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

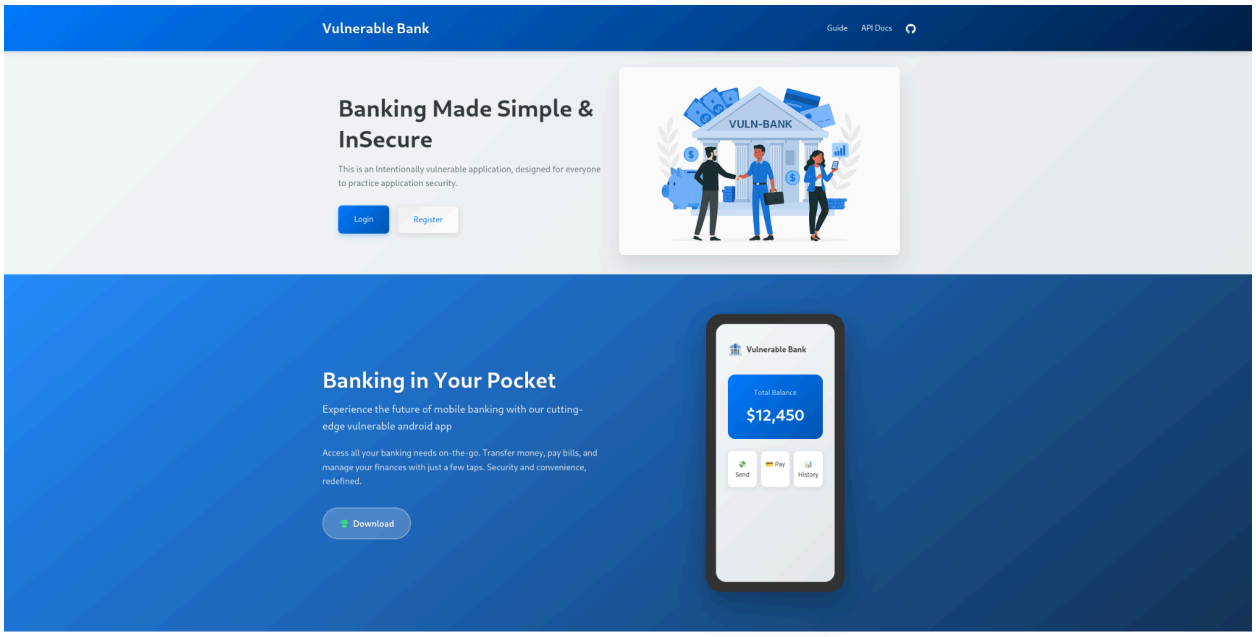
The testing approach combined black-box, gray-box, and white-box techniques:
Tools Used:

- 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**

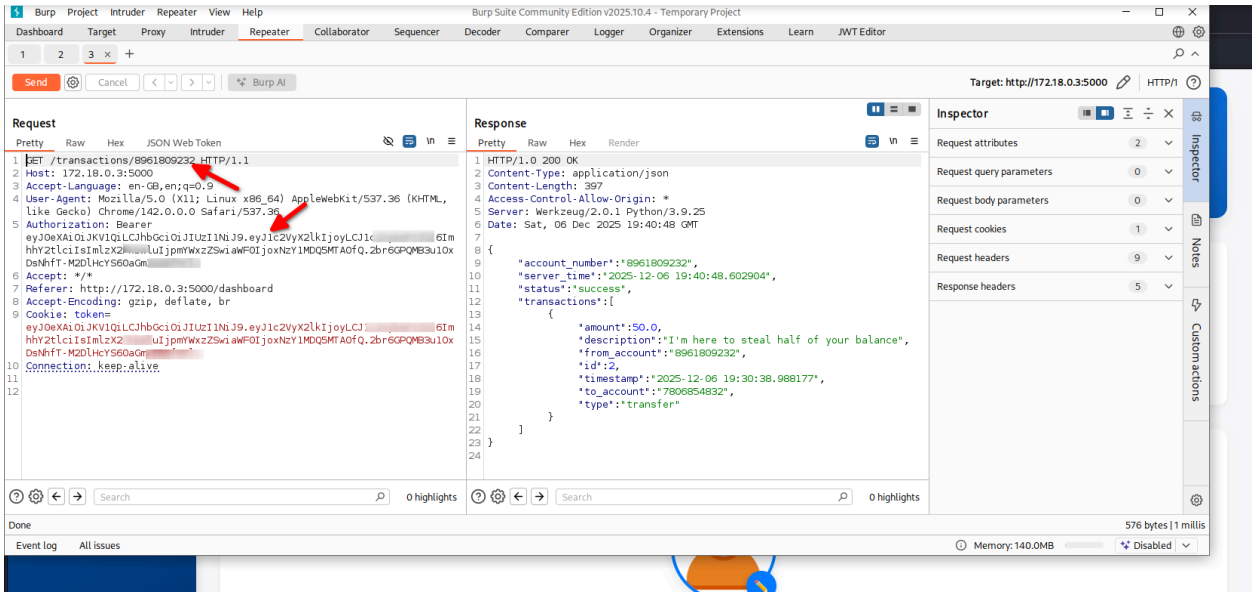
User2 ⇒ **7806854832**

Finding 1: Possible Broken Object Level Authorization (BOLA)

Affected endpoint:

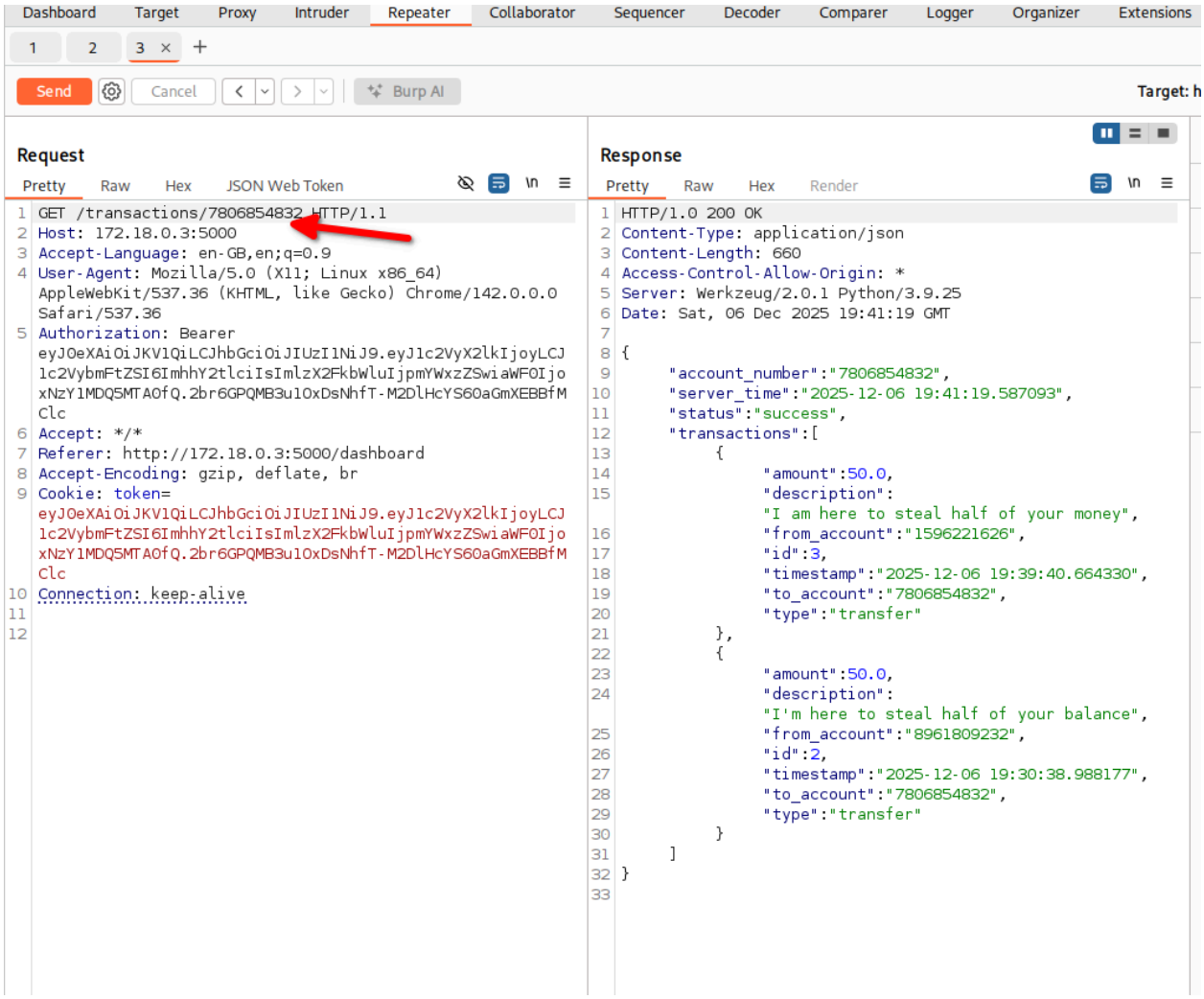
- GET /transactions/{account_number}
- POST /api/virtual-cards/{card_id}/toggle-freeze

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



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.

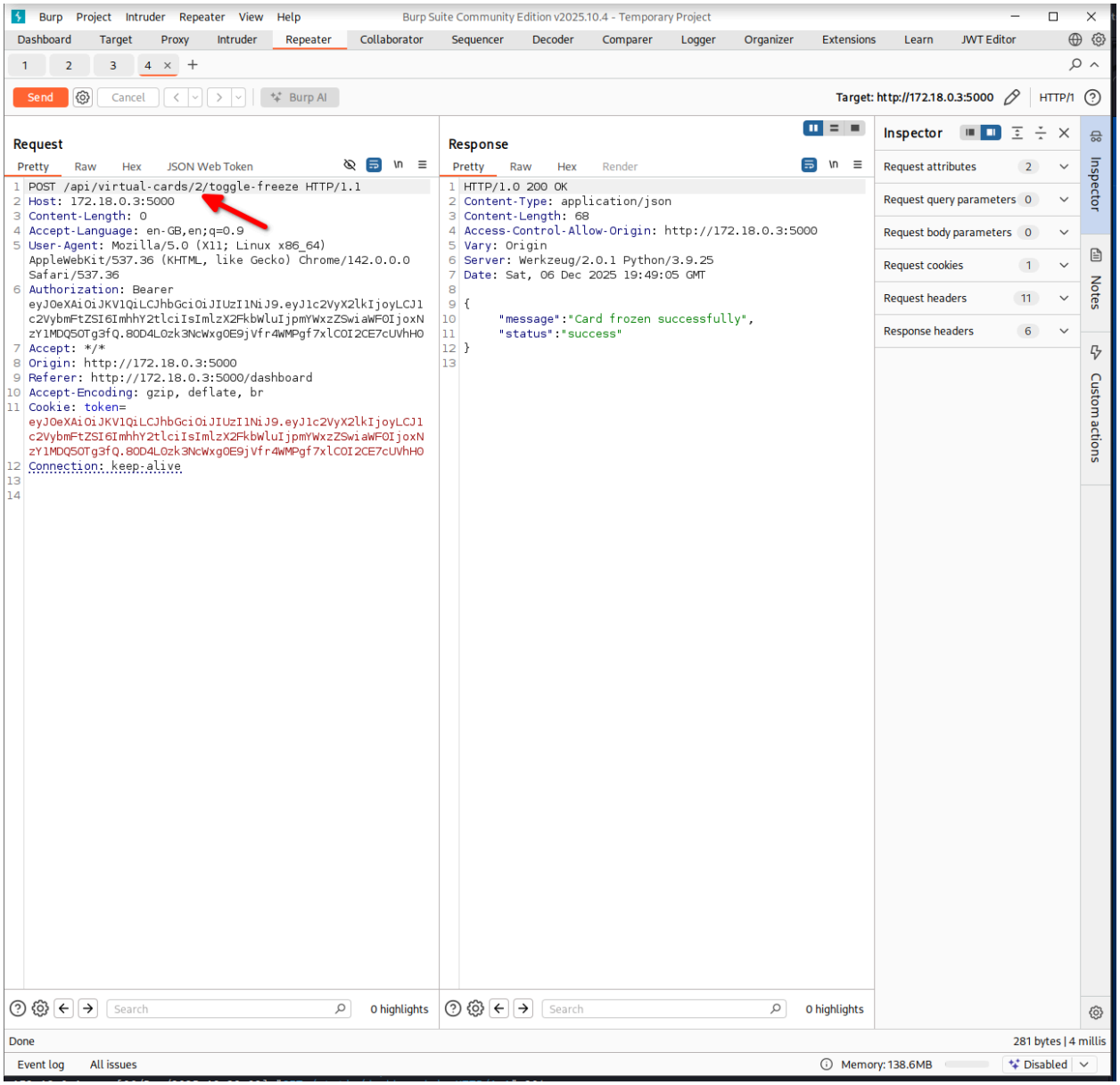
Exploitation



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**

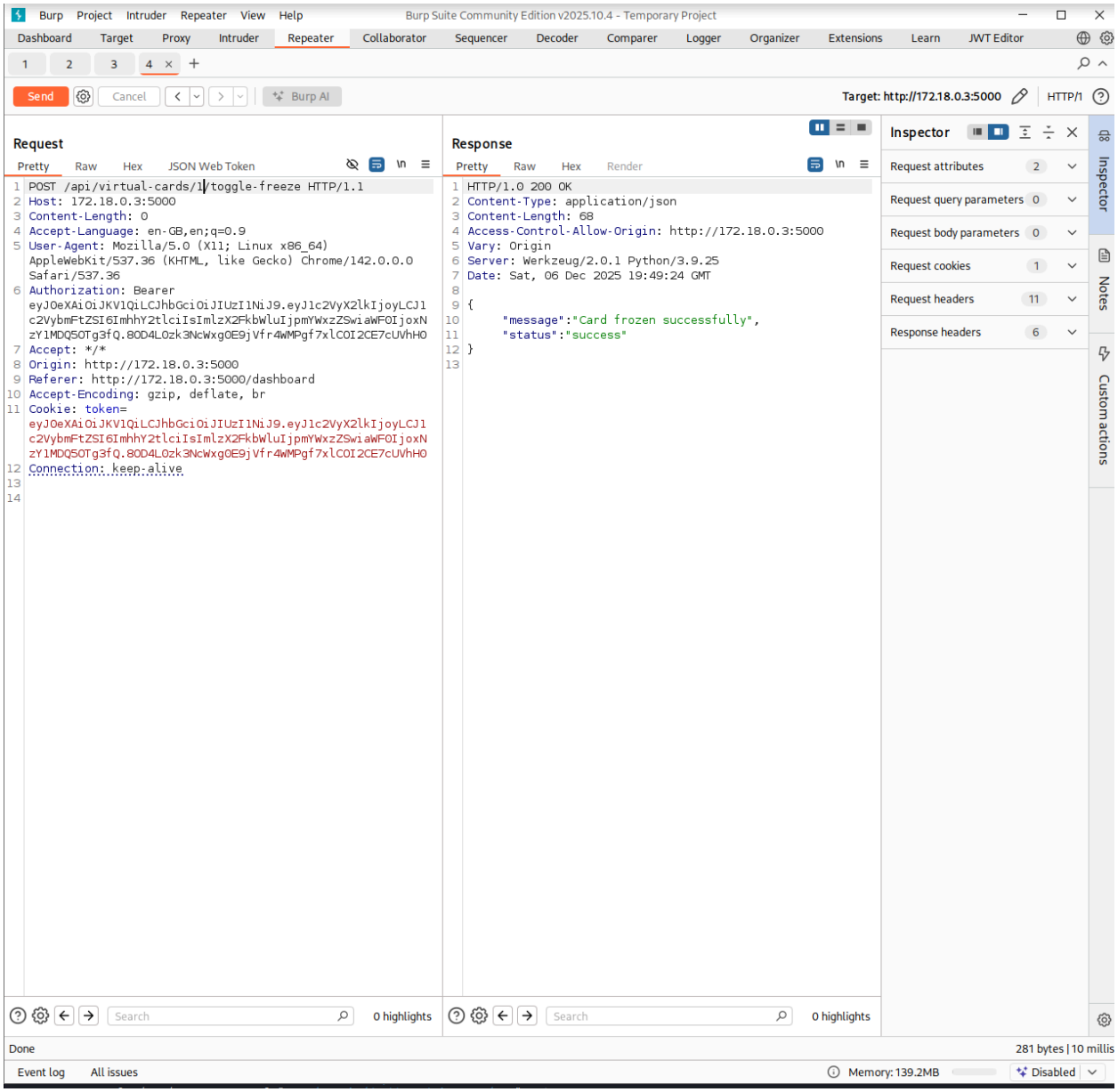


The hacker's account virtual ID card is ID 2 as seen above

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



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

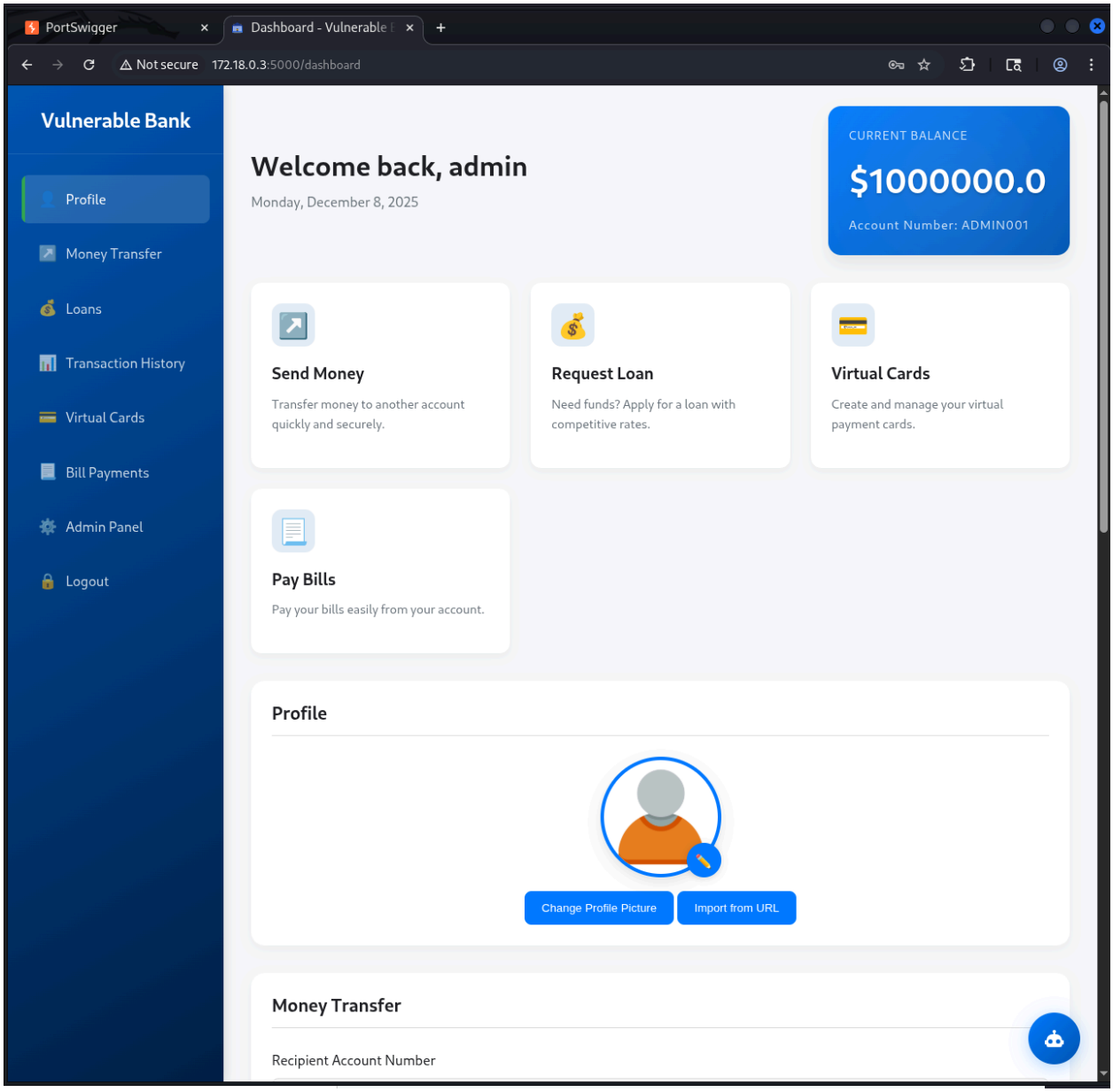
Description: The login endpoint lacks essential security controls such as rate limiting, login timeouts, and account lockout mechanisms. Because of this, an attacker can send an unlimited number of authentication attempts without restriction.

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.

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.

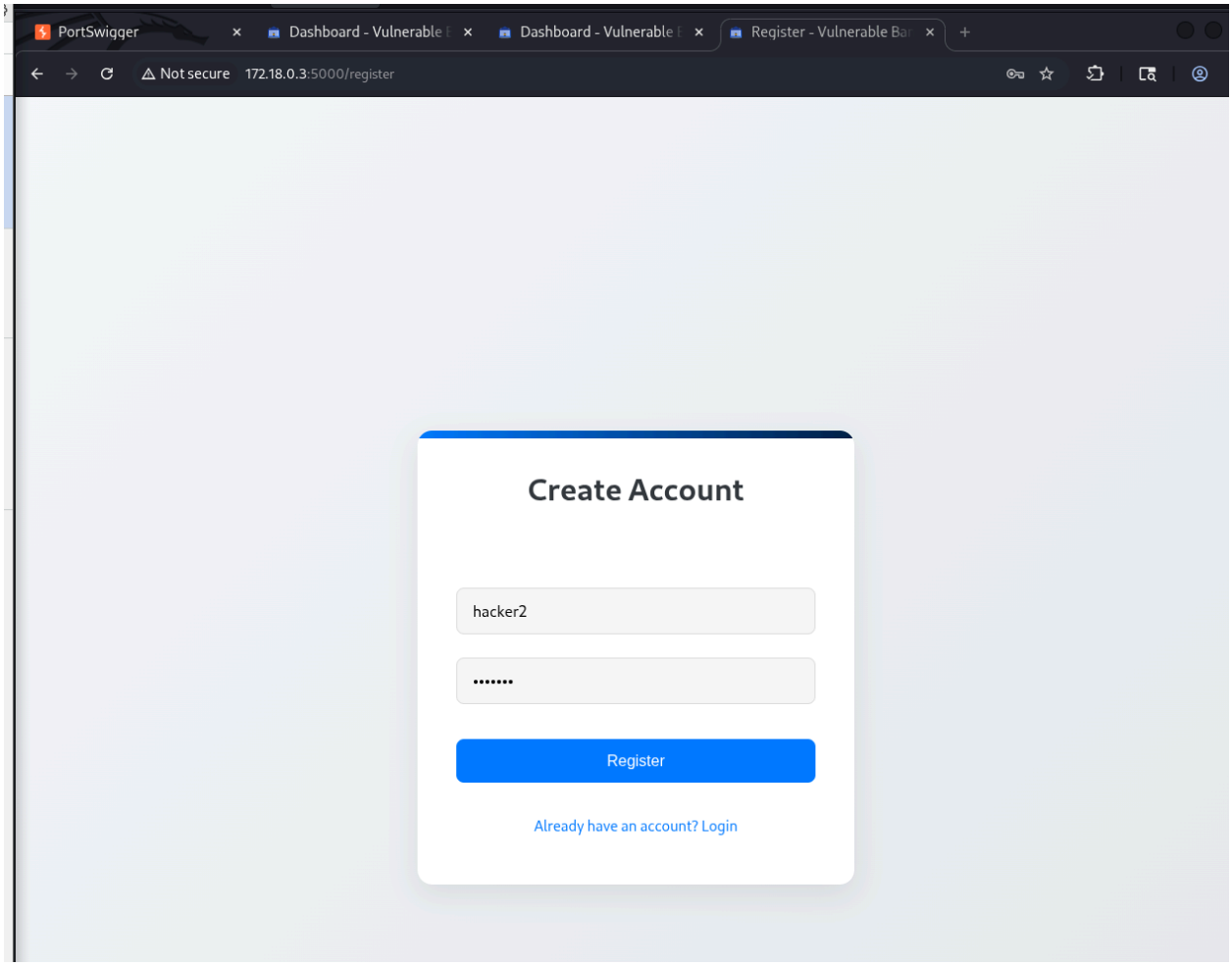


Login page of the admin accessed after using fuff to brute force the login details

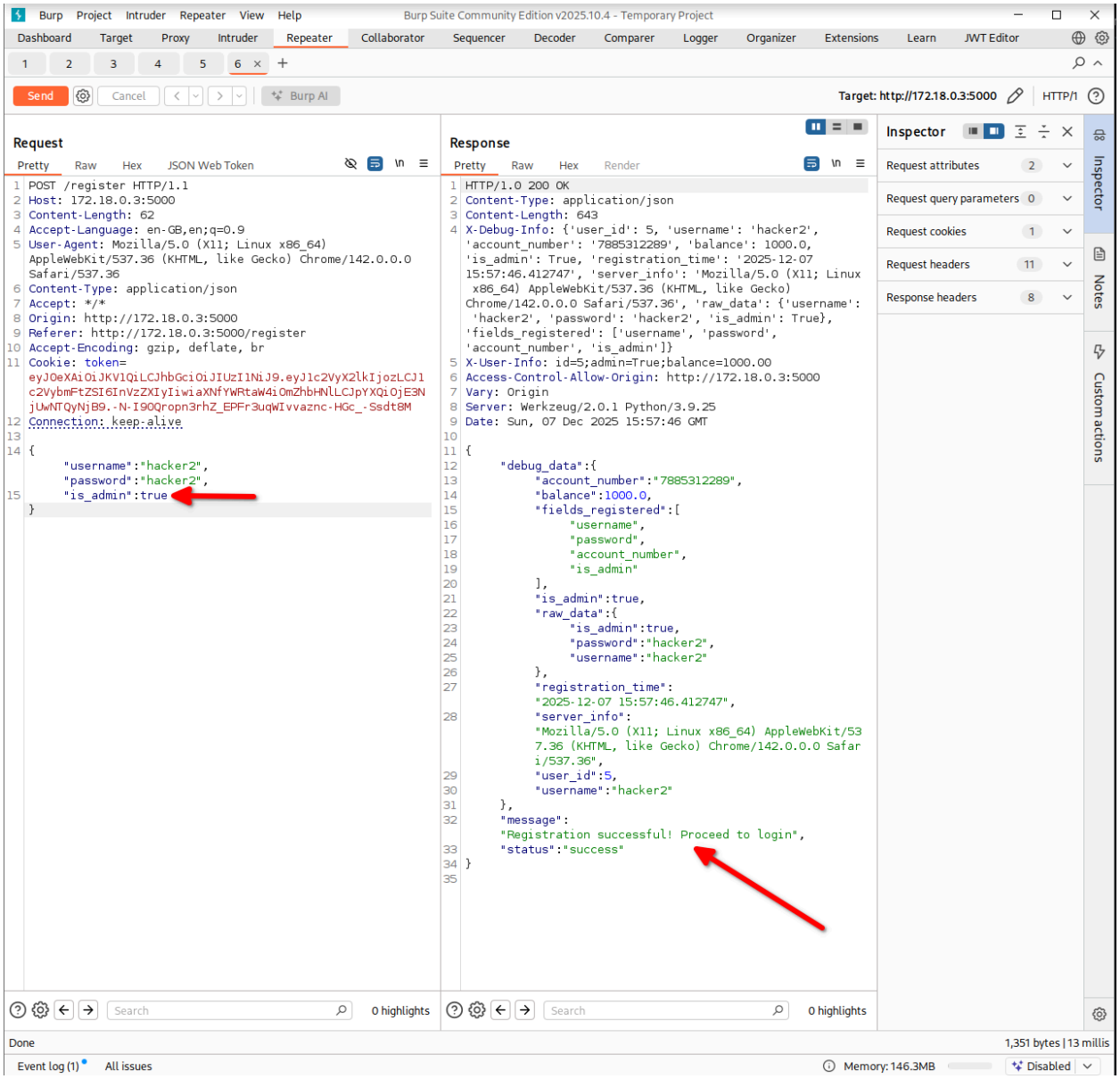
Business Impact: Attackers gain complete control of Admin and other users' accounts in the system, read their personal data, and perform sensitive actions on their behalf.

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

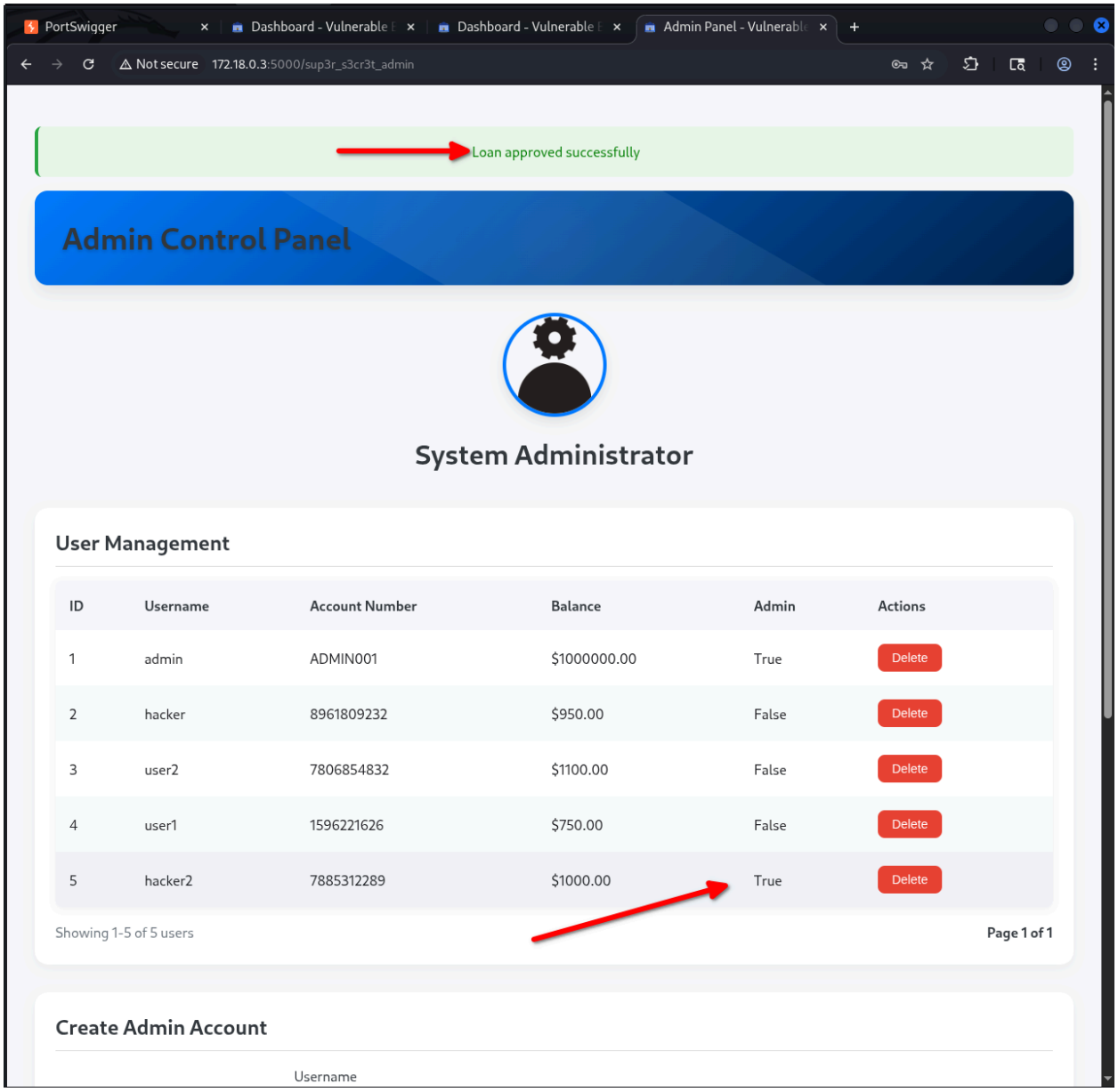
Affected Endpoint: POST/register



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



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!