

# Website Application Security

## Assessment Testing

Zero Bank

<b>Report Release Date</b>	
<b>Type of Audit</b>	
<b>Type of Audit Report</b>	
<b>Period</b>	

# Document Control

Document Preparation	
Document Title	
Document ID	
Document Version	
Tested By	
Prepared by	
Reviewed by	
Approved by	
Released by	
Release Date	

Document Change History		
Version	Date	Remarks / Reason of change

Document Distribution List			
Name	Organization	Designation	Email Id

## CONTENTS

<u>1</u>	<u>Introduction</u>	5
1.1	<u>Details of the Auditing Team</u>	5
1.2	<u>Audit Activities</u>	5
1.3	<u>Audit Timelines</u>	6
1.4	<u>Audit Methodology and Standard Referred for Audit</u>	6
1.5	<u>Tools/ Software used</u>	6
1.6	<u>Executive Summary</u>	7
1.7	<u>Tabular Summary</u>	23
1.8	<u>Graphical Summary</u>	23
1.9	<u>Severity Rating</u>	24
1.10	<u>Vulnerability Status</u>	24
<u>2</u>	<u>Detailed Observations</u>	25
2.1	<u>Insecure Http Usage</u>	<b>Error! Bookmark not defined.</b>
2.2	<u>Content Security Policy Not Implemented</u>	<b>Error! Bookmark not defined.</b>
2.3	<u>Server Version Disclosed</u>	<b>Error! Bookmark not defined.</b>
2.4	<u>Version Disclosure Asp.net</u>	<b>Error! Bookmark not defined.</b>
<u>3</u>	<u>Appendix</u>	<b>Error! Bookmark not defined.</b>
3.1	<u>Appendix A: Types of Assessments</u>	<b>Error! Bookmark not defined.</b>
3.2	<u>Appendix B: Severity Rating Details</u>	<b>Error! Bookmark not defined.</b>
3.3	<u>Appendix C: Vulnerability Management Methodology</u>	<b>Error! Bookmark not defined.</b>

## **Copyright**

---

The copyright for this document belongs to Info eShield Cyber Solutions LLP. This document is provided in confidence for its intended purpose. It must not be reproduced in whole or in part, or used for tendering or manufacturing purposes, without prior written agreement or consent from Info eShield Cyber Solutions LLP. Any reproduction must include this copyright notice. No information regarding the contents or subject matter of this document, or any part thereof, may be shared orally or in writing, or communicated in any manner to any third party—whether an individual, firm, company, or employee—without prior written consent from Info eShield Cyber Solutions LLP.

© Info eShield Cyber Solutions LLP. 2025.

## **Disclaimer**

---

By accessing and using this report, you agree to the following terms and conditions and all applicable laws, without limitation or qualification. Unless otherwise stated, the contents of this document, including text, images, and their arrangement, are the property of Info eShield Cyber Solutions LLP (Info eShield). Nothing in this document should be interpreted as granting, by implication, estoppel, or otherwise, any license or right to any copyright, patent, trademark, or other proprietary interests of Info eShield or any third party. This document and its contents—including graphic images and documentation—may not be copied, reproduced, republished, uploaded, posted, transmitted, or distributed in any manner without the prior written consent of Info eShield. Any use you make of the information provided is at your own risk.

## Introduction

---

<Add Report Introduction dynamically >

### Details of the Auditing Team

---

SN	Name	Designation	Email ID	Professional Qualifications/ Certifications	Whether the resource has been listed in the Snapshot information published on CERT-In's website (Yes/No)

### Audit Activities

---

SN	Activity Name	Start Date	End Date

## Audit Timelines

---

SN	Activity Name	Start Date	End Date

## Audit Methodology and Standard Referred for Audit

---

Our testing methodology adapts from the following security frameworks and vulnerability categories:

1. [Common Weakness Enumeration](#) (CWE)
2. [National Institute of Standards and Technology](#) (NIST)
3. [OWASP Top 10](#) (OWASP)
4. [OWASP Mobile Top 10](#) (OWASP)

Our standard approach of testing is mentioned in the [Appendix C](#) of this report document.

## Tools/ Software used

---

SN	Tools/ Software used	Version of the tool /Software used	Open Source/Licensed
1.	Kali linux	Kali-2025	Open
2.	Burp Suite	2025.7.4	Open
3.	N-Map	7.98	Open
4.	Python	3.13.7	Open
5.	VM-Ware	17 pro	Open
6.	CMD	10.0.26100.4946	Open
7.			
8.			

## Executive Summary

---

The security assessment identified a total of **32 vulnerabilities** categorized by severity as follows:

**Critical** Severity: 4 Vulnerability

**High** Severity: 18 Vulnerability

**Medium** Severity: 6 Vulnerability

**Low** Severity: 4 Vulnerabilities

**Informational** Severity: 0 Vulnerabilities

S N	URL	Name of vulnerability	Severity	Status	Recommendation
1.	http://zero.webappsecurity.com/login.html?login http://zero.webappsecurity.com/search.html?searchTerm=	Copy Pate Allow	Low	Open	Disable copy-paste functionality in password input fields using HTML or JavaScript. Additionally, implement security measures such as CAPTCHA, account lockout after failed attempts, and Multi-Factor Authentication.
2.	http://zero.webappsecurity.com/bank/pay-bills.html	Reflected XSS	HIGH	OPEN	Implement proper input validation and output encoding for all user-supplied data. Use context-aware encoding

					before rendering inputs in the browser. Apply security headers like Content Security Policy and HttpOnly/Secure flags on cookies. Use trusted frameworks that auto-escape output wherever possible.
3.	http://zero.webappsecurity.com/bank/account-activity.html?accountId=	<b>Insecure Direct Object Reference (IDOR)</b>	<b>HIGH</b>	<b>OPEN</b>	Implement proper access control checks on the server side to ensure that users can only access their own resources. Use indirect references (like UUIDs or random tokens) instead of sequential identifiers. Apply robust authentication and authorization mechanisms for all sensitive endpoints.

					Regularly test for IDOR as part of secure code reviews and penetration testing
4.	<a href="http://zero.webappsecurity.com/docs/">http://zero.webappsecurity.com/docs/</a>	<b>Directory Listing</b>	<b>LOW</b>	<b>OPEN</b>	Disable directory listing on the server and restrict public access to internal documentation or configuration files. Upgrade to a supported and patched version of Apache Tomcat. Implement proper server hardening practices, including limiting exposure of unnecessary files and directories.
5.	<a href="http://zero.webappsecurity.com/login.html?login_error=true">http://zero.webappsecurity.com/login.html?login_error=true</a>	<b>Weak Password</b>	<b>HIGH</b>	<b>OPEN</b>	Enforce strong password policy: Implement multi-factor authentication for additional security. Secure the website with HTTPS to protect credentials in transit. Introduce

					account lockout / CAPTCHA mechanisms after multiple failed login attempts.
6.	http://zero.webappsecurity.com/manager/html	Sensitive Data Exposure	HIGH	OPEN	Disable detailed error messages for unauthorized users; show generic error pages instead. Restrict access to Tomcat Manager to trusted IP addresses only. Keep Tomcat and related components patched and updated.
7.	http://zero.webappsecurity.com/bank/pay-bills.html	CSRF	HIGH	OPEN	Implement CSRF tokens. Use same site cookies to limit cookie sending in cross-origin requests. Require re-authentication (password, OTP, or MFA) for critical

					actions (like fund transfers). Enforce secure cookie attributes
8.	<a href="http://zero.webappsecurity.com/bank/account-summary.html">http://zero.webappsecurity.com/bank/account-summary.html</a>	<b>Plain Text Credentials</b>	<b>HIGH</b>	<b>OPEN</b>	Enforce the use of HTTPS (TLS/SSL) for all communication to encrypt sensitive information in transit. Implement strong authentication mechanisms such as hashing for password storage and use secure session management practices.
9.	<a href="http://zero.webappsecurity.com/sendFeedback.html">http://zero.webappsecurity.com/sendFeedback.html</a>	<b>Denial of Service</b>	<b>MEDIUM</b>	<b>OPEN</b>	Implement rate limiting and connection throttling to mitigate excessive requests. Deploy Web Application Firewalls (WAF) and Intrusion Detection/Prevention Systems (IDS/IPS) to filter malicious traffic.

					Optimize server capacity and use load balancing and redundancy to handle traffic spikes
10.	http://zero.webappsecurity.com/errors/errors.log	<b>Information Disclosure via Internal Files</b>	MEDIUM	OPEN	Restrict access to internal and sensitive files via proper authentication and authorization mechanisms. Implement proper file permissions, ensuring that only necessary processes or users can access sensitive files. Disable directory listing on all web servers.
11.	http://zero.webappsecurity.com/login.html	<b>SSL Certificate Missing</b>	HIGH	OPEN	Obtain and install a valid SSL/TLS certificate from a reputable Certificate Authority (CA). Configure the server to redirect all HTTP traffic to HTTPS, ensuring

					secure communication.
12.	http://zero.webappsecurity.com/admin/users.html	Insecure Storage	HIGH	OPEN	Store sensitive data securely using strong encryption algorithms and hashing functions for passwords. Avoid storing sensitive data on the client side; if necessary, ensure it is encrypted and protected with proper security flags. Implement strict access controls to prevent unauthorized access to stored data.
13.	http://zero.webappsecurity.com/login.html	Session Hijacking	HIGH	OPEN	Use secure, randomly generated session IDs that are long and unpredictable. Ensure session tokens are transmitted over HTTPS only. Implement session timeout and

					automatic logout after inactivity.
14.	http://zero.webappsecurity.com/bank/transfer-funds.html	<b>Browser Cache Weakness</b>	<b>HIGH</b>	<b>OPEN</b>	Configure HTTP headers to prevent caching of sensitive pages. Configure HTTP headers to prevent caching of sensitive pages. Implement session expiration and automatic logout mechanisms
15.	http://zero.webappsecurity.com/admin/users.html	<b>Authentication Bypass</b>	<b>CRITICAL</b>	<b>OPEN</b>	Enforce strong authentication mechanisms, including complex passwords and MFA. Validate all inputs and parameters to prevent manipulation. Ensure server-side authentication checks are mandatory for every request to

					restricted resources.
16.	http://zero.webappsecurity.com/forgot-password.html	Insecure Password Reset	CRITICAL	OPEN	Implement secure, time-limited, and random password reset tokens. Enforce strong verification for password reset requests. Use HTTPS to transmit all password reset links and tokens. Implement rate limiting and monitoring to prevent abuse of password reset functionality.
17.	http://zero.webappsecurity.com/bank/pay-bills.html	HTTP Strict Transport Security Missing	HIGH	OPEN	Enable and enforce HSTS by configuring the server. Ensure HTTPS is correctly configured with a valid SSL/TLS certificate. Register the domain for HSTS Preload List.

					Redirect all HTTP traffic to HTTPS
18.	http://zero.webappsecurity.com/bank/pay-bills.html	CORS Misconfiguration	HIGH	OPEN	<p>Review and restrict the cross-domain policy file to only allow trusted domains. Avoid using wildcards (*) in domain permissions.</p> <p>Restrict access to only the specific resources that are required by external domains</p>
19.	http://zero.webappsecurity.com/admin/	Privilege Escalation	CRITICAL	OPEN	<p>Implement strict role-based access control and enforce authorization checks at the server-side for every sensitive function. Validate user roles and permissions before executing privileged operations. Avoid relying solely on client-side controls</p>

					for access restrictions.
20.	http://zero.webappsecurity.com/admin/test.txt	<b>Server Version Disclosure</b>	<b>MEDIUM</b>	<b>OPEN</b>	Suppress or modify the Server and other version-related headers. Configure the web server to hide or obfuscate version information. Use custom error pages instead of default error messages to prevent information leakage
21.	http://zero.webappsecurity.com/admin/test.txt	<b>Weak Lockout mechanism</b>	<b>HIGH</b>	<b>OPEN</b>	Implement a robust lockout policy. Use progressive delays (rate-limiting) instead of simple lockouts to prevent automated brute force. Encourage strong password policies and multi-factor authentication (MFA).

22.	<a href="http://zero.webappsecurity.com/login.html?login_error=true">http://zero.webappsecurity.com/login.html?login_error=true</a>	<b>Captcha Missing</b>	<b>MEDIUM</b>	<b>OPEN</b>	Implement CAPTCHA on login pages. Combine with rate limiting and account lockout mechanisms for stronger protection. Regularly update CAPTCHA libraries to the latest versions to avoid bypass techniques.
23.	<a href="http://zero.webappsecurity.com/forgotten-password-send.html">http://zero.webappsecurity.com/forgotten-password-send.html</a>	<b>Server Side Validation Is Absent</b>	<b>HIGH</b>	<b>OPEN</b>	Implement server-side validation for all user inputs, even if client-side validation exists. Use prepared statements or parameterized queries for database operations. Apply proper encoding and escaping for output to prevent XSS.
24.	<a href="http://zero.webappsecurity.com/">http://zero.webappsecurity.com/</a>	<b>Clickjacking</b>	<b>HIGH</b>	<b>OPEN</b>	During testing, it was found that the

					application does not implement sufficient protection against Clickjacking attacks. The tested web pages can be embedded within an <iframe> on a malicious domain, allowing attackers to trick users into unintentionally performing actions
25.	http://zero.webappsecurity.com/	<b>WAF Missing</b>	<b>MEDIUM</b>	<b>OPEN</b>	The application is deployed without a Web Application Firewall (WAF). As a result, there is no additional security layer to detect and mitigate common web-based attacks such as SQL Injection, Cross-Site Scripting (XSS), Local File Inclusion (LFI), Remote File

					Inclusion (RFI), and automated bot attacks.
26.	<a href="http://zero.webappsecurity.com/">http://zero.webappsecurity.com/</a>	<b>Security Headers Missing</b>	<b>MEDIUM</b>	<b>OPEN</b>	During testing it was found that security headers in website are missing.
27.	<a href="http://zero.webappsecurity.com/bank/pay-bills.html">http://zero.webappsecurity.com/bank/pay-bills.html</a> <a href="http://zero.webappsecurity.com/bank/pay-bills-saved-payee.html">http://zero.webappsecurity.com/bank/pay-bills-saved-payee.html</a>	<b>Stored XSS</b>	<b>HIGH</b>	<b>OPEN</b>	During testing it was observed that stored XSS occur on the pay saved payee field. User supplied input is saved on the server.
28.	<a href="http://zero.webappsecurity.com/bank/pay-bills-saved-payee">http://zero.webappsecurity.com/bank/pay-bills-saved-payee</a>	<b>HTTP Parameter manipulation</b>	<b>HIGH</b>	<b>OPEN</b>	During test it was observed that amount and account id are manipulated in burp suite request. Due to which response is positive

					and successfully change the fields.
29.	<a href="http://zero.webappsecurity.com/index.html">http://zero.webappsecurity.com/index.html</a>	<b>Weak Lockout Suspension</b>	<b>CRITICAL</b>	<b>OPEN</b>	During testing it was observed that when user is login in website, server stores username and password in cookies attribute
30.	<a href="http://zero.webappsecurity.com/login.html?login_error=true">http://zero.webappsecurity.com/login.html?login_error=true</a>	<b>Insecure Remember Password Implementation</b>	<b>HIGH</b>	<b>OPEN</b>	During testing it was observed that after failing multiple login attempts account is not locked or suspended. This indicates that the account suspension mechanism is either missing or improperly implemented.
31.	<a href="http://zero.webappsecurity.com/bank/pay-bills.html">http://zero.webappsecurity.com/bank/pay-bills.html</a>	<b>Business Logic Break</b>	<b>LOW</b>	<b>OPEN</b>	During testing it was observed that website allows anything in the date

					field. By which, business logic is break.
32.	view-source:http://zero.webappsecurity.com/	<b>Outdated Library</b>	<b>LOW</b>	<b>OPEN</b>	During testing it was observed that in source code there is jquery library whose version is 1.8.2 which is old and is vulnerable to various attacks.

## Tabular Summary

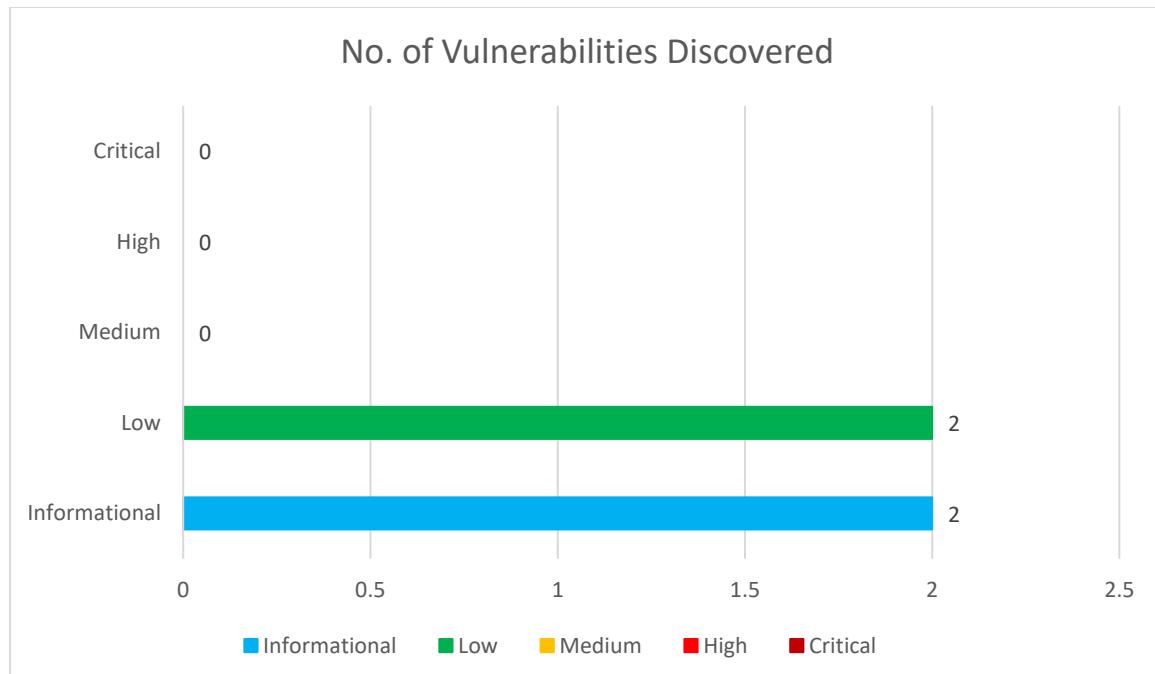
---

The following table summarizes the overall security assessment:

Category	Description				
Security Assessment Summary					
Number of Application	1				
Number of Vulnerabilities found	32				
Vulnerability Breakup based on Severity	4	18	6	4	0
	Critical	High	Medium	Low	Info

## Graphical Summary

---



## Severity Rating

---

Info eShield follows the Common Vulnerability Scoring System (CVSS) scoring system to rate vulnerabilities.

The CVSS assessment measures three areas of concern:

- **Base Metrics** for qualities intrinsic to a vulnerability
- **Temporal Metrics** for characteristics that evolve over the lifetime of vulnerability
- **Environmental Metrics** for vulnerabilities that depend on an implementation or environment

For more information on what the Base, Temporal, and Environment Metrics in the CVSS scoring system are, please visit <https://www.first.org/cvss/specification-document>

Unless otherwise stated, the findings in the report are scored on the base-metric rating of the vulnerability. NII may consider Environmental and Temporal Metrics depending on information provided at project initiation and if this is a mandatory client reporting requirement.

Based on the severity of the vulnerability, they are assigned below ratings:

**CVSS Qualitative severity rating scale (For detailed definitions and details of severity ratings, please refer to [appendix B in this report](#))**

CVSS Severity Rating	CVSS Score
Critical	9.0 - 10.0
High	7.0 - 8.9
Medium	4.0 - 6.9
Low	0.1 - 3.9
None	0.0

## Vulnerability Status

---

The definition of status of various vulnerabilities after revalidation is given below:

Status	Description
OPEN	The vulnerability has not been fixed.
OPEN (PARTIAL)	The vulnerability was not fixed completely. Either some aspects were left unpatched, or the issue was fixed to address only some components of the vulnerability but possibly resulting in a lower severity rating.
CLOSED (UNVERIFIED)	The target could no longer be accessed or the vulnerability could not be verified due to the configuration change. Thus, it is considered closed.
CLOSED (EXCEPTION)	The vulnerability has not been fixed. However, the management has accepted the risks or due to business requirements the issue cannot be fixed. Thus, the issue has been considered closed.
CLOSED	The vulnerability has been verified to be fixed.

# Detailed Observations

---

## 2.1 Copy-Paste Allow in Login Fields

### Severity

---

LOW

### Vulnerability Status

---

OPEN

### Affected URL

---

<http://zero.webappsecurity.com/login.html?login>

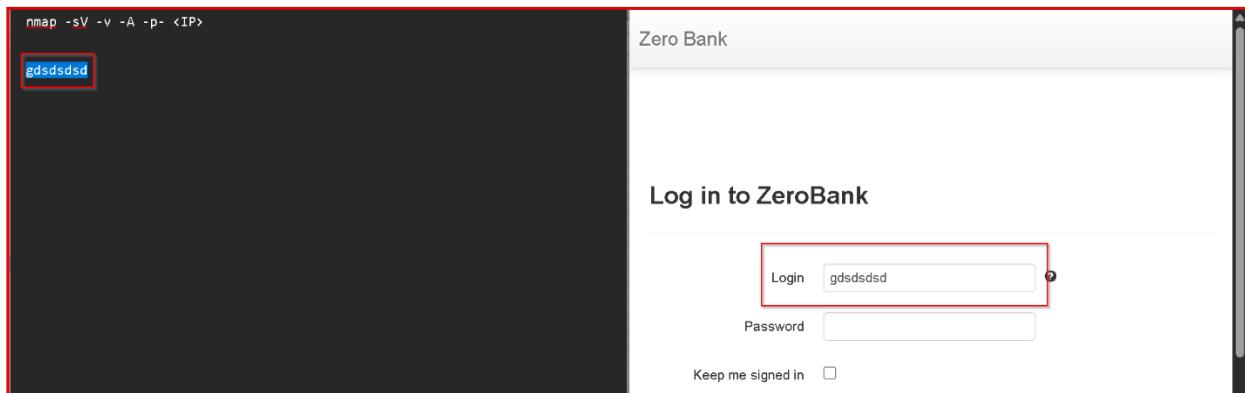
### Detailed Observation

---

During testing, it was observed that the login input fields (username/password) allow unrestricted copy-paste operations. This could allow attackers to paste stolen credentials, automate brute force attempts, or facilitate credential stuffing attacks.

### Testing PoC

---



### Impact

---

This increases the risk of automated attacks such as bot-based brute force and credential stuffing. Stolen credentials can be reused directly, reducing the overall security posture of the authentication mechanism.

### Recommendation

---

Disable copy-paste functionality in password input fields using HTML or JavaScript. Additionally, implement security measures such as CAPTCHA, account lockout after failed attempts, and Multi-Factor Authentication (MFA).

### References

---

[https://cheatsheetseries.owasp.org/cheatsheets/Authentication Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html)

## 2.2 Reflected XSS

### Severity

HIGH

### Vulnerability Status

OPEN

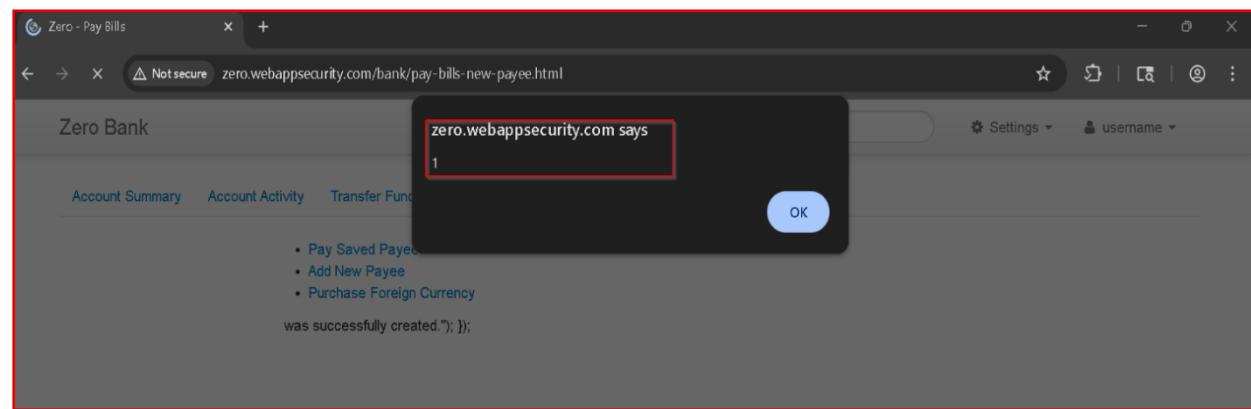
### Affected URL

<http://zero.webappsecurity.com/bank/pay-bills.html>

### Detailed Observation

It was identified that the application does not properly sanitize user-supplied input. When crafted payloads were injected into the request, the malicious script was reflected back and executed in the browser, as evidenced by the alert pop-up shown during testing.

### Testing PoC



### Impact

Successful exploitation allows attackers to execute arbitrary JavaScript in the victim's browser. This may lead to session hijacking, redirection to malicious sites, defacement, theft of sensitive information such as cookies, or performing unauthorized actions on behalf of the user.

### Recommendation

Implement proper input validation and output encoding for all user-supplied data. Use context-aware encoding (HTML, JavaScript, URL) before rendering inputs in the browser. Apply security headers like Content Security Policy (CSP) and HttpOnly/Secure flags on cookies. Use trusted frameworks that auto-escape output wherever possible.

### References

[https://cheatsheetseries.owasp.org/cheatsheets/Cross\\_Site\\_Scripting\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html)

## 2.3 Insecure Direct Object Reference (IDOR)

### Severity

HIGH

### Vulnerability Status

OPEN

### Affected URL

http://zero.webappsecurity.com/bank/account-activity.html?accountId=

### Detailed Observation

It was observed that the application exposes sensitive account details by using sequential or predictable account IDs in the request parameter (accountId). By changing the accountId value in the URL, an attacker is able to access transaction details of other users without authorization, as demonstrated in the provided evidence.

### Testing PoC

The screenshot shows a web browser window with the URL `zero.webappsecurity.com/bank/account-activity.html?accountId=1` in the address bar. The page title is "Zero Bank". The navigation menu includes "Account Summary", "Account Activity" (which is selected), "Transfer Funds", "Pay Bills", "My Money Map", and "Online Statements". Below the menu, there are buttons for "Show Transactions" and "Find Transactions". The main content area is titled "Show Transactions" with the sub-instruction "Choose an account to view.". A dropdown menu for "Account" is open, showing "Savings". A table displays transaction history:

Date	Description	Deposit	Withdrawal
2012-09-06	ONLINE TRANSFER REF #JKKSDRQG6L	984.3	
2012-09-05	OFFICE SUPPLY		50
2012-09-01	ONLINE TRANSFER REF #JKKSDRQG6L	1000	

The screenshot shows a web browser window for 'Zero Bank'. The URL bar displays 'zero.webappsecurity.com/bank/account-activity.htm?accountId=3'. The page has a navigation bar with tabs: 'Account Summary' (selected), 'Account Activity', 'Transfer Funds', 'Pay Bills', 'My Money Map', and 'Online Statements'. Below the tabs are buttons for 'Show Transactions' and 'Find Transactions'. A section titled 'Show Transactions' asks 'Choose an account to view.' with dropdowns for 'Account' (set to 'Savings') and 'Type' (set to 'All'). The main content area shows a table of transactions:

Date	Description	Deposit	Withdrawal
2012-09-06	ONLINE TRANSFER REF #IGREKLVC0D	636.4	
2012-09-05	ONLINE TRANSFER REF #IGREKLVC0D	55.9	
2012-09-01	PAYCHECK		1173.1

## Impact

This vulnerability can lead to unauthorized access to sensitive financial information, including account balance, deposits, and withdrawals. An attacker can exploit this flaw to view or manipulate other users' transaction data, leading to privacy violations, financial fraud, and loss of customer trust.

## Recommendation

Implement proper access control checks on the server side to ensure that users can only access their own resources. Use indirect references (like UUIDs or random tokens) instead of sequential identifiers. Apply robust authentication and authorization mechanisms for all sensitive endpoints. Regularly test for IDOR as part of secure code reviews and penetration testing.

## References

[https://owasp.org/Top10/A01\\_2021-Broken\\_Access\\_Control/](https://owasp.org/Top10/A01_2021-Broken_Access_Control/)

## 2.4 Directory Listing

### Severity

LOW

### Vulnerability Status

OPEN

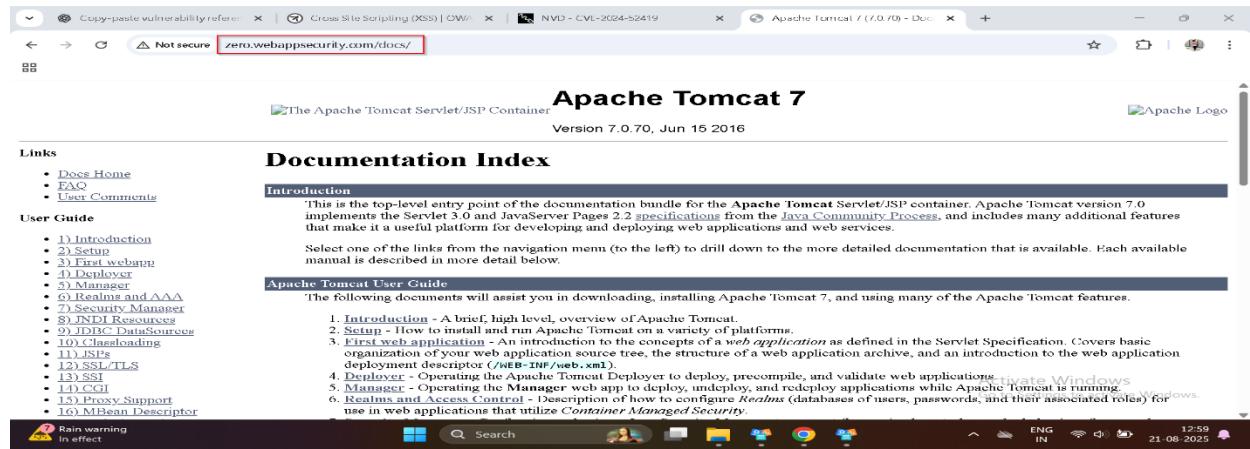
### Affected URL

<http://zero.webappsecurity.com/docs/>

### Detailed Observation

It was observed that the application allows direct access to the /docs/ directory on the web server, which exposes Apache Tomcat documentation and configuration details. This reveals sensitive information about the server setup, technologies in use, and version details (Apache Tomcat 7.0.70), which could assist attackers in identifying potential exploits.

### Testing PoC



The screenshot shows a browser window with the following details:

- Address bar: zero.webappsecurity.com/docs/
- Title bar: Apache Tomcat / (7.0.70) - Docs
- Content:
  - Apache Tomcat 7** (Version 7.0.70, Jun 15 2016)
  - Documentation Index**
  - Introduction**: Describes the top-level entry point for the documentation bundle, mentioning Servlet 3.0 and JavaServer Pages 2.2 specifications.
  - User Guide**: A list of 16 items including Introduction, Setup, First webapp, Deployer, Manager, Realm, Security Manager, JNDI Resources, JDBC Datasources, Classloading, JSPs, SSL/TLS, SSI, CGI, Proxy Support, and MBean Descriptor.
  - Apache Tomcat User Guide**: A list of 6 items including Introduction, Setup, Web Application, Deployer, Manager, and Realms and Access Control.

### Impact

An attacker can leverage exposed documentation and version details to perform reconnaissance, identify known vulnerabilities, and craft targeted attacks. Since Apache Tomcat 7 is outdated and no longer supported, this also increases the risk of exploitation through unpatched security flaws.

### Recommendation

Disable directory listing on the server and restrict public access to internal documentation or configuration files. Upgrade to a supported and patched version of Apache Tomcat. Implement proper server hardening practices, including limiting exposure of unnecessary files and directories.

### References

[https://owasp.org/www-community/attacks/Information\\_disclosure](https://owasp.org/www-community/attacks/Information_disclosure)

## 2.5 Weak Password

### Severity

HIGH

### Vulnerability Status

OPEN

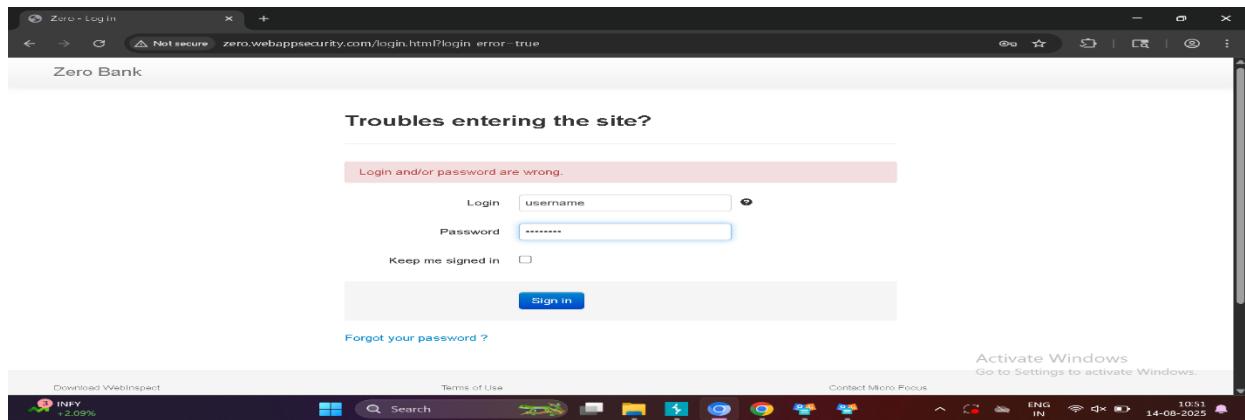
### Affected URL

[http://zero.webappsecurity.com/login.html?login\\_error=true](http://zero.webappsecurity.com/login.html?login_error=true)

### Detailed Observation

The login page does not enforce strong password policies. Users may set weak passwords such as short, simple, or dictionary-based passwords

### Testing PoC



### Impact

Attackers can easily guess or brute-force weak passwords to gain unauthorized access. The lack of HTTPS further increases the risk of man-in-the-middle (MITM) attacks.

### Recommendation

Enforce strong password policy: Implement multi-factor authentication (MFA) for additional security. Secure the website with HTTPS (TLS/SSL) to protect credentials in transit. Introduce account lockout / CAPTCHA mechanisms after multiple failed login attempts.

### References

[https://cheatsheetseries.owasp.org/cheatsheets/Authentication\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html)

## 2.6 Sensitive Data Exposure

### Severity

HIGH

### Vulnerability Status

OPEN

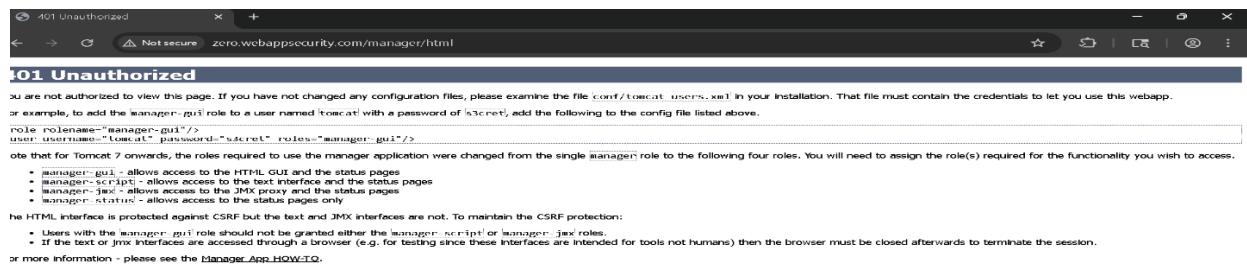
### Affected URL

<http://zero.webappsecurity.com/manager/html>

### Detailed Observation

The Tomcat Manager error page is revealing internal information about the server configuration. This includes the path to the tomcat-users.xml file, sample username and password, and security configuration details. Such information helps attackers craft targeted brute-force or privilege escalation attacks.

### Testing PoC



### Impact

Attackers gain knowledge of default credentials often used in misconfigured systems. Can lead to unauthorized access or complete server takeover if weak/default credentials are used. File path disclosure helps in local file inclusion (LFI)\ or directory traversal attacks.

### Recommendation

Disable detailed error messages for unauthorized users; show generic error pages instead. Restrict access to Tomcat Manager (/manager/html) to trusted IP addresses only. Keep Tomcat and related components patched and updated.

### References

[https://owasp.org/Top10/A02\\_2021-Cryptographic\\_Failures/](https://owasp.org/Top10/A02_2021-Cryptographic_Failures/)

## 2.7 Cross Site Request Forgery (CSRF)

### Severity

HIGH

### Vulnerability Status

OPEN

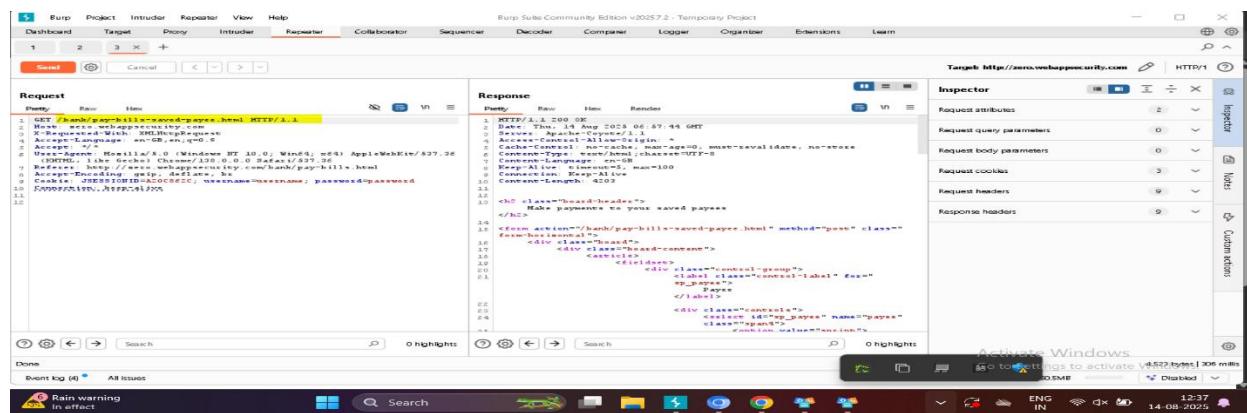
### Affected URL

http://zero.webappsecurity.com/bank/pay-bills.html

### Detailed Observation

Authentication is handled using a session cookie (JSESSIONID=...) along with credentials (username=username; password=password) stored in cookies, which makes it easier for attackers to forge requests. Since the application accepts GET or POST requests without additional verification, an attacker can trick a logged-in user into executing unwanted actions like adding a payee or transferring funds by embedding malicious links or forms on an external site..

### Testing PoC



### Impact

An attacker can perform unauthorized transactions (e.g., transferring money, modifying account details) by tricking an authenticated user into clicking a crafted link or visiting a malicious page. This compromises integrity of user accounts and can lead to financial fraud.

### Recommendation

Implement CSRF tokens. Use same site cookies to limit cookie sending in cross-origin requests. Require re-authentication (password, OTP, or MFA) for critical actions (like fund transfers). Enforce secure cookie attributes.

### References

[https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site\\_Request\\_Forgery\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html)

## 2.8 Plain Text Credentials

### Severity

HIGH

### Vulnerability Status

OPEN

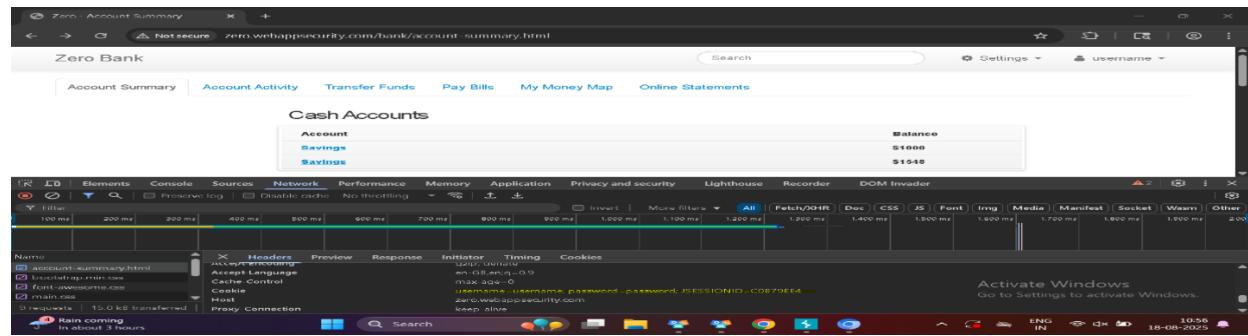
### Affected URL

<http://zero.webappsecurity.com/bank/account-summary.html>

### Detailed Observation

While analyzing the network traffic of the application, it was observed that the username and password are being transmitted in plain text within the HTTP request header.

### Testing PoC



### Impact

The transmission of credentials in plain text can lead to account compromise if an attacker intercepts the traffic over insecure channels. Once obtained, the attacker can log in to the application and gain unauthorized access to sensitive financial data or perform malicious transactions.

### Recommendation

Enforce the use of HTTPS (TLS/SSL) for all communication to encrypt sensitive information in transit. Implement strong authentication mechanisms such as hashing for password storage and use secure session management practices.

### References

[https://cheatsheetseries.owasp.org/cheatsheets/Transport\\_Layer\\_Protection\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.html)

## 2.9 Denial of Service Attack

### Severity

MEDIUM

### Vulnerability Status

OPEN

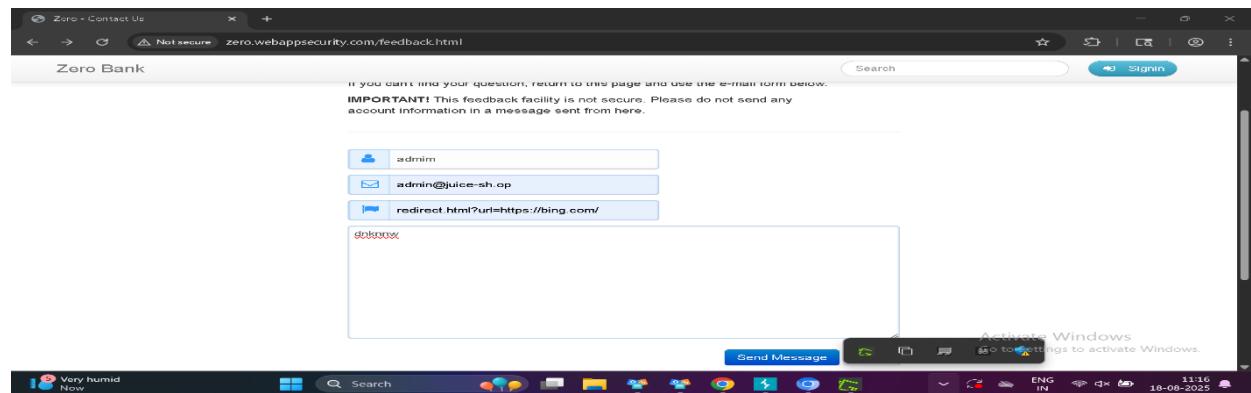
### Affected URL

<http://zero.webappsecurity.com/sendFeedback.html>

### Detailed Observation

During testing, it was observed that the feedback form can be overwhelmed by a high volume of automated submissions in a short period of time. The form does not implement rate limiting, CAPTCHA, or request throttling, allowing attackers to repeatedly submit requests without restriction.

### Testing PoC



### Impact

Legitimate users may be unable to access the service, causing downtime and operational disruption. Potential financial loss due to service unavailability.

### Recommendation

Implement rate limiting and connection throttling to mitigate excessive requests. Deploy Web Application Firewalls (WAF) and Intrusion Detection/Prevention Systems (IDS/IPS) to filter malicious traffic. Optimize server capacity and use load balancing and redundancy to handle traffic spikes.

### References

[https://owasp.org/www-community/attacks/Denial\\_of\\_Service](https://owasp.org/www-community/attacks/Denial_of_Service)

## 2.10 Information Disclosure via Internal Files

### Severity

MEDIUM

### Vulnerability Status

OPEN

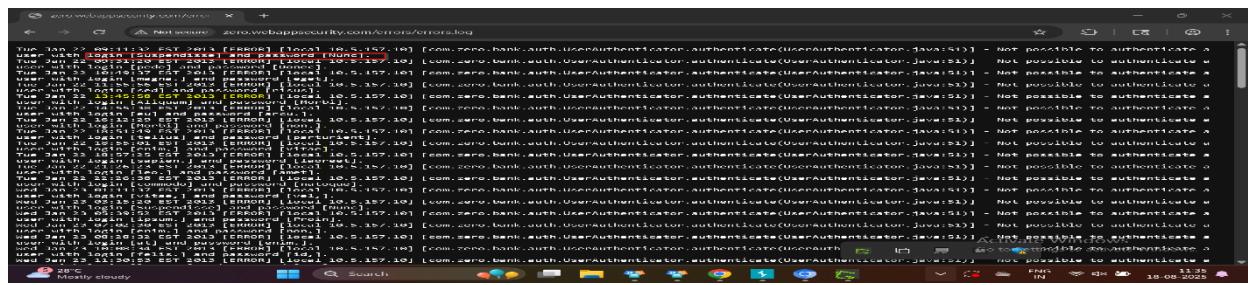
### Affected URL

http://zero.webappsecurity.com/errors/errors.log

### Detailed Observation

During testing, it was observed that the application exposes internal files ex; log file through predictable URLs. These files contain sensitive information such as database credentials.

### Testing PoC



### Impact

Potential compromise of the application, database, or other connected systems. Exposure of sensitive data such as passwords, API keys, or internal configuration, which can lead to unauthorized access.

### Recommendation

Restrict access to internal and sensitive files via proper authentication and authorization mechanisms. Implement proper file permissions, ensuring that only necessary processes or users can access sensitive files. Disable directory listing on all web servers.

### References

[https://owasp.org/www-community/vulnerabilities/Information\\_Leak\\_through\\_Comments\\_and\\_Logs](https://owasp.org/www-community/vulnerabilities/Information_Leak_through_Comments_and_Logs)

## 2.11 SSL Certificate Missing

### Severity

HIGH

### Vulnerability Status

OPEN

### Affected URL

<http://zero.webappsecurity.com/login.html>

### Detailed Observation

The Zero Bank website does not implement SSL/TLS encryption, making it accessible via HTTP instead of HTTPS.

### Testing PoC



### Impact

Sensitive user information transmitted over HTTP is vulnerable to interception by malicious actors. Without SSL/TLS, attackers can intercept and potentially alter communications between users and the server.

### Recommendation

Obtain and install a valid SSL/TLS certificate from a reputable Certificate Authority (CA). Configure the server to redirect all HTTP traffic to HTTPS, ensuring secure communication.

### References

<https://www.cloudflare.com/learning/ssl/common-errors/>

## 2.12 Insecure Storage

### Severity

HIGH

### Vulnerability Status

OPEN

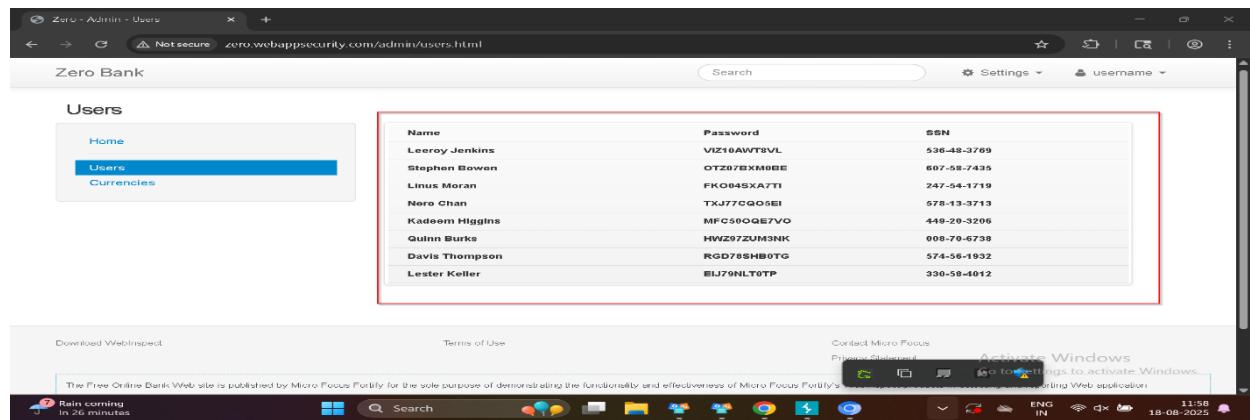
### Affected URL

<http://zero.webappsecurity.com/admin/users.html>

### Detailed Observation

During testing, it was observed that the application stores sensitive data such as passwords, authentication tokens, personal information, or financial data in an insecure manner

### Testing PoC



Name	Password	SSN
Leeroy Jenkins	VIZ10AWT8VL	536-48-3769
Stephen Bowen	DTZ07BXM0E	607-58-7435
Linus Moran	FK0845XA7T	247-54-1719
Nero Chan	TXJ77CG05E	578-13-3713
Kadeem Higgins	MFC59OQE7VO	448-20-3206
Quinn Burks	HWZ97ZUM3NK	008-70-6738
Davis Thompson	RGD78SHB0TG	574-56-1932
Lester Keller	EIJ79NLT0TP	330-58-4012

### Impact

Exposure of sensitive user information, leading to identity theft, account compromise, or financial fraud. Attackers may gain access to authentication credentials or session tokens, allowing unauthorized access.

### Recommendation

Store sensitive data securely using strong encryption algorithms and hashing functions for passwords. Avoid storing sensitive data on the client side; if necessary, ensure it is encrypted and protected with proper security flags. Implement strict access controls to prevent unauthorized access to stored data.

### References

<https://owasp.org/www-project-mobile-top-10/2016-risks/m2-insecure-data-storage>

## 2.13 Session Hijacking

### Severity

HIGH

### Vulnerability Status

OPEN

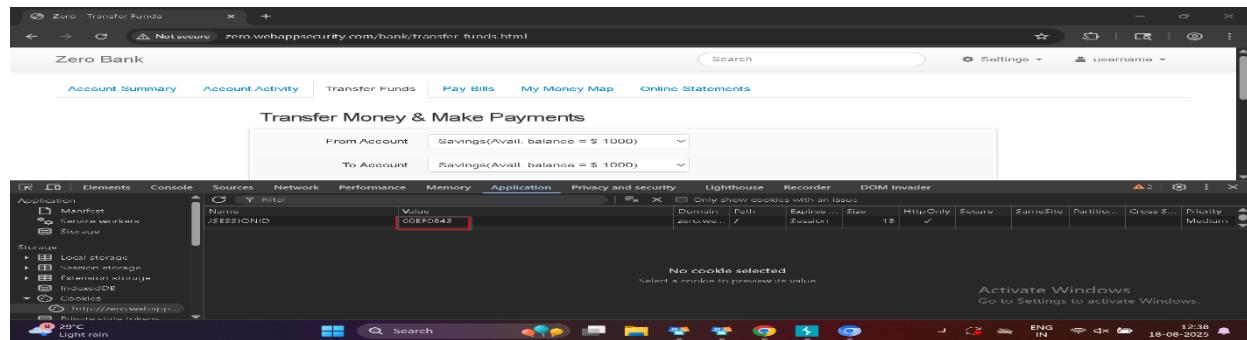
### Affected URL

<http://zero.webappsecurity.com/login.html>

### Detailed Observation

During testing, it was observed that the application's session management is vulnerable to hijacking attacks. Session tokens (e.g., cookies, URL tokens) are transmitted or stored without proper protection, such as secure flags, HttpOnly attributes, or encryption. During testing, it was observed that the application stores sensitive data such as passwords, authentication tokens, personal information, or financial data in an insecure manner.

### Testing PoC



### Impact

Unauthorized access to user accounts, including sensitive information and transactional capabilities. Potential full compromise of user data and administrative privileges if an admin session is hijacked. May facilitate further attacks such as privilege escalation or identity theft.

### Recommendation

Use secure, randomly generated session IDs that are long and unpredictable. Ensure session tokens are transmitted over HTTPS only. Implement session timeout and automatic logout after inactivity.

### References

[https://owasp.org/www-community/attacks/Session\\_hijacking\\_attack](https://owasp.org/www-community/attacks/Session_hijacking_attack)

## 2.14 Browser Cache Weakness

### Severity

HIGH

### Vulnerability Status

OPEN

### Affected URL

http://zero.webappsecurity.com/bank/transfer-funds.html

### Detailed Observation

During testing, it was observed that sensitive information, such as personal data, authentication tokens, or confidential pages, is being cached by the browser. This occurs because HTTP headers like are either missing, misconfigured, or set to allow caching of sensitive content.

### Testing PoC

The screenshots illustrate a browser cache weakness. In the first screenshot, the 'Signin' button is highlighted, suggesting that user credentials might be stored in the browser's cache. In the second screenshot, the transfer form fields are highlighted, indicating that sensitive transaction details could also be cached.

## **Impact**

---

Exposure of sensitive user information stored in the browser cache. Unauthorized access to confidential pages or data by anyone with physical or remote access to the client machine. Potential facilitation of session hijacking or other attacks if cached pages include session tokens or personal data.

## **Recommendation**

---

Configure HTTP headers to prevent caching of sensitive pages. Configure HTTP headers to prevent caching of sensitive pages. Implement session expiration and automatic logout mechanisms.

## **References**

---

[https://cheatsheetseries.owasp.org/cheatsheets/Cache\\_Control\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cache_Control_Cheat_Sheet.html)

## 2.15 Authentication Bypass

### Severity

**CRITICAL**

### Vulnerability Status

**OPEN**

### Affected URL

<http://zero.webappsecurity.com/admin/users.html>

### Detailed Observation

During testing, it was observed that the application's authentication mechanism is bypassed through parameter manipulation.

### Testing PoC

Name	Password	SSN
Leeroy Jenkins	VIZ10AWT8VL	536-48-3769
Stephen Bowen	OT207BXM0BE	607-58-7435
Linus Moran	FK004SXATI	247-54-1719
Nero Chan	TXJ77CQ0SEI	578-13-3713
Kadeem Higgins	MFC50OQE7VO	449-20-3206
Quinn Burks	HWW97ZUM3NK	008-70-6738
Davis Thompson	RGD75SHB0TG	574-56-1932
Lester Keller	EIJ79NLTOFP	330-58-4012

### Impact

Unauthorized access to sensitive data and functionalities. Potential data theft, modification, or deletion. Increased risk of further attacks like privilege escalation, data exfiltration, or fraudulent transactions

### Recommendation

Enforce strong authentication mechanisms, including complex passwords and MFA. Validate all inputs and parameters to prevent manipulation. Ensure server-side authentication checks are mandatory for every request to restricted resources.

### References

[https://cheatsheetseries.owasp.org/cheatsheets/Authentication\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html)

## 2.16 Insecure Password Reset

### Severity

**CRITICAL**

### Vulnerability Status

**OPEN**

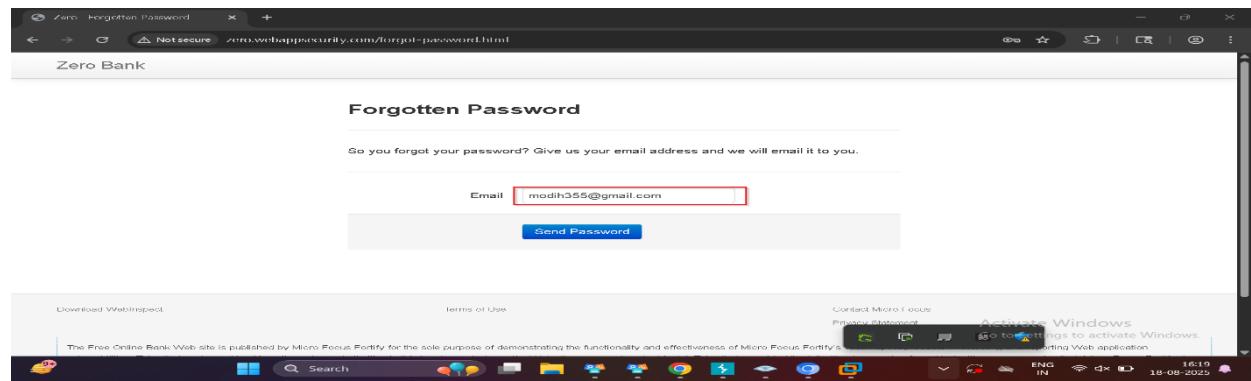
### Affected URL

<http://zero.webappsecurity.com/forgot-password.html>

### Detailed Observation

During testing, it was observed that the application's password reset functionality is insecure. It accepts the unformat email to give password link.

### Testing PoC



### Impact

Unauthorized access to user accounts. Compromise of sensitive personal or financial data. Potential account takeover leading to fraud or data manipulation.

### Recommendation

Implement secure, time-limited, and random password reset tokens. Enforce strong verification for password reset requests. Use HTTPS to transmit all password reset links and tokens. Implement rate limiting and monitoring to prevent abuse of password reset functionality.

### References

[https://cheatsheetseries.owasp.org/cheatsheets/Forgot\\_Password\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Forgot_Password_Cheat_Sheet.html)

## 2.17 HTTP Strict Transport Security Missing

### Severity

HIGH

### Vulnerability Status

OPEN

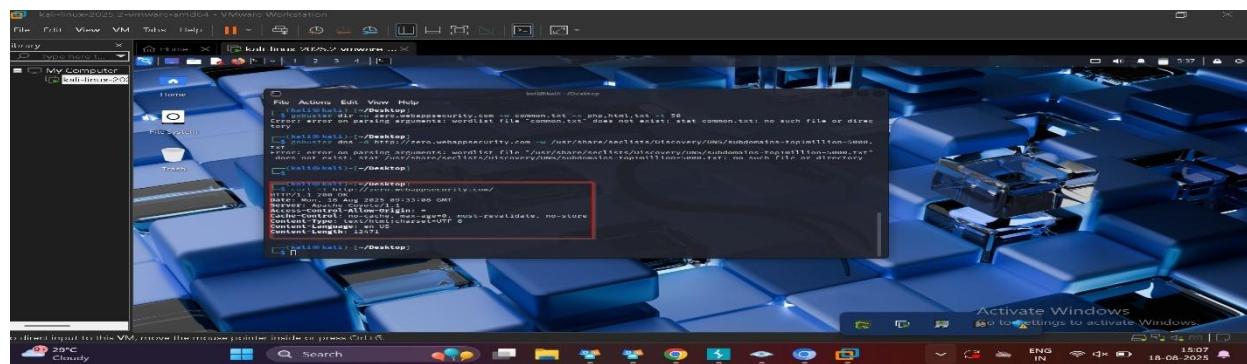
### Affected URL

<http://zero.webappsecurity.com/bank/pay-bills.html>

### Detailed Observation

During testing, it was observed that the application does not implement the HTTP Strict Transport Security (HSTS) response header.

### Testing PoC



### Impact

Sensitive information can be intercepted over an insecure HTTP connection. Increased risk of man-in-the-middle attacks, session hijacking, and credential theft.

### Recommendation

Enable and enforce HSTS by configuring the server. Ensure HTTPS is correctly configured with a valid SSL/TLS certificate. Register the domain for HSTS Preload List. Redirect all HTTP traffic to HTTPS

### References

<https://hstspreload.org/>

## 2.18 CORS Misconfiguration

### Severity

HIGH

### Vulnerability Status

OPEN

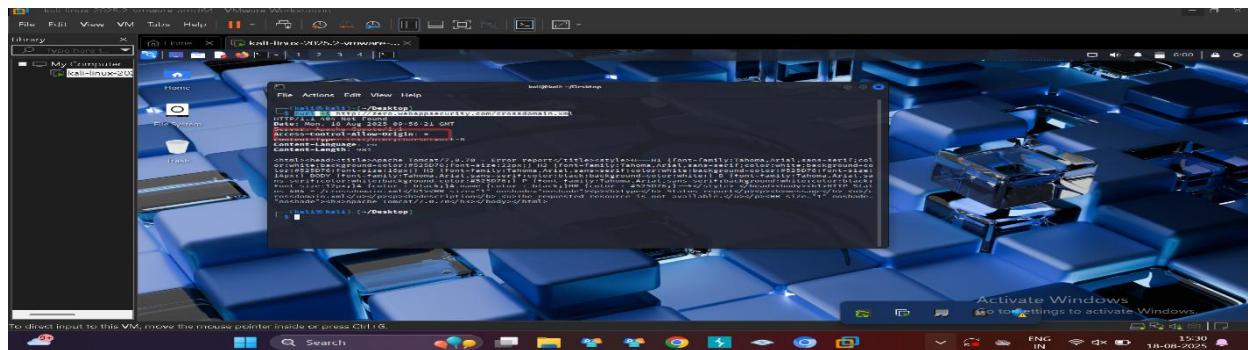
### Affected URL

http://zero.webappsecurity.com/bank/pay-bills.html

### Detailed Observation

During testing, it was observed that the application uses a cross-domain policy file that is either missing proper restrictions or overly permissive (e.g., allows access from all domains using \*). This insecure configuration enables untrusted domains to interact with the application and access sensitive data or services that should be restricted to trusted origins only.

### Testing PoC



### Impact

Malicious domains can send requests to the vulnerable application, potentially bypassing the same-origin policy. Exposure of sensitive data such as user information, authentication tokens, or API responses. Increased risk of cross-site request forgery (CSRF), data theft, and session hijacking.

### Recommendation

Review and restrict the cross-domain policy file to only allow trusted domains. Avoid using wildcards (\*) in domain permissions. Restrict access to only the specific resources that are required by external domains.

### References

[https://owasp.org/www-community/attacks/CORS\\_Configuration](https://owasp.org/www-community/attacks/CORS_Configuration)

## 2.19 Privilege Escalation

### Severity

**CRITICAL**

### Vulnerability Status

**OPEN**

### Affected URL

http://zero.webappsecurity.com/admin/

### Detailed Observation

During testing, it was observed that the application allows users with lower-level privileges to perform actions or access resources intended only for higher-privileged accounts.

### Testing PoC

The screenshot shows a web browser window displaying a banking application. The URL in the address bar is `zero.webappsecurity.com/bank/account-summary.html`. The page content includes:

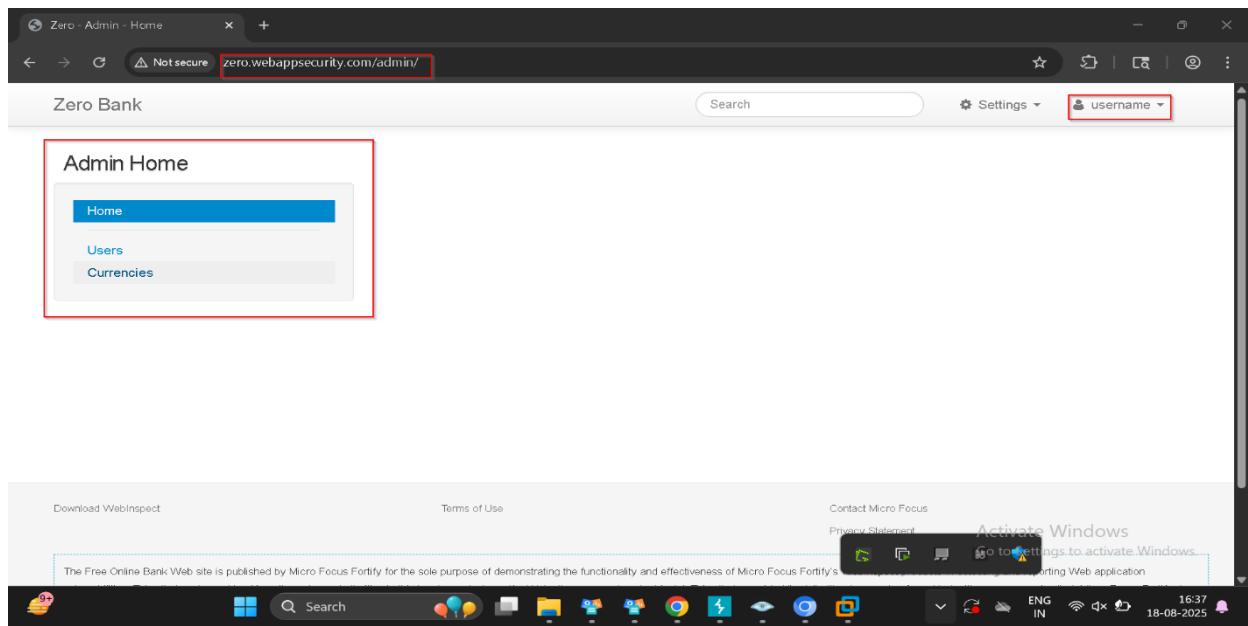
- Cash Accounts**

Account	Balance
Savings	\$1000
Savings	\$1548
- Investment Accounts**

Account	Balance
Brokerage	\$197
- Credit Accounts**

Account	Credit Card	Balance
Checking	VISA 4485-5368-3381-1879	\$500.2
Credit Card	VISA 4716-9811-6719-3943	-\$265

The top right of the page has a dropdown menu labeled "username". The taskbar at the bottom of the screen shows various system icons and the date/time "18-08-2025 16:35".



## Impact

Unauthorized users can gain administrative or higher-level privileges. Full compromise of the application, including creation/deletion of accounts, modification of financial transactions, or access to sensitive data. Potential for data manipulation, fraud, and service disruption.

## Recommendation

Implement strict role-based access control and enforce authorization checks at the server-side for every sensitive function. Validate user roles and permissions before executing privileged operations. Avoid relying solely on client-side controls for access restrictions.

## References

[https://owasp.org/Top10/A01\\_2021-Broken\\_Access\\_Control/](https://owasp.org/Top10/A01_2021-Broken_Access_Control/)

## 2.20 Server Version Disclosure

### Severity

MEDIUM

### Vulnerability Status

OPEN

### Affected URL

http://zero.webappsecurity.com/admin/test.txt

### Detailed Observation

During the assessment, it was observed that the web server discloses its version details through HTTP response headers or error messages.

### Testing PoC

The screenshot shows a web browser window for 'Zero - Account Summary' on 'zero.webappsecurity.com/bank/account-summary.html'. The browser interface includes a search bar, settings, and a user dropdown menu. Below the header, there's a navigation bar with 'Account Summary' (highlighted), 'Account Activity', 'Transfer Funds', 'Pay Bills', 'My Money Map', and 'Online Statements'. The main content area displays 'Cash Accounts' with two entries: 'Savings' (\$1000) and 'Savings' (\$1548). It also shows 'Investment Accounts' with one entry: 'Brokerage' (\$197). At the bottom right of the page, there's an 'Activate Windows' watermark. Below the browser is a Windows taskbar with various pinned icons like File Explorer, Google Chrome, and File Manager.

The terminal window at the bottom shows the command:

```
C:\Users\Admin>curl -X PUT http://zero.webappsecurity.com/test.txt -d "This is a test" -i
```

Output:

```
HTTP/1.1 403 Forbidden
Date: Tue, 19 Aug 2020 05:24:30 GMT
Server: Apache-Coyote/1.1
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, PUT, DELETE, TRACE, OPTIONS, PATCH
Allow: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS
Content-Length: 0
Content-Type: text/plain
```

The terminal then shows:

```
C:\Users\Admin>curl -X PUT http://zero.webappsecurity.com/test.txt -d "This is a test" -i
HTTP/1.1 403 Forbidden
Date: Tue, 19 Aug 2020 05:26:34 GMT
Server: Apache-Coyote/1.1
Access-Control-Allow-Origin: *
Content-Type: text/html; charset=utf-8
Content-Language: en
Content-Length: 961
```

Output:

```
<html><head><title>Apache Tomcat/7.0_70 - Error report</title><style>!--H1 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:10px;} H3 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:10px;} B {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;} P {font-family:Tahoma,Arial,sans-serif;background-color:white;color:black;font-size:12px;} A {color : black;}&name {color : black;}&HR {color : #525D76;} /><st><p><b>description:</b> <u>Access to the specified resource has been forbidden.</u></p><hr size="1" noshade="noshade"><h3>Apache To
mcat/7.0_70</h3><body></body></html>
```

The terminal then shows:

```
HTTP/1.1 405 Method Not Allowed
Date: Tue, 19 Aug 2020 05:27:22 GMT
Server: Apache-Coyote/1.1
Access-Control-Allow-Origin: *
Allow: POST, GET, DELETE, OPTIONS, PUT, HEAD
Content-Length: 0
```

The terminal output ends with:

```
Activate Windows Go to Settings to activate Windows
```

## **Impact**

---

Enables attackers to perform fingerprinting of the underlying server and technologies. Increases the risk of targeted attacks using publicly available exploits. May lead to full compromise if the disclosed version has unpatched vulnerabilities.

## **Recommendation**

---

Suppress or modify the Server and other version-related headers. Configure the web server to hide or obfuscate version information. Use custom error pages instead of default error messages to prevent information leakage.

## **References**

---

[https://owasp.org/www-community/attacks/Information\\_exposure\\_through\\_error\\_messages](https://owasp.org/www-community/attacks/Information_exposure_through_error_messages)

## 2.21 Weak Lockout Mechanism

### Severity

HIGH

### Vulnerability Status

OPEN

### Affected URL

http://zero.webappsecurity.com/admin/test.txt

### Detailed Observation

During testing, it was observed that the application has an inadequate account lockout policy after multiple failed login attempts. Accounts are not locked after repeated incorrect login attempts.

### Testing PoC

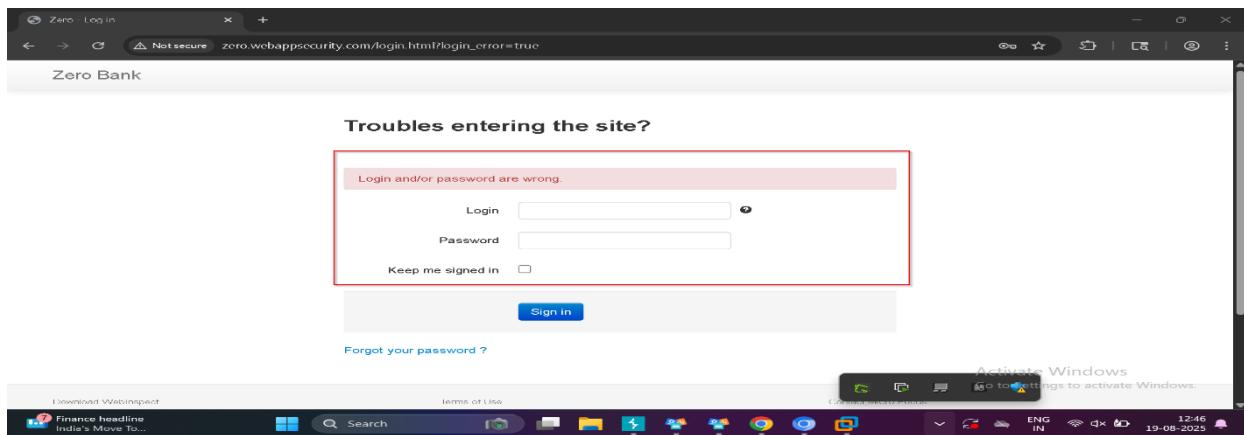
The screenshot shows a web browser window with the title 'Zero - Account Summary'. The address bar indicates the URL is 'zero.webappsecurity.com/bank/account-summary.html'. The main content area is titled 'Zero Bank'. At the top right, there is a 'username' input field with a dropdown arrow, which is highlighted with a red box. Below this, there are three sections: 'Cash Accounts', 'Investment Accounts', and 'Credit Accounts', each containing a table of account balances.

Account	Balance
Savings	\$1000
Savings	\$1548

Account	Balance
Brokerage	\$197

Account	Credit Card	Balance
Checking	VISA 4485-5368-3381-1879	\$-500.2
Credit Card	VISA 4716-9811-6719-3943	\$-265

At the bottom of the screen, there is a taskbar with various icons and a system tray showing the date and time (18-08-2025) and battery status (Activate Windows). A tooltip in the system tray says 'Activate Windows Go to Settings to activate Windows.'



## Impact

Increases the likelihood of **unauthorized access** via brute-force attacks. If common or weak passwords are used, accounts may be compromised. May lead to exposure of sensitive customer or financial data.

## Recommendation

Implement a robust lockout policy. Use progressive delays (rate-limiting) instead of simple lockouts to prevent automated brute force. Encourage strong password policies and multi-factor authentication (MFA).

## References

<https://pages.nist.gov/800-63-3/>

## 2.22 Captcha Missing

### Severity

MEDIUM

### Vulnerability Status

OPEN

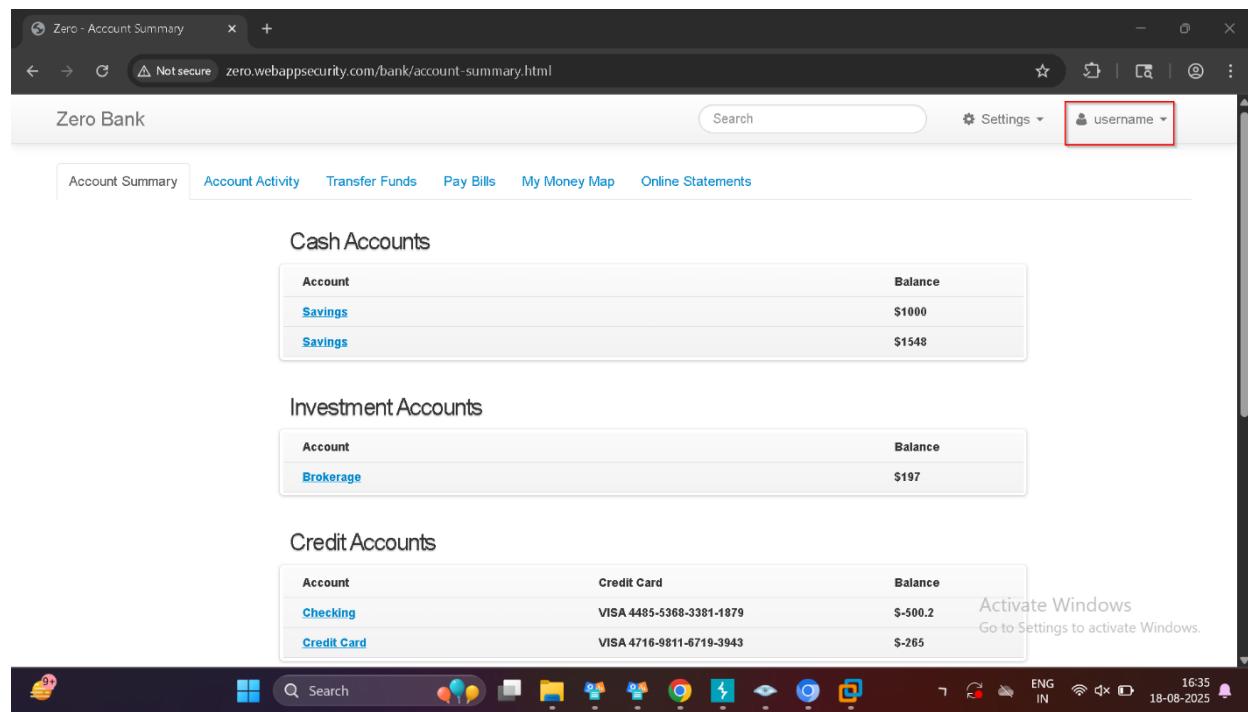
### Affected URL

http://zero.webappsecurity.com/login.html?login\_error=true

### Detailed Observation

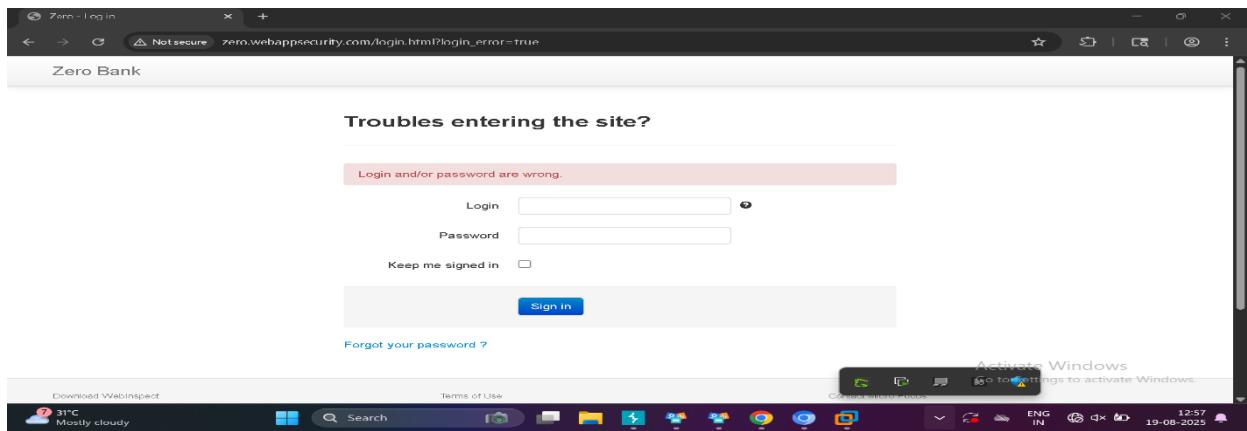
During assessment, it was observed that the application does not implement a CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) on sensitive forms such as login page.

### Testing PoC



The screenshot shows a web browser window with the following details:

- Address Bar:** Shows the URL `zero.webappsecurity.com/bank/account-summary.html`. A warning icon indicates "Not secure".
- Header:** Displays the title "Zero Bank".
- User Input:** A search bar and a "Settings" dropdown are visible. The "username" input field is highlighted with a red border.
- Navigation:** Buttons for Account Summary, Account Activity, Transfer Funds, Pay Bills, My Money Map, and Online Statements.
- Cash Accounts:** A table showing two entries for "Savings" accounts with balances of \$1000 and \$1548 respectively.
- Investment Accounts:** A table showing one entry for "Brokerage" account with a balance of \$197.
- Credit Accounts:** A table showing two entries: a "Checking" account with a VISA card ending in 1879 and a "Credit Card" account with a VISA card ending in 3943. Both have negative balances of -\$500.2 and -\$265 respectively.
- System Status:** A message at the bottom right says "Activate Windows Go to Settings to activate Windows." It also shows system icons for battery, signal, and date/time (18-08-2025).



## **Impact**

---

Increased risk of account takeover due to brute-force or credential stuffing attacks. Attackers can flood the application with fake requests, impacting availability (DoS). Increases server load, degrading performance for legitimate users.

## **Recommendation**

---

Implement CAPTCHA on login pages. Combine with rate limiting and account lockout mechanisms for stronger protection. Regularly update CAPTCHA libraries to the latest versions to avoid bypass techniques.

## **References**

---

<https://developers.google.com/recaptcha>

## 2.23 Server Side Validation is Absent

### Severity

HIGH

### Vulnerability Status

OPEN

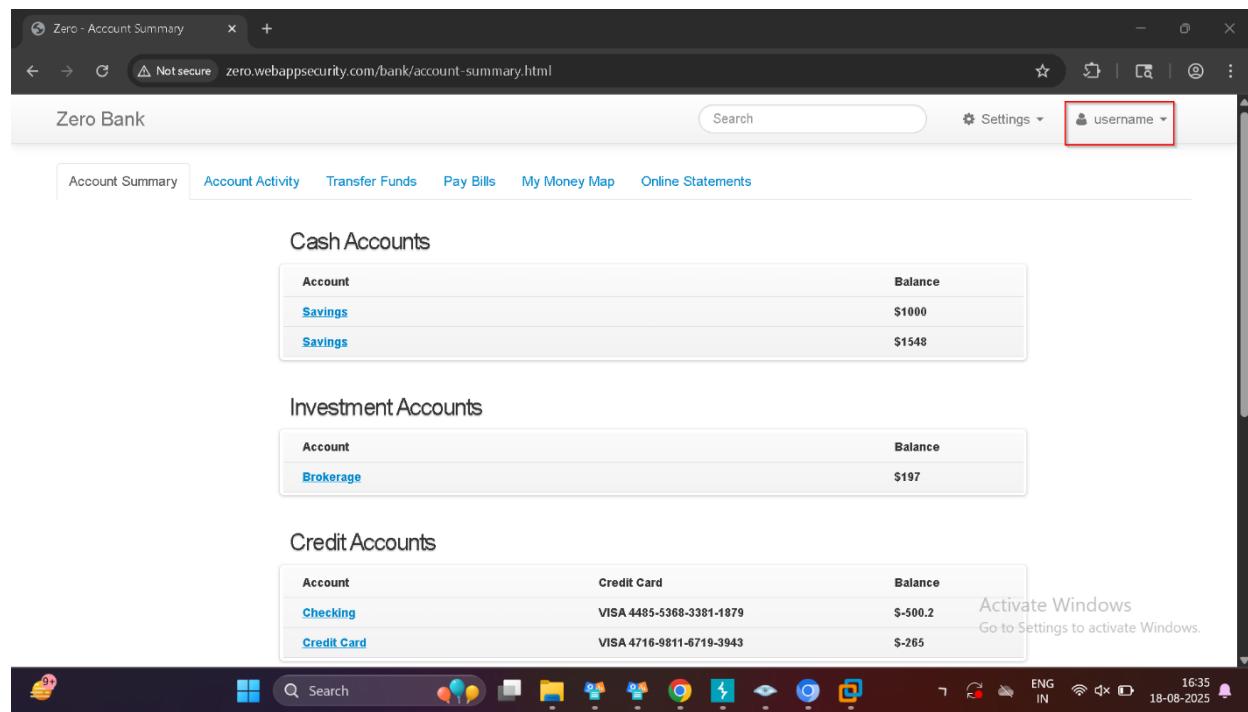
### Affected URL

http://zero.webappsecurity.com/forgotten-password-send.html

### Detailed Observation

During testing, it was observed that the application relies solely on client-side validation to validate user input. No proper server-side validation is enforced for critical input fields, including password reset form.

### Testing PoC



The screenshot shows a web browser window with the following details:

- Address Bar:** Shows the URL `zero.webappsecurity.com/bank/account-summary.html`.
- Header:** "Zero Bank" and a search bar.
- User Authentication:** A dropdown menu labeled "username" is highlighted with a red box.
- Navigation:** Tabs include "Account Summary" (selected), "Account Activity", "Transfer Funds", "Pay Bills", "My Money Map", and "Online Statements".
- Content Area:**
  - Cash Accounts:** A table showing two accounts: "Savings" with a balance of \$1000 and "Savings" with a balance of \$1548.
  - Investment Accounts:** A table showing one account: "Brokerage" with a balance of \$197.
  - Credit Accounts:** A table showing two accounts: "Checking" with a VISA card ending in 1879 and a balance of \$-500.2, and "Credit Card" with a VISA card ending in 3943 and a balance of \$-265.
- System Status Bar:** Shows system icons, language ("ENG IN"), battery level, signal strength, and the date/time ("18-08-2025 16:35"). A message "Activate Windows Go to Settings to activate Windows." is displayed.

The screenshot shows the Burp Suite interface. The top navigation bar includes 'Burp', 'Project', 'Interceptor', 'Repeater', 'View', 'Help', 'Dashboard', 'Target', 'Proxy', 'Interceptor', 'Repeater', 'Decoder', 'Computer', 'Logger', 'Organizer', 'Endpoints', and 'Learn'. Below the dashboard, there are tabs for 'HTTP history' and 'Websockets history'. A search bar at the top says 'Filter settings: Hitting CRLF, image and general identity content'. The main pane displays a list of captured requests with columns: Host, Method, URL, Params, Edited, Status code, Length, MIME type, Extension, Title, Notes, TLS, IP, and Cookies. The status codes range from 200 to 209. The title column shows various page titles like 'Apache Tomcat/7.0.40...', 'Apache Tomcat/7.0.40...', 'Zoho - Contact Us', and 'Zoho - Forget Password'. The notes column contains details such as 'Apache Tomcat/7.0.40...', 'Apache Tomcat/7.0.40...', 'Zoho - Contact Us', and 'Zoho - Forget Password'. The TLS column shows 'TLS' or 'SSL'. The IP column shows '127.0.0.1'. The Cookies column shows '0000' or '4000'. The Time column shows timestamps like '11:50:28 19-08-2025' and '11:50:22 19-08-2025'. The Listener port column shows '8080' and '443'. The Start and End columns show '0' and '1'. The bottom left shows the event log with one issue. The bottom right shows system status: Windows 10 Pro, Intel(R) Core(TM) i3-10110U CPU @ 1.60GHz, RAM 8.00GB, and a battery level of 120%.

## Impact

Increased risk of data manipulation, data corruption, or unauthorized access. Potential execution of malicious code on the server Exposure of sensitive data to attackers.

## Recommendation

Implement server-side validation for all user inputs, even if client-side validation exists. Use prepared statements or parameterized queries for database operations. Apply proper encoding and escaping for output to prevent XSS.

## References

[https://cheatsheetseries.owasp.org/cheatsheets/Input\\_Validation\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html)

## 2.24 Clickjacking

### Severity

HIGH

### Vulnerability Status

OPEN

### Affected URL

<http://zero.webappsecurity.com/>

### Detailed Observation

During testing, it was found that the application does not implement sufficient protection against Clickjacking attacks. The tested web pages can be embedded within an <iframe> on a malicious domain, allowing attackers to trick users into unintentionally performing actions.

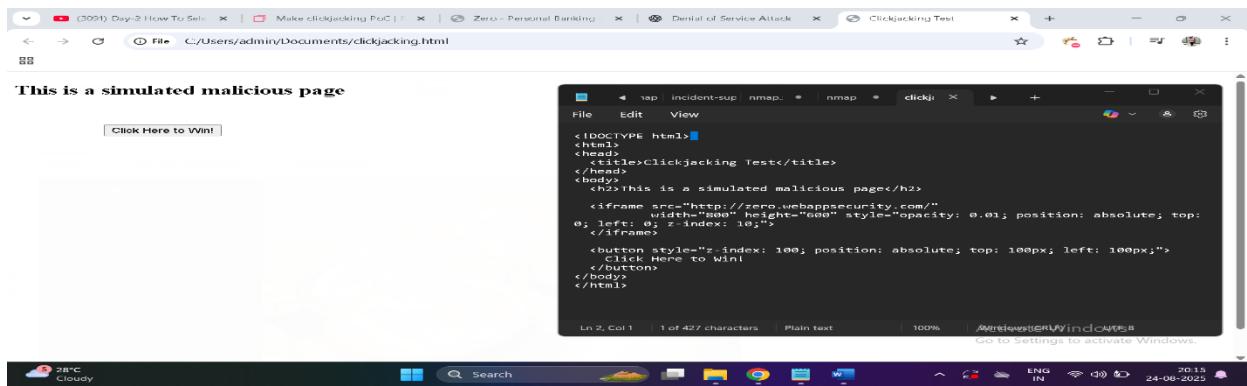
### Testing PoC

The screenshot shows a web browser window with the title 'Zero - Account Summary'. The address bar indicates the site is 'Not secure' and shows the URL 'zero.webappsecurity.com/bank/account-summary.html'. The main content area displays a 'Zero Bank' account summary page. At the top, there are tabs for 'Account Summary', 'Account Activity', 'Transfer Funds', 'Pay Bills', 'My Money Map', and 'Online Statements'. Below these tabs, there are three sections: 'Cash Accounts', 'Investment Accounts', and 'Credit Accounts'. Each section contains a table with columns for 'Account', 'Credit Card' (under Credit Accounts), and 'Balance'. A red box highlights the 'username' input field in the top right corner of the page header. At the bottom of the screen, a taskbar is visible with various icons and system status indicators, including a notification for 'Activate Windows'.

Account	Credit Card	Balance
Savings		\$1000
Savings		\$1548

Account	Credit Card	Balance
Brokerage		\$197

Account	Credit Card	Balance
Checking	VISA 4485-5368-3381-1879	\$-500.2
Credit Card	VISA 4716-9811-6719-3943	\$-265



## Impact

Attackers can trick users into executing unintended actions on their accounts. Potential for financial fraud, unauthorized transactions, or account takeover. May lead to loss of user trust and reputational damage. Attack could be combined with phishing/social engineering for higher impact.

## Recommendation

Implement the X-Frame-Options HTTP response header with values like DENY or SAMEORIGIN. Use Content-Security-Policy (CSP) frame-ancestors directive to restrict iframe embedding. Regularly test for Clickjacking by attempting to load sensitive pages within iframes.

## References

<https://owasp.org/www-community/attacks/Clickjacking>

## 2.25 WAF Missing

### Severity

MEDIUM

### Vulnerability Status

OPEN

### Affected URL

<http://zero.webappsecurity.com/>

### Detailed Observation

The application is deployed without a Web Application Firewall (WAF). As a result, there is no additional security layer to detect and mitigate common web-based attacks such as SQL Injection, Cross-Site Scripting (XSS), Local File Inclusion (LFI), Remote File Inclusion (RFI), and automated bot attacks.

### Testing PoC

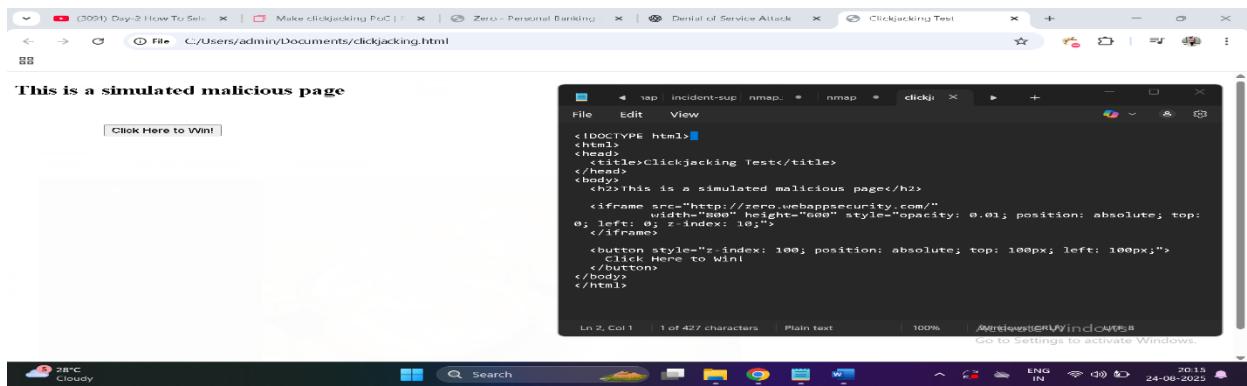
The screenshot shows a web browser window with the following details:

- URL:** zero.webappsecurity.com/bank/account-summary.html
- Header:** The top right corner of the page header contains a red box around the "username" input field.
- Content:**
  - Cash Accounts:**

Account	Balance
Savings	\$1000
Savings	\$1548
  - Investment Accounts:**

Account	Balance
Brokerage	\$197
  - Credit Accounts:**

Account	Credit Card	Balance
Checking	VISA 4485-5368-3381-1879	\$-500.2
Credit Card	VISA 4716-9811-6719-3943	\$-265
- Bottom Bar:** Shows various system icons and status information: 9+, Search, Notifications, Battery, ENG IN, 16:35, 18-08-2025.



## Impact

---

Increased risk of successful exploitation of web application vulnerabilities. Attackers can directly interact with the backend server without filtering or monitoring. Higher chances of data breach, service disruption, and reputational damage.

## Recommendation

---

Implement a Web Application Firewall. Configure WAF rules to block common attack patterns. Enable logging and monitoring to detect suspicious activity.

## References

---

[https://owasp.org/www-community/Web\\_Application\\_Firewall](https://owasp.org/www-community/Web_Application_Firewall)

## 2.26 Security Headers Missing

### Severity

MEDIUM

### Vulnerability Status

OPEN

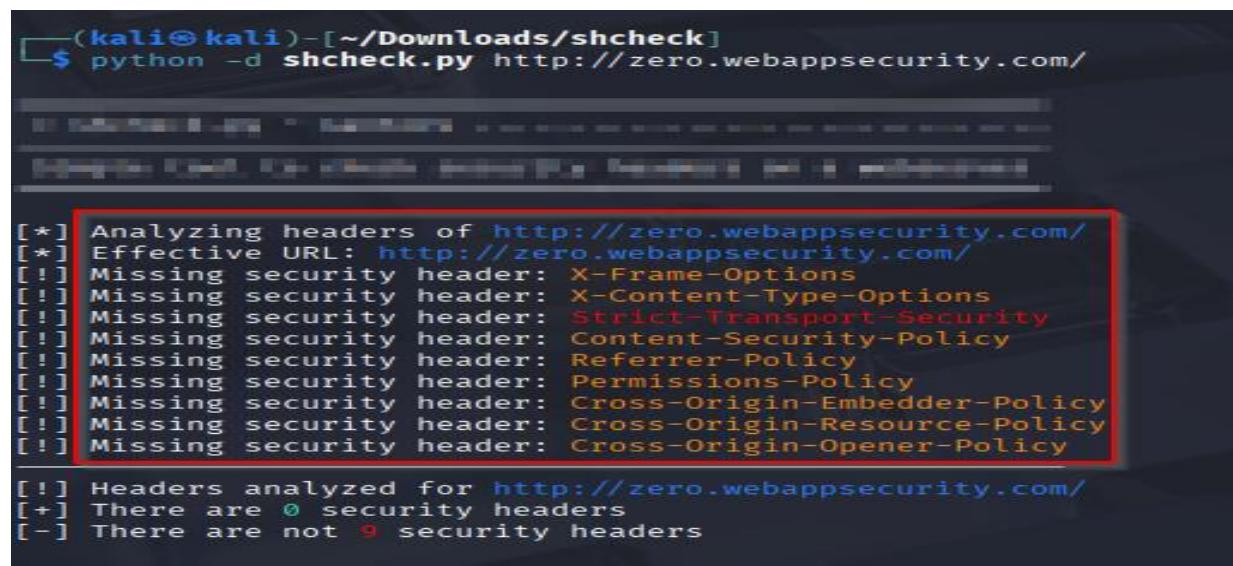
### Affected URL

<http://zero.webappsecurity.com/>

### Detailed Observation

During testing it was found that security headers in website are missing.

### Testing PoC



```
(kali㉿kali)-[~/Downloads/shcheck]
$ python -d shcheck.py http://zero.webappsecurity.com/
[*] Analyzing headers of http://zero.webappsecurity.com/
[*] Effective URL: http://zero.webappsecurity.com/
[!] Missing security header: X-Frame-Options
[!] Missing security header: X-Content-Type-Options
[!] Missing security header: Strict-Transport-Security
[!] Missing security header: Content-Security-Policy
[!] Missing security header: Referrer-Policy
[!] Missing security header: Permissions-Policy
[!] Missing security header: Cross-Origin-Embedder-Policy
[!] Missing security header: Cross-Origin-Resource-Policy
[!] Missing security header: Cross-Origin-Opener-Policy
[!] Headers analyzed for http://zero.webappsecurity.com/
[+] There are 0 security headers
[-] There are not 9 security headers
```

### Impact

Increased risk of Cross-Site Scripting (XSS). Possible Clickjacking attacks due to lack of framing protection. Exposure to MIME sniffing vulnerabilities.

### Recommendation

Implement the security headers like CSP, X-Frame Options, HSTS, etc.

### References

<https://owasp.org/www-project-secure-headers/>

## 2.27 Stored XSS

### Severity

HIGH

### Vulnerability Status

OPEN

### Affected URL

<http://zero.webappsecurity.com/bank/pay-bills.html>

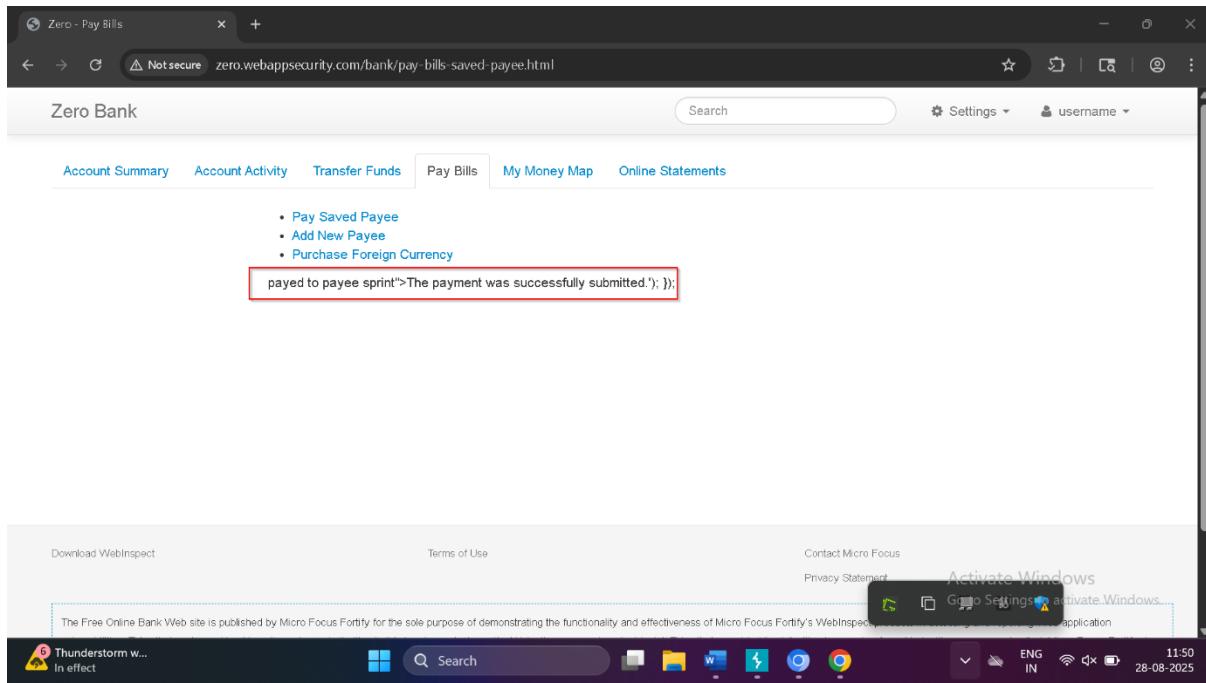
<http://zero.webappsecurity.com/bank/pay-bills-saved-payee.html>

### Detailed Observation

During testing it was observed that stored XSS occur on the pay saved payee field. User supplied input is saved on the server.

### Testing PoC

The screenshot shows a web browser window for 'Zero - Pay Bills' at 'zero.webappsecurity.com/bank/pay-bills.html'. The page title is 'Zero Bank'. The navigation bar includes 'Account Summary', 'Account Activity', 'Transfer Funds', 'Pay Bills' (selected), 'My Money Map', and 'Online Statements'. Below the navigation is a toolbar with 'Pay Saved Payee', 'Add New Payee', and 'Purchase Foreign Currency'. A main section titled 'Make payments to your saved payees' contains fields for 'Payee' (set to 'Sprint'), 'Account' (set to 'Savings'), 'Amount' (\$ <script>a), 'Date' (<script>alert('XSS')</script>), and 'Description' (<script>alert('XSS')</script>). The 'Amount', 'Date', and 'Description' fields are highlighted with a red box. At the bottom right is a blue 'Pay' button with a tooltip 'Private Windows Go to Settings to activate Windows.' The browser's status bar shows weather (29°C, Heavy rain), search, file, and system icons, along with the date '28-08-2025' and time '11:47'.



## Impact

A Stored Cross-Site Scripting (XSS) vulnerability allows attackers to inject malicious scripts that are permanently stored on the server and executed whenever other users access the affected page. This can lead to session hijacking, where attackers steal authentication cookies or tokens, phishing attacks through fake forms or UI manipulation, and malware distribution by redirecting users to malicious sites. Additionally, attackers may perform unauthorized actions on behalf of other users, potentially leading to privilege escalation or sensitive data modification.

## Recommendation

Sanitize and encode all user input before storing or rendering in HTML contexts. Avoid using inner HTML, document.write, etc. with untrusted input. Implement a Content Security Policy (CSP) to restrict script execution.

## References

[https://owasp.org/www-community/attacks/xss/?utm\\_source=chatgpt.com#stored-xss](https://owasp.org/www-community/attacks/xss/?utm_source=chatgpt.com#stored-xss)

## 2.28 Client side Resource Manipulation

### Severity

HIGH

### Vulnerability Status

OPEN

### Affected URL

<http://zero.webappsecurity.com/bank/pay-bills-saved-payee.html>

### Detailed Observation

During test it was observed that amount and account id are manipulated in burp suite request.

Due to which response is positive and successfully change the fields.

### Testing PoC

The screenshot shows the Burp Suite interface with the following details:

- Request:** A POST request to `/bank/pay-bills-saved-payee.html` with the following headers:
  - Content-Type: application/x-www-form-urlencoded
  - Content-Length: 67
  - Accept: \*/\*
  - Accept-Encoding: gzip, deflate, br
  - Cookie: JSESSIONID=BASCTD86
  - Connection: keep-alive
  - Upgrade-Insecure-Requests: 1
  - User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36
  - Accept-Language: en-US,en;q=0.9
  - Cache-Control: max-age=0
  - Accept: \*/\*
- Response:** An HTTP/1.1 200 OK response from Apache-Coyote/1.1. The response body contains HTML code for a payment page, including meta tags for viewport and character encoding, and links to CSS and JS files.
- Inspector:** Shows the Request attributes, Request query parameters, Request body parameters, Request cookies, Request headers, and Response headers sections.
- Bottom Bar:** Includes a search bar, highlights button, and system status indicators (Memory: 212.5MB, ENG IN, 17:50, 28-08-2025).

The screenshot shows the Burp Suite interface with the following details:

- Request:**

```
PoC/Pay-Bills-saved-payee.html HTTP/1.1
Host: www.webappsecurity.com
Content-Length: 60
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://www.webappsecurity.com
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://www.webappsecurity.com/bank/pay-bills.html
Accept-Encoding: gzip, deflate, br
Cookie: JSESSIONID=0A527B56
Accept-Language: en-US,en;q=0.9
payee=spint&amount=1000&date=2025-08-25&description=refwv
```
- Response:**

```
HTTP/1.1 200 OK
Date: Mon, 25 Aug 2025 12:11:20 GMT
Server: Apache-Coyote/1.1
Access-Control-Allow-Origin: *
Cache-Control: no-cache, max-age=0, must-revalidate, no-store
Content-Type: text/html;charset=UTF-8
Content-Length: 10239
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Length: 10239

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Bank - Pay Bills</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">
    <meta http-equiv="X-UA-Compatible" content="IE=Edge">
    <link type="text/css" rel="stylesheet" href="/resources/css/bootstrap.min.css"/>
    <link type="text/css" rel="stylesheet" href="/resources/css/main.css"/>
    <link type="text/css" rel="stylesheet" href="/resources/css/about-me.css"/>
    <script src="/resources/js/jquery-1.8.2.min.js">
```
- Inspector:** Shows various tabs for Request attributes, Request query parameters, Request body parameters, Request cookies, Request headers, Response headers, Notes, Custom actions, and more.
- Bottom Status Bar:** Shows system information including battery level (30°C Rain showers), network status (ENG IN), and date (28-08-2025).

## Impact

Unauthorized actions or business logic bypass. Possible authentication or authorization bypass.

## Recommendation

Ensure server validates and normalizes parameters before processing. Only accept expected number of parameters. Implement server-side checks to reject duplicate or unexpected parameters

## References

[https://owasp.org/www-community/attacks/HTTP\\_Parameter\\_Pollution?utm\\_source=chatgpt.com](https://owasp.org/www-community/attacks/HTTP_Parameter_Pollution?utm_source=chatgpt.com)

## 2.29 Weak Account Suspension

## Severity

## **CRITICAL**

## Vulnerability Status

OPEN

## Affected URL

<http://zero.webappsecurity.com/index.html>

## Detailed Observation

During testing it was observed that when user is login in website, server stores username and password in cookies attribute.

# Testing PoC

```
Windows PowerShell x + - □
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\username> for ($i=1; $i -le 10; $i++) {
    & curl -i -X POST -b "username=tester&password=wrongpass" http://zero.websappsecurity.com/login
}
At line:1 char:1
+ & curl -i -X POST -b "username=tester&password=wrongpass" http ...
+ CategoryInfo          : InvalidArgument: () [Invoke-WebRequest], ParameterBindingException
+ FullyQualifiedErrorId : PositionalParameterNotFound,Microsoft.PowerShell.Commands.InvokeWebRequestCommand

Invoke-WebRequest : A positional parameter cannot be found that accepts argument
'username=wrongpass'.
At line:1 char:1
+ & curl -i -X POST -b "username=wrongpass" http ...
+ CategoryInfo          : InvalidArgument: () [Invoke-WebRequest], ParameterBindingException
+ FullyQualifiedErrorId : PositionalParameterNotFound,Microsoft.PowerShell.Commands.InvokeWebRequestCommand

Invoke-WebRequest : A positional parameter cannot be found that accepts argument
'username=wrongpass'.
At line:1 char:1
+ & curl -i -X POST -b "username=wrongpass" http ...
+ CategoryInfo          : InvalidArgument: () [Invoke-WebRequest], ParameterBindingException
+ FullyQualifiedErrorId : PositionalParameterNotFound,Microsoft.PowerShell.Commands.InvokeWebRequestCommand

Invoke-WebRequest : A positional parameter cannot be found that accepts argument
'username=wrongpass'.
At line:1 char:1
+ & curl -i -X POST -b "username=wrongpass" http ...
+ CategoryInfo          : InvalidArgument: () [Invoke-WebRequest], ParameterBindingException
+ FullyQualifiedErrorId : PositionalParameterNotFound,Microsoft.PowerShell.Commands.InvokeWebRequestCommand

Invoke-WebRequest : A positional parameter cannot be found that accepts argument
'username=wrongpass'.
At line:1 char:1
+ & curl -i -X POST -b "username=wrongpass" http ...
+ CategoryInfo          : InvalidArgument: () [Invoke-WebRequest], ParameterBindingException
+ FullyQualifiedErrorId : PositionalParameterNotFound,Microsoft.PowerShell.Commands.InvokeWebRequestCommand

Invoke-WebRequest : A positional parameter cannot be found that accepts argument
'username=wrongpass'.
At line:1 char:1
+ & curl -i -X POST -b "username=wrongpass" http ...
+ CategoryInfo          : InvalidArgument: () [Invoke-WebRequest], ParameterBindingException
+ FullyQualifiedErrorId : PositionalParameterNotFound,Microsoft.PowerShell.Commands.InvokeWebRequestCommand

Invoke-WebRequest : A positional parameter cannot be found that accepts argument
'username=wrongpass'.
At line:1 char:1
+ & curl -i -X POST -b "username=wrongpass" http ...
+ CategoryInfo          : InvalidArgument: () [Invoke-WebRequest], ParameterBindingException
+ FullyQualifiedErrorId : PositionalParameterNotFound,Microsoft.PowerShell.Commands.InvokeWebRequestCommand

Invoke-WebRequest : A positional parameter cannot be found that accepts argument
'username=wrongpass'.
At line:1 char:1
+ & curl -i -X POST -b "username=wrongpass" http ...
+ CategoryInfo          : InvalidArgument: () [Invoke-WebRequest], ParameterBindingException
+ FullyQualifiedErrorId : PositionalParameterNotFound,Microsoft.PowerShell.Commands.InvokeWebRequestCommand

Invoke-WebRequest : A positional parameter cannot be found that accepts argument
'username=wrongpass'.
At line:1 char:1
+ & curl -i -X POST -b "username=wrongpass" http ...
+ CategoryInfo          : InvalidArgument: () [Invoke-WebRequest], ParameterBindingException
+ FullyQualifiedErrorId : PositionalParameterNotFound,Microsoft.PowerShell.Commands.InvokeWebRequestCommand

Invoke-WebRequest : A positional parameter cannot be found that accepts argument
'username=wrongpass'.
At line:1 char:1
+ & curl -i -X POST -b "username=wrongpass" http ...
+ CategoryInfo          : InvalidArgument: () [Invoke-WebRequest], ParameterBindingException
+ FullyQualifiedErrorId : PositionalParameterNotFound,Microsoft.PowerShell.Commands.InvokeWebRequestCommand

Invoke-WebRequest : A positional parameter cannot be found that accepts argument
'username=wrongpass'.
At line:1 char:1
+ & curl -i -X POST -b "username=wrongpass" http ...
+ CategoryInfo          : InvalidArgument: () [Invoke-WebRequest], ParameterBindingException
+ FullyQualifiedErrorId : PositionalParameterNotFound,Microsoft.PowerShell.Commands.InvokeWebRequestCommand

Invoke-WebRequest : A positional parameter cannot be found that accepts argument
'username=wrongpass'.
At line:1 char:1
+ & curl -i -X POST -b "username=wrongpass" http ...
+ CategoryInfo          : InvalidArgument: () [Invoke-WebRequest], ParameterBindingException
+ FullyQualifiedErrorId : PositionalParameterNotFound,Microsoft.PowerShell.Commands.InvokeWebRequestCommand
```

## Impact

Attackers can systematically guess or replay passwords against user accounts, significantly increasing the likelihood of unauthorized access. In addition, if account lockout is poorly implemented, attackers may be able to cause a denial-of-service by locking out legitimate users.

## Recommendation

Implement robust account suspension controls. After a predefined number of failed login attempts (e.g., 5), temporarily suspend the account and require manual resumption via email

verification or password reset. Ensure the resumption process itself is secure and not vulnerable to manipulation. Additionally, log all lockout events and alert administrators in case of repeated attempts.

## **References**

---

[https://owasp.org/www-project-web-security-testing-guide/stable/4-Web\\_Application\\_Security\\_Testing/05-Authentication\\_Testing/07-Testing\\_for\\_Account\\_Suspension\\_Resumption.html?utm\\_source=chatgpt.com](https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/05-Authentication_Testing/07-Testing_for_Account_Suspension_Resumption.html?utm_source=chatgpt.com)

## 2.30 Insecure Remember Password Implementation

### Severity

HIGH

### Vulnerability Status

OPEN

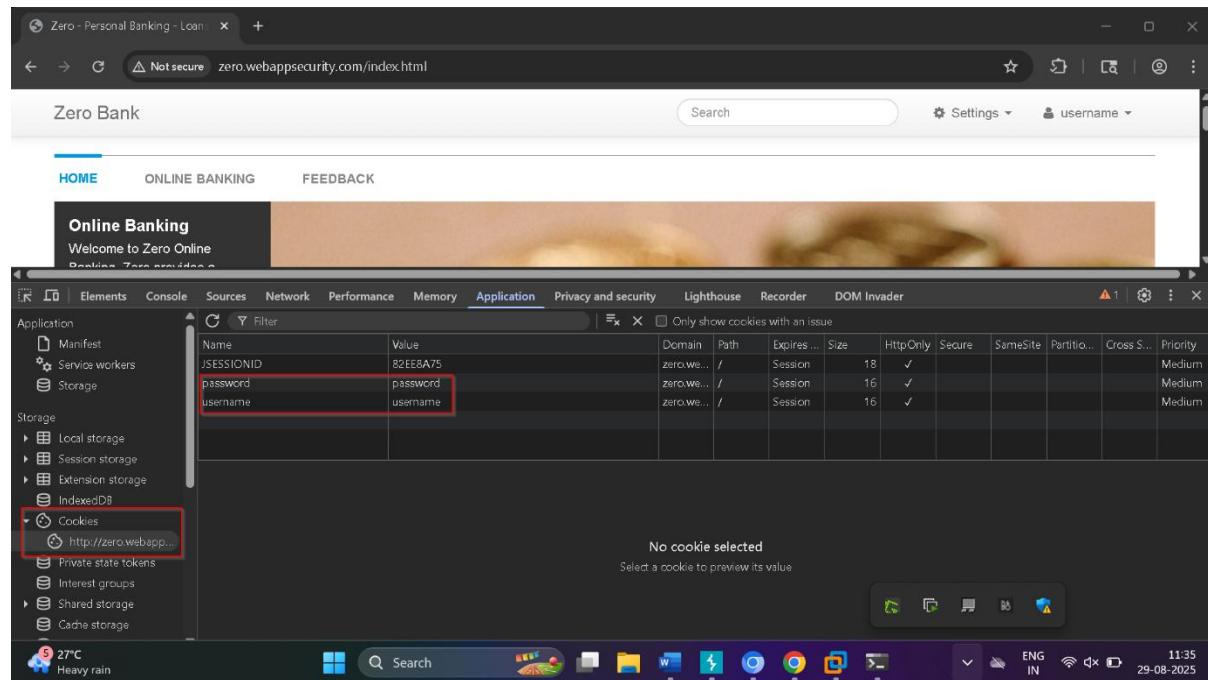
### Affected URL

[http://zero.webappsecurity.com/login.html?login\\_error=true](http://zero.webappsecurity.com/login.html?login_error=true)

### Detailed Observation

During testing it was observed that after failing multiple login attempts account is not locked or suspended. This indicates that the account suspension mechanism is either missing or improperly implemented.

### Testing PoC



The screenshot shows the Chrome DevTools Application tab open, displaying a list of stored cookies for the domain `http://zero.webappsecurity.com`. The table includes columns for Name, Value, Domain, Path, Expires..., Size, HttpOnly, Secure, SameSite, Partition, Cross Site, and Priority. Three cookies are listed:

Name	Value	Domain	Path	Expires...	Size	HttpOnly	Secure	SameSite	Partition	Cross Site	Priority
JSESSIONID	82EE8A75	zero.we...	/	Session	18	✓					Medium
password	password	zero.we...	/	Session	16	✓					Medium
username	username	zero.we...	/	Session	16	✓					Medium

### Impact

Storing the username and password in cookies exposes highly sensitive information to attackers. If an attacker gains access to the victim's browser through malware, shared systems, or physical access, they can easily retrieve the stored credentials. Furthermore, the credentials can also be stolen remotely through attacks such as Cross-Site Scripting (XSS) or via insecure transmission if the Secure flag is not set and cookies are sent over HTTP. Unlike session tokens, which can be invalidated, exposure of actual usernames and passwords results in permanent account compromise and may also impact other systems if the same

password is reused. This significantly increases the risk of unauthorized access and data breaches.

## **Recommendation**

---

The application should never store plaintext or encoded passwords in cookies. Instead, it should implement a secure “Remember Me” feature by generating a long, random token that is stored server-side and mapped to the user’s account. This token should be stored in a cookie protected with the Http Only, Secure, and Same Site flags to mitigate theft. Additionally, the token should be time-bound, expiring after a reasonable duration, and users should have the ability to revoke active tokens or remembered sessions from their account settings. Refactoring the authentication mechanism to align with OWASP best practices will ensure that sensitive credentials remain protected and are never directly exposed on the client side.

## **References**

---

[https://owasp.org/www-community/controls/Remember\\_me?utm\\_source=chatgpt.com](https://owasp.org/www-community/controls/Remember_me?utm_source=chatgpt.com)

## 2.31 Business Logic Break

### Severity

LOW

### Vulnerability Status

OPEN

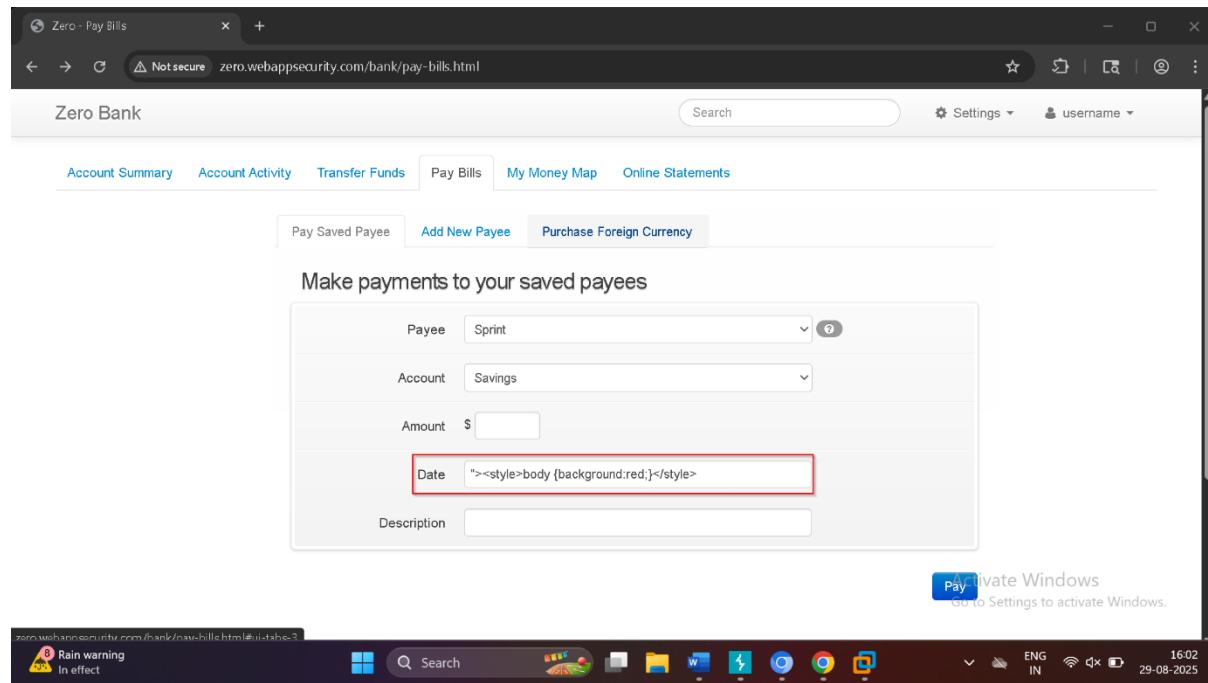
### Affected URL

<http://zero.webappsecurity.com/bank/pay-bills.html>

### Detailed Observation

During testing it was observed that website allows anything in the date field. By which, business logic is break.

### Testing PoC



### Impact

Attackers can exploit the lack of validation to manipulate workflows, bypass approval processes, or gain financial benefits, leading to revenue loss and potential fraud.

### Recommendation

Attackers can exploit the lack of validation to manipulate workflows, bypass approval processes, or gain financial benefits, leading to revenue loss and potential fraud. Enforce strict checks on

parameters such as amount, quantity, and roles. Log and monitor suspicious transactions. Apply segregation of duties and approval workflows for sensitive actions.

## References

---

[https://owasp.org/www-project-web-security-testing-guide/stable/4-Web\\_Application\\_Security\\_Testing/12-Business\\_Logic\\_Testing/01-Test\\_Business\\_Logic\\_Data\\_Validation.html?utm\\_source=chatgpt.com](https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/12-Business_Logic_Testing/01-Test_Business_Logic_Data_Validation.html?utm_source=chatgpt.com)

## 2.32 Outdated Library

### Severity

LOW

### Vulnerability Status

OPEN

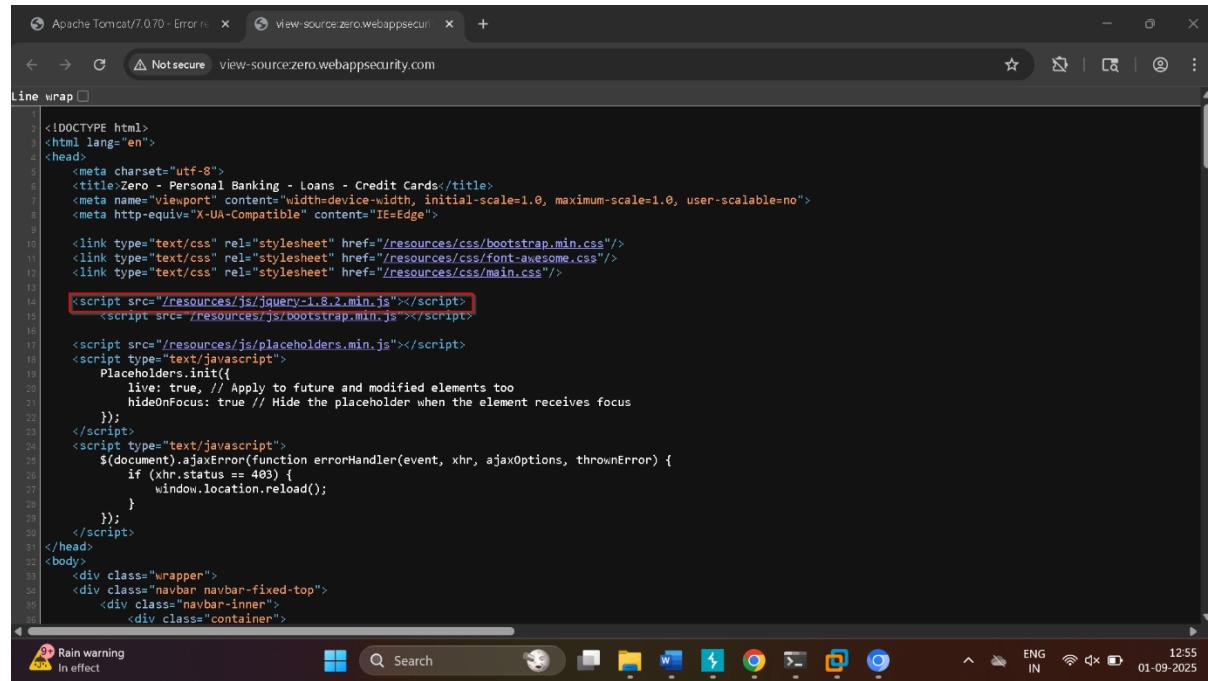
### Affected URL

view-source:<http://zero.webappsecurity.com/>

### Detailed Observation

During testing it was observed that in source code there is jquery library whose version is 1.8.2 which is old and is vulnerable to various attacks.

### Testing PoC



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="utf-8">
5     <title>Zero - Personal Banking - Loans - Credit Cards</title>
6     <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">
7     <meta http-equiv="X-UA-Compatible" content="IE=Edge">
8
9     <link type="text/css" rel="stylesheet" href="/resources/css/bootstrap.min.css"/>
10    <link type="text/css" rel="stylesheet" href="/resources/css/font-awesome.css"/>
11    <link type="text/css" rel="stylesheet" href="/resources/css/main.css"/>
12
13    <script src="/resources/js/jquery-1.8.2.min.js"></script>
14    <script src="/resources/js/bootstrap.min.js"></script>
15
16    <script src="/resources/js/placeholders.min.js"></script>
17    <script type="text/javascript">
18        Placeholders.init({
19            live: true, // Apply to future and modified elements too
20            hideOnFocus: true // Hide the placeholder when the element receives focus
21        });
22    </script>
23    <script type="text/javascript">
24        $(document).ajaxError(function errorHandler(event, xhr, ajaxOptions, thrownError) {
25            if (xhr.status == 403) {
26                window.location.reload();
27            }
28        });
29    </script>
30
31 </head>
32 <body>
33     <div class="wrapper">
34         <div class="navbar navbar-fixed-top">
35             <div class="navbar-inner">
36                 <div class="container">
```

### Impact

These outdated libraries are known to have multiple security vulnerabilities, such as cross-site scripting (XSS), prototype pollution, and UI manipulation attacks. An attacker could exploit these vulnerabilities to execute malicious scripts in users' browsers, hijack sessions, manipulate the user interface, or bypass client-side restrictions, potentially leading to unauthorized actions or data exposure.

### Recommendation

Update all frontend libraries and scripts to their latest stable versions. Specifically, upgrade jQuery to the latest 3.x release, update Bootstrap to the most recent stable version, and remove

or replace deprecated scripts like placeholders.min.js. Additionally, review all client-side code to ensure sensitive logic is validated server-side, implement proper input sanitization, and test the application after updates to confirm compatibility and security.

## **References**

---

[https://snyk.io/vuln/SNYK-JS-JQUERY-174006?utm\\_source=chatgpt.com](https://snyk.io/vuln/SNYK-JS-JQUERY-174006?utm_source=chatgpt.com)