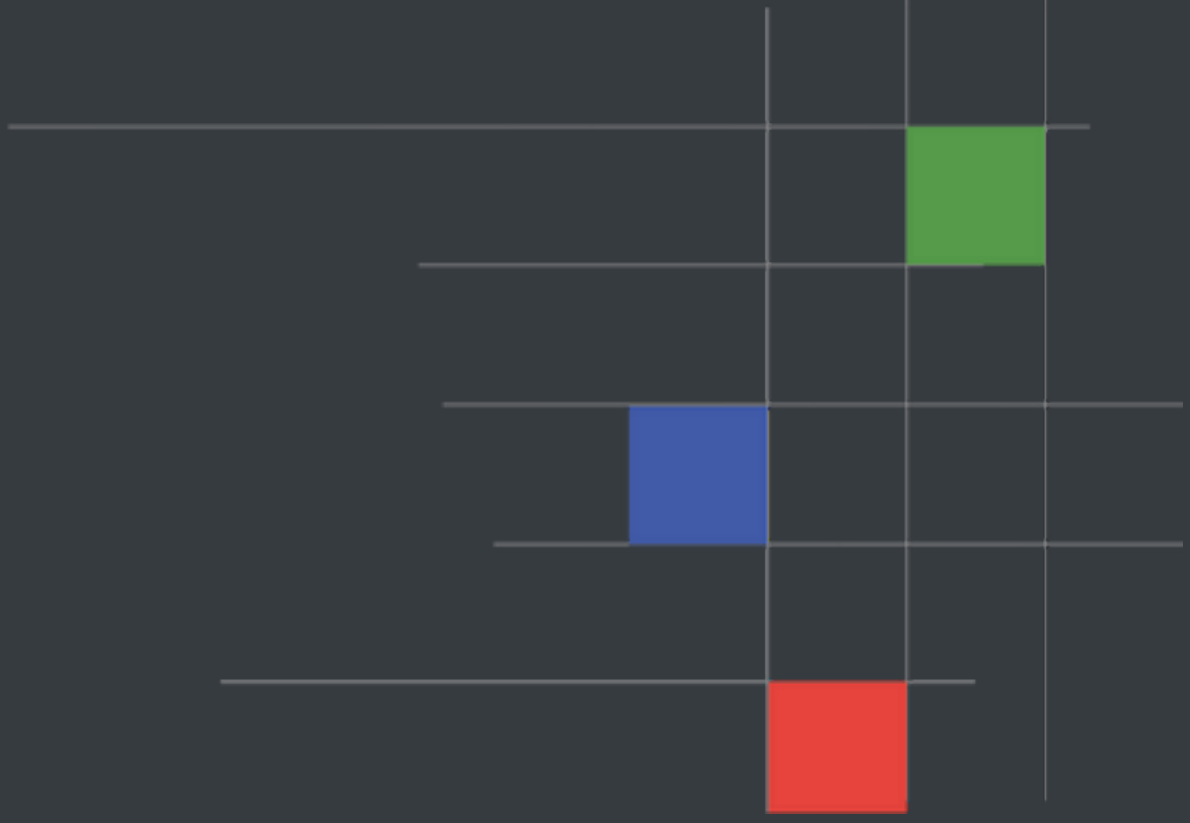
Volmex Finance

Collateralized Volatility Token

Security Assessment

May 21st, 2021



Disclaimer

CertiK reports are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security review.

CertiK Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

Overview

Project Summary

Project Name	Volmex Finance - Collateralized Volatility Token
Description	A positional token implementation backed by underlying collateral.
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	1. bf1ab0f21c9d77147ad34ca862ebdeadcb5c584b 2. efa11242ae872b4355a7ce3d20aea0ef6d955eab 3. 0a881616b47bf3a540662f59e4e951b9fcc8df7a 4. a0ab66ecdf0fe933fa319fb834511f229f6bb111

Audit Summary

Delivery Date	May 21st, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	1
Timeline	May 1st, 2021 - May 7th, 2021

Vulnerability Summary

Total Issues	4
Total Major	2
Total Informational	2

Executive Summary

We were tasked with auditing the codebase of Volmex Finance and in particular their collateralized volatility token implementation.

The system uses a volatility and inverse volatility token minting mechanism based on the supplied collateral which can then be redeemed via two methods depending on whether the price of the token has been settled or not, thereby increasing the redemption value of either the volatility or inverse volatility units depending on what the `settlementPrice` results in.

We were able to identify important issues in the notions the codebase uses to guard the contracts against flash loans and re-entrancies which we strongly recommend be adjusted to ensure the contract's security is future-proof.

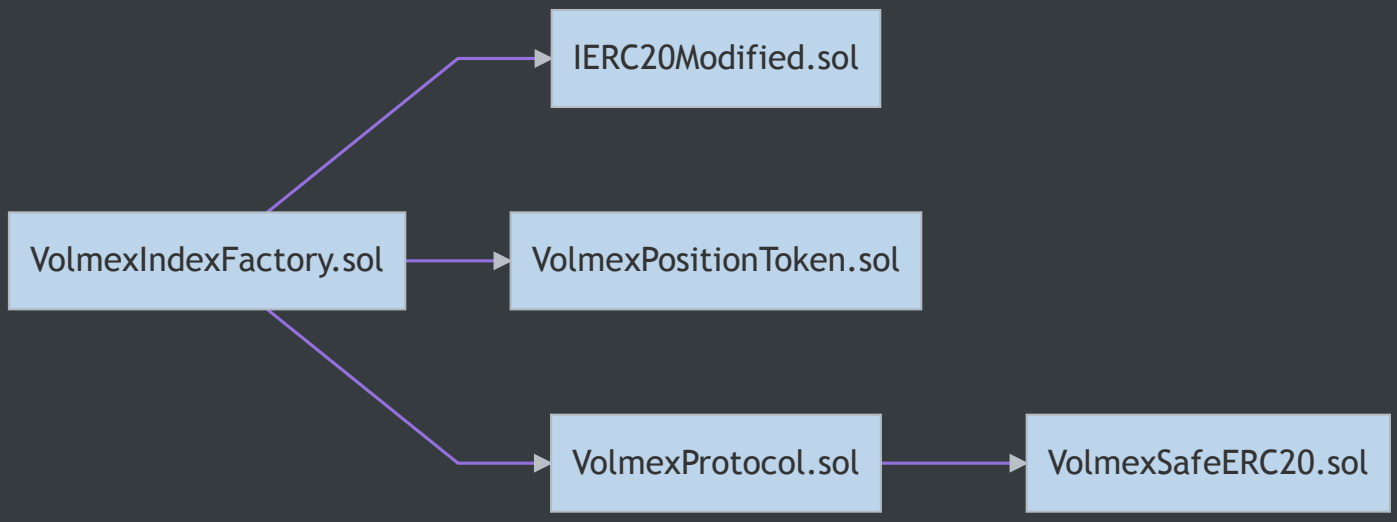
These issues were closely examined by the Volmex team which deduced that the mechanisms actually prohibit flash loan arbitrageurs from interacting with the protocol who are ultimately beneficial to its health. As a result, these mechanisms were removed from the system as they posed no issue to its operation.

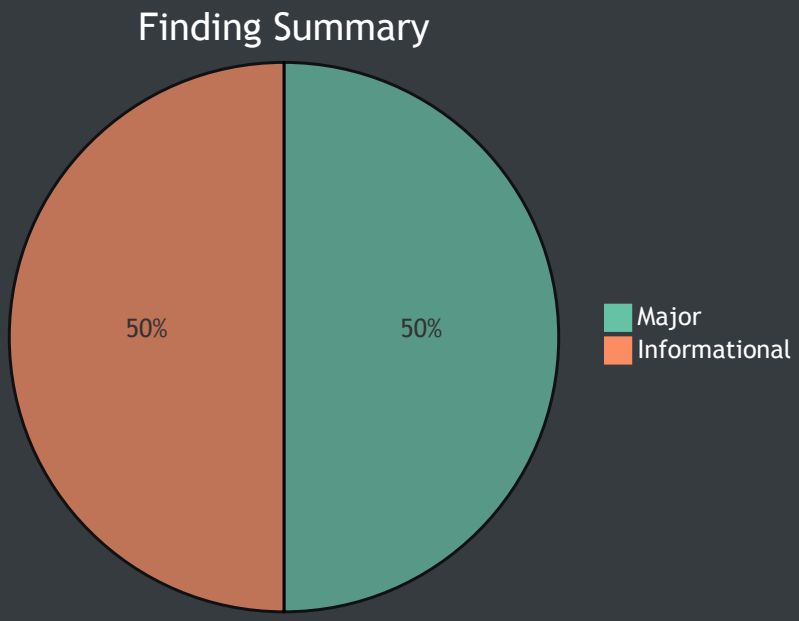
The system was investigated for other commonly identified attack vectors, such as re-entrancies and mathematical accuracies, however none were identified in this regard that severely affect the security of the system.

Files In Scope

ID	Contract	Location
VIF	VolmexIndexFactory.sol	contracts/VolmexIndexFactory.sol
VPL	VolmexProtocol.sol	contracts/VolmexProtocol.sol
IER	IERC20Modified.sol	contracts/interfaces/IERC20Modified.sol
IVP	IVolmexProtocol.sol	contracts/interfaces/IVolmexProtocol.sol
VSE	VolmexSafeERC20.sol	contracts/library/VolmexSafeERC20.sol
VPT	VolmexPositionToken.sol	contracts/tokens/VolmexPositionToken.sol

File Dependency Graph




Finding Summary
50%
50%
Major
Informational

Manual Review Findings

ID	Title	Type	Severity	Resolved
VPL-01	Incorrect Defence Mechanism	Logical Issue	Major	✓
VPL-02	Ineffectual Protection	Logical Issue	Major	✓
VPL-03	Inapplicacy of Checks-Effects-Interactions	Language Specific	Informational	✓
VPT-01	Redundant Variable Visibility	Gas Optimization	Informational	✓

VPL-01: Incorrect Defence Mechanism

Type	Severity	Location
Logical Issue	Major	VolmexProtocol.sol L112-L118

Description:

The EIP-3074 that will be included in the London fork in the summer will completely break this mechanism as a smart contract will be able to impersonate any `tx.origin` arbitrarily.

Recommendation:

We advise this particular mechanism to be thoroughly evaluated and its effects to be assimilated in the contract's design.

Alleviation:

The mechanism has since been removed from the codebase as the Volmex team assessed the ramifications of this protection and concluded that flash loans will only benefit the protocol instead of exploiting it.

VPL-02: Ineffectual Protection

Type	Severity	Location
Logical Issue	Major	VolmexProtocol.sol L101-L107, L225, L233, L266, L269, L293, L426-L430

Description:

The block protection utilized by the contracts is ineffectual as it can be easily circumvented.

Recommendation:

The position tokens of the protocol are freely transactable and as such, the block lock (which relies on `msg.sender`) can be circumvented by transferring the position tokens to a new address and performing the next call with that one. We advise the block lock mechanism to be revised and the ramifications it has to the protocol to be closely evaluated.

Alleviation:

The block lock mechanism has been removed as flash loan arbitrageurs would actually benefit the protocol instead of damaging it.

VPL-03: Inapplicacy of Checks-Effects-Interactions

Type	Severity	Location
Language Specific	Informational	VolmexProtocol.sol L334-L339

Description:

The `claimAccumulatedFees` function does not apply the Checks-Effects-Interactions pattern in that it utilizes the `accumulatedFees` storage variable for an external call before zeroing it out.

Recommendation:

Although this should have no impact as sane ERC20 implementations do not inform the recipient of a transfer, it is still advisable to apply this pattern as a matter of best practices.

Alleviation:

The variable is now properly cached to a temporary in-memory variable before being zeroed out and consequently used in the external call thereby conforming to the Checks-Effects-Interactions pattern in full.

VPT-01: Redundant Variable Visibility

Type	Severity	Location
Gas Optimization	Informational	VolmexPositionToken.sol L13

Description:

The `VOLMEX_PROTOCOL_ROLE` variable is meant to be an internally accessible variable and should not be exposed publicly.

Recommendation:

We advise the `public` specifier to be replaced by either `internal` or `private` and we also advise the `keccak256` execution to be replaced by the same value literal utilized in `VolmexIndexFactory` (`0x33ba6006595f7ad5c59211bde33456cab351f47602fc04f644c8690bc73c4e16`).

Alleviation:

The variable's visibility specifier was set to `private` according to our recommendation.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of `private` or `delete`.