Volmex Volatility Tokens v2 Smart Contract Audit by ZK Labs

MATTHEW DI FERRANTE

2022-06-15

Contents

Preface

Between 2022-03-21 and 2022-06-15, ZK Labs performed an audit of the Volmex Finance smart contracts. The findings are detailed below.

ZK Labs have no stake or vested interest in Volmex Finance. This audit was performed under a contracted rate with no other compensation.

Authenticity

The final version of document should have an attached cryptographic signature to ensure it has not been tampered with. The signature can be verified using the public key from https://keybase.io/mattdf

Audit Goals and Focus

Smart Contract Best Practices

This audit will evaluate whether the codebase follows the current established best practices for smart contract development.

Code Correctness

This audit will evaluate whether the code does what it is intended to do.

Code Quality

This audit will evaluate whether the code has been written in a way that ensures readability and maintainability.

Security

This audit will look for any exploitable security vulnerabilities, or other potential threats to either the operators of Volmex or its users.

Testing and testability

This audit will examine how easily tested the code is, and review how thoroughly tested the code is.

Overview

Audited Material

The audit consists of the Volmex volatility tokens v2 contracts (Volmex), with commit hash ff07616b3eb8e9d3f280101264aeb6fb9fc2c6da.

The contracts in scope for the audit are listed below:

```
9457be7241ed2c050fb57e14f004defc6972a2ee9e918e95317f138e035a5809  ./VolmexController.sol
44f142729b52e0648108470d2bd60ba472a9d99c926fb68284151d4924ed608b  ./VolmexPool.sol
deb42f4e0c976a616fbcc8f9fcdfb2d01b6b205cab03ca4a56a5f4e5c1e7d188  ./libs/tokens/Token.sol
54c7c87e0d2f7f8f51b360c85ebc6599cafd86394ef884df908a0e98067c71c0  ./libs/tokens/TokenMetadataGenerator.sol
2e58dae26937f73c5ac7d949084fa8a50d15e54d6783d75e6346e4e600b2408a  ./maths/Const.sol
d854349c4f55b24fef04ff2c232f68a5384c679b909600e8082f7ce143fc0c45  ./maths/Num.sol
de7dcaf663c529ea132f93e1f7812c24f4f5cf0377d7612288a3efe929af2e53  ./maths/NumExtra.sol
a03fe0157df4755c3e423a76924faaefdd2e541d5d7e7edc87997f8bb7fb33df  ./maths/Math.sol
d8d9ec079a0027a34b5cf1628dc4d42860e9af1d2d36e0050cf00ca9f89624ba  ./oracles/VolmexOracle.sol
1066b1aa3894ae00fc29e4bf6f91f50af4e6940ee3f1d03693de416074ac5ba8  ./oracles/VolmexTWAP.sol
10c56ae368eb46e79dd989a92081e8b913497c204a3fab00840429c3245ebe12  ./protocol/VolmexPositionToken.sol
bfc1362ee9097135e3238e4ab3c42713d2c7e8c894eba07756a118d999380f60  ./protocol/VolmexProtocol.sol
3690c5c1176f65378f262a41f9f8d798ca596cea59e686bffdb4fd93d35b1ade  ./protocol/VolmexProtocolWithPrecision.sol
4be0fa041e4c86e6eabf8b287295b8d1289547ee23a78476550a765045d1b1b1  ./protocol/TestCollateralToken.sol
7215d326c0055c2ab24527b5e72e0896f7d33f78a6ea6cfd0f6102ba0f06963d  ./repricers/VolmexRepricer.sol
3637f7288c73527ec94ad2e234905f67b70e0bcd806aa3ac59bde04081f216bd  ./VolmexPoolView.sol
```

Networks that Volmex will be deployed on are Polygon and Arbitrum - both EVM-based chains.

Test Coverage

Test coverage is very good, both functionally and in terms of line covered, with average coverage greater than 90% across all files:

```
-----------------------------------|----------|----------|----------|----------|----------------|
File                               |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
-----------------------------------|----------|----------|----------|----------|----------------|
 contracts/                        |    97.37 |    78.57 |    98.41 |     97.9 |                |
  VolmexController.sol             |    99.27 |       90 |      100 |      100 |                |
  VolmexPool.sol                   |    94.87 |    72.73 |     97.3 |    95.52 |... 936,940,944 |
  VolmexPoolView.sol               |      100 |    83.33 |      100 |      100 |                |
 contracts/interfaces/             |      100 |      100 |      100 |      100 |                |
  IEIP20NonStandard.sol            |      100 |      100 |      100 |      100 |                |
  IERC20.sol                       |      100 |      100 |      100 |      100 |                |
  IERC20Modified.sol               |      100 |      100 |      100 |      100 |                |
  IPausablePool.sol                |      100 |      100 |      100 |      100 |                |
  IVolmexController.sol            |      100 |      100 |      100 |      100 |                |
  IVolmexOracle.sol                |      100 |      100 |      100 |      100 |                |
  IVolmexPool.sol                  |      100 |      100 |      100 |      100 |                |
  IVolmexPoolView.sol              |      100 |      100 |      100 |      100 |                |
  IVolmexProtocol.sol              |      100 |      100 |      100 |      100 |                |
  IVolmexRepricer.sol              |      100 |      100 |      100 |      100 |                |
 contracts/libs/tokens/            |      100 |       90 |      100 |      100 |                |
  Token.sol                        |      100 |       90 |      100 |      100 |                |
  TokenMetadataGenerator.sol       |      100 |      100 |      100 |      100 |                |
 contracts/maths/                  |    97.67 |       75 |      100 |    97.67 |                |
  Const.sol                        |      100 |      100 |      100 |      100 |                |
  Math.sol                         |      100 |      100 |      100 |      100 |                |
  Num.sol                          |    92.31 |    66.67 |      100 |    92.31 |             12 |
  NumExtra.sol                     |      100 |      100 |      100 |      100 |                |
 contracts/mocks/                  |      100 |       50 |      100 |      100 |                |
  NonCollateral.sol                |      100 |      100 |      100 |      100 |                |
  VolmexPoolMock.sol               |      100 |       50 |      100 |      100 |                |
  VolmexPositionTokenMock.sol      |      100 |      100 |      100 |      100 |                |
 contracts/oracles/                |      100 |    95.45 |      100 |      100 |                |
  VolmexOracle.sol                 |      100 |      100 |      100 |      100 |                |
  VolmexTWAP.sol                   |      100 |    83.33 |      100 |      100 |                |
 contracts/protocol/               |      100 |       85 |      100 |      100 |                |
  TestCollateralToken.sol          |      100 |      100 |      100 |      100 |                |
  VolmexPositionToken.sol          |      100 |      100 |      100 |      100 |                |
  VolmexProtocol.sol               |      100 |    91.67 |      100 |      100 |                |
  VolmexProtocolWithPrecision.sol  |      100 |       50 |      100 |      100 |                |
 contracts/repricers/              |      100 |      100 |      100 |      100 |                |
  VolmexRepricer.sol               |      100 |      100 |      100 |      100 |                |
-----------------------------------|----------|----------|----------|----------|----------------|
All files                          |    98.31 |    81.86 |    99.34 |    98.62 |                |
-----------------------------------|----------|----------|----------|----------|----------------|
```

Background

Volmex v2 is an exchange contract optimized for volatility tokens which is paired with Volmex v1 issuance and redemption functionality. The protocol enables VIX-like indices for crypto assets and trading functionality powered by Ethereum.

A primary feature of the Volmex AMM is that it dynamically reprices volatility token liquidity to a 30 min TWAP of the Volmex volatility index, helping volatility tokens tightly track Volmex volatility indices. Repricing happens using the repricer contract each block.

Architectural risks

All the core implementation contracts are upgradeable, and barring additional restrictions via multisig, can be arbitrarily upgraded without any delays - hence there is a single point of failure, in trust terms.

The volatility oracle itself is also owned by Volmex, and a compromise of the `owner` key which is able to manipulate parameters and update asset prices could also potentially lead to total loss of funds.

Contract Breakdown

The main user-facing functionality is implemented inside `VolmexPool`, `VolmexProtocol`, and `VolmexController`. The `VolmexOracle` contract is managed by the Volmex team, and provides a price feed that is leveraged by `VolmexRepricer`.

The entry point for most of the system's actions is via `VolmexController`, as that is the only contract allowed to call `VolmexPool`. The only remaining stateful functions that are user-callable outside of `VolmexController` are `collateralize`, `redeem` and `redeemSettled` inside of `VolmexProtocol` and `reprice` in `VolmexPool`.

Calculations in the code rely heavily on `ABDKMathQuad`, a library by `ABDK Consulting`. As this library is used in many other projects, we have assumed correctness of its implementation.

General Notes

Overall, the construction of the Volmex contracts is robust and we were not able to find any major code-level issues that would compromise security. The protocol is self-contained, relying mainly on the off-chain oracle that is controlled by the Volmex team itself, and as long as the oracle's price feed does not have a fault, there's little surface area that can be exploited.

The major code risk lies with the collateral tokens that are allowed to be accepted into `VolmexPool`. As the pools are permissioned, as long as the Volmex team does not add a token that contradicts the

assumptions of the codebase or that is incompatible with the pricing model, then that risk category is minimized.

Findings

NOTE: Proxy implementation choice

The system relies on proxy contracts from OpenZeppelin, and the deployment scripts and tests use the default `deployProxy`, which deploys a `TransparentUpgradeableProxy`. It is suggested by OpenZeppelin to instead use the `UUPSUpgradeable` proxy going forward, as it prevents accidentally upgrading the implementation to something that does not support upgrading the proxy, which would lock the upgradability of the proxy forever.

Response from Volmex: Acknowledged.

NOTE: Unnecessary use of virtual in VolmexProtocol

Every function in `VolmexProtocol` is marked as virtual, but the only contract inheriting it is `VolmexProtocolWithPrecision`, which overrides just `collateralize` and `_redeem`. If there are functions that should not be overridden by extension contracts, it would be better practice not to mark them as `virtual`.

Response from Volmex: Conscious design decision.

NOTE: Shadowed variable in VolmexPool

In the `_bind` function of `VolmexPool`, the `_balance` variable from the inherited `Token` implementation is shadowed. Though `Token._balance` is marked as `internal`, it is still exposed to child contracts. If the intention is for this variable to not to be directly accessible by children either, then it must be marked **`private`**, and shadowing will not occur.

Response from Volmex: `Token._balance` was changed to `Token._balances`

ISSUE: Oracle contract does not record when prices were last updated

The oracle contract itself doesn't have any information about when prices were updated, so if the oracle doesn't get updated for many blocks (in the case that infrastructure is down, or if there's an issue with the price feed from deribit), the values can become stale but the protocol will have no idea about it, and neither will any protocols integrating with Volmex (protocols that hold the tokens). It is suggested to implement a way to know when the last real update was by the oracle, so integrating protocols can choose to "freeze" the asset if the oracle seems like it's down.

Response from Volmex: Fixed in ff07616b3eb8e9d3f280101264aeb6fb9fc2c6da.

ISSUE: Use of block.number instead of block.timestamp for recording repricing

The use of block.number to record repricing rather than using a block.timestamp has risks, especially on Polygon/Arbitrum. Polygon has often been down for hours at a time, so block.number will only increase by 1 when the chain resumes, but hours may have passed, and that might be misleading to protocols that read the repricingBlock value to know how stale/fresh the value is. It is suggested to use block.timestamp instead.

Response from Volmex: Fixed in ff07616b3eb8e9d3f280101264aeb6fb9fc2c6da.