Audit Report

Increment Finance

v0.3

April 28, 2022

Table of Contents

License

Disclaimer

This audit has been performed by

Oak Security

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security has been engaged by Increment Labs Ltd. to perform a security audit of the IncrementFi-Oracle & IncrementFi-MoneyMarket smart contracts.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the following GitHub repositories:

https://github.com/IncrementFi/Money-Market

Commit hash: `25c25610763e9a3a8085e3214b2177d7a726c3c2`

https://github.com/IncrementFi/Oracle

Commit hash: `052fa352bc4e049da42026dda719b19bc1c87339`

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line by line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
4. Report preparation

Functionality Overview

The submitted code implements the smart contracts for Increment Finance, a liquidity engine on Flow blockchain. The protocol consists of a pool-based DeFi money market and a price oracle which serves to provide price feeds across the Flow ecosystem.

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending,** **Acknowledged**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Summary of Findings

No	Description	Severity	Status
1	Liquidation penalty is not validated during struct initialization	Minor	Resolved
2	Interest rate model capability is not validated during update	Minor	Resolved
3	Local resource reference can be used for efficiency	Minor	Acknowledged
4	Feeders can set non expirable prices by changing the expired duration to arbitrary high value	Minor	Acknowledged
5	Sorting algorithm can be improved	Informational	Acknowledged
6	Typographical errors found in codebase	Informational	Resolved
7	Possible chances of path collisions	Informational	Resolved

Code Quality Criteria

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	High	Efficiently written code, which makes it easy to read and understand.
Level of documentation	High	External facing documentation is well crafted and covers overview of business logic and mathematical equations. Code is well documented as well that makes it easier to understand its nuances.
Test coverage	Medium-High	-

Detailed Findings

1. Liquidation penalty is not validated during struct initialization

Severity: Minor

In `cadence/contracts/LendingComptroller.cdc:125-127` of the `MoneyMarket` repository, the liquidation penalty is not validated to be below 1.0 (ie. 100%) during the `Market` struct initialization. As a result, misconfiguring the liquidation penalty value might cause a liquidator to receive more funds than intended, causing a loss of funds for the borrower.

We consider this to be a minor issue since it can only be caused by the owner.

Recommendation

We recommend verifying the liquidation penalty value to be below 1.0 as seen in line 84.

Status: Resolved

2. Interest rate model capability is not validated during update

Severity: Minor

In `cadence/contracts/LendingPool.cdc:959-970` of the `MoneyMarket` repository, the newly set interest rate model capability is not validated to exist and can be borrowed. As a result, setting a non-existing capability for the interest rate model would cause several functions in the contract to fail, notably `accrueInterestReadonly`, `getPoolBorrowRateScaled`, `getPoolBorrowAprScaled` and `getPoolSupplyAprScaled`.

We consider this to be a minor issue since it can only be caused by the owner.

Recommendation

We recommend verifying the interest rate model capability to be valid as seen in line 1045.

Status: Resolved

3. Local resource reference can be used for efficiency

Severity: Minor

In `cadence/contracts/PriceOracle.cdc:128-130` of the `Oracle` repository, the `certificate` argument is passed as an `OracleCertificate` resource reference which is implemented by the `OracleInterface` contract interface. The `certificate` is then

verified by checking its type against the local `OracleCertificate` resource to make sure the caller can only be the current contract. This validation can be removed to introduce efficiency by only accepting the local `OracleCertificate` resource reference as an argument.

Recommendation

We recommend modifying the `certificate` to directly accept the local `OracleCertificate` resource reference and removing the unnecessary precondition check.

Status: Acknowledged

4. Feeders can set non expirable prices by changing the expired duration to arbitrary high value

Severity: Minor

In `cadence/contracts/PriceOracle.cdc:122-124` of the `Oracle` repository, feeders can set an arbitrary high expired duration which makes their prices non-expirable. With that said, this doesn't directly affect the prices of assets until more than half of the feeders act maliciously and set the expired duration to an arbitrarily high value.

We consider this to be a minor issue since this requires more than half of the feeders to be malicious.

Recommendation

We recommend setting the constant value of expired duration or using some upper and lower bounds for the expired duration value.

Status: Acknowledged

5. Sorting algorithm can be improved

Severity: Informational

In `cadence/contracts/PriceOracle.cdc:222` of the `Oracle` repository, the current sorting algorithm is inefficient due to time complexity. Since `takeMedianPrice` functionality will be called many times, this causes every other operation that is dependent on it to become gas inefficient.

Recommendation

We recommend implementing the merge sort algorithm for sorting.

Status: Acknowledged

6. Typographical errors found in codebase

Severity: Informational

In `cadence/contracts/PriceOracle.cdc` of the `Oracle` repository, there were several typographical errors found in lines 104 and 383. Specifically, the word “oralce_public” should be replaced with “oracle_public” while the word “_ExpriedDuration” should be replaced with “_ExpiredDuration”.

Recommendation

We recommend correcting the misspellings mentioned above.

Status: Resolved

7. Possible chances of path collisions

Severity: Informational

In the `MoneyMarket` repository, the following storage and public paths are very generic:

- `cadence/contracts/LendingPool:1086`
- `cadence/contracts/LendingPool:1088`
- `cadence/contracts/LendingPool:1089`

This might lead to path collisions, potentially incorrect data access or resources overwriting.

Recommendation

We recommend choosing a less generic path to avoid collisions.

Status: Resolved