Florence Finance Security Review

Pashov Audit Group

Conducted by: pashov

October 19th, 2023

Contents

1. About pashov

Krum Pashov, or **pashov**, is an independent smart contract security researcher. Having found numerous security vulnerabilities in various protocols, he does his best to contribute to the blockchain ecosystem and its protocols by putting time and effort into security research & reviews. Check his previous work here or reach out on Twitter @pashovkrum.

2. Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where I try to find as many vulnerabilities as possible. I can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

3. Introduction

A time-boxed security review of the **Florence Finance** protocol was done by **pashov**, with a focus on the security aspects of the application's smart contracts implementation.

4. About Florence Finance

Copied from the first security review

Florence Finance is a protocol that allows users to fund real-world loans with crypto assets. It works by letting users deposit stablecoins to the protocol in return for receipt tokens ($FLR or Loan Vault Tokens). Then the protocol will provide fiat currency (euros) to real-world lending platforms, who in turn lend the fiat currency to businesses. When loans are repaid, interest and principal flow back to the protocol to be distributed back to users. The protocol does not guarantee liquidity and is similar to peer-2-peer lending.

Users of Florence Finance can use the platform in two ways:

1. Use your stablecoins in a Loan Vault to earn yield from a real-world loan
2. Stake $FLR and earn Medici Token ($MDC)

In both ways the user should have $FLR, which he can get by swapping his stablecoins on a DEX for $FLR, which is backed 1:1 to EUR by the principal amount in the real-world loans. Then he can deposit them in a Loan Vault and receive Loan Vault Tokens (the vaults are following ERC4626) with which he immediately starts earning interest and can later withdraw his $FLR, receiving the principal plus earned yield, by burning the Loan Vault Tokens. If the user chooses to earn $MDC rewards instead, he can stake his $FLR in the staking pool and then withdraw or claim rewards any time.

More docs

Continued

The protocol was initially deployed on Ethereum mainnet and has recently been deployed on Arbitrum (because of gas fees cost). The TVL migration to the L2 chain has almost been completed already, and the way it works is by making the `FlorinToken` bridgeable (it is upgradeable). Same is done for the `FlorenceFinanceMediciToken`. The `FlorenceFinanceMediciToken` contract is the new version of the `MediciToken` (which will be deprecated). Users can use the `MediciConverter` contract to migrate their balances.

Observations

The contracts under review were live on Arbitrum at the point of the audit.

All contracts are behind an upgradeable proxy, making the protocol have a high centralization factor.

Privileged Roles & Actors

- `MediciConverter` owner - can pause/unpause conversions
- `FlorinToken` owner - can freely mint tokens as well as call `registerTokenOnL2` and `initializeArbitrumBridging`, same for the Arbitrum version of the token but without the `registerTokenOnL2` method authority
- `FlorenceFinanceMediciToken` owner - can freely mint tokens as well as call `registerTokenOnL2` and `initializeArbitrumBridging`, same for the Arbitrum version of the token but without the `registerTokenOnL2` method authority

5. Risk Classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

5.1. Impact

- High - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- Medium - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- Low - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

5.2. Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

5.3. 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

6. Security Assessment Summary

***review commit hash* - f21fe8e3264a401a9732174c8ec6a2367102c563**

No fixes implemented.

Scope

The following smart contracts were in scope of the audit:

- `FlorenceFinanceMediciToken`
- `FlorenceFinanceMediciTokenArbitrum`
- `FlorinToken`
- `FlorinTokenArbitrum`
- `MediciConverter`
- `FlorinTreasury` (diff between `f21fe8e3264a401a9732174c8ec6a2367102c563` and `616e9d4ba18eef293dc76fb95144bd11fb29549b` commits)
- `LoanVault` (diff between `f21fe8e3264a401a9732174c8ec6a2367102c563` and `616e9d4ba18eef293dc76fb95144bd11fb29549b` commits)

7. Executive Summary

Over the course of the security review, pashov engaged with Florence Finance to review Florence Finance. In this period of time a total of **4** issues were uncovered.

Protocol Summary

Protocol Name	Florence Finance
Date	October 19th, 2023

Findings Count

Severity	**Amount**
Medium	1
Low	3
Total Findings	**4**

Summary of Findings

ID	Title	Severity	Status
[M-01]	Burned tokens can be re-minted into the totalSupply	Medium	Resolved
[L-01]	Missing Arbitrum Sequencer availability check	Low	Resolved
[L-02]	Upgradeability best practices are not followed	Low	Resolved
[L-03]	Enforce initializer methods to be callable just once	Low	Resolved

8. Findings

8.1. Medium Findings

[M-01] Burned tokens can be re-minted into the `totalSupply`

Severity

Impact: Low, as it won't lead to funds loss but breaks a protocol invariant/assumption

Likelihood: High, as it becomes a problem whenever someone burns their tokens

Description

The `FlorenceFinanceMediciToken` contract inherits from `ERC20CappedUpgradeable` and has a max supply limit of "1_000_000_000 * 10 ** 18" token units. The issue is that the contract also inherits from the `ERC20BurnableUpgradeable` contract, which means that when a user calls the `burn` method, the `totalSupply` will be subtracted from, meaning if 10 tokens existed and are all burned, but then 10 new tokens are minted, now `totalSupply = 10` which is not the assumption that the protocol has, which is that the total supply of minted tokens can be maximum "1_000_000_000 * 10 ** 18".

Recommendations

Remove the inheritance from `ERC20BurnableUpgradeable` in `FlorenceFinanceMediciToken` so that burning tokens with subtracting from `totalSupply` is not possible.

8.2. Low Findings

[L-01] Missing Arbitrum Sequencer availability check

The `getFundingTokenExchangeRate` method in `LoanVault` makes use of Chainlink price feeds by calling the `latestRoundData` method. While there are sufficient validations for the price feed answer, a check is missing for the L2 sequencer availability which has to be there since the protocol is moving from Ethereum to Arbitrum. In case the L2 Sequencer is unavailable the protocol will be operating with a stale price and also when the sequencer is back up then all of queued transactions will be executed on Arbitrum before new ones can be done. This can result in the `LoanVault::getFundingTokenExchangeRate` using a stale price, which means that a user might receive more or less shares from the `LoanVault` than he should have had. Still, since all of the funding requests are first approved off-chain, the probability of this happening is much lower. For a fix you can follow the Chainlink docs to add a check for sequencer availability.

[L-02] Upgradeability best practices are not followed

In `LoanVault` the storage layout has been changed in a non-upgradeability safe way. Between commit `616e9d4ba18eef293dc76fb95144bd11fb29549b` and the current HEAD of the `audit` branch it seems like the `fundingFee` storage variable has changed its place and also the whitelisting storage variables have been removed. This is quite dangerous as if a potential upgrade is done on the Ethereum contracts it can mess up the storage layout, potentially bricking the contract. This is highly unlikely to happen though as the protocol is using the OpenZeppelin's upgrades plugin which protects from this. Still, upgradeability best practices should always be applied - do not reorder storage variables and instead of removing old ones just deprecate them.

[L-03] Enforce initializer methods to be callable just once

The `setLoansOutstanding` method in `LoanVault` says in its NatSpec that "This is used for the initial deployment on Arbitrum." - the method should be callable only once. The same note is valid for the `initializeArbitrumBridging` method in token contracts in the codebase.