Term Finance

Smart Contract Changes (May 2024)

Security Assessment Report

Version: 2.1

May, 2024

Contents

Introduction 2
- Disclaimer 2
- Document Structure 2
- Overview 2

Security Assessment Summary 3
- Scope 3
- Approach 3
- Coverage Limitations 3
- Findings Summary 4

Detailed Findings 5

Summary of Findings 6
- No Ability To Revoke `ADMIN_ROLE`, `DEVOPS_ROLE` or `SPECIALIST_ROLE` 7
- Miscellaneous General Comments 8

A Test Suite 10

B Vulnerability Severity Classification 11

Introduction

Sigma Prime was commercially engaged to perform a time-boxed security review of the Term Finance smart contracts. The review focused solely on the security aspects of the Solidity implementation of the contract, though general recommendations and informational comments are also provided.

Disclaimer

Sigma Prime makes all effort but holds no responsibility for the findings of this security review. Sigma Prime does not provide any guarantees relating to the function of the smart contract. Sigma Prime makes no judgements on, or provides any security review, regarding the underlying business model or the individuals involved in the project.

Document Structure

The first section provides an overview of the functionality of the Term Finance smart contracts contained within the scope of the security review. A summary followed by a detailed review of the discovered vulnerabilities is then given which assigns each vulnerability a severity rating (see Vulnerability Severity Classification), an *open/closed/resolved* status and a recommendation. Additionally, findings which do not have direct security implications (but are potentially of interest) are marked as *informational*.

Outputs of automated testing that were developed during this assessment are also included for reference (in the Appendix: Test Suite).

The appendix provides additional documentation, including the severity matrix used to classify vulnerabilities within the Term Finance smart contracts.

Overview

Term Finance is a non-custodial fixed-rate liquidity protocol modelled on tri-party repo arrangements common in traditional finance.

Liquidity suppliers and takers are matched through a unique weekly auction process where liquidity takers submit bids and suppliers submit offers to the protocol, which then determines an interest rate that clears the market.

Bidders who bid more than the clearing rate receive liquidity and lenders asking less than the clearing rate supply it.

Security Assessment Summary

Scope

The scope of this time-boxed review was strictly limited to the code changes related to the following PRs:

- `PR 1259`
- `PR 1261`
- `PR 1263`
- `PR 1294`
- `PR 1312`
- `PR 1289`
- `PR 1292`
- `PR 1299`
- `PR 1309`
- `PR 1311`

Note: third party libraries and dependencies, such as OpenZeppelin, were excluded from the scope of this assessment.

Approach

The review was conducted on the files hosted on the term-finance repository at commit 93d0dd7.

Retesting was conducted on individual PRs (refer to "Resolution" section of each finding), which were then subsequently merged to term-finance-contracts repository at commit c97dd99 (release 0.9.0).

The manual review focused on identifying issues associated with the business logic implementation of the contracts. This includes their internal interactions, intended functionality and correct implementation with respect to the underlying functionality of the Ethereum Virtual Machine (for example, verifying correct storage/memory layout).

Additionally, the manual review process focused on identifying vulnerabilities related to known Solidity anti-patterns and attack vectors, such as re-entrancy, front-running, integer overflow/underflow and correct visibility specifiers.

For a more detailed, but non-exhaustive list of examined vectors, see [1, 2].

To support this review, the testing team also utilised the following automated testing tools:

- Mythril: `https://github.com/ConsenSys/mythril`
- Slither: `https://github.com/trailofbits/slither`
- Surya: `https://github.com/ConsenSys/surya`

Output for these automated tools is available upon request.

Coverage Limitations

Due to a time-boxed nature of this review, all documented vulnerabilities reflect best effort within the allotted, limited engagement time. As such, Sigma Prime recommends to further investigate areas of the code, and any related functionality, where majority of critical and high risk vulnerabilities were identified.

Findings Summary

The testing team identified a total of 2 issues during this assessment. Categorised by their severity:

- Low: 1 issue.
- Informational: 1 issue.

Detailed Findings

This section provides a detailed description of the vulnerabilities identified within the Term Finance smart contracts. Each vulnerability has a severity classification which is determined from the likelihood and impact of each issue by the matrix given in the Appendix: Vulnerability Severity Classification.

A number of additional properties of the contracts, including gas optimisations, are also described in this section and are labelled as "informational".

Each vulnerability is also assigned a **status**:

- ***Open:*** the issue has not been addressed by the project team.
- ***Resolved:*** the issue was acknowledged by the project team and updates to the affected contract(s) have been made to mitigate the related risk.
- ***Closed:*** the issue was acknowledged by the project team but no further actions have been taken.

Summary of Findings

ID	Description	Severity	Status
TRM4-01	No Ability To Revoke `ADMIN_ROLE`, `DEVOPS_ROLE` or `SPECIALIST_ROLE`	Low	Resolved
TRM4-02	Miscellaneous General Comments	Informational	Closed

TRM4-01	No Ability To Revoke `ADMIN_ROLE`, `DEVOPS_ROLE` or `SPECIALIST_ROLE`		
Asset	`TermController.sol`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Medium	Likelihood: Low

Description

There is currently no way to remove `ADMIN_ROLE`, `DEVOPS_ROLE` or `SPECIALIST_ROLE` permission once it was assigned to an external address.

`SPECIALIST_ROLE` is granted to addresses via `grantMintExposureAccess()` in order to enable minting rights, but there is no equivalent function to revoke them.

Note, `CONTROLLER_ADMIN_ROLE` has `updateControllerAdminWallet()` function that allows to update controller admin wallet address, while also revoking `CONTROLLER_ADMIN_ROLE` permission from previously set address.

The same should be possible for all other roles that grant permissions to external addresses.

Recommendations

Implement relevant functions to revoke roles and permissions assigned to external addresses.

Resolution

The issue has been resolved in PR-1324 - `revokeMintExposureAccess()` has been added to enable revoking `SPECIALIST_ROLE` permission.

TRM4-02	Miscellaneous General Comments
Asset	`contracts/*`
Status	**Closed:** See Resolution
Rating	Informational

Description

This section details miscellaneous findings discovered by the testing team that do not have direct security implications:

1. **Nested inheritance**

   Related Asset(s): `TermRepoToken.sol`

   `ERC20PermitUpgradeable` is also already `ERC20Upgradeable`, as such the explicit inheritance on line [**25**] is not necessary.

2. **Input address validation**

   Related Asset(s): `TermRepoToken.sol, TermInitializer.sol`

   In `TermRepoToken`, there are no checks in the `initialize()` function to ensure the `address` type variables of `TermRepoTokenConfig` are not set to `0x0`.

   Additionally, as there are no setter function for `config`, consider implementing it (callable by admin only), to cater for situations where it needs to be updated due to manual entry mistakes.

   In `TermInitializer`, consider introducing a check for `0x0` addresses in `setupTerm()` for all `address` type variables passed in to all `pairTermContracts()`.

   Currently, in case of a mistake, it's impossible to update the paired addresses as the `pairTermContracts()` function can only be called once.

3. **No checks for zero `amount`**

   Related Asset(s): `TermRepoServicer.sol`

   There are no checks in `submitRepurchasePayment()` to ensure the `amount` variable is not zero.

   Implement similar checks as it was done in PR-1294.

4. **No negative price checks**

   Related Asset(s): `TermPriceConsumerV3WithSequencer.sol`

   Checks for negative prices are not implemented in the same fashion as they are in `TermPriceConsumerV3` via PR-1309.

   Ensure the same changes are implemented in `TermPriceConsumerV3WithSequencer`.

5. **Redundant checks**

   Related Asset(s): `TermPriceConsumerV3.sol, TermPriceConsumerV3WithSequencer.sol`

   In `TermPriceConsumerV3` on line [**252**], the `fallbackPrice > 0` check is unnecessary as it's already in the `else` section of `if (fallbackPrice <= 0) { ... } else { ... }`, which infers the same outcome.

   Note, the same issue is present in `TermPriceConsumerV3WithSequencer.sol`.

6. **`_getLatestPrice()` may return stale prices**

   Related Asset(s): `TermPriceConsumerV3.sol, TermPriceConsumerV3WithSequencer.sol`

   The `_getLatestPrice()` always returns a price, even if extremely stale (either from primary or a fallback oracle). As such `usdValueOfTokens()` may produce outdated results.

   Ensure this is understood and, if deemed feasible, consider reverting if the returned price is beyond an acceptable staleness threshold.

Recommendations

Ensure that the comments are understood and acknowledged, and consider implementing the suggestions above.

Resolution

The comments above have been acknowledged by the development team, and relevant changes actioned in the following PR's:

- Oracle related changes: PR-1326
- Zero `amount` checks: PR-1327
- Input address validation: PR-1325

Appendix A Test Suite

A non-exhaustive list of tests were constructed to aid this security review and are given along with this document. The `brownie` framework was used to perform these tests and the output is given below.

```
Launching 'ganache-cli --chain.vmErrorsOnRPCResponse true --server.port 8545 --miner.blockGasLimit 12000000 --wallet.totalAccounts
      ↪  20 --hardfork istanbul --wallet.mnemonic brownie --wallet.defaultBalance 1000000'...

tests/test_TermAuction.py .....                                                      [  5%]
tests/test_TermAuctionBidLocker.py .................                                 [ 25%]
tests/test_TermAuctionOfferLocker.py ................                                [ 44%]
tests/test_TermController.py ...                                                     [ 48%]
tests/test_TermEventEmitter.py .                                                     [ 49%]
tests/test_TermInitializer.py ..                                                     [ 51%]
tests/test_TermPriceConsumerV3.py ...                                                [ 54%]
tests/test_TermRepoCollateralManager.py ......                                       [ 60%]
tests/test_TermRepoLocker.py ...                                                     [ 64%]
tests/test_TermRepoRolloverManager.py ..........                                     [ 75%]
tests/test_TermRepoServicer.py .....x...                                             [ 86%]
tests/test_TermRepoToken.py .........                                                [ 96%]
tests/test_poc_repurchase.py .                                                       [ 97%]
tests/test_poc_rolloverCollateral.py .                                               [ 98%]
tests/test_poc_rolloverToTwoAuctions.py .                                            [100%]

  ================= 87 passed, 1 xfailed, 2106 warnings in 165.09s (0:02:45) ==================
```

Appendix B Vulnerability Severity Classification

This security review classifies vulnerabilities based on their potential impact and likelihood of occurance. The total severity of a vulnerability is derived from these two metrics based on the following matrix.

Impact	High	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Low	Low	Medium
		Low	Medium	High
			Likelihood	

Table 1: Severity Matrix - How the severity of a vulnerability is given based on the *impact* and the *likelihood* of a vulnerability.

References

[1] Sigma Prime. Solidity Security. Blog, 2018, Available: `https://blog.sigmaprime.io/solidity-security.html`. [Accessed 2018].

[2] NCC Group. DASP - Top 10. Website, 2018, Available: `http://www.dasp.co/`. [Accessed 2018].