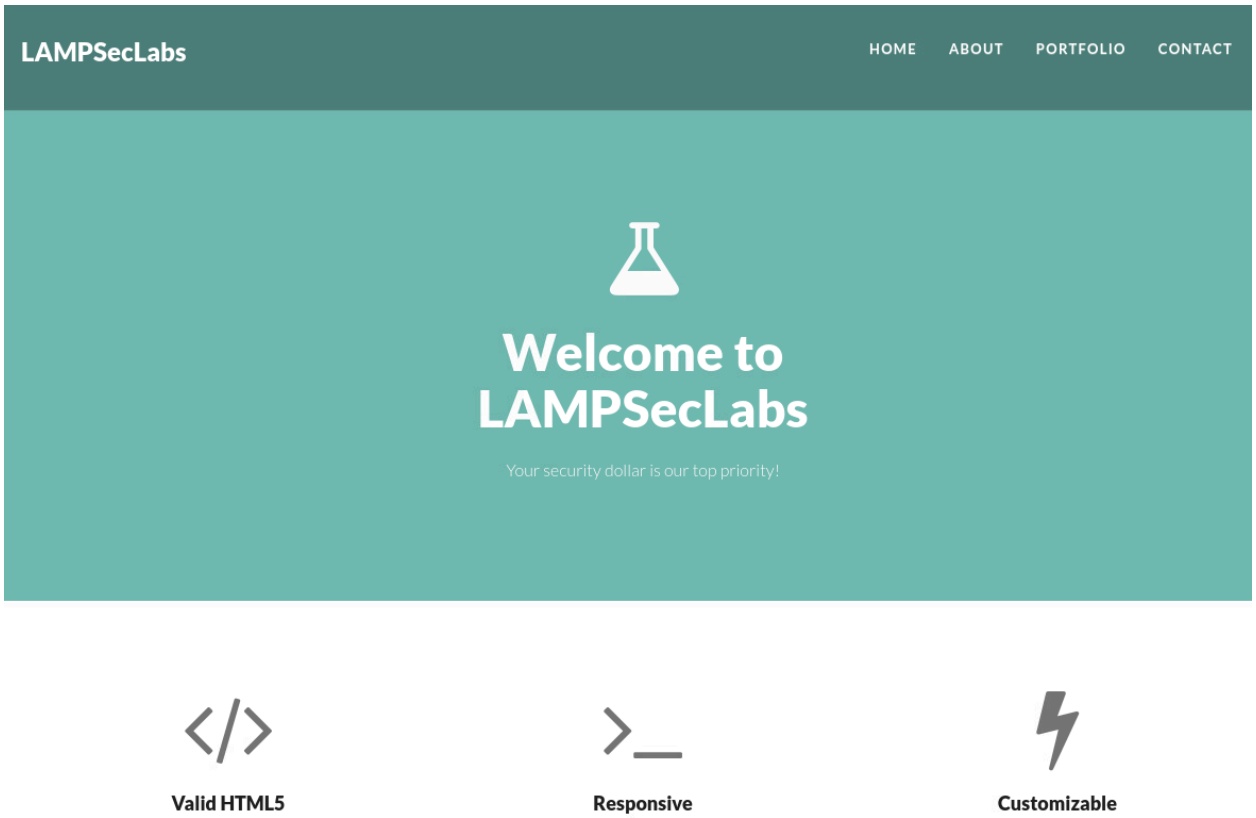


Table of Contents

Chapter 1: Summary of Services on CTF9

Externally Facing Services

An Nmap scan was performed to detect publicly facing services: **nmap -sV 10.161.7.39**

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV 10.161.7.39
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-21 20:13 EDT
Nmap scan report for 10.161.7.39
Host is up (0.021s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd 2.4.38 ((Ubuntu))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.o
Nmap done: 1 IP address (1 host up) scanned in 8.69 seconds
```

Internal Service Scan

Command: **systemctl list-units --type=service**

```
UNIT
accounts-daemon.service
acpid.service
apache2.service
apport.service
avahi-daemon.service
blk-availability.service
console-setup.service
cron.service
cups-browsed.service
cups.service
dbus.service
getty@tty1.service
grub-common.service
hddtemp.service
irqbalance.service
kerneloops.service
keyboard-setup.service
kmod-static-nodes.service
lightdm.service
lm-sensors.service
lvm2-lvmetad.service
lvm2-monitor.service
ModemManager.service
mysql.service
lines 1-25
```

Highlights:

- Apache HTTP Server 2.4.38 - Port 80
- MySQL - Port 3306 (Default)
- SSH - OpenSSH 7.6 - Port 22

Operating System: Linux Mint is one of the most popular Linux distributions due to being very lightweight and user friendly. It is open source and based on Debian and Ubuntu.

Linux Mint 19 - Kernel 4.15.0

```
bob@mint19:~$ hostnamectl
   Static hostname: mint19
         Icon name: computer-vm
           Chassis: vm
        Machine ID: ef635a010d284bc38d762d2b9f0e65ac
           Boot ID: 21dfeb2456db4748bfa7d097db2395a6
    Virtualization: vmware
  Operating System: Linux Mint 19
            Kernel: Linux 4.15.0-20-generic
      Architecture: x86-64
```

Web Server: Apache is a software that runs an HTTP server. It is commonly paired with Linux to host web applications but is also compatible with Windows and Mac. Its job is to establish a connection between a server and the browsers of website clients while delivering files back and forth between them.

Apache 2.4.38

```
bob@mint19:~$ apache2 -v
Server version: Apache/2.4.38 (Ubuntu)
Server built:   2019-03-02T15:45:45
```

Database: MySQL (MariaDB) is a relational database management system meaning that it's used as a source to store data into tables with columns and rows. MySQL is extremely popular due to its reliability and user-friendly style, causing many businesses and developers to utilize it.

MySQL 5.7.25

```
bob@mint19:~$ mysql -v
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10385
Server version: 5.7.25-0ubuntu0.18.04.2 (Ubuntu)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Reading history-file /home/bob/.mysql_history
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Programming Language: PHP is a server-side scripting language. It works with HTML and CSS to create dynamic content for websites, web applications, and other online services.

PHP 5.6.40

```
┌──(kali㉿kali)-[~]
└─$ nmap --script=http-php-version -p 80,443 10.161.7.39
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-22 10:44 EDT
Nmap scan report for 10.161.7.39
Host is up (0.00046s latency).

PORT    STATE  SERVICE
80/tcp  open   http
|_http-php-version: Version from header x-powered-by: PHP/5.6.40-5+ubuntu18.04.1+deb.sury.org+1
443/tcp closed https

Nmap done: 1 IP address (1 host up) scanned in 0.46 seconds
```

SSH: SSHD is a cryptographic network protocol designed for secure communication over an unsecured network. It is primarily used for remote login and command-line execution, replacing older, less secure protocols

OpenSSH 7.6p1

```
bob@mint19:~$ ssh -V
OpenSSH_7.6p1 Ubuntu-4ubuntu0.1, OpenSSL 1.0.2n  7 Dec 2017
```

SSH Encryption Algorithm: **ssh -Q cipher**

```
bob@mint19:~$ ssh -Q cipher
3des-cbc
aes128-cbc
aes192-cbc
aes256-cbc
rijndael-cbc@lysator.liu.se
aes128-ctr
aes192-ctr
aes256-ctr
aes128-gcm@openssh.com
aes256-gcm@openssh.com
chacha20-poly1305@openssh.com
```

Highlights (Common)

- 3DES
- AES

SSH Key Exchange Algorithm: **ssh -Q kex**

```
bob@mint19:~$ ssh -Q kex
diffie-hellman-group1-sha1
diffie-hellman-group14-sha1
diffie-hellman-group14-sha256
diffie-hellman-group16-sha512
diffie-hellman-group18-sha512
diffie-hellman-group-exchange-sha1
diffie-hellman-group-exchange-sha256
ecdh-sha2-nistp256
ecdh-sha2-nistp384
ecdh-sha2-nistp521
curve25519-sha256
curve25519-sha256@libssh.org
```

Highlights (Common

- Diffie Hellman
- ECDH

Chapter 2: Initial Vulnerability Report

Vulnerability Severity

Low	Medium	High

Nmap

Nmap is an open source network scanning reconnaissance tool which can be used for network discovery and vulnerability assessment. The main reason to use the Nmap tool on this system is to perform penetration testing, security checks, and ethical hacking for testing purposes. It identifies open ports, unsecured services, version vulnerability, and can even help in catching misconfigurations as well as authentication weaknesses.

Vulnerability Scan Results:

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV -F 10.161.7.39
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-20 20:06 EDT
Nmap scan report for 10.161.7.39
Host is up (0.00024s latency).
Not shown: 98 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd 2.4.38 ((Ubuntu))
MAC Address: 00:50:56:82:12:CA (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.99 seconds
```

Vulnerability	Description
OpenSSH 7.6p1 Ubuntu 4ubuntu0.1	An **older version** of OpenSSH has been detected but as it stands it seems that the version is still **stable** making it not an immediate issue. However, in regards to it being an older version the necessary patches to preventing timing attacks and user enumeration are not present but are given in the updated versions. The **impact** isn't major but given misconfigurations and weak credentials could heighten the impact.

Apache httpd 2.4.38	This version of Apache is out of date which is known to have the following vulnerabilities: ● Denial of Service ● Privilege Escalation ● Server-Side Request Forgery All are used in means to take down or steal information from a system or network. Each is individually dangerous for large and small scale groups with the best recommendation being to upgrade the version, secure rules with a firewall, and monitor for potential malicious behavior.

Nikto

Nikto is an open-source web server and web application scanner that automates vulnerability detection. It identifies security issues like misconfigurations, insecure files/programs, outdated software, and web app endpoints.

Vulnerability Scan Results:

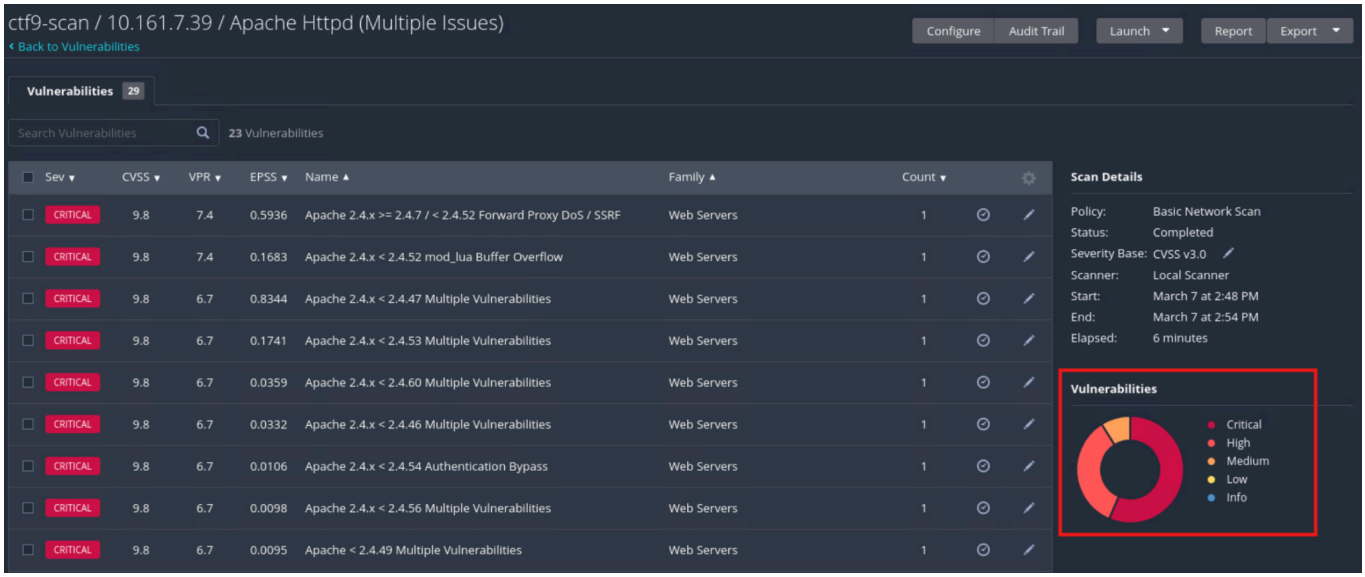
```
└─$ nikto -h http://10.161.7.39
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          10.161.7.39
+ Target Hostname:    10.161.7.39
+ Target Port:        80
+ Start Time:         2025-04-16 14:15:22 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.38 (Ubuntu)
+ /: Retrieved x-powered-by header: PHP/5.6.40-5+ubuntu18.04.1+deb.sury.org+1.
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X
-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a differen
t fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.38 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ /: Web Server returns a valid response with junk HTTP methods which may cause false positives.
+ /test.php: Output from the phpinfo() function was found.
+ /admin/login.php?action=insert&username=test&password=test: phpAuction may allow user admin accounts to be inserted without prop
er authentication. Attempt to log in with user 'test' password 'test' to verify. See: http://cve.mitre.org/cgi-bin/cvename.cgi?nam
e=CVE-2002-0995
+ /admin/: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /css/: Directory indexing found.
+ /css/: This might be interesting.
+ /img/: Directory indexing found.
+ /img/: This might be interesting.
+ /logs/: Directory indexing found.
+ /logs/: This might be interesting.
+ /db.php: This might be interesting: has been seen in web logs from an unknown scanner.
+ /test.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of system information. See: CWE-
552
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /admin/login.php: Admin login page/section found.
+ /test.php: This might be interesting.
+ 8102 requests: 0 error(s) and 19 item(s) reported on remote host
```

Vulnerability	Description
Apache/2.4.38	Version 2.4.38 is outdated. The newest stable version is 2.4.54. Older versions may have unpatched vulnerabilities.
/test.php	The phpinfo() function was able to be called by visiting the /test.php endpoint. This file displays critical system information which can allow attackers to target vulnerable software versions, configurations, etc.
/admin/login.php?action=insert&username=test&password=test	This endpoint may allow user admin accounts to be inserted into a URL without proper authentication.
/admin /img /css /logs /db.php /icons/README	These were all found as web endpoints that could be accessed. Leaving these endpoints available to public users may pose a security risk since outsiders may be able to see raw files straight from the backend server. Severity varies per endpoint.

Nessus

Nessus is a scanning tool that does reconnaissance on networks and systems to look for possible vulnerabilities and misconfigurations. Once the scan has completed, a highly detailed report will be available to view allowing us to make configurations. After extensive threat detection and scanning with the Nessus software it can be determined that this particular machine needs intensive work to strengthen the uncovered vulnerabilities.

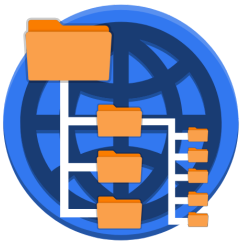
Vulnerability Scan Results:



Vulnerability	Description
PHP Unsupported Version Detection	Unsupported version of PHP 5.6.40 is outdated and cannot access new patches from the current most secure and supported versions 8.1.x/ 8.2.x / 8.3.x. New security updates of the system will not be present, leaving attackers to easily search this particular outdated version's vulnerabilities and exploit them. This could lead to unauthorized access from an unknown user and could even allow an attacker to perform a Denial of Service attack or steal sensitive/compromised data.
Apache 2.4.x Multiple Vulnerabilities	During the investigation it has been uncovered that the Apache web server's httpd and HTTP versions are out of date causing several critical vulnerabilities to be identified. Potential attacks may include: ● **Authentication Bypass:** Misinterpreted security rules can expose sensitive parts of a website and allows an attacker to access restricted resources. This could mean a leak in customer data or administrative credentials. This is a very critical vulnerability because it allows attackers to make malicious changes and steal private information. ● **Denial of Service (DoS):** A crafted URI sent to httpd configured as a forward proxy can cause a crash. Not only that but another example of a crash can happen when the Apache web server is given a HTTP/2 connection and attempts to upgrade to WebSocket which is a communication protocol between client and server using a single TCP connection. There are many other ways to cause a DoS and the main connection between them all is to overload a certain aspect of the web server and cause a crash. ● **Buffer Overflow:** Using writing requests to take up more resources than the memory of the system can handle and can cause corruption of data or critical data being overwritten.

	● **Cross-Site-Scripting:** Triggering errors made in the proxy can lead attackers to manipulate it to send a victim to a different malicious page. ● **Request splitting:** The backend server is very susceptible to misinformation. It can get confused easily and because of this, an attacker can inject malicious requests to manipulate user responses or steal data.
Apache 2.4.x >= 2.4.7 / <2.4.52 Forward Proxy DoS / SSRF	Like mentioned above the Apache HTTP server is using an out of date version but what makes this vulnerability so specific and individual is that an attacker could perform: ● **Server-Side Request Forgery**: A way of manipulating requests to access internal resources. This can be done by manipulating weak responses which allow attackers to force requests to the server to gain access to the internal network. Knowing this an attacker can leak sensitive information and cause further exploitation. If left unchanged a potential concentration of control and hold over this system could be completely taken away giving an attacker almost all data within. There needs to be an upgrade and tighter restrictions on the proxy settings.
JQuery 1.2 < 3.5.0 Multiple XXS	The jQuery version is out of date causing a mishandling of HTML parsing and manipulation in API methods. Due to this flaw it can cause Cross-Site-Scripting allowing the attacker to potentially escalate privileges, cause malicious redirects, capture keystrokes, or even impersonate users in the system. This is classified as a medium vulnerability but the impact could be critical depending on the type of information being held.

DirBuster



DirBuster is a powerful directory and file brute-forcing tool used in penetration testing to discover hidden directories and files on web servers. Wordlists can be given to DirBuster in order to systematically test for the presence of specific directories and files on a web server, helping to uncover sensitive endpoints/resources that might be vulnerable to exploitation.

Vulnerability Scan Results:

```
┌──(kali㉿kali)-[~]
└─$ dirbuster
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Starting OWASP DirBuster 1.0-RC1
Starting dir/file list based brute forcing
File found: /index.php - 200
Dir found: / - 200
Dir found: /contact/ - 200
Dir found: /img/ - 200
File found: /contact/contact.php - 200
Dir found: /icons/ - 403
Dir found: /img/parallax/ - 200
Dir found: /img/portfolio/ - 200
File found: /feedback.php - 200
Dir found: /img/team/ - 200
File found: /static.php - 200
```

Vulnerability	Description
/img/*	An **img directory** was able to be accessed within the browser. Not super dangerous but the directory can be traversed without authentication.
/logs/access_log.txt	An **access_log.txt** file was able to be accessed within the browser. This file which tracks who or what is trying to get a hold of the server. Having this exposed can allow attackers to see if their penetration methods are detected.
/admin/login.php /admin/upload.php /admin/users.php /admin/auth.php /admin/add_edit.php	An **admin directory** was able to be found by DirBuster. Thankfully, only the login.php page is accessible in the browser without login. Other endpoints redirect to the login page but may be able to be bypassed with an attack proxy. Best practice is to hide all endpoints except login.php until the session is properly authenticated.
/js/*	A **js directory** was able to be accessed within the browser. Not super dangerous but the directory can be traversed without authentication.

Manual Discovery

SQLMap

SQLMap is an open-source penetration testing tool that automates the process of detecting and exploiting SQL injection vulnerabilities in web applications. It supports a wide range of databases and can extract data, access the file system, and even execute commands on the database server.

Command: **sqlmap -u "http://10.161.7.39/admin" -data "username=admin&password=admin" -p password -D lampsec -T user -C user_name,user_password --dump**

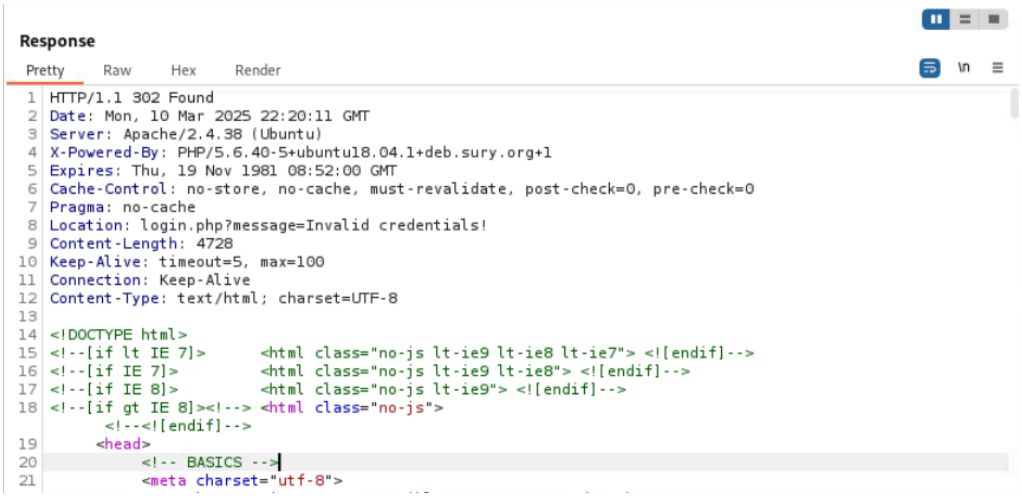
SQLMap Output:

```
+----------------+-----------------------------------------+
| user_name      | user_password                           |
+----------------+-----------------------------------------+
| administrator  | dab64765f3d4fc29ced777be274337ea (hacking)   |
| alice          | fdfaf065cbbfe6e453229e536924b0f1              |
| bob            | 93b542f0c7a6f2279fc94f44b013baf1 (billybob)   |
| eve            | 4c4999ac17adcef1a5a75fab71e5c857 (invisible)  |
| justin         | b6e4e4d617ec2406cea4555a5c40e137              |
+----------------+-----------------------------------------+
```

Vulnerability	Description
SQL Injection **/admin/login.php**	SQLMap was used to exploit a SQL **injection** vulnerability in the login form by crafting custom HTTP POST requests. This allowed the program to bypass authentication, enumerate the company's database, extract usernames and password hashes, and successfully crack three of the recovered hashes.

Burp Suite

Burp Suite is a web vulnerability scanner and penetration testing tool that acts as an intercepting proxy between your browser and target applications. Its attack proxy feature allows you to capture, inspect, and modify HTTP/S traffic in real time to find and exploit security flaws.



Failed login attempts redirect the user in the browser to the login screen but BurpSuite was leveraged to remove the **"Location"** header, causing the admin dashboard to render without proper credentials.

After accessing the admin dashboard, an **insecure file upload** was found.

A **reverse shell** was then able to be uploaded directly to the web server. This shell was then triggered remotely to connect to the attacking machine without proper credentials.

Vulnerability	Description
HTTP Redirect Logic	Allowing the server to return the full HTML of a protected page upon failed login attempts poses a significant security risk, as it exposes sensitive client-side components that can be easily inspected or manipulated to facilitate unauthorized access.
Unsanitized File Upload	Sanitizing user input is essential for web servers handling multiple users. Allowing non-image files to be uploaded through an image upload field introduces a serious security risk, as attackers can upload malicious code, such as reverse shells, that may be executed, leading to severe breaches and long-term damage to the server.

Chapter 3: Table of Defensive Deliverables

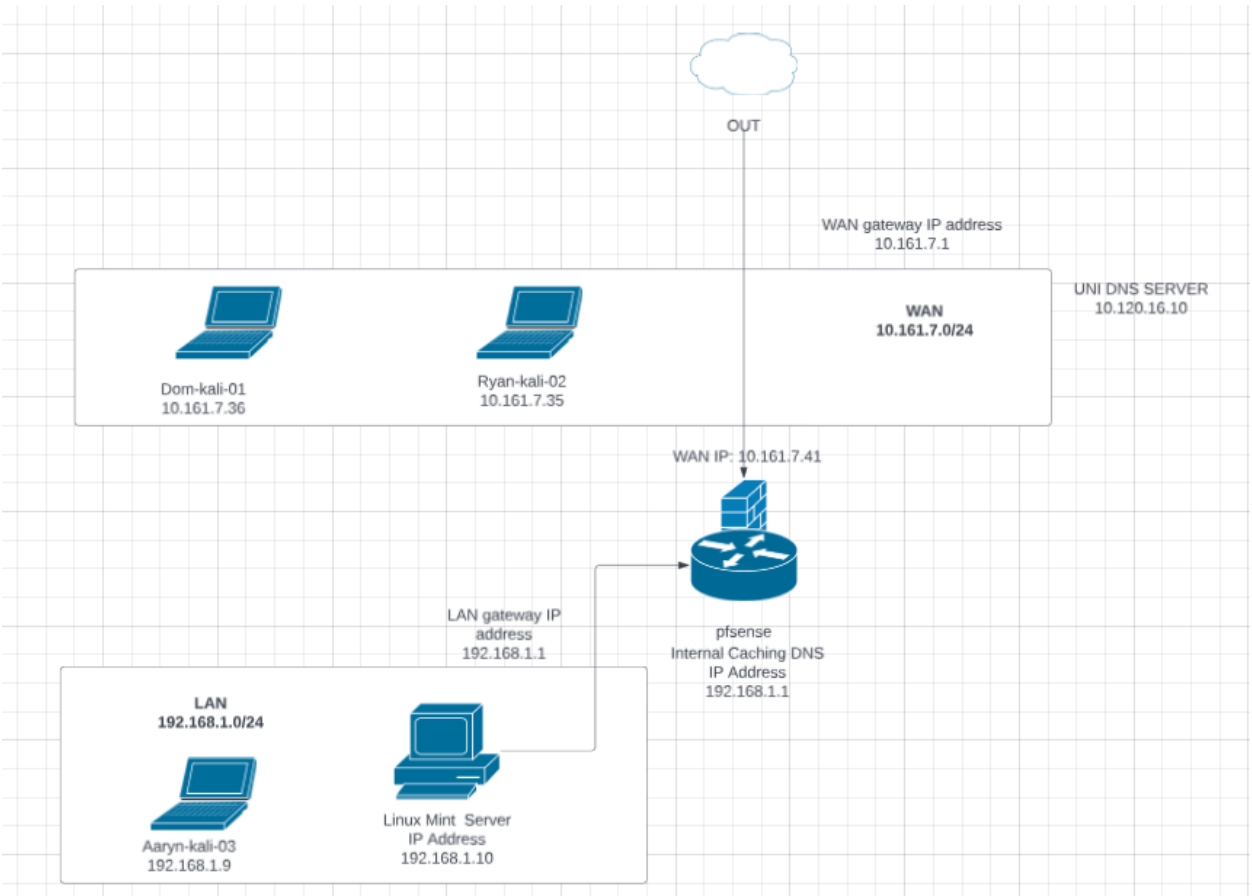
Task	**Deliverable**	**Person**	**Due Date**	**Done?** (Notes)
Chapter 2	Investigate ctf9 machine with Nmap and Nessus for chapter 2	Aaryn	4/20	Yes
Chapter 2	Nikto, Dirbuster, Manual for chapter 2	Dom	4/20	Yes
Chapter 1	Research system info for chapter 1	Ryan	4/20	Yes
Chapter 1	Find the OS version and services for chapter 1	Aaryn	4/20	Yes
Chapter 4	pfSense Setup	Aaryn	4/27	Yes
Chapter 4	Modsecurity (Apache)	Ryan	4/27	Yes
Chapter 4	Update php to 8.1, MySQLi, MySQL Password	Dom	4/27	Yes
Chapter 4	UFW (Uncomplicated Firewall) **[Medium]**	Ryan	5/5	Yes
Chapter 4	Install and configure Snort on pfsense **[Hard]**	Aaryn	5/5	Yes
Chapter 4	SQL Injection Prevention and File Upload Sanitization **[Hard]**	Dom	5/5	Yes
Chapter 5	Finish configurations and rerun vulnerability scans.	All of Us	5/5	Yes
Chapter 6	Future Work (Other things we can do to defend in the future)	Dom and Aaryn	5/7	Yes

Chapter 4: Defensive Deliverable

PfSense

pfSense is an open-source firewall and router software based on FreeBSD, designed to provide robust network security and management capabilities. It offers a wide range of features including stateful packet filtering, VPN support, load balancing, and detailed reporting, making it suitable for both small and large network environments.

Network Diagram [LINK]



After getting the main pfSense configuration completed, there needs to be setup on the actual machine to put the server behind the firewall. Below there are images of how to configure a static IP address so that pfSense can identify and manage the server using firewall rules, port forwarding, and other services.

```
bob@mint19:~$ cd /etc/netplan
bob@mint19:/etc/netplan$ nmcli connection show
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  3554eccb-a106-3a05-b3fd-9b7b8c8038f2  ethernet  ens33
bob@mint19:/etc/netplan$
```

```
bob@mint19:~$ sudo nmcli con mod "Wired connection 1" ipv4.addresses 192.168.1.10/24
bob@mint19:~$ sudo nmcli con mod "Wired connection 1" ipv4.gateway 192.168.1.1
bob@mint19:~$ sudo nmcli con mod "Wired connection 1" ipv4.dns 10.120.16.10
bob@mint19:~$ sudo nmcli con mod "Wired connection 1" ipv4.method manual
```

```
bob@mint19:~$ sudo nmcli con down "Wired connection 1" && nmcli con up "Wired connection 1"
Connection 'Wired connection 1' successfully deactivated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveCon
nection/1)
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/2)
```

There should now be a static IP address set for the server! **192.168.1.10**

```
bob@mint19:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:50:56:82:12:ca brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.10/24 brd 192.168.1.255 scope global noprefixroute ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::e0fc:137a:e6d8:2644/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Setting up the Port Forwarding rules allows the current open ports to be more secure and manageable. The following images are the newly created rules and port forwarding that are necessary to access the machine from the WAN.

Firewall / NAT / Port Forward

Port Forward | 1:1 | Outbound | NPt

Rules

	Interface	Protocol	Source Address	Source Ports	Dest. Address	Dest. Ports	NAT IP	NAT Ports	Description	Actions
	WAN	TCP	*	*	WAN address	443 (HTTPS)	192.168.1.10	443 (HTTPS)	Port Forward HTTPS	
	WAN	TCP	*	*	WAN address	80 (HTTP)	192.168.1.10	80 (HTTP)	Port Forward HTTP	
	WAN	TCP	*	*	WAN address	22 (SSH)	192.168.1.10	22 (SSH)	Port Forward SSH	

Add | Add | Delete | Toggle | Save | Separator

WAN Firewall Rules

Floating WAN LAN

Rules (Drag to Change Order)

States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
0/0 B	IPv4 TCP	*	*	192.168.1.10	22 (SSH)	*	none		NAT Port Forward SSH	
0/0 B	IPv4 TCP	*	*	192.168.1.10	80 (HTTP)	*	none		NAT Port Forward HTTP	
0/0 B	IPv4 TCP	*	*	192.168.1.10	443 (HTTPS)	*	none		NAT Port Forward HTTPS	

Add Add Delete Toggle Copy Save Separator

Now the only thing missing is to make a few more firewall rules to the machine so that it is truly secure. There needs to be rules to manage traffic so that it can be more closely monitored especially since we don't need to be using all ports. The following rules allow SSH, DNS, HTTP, HTTPS, and Ping requests.

LAN Firewall Rules

Floating WAN LAN

Rules (Drag to Change Order)

States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
0/167.58 MiB	*	*	*	LAN Address	443 80	*	*		Anti-Lockout Rule	
0/0 B	IPv4 TCP	*	*	*	22 (SSH)	*	none			
1/3 KiB	IPv4 TCP/UDP	*	*	*	53 (DNS)	*	none			
0/335 KiB	IPv4 TCP	*	*	*	80 (HTTP)	*	none			
1/45 KiB	IPv4 TCP	*	*	*	443 (HTTPS)	*	none			
0/0 B	IPv4 ICMP echoreq	*	*	*	*	*	none			
0/0 B	IPv4 *	LAN subnets	*	*	*	*	none		Default allow LAN to any rule	
0/0 B	IPv6 *	LAN subnets	*	*	*	*	none		Default allow LAN IPv6 to any rule	

Add Add Delete Toggle Copy Save Separator

To conclude this section, the server is more secure since the firewall allows specific port traffic with port forwarding. From outside the internal network, access can be granted using the ip address **10.161.7.41** while internal network access will be granted using **192.168.1.10**.

Inside Access

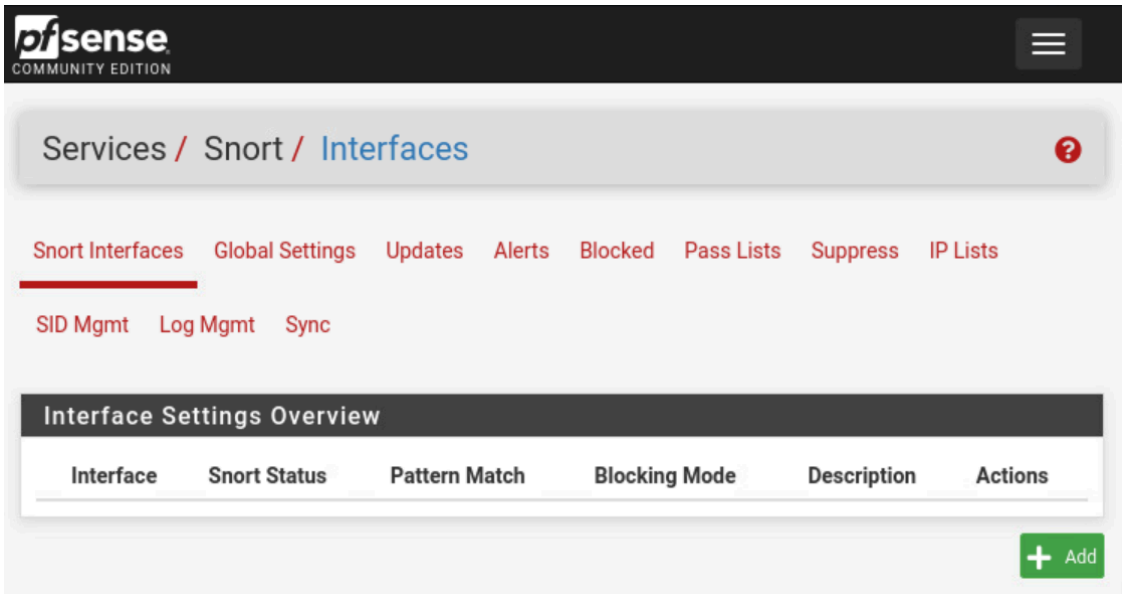
```
┌──(kali㉿kali)-[~]
└─$ ssh bob@192.168.1.10
bob@192.168.1.10's password:
Last login: Wed Apr 23 13:28:14 2025 from 192.168.1.9
bob@mint19:~$
```

Outside Access

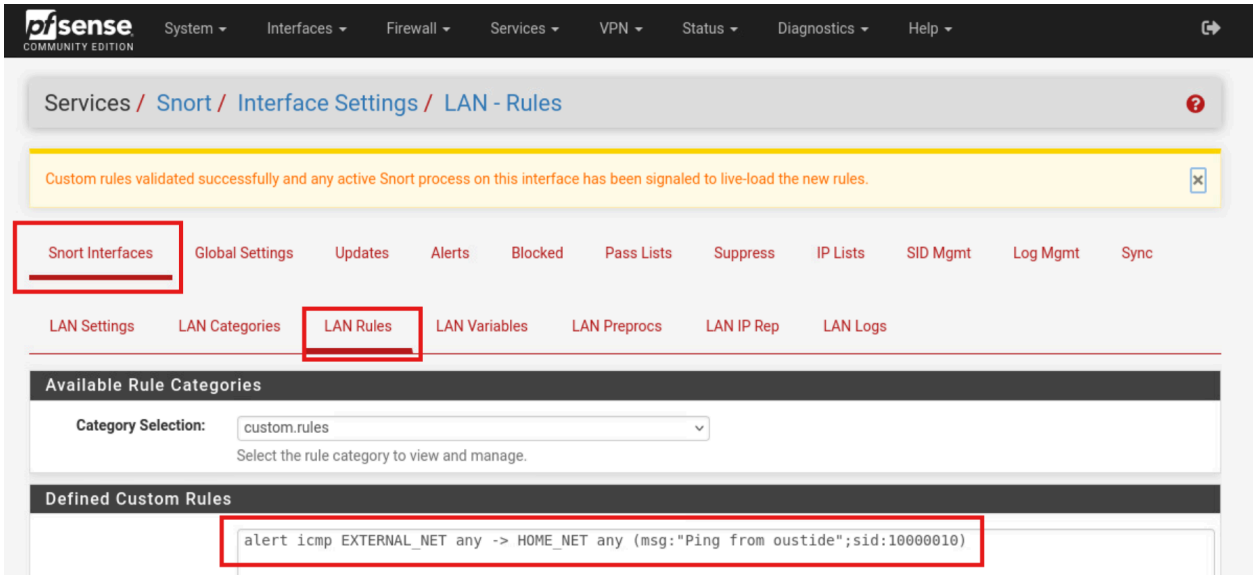
```
┌──(kali㉿kali)-[~]
└─$ ssh bob@10.161.7.41
bob@10.161.7.41's password:
Last login: Wed Apr 23 13:33:04 2025 from 192.168.1.9
bob@mint19:~$
```

Snort on PfSense

Within pfSense, the firewall can be elevated from basic protection to an advanced intrusion detection and prevention system (IDPS). By integrating Snort, a powerful open source network intrusion detection and prevention software, administrators gain access to a wide range of configurable options designed to detect, analyze, and block potential threats in real time. For this server, a custom firewall rule set was developed in conjunction with global and community-driven Snort rules. This configuration significantly enhances network visibility and actively defends against malicious activity by automatically blocking IP addresses that exhibit suspicious or predefined malicious behavior.



Custom Rule: Monitors Ping from the outside network to the inside network (IDS)



Global Settings

Snort GPLv2 Community Rules

Enable Snort GPLv2	☑ Click to enable download of Snort GPLv2 Community rules
	The Snort Community Ruleset is a GPLv2 Talos certified ruleset that is distributed free of charge without any Snort Subscriber License restrictions. This ruleset is updated daily and is a subset of the subscriber ruleset.

Emerging Threats (ET) Rules

Enable ET Open	☑ Click to enable download of Emerging Threats Open rules
	ETOpen is an open source set of Snort rules whose coverage is more limited than ETPro.
Enable ET Pro	☐ Click to enable download of Emerging Threats Pro rules
	Sign Up for an ETPro Account ETPro for Snort offers daily updates and extensive coverage of current malware threats.

FEODO Tracker Botnet C2 IP Rules

Enable FEODO Tracker Botnet C2 IP Rules	☑ Click to enable download of FEODO Tracker Botnet C2 IP rules
	Feodo Tracker tracks certain families that are related to, or that evolved from, Feodo. Originally, Feodo was an ebanking Trojan used by cybercriminals to commit ebanking fraud. Since 2010, various malware families evolved from Feodo, such as Cridex, Dridex, Geodo, Heodo and Emotet.

Some Enabled Rules on WAN (Block)

Selected Category's Rules

Legend: Default Enabled, Enabled by user, Auto-enabled by SID Mgmt, Action/content modified by SID Mgmt, Rule action is alert, Default Disabled, Disabled by user, Auto-disabled by SID Mgmt

State	Action	GID	SID	Proto	Source	SPort	Destination	DPort	Message
		1	2002023	tcp	any	any	any	6666:7000	ET CHAT IRC USER command
		1	2002024	tcp	any	any	any	6666:7000	ET CHAT IRC NICK command
		1	2002025	tcp	any	any	any	6666:7000	ET CHAT IRC JOIN command
		1	2002026	tcp	any	any	any	6666:7000	ET CHAT IRC PRIVMSG command
		1	2002027	tcp	any	6666:7000	any	any	ET CHAT IRC PING command
		1	2101640	tcp	$HOME_NET	any	$EXTERNAL_NET	6666:7000	GPL CHAT IRC DCC chat request
		1	2101639	tcp	$HOME_NET	any	$EXTERNAL_NET	6666:7000	GPL CHAT IRC DCC file transfer request
		1	2025066	tcp	any	any	any	6666:7000	ET CHAT IRC USER Likely bot with 0 0 colon checkin
		1	2025067	tcp	any	any	any	!6666:7000	ET CHAT IRC USER Off-port Likely bot with 0 0 colon checkin
		1	2101729	tcp	$HOME_NET	any	$EXTERNAL_NET	6666:7000	GPL CHAT IRC Channel join
		1	2002028	tcp	any	any	any	6666:7000	ET CHAT IRC PONG response
		1	2017294	tcp	$HOME_NET	any	$EXTERNAL_NET	$HTTP_PORTS	ET INFO Adobe PKG Download Flowbit Set

Some Enabled Rules on LAN (Monitor)

Selected Category's Rules

Legend: Default Enabled, Enabled by user, Auto-enabled by SID Mgmt, Action/content modified by SID Mgmt, Rule action is alert, Default Disabled, Disabled by user, Auto-disabled by SID Mgmt

State	Action	GID	SID	Proto	Source	SPort	Destination	DPort	Message
		1	2002023	tcp	any	any	any	6666:7000	ET CHAT IRC USER command
		1	2002024	tcp	any	any	any	6666:7000	ET CHAT IRC NICK command
		1	2002025	tcp	any	any	any	6666:7000	ET CHAT IRC JOIN command
		1	2002026	tcp	any	any	any	6666:7000	ET CHAT IRC PRIVMSG command
		1	2002027	tcp	any	6666:7000	any	any	ET CHAT IRC PING command
		1	2101640	tcp	$HOME_NET	any	$EXTERNAL_NET	6666:7000	GPL CHAT IRC DCC chat request
		1	2101639	tcp	$HOME_NET	any	$EXTERNAL_NET	6666:7000	GPL CHAT IRC DCC file transfer request
		1	2025066	tcp	any	any	any	6666:7000	ET CHAT IRC USER Likely bot with 0 0 colon checkin
		1	2025067	tcp	any	any	any	!6666:7000	ET CHAT IRC USER Off-port Likely bot with 0 0 colon checkin
		1	2101729	tcp	$HOME_NET	any	$EXTERNAL_NET	6666:7000	GPL CHAT IRC Channel join
		1	2002028	tcp	any	any	any	6666:7000	ET CHAT IRC PONG response
		1	2035003	tcp	any	any	$HOME_NET	any	ET EXPLOIT Apache Spark RPC - Unauthenticated RegisterApplication Request (CVE-2020-9480)

Explanation:
For the global settings, I've enabled rules from the Snort GPLv2 Community Rules, Feodo Tracker, and Emerging Threats. I chose these because they offer essential baseline protection with generally low false positive rates. Of the three, the Emerging Threats ruleset is the most aggressive, as it's primarily designed to protect the internal LAN from external WAN threats. While these rules can sometimes generate false positives, I ran tests against the web server and found that none of the enabled rules triggered any serious alerts or blocks, which gave me confidence in their current configuration.

Conclusion:
Following the implementation and fine-tuning of the selected Snort rules, comprehensive testing was conducted on the web server to ensure complete functionality. The results indicate that the server maintains a seamless operational level across both internal and external network interfaces. Core features, including web-based services and interactive components such as the contact form, continue to perform as expected without interruption. This confirms that the enhanced security measures have been effectively integrated without compromising the usability or availability of essential services. Overall, the system now benefits from improved threat detection and mitigation capabilities while preserving the user experience and application reliability.

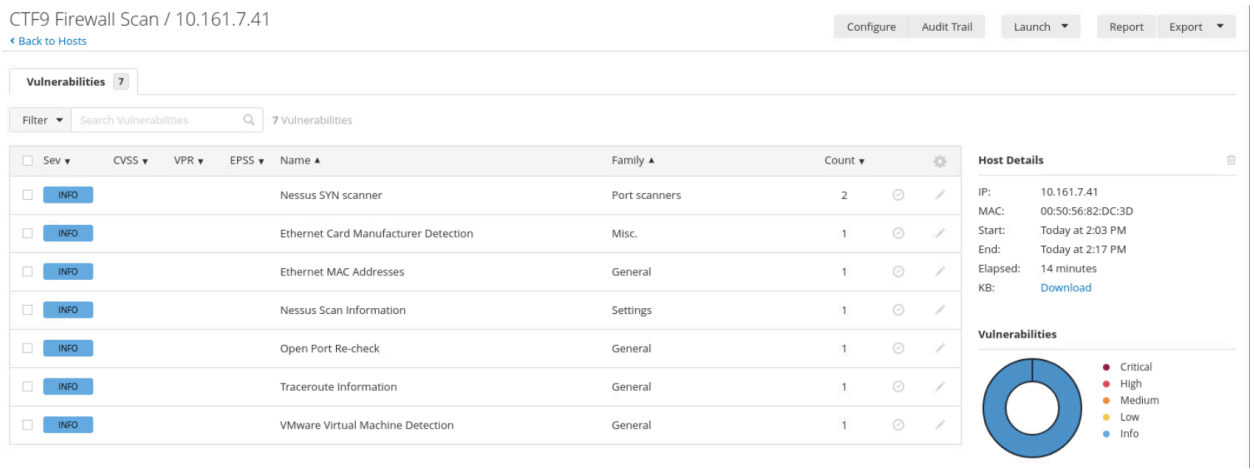
Scanning Denial:
I've enabled a rule that blocks scanning attempts to prevent attackers from gathering information they could use to exploit vulnerabilities in the system. This helps keep the system more secure by limiting what potential threats can see. However, for Chapter 5, I'll temporarily disable this rule to demonstrate that the major security issues have been addressed and are no longer exploitable.

Tried a standard Nmap service scan and the packet traffic for probing is blocked:

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV 10.161.7.41
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-28 14:20 EDT
Nmap scan report for 10.161.7.41
Host is up (0.00027s latency).
All 1000 scanned ports on 10.161.7.41 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 00:50:56:82:DC:3D (VMware)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 21.54 seconds
```

Tried a Nessus Scan only server information was captured:

- SSH on Port 22
- HTTP on Port 80



Apache PHP 8.1 Upgrade

The Apache server on the CTF9 machine was running PHP version 5.6.4. Released in 2014, **PHP 5.6.4** has accumulated numerous vulnerabilities and exploits over the past decade. To enhance the security of the web server, upgrading to **PHP 8.1**, which was released in 2021, was crucial.

An **apache2** repository with updated php modules was added to the server:
sudo add-apt-repository ppa:ondrej/apache2
sudo apt update

php8.1 and the Apache php8.1 modules were then installed:
sudo apt install php8.1 libapache2-mod-php8.1

The old php5.6 module was disabled and the new php8.1 module was enabled:
ls /etc/apache2/mods-available | grep php

```
bob@mint19:/etc/php/8.1/apache2$ ls /etc/apache2/mods-available | grep php
php5.6.conf
php5.6.load
php7.2.conf
php7.2.load
php8.1.conf
php8.1.load
```

sudo a2dismod php5.6
sudo a2enmod php8.1
sudo systemctl restart apache2

In order to check the current Apache PHP version, navigate to
http://10.161.7.41/info.php
(This endpoint will be removed after proper installation is complete)

Before Upgrade

PHP Version 5.6.40-5+ubuntu18.04.1+deb.sury.org+1

System	Linux mint19 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/5.6/apache2
Loaded Configuration File	/etc/php/5.6/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/5.6/apache2/conf.d
Additional .ini files parsed	/etc/php/5.6/apache2/conf.d/10-mysqlnd.ini, /etc/php/5.6/apache2/conf.d/10-opcache.ini, /etc/php/5.6/apache2/conf.d/10-pdo.ini, /etc/php/5.6/apache2/conf.d/20-calendar.ini, /etc/php/5.6/apache2/conf.d/20-ctype.ini, /etc/php/5.6/apache2/conf.d/20-exif.ini, /etc/php/5.6/apache2/conf.d/20-fileinfo.ini, /etc/php/5.6/apache2/conf.d/20-ftp.ini, /etc/php/5.6/apache2/conf.d/20-gettext.ini, /etc/php/5.6/apache2/conf.d/20-iconv.ini, /etc/php/5.6/apache2/conf.d/20-json.ini, /etc/php/5.6/apache2/conf.d/20-mysql.ini, /etc/php/5.6/apache2/conf.d/20-mysqli.ini, /etc/php/5.6/apache2/conf.d/20-pdo_mysql.ini, /etc/php/5.6/apache2/conf.d/20-phar.ini, /etc/php/5.6/apache2/conf.d/20-posix.ini, /etc/php/5.6/apache2/conf.d/20-readline.ini, /etc/php/5.6/apache2/conf.d/20-shmop.ini, /etc/php/5.6/apache2/conf.d/20-sockets.ini, /etc/php/5.6/apache2/conf.d/20-sysvmsg.ini, /etc/php/5.6/apache2/conf.d/20-sysvsem.ini, /etc/php/5.6/apache2/conf.d/20-sysvshm.ini, /etc/php/5.6/apache2/conf.d/20-tokenizer.ini

After Upgrade

PHP Version 8.1.3

System	Linux mint19 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Build Date	Feb 21 2022 14:48:26
Build System	Linux
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.1/apache2
Loaded Configuration File	/etc/php/8.1/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/8.1/apache2/conf.d
Additional .ini files parsed	/etc/php/8.1/apache2/conf.d/10-mysqlnd.ini, /etc/php/8.1/apache2/conf.d/10-opcache.ini, /etc/php/8.1/apache2/conf.d/10-pdo.ini, /etc/php/8.1/apache2/conf.d/15-xml.ini, /etc/php/8.1/apache2/conf.d/20-calendar.ini, /etc/php/8.1/apache2/conf.d/20-ctype.ini, /etc/php/8.1/apache2/conf.d/20-dom.ini, /etc/php/8.1/apache2/conf.d/20-exif.ini, /etc/php/8.1/apache2/conf.d/20-ffi.ini, /etc/php/8.1/apache2/conf.d/20-fileinfo.ini, /etc/php/8.1/apache2/conf.d/20-ftp.ini, /etc/php/8.1/apache2/conf.d/20-

Apache Configuration

The Apache configuration file can enhance security by controlling access permissions, enabling SSL/TLS encryption, and setting up security modules like mod_security to protect against various threats.

Edit the Apache configuration file: **sudo vim /etc/apache2/apache2.conf**

The file below:

- Allows .htaccess overrides in the /var/www/html directory
- Denies access to the README file in the /usr/share/apache2/icons directory
- Disables directory listing for both HTTP and HTTPS connections while hiding Apache version info (ServerTokens Prod, ServerSignature Off)

```
<Directory /var/www/html>
    AllowOverride All
</Directory>

# Restrict access to /icons/README globally
<Directory "/usr/share/apache2/icons">
    <Files "README">
        Require all denied
    </Files>
</Directory>

ServerTokens Prod
ServerSignature Off

<VirtualHost *:80>
    Options -Indexes
</VirtualHost>

<VirtualHost *:443>
    Options -Indexes
</VirtualHost>
```

Security Headers

Security headers in Apache can be configured to protect against common web vulnerabilities by specifying directives like Content-Security-Policy, X-Frame-Options, and Strict-Transport-Security.

Create a security header configuration file:
sudo nano /etc/apache2/conf-available/security-headers.conf

```
GNU nano 2.9.3                                          /etc/apache2/conf-available/sec

<IfModule mod_headers.c>
    # Prevent MIME sniffing
    Header always set X-Content-Type-Options "nosniff"

    # Prevent your site from being framed (clickjacking protection)
    Header always set X-Frame-Options "SAMEORIGIN"

    # Enable XSS protection in older browsers (modern browsers ignore it)
    Header always set X-XSS-Protection "1; mode=block"

    # Protect referrer data
    Header always set Referrer-Policy "strict-origin-when-cross-origin"

    # Minimal Content Security Policy
    Header always set Content-Security-Policy "default-src 'self'; script-src 'self'; style-src 'self';"

    # Enforce HTTPS
    Header always set Strict-Transport-Security "max-age=63072000; includeSubDomains; preload"
</IfModule>
```

Enable the security headers: **sudo a2enconf security-headers**
Restart Apache to apply changes: **sudo systemctl reload apache2**

.htaccess

.htaccess files are configuration files used by Apache servers to control various aspects of web server behavior, including security settings. They can restrict access to certain files or directories, enforce HTTPS, and implement password protection, enhancing the overall security of your website.

Enable the use of .htaccess files by editing the apache configuration file:
sudo nano /etc/apache2/apache2.conf

```
<Directory /var/www/html>
    AllowOverride All
</Directory>
```

Restart Apache to apply changes: **sudo systemctl restart apache2**

Created a .htaccess file in **/var/www/html** in order to deny access to the **/db.php** and /logs/access_logs.txt endpoints: **sudo nano .htaccess**

```
  GNU nano 2.9.3

# Block direct access to db.php
<Files "db.php">
    Require all denied
</Files>

# Block direct access to /logs/access_log.txt
<Files "access_log.txt">
    Require all denied
</Files>
```

Created a .htaccess file in **/var/www/html/img** to deny any code execution, including php, for file uploads: **sudo nano .htaccess**

```
  GNU nano 2.9.3

# Disable PHP execution
<FilesMatch "\.(php|php5|phtml|phar)$">
    Require all denied
</FilesMatch>

# Disable CGI, Perl, and other executable scripts (if enabled on your server)
<FilesMatch "\.(pl|py|cgi|sh)$">
    Require all denied
</FilesMatch>

# Extra hardening: Disable .htaccess overrides in subdirs
AllowOverride None
```

ModSecurity

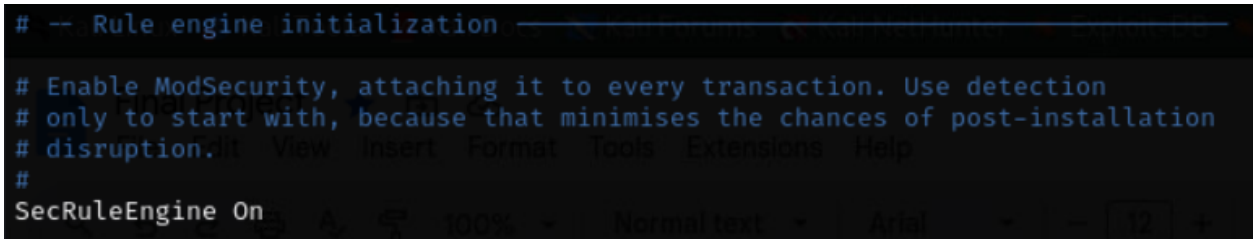
ModSecurity is an open-source web application firewall (WAF) designed to protect Apache web servers from various cyber threats by monitoring and filtering HTTP traffic. Attackers will be blocked by ModSecurity with a 403 error. It operates by using a set of rules to detect and block malicious activities such as SQL injection, cross-site scripting (XSS), and other common web attacks.

ModSecurity Setup
Download and install the ModSecurity Apache module:
sudo apt install libapache2-mod-security2

Check to see what version you have installed (in this case it is 2.9.2-1):
apt-cache show libapache2-mod-security

Edit the ModSecurity configuration file:
sudo nano /etc/modsecurity/modsecurity.conf

Set "**SecRuleEngine DetectionOnly**" to "**SecRuleEngine On**" and then save and quit inside of the modsecurity configuration file



Enable ModSecurity: **sudo a2enmod security2**

Restart Apache to solidify changes: **sudo systemctl restart apache2**

Check if ModSecurity is loaded and active by running: **sudo apachectl -M | grep security2**
Expected Output: **security2_module (shared)**

Download OWASP Core Rule Set (CRS)
To ensure you have the latest ModSecurity rules, you can download the latest ModSecurity Core Rule Set from the Open Web Application Security Project (OWASP)

Download the OWASP Rules: **sudo apt install modsecurity-crs**

Add the CRS: **sudo nano /etc/apache2/mods-enabled/security2.conf**

At the bottom of **security2.conf** replace the line:
IncludeOptional /usr/share/modsecurity-crs/owasp-crs.load”

With the following:
IncludeOptional /etc/modsecurity/crs-setup.conf
IncludeOptional /usr/share/modsecurity-crs/rules/*.conf

This ensures CRS is enabled for ModSecurity.

Restart Apache: **sudo systemctl restart apache2**

Test Apache for syntax errors: **sudo apachectl configtest**
Expected Output: **Syntax OK**

Test the rules: **curl http://localhost/?param=<script>alert('xss')</script>**

```
bob@mint19:~$ curl "http://localhost/?param=<script>alert('xss')</script>"
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access /
on this server.<br />
</p>
<hr>
<address>Apache/2.4.38 (Ubuntu) Server at localhost Port 80</address>
</body></html>
```

ModSecurity sees this request → The Core Rule Set then recognizes <script>alert('xss')</script> as a known malicious pattern (an XSS attempt) → Then ModSecurity blocks it and returns a 403 Forbidden HTTP response (which should be logged)

MySQL Password Protection

Database access is very important for web applications. The CTF9 server has no restrictions on what user can access the MySQL DB as the root user. SQLMap leveraged this vulnerability to access the usernames and passwords of various users on the server. If MySQL is not properly secured with a strong password, it can lead to ongoing issues with broken access control and increased security vulnerabilities.

Logging in as root to MySQL without a password: **mysql -u root**

```
bob@mint19:~$ mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.7.25-0ubuntu0.18.04.2 (Ubuntu)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

The following MySQL commands were used to password protect the root user:
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'ilovesql';
FLUSH PRIVILEGES;

It is best practice to create a highly privileged user, distinct from the root, to handle database calls for the web server. This adds an extra layer of security in case the user account is compromised.

The following commands were used to create a new MySQL user with admin privileges:
CREATE USER 'apache'@'localhost' IDENTIFIED BY 'goodpassword';
GRANT ALL PRIVILEGES ON *.* TO 'apache'@'localhost' WITH GRANT OPTION;
FLUSH PRIVILEGES;

You now have to use a password to sign in with all MySQL users:
mysql -u <username> -p

MySQL Username	Password
root	**ilovesql**
apache	**goodpassword**

PHP MySQLi Upgrade

MySQLi (MySQL Improved) is an extension for PHP that implements secure database interactions by utilizing prepared statements and parameterized queries, which helps prevent SQL injection attacks.

Created a MySQLi connection object in place of the older and insecure mysql_connect() function in the file: **/var/www/html/db.php**

```
GNU nano 2.9.3                                    db.php

<?php
$fp = fopen('/var/www/html/logs/access_log.txt', 'a+');
$today = getdate();
$dstamp = $today['year'] . '-' . $today['mon'] . '-' . $today['mday'] . ' ' . $today['hours'] . ':' . $today['minutes'] . ':' . $today['seconds'];
$ref = isset($_SERVER['HTTP_REFERER']) ? $_SERVER['HTTP_REFERER'] : '';
fwrite($fp, $dstamp . ' ' . rawurldecode($_SERVER['REQUEST_URI']) . ' ' . $_SERVER['HTTP_USER_AGENT'] . ' ' . $ref . "\n");
fclose($fp);

// Database connection (using mysqli)
$mysqli = new mysqli('localhost', 'apache', 'goodpassword', 'lampsec');

// Check for connection errors
if ($mysqli->connect_error) {
    die('Connection error: ' . $mysqli->connect_error);
}

// $mysqli is now your connection object you can use everywhere
?>
```

In order to use MySQLi powered database queries, the grep command was utilized to see where the database connection is referenced in the **/var/www/html** directory:
grep -r "db.php" .

```
bob@mint19:/var/www/html$ grep -r "db.php" .
./admin/index.php:include("../db.php");
./admin/users.php:include("../db.php");
./admin/upload.php:include("../db.php");
./admin/logs.php:include("../db.php");
./admin/add_edit.php:include("../db.php");
./index.php:include_once('db.php');
Binary file ./logs/access_log.txt matches
./static.php:include_once('db.php');
./feedback.php:include_once('db.php');
./contact/contact.php:include("/var/www/html/db.php");
```

The grep command was further utilized to locate exactly where the old and insecure mysql_connect() functions were used: **grep -r "mysql_" .**

Old Queries look like:

- mysql_query()
- mysql_fetch_row()
- mysql_fetch_assoc()

```
bob@mint19:/var/www/html$ grep -r "mysql_" .
./admin/index.php:        mysql_query($sql);
./admin/index.php:        mysql_query($sql);
./admin/index.php:                                        $result = mysql_query('select * from content');
./admin/index.php:                                        while($row = mysql_fetch_assoc($result)) {
./admin/index.php:<?php mysql_close($conn);?>
./admin/users.php:                $result = mysql_query('select user_id, user_name from user');
./admin/users.php:                 while($row = mysql_fetch_assoc($result)) {
./admin/users.php:                      mysql_query('update user set user_name="' . $_POST['name'] . '", ' .
./admin/users.php:                      $result = mysql_query('select user_id, user_name from user');
./admin/users.php:                      while($row = mysql_fetch_assoc($result)) {
./admin/users.php:                      $result = mysql_query('select * from user where user_id = ' . $_GET['id']);
./admin/users.php:                      while ($row = mysql_fetch_assoc($result)) {
./admin/users.php:<?php mysql_close($conn);?>
./admin/upload.php:<?php mysql_close($conn);?>
./admin/logs.php:<?php mysql_close($conn);?>
./admin/add_edit.php:  $result = mysql_query($sql);
./admin/add_edit.php:  while ($cat = mysql_fetch_assoc($result)) {
./admin/add_edit.php:   $result = mysql_query($sql);
./admin/add_edit.php:   while ($row = mysql_fetch_assoc($result)) {
./admin/add_edit.php:<?php mysql_close($conn);?>
./admin/auth.php:       $result = mysql_query($sql);
./admin/auth.php:       $row = mysql_fetch_row($result);
./index.php:                                        $result = mysql_query('select * from user where user_display IS NOT NULL');
./index.php:                                        while ($row = mysql_fetch_assoc($result)) {
./index.php:                              $result = mysql_query('select * from category');
./index.php:                              while ($row = mysql_fetch_assoc($result)) {
./index.php:                              $articles = mysql_query('select a.*,c.category_name from content a, category c where a.category_id = c.category_id');
./index.php:                              while ($row = mysql_fetch_assoc($articles)) {
```

File	MySQLi Upgrade
index.php	Complete
static.php	Complete
db.php	Complete (New DB Connection Logic)
feedback.php	Complete
/admin/index.php	Complete
/admin/auth.php	Complete (Parameterized Queries)
/admin/users.php	Complete (Parameterized Queries)
/admin/upload.php	Complete (Sanitized Inputs)
/admin/logs.php	Complete
/admin/add_edit.php	Complete
/contact/contact.php	Complete (Parameterized Queries)

MySQLi upgrade for the file **db.php:**

- Old connection object: $conn
- New connection object: **$mysqli**

```php
<?php
$fp = fopen('/var/www/html/logs/access_log.txt', 'a+');
$today = getdate();
$dstamp = $today['year'] . '-' . $today['mon'] . '-' . $today['mday'] . ' ' . $today['hours'] . ':' . $today['minutes'] . ':' . $today['seconds'];
$ref = isset($_SERVER['HTTP_REFERER']) ? $_SERVER['HTTP_REFERER'] : '';
fwrite($fp, $dstamp . ' ' . rawurldecode($_SERVER['REQUEST_URI']) . ' ' . $_SERVER['HTTP_USER_AGENT'] . ' ' . $ref . "\n");
fclose($fp);

// Database connection (using mysqli)
$mysqli = new mysqli('localhost', 'apache', 'goodpassword', 'lampsec');

// Check for connection errors
if ($mysqli->connect_error) {
    die('Connection error: ' . $mysqli->connect_error);
}

// $mysqli is now your connection object you can use everywhere
?>
```

Sample upgrade from **mysql** to **mysqli** statements:

- mysql_query(‘query’) → mysqli_query($mysqli, ‘query’);
- mysql_fetch_assoc($result) → mysqli_fetch_assoc($result)

MySQLi upgrade for the file **/admin/users.php:**

```php
$result = mysqli_query($mysqli, 'select user_id, user_name from user');
while($row = mysqli_fetch_assoc($result)) {
    print '<li><a href="users.php?id=' . $row['user_id'] . '">' . $row['user_name'] . '</a>';
    print '</li>';
}
```

SQL Injection Prevention

Parameterized queries in MySQLi prevent SQL injection by ensuring that user inputs are treated as data rather than executable code.

MySQLi utilizes **? as placeholders for SQL arguments** and later binds parameters to data types with the function **bind_param()**. This approach ensures that user inputs are safely handled and properly escaped, preventing SQL injection attacks by treating inputs as data rather than executable code.

Below shows the addition of parameterized queries for user inputs in the mysqli overhaul:

/admin/auth.php - Login Form

Before

```php
<?php
// Make sure the user is logged in
$message = 'You need to log in!';
#syslog(LOG_INFO, 'Auth included.');
if (isset($_POST['username']) && isset($_POST['password'])) {
    syslog(LOG_INFO, 'User login attempt detected.');
    $sql = 'SELECT user_id FROM user WHERE ' .
           'user_name = "' . $_POST['username'] . '" AND ' .
           'user_password = MD5("' . $_POST['password'] . '")';
```

After

```php
<?php
// Make sure the user is logged in
$message = 'You need to log in!';
#syslog(LOG_INFO, 'Auth included.');
if (isset($_POST['username']) && isset($_POST['password'])) {
    syslog(LOG_INFO, 'User login attempt detected.');

    // Prepare the SQL statement
    $sql = 'SELECT user_id FROM user WHERE user_name = ? AND user_password = MD5(?)';

    // Prepare the statement
    if ($stmt = mysqli_prepare($mysqli, $sql)) {
        // Bind the user input to the prepared statement
        mysqli_stmt_bind_param($stmt, 'ss', $_POST['username'], $_POST['password']);

        // Execute the statement
        mysqli_stmt_execute($stmt);

        // Bind the result
        mysqli_stmt_bind_result($stmt, $id);
```

/admin/users.php - Edit User Data Form

Before

```php
if (isset($_GET['update']) && $_GET['update'] == 'yes') {
        $picture = '';
        // Upload a new picture
        if ($_FILES["picture"]["error"] == 0) {
                move_uploaded_file($_FILES['picture']['tmp_name'], '/var/www/html/img/team/' . $_FILES['picture']['name']);
                $picture = $_FILES['picture']['name'];
        }
        mysql_query('update user set user_name="' . $_POST['name'] . '", ' .
                        'user_display="' . $_POST['display'] . '", ' .
                        'user_jobtitle="' . $_POST['title'] . '", ' .
                        'user_picture="' . $picture . '" ' .
                        'where user_id = ' . $_GET['id']);
        $result = mysql_query('select user_id, user_name from user');
        while($row = mysql_fetch_assoc($result)) {
                print '<li><a href="users.php?id=' . $row['user_id'] . '">' . $row['user_name'] . '</a>';
                print '</li>';
        }
}
```

After

```php
if (isset($_GET['update']) && $_GET['update'] == 'yes') {
        $picture = '';
        // Upload a new picture
        if ($_FILES["picture"]["error"] == 0) {
                move_uploaded_file($_FILES['picture']['tmp_name'], '/var/www/html/img/team/' . $_FILES['picture']['name']);
                $picture = $_FILES['picture']['name'];
        }

        // Prepare the update query
        $stmt = mysqli_prepare($mysqli, 'UPDATE user SET user_name = ?, user_display = ?, user_jobtitle = ?, user_picture = ? WHERE user_id = ?');
        if ($stmt) {
                // Bind parameters
                mysqli_stmt_bind_param($stmt, 'ssssi', $_POST['name'], $_POST['display'], $_POST['title'], $picture, $_GET['id']);

                // Execute the query
                mysqli_stmt_execute($stmt);

                // Close the statement
                mysqli_stmt_close($stmt);
        }

        // Query to fetch user names
        $result = mysqli_query($mysqli, 'SELECT user_id, user_name FROM user');
        while ($row = mysqli_fetch_assoc($result)) {
                print '<li><a href="users.php?id=' . $row['user_id'] . '">' . $row['user_name'] . '</a>';
                print '</li>';
        }
```

NOTE: The image upload will be updated to prevent malicious files in the next section!

/contact/contact.php - Contact Form

Before

```php
<?php
$to      = 'justin@localhost';
$subject =  $_POST['subject'];
$message = $_POST['name'] . ' writes: ' . "\r\n" .  $_POST['message'];
$headers = 'From: ' . $_POST['email'] . "\r\n" .
    'Reply-To: ' . $_POST['email'] . "\r\n" .
    'X-Mailer: PHP/' . phpversion();

mail($to, $subject, $message, $headers);
include("/var/www/html/db.php");
$sql = 'insert into contact ' .
                        'set contact_name="' . $_POST['name'] . '", ' .
                        'contact_email="' . $_POST['email'] . '", ' .
                        'contact_subject="' . $_POST['subject'] . '", ' .
                        'contact_message="' . $_POST['message'] . '"';
mysql_query($sql);
?>
```

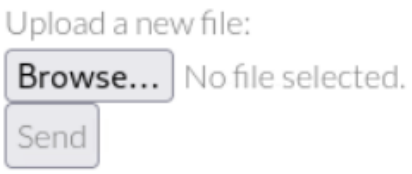
After

```php
<?php
$to      = 'justin@localhost';
$subject = $_POST['subject'];
$message = $_POST['name'] . ' writes: ' . "\r\n" . $_POST['message'];
$headers = 'From: ' . $_POST['email'] . "\r\n" .
    'Reply-To: ' . $_POST['email'] . "\r\n" .
    'X-Mailer: PHP/' . phpversion();

mail($to, $subject, $message, $headers);

// Include database connection
include("/var/www/html/db.php");

// Prepare the parameterized insert query
$sql = 'INSERT INTO contact (contact_name, contact_email, contact_subject, contact_message) VALUES (?, ?, ?, ?)';

$stmt = mysqli_prepare($mysqli, $sql);

if ($stmt) {
    // Bind parameters and execute the query
    mysqli_stmt_bind_param($stmt, 'ssss', $_POST['name'], $_POST['email'], $_POST['subject'], $_POST['message']);
    mysqli_stmt_execute($stmt);
    mysqli_stmt_close($stmt);
} else {
    // Handle the error if the statement preparation fails
    echo 'Error preparing query: ' . mysqli_error($mysqli);
}
?>
```

File Upload Sanitization

The CTF9 web server's admin portal struggles with file upload sanitization. There are two pages within the admin portal which allow file uploads. Originally, neither page's source code verifies the file uploaded by a user. This allowed for uploads of reverse shells during pentesting. In order to prevent malicious files from being uploaded to the server, **/admin/upload.php** and **/admin/users.php** will be refactored to sanitize file uploads and verify contents to be safe and accurate. **Both of these files execute server-side making it hard for attackers to bypass the checks.**

The page http://10.161.7.41/admin/upload.php allows users to upload image files to the server.



The file **/var/www/html/admin/upload.php** was refactored to ensure uploaded files are sanitized and verified to be images. No other files will be allowed to be uploaded. If a false image is uploaded, an additional check was put into place to scan for common malicious code in php, python, and bash scripts.

MIME is a standardized way to describe the type and format of a file's content so systems know how to handle it correctly. This software was utilized to check a file's real file type. It is very common for attackers to disguise malicious scripts as "harmless" text or image files. If the MIME does not align with the allowed file types, the file is not able to be uploaded.

Allowed Files: jpg, jpeg, png, gif, webp

```php
if (isset($_GET['upload'])) {
  if (is_uploaded_file($_FILES['imagefile']['tmp_name'])) {

    $file_tmp  = $_FILES['imagefile']['tmp_name'];
    $file_name = basename($_FILES['imagefile']['name']);
    $file_ext  = strtolower(pathinfo($file_name, PATHINFO_EXTENSION));

    // Allowed image extensions and MIME types
    $allowed_extensions = ['jpg', 'jpeg', 'png', 'gif', 'webp'];
    $allowed_mime_types = ['image/jpeg', 'image/png', 'image/gif', 'image/webp'];

    // Get MIME type using finfo
    $finfo = new finfo(FILEINFO_MIME_TYPE);
    $mime  = $finfo->file($file_tmp);

    if (!in_array($file_ext, $allowed_extensions) || !in_array($mime, $allowed_mime_types)) {
        echo "Invalid file type. <br>";
        echo "Detected extension: " . $file_ext . "<br>";
        echo "Detected MIME type: " . $mime . "<br>";

        exit;
    }
```

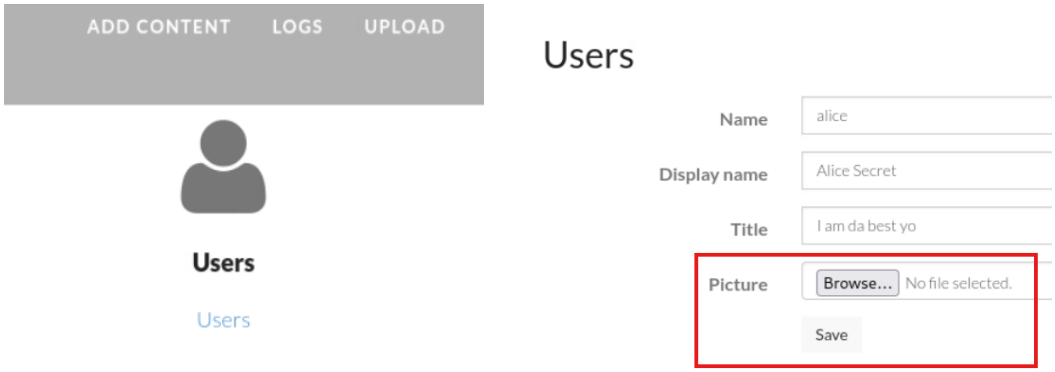
If an uploaded file was falsely thought to be an image, an additional check was put into place to scan for common malicious code in php, python, or bash scripts. If malicious content was found, the server replies with a message warning the user.

```php
// Read file contents
$contents = file_get_contents($file_tmp);

// Look for common malicious patterns
$malicious_patterns = [
  '/<\?php/i',              // PHP code
  '/<script/i',             // HTML/JS
  '/#!\/usr\/bin\/python/i', // Python shebang
  '/import\s+os/i',         // Python: OS module
  '/import\s+sys/i',        // Python: SYS module
  '/eval\s*\(/i',           // Potential code execution
  '/exec\s*\(/i',           // Python or Bash
  '/#!\/bin\/bash/i',       // Bash shebang
  '/\b(cat|ls|rm|mv|cp|wget|curl)\b/i', // Common shell commands
];

foreach ($malicious_patterns as $pattern) {
  if (preg_match($pattern, $contents)) {
    echo "File contains potentially malicious code.";
    exit;
  }
}


$upload_dir = '/var/www/html/img/team/';
$upload_path = $upload_dir . $file_name;

if (move_uploaded_file($file_tmp, $upload_path)) {
  //echo "Upload successful.";
} else {
```

Sample Error Message → Tried to upload a text file

Invalid file type.
Detected extension: txt
Detected MIME type: text/plain

The page http://10.161.7.41/admin/users.php also allows for users to upload profile pictures.



The file **/var/www/html/users.php** was refactored to ensure uploaded files are sanitized and verified to be images. No other files will be allowed to be uploaded. If a false image is uploaded, an additional check was put into place to scan for common malicious code in php, python, and bash scripts.

The php code for users.php is almost identical to upload.php.

MIME Extension Check

```php
// Upload a new picture
if (isset($_FILES['picture']) && $_FILES["picture"]["error"] === 0) {
    $file_tmp  = $_FILES['picture']['tmp_name'];
    $file_name = basename($_FILES['picture']['name']);
    $file_ext  = strtolower(pathinfo($file_name, PATHINFO_EXTENSION));

    $allowed_extensions = ['jpg', 'jpeg', 'png', 'gif', 'webp'];
    $allowed_mime_types = ['image/jpeg', 'image/png', 'image/gif', 'image/webp'];

    $finfo = new finfo(FILEINFO_MIME_TYPE);
    $mime  = $finfo->file($file_tmp);

    if (!in_array($file_ext, $allowed_extensions) || !in_array($mime, $allowed_mime_types)) {
        echo "Invalid file type. <br>";
        echo "Detected extension: " . $file_ext . "<br>";
        echo "Detected MIME type: " . $mime . "<br>";
        exit;
    }
```

Malicious Content Check

```php
    $contents = file_get_contents($file_tmp);
    $malicious_patterns = [
        '/<\?php/i',
        '/<script/i',
        '/#!\/usr\/bin\/python/i',
        '/import\s+os/i',
        '/import\s+sys/i',
        '/eval\s*\(/i',
        '/exec\s*\(/i',
        '/#!\/bin\/bash/i',
        '/\b(cat|ls|rm|mv|cp|wget|curl)\b/i',
    ];

    foreach ($malicious_patterns as $pattern) {
        if (preg_match($pattern, $contents)) {
            echo "File contains potentially malicious code.";
            exit;
        }
    }

    $upload_dir = '/var/www/html/img/team/';
    $upload_path = $upload_dir . $file_name;

    if (move_uploaded_file($file_tmp, $upload_path)) {
        $picture = $file_name;
    } else {
        echo "Failed to move uploaded file.";
    }
} else {
    echo "No valid uploaded file detected.";
```

Sample Error Message → Tried to upload a PHP script with a .png extension

Users

Invalid file type.

Detected extension: png

Detected MIME type: text/x-php

UFW

UFW (Uncomplicated Firewall) is a **host-based** firewall configuration tool designed to simplify the process of managing firewall rules. It provides a user-friendly command-line interface for configuring common firewall use cases, making it accessible for regular users without requiring them to write complex rules manually. Upon installation, UFW is active by default, which may initially block some network traffic until properly configured.

Install UFW: **sudo apt install ufw**

Test to see if UFW is enabled: **sudo ufw status**
Activate the firewall: **sudo ufw enable**
Disable the firewall: **sudo ufw disable**
Delete a firewall rule: **sudo ufw delete allow <port_number>**

Since CTF9 hosts a web service, ports **80** and **443** must remain open. Additionally, external access to the SSH service requires enabling traffic on port **22**.

Blocking
A specific IP address: **sudo ufw deny from <ip address>**
A full subnet: **sudo ufw deny from <ip address/subnet>**

Allow
Allow an IP address: **sudo ufw allow from <ip address>**
Allow a port: **sudo ufw allow <port number>**

Finalized Firewall Rules:

```
bob@mint19:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
80                         ALLOW       Anywhere
443                        ALLOW       Anywhere
22                         ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
80 (v6)                    ALLOW       Anywhere (v6)
443 (v6)                   ALLOW       Anywhere (v6)
22 (v6)                    ALLOW       Anywhere (v6)
```

Chapter 5: Final Vulnerability Report

The original vulnerability scanning methods were employed for a second time after securing the CTF9 server. Note that this round of testing has been done from inside the pfSense LAN where the server is now assigned. Numerous previously identified vulnerabilities have been successfully patched. **Outstanding vulnerabilities will be assessed based on severity, while patched vulnerabilities will be documented with the methods used.**

Vulnerability Severity

Low	Medium	High

Nmap

Vulnerability Scan Results

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV 192.168.1.10
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-30 18:19 EDT
Nmap scan report for 192.168.1.10
Host is up (0.00016s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd 2.4.38 ((Ubuntu))
MAC Address: 00:50:56:82:12:CA (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.10 seconds
```

Vulnerability	Description
OpenSSH 7.6p1 Ubuntu 4ubuntu0.1	SSH software was left unchanged because of its reliable service availability. The MySQL database has been better secured to prevent sensitive data leaks that could lead to malicious SSH logins.
Apache httpd 2.4.38	Though the current Apache version is out of date, preserving it is crucial for server **compatibility** with pre-existing services and modules. Attempting to upgrade or modify it could introduce functionality issues or break dependencies. However, since we are fully aware of these risks, we are in a better position to monitor the service closely and implement mitigation strategies to reduce the likelihood of exploitation.

Nikto

Vulnerability Scan Results:

```
┌──(kali㉿kali)-[~]
└─$ nikto -h http://192.168.1.10
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          192.168.1.10
+ Target Hostname:    192.168.1.10
+ Target Port:        80
+ Start Time:         2025-05-02 16:09:18 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.38 (Ubuntu)
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.38 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is
+ /admin/login.php?action=insert&username=test&password=test: phpAuction may allow user admin
est' to verify. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-0995
+ /admin/: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla
+ /admin/login.php: Admin login page/section found.
+ 8102 requests: 0 error(s) and 4 item(s) reported on remote host
+ End Time:         2025-05-02 16:09:39 (GMT-4) (21 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Vulnerability	Description
Apache/2.4.38	Version 2.4.38 is outdated. Preserving the current Apache version is crucial for server **compatibility** with pre-existing services and modules.
~~/test.php~~	This vulnerability was mitigated by disabling the server's test.php file.
/admin/login.php	This endpoint is necessary for administrative logins and website management. The user input fields for sign in and file uploads have been sanitized and retooled to prevent SQL injection and malicious file uploads.
/admin ~~/img~~ ~~/css~~ ~~/logs~~ ~~/db.php~~ ~~/icons/README~~	All web endpoints but /admin deny user access with the use of ModSecurity and an improved Apache configuration file.

Nessus

Vulnerability Scan Results:

Sev	CVSS	VPR	EPSS	Name	Family	Count
CRITICAL	10.0			Canonical Ubuntu Linux SEoL (18.04.x)	General	1
MIXED	...	...	...	Apache Httpd (Multiple Issues)	Web Servers	23
MIXED	...	...	...	Apache HTTP Server (Multiple Issues)	Web Servers	2
MEDIUM	6.1	5.7	0.0914	jQuery 1.2 < 3.5.0 Multiple XSS	CGI abuses : XSS	1
MIXED	...	...	...	Openbsd Openssh (Multiple Issues)	Misc.	6
LOW	2.1 *	2.2	0.0037	ICMP Timestamp Request Remote Date Dis...	General	1
INFO	...	...	...	HTTP (Multiple Issues)	Web Servers	2
INFO	...	...	...	SSH (Multiple Issues)	Misc.	2

Vulnerability	Description
Canonical Ubuntu Linux SEoL (18.04.x)	Upgrading from Ubuntu Linux 18.04 may cause compatibility issues with Apache, MySQL, and other software, potentially disrupting existing configurations and functionalities. We recognize the risks associated with using a deprecated operating system and have implemented additional security practices to harden the hosted services, ensuring secure access.
Apache 2.4.x Multiple Vulnerabilities	As stated previously, preserving the current Apache version is crucial for server compatibility with pre-existing services and modules. We are fully aware of the risks associated with using an older Apache version and will focus on hardening existing modules and plugins to ensure secure operation.

DirBuster

Vulnerability Scan Results:

```
┌──(kali㉿kali)-[~]
└─$ dirbuster
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Starting OWASP DirBuster 1.0-RC1
Starting dir/file list based brute forcing
Dir found: /img/ - 500
DirBuster Stopped
```

Vulnerability	Description
/img/*	The /img directory contains images publicly available on the website. Having the ability to access this directory via the browser is not a large security concern
~~/logs/access_log.txt~~	User access to this endpoint was denied with the use of ModSecurity and an improved Apache configuration file.
~~/admin/login.php~~ ~~/admin/upload.php~~ ~~/admin/users.php~~ ~~/admin/auth.php~~ ~~/admin/add_edit.php~~	The admin directory was not able to be found by DirBuster despite being in the wordlist used. The directory was most likely hidden with the use of ModSecurity and an improved Apache configuration file.
~~/js/*~~	User access to this endpoint was denied with the use of ModSecurity and an improved Apache configuration file.

SQLMap

The same command from initial testing was used: **sqlmap -u "http://192.168.1.10/admin" -data "username=admin&password=admin" -p password -D lampsec -T user -C user_name,user_password --dump**

Pentesting Results:

```
it is recommended to perform only basic UNION tests if there is not at least on
[17:36:18] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[17:36:18] [CRITICAL] unable to connect to the target URL. sqlmap is going to r
[17:36:18] [WARNING] most likely web server instance hasn't recovered yet from
ion '--technique' (e.g. '--flush-session --technique=BEUS') or try to lower the
[17:36:18] [WARNING] POST parameter 'password' does not seem to be injectable
[17:36:18] [CRITICAL] all tested parameters do not appear to be injectable. Try
me kind of protection mechanism involved (e.g. WAF) maybe you could try to use
[17:36:18] [WARNING] HTTP error codes detected during run:
403 (Forbidden) - 75 times

[*] ending @ 17:36:18 /2025-05-02/
```

Since all database queries were parameterized, the SQL Map program detected that SQL injection was not possible and was unable to access sensitive user information identified during initial testing.

Server-Side Input Validation

SQL Injection

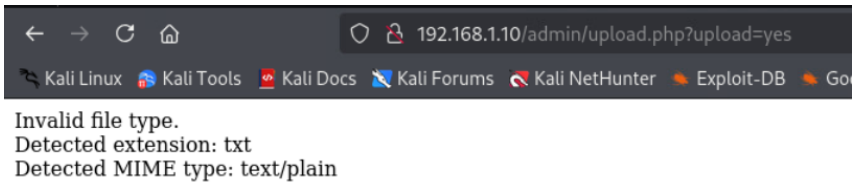
As demonstrated above, SQL Injection was prevented by updating Apache PHP to version 8.1 and using MYSQLi to rewrite all database queries. User input queries were parameterized to prevent injection on the server.

```php
<?php
// Make sure the user is logged in
$message = 'You need to log in!';
#syslog(LOG_INFO, 'Auth included.');
if (isset($_POST['username']) && isset($_POST['password'])) {
    syslog(LOG_INFO, 'User login attempt detected.');

    // Prepare the SQL statement
    $sql = 'SELECT user_id FROM user WHERE user_name = ? AND user_password = MD5(?)';

    // Prepare the statement
    if ($stmt = mysqli_prepare($mysqli, $sql)) {
        // Bind the user input to the prepared statement
        mysqli_stmt_bind_param($stmt, 'ss', $_POST['username'], $_POST['password']);

        // Execute the statement
        mysqli_stmt_execute($stmt);

        // Bind the result
        mysqli_stmt_bind_result($stmt, $id);
```

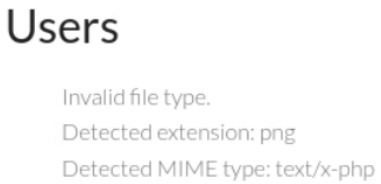
File Sanitization

PHP code was written for **/admin/users.php** and **/admin/upload.php** to sanitize file uploads by only allowing image files and analyzing file contents for malicious payloads.

/admin/users.php → Invalid file upload



/admin/upload.php → Invalid file upload

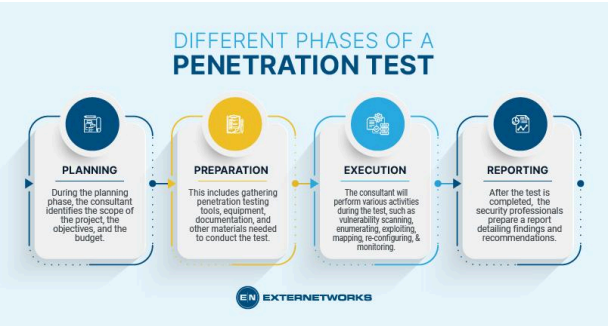


Flawed HTTP Redirect Logic

Vulnerability	Description
HTTP Redirect Logic	The Burp Suite exploit can still be used to bypass authentication and get access to the admin page in the browser. Fixing this error will require more php knowledge so session management can be reprogrammed. See *Chapter 6: Future Work* for more details.

Chapter 6: Future Work

Snort



Given additional time, further testing of **Snort's** rule management such as activating or deactivating specific detection rules would have been beneficial for refining detection and response strategies. Developing custom rules tailored to specific monitoring needs could have enhanced security by targeting particular types of traffic or providing better insight during potential attacks. The testing process would also have gained value from incorporating penetration testing, ensuring that Snort's configurations effectively detect and block malicious activity. Overall, Snort proves to be a valuable tool, enhancing the capabilities of the pfSense firewall while serving as an effective alert system for cybersecurity defense.

Logging Software

Future work also includes the implementation of detailed logging alerts and utilizing logging software like **syslogd** can significantly enhance the consolidation of logs from various services and website functions into a single location. Centralized log visibility facilitates comprehensive system monitoring, enabling quick detection of malicious activities and prompt responses to patch vulnerabilities and defend against threats.

HTTP Redirect Fix

Another priority for future work involves **patching the HTTP Redirect logic for the admin page**. Currently, attackers can drop the Location header when redirected from failed admin logins and bypass authentication to access the admin portal. Addressing this vulnerability will require more advanced PHP knowledge to reprogram session management and HTTP redirects for admin web pages.

Bibliography

NMCLI Configuration Commands:
https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/7/html/networking_guide/sec-configuring_ip_networking_with_nmcli#sec-Adding_and_Configuring_a_Dynamic_Ethernet_Connection_with_nmcli
Last Accessed APR23

Modsecurity:
https://www.inmotionhosting.com/support/server/apache/install-modsecurity-apache-module/
Last Accessed APR30

How to use Snort: https://www.fortinet.com/resources/cyberglossary/snort
Last Accessed APR25

Snort Basic Setup: https://www.youtube.com/watch?v=SapAcfHbQSE
Last Accessed APR25

MySQLi Documentation: https://www.php.net/manual/en/book.mysqli.php
Last Accessed APR27

Apache HTTP Security Headers:
https://stackoverflow.com/questions/53700968/how-to-add-http-security-headers
Last Accessed MAY02

SQL Map: https://hackertarget.com/sqlmap-tutorial/
Last Accessed APR25

UFW:
https://www.digitalocean.com/community/tutorials/ufw-essentials-common-firewall-rules-and-commands
Last Accessed APR30