Penetration Test Report

Aarya Dahal

BSc. Hons. Ethical Hacking and Cybersecurity, Softwarica College of IT and E-commerce

ST5067CEM - Web Security

Nirmal Dahal

August 12, 2023

Abstract

A white box penetration testing was conducted on the Nittam bank. The list of all vulnerabilities are documented in this report. Initially, source code was analyzed and client side vulnerability was checked through the website. OWASP top 10 vulnerability was checked for this report. Severe vulnerabilities were discovered which fully compromised the system exposing sensitive data of the users. Remediation for every vulnerability that has been exploited has been listed below.

AARYA DAHAL

VULNERABILITY ASSESSMENT AND PENETRATION TESTING

FOR THE NITTAM BANK

2023

Confidentiality Statement

The below document contains exclusive and confidential information which is solely owned by Tryhack.me and Aarya Dahal. The content within is not to be duplicated, redistributed, or used in any manner without explicit consent from both parties.

Disclaimer

The contents of this report are based on the assessment conducted within the allotted time frame. Any conclusions or recommendations made in this report are only applicable to the assessment period and are not guaranteed to remain valid beyond that time. Due to the limitations of the time-limited involvement, a comprehensive evaluation of all security safeguards may not have been possible.

Document Details

Company	TheNittam Bank.
Document Title	Vulnerability Assessment and Penetration Testing on TheNittam Bank.
Duration	July-August, 2023
Abstract	Performing a white box penetration test on TheNittam Bank to identify and exploit any potential vulnerabilities and security breaches to make it secure from the attackers.
Classification	Confidential

Table of Contents

Table of Figures

Analytical Summary

The Nittam Bank recently underwent a comprehensive white box penetration testing to evaluate the security posture of its system. The assessment encompassed a thorough analysis of the source code, along with an examination of client-side vulnerabilities through the bank's website.

Several security vulnerabilities were discovered throughout the examination, opening up opportunities for hacking, data breaches, and other forms of exploitation. On the login pages for user, manager, and cashier accounts contained SQL injection vulnerabilities that allowed unauthorized access to these accounts. A business logic flaw was also found, allowing for unauthorized financial transfers and manipulation of transfer amounts. These results highlight how urgent it is to address not only the technical components of these vulnerabilities but also the security consequences they imply.

The operations of the bank, client data, and overall security posture are potential targets of these vulnerabilities, which are examined in detail in this report along with their impact on the bank. Each discovered vulnerability is presented within the framework of the OWASP Top 10, highlighting the system's vulnerabilities. To strengthen the bank's defenses against potential cyber threats and data breaches, the report also includes concrete recommendations for risk reduction and remediation. CVSv3.1 was used to identify the level of critical threats it possessed for the system.

The conclusions drawn from this assessment serve as a basis for improving Nittam Bank's security strategy, supporting its initiatives to preserve private financial data, uphold client confidence, and guarantee compliance with industry best practices.

Scope

Assessment	Details
White box Penetration Test	TheNittam Bank

Objectives

The objectives of conducting a white box penetration testing for The Nittam Bank are listed below.

Identify Vulnerabilities

The primary objective is to systematically identify vulnerabilities present in the bank's systems, applications, and infrastructure. This includes uncovering weaknesses that might be exploited by attackers to compromise sensitive financial data, gain unauthorized access, or disrupt services.

Assess Security Controls

Evaluate the effectiveness of existing security controls and measures implemented by The Nittam Bank. This involves analyzing the adequacy of access controls, authentication mechanisms and other security measures in place.

Test Business Logic

Test the application's business logic to uncover vulnerabilities that might not be apparent through automated tools. Identify potential flaws in how the bank's software processes transactions, authorizes actions, and enforces rules.

Assess OWASP Top 10

Specifically target the OWASP Top 10 vulnerabilities. By addressing these vulnerabilities, the bank can significantly reduce the likelihood of successful attacks.

Discover Attack Vectors

Identify potential entry points and attack vectors that could be exploited by hackers. This includes scrutinizing the source code, application interfaces, user inputs, and interaction points between different system components.

Provide Remediation Recommendations

Offer detailed recommendations to address the identified vulnerabilities. Provide guidance on how to patch security holes, improve application code, enhance access controls, and bolster overall system security and help prioritize security initiatives and allocate resources to areas most susceptible to threats.

By attaining all of these objectives, TheNittam Bank will be able to address security flaws, fortify its security barriers, and guarantee the availability, integrity, and confidentiality of its financial services.

Methodology

Pre-Assessment Planning

- During this phase, the assessment's scope, objectives, and rules are established to ensure a clear direction.
- Gathering documentation and comprehending the system's architecture provide a solid foundation for the upcoming assessment.

Vulnerability Assessment

- This step involves a comprehensive scan for OWASP Top 10 vulnerabilities, ensuring critical risks are identified.
- Manual testing is employed to uncover nuanced vulnerabilities that automated tools might overlook.

Exploitation and Validation

- Detected vulnerabilities are exploited to understand their potential impact and validate their existence.
- The business logic, transaction processes, and authorization mechanisms are thoroughly tested to identify any weaknesses.

Data Manipulation and Transfer Testing

- The funds transfer functionality undergoes scrutiny to detect possibilities of unauthorized transactions.
- Evaluating data exposure risks through manipulation assesses the system's resilience against unauthorized data access.

Reporting and Recommendations

- Findings are documented with technical specifics and supporting evidence, allowing for informed decision-making.
- Assigning risk ratings guides from CVSsv3.1 prioritization efforts, and actionable steps are suggested for mitigation.

Vulnerabilities Report

SQL injection

Description

SQL injection vulnerabilities were identified on the login pages of user, manager, and cashier accounts. Due insufficient input validation on these login pages, the payload 'OR ‘1’=’1’-- - was injected in the password field to bypass authentication mechanisms and gain unauthorized access to the accounts of the first user, manager and cashier.

Impact

The SQL injection vulnerability discovered within the system grants unauthorized access to sensitive customer data, enabling unauthorized modification or deletion of financial records. Moreover, it facilitates unauthorized account access, leading to illicit transactions and breaches of regulatory compliance. Beyond the immediate impact on data and transactions, this vulnerability could disrupt normal business operations, tarnish the bank's reputation, and result in significant financial losses.

Remediation

- Rewrite SQL queries to use parameterized queries or prepared statements.
- Sanitize and validate user inputs before embedding them in queries.

Reference

CWE-89

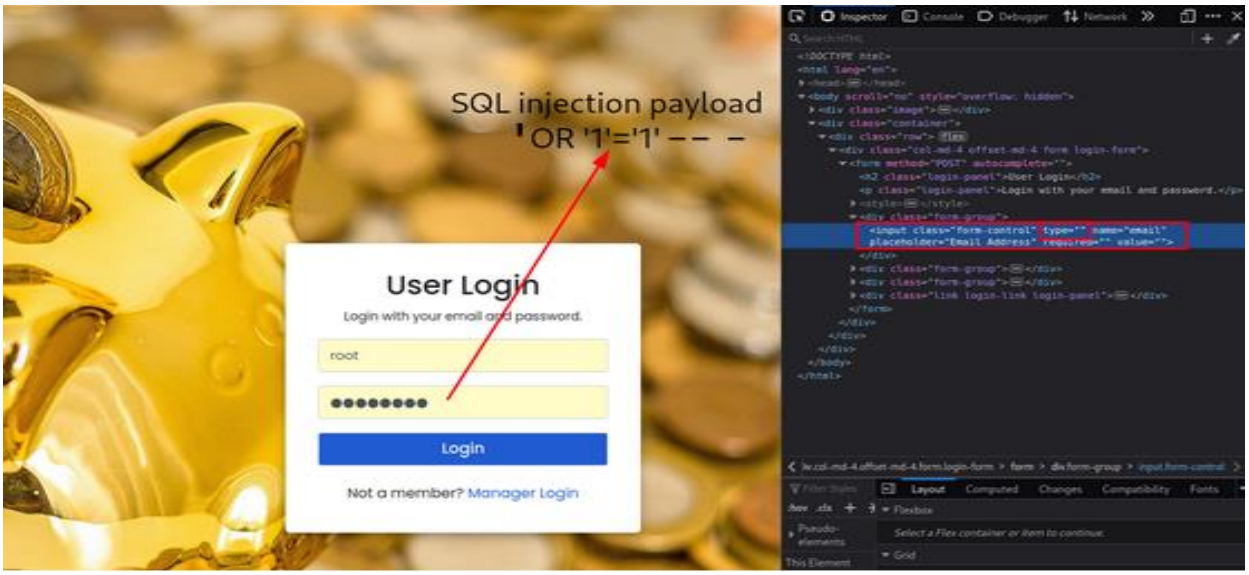
Proof of concept

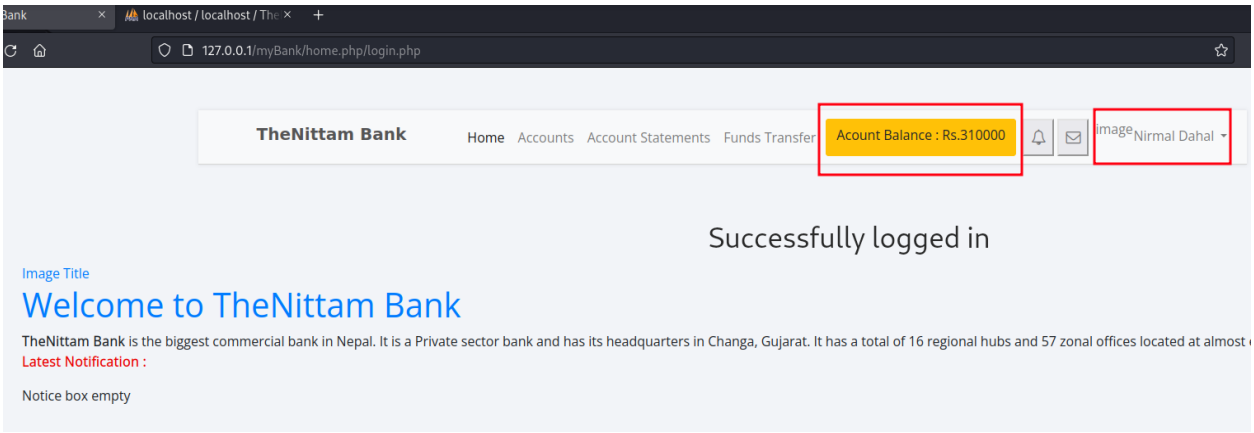
Firstly, the ‘root’ was used in the input field of email. Then, the page was inspected to remove the email type, this bypassed the client side validation. In the password filed the above

mentioned payload was injected which successfully logged into the account of the first user ‘Nirmal Dahal’.

Figure 1

SQL injection in login page





The initial balance of the user was Rs.31000. Due to the SQL injection login authentication was also bypassed.

CVSsv3.1

The CVS score for SQL injection is 6.4
CVSS:3.1/AV:P/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:L

Sensitive data exposer

Description

When logged in into the user ‘Nirmal Dahal’ sensitive data transaction history with the account number of another user, and personal information was exposed.

Impact

It led to the compromise of confidential customer information, including financial records, personal identification, and transaction details. This breach of data confidentiality jeopardizes customer privacy and opens the door to identity theft and financial fraud. Additionally, regulatory non-compliance might result in legal consequences, fines, and damage to the bank's reputation. The loss of customer trust and potential legal liabilities could ultimately impact the bank's financial stability and standing within the industry.

Remediation

- Enforce strict access controls and permissions.
- Store only necessary sensitive data and regularly delete obsolete records.
- Deploy a WAF to filter incoming traffic.
- Conduct frequent security assessments.
- Develop and test an incident response plan.
- Continuously monitor and audit systems and data access logs.

Reference

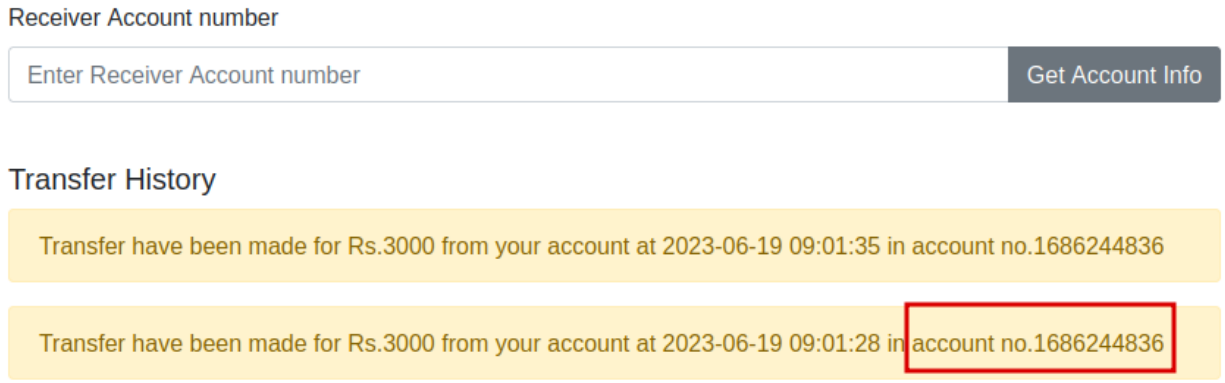
CWE-200

Proof of concept

When logged into the account of user, the account number of another user was clearly seen in transfer history.

Figure 2

Account number information of other user



CVSsv3.1

The CVS score for sensitive data exposure is 5.7

CVSS:3.1/AV:P/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:L

Business logic flaw

Description

The business logic flaw within the bank's system allowed unauthorized manipulation of transfer amounts. By modifying the minimum transfer amount and inputting negative values user was able to transfer funds to themselves from a different account.

Impact

It led to unauthorized financial transactions, allowing attackers to manipulate transfer amounts, potentially resulting in financial loss to both the bank and its customers. Such vulnerability could disrupt the integrity of financial records

Remediation

- Implement robust input validation to prevent manipulation of transfer amounts.
- Ensure that only authorized users can initiate fund transfers.
- Set transaction limits for transfers in both server and client side and implement alerts for exceeding limits.
- Implement two factor authentication for initiating transactions.
- Maintain detailed audit logs of funds transfer activities.

Reference

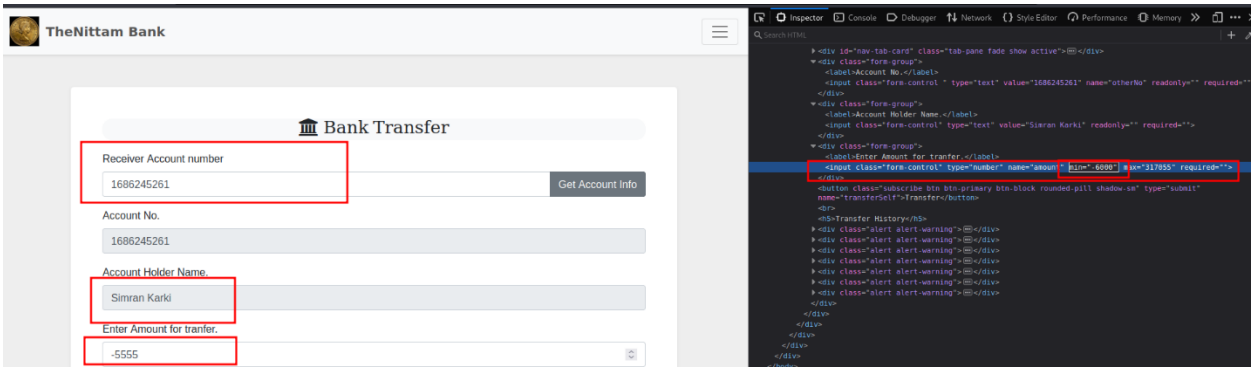
CWE-840

Proof of Concept

Initially, input the account number that was exposed in the transfer history. Then input a negative value and change the minimum transaction amount from the inspection filed and send the fund.

Figure 3

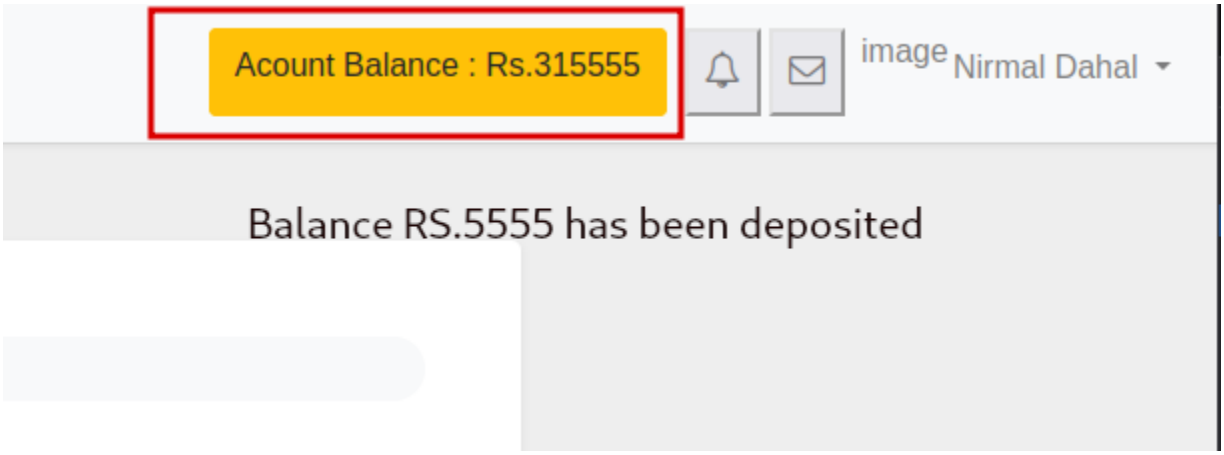
Changing the minimum transaction amount for funds



After the fund was transferred, observe that the money is now transferred to the same user from the other account.

Figure 4

Balance transferred to the same user



CVSsv3.1

The CVS score for business logic flaw is 7.6

CVSS:3.1/AV:P/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

CSRF

Description

Cross-Site Request Forgery (CSRF) is a type of security vulnerability that allowed to perform unauthorized transfers on behalf of the cashier without their consent. The form action and funds value in the HTML form used for fund deposits was manipulated which initiated unauthorized transfers from the cashier's account.

Impact

The cashier was tricked into unknowing transactions that was not intend to initiate which can result in unauthorized fund transfers or changes to account settings. Unauthorized transactions and data manipulation can lead to direct financial loss for both the bank and its customers.

Remediation

- Implement and validate CSRF tokens in every form submission to ensure that requests originate from the legitimate user.
- Use same-site cookie attributes to restrict the scope of cookies and prevent unauthorized requests from other websites.
- Require users to authenticate for sensitive actions or transactions.
- Developers should follow secure coding practices.

References

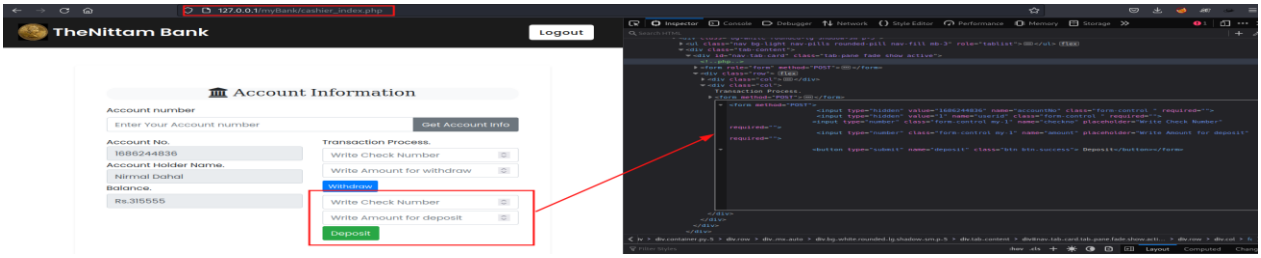
CWE-352

Proof of Concept

Firstly, log into cashier’s account via SQL injection. Then inspect the element of depositing funds and copy the HTML.

Figure 5

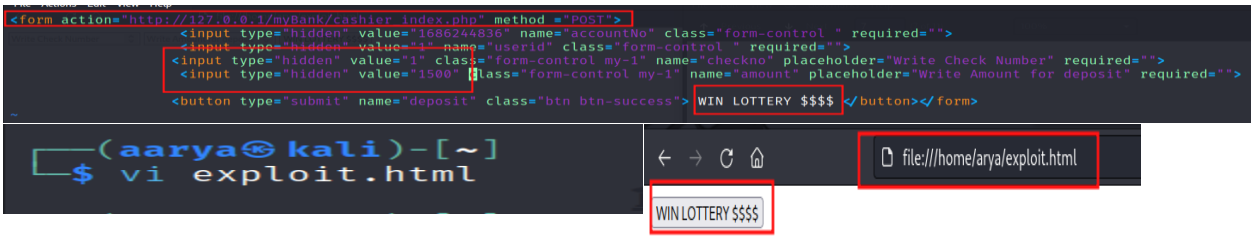
Inspecting and coping the deposit function of cashier



Then, Set the form's action attribute to the cashier's URL, so that the payload is initiated from the cashier. Put the id of the user and amount deposit value according to desire. Change the 'deposit' statement to 'WIN LOTTERY $$$$' and send the file to cashier.

Figure 6

Crafting the payload



When the cashier opens the file and clicks on the win lottery button, a transaction is initiated and the user receives the funds.

Figure 7

Funds transferred to the user

CVSsv3.1

The CVS score for CSRF is 7.

CVSS:3.1/AV:P/AC:H/PR:N/UI:R/S:C/C:H/I:H/A:H

XSS

Description

The application lacks proper input validation and output encoding. User-provided data is directly embedded into HTML responses without proper sanitization.

Impact

Malicious links can be crafted that, when clicked by users, execute arbitrary JavaScript code within the user's browser, potentially stealing session cookies or sensitive data.

Remediation

- Implement output encoding for user-generated content displayed in HTML responses.
- Validate and sanitize input data to prevent the injection of script tags.
- Implement validation and encoding for all user inputs that are stored and displayed.
- Regularly scan and sanitize stored data to prevent stored XSS attacks.

Reference

CWE-79

Proof of concept

Reflected XSS

Before exploiting reflected XSS check if the HTML tags are sanitized before being reflected in the web page. Here, <body><H1>Hello</H1></body> was entered in the input field which is not sanitized. Then, enter the payload <script>alert(document.cookie)</script> and steal the cashier’s cookie.

Figure 8

Checking for sanitization

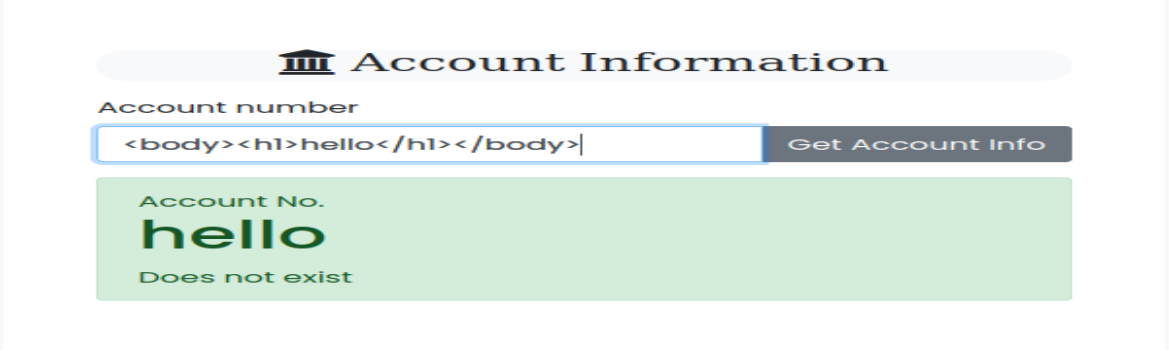
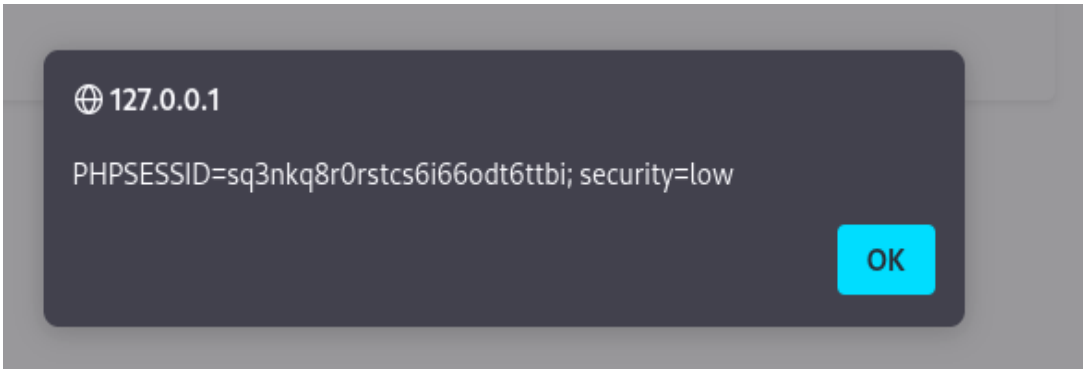


Figure 9

Reflected XSS



Stored XSS

Login from manager's account via SQL injection and send notice to one of the user with the payload <script>alert(document.cookie)</script>. When the user logins, the cookie is displayed in their account.

Figure 10

Stored XSS

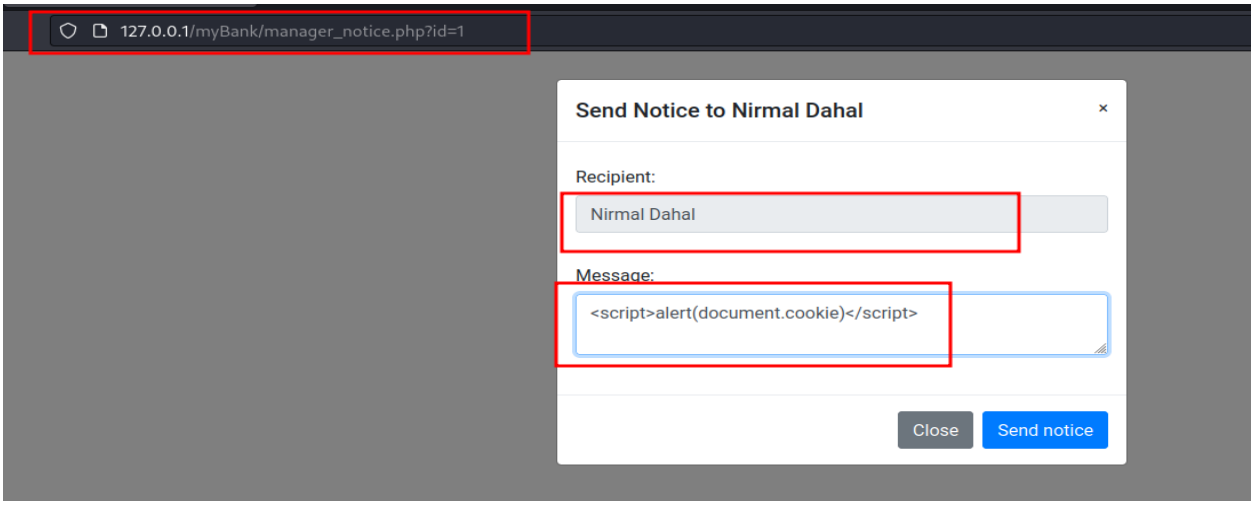
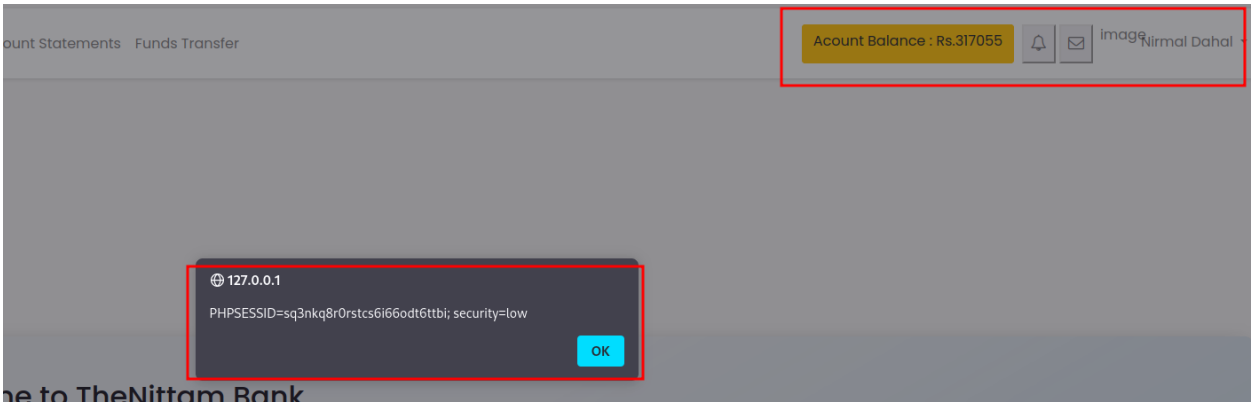


Figure 11

Pop-up alert in user's account



CVSsv3.1

The CVS score for XSS is 6.8

CVSS:3.1/AV:P/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Improper input validation

Description

The multiple instances of vulnerabilities, such as the SQL injection, XSS, and client-side input vulnerabilities, all point to improper input validation. The successful exploitation of these vulnerabilities indicates that the application does not adequately validate user inputs before processing them, leading to various security issues including unauthorized data access, code injection, and unintended content rendering.

Impact

This vulnerability can inject malicious code, compromising data integrity and potentially executing unauthorized actions which results in various security risks, including SQL injection, cross-site scripting (XSS), and unintended data manipulation. Financial information, personal details, and transaction records is exposed, leading to potential identity theft, fraud, and regulatory non-compliance.

Remedies

- Implement thorough input validation by checking user inputs against expected formats and disallowing any inputs that contain malicious content.
- Implement input sanitization and output encoding techniques to prevent security vulnerabilities like SQL injection and XSS.

Reference

CWE-20

CVSv3.1

CVS score for improper input validation is 6.8.
Vector string: CVSS:3.1/AV:L/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:L

Remediation

To address the identified vulnerabilities, a comprehensive strategy is needed. For SQL injection, robust input validation and parameterized queries must be implemented alongside regular software updates. Sensitive data exposure can be countered with data encryption, strict access controls, and diligent monitoring. Insecure funds transfer vulnerabilities demand input validation, strong authentication, and transaction limits. Improper input validation requires secure coding practices and continuous software patching. These measures collectively enhance the bank's security posture, safeguard sensitive data, and fortify against potential attacks.

Conclusion

In conclusion, the white box penetration testing conducted on The Nittam Bank's systems has unveiled critical vulnerabilities that could significantly compromise the security and integrity of its operations. The exposure of vulnerabilities such as SQL injection, sensitive data exposure, insecure funds transfer, and improper input validation underscores the urgency of implementing robust security measures. The potential impacts of unauthorized access, data breaches, and financial losses are substantial, warranting immediate attention. By adopting a proactive approach to remediation, the bank can fortify its defenses, safeguard customer trust, and align with industry best practices. Through vigilant monitoring, continuous assessment, and the implementation of comprehensive security measures, The Nittam Bank can strengthen its resilience against cyber threats and maintain a secure environment for its customers and financial operations.

References

CWE - CWE-20: Improper Input Validation (4.12). (n.d.). CWE - CWE-20: Improper Input Validation (4.12). https://cwe.mitre.org/data/definitions/20.html

CWE - CWE-200: Exposure of Sensitive Information to an Unauthorized Actor (4.12). (n.d.). CWE - CWE-200: Exposure of Sensitive Information to an Unauthorized Actor (4.12). https://cwe.mitre.org/data/definitions/200.html

CWE - CWE-352: Cross-Site Request Forgery (CSRF) (4.12). (n.d.). CWE - CWE-352: Cross-Site Request Forgery (CSRF) (4.12). https://cwe.mitre.org/data/definitions/352.html

CWE - CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') (4.12). (n.d.). CWE - CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') (4.12). https://cwe.mitre.org/data/definitions/79.html

CWE - CWE-840: Business Logic Errors (4.12). (n.d.). CWE - CWE-840: Business Logic Errors (4.12). https://cwe.mitre.org/data/definitions/840.html

CWE - CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') (4.12). (n.d.). CWE - CWE-89: Improper Neutralization of Special Elements Used in an SQL Command ('SQL Injection') (4.12). https://cwe.mitre.org/data/definitions/89.html