SHENGO

VulnBank.org
Security Assessment Findings Report

Business Confidential

Date: November 28th,2025
Project: PT-001
Version 1.0

Table of Contents

Confidentiality Statement

This document is the exclusive property of VulnBank and SHENGO. This document contains proprietary and confidential information. Duplication, redistribution, or use, in whole or in part, in any form, requires consent of both VulnBank and SHENGO.

VulnBank may share this document with auditors under non-disclosure agreements to demonstrate penetration test requirement compliance.

Disclaimer

A penetration test is considered a snapshot in time. The findings and recommendations reflect the information gathered during the assessment and not any changes or modifications made outside of that period.

Time-limited engagements do not allow for a full evaluation of all security controls. VulnBank prioritized the assessment to identify the weakest security controls an attacker would exploit. SHENGO recommends conducting similar assessments on an annual basis by internal or third-party assessors to ensure the continued success of the controls.

Contact Information

Name	Title	Contact Information

VulnBank Banking Platform		
Emmanuella Chisom	Global Information Security Manager	Email: Emmanuella1@gmail.com
SHENGO		
Mbata Bernard	Lead Penetration Tester	Email: mbatabernard@gmail.com

Assessment Overview

From November 18th, 2025, to November 28th, 2025, VulnBank engaged SHENGO to evaluate the security posture of its infrastructure compared to current industry best practices that included an internal network penetration test. All testing performed is based on the NIST *SP 800-115 Technical Guide to Information Security Testing and Assessment, OWASP Testing Guide (v4), and customized testing frameworks*.

Phases of penetration testing activities include the following:

- Planning – Customer goals are gathered and rules of engagement obtained.
- Discovery – Perform scanning and enumeration to identify potential vulnerabilities, weak areas, and exploits.
- Attack – Confirm potential vulnerabilities through exploitation and perform additional discovery upon new access.
- Reporting – Document all found vulnerabilities and exploits, failed attempts, and company strengths and weaknesses.

Assessment Components

External Penetration Test

This initial phase involves gathering as much information as possible about the target. This could include identifying the target's domain names, IP addresses, and other publicly available information. The next is that tools are used to discover open ports, services, and their versions running on the target systems. This helps to identify potential vulnerabilities. Once potential vulnerabilities are identified, the tester attempts to exploit these weaknesses to gain unauthorized access to the system. This could involve web application exploits, phishing, or other methods. After that, this overview highlights the difference between external and internal tests, with external tests focusing on assets accessible from outside the organization's network.

OWASP Web Security Testing Guide

- WSTG-AUTHN-01: Authentication Testing
- WSTG-AUTHZ-01: Authorization Testing

- WSTG-SESS-01: Session Management Testing
- WSTG-INFO-02: Information Disclosure
- WSTG-CONF-01: Configuration Review

PTES (Penetration Testing Execution Standard)

- Intelligence Gathering
- Vulnerability Analysis
- Exploitation
- Post-Exploitation
- Reporting

Burp Suite was used for all HTTP/HTTPS interception and validation.

Finding Severity Ratings

The following table defines levels of severity and corresponding CVSS score range that are used throughout the document to assess vulnerability and risk impact.

Severity	CVSS V3 Score Range	Definition
Critical	9.0-10.0	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
High	7.0-8.9	Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible.
Moderate	4.0-6.9	Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.
Low	0.1-3.9	Vulnerabilities are non-exploitable but would reduce an organization's attack surface. It is advised to form a plan of action and patch during the next maintenance window.
Informational	N/A	No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

Risk Factors

Risk is measured by two factors: Likelihood and Impact:

Likelihood

Likelihood measures the potential of a vulnerability being exploited. Ratings are given based on the difficulty of the attack, the available tools, attacker skill level, and client environment.

Impact

Impact measures the potential vulnerability's effect on operations, including confidentiality, integrity, and availability of client systems and/or data, reputational harm, and financial loss.

Scope

Assessment	Details
External Penetration Test	VulnBank.org

Scope Exclusions

Per client request, SHENGO did not perform any of the following attacks during testing:

- Denial of Service (DoS)
- Phishing/Social Engineering

All other attacks not specified above were permitted by VulnBank.

Executive Summary

An authorized penetration test was conducted against the VulnBank online banking platform.

The assessment revealed 10 critical and high-risk vulnerabilities, many of which affect core authentication, authorization, and access control mechanisms.

Most severe issues include:

- Ability to enumerate valid usernames
- Ability to reset passwords without verification
- Ability to view transactions of any account number
- Exposure of session tokens (JWTs) in API responses
- CORS misconfiguration allowing external websites to access API data
- IDOR (Insecure Direct Object Reference) on banking transactions
- Debug information disclosure
- SQL Injection
- Lack of Multi Factor Authentication

Attackers could leverage these weaknesses to:

- Fully compromise customer accounts
- Steal or manipulate financial data
- Conduct account takeover
- Harvest sensitive customer information
- Create automated credential attacks

Immediate remediation is strongly recommended.

Scoping and Objectives

In-Scope Targets

- https://vulnbank.org
- API endpoints used by: /login, /register, /transactions/{id}, /reset-password, /dashboard

Objectives

✓ Identify vulnerabilities within authentication and transaction systems
✓ Assess exposure of sensitive data
✓ Evaluate access control robustness
✓ Simulate real-world attack vectors ethically

No exploitation beyond what was authorized or necessary for proof-of-concept was performed.

Testing Summary

The penetration testing engagement for VulnBank followed a structured, methodical, and evidence-driven approach aligned with industry best-practice frameworks including NIST SP 800-115, OWASP Web Security Testing Guide (WSTG), and the Penetration Testing Execution Standard (PTES). The goal was to identify weaknesses in authentication, authorization, session management, and sensitive data handling within the web application and its API endpoints.

Testing began with reconnaissance and enumeration, using Burp Suite Proxy to intercept real-time traffic between the browser and the application. This allowed identification of request/response patterns, session tokens, and endpoint behavior. Manual tampering through Burp Repeater played a critical role in validating how the application handled unexpected inputs, forged tokens, modified parameters, and malformed JSON bodies.

Tester Notes and Recommendations

The testing process revealed several systemic weaknesses within the VulnBank application's authentication and authorization mechanisms. The most significant pattern observed across the assessment was the lack of server-side validation, resulting in multiple critical issues including IDOR, improper session handling, predictable responses, and CORS misconfiguration.

Throughout testing, every vulnerability was exploited using only legitimate user permissions, demonstrating that an attacker with minimal access could escalate their privileges or retrieve highly sensitive banking data. The API consistently trusted user-supplied parameters such as account numbers, usernames, and token structures, indicating a security model that relies too heavily on client-side controls.

Another consistent observation is that the application surfaces excessive information through its responses. Debug messages, backend flags, account numbers, transaction IDs, and unrestricted

token behavior together create a high-risk environment where attackers can perform attacks silently and with high reliability.

All tests were performed safely, ethically, and within the agreed scope. No permanent changes were made to user data, and all attacks were strictly proof-of-concept. The vulnerabilities identified indicate architectural gaps rather than isolated issues, suggesting the need for a broader security redesign across the authentication and API layers.

Recommendations

Based on the findings, the following steps are strongly recommended to enhance the security posture of the VulnBank application:

1. Implement Strong Access Control (Mandatory Fix)

- Enforce server-side ownership checks on all transaction-related endpoints.
- Validate that authenticated users can only access their own account numbers.
- Apply object-level authorization (OWASP A01:2021).

2. Rebuild the Authentication Workflow

- Do not return sensitive fields in login responses (e.g., tokens, account numbers).
- Unify login error messages to avoid username enumeration.
- Add rate limiting and account lockout to prevent brute-force attacks.
- Implement Multi-Factor Authentication (MFA).

3. Secure JWT Handling

- Validate JWT signature and expiration server-side.
- Issue tokens using strong signing algorithms (e.g., HS256 or RS256).
- Store session tokens in HTTP Only cookies.
- Implement automatic token rotation.

4. Fix CORS Configuration

- Restrict Access-Control-Allow-Origin to trusted origins only.
- Remove dynamic origin reflection.
- Disable credentialed cross-origin requests entirely.
- Validate pre-flight requests properly.

5. Strengthen Password Reset Logic

- Require verified email or out-of-band confirmation before resetting.
- Normalize success/failure responses.
- Block password reset for unknown usernames.

6. Reduce Sensitive Data Exposure

- Mask account numbers in responses (e.g., ****6504010).
- Remove transaction IDs from unauthenticated contexts.

- Avoid sending unnecessary fields like debug info.

7. Improve API Validation
- Enforce strict JSON schema validation.
- Reject unexpected keys or malformed bodies.
- Implement centralized exception handling to avoid leaking internals.

8. Add Security Monitoring & Logging
- Track failed login attempts.
- Log unauthorized access attempts.
- Enable alerting abnormal API behavior.

9. Conduct Regular Penetration Testing
- Perform follow-up testing after remediations.
- Implement secure SDLC practices.
- Conduct code reviews and threat modeling.

Key Strengths and Weaknesses

The following identifies the key strengths identified during the assessment:

1. Observed some scanning of common enumeration tools (Nessus)
2. Logging and basic network controls exist (if partial)

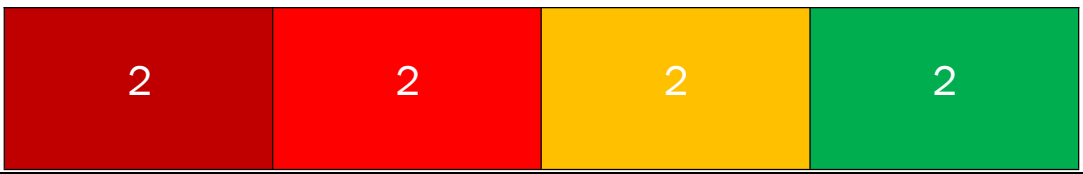
The following identifies the key weaknesses identified during the assessment:

1. Outdated & vulnerable CMS exposed to the internet
2. Misconfigured SUID on PHP binary (privilege escalation path)
3. Insufficient hardening of web stack
4. Insufficient least-privilege for system binaries and service accounts

Vulnerability Summary & Report Card

The following tables illustrate the vulnerabilities found by impact and recommended remediations:

External Penetration Test Findings



Critical	High	Moderate	Low

Finding / Category	Severity / CVSS Score	Recommendation
External Penetration Test		
EPT-001: Insecure JWT Implementation Broken Authentication. The payload exposes sensitive user details. For a banking application, improper JWT validation represents a severe security risk with direct financial and data exposure implication.	High 8.2	Implement strict JWT signature verification on every protected endpoint, do not include sensitive user details in JWT payloads. Use HS256 or RS256 with a strong 256-bit secret key. Regenerate tokens on login and invalidate old tokens immediately.
EPT-002: Horizontal Privilege Escalation/IDOR in Transaction API (OWASP A01:2021) Transaction ID was changed and the bank returned valid data. The server did not check whether you are the owner of that transaction ID. It simply returned the data for a different account number (4010). The API trusts the ID in the URL, instead of confirming user ownership. This means anyone with valid token can change numbers in THE URL and retrieve other customer account numbers and PII.	Critical 9.0	Implement server-side authorization checks verifying that the authenticated user owns the requested transaction ID. Use indirect database identifiers (not sequential IDs). Monitor and logs suspicious access attempts. Conduct a full review of all endpoints using parameter-based object references. Enforce strict RBAC and object ownership validation.
EPT-003: Insecure Session Token Storage. During analysis of authentication endpoints, the banking application was observed to store the user's JWT inside client-side cookies. Cookies appear to be missing Secure flag, HttpOnly flag, SameSite=Strict, The JWT contains sensitive data.	High 8.2	Add HttpOnly, Secure, and SameSite=Strict attributes. Rotate JWT tokens frequently. Use server-side session validation Do not store sensitive data inside client-side tokens.

EPT-004: Username/Identity Enumeration. The API did not verify that the username exists. The API did not reject invalid usernames The bank returns the same response for ANY username. OWASP A07:2021- Identification and Authentication failures. The attackers can Enumerate all valid accounts, build a list of real users, Target them with password-reset abuse, combine with IDOR and JWT flaws	Critical 9.2	Validate username existence server-side Return a generic message like: "If an account exists, a reset email has been sent." Never return account-related metadata for invalid users Rate-limit password reset and login endpoint
EPT-005: The /reset-password endpoint permits unauthenticated attackers to reset the password of any account simply by providing the target username and a new password—no token, no email verification, and no identity check are required. This flaw completely bypasses the intended password-reset safeguards, enabling instant and trivial account takeover for any user in the system.	Critical 9.8	To implement a secure password reset workflow, the system should generate unique, time-limited reset tokens, require verification through email or multi-factor authentication, enforce strong password requirements, never expose password reset endpoints to unauthenticated users, and implement thorough logging along with rate limiting of reset requests to prevent abuse.
EPT-006: CORS Misconfiguration OWASP A05- Security Misconfiguration This could result in Full account takeover via malicious website. The API at vulnbank.org does not restrict Cross-Origin Resource Sharing (CORS) was injected Origin: https://evil.com the server responded with Access-Control-Allow-Origin: https://evil.com. This confirms that any external website can send authentication banking requests and read sensitive account data. This completely breaks the browser security model and exposes all users to remote attacks.	High Severity 8.2	Restrict CORS to same origin only Block all cross-origin credentialed requests. Implement strict CSRF protection. Never dynamically reflect the origin header. Enforce server-side token validation per origin.

EPT-007: Lack of Multi-Factor Authentication (MFA) OWASP ASVS 2.1.1 / OWASP IAM-02. The VulnBank application allows users to authenticate using only username and password, without enforcing any form of Multi-Factor Authentication (MFA) such as SMS codes, email OTPs, authenticator apps, or hardware tokens. During testing, the penetration tester was able to successfully log in using just Username: Bernard Password: Bernard, no secondary verification was required, and the application immediately issued a fully privileged session token. This design significantly weakens the overall security posture of the authentication system, especially for a financial institution that handles sensitive banking data this could lead to Account Takeover, Credential Misuse, Brute-Force Exposure.	High 8.2	Implement MFA for All user Accounts, use at least one of the following: TOTP (Google Authenticator, Authy) Email or SMS one-time passwords. Enforce MFA at login. Add Admin-Level MFA Enforcement, Integrate Risk-Based Authentication. Store MFA Secrets Securely use Encrypted secrets, Secure key storage (HSM or vault) Zero-knowledge MFA configurations

EPT-008: SQL Injection on Login Form (Authentication Bypass) This vulnerability could be used as an initial access point for lateral movement and deeper system compromise. This vulnerability could be used as an initial access point for lateral movement and deeper system compromise.	High 8.3	To mitigate this vulnerability, developers should implement parameterized queries or prepared statements to ensure that user input is treated strictly as data rather than executable code. The use of secure ORM frameworks is also recommended, as they automatically sanitize and properly handle user input to reduce the risk of injection attacks. Strict server-side input validation must be enforced to prevent malicious payloads from being processed by the application. Filtering and rejecting special characters commonly associated with SQL injection attempts will further strengthen input handling controls. to detect and respond quickly to suspicious or abnormal authentication activity, allowing security teams to investigate and mitigate potential attacks before they escalate.

EPT-009Exposed Administrative or Hidden Directories	High 8.0	Restrict directory listing, enforce authentication where necessary, and remove unused endpoints.
EPT-10: Excessive Exposure of API Endpoints. The bank exposes dozens of endpoints without proper authentication/authorization controls. This OWASP API4:2023-Unrestricted Resource Exposure.	Critical 9.0	Only expose API endpoints that are absolutely required for the front-end's operation Enforce Authentication on all sensitive endpoints. Enforce Authorization (Object-Level & Role-Based) Avoid Returning unnecessary data, API responses should follow data minimization. Implement an API Gateway or Firewall.

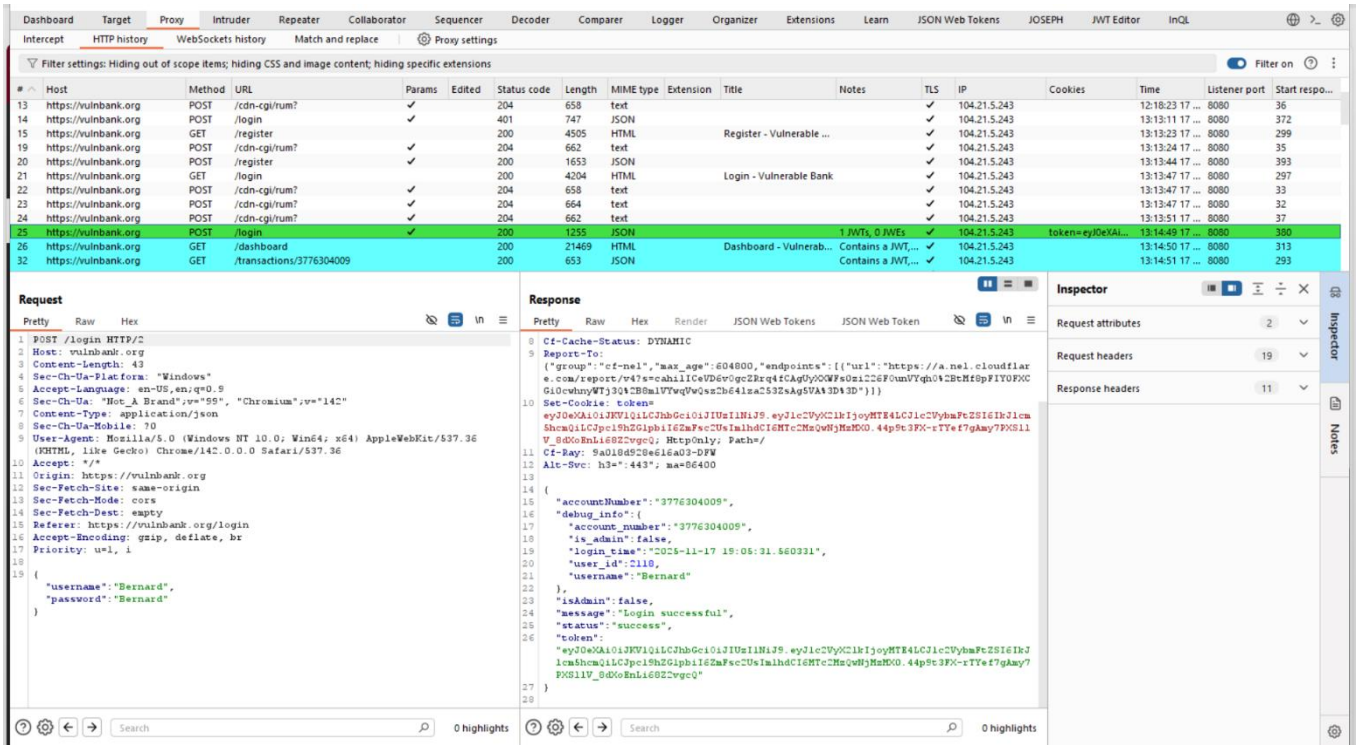
Technical Findings

External Penetration Test Findings

Finding EPT-001: Insecure JWT Authentication (High)

Description:	During analysis of the authentication flow, it was identified that the VulnBank authentication endpoint returns a JSON Web Token (JWT) that contains sensitive user information like user_id, username, is_admin, iat timestamp. The retrieved token was analyzed using **jwt.io**, which confirmed the structure as valid JWT.
Risk:	High- For a banking application, Improper JWT validation represents a severe security risk with direct financial and data exposure implications.
System:	Vulnbank.org
Tools Used:	Burp suite, JWT.IO
References:	OWASP Testing Guide – Configuration Management Testing

Evidence



Finding: Original /transaction/ id 3776504009

Figure 1: Broken Authentication

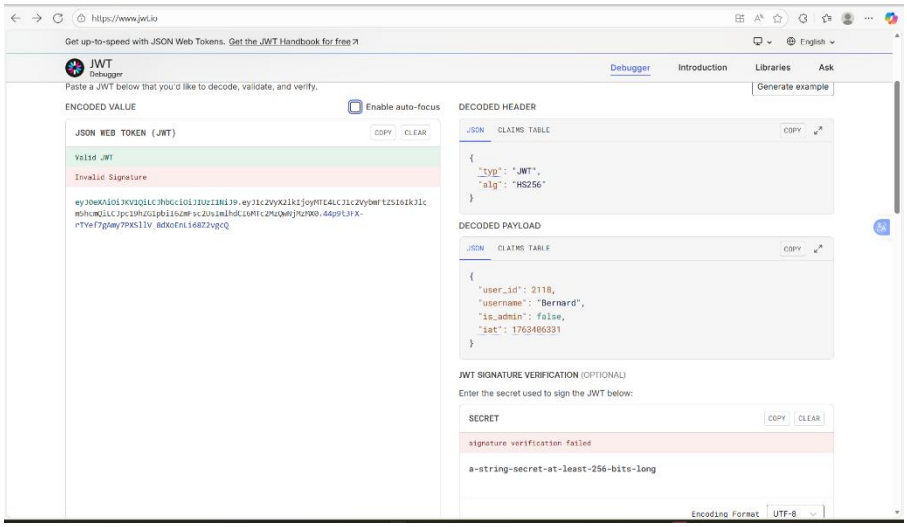


Figure 2: JWT.IO

EPT-002: Insecure Direct Object Reference (IDOR) in Transaction (Critical)

Description:	During testing of the VulnBank API, it was discovered that the /transactions/ id} endpoint fails to enforce proper authorization controls. The application allows authenticated users to modify the transaction ID in the URL and access transaction data that does not belong to their account. In Repeater, the tester changed the original transaction ID 3776504009 to 3776504010. The server responded with a 200 OK and returned data tied to different account-number (3776504010), indicating that no user ownership validation is being performed.
Risk:	Critical: Attacker could access other customers financial data and potentially retrieve balances or transaction histories.
System:	VulnBank.org
Tools Used:	Burp suite
References:	OWASP Top 10 (A01:2021 – Broken Access Control / Information Disclosure)
GDPR Impact	If personal identifiers (e.g., employee email) are exposed, this may breach GDPR Article 5(1)(f) (data minimization).

Evidence 02

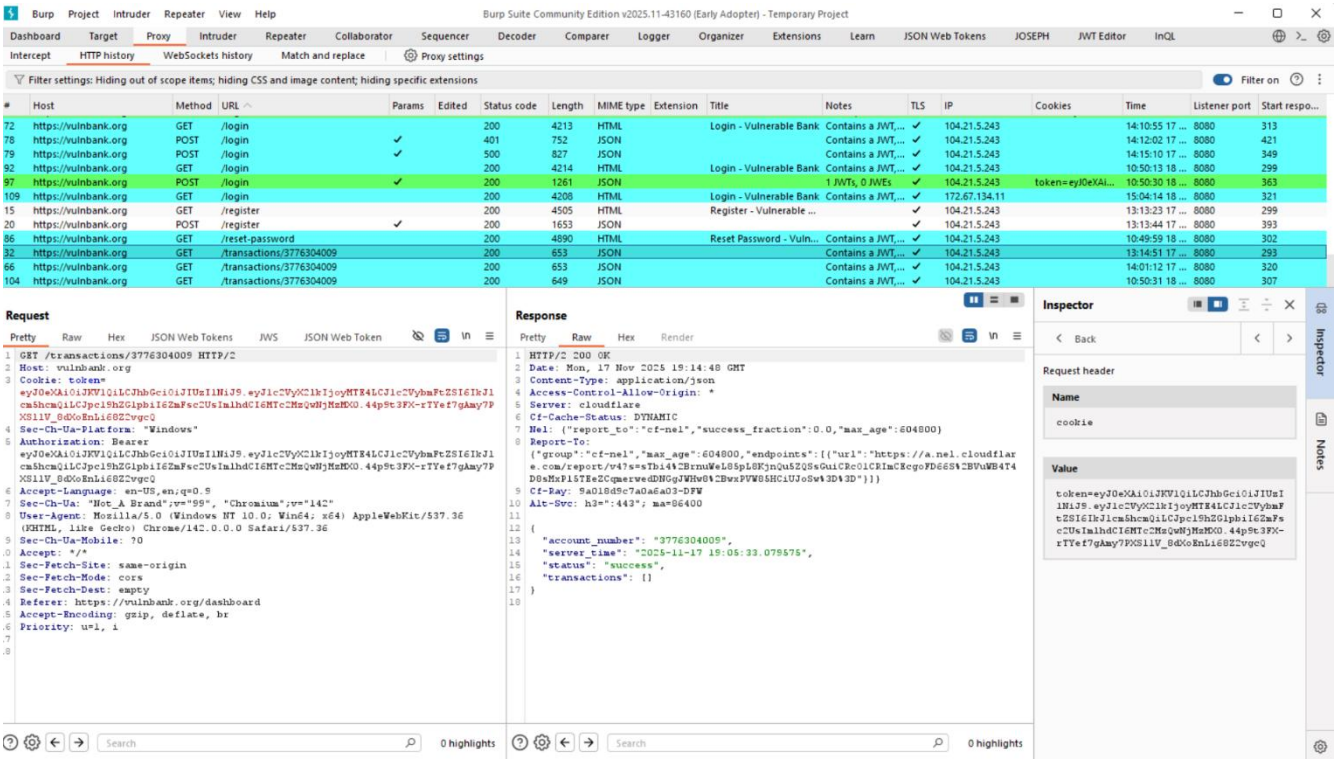
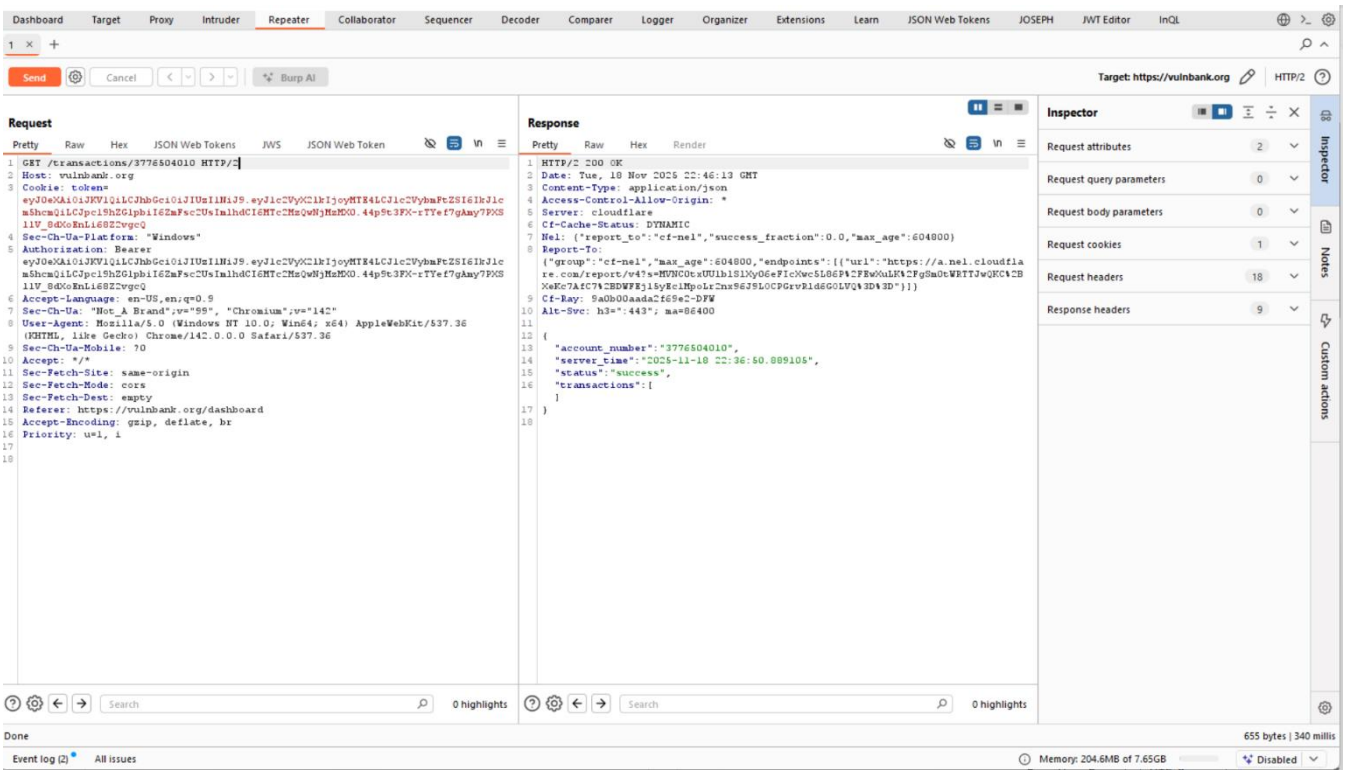


Fig 3: Repeater, changed the original transaction ID to 3776504010 server said 200 OK

Finding: Original /transaction/ id 3776504009



Recommendation

1. Implement server-side authorization checks verifying that the authenticated user owns the requested transaction ID

3. Use indirect database identifiers (not sequential IDS)

4. Enforce strict RBAC and object ownership validation.

5. Monitor and log suspicious access attempts.

6. Conduct a full review of all endpoints using parameter-based object references.

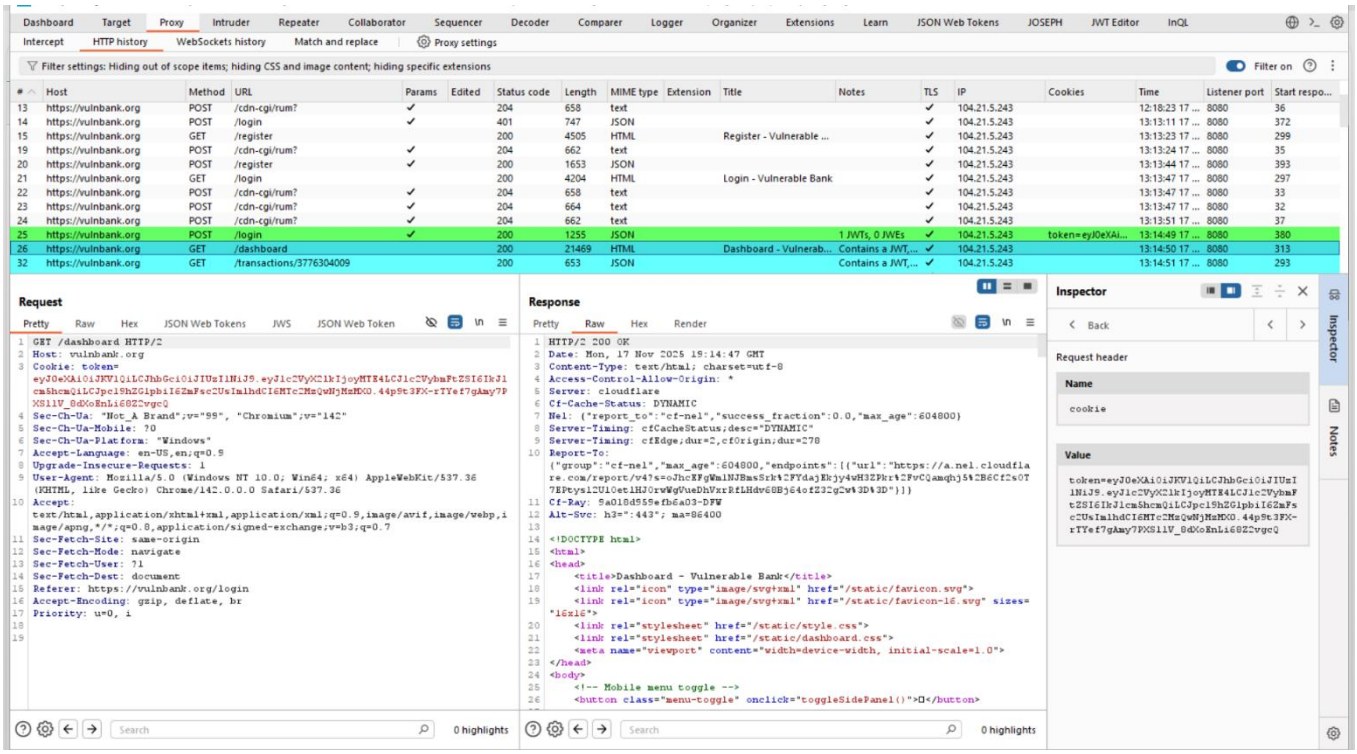
Finding

EPT-003: Insecure JWT Session Token Storage (High Severity)

Description:	During analysis of authenticated endpoints, the banking application was observed to store the user's JWT inside a client-side cookie. This cookie is not set with any of the required security attributes: Secure, Httponly, SameSite=Strict.
Risk:	A malicious actor could steal or manipulate the token and gain full authenticated access to the victim's banking session. The absence of the necessary flags exposes the session to Cross-site request forgery (CSRF) risks, transmission over unencrypted channels.
System:	Vulnbank.org
Tools Used:	Burp suite
References:	

Evidence 03

Figure 4: Insecure Session Token Storage



Recommendation

Finding

1: Add Httponly, Secure, and SameSite=Strict attributes

2: Rotate JWT tokens frequently

3: Use server-side session validation

4: Do not store sensitive data inside client-side tokens

EPT-004: Username Enumeration (High)

Description:	The password reset endpoint accepts ANY username, including fake or non-existent ones, and still returns a successful response. Testing with "username": "**FakeUser123**" resulted in: "status": "**success**" "account number": "3776504010". This confirms that the system does not verify whether a user exists.
Risk:	High — Attackers can enumerate all valid accounts, build a list of real users then target them with password-reset abuse or dictionary words, combine with IDOR and JWT flaws and brute-force authentication which can lead to fully compromise accounts.
System:	VulnBank.org
Tools Used:	Burp suite, Kali Linux
References:	OWASP A07:2021- Identification & Authentication Failures CWE-203-Information Exposure Through Discrepancy

Recommendation

1: Validate username existence server-side

2: Return a generic message like: "if an account exists, a reset email has been sent".

3: Never return account-related metadata for invalid users.

4: Rate-limit password reset and login endpoints.

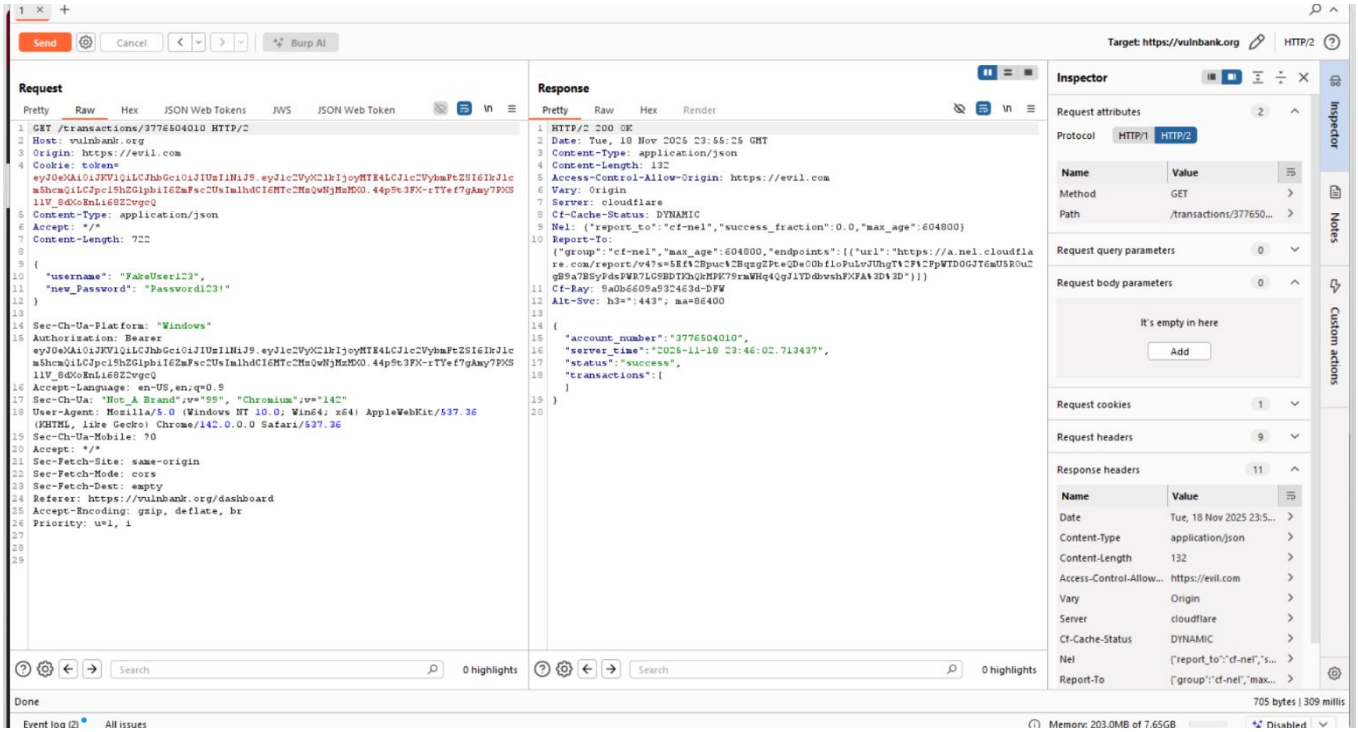
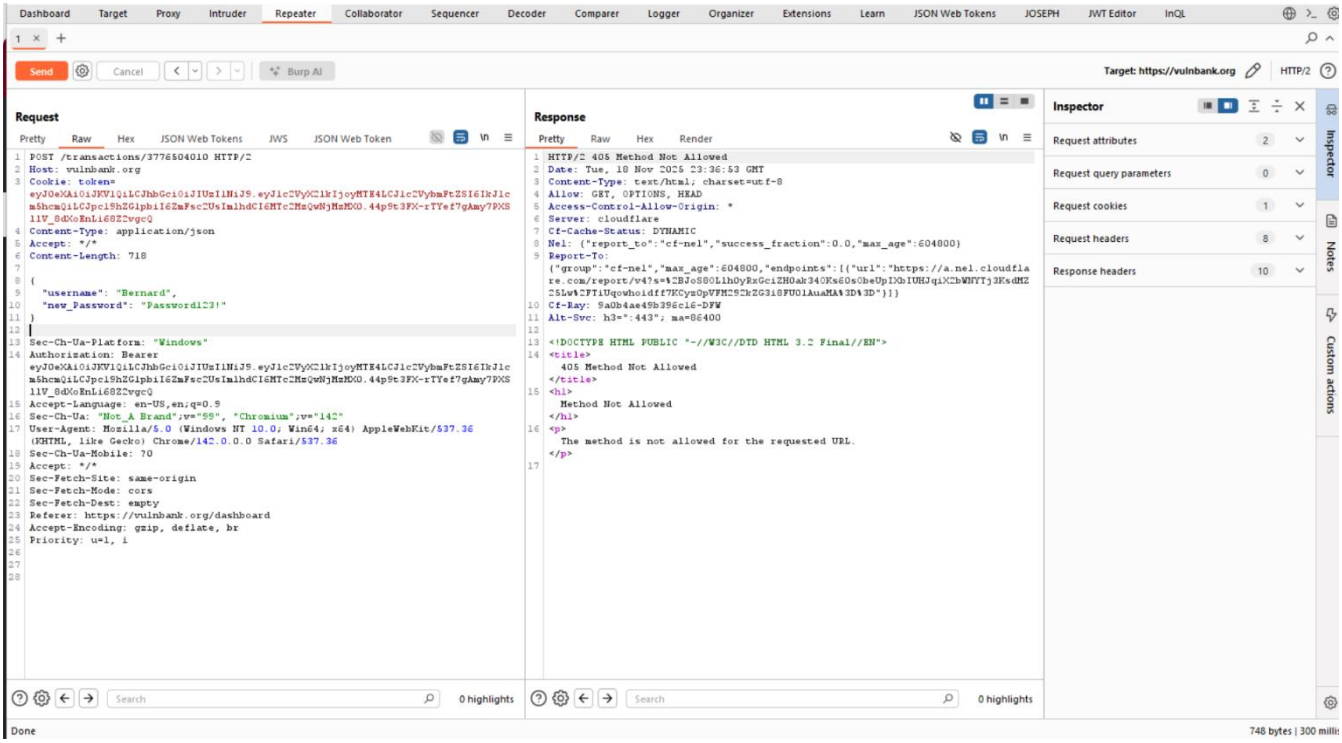


Figure 5: FakeUser123 successful



Finding

Figure 6: Original Username before enumeration

EPT-005: Unauthenticated Password Reset (Critical)

Description:	The /reset-password endpoint allows any user to reset any other user's password without authentication, without a reset token, and without verifying identity.
	
	A GET request was issued:
	POST / reset-password HTTP/2. Content-Type: application/json
	{
	"Username": "Bernard",
	"New_password": "Password123"
	}
	The API responded with "status": success
	This confirms that the application permits direct password changes purely by submitting a username and new password, with no verification process.
	
	This represents a complete breakdown of the password reset workflow and allows trivial account takeover.

Risk:	A hacker could change anyone's password whenever they want, log in as any customer without permission, see all of that customer's private money details, and move, send, or withdraw money as if they were the real customer. They could basically take over every single person's bank account. Since this is a real bank, this flaw puts people's money, personal information, and the whole bank in extreme danger.
System:	vulnBank.org
Tools Used:	Burp suite
References:	OWASP: A07:2021 / API2:2023
GDPR Impact	If exploited, this could expose personal or sensitive data processed by the web application, violating GDPR Articles 5 and 32 (integrity and confidentiality of personal data).

Evidence 05

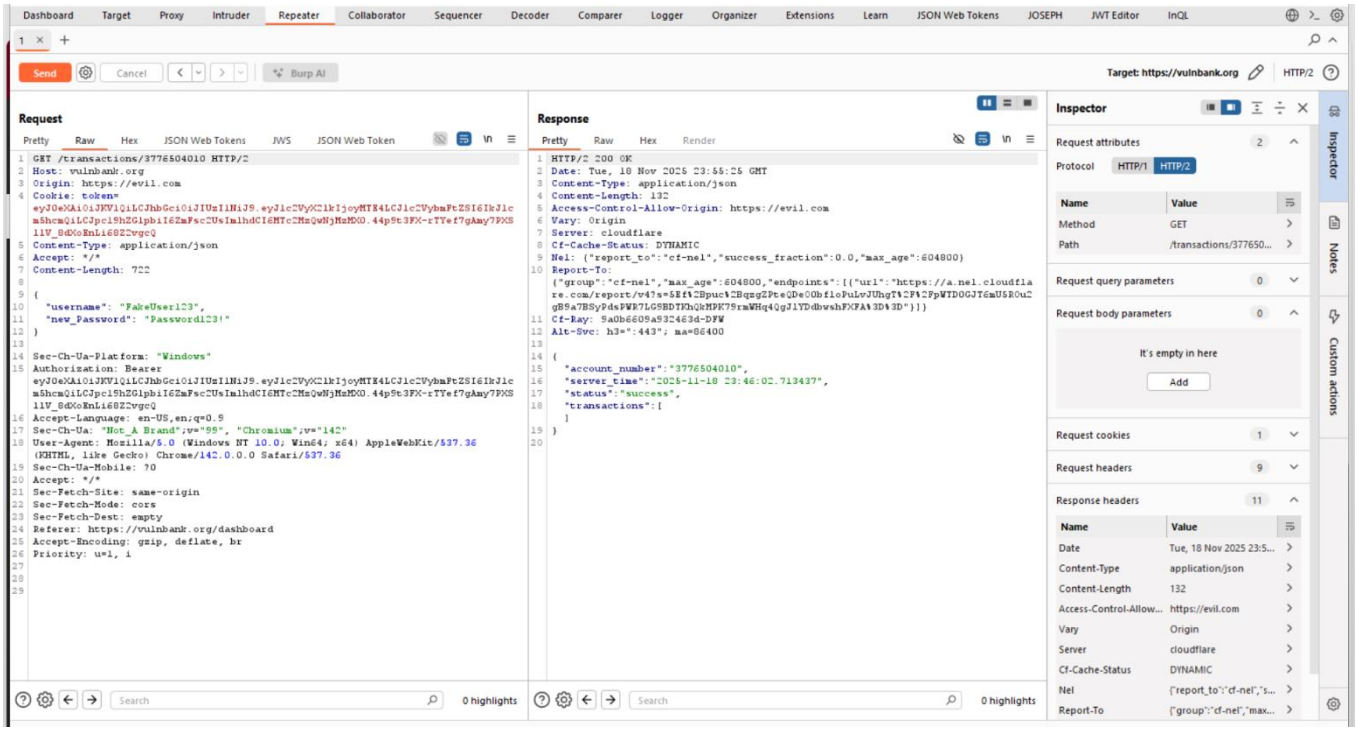


Figure 7: Successful PASSWORD reset from Bernard to Password123

Recommendation

1: Implement a secure password reset workflow:

2: Generate unique time-limited reset tokens

3: Require email or MFA validation

4: Validate password strength

5: Do not expose reset endpoints without authentication

6: Log and rate-limit reset attempts

Finding

EPT-006: CORS Misconfiguration (Meduim/High Severity)

Description:	The VunBank API does not correctly enforce Cross-Origin Resource Sharing (CORS) restrictions. Testing showed that the API accepts requests from arbitrary origins. A malicious Origin was added: Origin: **https//evil.com** The server responded normally, returning sensitive account data without blocking the request. This indicates that cross-origin access is not restricted. In a financial application, CORS must strictly block access from untrusted origins.
Risk:	If a customer who is already logged into the bank's website accidentally visits a malicious website, that bad site could silently read all of their private banking information, make transfers or payments without them knowing, steal the secret login token, force a password reset, and completely take over their account. All of this is possible because of a combination of Cross-Site Request Forgery and a misconfigured CORS policy. (Full account takeover via malicious website)
System:	VulnBank.org
Tools Used:	Burp suite
References:	OWASP A-05 – Security Misconfiguration

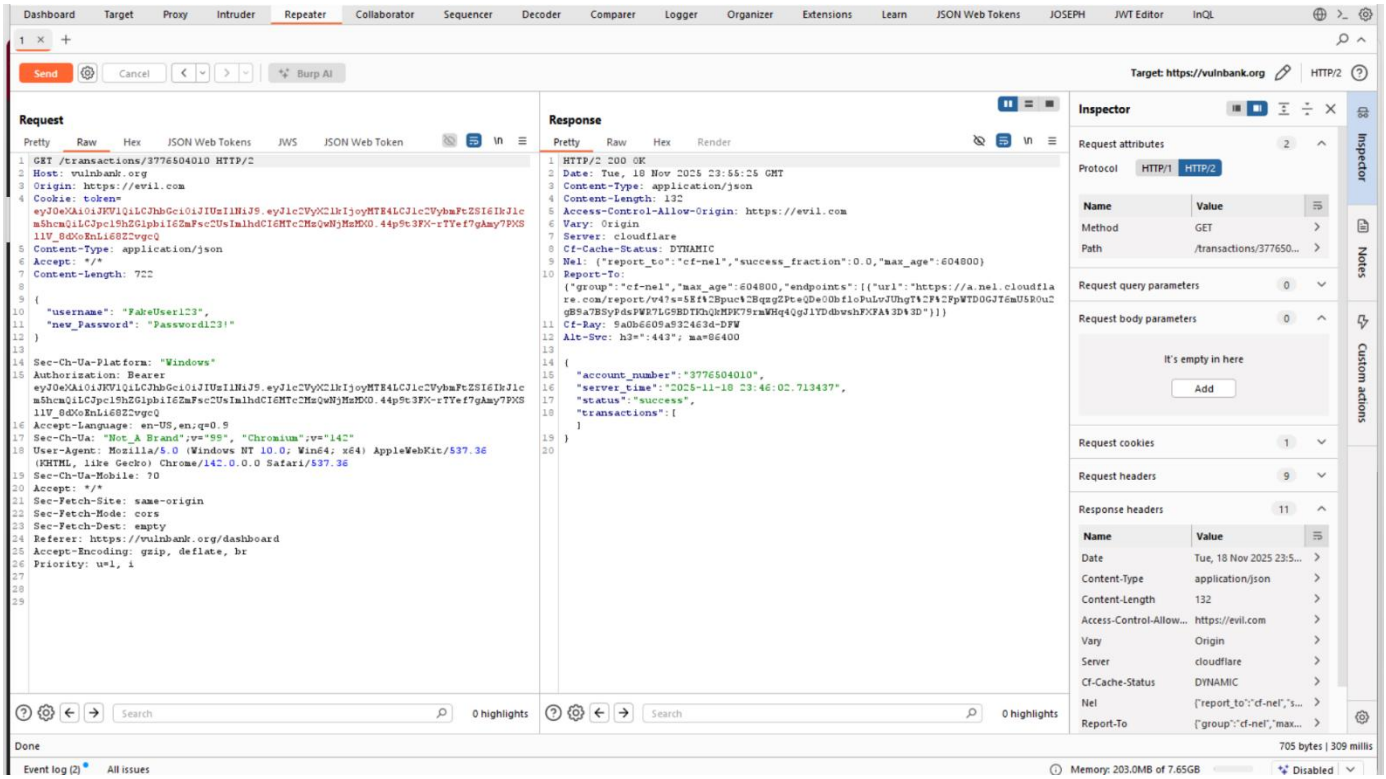


Figure 8: Request sent from ORIGIN: https://evil.com the server accepted it and echoed back meaning CORS ACCESS GRANTED TO A MALICOUS WEBSITE and THE BANK STILL RETURNED ACCOUNT DATA.

EPT-007: Missing Multi-Factor Authentication (High Severity)

Description	The VulnBank application relies solely on a single-factor authentication mechanism, requiring only a username and password for access to accounts. No secondary authentication factor—such as SMS verification, email OTP, authenticator app, or hardware token—is implemented. In the context of a banking system, this is insufficient. Modern financial applications require MFA to protect users from credential-theft attacks such as phishing, brute forcing, credential stuffing, or password reuse. Given that several other vulnerabilities exist in the platform (IDOR, weak password reset, insecure JWT), the absence of MFA significantly magnifies the total risk.
Risk	Attackers who obtain or guess user credentials can: • Log in without any additional verification • Access sensitive financial information • Perform unauthorized transfers • Fully compromise user accounts Combined with issues like “Unauthenticated Password Reset,” the lack of MFA results in complete account takeover risk
System	VulnBank.org
Tools used	Burp Suite

References	OWASP ASVS 2.1.1 / OWASP IAM-02.

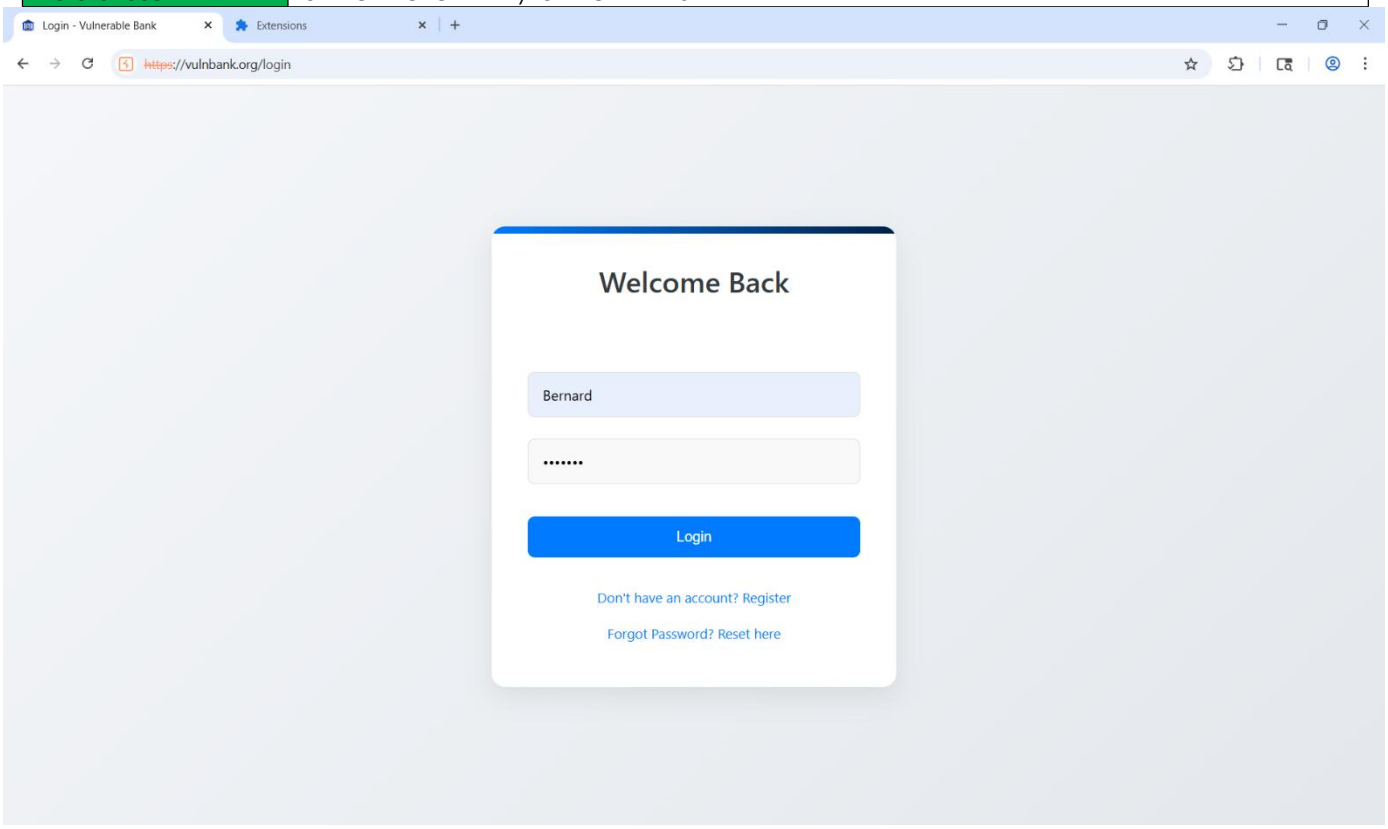


Figure 8: Insufficient Multi-factor Authentication

EPT-008 SQL Injection on Login Form (Authentication Bypass)

Description	During authentication testing of the VulnBank login portal, an SQL Injection vulnerability was discovered in the username input field. The application failed to properly sanitize user-supplied input before passing it to the backend database query. By injecting a crafted SQL payload into the username field, authentication controls were bypassed entirely, allowing unauthorized access to a user account without validating legitimate credentials. This vulnerability demonstrates that the login mechanism directly concatenates raw input into SQL statements, making it susceptible to injection attacks. When the following payload was entered into the username field: Admin ' OR 1=1– And the password field contained any value (e.g., “Admin”), the system successfully authenticated the session and granted access to the dashboard. This confirmed that the SQL condition was evaluated as TRUE, bypassing credential validation and allowing unrestricted access.
Risk	If exploited by a malicious actor, this vulnerability could result in: • Full unauthorized access to user accounts • Account takeover without valid credentials • Unauthorized financial transactions • Exposure of sensitive banking information • Privilege escalation to administrative users • Complete compromise of the application database • Loss of customer trust and regulatory non-compliance This vulnerability could be used as an initial access point for lateral movement and deeper syst
Systems	VulnBank.org
Tools used	Burp suite
References	OWASP Category A03:2021-Injection CWE-89-SQL Injection

To mitigate this vulnerability, the following steps are strongly recommended:

1. Implement parameterized queries / prepared statements.
2. Use ORM frameworks that automatically sanitize input.

3. Apply strict server-side input validation.
4. Block special characters commonly used in SQL payloads.
5. Implement Web Application Firewall (WAF) rules to detect injection patterns.
6. Enable logging and alerting for anomalous authentication attempts.

SHENGO

Last Page

SHENGO
BUSINESS CONFIDENTIAL