Rico Credit System Security Review

Reviewers

Kurt Barry (Fixed Point Solutions, LLC)

Report prepared by: Kurt Barry

January 9, 2024

Contents

1 About The Review Team

Fixed Point Solutions, LLC is a provider of Web3 security reviews and other bespoke technical consulting services.

2 Introduction

Rico is a protocol for decentralized lending with debt denominated in a stable asset created by the system against various types of collaterals. It refines and extends the Multi-Collateral DAI system, introducing a number of novel mechanisms and concepts.

Two cumulative weeks of research effort by a single security researcher were devoted to the review.

Disclaimer: No assessment can guarantee the absolute safety or security of a software-based system. Further, a system can become unsafe or insecure over time as it and/or its environment evolves. This assessment aimed to discover as many issues and make as many suggestions for improvement as possible within the specified timeframe. Undiscovered issues, even serious ones, may remain. Issues may also exist in components and dependencies not included in the assessment scope.

3 Risk classification

Severity level	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

3.1 Impact

- High - leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
- Medium - global losses <10% or losses to only a subset of users, but still unacceptable.
- Low - losses will be annoying but bearable--applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.

3.2 Likelihood

- High - almost certain to happen, easy to perform, or not easy but highly incentivized
- Medium - only conditionally possible or incentivized, but still relatively likely
- Low - requires stars to align, or little-to-no incentive

3.3 Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

4 Executive Summary

Over the course of 10 days in total, Halys, Inc. engaged with Fixed Point Solutions, LLC to review Rico Credit System. In this period of time a total of 5 issues were found.

In addition to the officially locked commits for each repo, a number of fixes were reiviewed, which are noted as need in the write-ups for the issues they addressed. Changes were also made to the pricing mechanism for sales of collateral, credit token surplus, and risk tokens during the course of the audit, which were assessed at commit 0114e99c310b13a9d68b6ef131b10cacc151d6cc in the ricobank/ricobank repository.

Summary

Project Name	Rico Credit System
Repository	ricobank/ricobank
Commit	f1da34bee8bccb2651107f9840b3ff2b60562db9
Files	src/diamond.sol src/bank.sol src/file.sol src/vat.sol src/vow.sol src/vox.sol src/mixin/flog.sol src/mixin/math.sol src/hook/hook.sol src/hook/erc20/ERC20Hook.sol src/hook/nfpm/UniV3NFTHook.sol src/hook/nfpm/interfaces/INonfungiblePositionManager.sol
Repository	ricobank/gemfab
Commit	9d7261ede3070a75fb61097f28206954bd9888f5
Files	src/gem.sol
Repository	feedbase/feedbase
Commit	c4e307dbd6df4f382b703d84dc7046d42a687c8e
Files	src/Feedbase.sol src/adapters/ChainlinkAdapter.sol src/adapters/UniswapV3Adapter.sol src/adapters/UniswapWrapper.sol src/combinators/Divider.sol src/combinators/Multiplier.sol src/mixin/Read.sol src/mixin/ward.sol
Type of Project	Lending and Stablecoin, DeFi
Audit Timeline	Nov 27th - Dec 11th
Methods	Manual Review

Issues Found

Critical Risk	0
High Risk	1
Medium Risk	0
Low Risk	1
Gas Optimizations	2
Informational	1
Total Issues	5

5 Findings

5.1 High Risk

5.1.1 Uniswap V3 Position Pricing Vulnerable to Manipulation

Context: UniV3NFTHook.sol#L164

Description: The logic for pricing Uniswap V3 position NFTs relies on the current spot price of the pool. This is vulnerable to manipulation, especially via flashloans. The mathematics of constant product curves generally result in higher represented values for positions under such manipulation, which could allow more funds to be borrowed than intended, circumventing collateral requirements and potentially creating bad debt.

By way of example, consider a pool with a single position which is full-range (i.e. equivalent to a V2-style pool), with reserves `X` and `Y` and product `K`. Let `p` be the price of the first reserve and `q` be the price of the second reserve in some numeraire. Note that in equilibrium, `p*X = q*Y` and the value of the position is `V := p*X + q*Y`. Let `D` units of the first reserve be sold into the pool (note: fees neglected for simplicity). The new first reserve amount is `X' = X + D`, and the new second reserve amount is `Y' = K / (X + D)`. The code would calculate the value of the position using the oracle-reported prices `p` and `q` as: `p*X' + q*Y' = p*X + p*D + q*K/(X + D)`. Using `K = X*Y`, we get `p*X + p*D + q*Y/(1+D/X) = p*X + p*D + q*Y - q*Y*/(X/D+1) = (p*X + q*Y) + (p*X + p*D - q*Y)/(X/D + 1)`. Using `p*X = q*Y` and `V = p*X + q*Y`, we get `V + p*D/(X/D + 1)`. Hence we see that the value of the "manipulated" position is strictly greater than the true value at oracle prices `p` and `q`.

Recommendation: Use the feed prices to compute the "true" `sqrtPriceX96` for the pool, and leverage the Uniswap libraries to deduce fair reserve values for the position.

Rico: Fixed in commit f1da34bee8bccb2651107f9840b3ff2b60562db9.

FPS: Fix appears logically correct, but consider reverting when `uint(val0) > MAX_SHIFT` as in this case using the upper end of the range may not be the correct behavior (`uint(val1)` could be similarly large or larger) and such a high value is likely to be erroneous anyway.

Rico: Downscaled prices and added a revert is the downscaled price of reserve zero still exceeds `MAX_SHIFT` in commit b4349802bab34fd58a87e64bfa8a02df52d05e56.

FPS: Additional change looks good, addresses concern around behavior when the 96 bit shift would overflow.

5.2 Low Risk

5.2.1 ERC20 Hook Will Not Work With Common Non-Standard Token Behaviors

Context: ERC20Hook.sol#L123

Description: Some common tokens are not fully ERC20-compliant. For example, USDT does not return a boolean value on transfers, but the `gem` interface expects this, which will cause reverts as Solidity will check for the presence of a return value.

Recommendation: Review common non-standard behaviors (see e.g. https://github.com/d-xo/weird-erc20) and decide which ones, if any, the base ERC20 hook should support and explicitly document this in collateral onboarding guidelines, making changes to the code if necessary.

Rico: The ERC20 hook is only intended to handle tokens which revert or return false on failed transfer.

FPS: Response noted.

5.3 Gas Optimization

5.3.1 UniswapV3Adapter.Config Struct Members Could Be Change to Reduce Storage Operations

Context: UniswapV3Adapter.sol#L21-L26

Description: The `UniswapV3Adapter.Config` struct is defined as follows in the audited commit:

```
    struct Config {
        address pool;    // univ3 pool address
        uint    range;   // twap window size
        uint    ttl;     // how much to advance ttl from uni twap output ttl
        bool    reverse; // true if token0 > token1
    }
```

This ordering of members results in four storage slots being used, meaning four `SLOAD` instructions to retrieve all the values in `read()`. At a minimum, the `pool` and `reverse` fields can be made adjacent so that they pack into a single slot. Further, `range` is only used after a cast to `uint32`, so if it were made such in the struct, `pool`, `reverse`, and `range` could all pack into a single slot, eliminating two `SLOAD` operations.

Recommendation: The cost of oracle reads can add up; consider making this optimization.

Rico: Struct reordered in commit ed0c77074e6278dc8999d026ed481b24c138c60e; not changing width of the `range` field at this time. New struct consumed in commit c40a960d8eceb5889fd23262dc93cbbd07ae90e4.

FPS: Fix verified.

5.3.2 Minor Gas Optimizations

Context: [1] ERC20Hook.sol.sol#L89 [2] vow.sol#L62 [3] UniV3NFTHook.sol#L137 [4] UniV3NFTHook.sol#L94

Description: [1] This line is safe from underflow by construction and could be `unchecked`. [2] The `toll` parameter is only ever used as `RAY - toll`; it could be encoded as a parameter that does not require a subtraction to be performed at every use. [3] This line is safe from underflow by construction and could be `unchecked`. [4] The suggestion in this TODO is sound and would save gas in cases where the check is passed.

Recommendation: If desired, these gas optimizations can be implemented with no security or safety tradeoff.

Rico: Suggestions implemented in commit 237411be608dc43048d5c6873f499870da3b7965 and commit 951167b1070d821fc782a65561da70ede2f8f8c4.

FPS: Changes verified.

5.4 Informational

5.4.1 Hooks Do Not Explicitly Account For Token Decimals

Context: ERC20Hook.sol#L107, UniV3NFTHook.sol#L186-L193

Description: Neither the ERC20 nor UniV3 hook explicitly accounts for tokens that may have non-18 decimals, such as USDC. The feed prices may take this into account, providing adjusted values that make the hook logic correct, but it's not clear whether this is the case or not.

Recommendation: If the feed prices will have a uniform precision, account for token decimals in the hook contracts. Otherwise, document that feeds will be normalizing values based on decimals. In this latter case, take extra care if multiple prices are ever combined in a way that doesn't involve multiplication by token amounts as the initial step, as explicit decimal adjustments may be necessary.

Rico: The feeds will normalize prices to account for different token decimals.

FPS: Noted.