Fei Protocol contest
Findings & Analysis Report

2022-01-25

Table of contents

Overview

About C4

Code4rena (C4) is an open organization consisting of security researchers, auditors, developers, and individuals with domain expertise in smart contracts.

A C4 code contest is an event in which community participants, referred to as Wardens, review, audit, or analyze smart contract logic in exchange for a bounty provided by sponsoring projects.

During the code contest outlined in this document, C4 conducted an analysis of Fei Protocol contest smart contract system written in Solidity. The code contest took place between November 30—December 2 2021.

Wardens

22 Wardens contributed reports to the Fei Protocol contest:

1. WatchPug (jtp and ming)
2. danb
3. cmichel
4. Meta0xNull
5. loop
6. 0x0x0x
7. defsec
8. gzeon
9. hickuphh3
10. TomFrenchBlockchain
11. Czar102
12. tqts
13. jayjonah8
14. robee
15. GeekyLumberjack
16. ye0lde
17. jierlich
18. egjlmn1
19. 0x1f8b
20. sabtikw
21. hagrid

This contest was judged by pauliax.

Final report assembled by moneylegobatman and CloudEllie.

Summary

The C4 analysis yielded an aggregated total of 9 unique vulnerabilities and 48 total findings. All of the issues presented here are linked back to their original finding.

Of these vulnerabilities, 0 received a risk rating in the category of HIGH severity, 0 received a risk rating in the category of MEDIUM severity, and 9 received a risk rating in the category of LOW severity.

C4 analysis also identified 14 non-critical recommendations and 25 gas optimizations.

Scope

The code under review can be found within the C4 Fei Protocol contest repository, and is composed of 4 smart contracts written in the Solidity programming language and includes 342 lines of Solidity code.

Severity Criteria

C4 assesses the severity of disclosed vulnerabilities according to a methodology based on OWASP standards.

Vulnerabilities are divided into three primary risk categories: high, medium, and low.

High-level considerations for vulnerabilities span the following key areas when conducting assessments:

- Malicious Input Handling
- Escalation of privileges
- Arithmetic
- Gas use

Further information regarding the severity criteria referenced throughout the submission review process, please refer to the documentation provided on the C4 website.

Low Risk Findings (9)

- [L-01] unsafe cast *Submitted by danb.*
- [L-02] denial of service *Submitted by danb.*
- [L-03] PegExchanger expiry block must be set to at least `MIN_EXPIRY_WINDOW + 1` into the future *Submitted by loop.*
- [L-04] Wrong `preMergeCirculatingTribe` value *Submitted by cmichel.*
- [L-05] `TribeRagequit.sol` minter role to FEI is unnecessary *Submitted by WatchPug.*
- [L-06] `PegExchanger.sol` unused tribe tokens can be frozen in the contract *Submitted by WatchPug.*
- [L-07] In TRIBERagequit.sol, users can get frontrunned *Submitted by 0x0x0x, also found by cmichel.*
- [L-08] Value of token1OutBase might became stale in TRIBERagequit.sol *Submitted by gzeon, also found by Meta0xNull and defsec.*
- [L-09] Ragequit function ngmi() Will Fail Even If Follow All Steps in Simulations *Submitted by Meta0xNull.*

Non-Critical Findings (14)

- [N-01] Open TODOs *Submitted by robee, also found by Meta0xNull and hagrid.*
- [N-02] Require with not comprehensive message *Submitted by robee.*
- [N-03] Outdated compiler version *Submitted by WatchPug.*
- [N-04] Inaccurate revert reason in TRIBERagequit.sol *Submitted by gzeon.*
- [N-05] False information given to the user *Submitted by Czar102.*
- [N-06] Missing events for critical operations *Submitted by WatchPug, also found by 0x0x0x.*
- [N-07] Wrong comments about key in TRIBERagequit *Submitted by 0x0x0x.*
- [N-08] Expiration time shift *Submitted by Czar102, also found by cmichel.*

- [N-09] Code Style: constants should be named in all caps *Submitted by WatchPug.*
- [N-10] Improve readability of constants *Submitted by WatchPug.*
- [N-11] Consider change some constants into immutable variables for settings that can be configured at deploy time *Submitted by WatchPug, also found by 0x1f8b.*
- [N-12] Unnatural interface *Submitted by Czar102.*
- [N-13] TRIBERagequit: Make verifyClaim() public *Submitted by hickuphh3.*
- [N-14] No restriction for expiration block in TRIBERagequit.sol *Submitted by 0x0x0x, also found by Czar102.*

Gas Optimizations (25)

- [G-01] `PegExchanger.sol#exchange()` Redundant code *Submitted by WatchPug, also found by Czar102, defsec, and jierlich.*
- [G-02] PegExchanger#giveTo(): Use transfer() method instead of transferFrom() *Submitted by hickuphh3, also found by Czar102, WatchPug, and danb.*
- [G-03] Gas saving in ngmi(uint256,uint256,bytes32[]) *Submitted by tqts.*
- [G-04] Use short reason strings can save gas *Submitted by WatchPug, also found by Meta0xNull.*
- [G-05] Don’t cache bool `check` *Submitted by 0x0x0x, also found by gzeon, jierlich, and loop.*
- [G-06] Use else if can save gas *Submitted by WatchPug.*
- [G-07] Loops can be implemented more efficiently *Submitted by 0x0x0x, also found by Meta0xNull and WatchPug.*
- [G-08] `preMergeCirculatingTribe` can be constant *Submitted by loop, also found by 0x0x0x, Czar102, WatchPug, robee, and yeOlde.*
- [G-09] Unused local variables in requery (TRIBERagequit.sol) *Submitted by yeOlde, also found by Czar102, GeekyLumberjack, gzeon, loop, and robee.*
- [G-10] Comparison with literal boolean values *Submitted by yeOlde, also found by Czar102, 0x0x0x, 0x1f8b, WatchPug, and loop.*
- [G-11] constructor should be removed if not used *Submitted by jayjonah8.*

- [G-12] TRIBERageQuit: Redundant oracleAddress variable *Submitted by hickuphh3.*
- [G-13] Use `calldata` instead of `memory` for function parameters *Submitted by defsec.*
- [G-14] Testing for initial condition on oracle query last saves gas. *Submitted by TomFrenchBlockchain.*
- [G-15] Gas Optimization: Unchecked safe logic in TRIBERagequit.sol *Submitted by gzeon, also found by WatchPug and defsec.*
- [G-16] Assignment Of State Variables To Default *Submitted by yeOlde, also found by 0x0x0x, Czar102, TomFrenchBlockchain, WatchPug, and sabtikw.*
- [G-17] Public functions to external *Submitted by robee, also found by cmichel, jayjonah8, 0x0x0x, Czar102, GeekyLumberjack, WatchPug, defsec, loop, and tqts.*
- [G-18] Internal functions to private *Submitted by robee, also found by WatchPug.*
- [G-19] `++i` is more efficient than `i++` *Submitted by WatchPug, also found by defsec.*
- [G-20] Avoid On Chain Computation That Have Known Answer to Save Gas *Submitted by Meta0xNull, also found by WatchPug.*
- [G-21] Gas Optimization On The 2^256-1 *Submitted by defsec, also found by WatchPug.*
- [G-22] Remove unnecessary variables can make the code simpler and save some gas *Submitted by WatchPug, also found by Czar102 and defsec.*
- [G-23] Storage double reading. Could save SLOAD *Submitted by robee, also found by WatchPug and egjlmn1.*
- [G-24] Not used return value at recalculate and requery in TRIBERagequit.sol *Submitted by 0x0x0x, also found by GeekyLumberjack and WatchPug.*
- [G-25] Remove unnecessary function can make the code simpler and save some gas *Submitted by WatchPug, also found by gzeon.*

Disclosures

C4 is an open organization governed by participants in the community.

C4 Contests incentivize the discovery of exploits, vulnerabilities, and bugs in smart contracts. Security researchers are rewarded at an increasing rate for finding higher-risk issues. Contest submissions are judged by a knowledgeable security researcher and solidity developer and disclosed to sponsoring developers. C4 does not conduct formal verification regarding the provided code but instead provides final verification.

C4 does not provide any guarantee or warranty regarding the security of this project. All smart contract software should be used at the sole risk and responsibility of users.

Top