G₀ Group

Security Review of

TrueFi

December 2022

TrueFi / December 2022

Files in scope

Following files in https://github.com/trusttoken/contracts-beryllium/tree/64f78e332293a9cf9c1810f458520bb3ba201e77:

```
- controllers/
    - BlockedTransferController.sol
    - DepositController.sol
    - FeeStrategy.sol
    - FixedInterestOnlyLoansValuationStrategy.sol
    - LegacyWhitelistDepositController.sol
    - LegacyWithdrawController.sol
    - MultiFarmAllowedTransferController.sol
    - MultiInstrumentValuationStrategy.sol
    - OpenTransferController.sol
    - WithdrawController.sol
    - WithdrawOncePortfolioIsClosedController.sol
- proxy/ProxyWrapper.sol
- AutomatedLineOfCredit.sol
- AutomatedLineOfCreditFactory.sol
- FlexiblePortfolio.sol
- FlexiblePortfolioFactory.sol
- PortfolioFactory.sol
```

Current status

All discovered issues have been acknowledged by the developer. Some will be fixed in future versions of the code, when that happens an updated version of this report will be issued.

Issues

1. A re-entrancy issue allows user to shift all time based fees on other users when tokens with transfer callbacks are used

type: security / severity: major

In following places:

https://github.com/trusttoken/contracts-beryllium/blob/64f78e332293a9cf9c1810f458520bb3ba201e77/AutomatedLineOfCredit.sol#L174

https://github.com/trusttoken/contracts-beryllium/blob/64f78e332293a9cf9c1810f458520bb3ba201e77/AutomatedLineOfCredit.sol#L262

https://github.com/trusttoken/contracts-beryllium/blob/64f78e332293a9cf9c1810f458520bb3ba201e77/FlexiblePortfolio.sol#L190

token transfer to user's address happens before full fee accounting has been performed. If the tokens performs a callback to the user's address, it allows the user to re-enter the contract in an inconsistent state where all time based fees are seemingly paid, but the balance of the contract hasn't been decreased yet. This enables the following attack:

a. wait for some fees to accrue

b. withdraw minimal possible amount of shares

c. re-enter the contract from the first withdrawal with withdrawal of your remaining shares

d. let the first (and second) withdrawal finish and deposit all received funds back into the contract increasing the amount of shares for free

This attack effectively allows the attacker to shift time based fees onto other users and collect just interest.

status - acknowledged

The developer will work around the issue by not allowing tokens with callbacks to be used in the system, in future releases, the issue will be fixed.

2. Debtor can strategically buy into the contract before repaying interest ahead of schedule or repaying a defaulted loan

type: security / severity: medium

In `FlexiblePortfolio`, repaying a defaulted loan or repaying interest ahead of schedule leads to a sudden increase in the value of outstanding shares. Debtor can take advantage of this fact to buy into the lending pool just before they repay the loan to recover the repaid value when they sell the shares later. This danger is partially mitigated by the use of `WithdrawOncePortfolioIsClosedController`, but not entirely since a defaulted loan can still be repaid after the portfolio is closed.

status - acknowledged

The issues have been acknowledged by the developer.

3. In `AutomatedLineOfCredit._burnFrom` `msg.sender` is used instead of the `spender` argument

type: code fragility / severity: minor

`AutomatedLineOfCredit._burnFrom` internal function has a `spender` argument that is supposed to identify the account whose allowance or balance is used to cover cost of burning. In the function itself, this argument is only used once and is replaced by `msg.sender` in other parts of the code. This doesn't cause any issues in current version of the code because the function is only ever called with `spender == msg.sender`, but might cause issues in the future.

status - acknowledged

The issues have been acknowledged by the developer and will be fixed in future versions of the code.