Development Security Guide
Oracle Banking Liquidity Management
Release 14.4.0.0.0

Part No. F30579-01
[May] [2020]

ORACLE
FINANCIAL SERVICES

ORACLE

Table of Contents

1. ABOUT THIS MANUAL 1-1

1.1 PURPOSE: 1-1
1.2 AUDIENCE: 1-1
1.3 SCOPE 1-1

2. HOW TO ADDRESS THE OWASP TOP10 IN ORACLE BANKING LIQUIDITY MANAGEMENT 2-2

2.1 INJECTION 2-2
2.2 BROKEN AUTHENTICATION AND SESSION MANAGEMENT 2-2
2.3 CROSS-SITE SCRIPTING (XSS) 2-3
2.4 INSECURE DIRECT OBJECT REFERENCES 2-3
2.5 SECURITY MISCONFIGURATION 2-4
2.6 SENSITIVE DATA EXPOSURE 2-4
2.7 MISSING FUNCTION LEVEL ACCESS CONTROL 2-5
2.8 CROSS-SITE REQUEST FORGERY (CSRF) 2-6
2.9 USING COMPONENTS WITH KNOWN VULNERABILITIES 2-6
2.10 UNVALIDATED REDIRECTS AND FORWARDS NETWORK SECURITY 2-6

3. SECURING API SERVICES 3-7

1. About this Manual

1.1 Purpose:

This document provides security-related usage and configuration recommendations for Oracle Banking Liquidity Management 14.4.0.0.0. This guide may outline procedures required to implement or secure certain features, but it is also not a general-purpose configuration manual.

1.2 Audience:

This guide is primarily intended for Developers for Oracle Banking Liquidity Management and third party or vendor software's. Some information may be relevant to IT decision makers and users of the application are also included. Readers are assumed to possess basic operating system, network, and system administration skills with awareness of vendor/third-party software's and knowledge of Oracle Banking Liquidity Management application.

1.3 Scope

1.3.1 Read Sections Completely

Each section should be read and understood completely. Instructions should never be blindly applied. Relevant discussion may occur immediately after instructions for an action, so be sure to read whole sections before beginning implementation.

1.3.2 Understand the Purpose of this Guidance

The purpose of the guidance is to provide security-relevant code and configuration recommendations.

1.3.3 Limitations

This guide is limited in its scope to security-related guideline for developers.

2. How to address the OWASP Top10 in Oracle Banking Liquidity Management

2.1 Injection

Injection flaws occur when an application sends untrusted data to an interpreter. Injection flaws are very prevalent, particularly in legacy code. They are often found in SQL, LDAP, Xpath, or SQL queries; OS commands; XML parsers, SMTP Headers, program arguments, etc. Injection flaws are easy to discover when examining code.

Oracle Banking Liquidity Management uses Oracle database and it has adequate inbuilt techniques to prevent SQL injections as underlined below:-

1. **Use of prepared statements (parameterized queries)** - Oracle Banking Liquidity Management uses Prepared Statement with bind variables to construct and execute SQL statements in JAVA.

2. **Use of Stored procedures**-- Stored procedures have the same effect as the use of prepared statements when implemented safely. 'Implemented safely' means the stored procedure does not include any unsafe dynamic SQL generation. Oracle Banking Liquidity Management uses safe Java stored procedures calls.

   In addition to the above, wherever dynamic queries exist, Oracle Banking Liquidity Management uses adequate defence to sanitize the untrusted input.

3. **Escaping all user supplied input--** This third technique is to escape user input before putting it in a query. If it's a concern that rewriting the dynamic queries as prepared statements or stored procedures might break the application or adversely affect performance, then this might be the best approach for the purpose. However, this methodology is frail compared to using parameterized queries and there's no guarantee that it will prevent all SQL Injection in all situations.

2.2 Broken Authentication and Session Management

In Oracle Banking Liquidity Management, applications are stateless micro services based architecture.

2.2.1 Cryptography used

PCI council defines Strong Cryptography as:
Cryptography based on industry-tested and accepted algorithms, along with strong key lengths and proper key-management practices. Cryptography is a method to protect data and includes both encryption (which is reversible) and hashing (which is not reversible, or "one way"). SHA-1 is an example of an industry-tested and accepted hashing algorithm. Examples of industry-tested and accepted standards and algorithms for encryption include AES (128 bits and higher), TDES (minimum double-length keys), RSA (1024 bits and higher), ECC (160 bits and higher), and ElGamal (1024 bits and higher).

- Encryption algorithm: The application leverages AES encryption algorithm to store sensitive information into properties file. This algorithm uses 256 bit secret key for encryption and decryption which would be stored at property file.
- Hashing algorithm: Oracle Banking Liquidity Management platform service leverages HS-512 hashing algorithm with random salt for JWT.

2.3 Cross-Site Scripting (XSS)

XSS for Oracle Banking Liquidity Management handled by OJET. Hence application developer's need not to handle specifically

2.4 Insecure Direct Object References

- Use of prepared statements (parameterized queries)

- Oracle Banking Liquidity Management uses parameterized JPQL/SQL queries with bind variables to construct and execute SQL statements in JAVA.

- Input Validation: Oracle Banking Liquidity Management is a web based application, the request data from browser to server will be passed using request headers and request parameters. All the request fields coming from the client are validated using white list validation to prevent cross site scripting.

- User defined methods used for input validation which checks each character of the request field with a range of allowed characters. In addition, OJET framework handles the input attribute validations. User defined methods escapeJavaScript(), escapeHTML() and escapeURL() will sanitize the output data before flushing it into client browser.

- escapeJavaScript() will escape all characters except immune JavaScript characters and alphanumeric characters in the ASCII character set. All other characters are encoded using the \\xHH or \\uHHHH notation for representing ASCII or Unicode sequences.

- escapeHTML() will escape the characters with equivalent HTML entities obtained from the lookup map. Lookup map will have entities such as amp, quot, lt, gt etc.

- escapeURL() will encode the URL using URLEncoder class.

- White list validation is also used to restrict Image/signature/excel upload and to check rights for every operation performed by user.

- Field validation: Field level validations exist for all mandatory fields. Database too had limits on the type and the length of data. Blacklisted characters are not allowed in the mandatory fields. Nevertheless, Oracle Banking Liquidity Management has free-text fields, which takes all data, entered by the user, as a String.

- Restriction on Blacklist characters: Blacklisted characters on Oracle Banking Liquidity Management handled by OJET. Hence application developer's need not to handle specifically..

2.5 Security Misconfiguration

- Configuration files: Configuration files are securely placed inside the Classes folder of the WEB-INF folder which is not publicly accessible.

- Exception handling in java: Different types of exceptions can rise in application. Java exceptions handled using try catch blocks available in java. Sometimes we use the Throw statement to throw an exception which is caught by the catch block. Caught exceptions will be written into the log files for the debug purpose whenever required. Whenever any exception occurs in application, proper information used to send to the front end user by showing alert.

- BI Publisher Reports - generation and access: The application uses a sandbox for placing the generated reports file into a sandbox area. The sandbox is placed in a specified location (the location will be specified in the properties file) on the server. The application validates if the user has explicit Rights to generate Reports.

2.6 Sensitive Data Exposure

- **Secure Transformation of Data (SSL)**

  The Oracle Banking Liquidity Management allows HTTP connections over SSL/TLS. In other words, all HTTP traffic in the clear will be prohibited; only HTTPS traffic will be allowed. It is mandatory to enable this option in a production environment, especially when WebLogic Server acts as the SSL terminator.

  A two-way SSL is used when the server needs to authenticate the client. In a two-way SSL connection the client verifies the identity of the server and then passes its identity certificate to the server. The server then validates the identity certificate of the client before completing the SSL handshake.

  In order to establish a two-way SSL connection, need to have two certificates, one for the server and the other for client.This is required for de-centralized setup of application.

  For Oracle Banking Liquidity Management, need to configure a single connector. This connector is related to SSL/TLS communication between host or browser and the branch which uses two-way authentication.

  If the secure flag is set on a cookie, then browsers will not submit the cookie in any requests that use an unencrypted HTTP connection, thereby preventing the cookie from being trivially intercepted by an attacker monitoring network traffic.

  Below configuration has to be ensured in weblogic.xml within the deployed application ear.
  - Cookies are set with Http only as true
  - Cookie secure flag set to true
  - Cookie path to refer to deployed application

  <wls: session-descriptor>
  <wls: cookie-http-only>true</wls: cookie-http-only>
  </wls: session-descriptor>

  <wls: session-descriptor>
  <wls: cookie-secure>true</wls: cookie-secure>
  <wls: url-rewriting-enabled>false</wls: url-rewriting-enabled>
  </wls: session-descriptor>

```
<session-descriptor>
  <cookie-name>JSESSIONID</cookie-name>
  <cookie-path>/<DeployedApplicationPath></cookie-path>
  <cookie-http-only>true</cookie-http-only>
  <cookie-secure>true</cookie-secure>
  <url-rewriting-enabled>false</url-rewriting-enabled>
        </session-descriptor>
```

Always make sure Cookies are set with always Auth Flag enabled by default for WebLogic server.

2.6.1 Sign-On messages

Below table shows the general Sign-On messages which would be displayed to the user during invalid authentication.

Message	Explanation
User Authentication Failed	An incorrect user ID or password was entered.
User Status is Disabled. Please contact your System Administrator	The user profile has been disabled due to number of dormancy days allowed for the user has exceeded the dormancy days configured in the system.
User Status is Locked. Please contact your System Administrator	The user profile has been locked due to an excessive number of attempts to login, using an incorrect user ID or password. The number of attempts could have matched either the successive or cumulative number of login failures (configured for the system).

2.6.2 CACHE Control in Servlet and jsp

There are three basic HTTP response headers that prevent a page from being cached to disk. Different browsers handle them in slightly different ways, so they need to be used in combination to ensure all browsers do not cache the specific page. These headers are "Expires", "Pragma" and "Cache-control". In addition, these headers can either be sent directly by the server or placed in the HTML code as HTTP-EQUIV META tags within the HEAD section. The "Expire" header gives a date at which point the page should expire and no longer be cached. Internet Explorer supports a date of "0" for immediately and any negative number for already expired. The "Pragma: no-cache" header indicates that the page should not be cached.

2.6.3 Clickjacking/Frame-bursting

Oracle JET handles clickjacking/Frame-bursting attack. Oracle Banking Liquidity Management uses the X-Frame-Options HTTP response header to indicate whether or not abrowser should be allowed to render a page in a <frame> or <iframe>. This is used to avoid Clickjacking attacks, by ensuring that the content is not embedded into other sites.

2.7 Missing Function Level Access Control

It is likely that users working in the same department at the same level of hierarchy need to have similar user profiles. In such cases, you can define a Role Profile that includes access rights to the functions that are common to a group of users. A user can be linked to a Role Profile by which you give the user access rights to all the functions in the Role Profile.

Application level access has implemented via the Security Management System (SMS) module. SMS supports “ROLE BASED” access of Screens and different types of operations.
Oracle Banking Liquidity Management supports dual control methodology, wherein every operation performed has to be authorized by another user with the requisite rights.

2.8 Cross-Site Request Forgery (CSRF)

Oracle Banking Liquidity Management services are stateless. Oracle Banking Liquidity Management generates JWT upon successful authentication of the users. The generated token works to prevent CSRF.

2.9 Using Components with Known Vulnerabilities

Source code scanning done using the latest fortify to identify the sources code issue and will provide the proper fix for the reported issues.

3rd party libraries scanning for every release has been done to validate if any security issues rise for any of the components or not. Update the 3PL with latest security patch or upgraded to latest version.

2.10 Unvalidated Redirects and Forwards Network Security

Application uses 302 redirect wherever required. Oracle Banking Liquidity Management uses response.sendRedirect(newURL);

3. Securing API Services

Different applications deployed on disparate platforms and using different infrastructure need to be able to communicate and integrate seamlessly with Oracle Banking Liquidity Management in order to exchange data. The Oracle Banking Liquidity Management Service API Gateway will cater to these integration needs.

The integration needs supported by the Gateway can be broadly categorized from the perspective of the Gateway as follows:

- Inbound application integration: Used when any external system needs to add, modify or query information within Oracle Banking Liquidity Management.

- Outbound application integration: Used when any external system needs to be accessed for processing transactions within Oracle Banking Liquidity Management.

Development Security Guide
[May] [2020]
Version 14.4.0.0.0
Oracle Financial Services Software Limited
Oracle Park
Off Western Express Highway
Goregaon (East)
Mumbai, Maharashtra 400 063
India

Worldwide Inquiries:
Phone: +91 22 6718 3000
Fax:+91 22 6718 3001
www.oracle.com/financialservices/