Felix Protocol Audit Report

Prepared by 0xSimao

Version 2.0

November 30, 2025

Contents

1 About 0xSimao

0xSimao is an independent security researcher #2 on Sherlock, top-ranked on Cantina and Code4rena, and a member of Blackthorn, a leading auditing firm. Previously, he served as Head of Security at Three Sigma.

0xSimao has placed in the Top 3 in 28 public audits and has led over 60 private engagements. For private audits or collaboration opportunities, feel free to reach out to him on X (@0xSimao), Telegram (@OxSimao), or Discord (@0xSimao).

2 Disclaimer

0xSimao makes every effort to find as many vulnerabilities in the code as possible in the given time but holds no responsibility for the findings in this document. A security audit by the team does not endorse the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the solidity implementation of the contracts.

3 Risk Classification

	Impact: High	**Impact: Medium**	**Impact: Low**
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

4 Protocol Summary

Felix is a CDP (Collateralized Debt Position) stablecoin protocol. With Felix, users can permissionlessly use various tokens as collateral to mint feUSD, a stablecoin pegged to the U.S. dollar. feUSD can then be used across other applications or staked with the Felix protocol via Vaults

5 Audit Scope

The scope is the new Redstone price feed.

6 Executive Summary

Over the course of 2 days, 0xSimao conducted an audit on the Felix Protocol smart contracts provided by Felix Protocol. In this period, a total of 7 issues were found.

Summary

Project Name	Felix Protocol
Repository	felix-contracts
Commit	3312d7a0e5fb...
Fix Commit	9f2906b58f9f...
Audit Timeline	Mar 14th - Mar 17th
Methods	Manual Review, Stateful Fuzzing

Issues Found

Critical Risk	0
High Risk	0
Medium Risk	1
Low Risk	0
Informational	6
Gas Optimizations	0
Total Issues	7

Summary of Findings

[M-1] Attacker can trigger temporary shutdown due to RedStonePriceFeedBase missing gas check	Resolved
[I-1] Missing provideToSpOnBehalfOf interface	Acknowledged
[I-2] In case of success, some leftover funds could still be present in the adapter	Resolved
[I-3] Missing priceFeedDisabled event in RedStonePriceFeedBase	Resolved
[I-4] Gas forwarded by the CurveGaugeDistributor could be limited to further prevent OOG	Acknowledged
[I-5] The check rewardSelector != ZERO BYTES4 rewardDestination.isContract() could be improved	Resolved
[I-6] InterestRouterV2::triggerDistribution() may be vulnerable to arbitrage stratregies due to being permissionless	Resolved

7 Findings

7.1 Medium Risk

7.1.1 Attacker can trigger temporary shutdown due to RedStonePriceFeedBase missing gas check

Description: RedStonePriceFeedBase::_getCurrentRedStoneResponse() does not check gas left in the catch block, allowing attackers to forward just a bit of gas to make the call to chainlink revert and trigger a temporary shutdown.

Impact: Protocol DoS.

Recommended Mitigation: See the LiquityV2 check here.

0xSimao:

Fixed in #a0041e1.

7.2 Informational

7.2.1 Missing provideToSpOnBehalfOf interface

Description: `StabilityPool::provideToSpOnBehalfOf()` is missing the declaration in the `IStabilityPool` interface.

Recommended Mitigation: Add the function to the interface and override in the implementation.

7.2.2 In case of success, some leftover funds could still be present in the adapter

Description: The `CurveGaugeDistributor::distributeRewardsToGauge()` sends the funds to curve to deposit rewards, but does not check if there is any leftover balance after doing so. Currently curve pulls all tokens so this should not happen, but other adapters could have leftover funds.

Recommended Mitigation: Check if there are leftover funds after success and send them to the `InterestRouter` or add a helper to send any `feUSD` in the adapter to the `InterestRouter`.

0xSimao:

Fixed in #ba81f93.

7.2.3 Missing priceFeedDisabled event in RedStonePriceFeedBase

Description: Missing `priceFeedDisabled` event in `RedStonePriceFeedBase`.

Recommended Mitigation: Emit an event when `priceFeedDisabled` is set to true.

0xSimao:

Fixed in #b5e5524.

7.2.4 Gas forwarded by the CurveGaugeDistributor could be limited to further prevent OOG

Description: `CurveGaugeDistributor::distributeRewardsToGauge()` calls the external contract without limiting the gas sent, which could lead to OOG in case it becomes malicious.

Recommended Mitigation: To prevent this, ExcessivelySafeCall can be used.

7.2.5 The check rewardSelector != ZERO BYTES4 andand rewardDestination.isContract() could be improved

Description: When the destination reward is a contract and the selector is not 0, the function is called on the contract. However, there seems to be no logical scenario to have a reward selector different than 0 and the reward destinaton not being a contract. In case this happens by mistake, rewards could be sent to the destination contract without calling the function, which could be problematic.

Recommended Mitigation: It would be better to either revert in this case or perform the contract check when setting the reward selector.

0xSimao:

Fixed in #0d6233e.

7.2.6 InterestRouterV2::triggerDistribution() may be vulnerable to arbitrage stratregies due to being permissionless

Description: `InterestRouterV2::triggerDistribution()` is permissionless and may add significant rewards to the external contracts. In this process, arbitragers could employ arbitrage strategies, such as depositing, calling `InterestRouterV2::triggerDistribution()` and then withdrawing, stealing most of the rewards.

Recommended Mitigation: While curve gauges are not vulnerable to this as rewards are distributed over 1 week, it is something to keep in mind, even more so because other strategies could be used in the future that are vulnerable.

0xSimao:

Fixed in #9f2906b.