Volt System Oracle Security Review

AlwaysBeGrowing

Reviewer
Kyle Trusler | Namaskar

August 1, 2022

1 Executive Summary

I performed a security review of the Volt System Oracle over two days for a total of 20 hours. During this time, I became familiar with the protocol, did a line by line review, and considered potential attack vectors. An area of particular concern was the deployment script for VIP 2. The review was over PR 82 in its entirety. Specifically, commit 73a7ead. After the commit in question, there were related changes to roles and an addition of simulating the proposals. These changes includes up until commit 3d6156d.

I found no issues with the implementation other than one code-style informational finding.

PR	Commit
PR 82	73a7ead

Summary

Timeline	July 11, 2022 - August 1, 2022
Methods	Manual Review
Documentation	Good
Testing Coverage	High

Total Issues

Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	0
Informational	1

Contents

2 Always Be Growing

ABG is a talented group of engineers focused on growing the web3 ecosystem. Learn more at https://abg.garden

3 Introduction

For this PR, a new "oracle" will be introduced as a stop-gap measure. This will replace the existing CPI based oracle.

This VoltSystemOracle has a pre-defined target set at deployment for interest to compound over an, also set at deployment, timeframe. While reviewing the contract, there are only two functions, each with concerns.

1. Does these two functions work in the way they were intended?
2. Do the tests have coverage over the functionality?
3. Is the deployment strategy going to be successful?

As part of this PR, a new OraclePassThrough and this VoltSystemOracle will be deployed. Additionally, other contract calls will be made with an Optimistic-Timelock.

Disclaimer: This security review does not guarantee against a hack. It is a snapshot in time of brink according to the specific commit by a single human. Any modifications to the code will require a new security review.

4 Methodology

The following is steps I took to evaluate the change.

- Clone & Setup project
- Read Readme and related documentation
- Use Surya on VoltSystemOracle - Not super helpful except it shows the lack of inherent smart contract interaction
  - Report
  - Contract interaction
  - Inheritance

- Line by line review
    - Contract
    - Interface
    - Unit Tests
    - Integration Tests
- Proposal Deployment

4.0.1 Line by line review

Read through Interface and NatSpec. **Assumed** contract deployment would be as intended & values used to construct would be reasonable. Kept in mind Solcutiry guidelines.

The tests and fuzz tests appear to cover every aspect of the state variables being correctly set.

getCurrentOraclePrice Correctly a public view function, has no external calls,

Understood the time delta is capped on range of 1 - TIMEFRAME, the price percentage change is the total amount of monthly change, and price delta is capped on a range of nearly 0 - 1 times the total monthly change.

The tests and fuzz tests appear to cover every aspect of this.

compoundInterest **Assumed** the time period will not be out of date due to the keepers, the function is correctly external and has no external calls.

Understood periodStartTime and TIMEFRAME are both in the same denomination, and even if block timestamp was manipulated after the `require`, the time would be in a valid state. A valid state being also means the period will not be more than one TIMEFRAME worth of compounding.

The tests and fuzz tests appear to cover every aspect of this.

4.0.2 Integration Tests

I considered results of pointing the new oracle and oracle pass through. I did a line-by-line review of the integration tests.

1. Does the new oracle implement the correct methods?
2. Is the price returned correctly?
3. Is the configuration correct?

The integration tests for both the Arbitrum and Mainnet networks covered these questions and were all passing. The tests looked sound.

4.0.3 Proposal Deployment

The deployment and proposal process was new to me and seemed like a particular area of interest as deployment, role changing, and setting configuration happens. I reviewed the flow and believe it to be valid. During deployment the starting oracle price must be set as close to the current price as possible to avoid potential arbitrage. I ran through the `checkProposal` script on the forked node which succesfully ran.

After my initial review, a robust Solidity-based simulation environment was introduced. The simulation covers the proposal, scheduling, and execution of a timelock transaction. This simulated action, set out by VIP-2 is included in an integration test.

5 Findings

5.1 Critical Risk

No findings

5.2 High Risk

No findings

5.3 Medium Risk

No findings

5.4 Low Risk

No findings

5.5 Informational

1 finding

5.5.1 Unnecessary / Inconsistent Override of Properties

Severity: Informational

Context: VoltSystemOracle.sol#L30

Description:

Currently, all of the state variables and methods (except monthlyChangeRateBasisPoints) are decorated with "override". This usually indicates that a base method has functionaly replaced.

```
uint256 public immutable monthlyChangeRateBasisPoints;
```

Recommendation:

```
- uint256 public immutable monthlyChangeRateBasisPoints;
+ uint256 public immutable override monthlyChangeRateBasisPoints;
```

An alternative recommendation is to remove all override decoration as the interface is mearly being implemented, and override is not needed. See the issue and the PR for more discussion.

Resolution VOLT removed the override modifier on state variables in commit 05894cb

6 Additional Comments

The temporary oracle code is well-tested, well-documented, and appears to work as intended. The changes to implement, by simulating the timelock, gives a higher confidence in the integration process of switching to the new oracle.