Securing CTF 9

Technical Documentation

Prepared by: Ryan Paradee, Hoo Won, Max Riesberg
Date of Initial Preparation:
Revision Date(s): N/A

Overview

In this “Securing CTF 9” final project for system security (CS 4100) here at The University of Northern Iowa, our group will be securing the CTF9 virtual machine that we penetrated in project 4 by fixing problems already present on the system and adding another layer of security. Some of the enhancements include; adding modsecurity to apache, configuring pfSense to work with CTF9 and move the machine “behind it,” and updating the php code to version 8.1 and secure it using the mysqli library. There will also be additional measures put in place such as SQL injection prevention for the website.

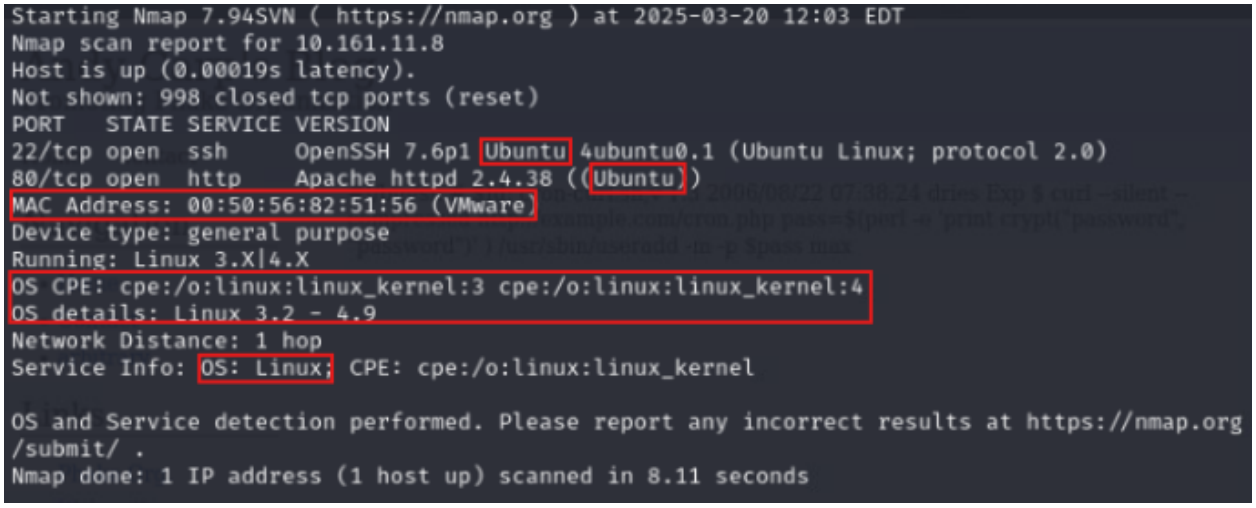
Table of Contents

Chapter 1: Summary of Services on CTF9

CTF 9 is a Linux based server that has multiple outdated services running, which present security risks. This section will touch on CTF9's operating system as well as installed service versions.

Operating System

With the use of nmap, we can see the CTF9 machine is running a version of Linux, specifically **Ubuntu Linux** and some other useful information:



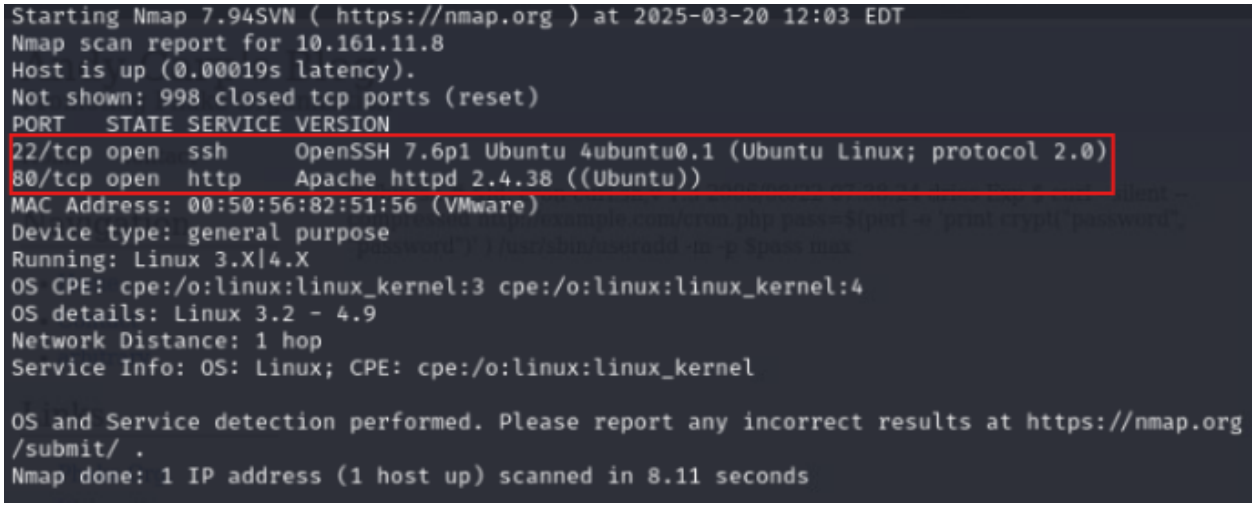
According to nmap fingerprinting, the kernel version is likely **3.x** with an architecture of **x86_64**. The machine's MAC address vendor is **VMware**, indicating its running in a virtualized environment.

Installed Services

There are two types of services covered in this part; services exposed to the network, and services that require login to the server to uncover which also provide what version of the service is on the machine.

Services Exposed to the Network

Before logging in as root, we will see what we can fingerprint with the use of nmap, a network scanner created by Gordon Lyon. We can uncover some of the services, the version of the service, and what port they are open on:

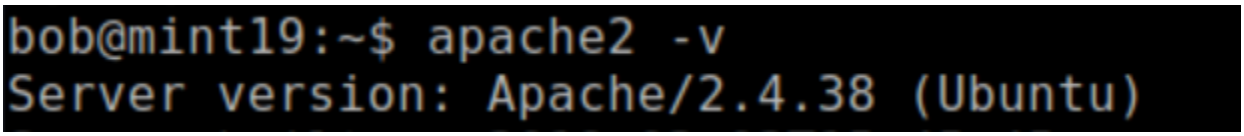


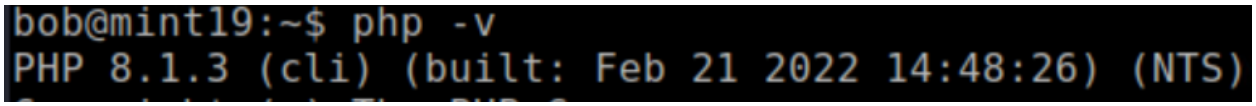
Breakdown of the nmap findings:

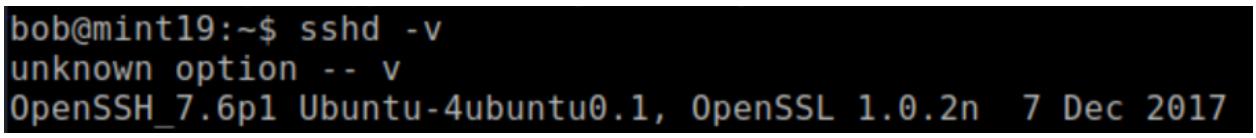
- OpenSSH service open on port 22, version 7.6p1, protocol 2.0
- Apache (HTTP) service open on port 80, version 2.4.38

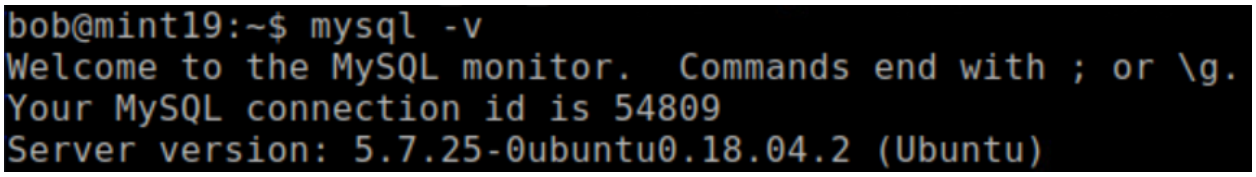
Other Services

With the permission from the server owner, we logged in as Bob and ran a few commands to check the version of the services.









Breakdown of the findings:

- Verification of Apache version 2.4.38
- PHP version 5.6.40
- MySQL version 5.7.25
- OpenSSL version 1.0.2n

Summary

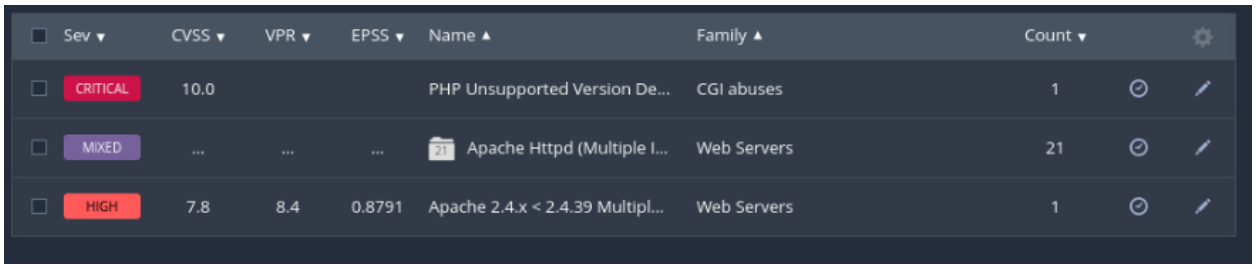
The CTF9 target machine is running a Linux-based operating system, specifically Ubuntu. This server is part of a LAMP stack and currently runs Apache HTTP web server, PHP, MySQL. Overall, the system exposes a variety of services and software versions that are outdated and potentially vulnerable, making them high-priority targets for patching or upgrading in the next chapter!

Chapter 2: Initial Vulnerability Report

This section will touch on vulnerability scans using tools such as Nessus and the vulnerabilities found on the CTF9 machine.

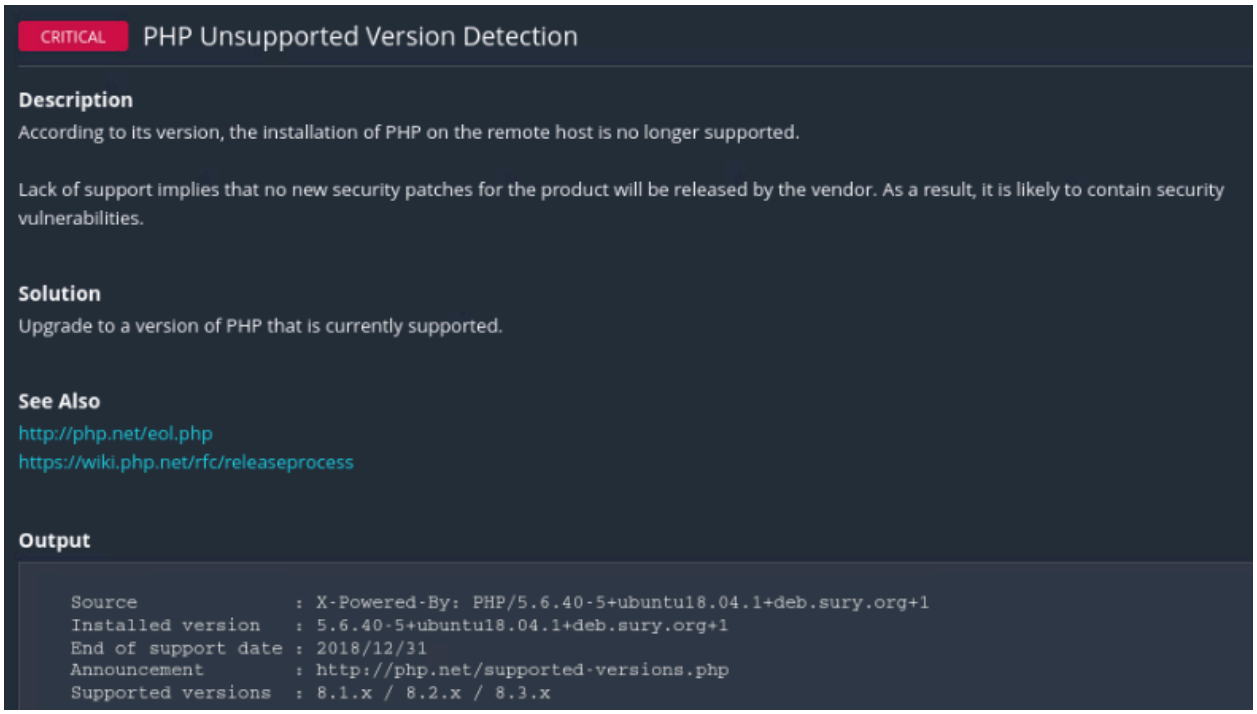
Nessus

After running a scan for vulnerabilities in CTF9 with Nessus, you can see a report like this:



Sev	CVSS	VPR	EPSS	Name	Family	Count
CRITICAL	10.0			PHP Unsupported Version De...	CGI abuses	1
MIXED	...	...	...	Apache Httpd (Multiple I...	Web Servers	21
HIGH	7.8	8.4	0.8791	Apache 2.4.x < 2.4.39 Multipl...	Web Servers	1

As mentioned above, the first vulnerability in the report tells us that the PHP version installed on the CTF9 machine is unsupported and could likely lead to security vulnerabilities.



CRITICAL PHP Unsupported Version Detection

Description

According to its version, the installation of PHP on the remote host is no longer supported.

Lack of support implies that no new security patches for the product will be released by the vendor. As a result, it is likely to contain security vulnerabilities.

Solution

Upgrade to a version of PHP that is currently supported.

See Also

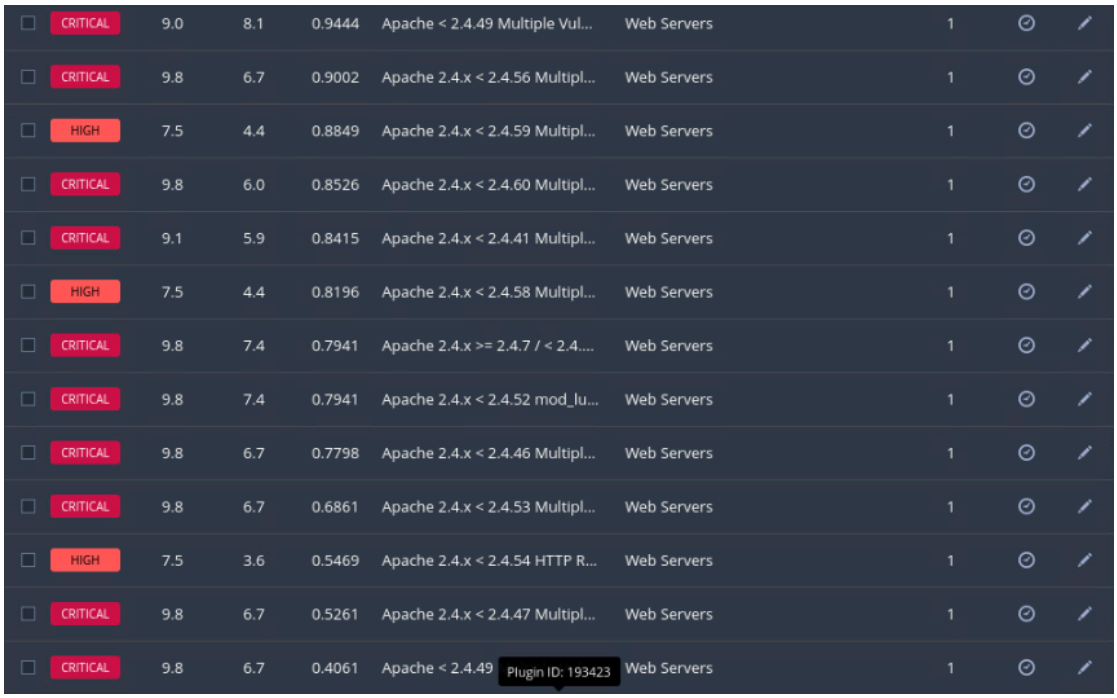
http://php.net/eol.php
https://wiki.php.net/rfc/releaseprocess

Output

```
Source              : X-Powered-By: PHP/5.6.40-5+ubuntu18.04.1+deb.sury.org+1
Installed version   : 5.6.40-5+ubuntu18.04.1+deb.sury.org+1
End of support date : 2018/12/31
Announcement        : http://php.net/supported-versions.php
Supported versions  : 8.1.x / 8.2.x / 8.3.x
```

In the output box under source, we see the message “X-Powered-By: PHP/5.6.40-5+ubuntu18.04.1+deb.sury.org+1” which tells us that the host has PHP version 5.6.40. Attackers can look up vulnerabilities and exploits for this PHP version to use.

The second vulnerability from the report is a folder full of Apache version vulnerabilities that we can analyze:

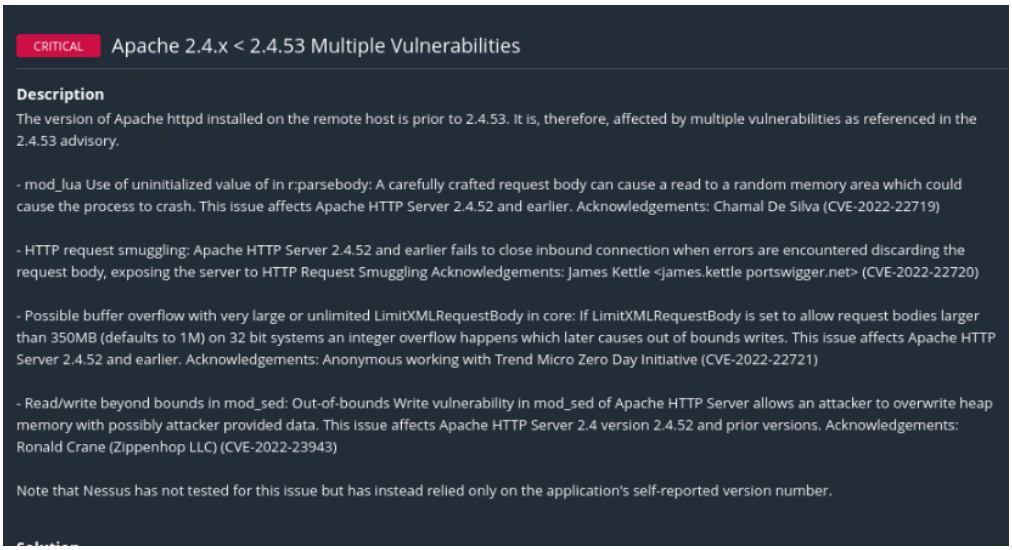


☐	CRITICAL	9.0	8.1	0.9444	Apache < 2.4.49 Multiple Vul...	Web Servers	1	⊙	✎
☐	CRITICAL	9.8	6.7	0.9002	Apache 2.4.x < 2.4.56 Multipl...	Web Servers	1	⊙	✎
☐	HIGH	7.5	4.4	0.8849	Apache 2.4.x < 2.4.59 Multipl...	Web Servers	1	⊙	✎
☐	CRITICAL	9.8	6.0	0.8526	Apache 2.4.x < 2.4.60 Multipl...	Web Servers	1	⊙	✎
☐	CRITICAL	9.1	5.9	0.8415	Apache 2.4.x < 2.4.41 Multipl...	Web Servers	1	⊙	✎
☐	HIGH	7.5	4.4	0.8196	Apache 2.4.x < 2.4.58 Multipl...	Web Servers	1	⊙	✎
☐	CRITICAL	9.8	7.4	0.7941	Apache 2.4.x >= 2.4.7 / < 2.4....	Web Servers	1	⊙	✎
☐	CRITICAL	9.8	7.4	0.7941	Apache 2.4.x < 2.4.52 mod_lu...	Web Servers	1	⊙	✎
☐	CRITICAL	9.8	6.7	0.7798	Apache 2.4.x < 2.4.46 Multipl...	Web Servers	1	⊙	✎
☐	CRITICAL	9.8	6.7	0.6861	Apache 2.4.x < 2.4.53 Multipl...	Web Servers	1	⊙	✎
☐	HIGH	7.5	3.6	0.5469	Apache 2.4.x < 2.4.54 HTTP R...	Web Servers	1	⊙	✎
☐	CRITICAL	9.8	6.7	0.5261	Apache 2.4.x < 2.4.47 Multipl...	Web Servers	1	⊙	✎
☐	CRITICAL	9.8	6.7	0.4061	Apache < 2.4.49	Web Servers	1	⊙	✎

Plugin ID: 193423

As you can see, the Apache version on CTF9 is full of critical and high risk vulnerabilities. These vulnerabilities can range from vulnerabilities that allow attackers to manipulate requests, bypass security controls, access internal resources, exploit backend services, etc.

Vulnerability example from report folder:



CRITICAL Apache 2.4.x < 2.4.53 Multiple Vulnerabilities

Description

The version of Apache httpd installed on the remote host is prior to 2.4.53. It is, therefore, affected by multiple vulnerabilities as referenced in the 2.4.53 advisory.

- mod_lua Use of uninitialized value of in r:parsebody: A carefully crafted request body can cause a read to a random memory area which could cause the process to crash. This issue affects Apache HTTP Server 2.4.52 and earlier. Acknowledgements: Chamal De Silva (CVE-2022-22719)

- HTTP request smuggling: Apache HTTP Server 2.4.52 and earlier fails to close inbound connection when errors are encountered discarding the request body, exposing the server to HTTP Request Smuggling Acknowledgements: James Kettle <james.kettle portswigger.net> (CVE-2022-22720)

- Possible buffer overflow with very large or unlimited LimitXMLRequestBody in core: If LimitXMLRequestBody is set to allow request bodies larger than 350MB (defaults to 1M) on 32 bit systems an integer overflow happens which later causes out of bounds writes. This issue affects Apache HTTP Server 2.4.52 and earlier. Acknowledgements: Anonymous working with Trend Micro Zero Day Initiative (CVE-2022-22721)

- Read/write beyond bounds in mod_sed: Out-of-bounds Write vulnerability in mod_sed of Apache HTTP Server allows an attacker to overwrite heap memory with possibly attacker provided data. This issue affects Apache HTTP Server 2.4 version 2.4.52 and prior versions. Acknowledgements: Ronald Crane (Zippenhop LLC) (CVE-2022-23943)

Note that Nessus has not tested for this issue but has instead relied only on the application's self-reported version number.

Each vulnerability is given an EPSS score which is a numerical value (ranging from 0-1) that represents the probability of the vulnerability being exploited in the next thirty days. This specific vulnerability has an EPSS score of 69% and could lead to crashes, HTTP request smuggling, allowing out of bound writes, and allowing overwriting of the heap memory with possibly malicious data.

Nikto

Nikto is an open-source web server scanner used to identify vulnerabilities and security issues on web servers.

Below is the scan result from Nikto for CTF9:

We can see here that Nikto has found some important high risk issues. Just like we observed before, we can see that the server is running Apache version 2.4.38, which is an outdated version known to have vulnerabilities as seen in our Nessus scan. We can also see that Nikto ran a test on PHP which found information that revealed server configurations, settings, and loaded modules. This could help attackers identify vulnerabilities and tailor attacks.

Nikto also found that phpAuction may allow user admin accounts to be inserted without proper authentication. One last thing to note is that the PHPSESSID is created without the httponly flag which makes it vulnerable to cross site scripting attacks.

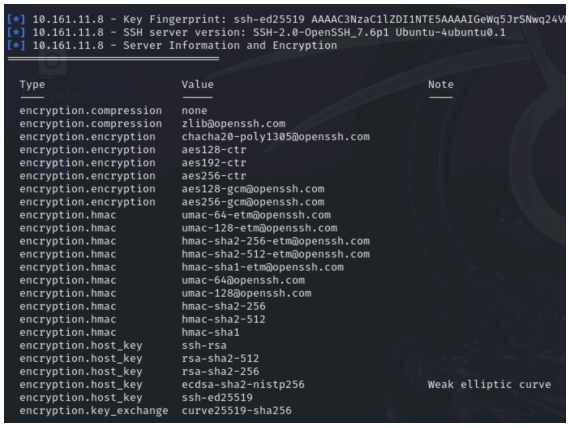
Metasploit

Since we know that ports 22 and 80 are open from our NMAP scan, we can use Metasploit to banner grab for those specific services.

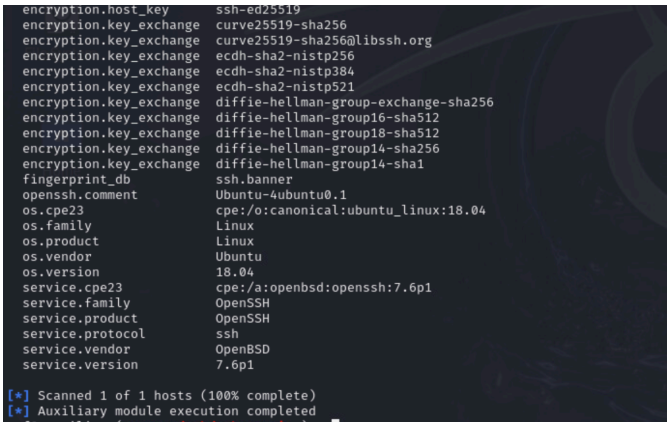
Once Metasploit is up and running, to check SSH version, run the commands:

```
msf6 > use auxiliary/scanner/ssh/ssh_version
msf6 auxiliary(scanner/ssh/ssh_version) > set RHOSTS 10.161.11.8
RHOSTS ⇒ 10.161.11.8
msf6 auxiliary(scanner/ssh/ssh_version) > run
```

This will display information retrieved from the SSH service:



```
[*] 10.161.11.8 - Key Fingerprint: ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGeWq5JrSNwq24V
[*] 10.161.11.8 - SSH server version: SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.1
[*] 10.161.11.8 - Server Information and Encryption

Type                  Value                                Note
encryption.compression   none
encryption.compression   zlib@openssh.com
encryption.encryption    chacha20-poly1305@openssh.com
encryption.encryption    aes128-ctr
encryption.encryption    aes192-ctr
encryption.encryption    aes256-ctr
encryption.encryption    aes128-gcm@openssh.com
encryption.encryption    aes256-gcm@openssh.com
encryption.hmac          umac-64-etm@openssh.com
encryption.hmac          umac-128-etm@openssh.com
encryption.hmac          hmac-sha2-256-etm@openssh.com
encryption.hmac          hmac-sha2-512-etm@openssh.com
encryption.hmac          hmac-sha1-etm@openssh.com
encryption.hmac          umac-64@openssh.com
encryption.hmac          umac-128@openssh.com
encryption.hmac          hmac-sha2-256
encryption.hmac          hmac-sha2-512
encryption.hmac          hmac-sha1
encryption.host_key      ssh-rsa
encryption.host_key      rsa-sha2-512
encryption.host_key      rsa-sha2-256
encryption.host_key      ecdsa-sha2-nistp256                  Weak elliptic curve
encryption.host_key      ssh-ed25519
encryption.key_exchange  curve25519-sha256
```



```
encryption.host_key      ssh-ed25519
encryption.key_exchange  curve25519-sha256
encryption.key_exchange  curve25519-sha256@libssh.org
encryption.key_exchange  ecdh-sha2-nistp256
encryption.key_exchange  ecdh-sha2-nistp384
encryption.key_exchange  ecdh-sha2-nistp521
encryption.key_exchange  diffie-hellman-group-exchange-sha256
encryption.key_exchange  diffie-hellman-group16-sha512
encryption.key_exchange  diffie-hellman-group18-sha512
encryption.key_exchange  diffie-hellman-group14-sha256
encryption.key_exchange  diffie-hellman-group14-sha1
fingerprint_db           ssh.banner
openssh.comment          Ubuntu-4ubuntu0.1
os.cpe23                 cpe:/o:canonical:ubuntu_linux:18.04
os.family                Linux
os.product               Linux
os.vendor                Ubuntu
os.version               18.04
service.cpe23            cpe:/a:openbsd:openssh:7.6p1
service.family           OpenSSH
service.product          OpenSSH
service.protocol         ssh
service.vendor           OpenBSD
service.version          7.6p1

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

As we can see, CTF9 is running OpenSSH version 7.6p1. Knowing this, attackers can look up exploits/vulnerabilities in that specific version. Metasploit also tells us the vulnerabilities it scans in the “Note” column. We can see here that the “encryption.host_key” with the value “ecdsa-sha2-nistp256” has the note “Weak elliptic curve”, which means the SSH server is using an insecure elliptic curve algorithm for authentication. This vulnerability allows attackers to crack and exploit private keys.

We will do the same thing with HTTP port 80:

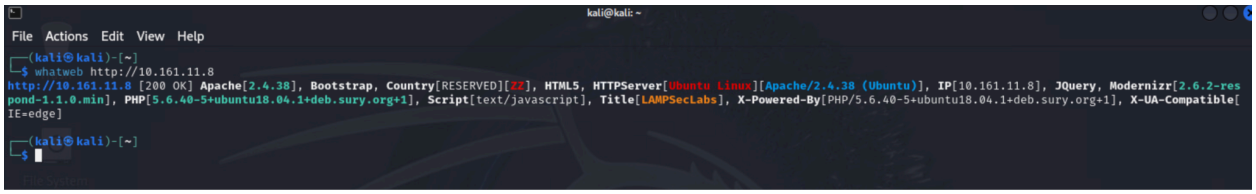
```
msf6 > use auxiliary/scanner/http/http_version
msf6 auxiliary(scanner/http/http_version) > set RHOSTS 10.161.11.8
RHOSTS ⇒ 10.161.11.8
msf6 auxiliary(scanner/http/http_version) > run

[+] 10.161.11.8:80 Apache/2.4.38 (Ubuntu) ( Powered by PHP/5.6.40-5+ubuntu18.04.1+deb.sury.org+1 )
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

The output shows that HTTP is running on Apache version 2.4.38 (Ubuntu). We know from our Nessus scan report that this version has high risk and critical vulnerabilities that can be exploited. The output also shows us that Apache is powered by PHP version 5.6.40 which PHP versions 5.x and 7.x are known to have Remote Code Execution (RCE) flaws.

WhatWeb

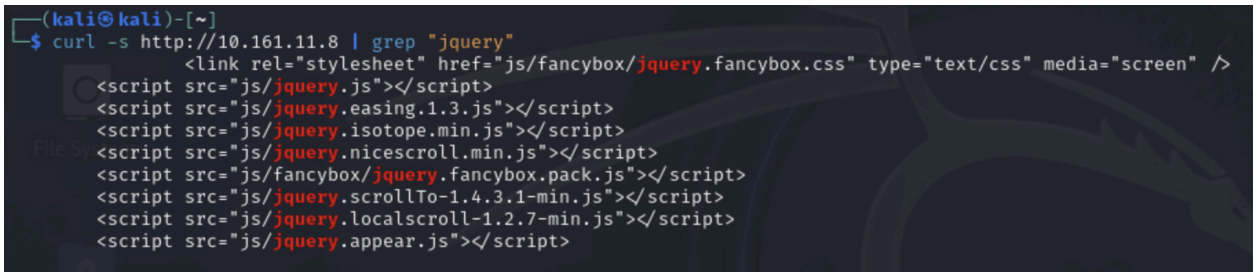
WhatWeb is a web application fingerprinting tool that gathers detailed information about a website.

This is the output after running a basic WhatWeb scan on CTF9:



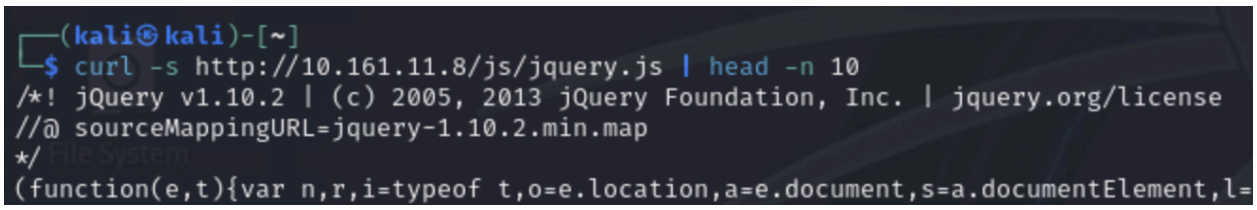
It shows us that HTTP is running on `Apache[2.4.38]`, and the same PHP version `PHP[5.6.40-5+ubuntu18.04.1+deb.sury.org+1]` that we got from our previous vulnerability scans. More importantly, we can see that the site uses `JQuery`, which can be another potential vulnerability as jQuery versions older than 3.5.0 could be vulnerable to XSS attacks.

Let's check for the jQuery version by using the curl command:



The output shows a list of multiple jQuery related JavaScript files but none that outright shows the version of it. Let's try inspecting some of the files to see if we can find the version.

Using the curl command again, we can see the contents of selected file:



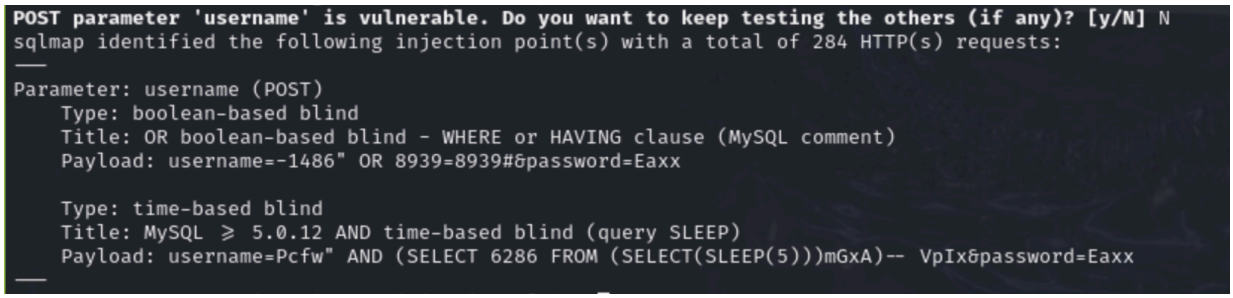
In the file "jquery.js", we can see jQuery version 1.10.2 is being used. As this version of jQuery is lower than 3.5.0, this jQuery might be vulnerable to XSS attacks.

SQLMap

SQLMap is an open-source tool that automates the process of detecting and exploiting SQL injection vulnerabilities in web applications.

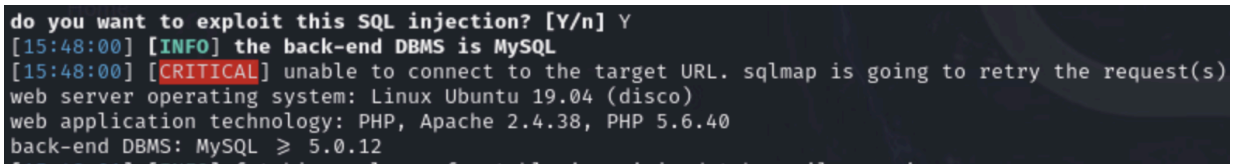
Use this SQLMap command to scan CTF9 web server:

This command will test any form inputs for SQL injection up to 2 levels deep and if possible, extract data from the “user” table in the “lampsec” dataset.

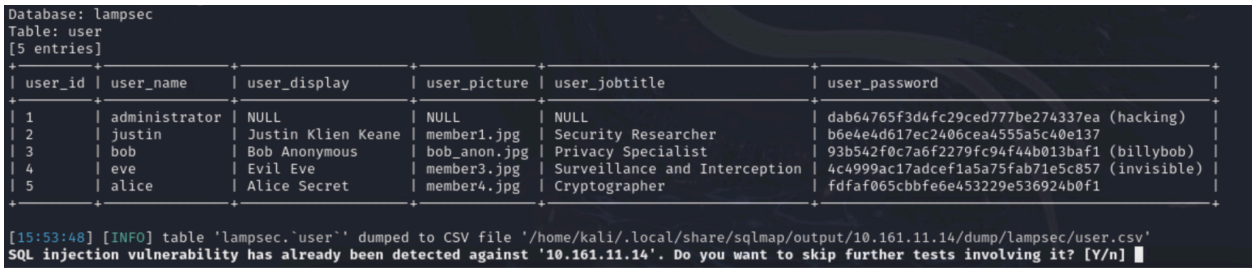


We can see here that the POST parameter “username” is vulnerable to injection attacks.

Let's try and exploit this vulnerability with SQLMap’s automation:



Here is the output of the “user” table from the “lampsec” database:



SQLMap can automatically detect and crack hashed passwords using its built-in dictionary. In the “user_password” column, we can see that SQLMap cracked a few of the hashed passwords. This is terrible for CTF9 as this type of data should not be accessible. The SQLMap scan and testing has shown us that CTF9 is vulnerable to SQL injection attacks.

Summary

The vulnerability scans revealed outdated and vulnerable software, including PHP 5.6.40, Apache 2.4.38, OpenSSH 7.6p1, and jQuery 1.10.2. These versions expose the system to serious risks like RCE, XSS, weak encryption, and unauthorized access due to misconfigurations and unpatched exploits.

Chapter 3: Table of Defensive Deliverables

Task	Description	Person	Due Date	Status
Firewall (REQ)	Configure pfsense and move CTF9 behind it	Hoo	4/30	Done 5/6/2025
Modsecurity (REQ)	Configure WAF for Apache	Ryan	4/30	Done 4/25/2025
PHP (REQ)	Update to version 8.1 and secure using MySQLi	Max	4/30	Done 5/5/2025
Light Firewall (OPT- MED)	Install local firewall configured with rules	Hoo	5/4	Done 5/6/2025
Permissions (OPT- EASY)	Fix permissions for users and applications	Ryan	5/4	Done 5/7/2025
Clean Banners (OPT- EASY)	Lessen information leaked through banners	Max	5/4	Done 5/8/2025

Chapter 4: Defensive Deliverable

Modsecurity

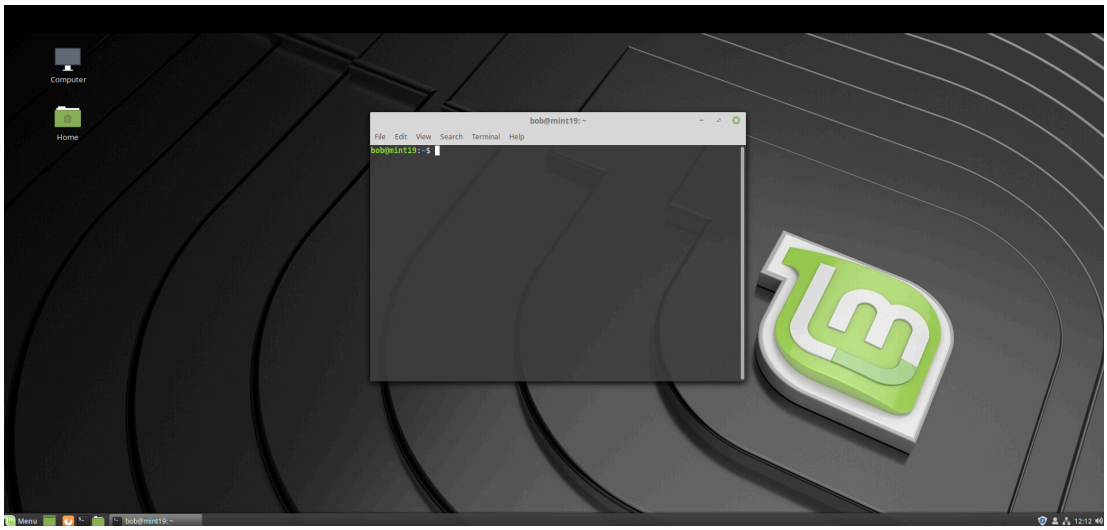
What is Modsecurity?

Modsecurity, also known as Modsec, is an open source, cross-platform web application firewall (WAF) module. It enables web applications defenders (system administrators) to gain insight of the HTTP(S) traffic and provides a power rules language and API to implement advanced protections.

Modsecurity is used by internet service providers, government organizations, and businesses worldwide. It doesn’t simply improve web security, Modsecurity paired with the OWASP CRS, the dominant web application firewall rule set, takes prevention of HTTP attacks to a new level.

Installing Modsecurity

First, log into CTF 9 as **Bob:billybob** and open a terminal window.



Now run “sudo apt update”

```
bob@mint19:~$ sudo apt update
[sudo] password for bob:
Ign:1 http://packages.linuxmint.com tara InRelease
```

Then install Modsecurity:

```
bob@mint19:~$ sudo apt install libapache2-mod-security2
```

Ensure Modsecurity is enabled on apache, then restart apache:

```
bob@mint19:~$ sudo a2enmod security2
Considering dependency unique_id for security2:
Module unique_id already enabled
Module security2 already enabled
bob@mint19:~$ sudo systemctl restart apache2
```

Navigate to the Modsecurity directory notice the name of the configuration file:

```
bob@mint19:~$ cd /etc/modsecurity/
bob@mint19:/etc/modsecurity$ ls
modsecurity.conf-recommended  unicode.mapping
bob@mint19:/etc/modsecurity$ sudo nano modsecurity.conf-recommended
```

Rename the configuration file:

```
bob@mint19:/etc/modsecurity$ sudo mv modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf
bob@mint19:/etc/modsecurity$ ls
crs  modsecurity.conf  unicode.mapping
```

Inside of the configuration file, “SecRuleEngine” is set to “DetectionOnly”, set this to “On”

```
# Switched to on to actually prevent attacks
SecRuleEngine On
```

Changing this setting will make the rules actually intercept the attacks and prevent them.

OWASP CRS Implementation

Install CRS with “sudo apt install modsecurity-crs”. There are two places where this adds files:

```
bob@mint19:/$ ls /etc/modsecurity/crs/
crs-setup.conf                          RESPONSE-999-EXCLUSION-RULES-AFTER-CRS.conf
REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf
```

And

```
bob@mint19:/$ ls /usr/share/modsecurity-crs/
id_renumbering  owasp-crs.load  rules  util
```

Upon installing this, restart apache and then we can move on to testing to see if Modsecurity and CRS are working properly.

Verification

Run this command to try test the XSS prevention functionality:

```
bob@mint19:/$ curl 'http://localhost/?q="><script>alert(1)</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>
```

Then check **/var/log/apache2/modsec_audit.log** to see the message:

```
bob@mint19:/$ sudo cat /var/log/apache2/modsec_audit.log
--e68bc905-A--
[25/Apr/2025:13:52:47 --0500] aAvZ-7H@Tm1B67H3Htc5twAAAAE 127.0.0.1 48580 127.0.0.1 80
--e68bc905-B--
GET /?q="><script>alert(1)</script> HTTP/1.1
Host: localhost
User-Agent: curl/7.58.0
Accept: */*

--e68bc905-F--
HTTP/1.1 403 Forbidden
Content-Length: 284
Content-Type: text/html; charset=iso-8859-1

--e68bc905-E--
<!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>

--e68bc905-H--
Message: Warning. detected XSS using libinjection. [file "/usr/share/modsecurity-crs/rules/REQUEST-941-APPLICATION-ATTACK-XSS.conf"]
```

This is exactly what we were trying to do! This means that Modsecurity and CRS are working!

PHP Update

Currently, the PHP version being used by Apache is still running PHP 5.6.40, which was shown in the above section covering the summary of services. From running the command

```
php --version
```

we get the following output:

```
bob@mint19:~$ php --version
PHP 8.1.3 (cli) (built: Feb 21 2022 14:48:26) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.1.3, Copyright (c) Zend Technologies
    with Zend OPcache v8.1.3, Copyright (c), by Zend Technologies
```

We can see from this output that the PHP version 8.1.3 is installed. We will be switching Apache from running 5.6.40 to the new version 8.1.3 and migrate mySQL functions to mySQLi functions.

First, we want to enable PHP 8 in Apache. In order to enable version 8.1, we need to disable version 5.6 by running the command

```
sudo a2dismod php5.6
```

Then we will want to enable version 8.1 using the command

```
sudo a2enmod php8.1
```

Finally we will need to restart the Apache web server using the command

```
sudo systemctl restart apache2.service
```

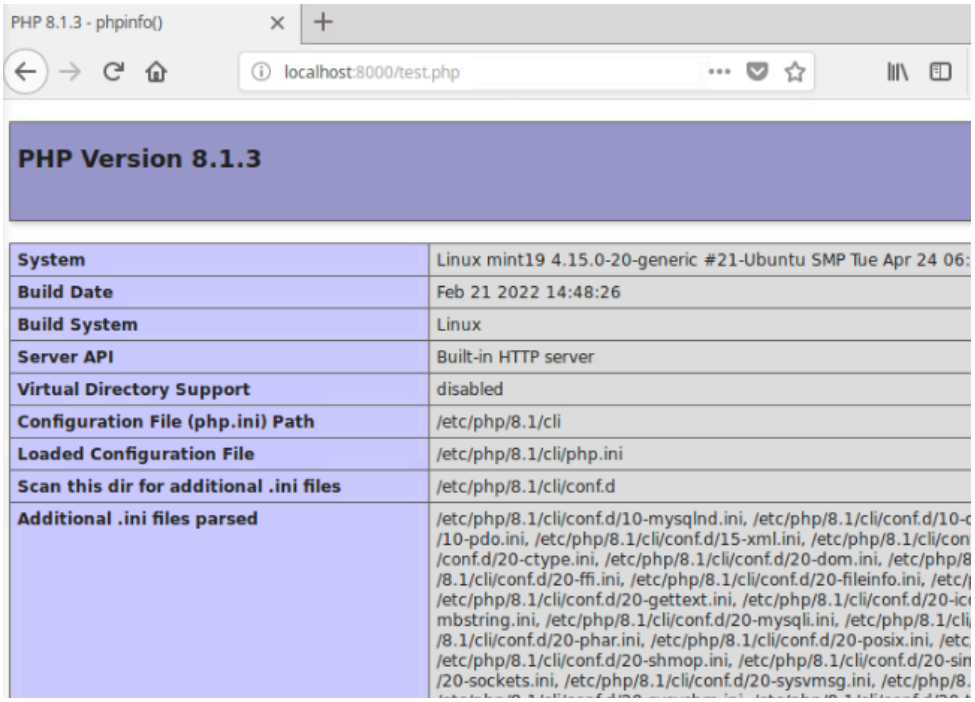
We now want to verify that Apache is using PHP version 8. Change directories to the /var/www/html directory using the command

```
cd /var/www/html
```

We notice that the test.php file in this directory is used to check the php version that apache is running is now 8.1. In order to utilize this file, we will want to set up a local server to access the php files. While still inside the /var/www/html directory run

```
php -S localhost:8000
```

Then open Mozilla Firefox and visit localhost:8000/test.php



Here we can clearly see that Apache is now running PHP version 8.1.3.

Next, changing the files in the /var/www/html directory to utilize mySQLi functions must be done. Inside the /var/www/html directory, we can list the files using ls:

```
bob@mint19:/var/www/html$ ls
admin    css      feedback.php  img         js    skin        test.php
contact  db.php   fonts         index.php   logs  static.php
```

Here we need to edit the files feedback.php, db.php, index.php and static.php to use mySQLi functions instead of mySQL functions that they are currently using.

First, open the feedback.php file using

```
nano feedback.php
```

and editing the file in the corresponding section to the following configurations found in the highlighted text:

```
<?php
include_once('db.php');
$name = $email = '';
$subject = 'There was an error processing your request.';
$message = 'Invalid contact id';
if (isset($_GET['id'])) {
        $sql = 'select * from contact where contact_id = ' . intval($_GET['id'])
;
        $result = mysqli_query($sql);
        if ($result === FALSE ) {
                print mysqli_error();
        }
        while ($row = mysqli_fetch_assoc($result)) {
                $name = $row['contact_name'];
                $email = $row['contact_email'];
                $subject = $row['contact_subject'];
                $message = $row['contact_message'];
        }
}
?><!DOCTYPE html>
<!--[if lt IE 7]>      <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
<!--[if IE 7]>         <html class="no-js lt-ie9 lt-ie8"> <![endif]-->
<!--[if IE 8]>         <html class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!--> <html class="no-js"> <!--<![endif]-->
```

Next, open the db.php file using

nano db.php

and editing the file in the corresponding section to the following configurations found in the highlighted text:

```
<?php
$fp = fopen('/var/www/html/logs/access_log.txt', 'a+');
$today = getdate();
$dstamp = $today['year'] . '-' . $today['mon'] . '-' . $today['mday'] . ' ' . $t
oday['hours'] . ':' . $today['minutes'] . ':' . $today['seconds'];
$ref = isset($_SERVER['HTTP_REFERER']) ? $_SERVER['HTTP_REFERER'] : '';
fwrite($fp, $dstamp . ' ' . rawurldecode($_SERVER['REQUEST_URI']) . ' ' . $_SERV
ER['HTTP_USER_AGENT'] . ' ' . $ref . "\n");
fclose($fp);


$conn = mysqli_connect('localhost', 'root','U<Iowa>76!', 'lampsec');
if (!$conn) {
        die("Connection failed: " . mysqli_connect_error());
}
?>
```

This will set the password for the mySQL root access to **U<Iowa>76!**. In order for this to take effect we will want to reset the mySQL root password. First run

sudo mysql

Then once in the mysql command shell run the following

ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'U<Iowa>76!';

FLUSH PRIVILEGES;

EXIT;

This will reset the password, switch to password-based login and flushes the user privilege cache. We can verify that this worked by first running

mysql -u root

```
bob@mint19:/var/www/html$ mysql -u root
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: N
O)
```

Notice that it gave an error since we did not enter a password! This is great. Now we should make sure the password we specified works. Run

mysql -u root -p

and enter **U<Iowa>76!** for the password.

```
bob@mint19:/var/www/html$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 16
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>
```

We were able to access the mysql command shell, this means our password does work!

Next, open the index.php file using

nano index.php

and editing the file in the corresponding sections to the following configurations found in the highlighted text:

```
                        </div>

                                <div class="row align-center mar-bot40">
                                <?php
                                        $result = mysqli_query('select *
 from user where user_display IS NOT NULL');
                                        while ($row = mysqli_fetch_assoc
($result)) {
                                                ?>
```

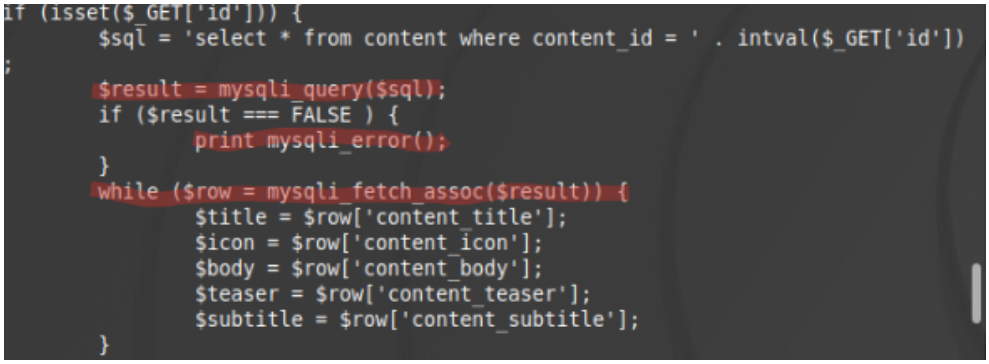
```
                       <?php
                         $result = mysqli_query('select * from category')
;
                         while ($row = mysqli_fetch_assoc($result)) {
                                 print "\t\t\t\t";
                                 print '<li><a href="#"  class="btn-theme
 btn-small" data-filter="." . $row['category_name'] . '" >' . $row['category_nam
e'] . '</a></li>';
                                 print "\n";
                          }
                       ?>
```

```
                                                        <?php
                         $articles = mysqli_query('select a.*,c.category_
name from content a, category c where a.category_id = c.category_id');
                         while ($row = mysqli_fetch_assoc($articles)) {
                                 print "\t\t\t\t";
                                 print '<article class="col-md-4 isotopeI
tem ' . $row['category_name'] . '">';
                                                                print '<
div class="portfolio-item">';
                                                                print '<
img src="img/portfolio/' . $row['content_image'] . '" alt="" />';
                                                                print '<
div class="portfolio-desc align-center">';
                                                                print '<
div class="folio-info">';
                                                                print '<
h5><a href="static.php?id=' . $row['content_id'] . '">' . $row['content_title']
. '</a></h5>';
                                                                print '<
a href="img/portfolio/img1.jpg" class="fancybox"><i class="fa fa-plus fa-2x"></i
></a>';
```

Finally, open the static.php file using

nano static.php

and editing the file in the corresponding section to the following configurations found in the highlighted text:



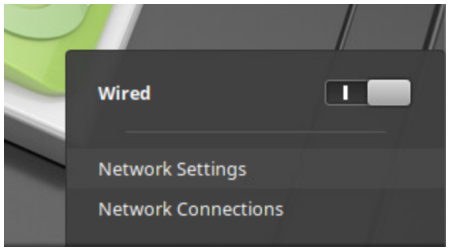
```
if (isset($_GET['id'])) {
        $sql = 'select * from content where content_id = ' . intval($_GET['id'])
;
        $result = mysqli_query($sql);
        if ($result === FALSE ) {
                print mysqli_error();
        }
        while ($row = mysqli_fetch_assoc($result)) {
                $title = $row['content_title'];
                $icon = $row['content_icon'];
                $body = $row['content_body'];
                $teaser = $row['content_teaser'];
                $subtitle = $row['content_subtitle'];
        }
```

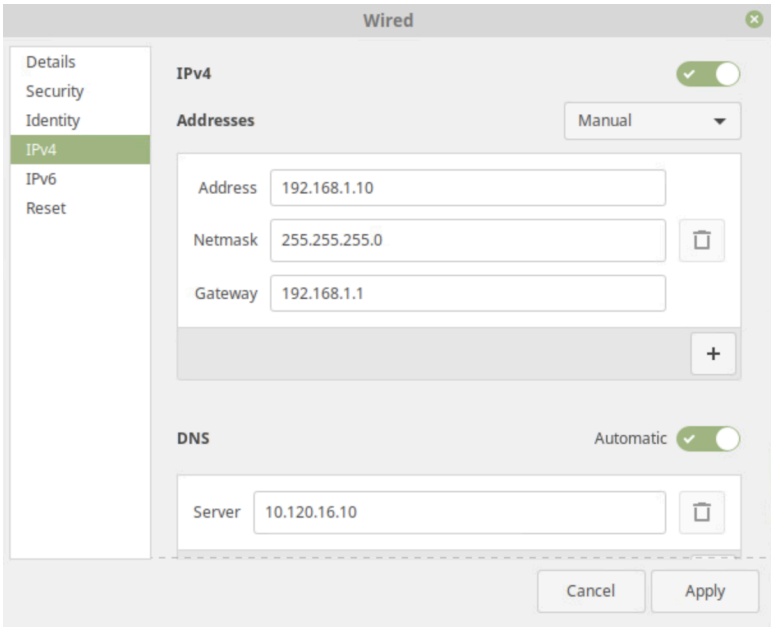
Apache is now properly using version 8.1.3 of PHP.

Firewall

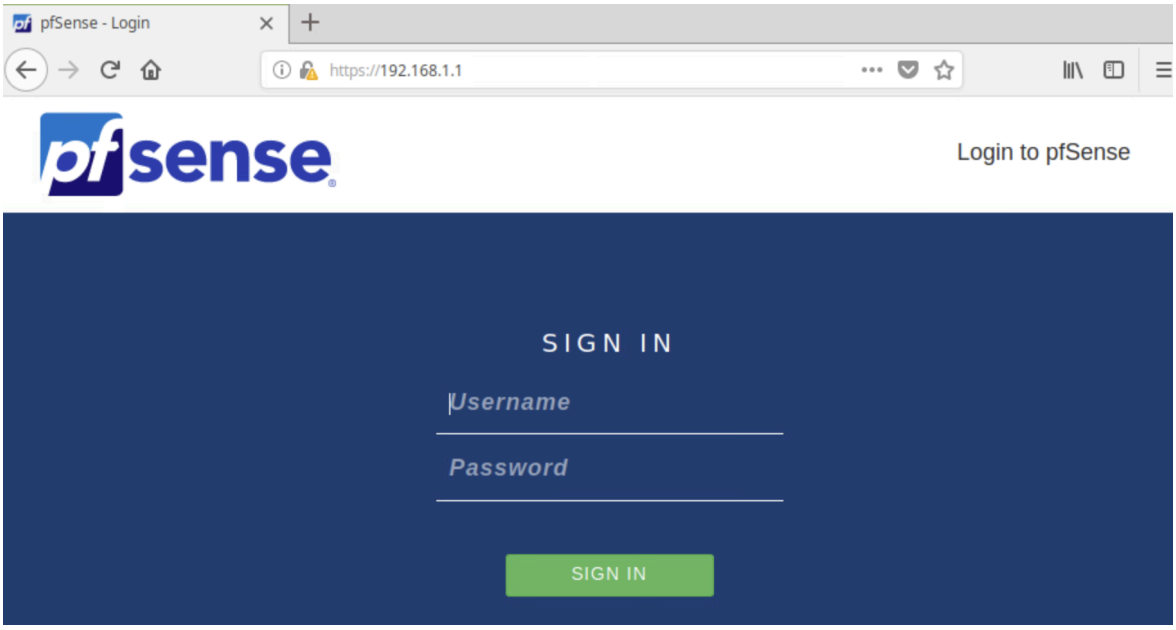
pfSense

Currently, the CTF9 server is not configured to be behind pfSense. To do this we will first log into CTF9 and make our way to “Network Settings -> Settings -> IPv4”:

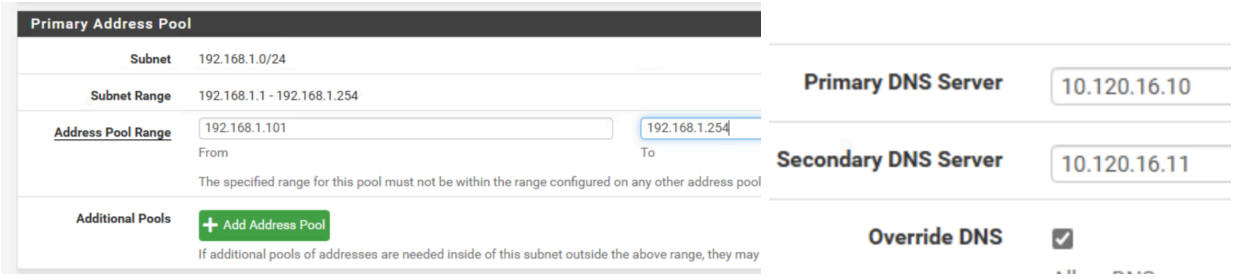




Give the IPv4 the new address of “192.168.1.10” with the LAN gateway address “192.168.1.1”. Add the UNI DNS server “10.120.16.10” to the DNS server address.

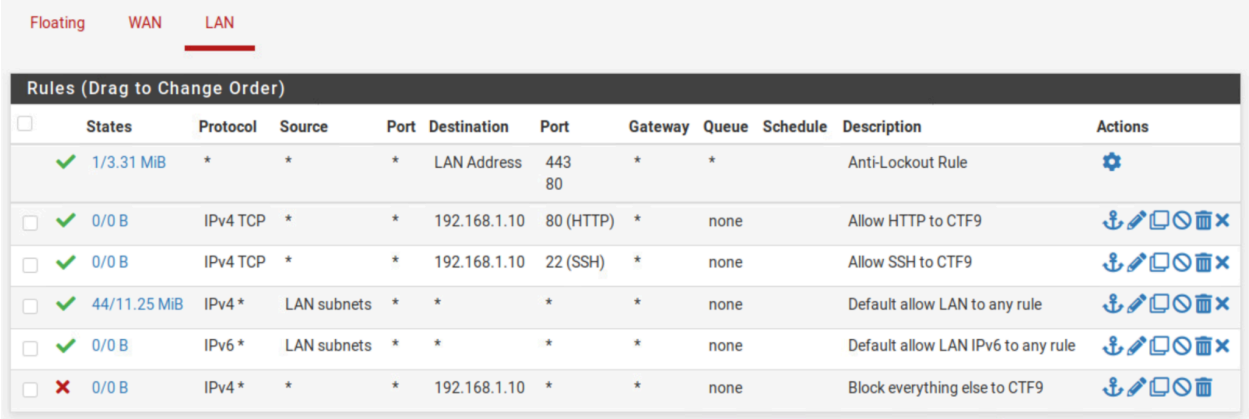


Next we will go to the browser and type in the gateway address “192.168.1.1”. This will take us to pfSense. Login with username “admin” and password “pfsense”.

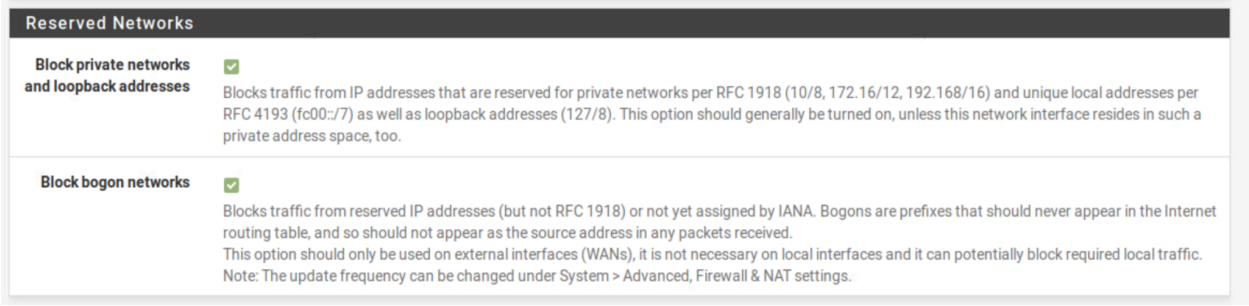


While going through the pfSense setup, make sure the DHCP server has the correct primary address pool (LAN) and the UNI DNS server address. Restart your CTF9 network connection and now CTF9 should be behind pfSense.

Next, we will add 3 new rules to LAN:



Two of the rules will be added at the top of the list to allow HTTP (port 80) and SSH (port 22) to CTF9. The third rule will block all other traffic to CTF9. Make sure to place this rule below the allowed rules! This will ensure that only specific allowed services are reachable.



In Interfaces -> WAN and make sure that “Block private networks and loopback addresses” and “Block bogon networks” is checked. This prevents spoofed traffic or local networks from reaching WAN.

Uncomplicated Firewall

UFW (Uncomplicated Firewall) is a simple tool in Linux that helps you control which network connections are allowed or blocked.

First, we will install UFW in CTF9 with the command:

```
bob@mint19:~$ sudo apt install ufw
```

Once UFW is installed, use the commands:

```
bob@mint19:~$ sudo ufw default deny incoming
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
```

```
bob@mint19:~$ sudo ufw default allow outgoing
Default outgoing policy changed to 'allow'
(be sure to update your rules accordingly)
```

This will make it so that the firewall will deny all incoming traffic by default while allowing all traffic to go out. We do this so that later we can ensure that only explicitly allowed traffic will be allowed to get in.

Next, use the “ufw allow” command to allow specific services:

```
bob@mint19:~$ sudo ufw allow ssh
Rules updated
Rules updated (v6)
```

```
bob@mint19:~$ sudo ufw allow 80/tcp
Rule added
Rule added (v6)
```

We will want to allow services from SSH (port 22) and HTTP (port 80).

Lastly, we can activate the firewall with the command:

```
bob@mint19:~$ sudo ufw enable
Firewall is active and enabled on system startup
```

Once your firewall is up and running, you can check if their active by typing the command:

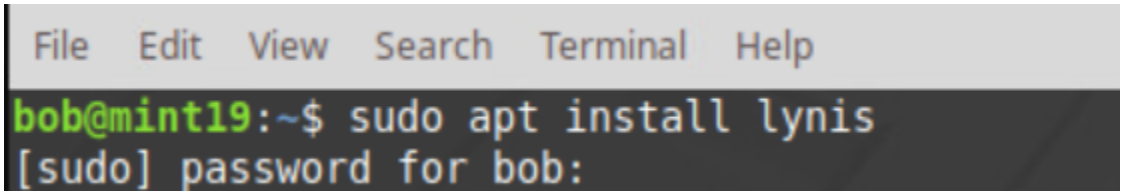
```
bob@mint19:~$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22/tcp                     ALLOW IN    Anywhere
[ 2] 80/tcp                     ALLOW IN    Anywhere
[ 3] 22/tcp (v6)                ALLOW IN    Anywhere (v6)
[ 4] 80/tcp (v6)                ALLOW IN    Anywhere (v6)
```

As we can see here, by default the firewall is denying all incoming traffic and only allowing from ports 22 and 80.

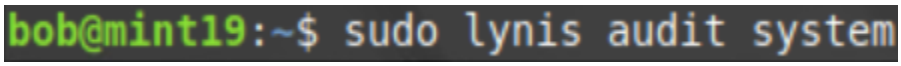
Permissions

Ensuring proper file and directory permissions on Linux based systems, especially servers, is especially important for a machine as vulnerable as the CTF 9 machine. Having improper permissions can expose sensitive files, allow unauthorized access, or even enable privilege escalation for potential attackers. Granting the minimal permissions to non root users is going to be the main focus of this section so we can prevent accidental or intentional system compromise. Securing file and directory permission is not only a good practice, but a necessity when creating a secure Linux server environment.

The first thing we will be doing is log into the CTF 9 machine as **Bob**:**billybob** and open a terminal and install the lynis auditing tool.



Run lynis



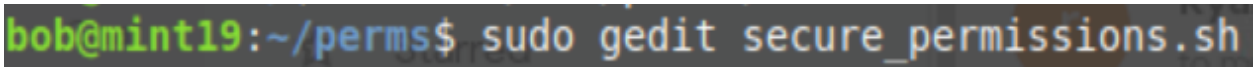
Lynis will:

- Scan system configuration
- Check permissions
- Assess installed software
- Recommend actions to improve security

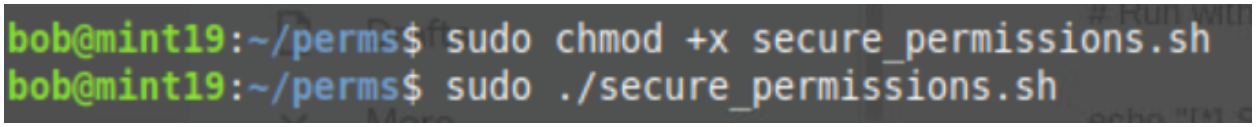
We are more interested in the permissions for this section though.

The lynis report tells us vaguely of some things that should be done for the permissions. Things like scanning the general binary file locations (/bin, /sbin, /usr/bin, /usr/sbin, /usr/local/bin, /usr/local/sbin) for setuid bits. Permissions should be standardized, so I used a bash script to automatically secure all important files and directories and remove unnecessary setuid bits. The bash script can be found here: https://pastebin.com/Z1uznbz8 and each section explains what it does. TLDR: it sets the owner of sensitive files and directories to root and adjusts the permissions accordingly.

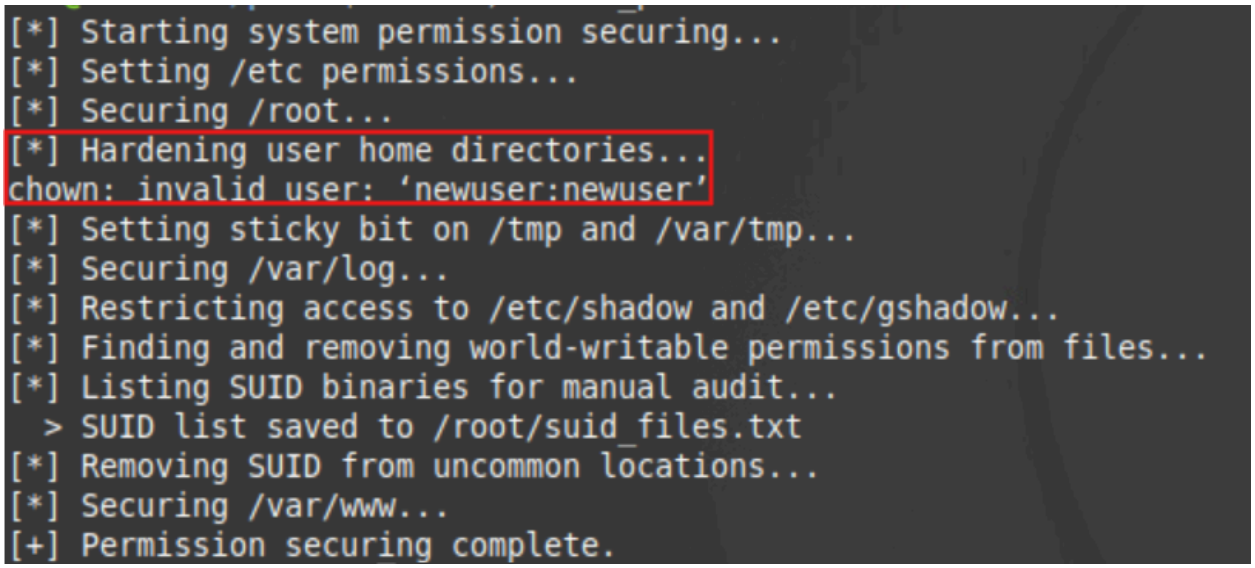
Create a .sh file



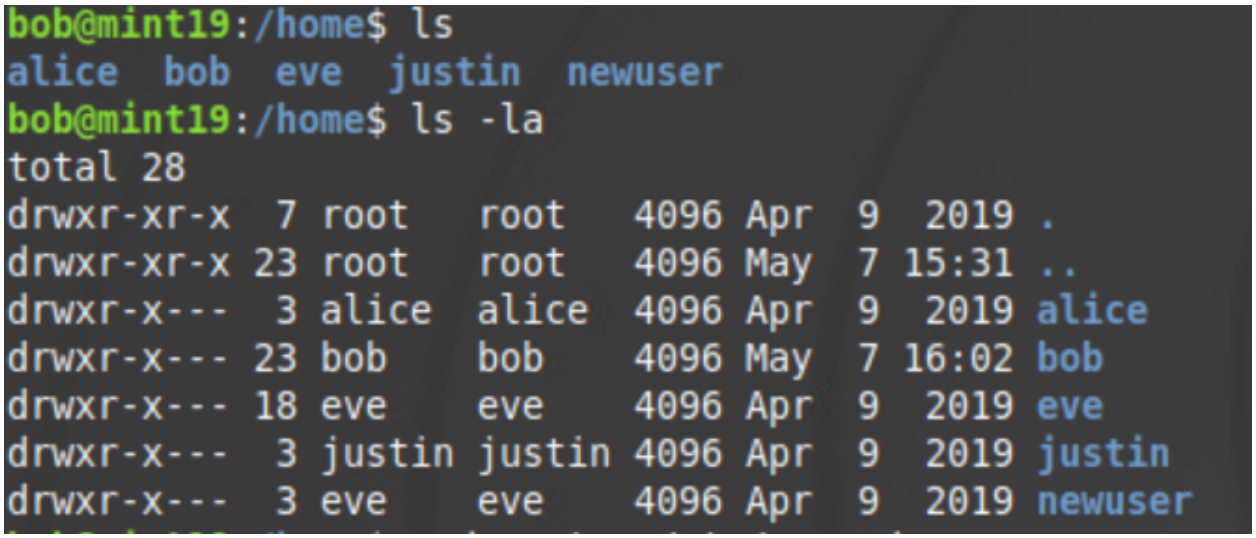
Allow it to be executable, then run it as sudo



Everything except the user home directory section worked correctly



This will need to be done separately, so we will double check what the permissions on those look like.



The issue with how the home directories are set up right now is that only the owner should be able to access their home directory. So the permissions should be “chmod 700”

To simplify this, I will be using a new bash script to automatically change this for me. As before, the script can be found here: https://pastebin.com/pR7j4dPd.

Now I will run the script

```
bob@mint19:~/perms$ sudo ./home-perms.sh
Fixing permissions for: /home/alice
Fixing permissions for: /home/bob
Fixing permissions for: /home/eve
Fixing permissions for: /home/justin
Fixing permissions for: /home/newuser
chown: invalid user: ‘newuser:newuser’
```

It would seem that the user “newuser” is not a registered user in the system, so it can just be removed.

```
bob@mint19:/home$ sudo rm -rf newuser/
bob@mint19:/home$ ls
alice  bob  eve  justin
```

Now verify the permissions on each of the user home directories.

```
bob@mint19:/home$ sudo ls -la
total 24
drwxr-xr-x  6 root   root   4096 May  7 16:25 .
drwxr-xr-x 23 root   root   4096 May  7 15:31 ..
drwx------  3 alice  alice  4096 Apr  9  2019 alice
drwx------ 23 bob    bob    4096 May  7 16:02 bob
drwx------ 18 eve    eve    4096 Apr  9  2019 eve
drwx------  3 justin justin 4096 Apr  9  2019 justin
```

The bash script for the home directories sets the owner to the user corresponding to the name of the file so no other permissions will need to be granted other than the first rwx.

Now let's run lynis again and see if some things have improved. According to the report and log, after running lynis this time, the permission warnings of the binaries and user directories were not present this time.

perms.zip are the report and log of the first time I ran the scan with no changes made and a zipped file size of 107.5kb, and perms2.zip are the same files but after the changes were made with a zipped file size of 107.4kb. This means that there is some text missing from the scan after the permission changes were made which would make sense since we went through and updated the permissions, removing some of the warning messages.

Clean Banners

What is Banner Grabbing?

Banner grabbing refers to the extraction of software banner information from remote or local servers. Banner grabbing assists defenders in reducing and managing the attack surface exposure. This will prevent attackers from pinpointing insecurities and vulnerable applications. Disabling banners in applications will prevent attackers from grabbing vulnerable information if our other lines of defense go down.

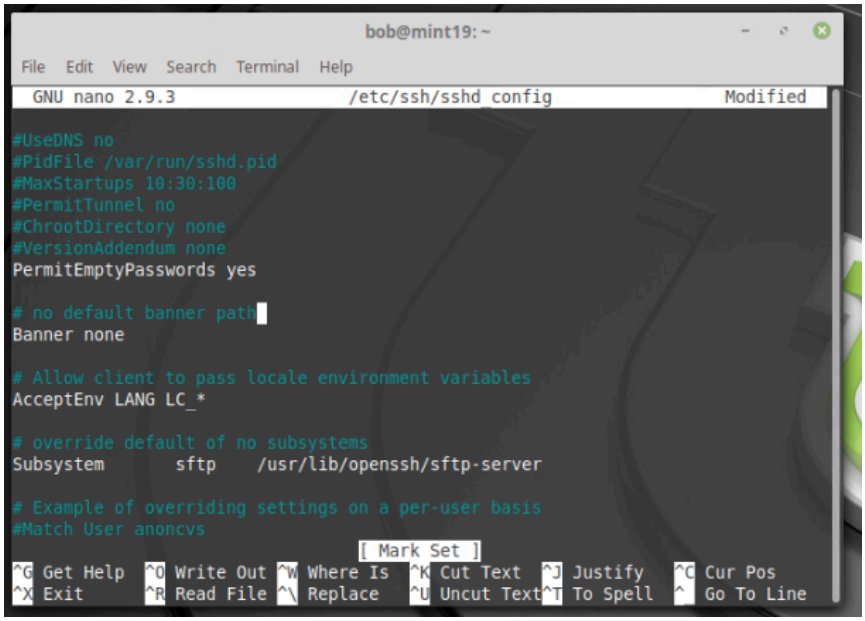
SSH Banner Prevention

In order to disable the ssh banner, we will edit the /etc/ssh/sshd_config file by running the command

```
sudo nano /etc/ssh/sshd_config
```

Then find the line **Banner none** as in the image below and make sure it is no longer commented out by deleting the preceding # character. This will prevent banners from being displayed in the future. Save and exit this file and then run the following command to restart ssh with the newly configured settings:

```
sudo systemctl restart sshd
```



Apache Banner Prevention

In order to protect the apache banner, we will edit the /etc/apache2/conf-available/security.conf file by running the command

```
sudo nano /etc/apache2/conf-available/security.conf
```

Then find the line **ServerTokens OS** as in the image below and make sure it is commented out. Then below add the line **ServerTokens Prod**. This will ensure that only “Apache” appears in headers. Then scroll down and find the line **ServerSignature On** and comment this line out by adding a preceding # character, then find the line **ServerSignature Off** and removing the preceding # character. Save and exit this file and then run the following command to restart apache with the newly configured settings:

```
sudo systemctl reload apache2
```

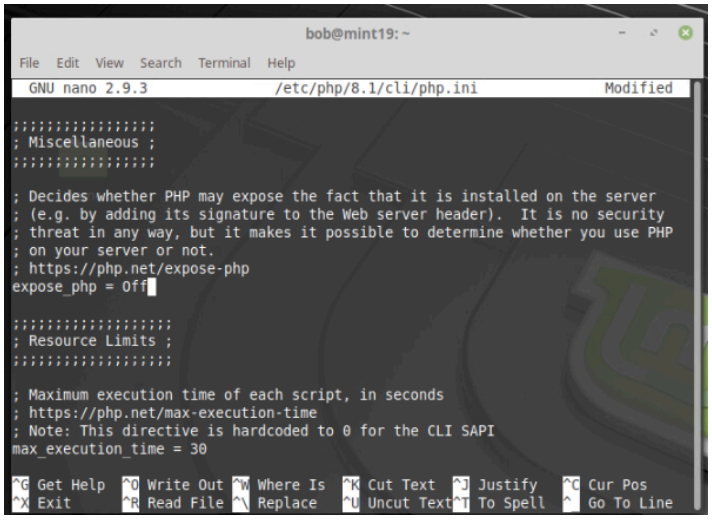
PHP Banner Prevention

In order to protect the php banner, we will edit the /etc/php/8.1/cli/php.ini file by running the command

```
sudo nano /etc/php/8.1/cli/php.ini
```

Then find the line **expose_php = On** and set it to off instead as in the image. Save and exit the file and then run the command

```
sudo systemctl restart apache2
```



MySQL Banner Prevention

By default, MySQL will display version information when a client connects. In order to prevent banner grabbing, we can add a rule to our ufw to block remote access to MySQL. Running the command

```
sudo ufw deny from any to any port 3306 proto tcp
```

will block all remote access to MySQL unless explicitly needed.

Chapter 5: Final Vulnerability Report

Now that all of the security has been updated on our system, we will rerun our vulnerability scans and report the new results.

Nessus Final Report

Once a scan has been completed on our system (192.168.1.10), we receive the following vulnerability report:

This is great news, Nessus was no longer able to find any vulnerabilities on our system due to our security protections.

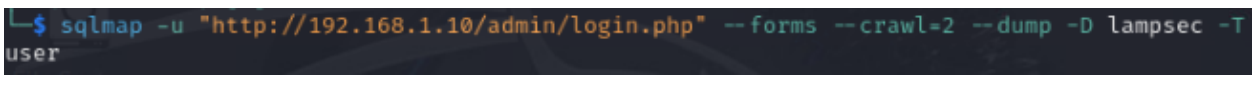
Nikto Final Report

After running a Nikto report on the system, we receive the following outcome:

Yet more great news, Nikto was unable to gather any information from the system host unlike before where it found several versions of the services running on CTF9.

SQLMap Final Report

After running the command:



The following output is produced:

SQLMap is unable to reach the target url meaning our computer is no longer vulnerable to an SQLMap attack.

Other Vulnerabilities Not Found

While our scans were unable to identify these vulnerabilities, things like file permissions and banner grabbing methods could still allow unauthorized access or vulnerability findings. However, we took precautions to prevent these types of attacks by changing configurations of certain files to protect the system as much as possible in case our firewall were compromised or other issues were to arise that would make the system vulnerable in other ways.

Chapter 6: Future Work

The final outcome of this project is a highly protected system that is no longer vulnerable to vulnerability scans or remote attacks. If an attacker were able to gain access through the users of the system, the file permissions have been changed and root access to MySQL has been protected with a strong password. This system must continue to be monitored and updated accordingly to retain this protected state.

Bibliography/Read Next

Modsecurity - https://modsecurity.org/
Last accessed - 4/22/25

Upgrade PHP -
https://docs.vultr.com/upgrade-from-php-7-to-php-8-on-ubuntu-20-04-with-apache
Last accessed - 5/5/25

Banner Grabbing -
https://www.recordedfuture.com/threat-intelligence-101/tools-and-techniques/banner-grabbing
https://man.openbsd.org/sshd_config
https://www.php.net/manual/en/ini.core.php#ini.expose-php
Last accessed - 5/8/25

UFW - https://documentation.ubuntu.com/server/how-to/security/firewalls/index.html
Last accessed: 5/5/24

pfSense Firewall - https://docs.netgate.com/pfsense/en/latest/firewall/configure.html
Last accessed: 5/4/24

SQLMap - https://hackertarget.com/sqlmap-tutorial/
https://www.vaadata.com/blog/sqlmap-the-tool-for-detecting-and-exploiting-sql-injections/
Last accessed: 5/5/24

Nikto - https://www.freecodecamp.org/news/an-introduction-to-web-server-scanning-with-nikto/
Last accessed: 4/25/24