BXH LEND Audit Report

Version 1.0.0

Serial No. 2021100700022017

Presented by Fairyproof

October 7, 2021

FAIRYPROOF

01. Introduction

This document includes the results of the audit performed by the Fairyproof team on the BXH LEND project, at the request of the BXH team.

Audit Start Time:

September 29, 2021

Audit End Time:

October 6, 2021

Audited Code's Github Repository:

https://github.com/BXHash/loan

Audited Code's Github Commit Number When Audit Started:

5958d6a82cb1d3f4455e85469be35409c46d241f

Audited Code's Github Commit Number When Audit Ended:

6d16c68e839f27da50c68cd92fcc9b11780d004c

Audited Source Files:

The calculated SHA-256 values for the audited files when the audit was done are as follows:

```
AggregatorV3Interface.sol:
0xb540bbbb6901ed3547431d588d550bdc35c24aafdf598c6772be34d16b765cd8

BaseJumpRateModelV2.sol:
0x89c79cd9f48ac1407d2ad568ed6dba985a09366a13e0b01bd9605f656283a704

CErc20.sol:
0xde587dc31531b39de4831d3ee05d00a3eae11fa4c1b7f999b71485765df824cb

CErc20Delegate.sol:
0x9e4f5b92705c66f910bd0c38600bede344b592f1655a07e63a6ecfad45275a3b

CErc20Delegator.sol:
0x525e15dac623328c8c5cc9591be4fc7b5af85fdb96496ac3569201b63c26614b

CEther.sol:
0x399c4c27f058a42c91f95b214c2804ca4aaf6e17718a0445c5dda9eaa070105b

CToken.sol:
```

0x86436e5db2868a1ef6a9d7963036d97b6773c6878a800fc58acd3c311c04911e

CTokenInterfaces.sol:
0xa9ea77a303ce6bf9e2b179574ce9968394db8c492704c2030b288029478073d1

CarefulMath.sol:
0xdcb5b6857f6455d1daf77feb84a4cd11d3fb191fbc8097315479e88308f89083

Comptroller.sol:
0x3d8aca836d4bd32406ff4a80f5098d5a9cecc397bfa7a959146a9c9d0713ecf7

ComptrollerInterface.sol:
0xffaa9223853b78666686c98662fd3ddbd9cf1971032674c4041aa946761b7fd7

ComptrollerStorage.sol:
0x1017244ef893145544cf2121153cd73caf67fa0139e3401c3f32f568ea8850ec

EIP20Interface.sol:
0xbc2ecd2927c202aab91222af287c07503cb348d8a96da3d368f195648356c4b7

EIP20NonStandardInterface.sol:
0x918d5790253d16e1b5221918d040399ad3598aec848b6a9007428965fe57e058

ErrorReporter.sol:
0xb8a2e67f14e1fdc7b1eaa2df734a23497220ea5051cf2ffae97456785b2658bf

Exponential.sol:
0x35cd0b89d935713f89f679190d92764519f5afeb08accec6f813f6b7a0db5f4e

ExponentialNoError.sol:
0x418ae000ba621eb3e8ef0e4f2347310f0c2e5f3bb75b183681d8bf67c7c14b11

InterestRateModel.sol:
0x8bba52751bf2ca58e1d47012d0879a69d73e49c3de841bee79e3dfb5387b2433

JumpRateModelV2.sol:
0x3c0a342bcce0fca28a0b460fc6a9c51c03eb4b3258c73140a14c0da8de242130

PriceOracle.sol:
0x8a5a574ee7b71ab417d5065cff4759ea32ce5c15f65e6e70fcbdd9a41d19c153

PriceOracleChainlink.sol:
0x35a8331da448be67816f3da94dfedccdfa2d3f6841bff1fe39cb88dc623ca905

SafeMath.sol:
0x204a19fb7a661c5bafcd5f7916254a457ca1fd9104e5708a73dd5010b11353dc

SimplePriceOracle.sol:
0xdaebe63435b50a636f65496d286461820909a3bc895166c70c49f775554c465b

```
Unitroller.sol:
0xa56f8cf884f0bceb918bbb078aaa5cd3ef90002323787729d70fdee6b4a1c602
```

The source files audited include all the files with the extension "sol" as follows:

```
contracts/
├── AggregatorV3Interface.sol
├── BaseJumpRateModelV2.sol
├── CErc20.sol
├── CErc20Delegate.sol
├── CErc20Delegator.sol
├── CEther.sol
├── CToken.sol
├── CTokenInterfaces.sol
├── CarefulMath.sol
├── Comptroller.sol
├── ComptrollerInterface.sol
├── ComptrollerStorage.sol
├── EIP20Interface.sol
├── EIP20NonStandardInterface.sol
├── ErrorReporter.sol
├── Exponential.sol
├── ExponentialNoError.sol
├── InterestRateModel.sol
├── JumpRateModelV2.sol
├── PriceOracle.sol
├── PriceOracleChainlink.sol
├── SafeMath.sol
├── SimplePriceOracle.sol
└── Unitroller.sol

24 files
```

The goal of this audit is to review BXH LEND’s solidity implementation for its decentralized lending application, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

We make observations on specific areas of the code that present concrete problems, as well as general observations that traverse the entire codebase horizontally, which could improve its quality as a whole.

This audit only applies to the specified code, software or any materials supplied by the BXH team for specified versions. Whenever the code, software, materials, settings, enviroment etc is changed, the comments of this audit will no longer apply.

— Disclaimer

Note that as of the date of publishing, the contents of this report reflect the current understanding of known security patterns and state of the art regarding system security. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk.

The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. If the audited source files are smart contract files, risks or issues introduced by using data feeds from offchain sources are not extended by this review either.

Given the size of the project, the findings detailed here are not to be considered exhaustive, and further testing and audit is recommended after the issues covered are fixed.

To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

— Methodology

The above files' code was studied in detail in order to acquire a clear impression of how the its specifications were implemented. The codebase was then subject to deep analysis and scrutiny, resulting in a series of observations. The problems and their potential solutions are discussed in this document and, whenever possible, we identify common sources for such problems and comment on them as well.

The Fairyproof auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Fairyproof to make sure we understand the size, scope, and functionality of the project's source code.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Fairyproof describe.
2. Testing and automated analysis that includes the following:

   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run the test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the source code to improve maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

— Structure of the document

This report contains a list of issues and comments on all the above source files. Each issue is assigned a severity level based on the potential impact of the issue and recommendations to fix it, if applicable. For ease of navigation, an index by topic and another by severity are both provided at the beginning of the report.

— Documentation

For this audit, we used the following sources of truth about how the decentralized lending application should work:

https://github.com/BXHash/loan

These were considered the specification, and when discrepancies arose with the actual code behavior, we consulted with the BXH team or reported an issue.

— Comments from Auditor

No vulnerabilities with critical, high, medium or low-severity were found in the above source code.

Additional notice: 0.

02. About Fairyproof

Fairyproof is a leading technology firm in the blockchain industry, providing consulting and security audits for organizations. Fairyproof has developed industry security standards for designing and deploying blockchain applications.

03. Introduction to BXH LEND

BXH LEND is a decentralized lending application.

04. Major functions of audited code

The audited code mainly implements the following functions:

- Depositing Assets: users can deposit crypto assets to gain interests
- Borrowing Assets: users can deposit assets as collateral to borrow crypto assets
- Liquidating Collateral: users can actively initiate liquidation of collateral and earn profits. Note: the application has a whitelist. When the whitelist is enabled, only the whitelisted addresses can initiate liquidation, otherwise any addresses can initiate liquidation.

05. Admin Rights

The Admin in this application has the following rights:

- managing daily operations
- enabling/disabling or setting a whitelist
- upgrading contracts
- setting an oracle and feeding prices

06. Key points in audit

During the audit we worked closely with the BXH team and mainly checked the following things:

- whether or not some core parameters were set properly,
- whether or not the whitelist worked properly,
- whether or not prices were calculated correctly and
- whether or not there were other possible vulnerabilities

07. Coverage of issues

The issues that the Fairyproof team covered when conducting the audit include but are not limited to the following ones:

- Re-entrancy Attack
- DDos Attack
- Integer Overflow
- Function Visibility
- Logic Vulnerability
- Uninitialized Storage Pointer
- Arithmetic Precision
- Tx.origin
- Shadow Variable
- Design Vulnerability
- Token Issurance
- Asset Security
- Access Control

08. Severity level reference

Every issue in this report was assigned a severity level from the following:

Critical severity issues need to be fixed as soon as possible.

High severity issues will probably bring problems and should be fixed.

Medium severity issues could potentially bring problems and should eventually be fixed.

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

09. List of issues by severity

A. Critical

- N/A

B. High

- N/A

C. Medium

- N/A

D. Low

- N/A

10. List of issues by source file

- N/A

11. Issue descriptions

- N/A

12. Recommendations to enhance the overall security

We list some recommendations in this section. They are not mandatory but will enhance the overall security of the system if they are adopted.

- auditing new contracts prior to contract upgrade/migration,
- validating prices prior to price feeding, and
- managing Admin's private key with great care, and transferring Admin's private key to a multi-sig wallet in a future upgrade