fMoney

Security Review

Jan 31, 2025

Conducted by:
Blckhv, Lead Security Researcher
Slavcheww, Lead Security Researcher

Contents

1. About SBSecurity

SBSecurity is a duo of skilled smart contract security researchers. Based on the audits conducted and numerous vulnerabilities reported, we strive to provide the absolute best security service and client satisfaction. While it's understood that 100% security and bug-free code cannot be guaranteed by anyone, we are committed to giving our utmost to provide the best possible outcome for you and your product.

Book a Security Review with us at sbsecurity.net or reach out on Twitter @Slavcheww.

2. Disclaimer

A smart contract security review can only show the presence of vulnerabilities **but not their absence**. Audits are a time, resource, and expertise-bound effort where skilled technicians evaluate the codebase and their dependencies using various techniques to find as many flaws as possible and suggest security-related improvements. We as a company stand behind our brand and the level of service that is provided but also recommend subsequent security reviews, on-chain monitoring, and high whitehat incentivization.

3. Risk classification

	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 significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - leads to a moderate loss of assets in the protocol or some disruption of the protocol's functionality.
- **Low** - funds are not at risk.

3.2. Likelihood

- **High** - almost **certain** to happen, easy to perform, or highly incentivized.
- **Medium** - only **conditionally possible**, but still relatively likely.
- **Low** - requires specific state or **little-to-no incentive**.

3.3. Action required for severity levels

- High - **Must** fix (before deployment if not already deployed).
- Medium - **Should** fix.
- Low - **Could** fix.

4. Executive Summary

Overview

Project	fMoney
Repository	Private
Commit Hash	7763e955ec5ec2599e969e9227b76197b77c9ea8
Resolution	19cbbcb2e25e8836f1e69420a53d0657decf2322
Timeline	January 15 - January 18, 2024

Scope

CErc20.sol
CToken.sol

Issues Found

Critical Risk	0
High Risk	0
Medium Risk	1
Low/Info Risk	0

5. Findings

5.1. Medium severity

5.1.1. Repaying with cToken won't be possible due to overconstrained redeemAllowed check

Severity: Medium Risk

Description: Repaying with cToken won't be possible when a user has borrowed towards his debt ceiling, since `redeemAllowed` is called before repaying the borrows. Due to that, the comptroller will assume that the user is simply trying to redeem his collateral without first repaying the borrowed amount.

Let's say Bob supplied 1 USDC and borrowed the 0.7 USDC (collateral factor = 70%), now any attempt to repay with cToken, even 1 wei, will be reverting since in `ComptrollerV2::getHypotheticalAccountLiquidityInternal` Bob's borrow has not been decreased yet.

Repayments with cToken will always revert, indicating insufficient liquidity, while in reality position is in a completely healthy state.

Recommendation: The flow should be modified to replicate the repay/redeem:

1. Decrease the `accountBorrows` and `totalBorrows` as the user has repaid.

2. Perform the `redeemAllowed`, at this point, the HF is increased, if it reverts, the position has already been in a liquidation state.

3. Burn the cTokens and decrease the `totalSupply`.

Resolution: Fixed