CERTIK

Fixed Rate Protocol

Security Assessment

Mar 8th 2021

Summary

This report has been prepared for Fixed Rate Protocol smart contracts, a fixed-rate yield-generation protocol, to discover issues and vulnerabilities in the source code as well as any dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing static analysis and manual review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by security experts.

The security assessment resulted in 13 findings that ranged from minor to informational. We recommend addressing these findings as potential improvements that can benefit the long run as both smart contracts would lock a significant amount of FIX tokens for a significant amount of time. We have done rounds of communications over the general understanding and the Fixed team has resolved the questions promptly.

Overall the source code is well written with security practices. The business logic is comprehensive and implemented accordingly. Yet we suggest a few recommendations that could better serve the project from the security perspective:

1. Enhance general coding practices for better structures of source codes;
2. Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
3. Provide more comments per each function for readability, especially contracts that are verified in public.

Overview

Fixed is a fixed-rate yield-generation protocol that aims to pool stable assets into different money markets for gaining high yields, while also balancing the major two players (depositors and funders) to be compensated both by a fixed APY of the deposits and a comparatively fluctuated but earning-secured APY (unless the underlying APY massively dropped more than 25%).

DInterest

The entry point of the fixed protocol for users to interact with on things like deposit, withdraw and fund.

PercentageFeeModel

Protocol level on fee settings that would be charged against depositors when withdrawing the interests. It is currently hardcoded into 10% of the interests owed to the depositors.

MoneyMarkets

The wrapper contracts for interacting with underlying yield-generating protocols and where the user deposits are locked. Each interacting protocol would require a stand-alone instance. Currently there are three underlying yield protocols that DInterest interacts with: LendHubHRC20Market, FilDAHRC20Market and ChannelsHRC20Market. All three implementations are considered identical given the underlying dependencies shared the same interfaces and logics similar to typical protocols like Compound. Yielding protocols are not always bulletproof and third party dependencies can vary and may introduce unexpected events that result in assets lost.

EMAOracle

The oracles used for the interest rates of the underlying yield-generating protocols. Exponential Moving Average (EMA) method is implemented for the purpose of avoiding potential spikes of the underlying yielding protocols and make the indexes relatively stable over the time intervals.

FIXIssuanceModel01

The issuance model for releasing FIX token rewards to protocol participants. Multipliers at pool level are configurable by the owner of FIXMinter.

FIXToken

The standard ERC20 implementation token of FIX in reward of the protocol. Owner has the privilege to mint or burn tokens, and there is no max supply so circulating supply is more critical and should be determined by the protocol activities from end users.

FIXMinter

Handlings of minting & taking back FIX tokens for depositors & bond buyers. It handles the logics on proportional splits of tokens to gov and dev addresses.

Dumper

Responsible for accumulating the protocol fees & yield-farming rewards generated by FIX pools and exchanging them for stable coins.

Rewards

A fork of synthetix, the rewards staking contract for distributing the protocol fees & yield-farming rewards.

Vesting

Handles the FIX rewards in a specific vesting schedule defined by the FIXMinter.

NFT

Two non-fungible ERC721 based tokens, used for representing the ownerships of deposits and bonds. DInterest controls the issuance of NFTs to users.

Economic Model

Given an underlying money market that produces a varied APY, the protocol would provide depositors a fixed rate return opportunity while allowing funders using deposited money as leverages to earn a potentially high APY, but for funders high APY also comes with higher risks.

Workflow Example

1. Alice **deposited** 100 Lendhub Heco, with a maturity period of 365 days.
2. The protocol calculated the **interestOwned** on maturity date and recorded the deposit and the interestOwned as a **Deposit** entity, and appended it to a deposit queue.
3. Bob decided to **fund** the deficit and as a result own the right to claim returns gained by those deposits in the underlying money market. However the return is not guaranteed, it can be super high or be negative. Bob can choose to **fundAll** open deposits or **fundMultiple** just a few open deposits in front of the deposit queue. In order to fund Alice's deposit, Bob needs to pay amount = **interestOwned** in the simplified one-deposit-by-immediate-one-fund case.
4. Only Alice can start the withdraw
    a. If Alice withdraw **after** the maturity date
    Alice claim amount = **deposit** + 0.9 * **interestOwned** (the other 10% was charged as fee)
    Bob get amount = **interest(deposit + interestOwned)** when Alice finished this withdraw process (actually Bob will get all interest in the same funding which can include multiple deposits)
    b. If Alice withdraw **before** the maturity date
    Alice claim amount = **deposit**
    Bob get amount = **interest(deposit + interestOwned) + negativeSurplusForDeposit** (actually Bob will get all interest in the same funding which can include multiple deposits)

Pool Surplus Analysis

Withdraw Type	**Deposit Is Funded**	**Income Index**	**Pool Surplus Impact**
Early Withdraw	No	Up	+
Early Withdraw	No	Down	-
Early Withdraw	Yes	Up	=

Early Withdraw	Yes	Down	= (cannot withdraw)
Mature Withdraw	No	Up	+ / - (depends on up % - interest %)
Mature Withdraw	No	Down	-
Mature Withdraw	Yes	Up	=
Mature Withdraw	Yes	Down	= (cannot withdraw)

FIX Tokenomics

Supply

Event	depositorReward	devReward	govReward	funderReward	totalSupplyChange
Deposit: n	n * depositMultiplier	0	0	0	+ n * depositMultiplier
Early Withdraw: n	- n * depositMultiplier (takeback)	0	0	0	- n * depositMultiplier
Mature Withdraw: n	- n * depositMultiplier * takebackMultiplier (takeback)	n * depositMultiplier * takebackMultiplier * devRewardMultiplier + n * poolFunderRewardMultiplier * (maturationTimestamp - fundingCreationTimestamp) * devRewardMultiplier	n * depositMultiplier * takeback Multiplier	n * poolFunderRewardMultiplier * (maturationTimestamp - fundingCreationTimestamp)	+ n * depositMultiplier * takebackMultiplier * devRewardMultiplier + n * poolFunderRewardMultiplier * (maturationTimestamp - fundingCreationTimestamp) * (1 + devRewardMultiplier)

Vesting

Deposit Period	Vesting Period	Vesting Schedule
0.5 * poolVestingPeriod	0.5 * poolVestingPeriod	Linear
1 * poolVestingPeriod	poolVestingPeriod	Linear
2 * poolVestingPeriod	poolVestingPeriod	Linear

Findings

ID	Severity	Response
CTK-FIXED-1	Minor	Acknowledged
CTK-FIXED-2	Informational	Acknowledged
CTK-FIXED-3	Informational	Acknowledged
CTK-FIXED-4	Informational	Acknowledged
CTK-FIXED-5	Informational	Acknowledged
CTK-FIXED-6	Informational	Resolved
CTK-FIXED-7	Informational	Resolved
CTK-FIXED-8	Informational	Acknowledged
CTK-FIXED-9	Informational	Resolved
CTK-FIXED-10	Informational	Acknowledged
CTK-FIXED-11	Informational	Acknowledged
CTK-FIXED-12	Informational	Resolved
CTK-FIXED-13	Informational	Acknowledged
CTK-FIXED-14	Informational	Acknowledged

CTK-FIXED-1 | Depositing Investor Unable to Withdraw Funded Deposit When Income Index Dropped

Type	Severity	Location
Volatile Code	Minor	DInterest: L827

Description

Withdraw requires the underlying money market to have a higher income index than the time deposit happens. If the underlying money market declines, depositing investors will be unable to withdraw their money.

```
# DInterest.sol line 827
function _payInterestToFunder(

 # the following sub method throws & revert tx
 # when currentMoneyMarketIncomeIndex is lower
 # than recordedMoneyMarketIncomeIndex

 uint256 interestAmount =
           f
              .recordedFundedDepositAmount
              .mul(currentMoneyMarketIncomeIndex)
              .div(f.recordedMoneyMarketIncomeIndex)
              .sub(f.recordedFundedDepositAmount);

# @openzeppelin/contracts/math/SafeMath.sol
function sub(uint256 a, uint256 b) internal pure returns (uint256) {
       require(b <= a, "SafeMath: subtraction overflow");
       return a - b;
}
```

Recommendation

1. Use `trySub` instead of `sub`
2. Advice depositing investors to do their own research for the underlying money market before investing

CTK-FIXED-2 | Funding Investor Unable to Get Return

Type	Severity	Location
Business Model	Informational	DInterest

Description

Because funding investors cannot trigger a withdrawal by themselves, they cannot control when and even how much interest they can earn(their return is related to the ever-changing money market income index). In some edge cases, if depositors were unable to withdraw (e.g. lost key), funding investors will also be unable to get their interest.

Recommendation

Consider adding a function to allow the funding investor to collect returns for deposits that have passed maturity dates (if they're willing to pay the takeback FIX amount).

CTK-FIXED-3 | DDos via Money Market Price Manipulation

Type	Severity	Location
Business Model	Informational	DInterest

Description

If the money market income index keeps dropping, the protocol would become very brittle with a lot of potential deficit and almost no liquidity. The brittle may be utilized by some form of attacks.

Recommendation

Design a method to prevent further losses when the worst case happens.

CTK-FIXED-4 | Centralized Governance

Type	Severity	Location
Business Model	Informational	FIXMinter

Description

FIXMinter contract owners have strong power and can control the addresses for `devReward` and `governanceReward`. The damage can be huge if the owner account is ever lost or hacked.

Recommendation

Renounce ownership when it is the right timing; or gradually migrate to a timelock plus multisig governing procedure and let the community to monitor in respect of transparency considerations.

CTK-FIXED-5 | Many Fee Types

Type	Severity	Location
Business Model	Informational	FIXToken

Description

In the protocol comes 3 types of fees: devReward FIX token, governanceReward FIX token, fee. It is not super clear how the fees will eventually be used to build the community.

Recommendation

Consider lower or even eliminate the 10% fee to make the return rate even more competitive and make the protocol more decentralized.

CTK-FIXED-6 | Uninitialized local variable

Type	Severity	Location
Volatile Code	Informational	DInterest: L772

Description

In `_withdraw()`, the local variable `feeAmount` will be never initialized if early is false. However, it is used to calculate the `withdrawalAmount` and used as `feeAmount` in the later transfer of stablecoin.

Recommendation

Recommend initializing the local variable either inside the if statement when `early == true` or before the if checks.

Alleviation

The team has assigned the initial value to the `feeAmount`.

CTK-FIXED-7 | State variables are possible to be address(0)

Type	Severity	Location
Volatile Code	Informational	FIXMinter: L64, 65

Description

State variables `govTreasury` and `devWallet` are addresses set in `constructor()`. No checks to make sure that these two addresses should never be `address(0)` in the code implementation.

Recommendation

In terms of avoiding human factor errors, please make sure when constructing the contract, these two parameters are not `address(0)`.

Alleviation

The team has updated the function to check against invalid addresses.

```
require(_govTreasury != address(0), "FIXMinter: govTreasury 0 address");
require(_devWallet != address(0), "FIXMinter: devWallet 0 address");
```

CTK-FIXED-8 | Unused variables

Type	Severity	Location
Dead Code	Informational	DInterest: L29, 30 FIXIssuanceModel01: L85, 151

Description

In `DInterest`:

- constant variable `PRECISION` is never used.
- constant variable `ONE` is never used.

In `FIXIssuanceModel01`:

- function `computeDepositorReward()` takes four parameters but `interestAmount` is never used.
- function `computeFunderReward()` takes six parameters but `interestPayoutAmount` is never used.

Recommendation

Recommend removing unused code to reduce gas consumption in deployment.

CTK-FIXED-9 | Function can be declared as `external`

Type	Severity	Location
Volatile Code	Informational	FIXToken

Description

Functions `init()` and `ownerMint()`, are not called internally and inherited-ly but not declared as `external`.

Recommendation

In this case these two functions can be declared as `external` to save gas.

Alleviation

The team has updated the function types to be `external`.

CTK-FIXED-10 | No limits on Privileged Activities

Type	Severity	Location
Business Model	Informational	FIXIssuanceModel01

Description

The issuance model contract has the owner-only functions that handle the settings of a list of multipliers, i.e. `setPoolFunderRewardMultiplier` and `setDevRewardMultiplier`; and vesting period changes via `setPoolFunderRewardVestPeriod`. We understand that these are essential from the perspective of protocol governance. However, currently there are no so-called `reasonable` `require` statements applied so that the new settings are within a range of numbers.

Recommendation

Consider applying a series of `require` statements and fill the multiplier number within a certain range, say an upper bound of 15% for dev rewards.

CTK-FIXED-11 | Centralized Mintable Token

Type	Severity	Location
Business Model	Informational	FIXToken

Description

The `FIXToken` is the standard implementation of ERC20 with additional features on mint and burn, which introduced some levels of decentralization concerns.

Recommendation

We understand that from a business perspective the FIX token is inevitably in need of such functionalities given its token metrics are not predefined, in a way there is no max supply but a potentially ascending circulating supply. We encourage the team to have more transparency on privileged activities once the protocol is live.

CTK-FIXED-12 | Unused Event

Type	Severity	Location
Dead Code	Informational	FIXIssuanceModel01

Description

There is an event that is only defined but not used anywhere within the contract.

```
    event ESetParamAddress(
        address indexed sender,
        string indexed paramName,
        address newValue
    );
```

Recommendation

Consider to delete the event definition to avoid potential misunderstandings when open sourced for the public to review; or revisit the contract implementation in case there is any missing feature.

Alleviation

The team has removed the unused event.

CTK-FIXED-13 | 3rd Party Dependencies

Type	Severity	Location
Business Model	Informational	moneyMarkets

Description

MoneyMarkets is serving as the underlying investment entity to interact with third party yield protocols by depositing assets in return for rewards. The scope of the audit would treat those 3rd party entities as black boxes and assume its functional correctness. However in the real world, 3rd parties may be compromised that led to assets lost or stolen.

Recommendation

We understand that the business logics of Fixed Rate Protocol require the interaction with yield protocols for the sake of pursuing capital gains of its users. We encourage the team to constantly monitor the statuses of those 3rd parties to mitigate the side effects when unexpected activities are observed.

CTK-FIXED-14 | Centralized DInterest

Type	Severity	Location
Business Model	Informational	DInterest

Description

Majority of the protocol parameters and settings are configured by the owner role, which as results could lead to the potential financial imbalance of the protocol. We believe that the team holds the goodwill to develop and maintain the protocol, yet there is still a chance for a leaked owner key with unexpected side effects.

Recommendation

It is recommended that to give back the power once the protocol reaches a healthy stage of community goverancence. Consider introducing a multisig practice on the owner role and apply a timelock to allow buffer time and enough transparency to the end users.

Appendix | Finding Categories

Gas Optimization

Refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Refer to exhibits that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Refer to exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

Control Flow

Concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Describe faults in the way data is handled at rest and in memory, such as the result of a `struct` assignment operation affecting an in-memory `struct` rather than an instorage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of `private` or `delete`.

Coding Style

Usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Refer to functions that should seemingly behave similarly yet contain different code, such as a `constructor` assignment imposing different `require` statements on the input variables than a setter function.

Magic Numbers

Refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as `constant` contract variables aiding in their legibility and maintainability.

Compiler Error

Refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Dead Code

Code that otherwise does not affect the functionality of the codebase and can be safely omitted.

Business Model

Refer to contract or function logics that are debatable or not clearly implemented according to the design intentions.

Disclaimer

About CertiK

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

CERTIK

Provable Trust For All