VulnBank Audit Report(PCi-DSS 4.0)

Prepared by: Falilat Owolabi
Date: Dec 26th 2026
Target Application: Vuln Bank website
Audit Window: Jan 19th - Jan 26th 2026
Audit Type: PCI-DSS Compliant

1. Format

The PCI DSS requirement was used to analyse the vuln Bank App in addition with reports of the OWASP Top 10 Application analysis done in week 3 and 4, This report aims the understanding the vulnbank in assessing if the requirement were implemented in accordance with PCI-DSS 4.0

2. Executive Summary

The VulnBank application demonstrates multiple security weaknesses that place it out of compliance with several PCI DSS v4.0 requirements, particularly Requirement 6 (Secure Software Development) and Requirement 7 (Strong Access Control). The application's API endpoints expose sensitive banking functionality with insufficient authentication, authorization, and runtime protections, making them highly susceptible to abuse.

Attackers can exploit weak JWT handling, broken authorization logic, and insecure API design to access unauthorized account data, perform financial transactions, and escalate privileges.

NOTE: Not all PCI DSS requirements are assessed but the ones that are specifically related to API in the webAPP for example Requirement 1 is not accessed as it is not required for API and requirement 2.27 is the only one assessed in this report as it is required for API specification.

3. Methodology

The Auditing approach combined the information gotten from the report on the testing done in the past weeks, new findings and the requirement of The PCI DSS standard.

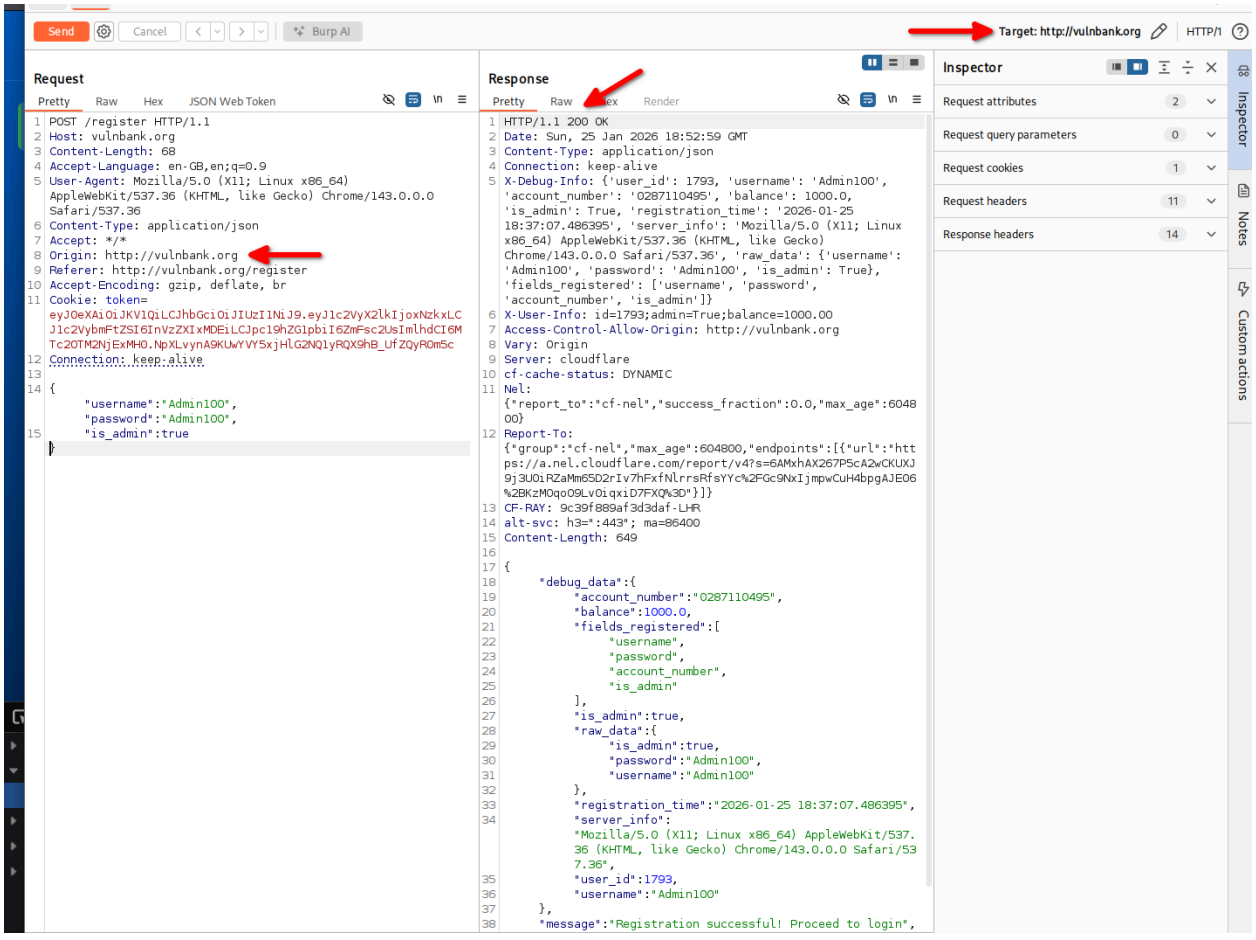
- Manually accessed the codebase and report with PCI-DSS requirement

4. Findings

1) Requirement 2.2.7: All non-console administrative access is encrypted using strong cryptography

Description: Administrative functionality is accessible over unencrypted HTTP, violating PCI-DSS requirements for encrypted administrative access.

The image below indicate non-console admin access is possible through an unencrypted channel which prone the communication to man in the middle attack



2) Requirement 6: Develop and Maintain Secure Systems and Software

- **6.2.1. Bespoke and custom software are developed securely**

  Status: Non-Compliant

  Assessment:

  No documented secure coding standards or evidence of security testing during development were identified. Multiple input validation and authorization weaknesses were observed during testing.

  Vulnerability:

```
# Vulnerable JWT implementation with common security issues

# Weak secret key (CWE-326)
JWT_SECRET = "secret123"
```

  Risk

  - Token forgery
  - Authentication bypass

  Why

  Use of weak cryptography violates secure coding practices for custom software.

  Vulnerability:

```
    return jsonify({'error': 'Missing transfer details'}), 400

# Vulnerability: No amount validation
amount = float(data.get('amount'))
to_account = data.get('to_account')

conn = sqlite3.connect('bank.db')
c = conn.cursor()

# Vulnerability: Race condition in transfer
c.execute(f"SELECT balance FROM users WHERE id={current_user['user_id']}")
balance = c.fetchone()[0]

if balance >= amount:      Commando-X, 11 months ago • Added new functionalities with vulnerabilities,…
    # Vulnerability: SQL injection possible in to_account
    c.execute(f"UPDATE users SET balance = balance - {amount} WHERE id={current_user['user_id']}")
    c.execute(f"UPDATE users SET balance = balance + {amount} WHERE account_number='{to_account}'")
    conn.commit()
```

  Risk:

  Negative transfer will be successful

Why:
Missing input validation in custom software.

- ## Requirement 6.2.2 Annual Developer Training

  Assessment:
  The presence of multiple common secure coding issues, including SQL injection and weak JWT handling, suggests that secure coding practices may not be consistently applied. This indicates potential non-compliance with PCI DSS v4.0 Requirement 6.2.2, which mandates annual secure coding training for developers.

  Conclusion:
  This requirement cannot be proven with code and the API documentation provided for testing, but the code review confirmed the availability of common vulnerabilities that are not expected to be there if developers are well trained. The management should provide a proven document that indicates the developers are trained in accordance with PCI-DSS standard and if not a programme should be organised to train developers to avoid having common vulnerabilities that are easily exploited in the codebase.

- ## Requirement 6.2.3 – Software Code Review

  Assessment
  Multiple critical vulnerabilities identified in the VulnBank application during the week3 and 4 of **OWASP Top10 API Assessment**, including SSRF, insecure JWT handling, and broken authorization, indicate that security-focused code reviews are either not being performed or are ineffective. This suggests non-compliance with PCI DSS v4.0 Requirement 6.2.3, which requires that custom software be reviewed for security vulnerabilities prior to production deployment.

  Conclusion. Although no document was submitted to show the system has undergone software code review with the specialised team, from the analysis of the system it can be said that The PCI-DSS requirement is not applicable before the deployment of the software for use.

- **6.2.4- Prevent Common Vulnerabilities**

The VulnBank application contains multiple common vulnerabilities, including insecure JWT implementation, mass assignment, and broken object-level authorization. These issues indicate that secure coding practices to prevent known vulnerability classes are not effectively implemented, resulting in non-compliance with PCI DSS v4.0 Requirement 6.2.4.

- **6.3.1 – Vulnerability Management**
  VulnBank lacks a documented and effective vulnerability management process. Multiple high risk vulnerabilities including insecure JWT handling, and broken authorization remain unremediated, indicating failure to identify, prioritize, and address security weaknesses in accordance with PCI DSS v4.0 Requirement 6.3.1.

  Conclusion: PCI DSS 6.3.1 ensures organizations continuously identify, prioritize, and remediate vulnerabilities in software before attackers can exploit them which vulnbank developers do not adhere to

- ## Requirement 6.4.1, 6.4.2 – Attack Protection

  6.4.1 – Active Protection (Assessment or Automation)

  Observed Vulnerabilities Requiring Coverage:

  - ➔ JWT authentication bypass (none algorithm, signature verification disabled)
  - ➔ Broken object-level authorization (any token can access any account)
  - ➔ Sensitive data exposure in API responses
  - ➔ Lack of rate limiting and abuse protection
  - ➔ Cors misconfiguration
  - ➔ Business logic abuse (unrestricted transfers, race conditions)

  Compliance Gap

  No evidence of:

  - Annual or post-change application-layer security assessments, or
  - Inline automated protection (WAF, API gateway, runtime protection)

  Status: Non-compliant with 6.4.1

  6.4.2 – Mandatory Automated Protection

  Requirement

- Deploy automated technical controls in front of public facing applications
- Must continuously detect and block attacks
- Must generate audit logs

Observed State

- No WAF, API Gateway, or API runtime protection
- No automated blocking of:
  - Injection attacks
  - Credential abuse
- JWT manipulation
- High-volume or brute-force attacks
- No centralized attack logging or alerting

Status: Non-compliant with 6.4.2

3) Requirement 7.0 – Strong Access Control

It was stated from the API testing done that some endpoints do not possess proper authorization mechanism allowing another user access other user information which include:

- Accessing another user transaction history without their approval
- Using another user card to process payment
- Accessing Admin porta

PCI DSS Impact:

- Violates **Requirement 7.0 – Strong Access Control**
- Violates least privilege and role-based access enforcement

4) Requirement 10.0– Logging & Monitoring

VulnBank is non-compliant with PCI DSS v4.0 Requirement 10.0.

- The absence of application and API-level logging means:
- Attacks cannot be detected
- Incidents cannot be investigated
- Compliance cannot be demonstrated

5) Requirement 11.0 – Security Testing

VulnBank is non-compliant with PCI DSS v4.0 Requirement 11.0.
Key failures include:

- No regular application or API security testing
- No penetration testing
- No automated vulnerability scanning
- No post-change security validation

Top 5 immediate actions (executive checklist):

In accordance with PCI-DSS requirement, the following should be taken into consideration by the management and developers in order to have a properly secured website safe from exploit which include but not limited to:

1. Disallow communication with the server Over unencrypted channel
2. Train developers from time to time on security practice when they are developing code
3. Perform a proper penetration testing and vulnerability assessment before launching the app for user
4. Implement strong access control on all the required endpoints
5. Test your app from time to time and do well to monitor requests on the app and log all the necessary information which can help understand the information users are having access to from the server.

Conclusion

The cumulative findings from this assessment demonstrate that **VulnBank is critically non-compliant with multiple core requirements of PCI DSS v4.0** and exhibits **systemic weaknesses across application security, API security, access control, logging, and security testing practices**.

Throughout the review, vulnerabilities were consistently identified at the **API and application logic layer**, confirming that VulnBank lacks fundamental secure-by-design principles expected of bespoke and custom software handling sensitive financial data.

End of Report!