CERTIK

Code Security Assessment

Volmex - AMM

Feb 1st, 2022

Table of Contents

Summary

This report has been prepared for Volmex Labs to discover issues and vulnerabilities in the source code of the Volmex - AMM project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Volmex - AMM
Platform	ethereum
Language	Solidity
Codebase	• https://github.com/volmexfinance/volmex-amm
Commit	• 1eb714db41b42f9fadc15624b81f9fae09af7ac5 • 2d44b46e46a511d72ef09cfd0d990bc7ba5d525e • 07580789ebc844f97394923f8c015c63e8e22ec6

Audit Summary

Delivery Date	Feb 01, 2022
Audit Methodology	Static Analysis, Manual Review
Key Components	AMM

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Mitigated	Resolved
● Critical	3	0	0	0	0	0	3
● Major	11	0	0	5	0	0	6
● Medium	4	0	0	2	0	0	2
● Minor	12	0	0	9	0	0	3
● Informational	8	0	0	0	0	0	8
● Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
IDF	projects/volmex/volmex-amm/contracts/interfaces/IDynamicFee.sol	f84c23255884e2b84c07dbfc6b890d7887692f00f7a3d10529c5847482189dfe
IER	projects/volmex/volmex-amm/contracts/interfaces/IERC20Modified.sol	d5051e7866452ac61b0fc8254bbf845b5ed319496da0024c621cdd731c6fc36d
IFL	projects/volmex/volmex-amm/contracts/interfaces/IFlashLoanReceiver.sol	64c85aeff39556d79cebfb0c5efe170e554ffeaef5a57e03065ce87baaf3fe76
IPP	projects/volmex/volmex-amm/contracts/interfaces/IPausablePool.sol	98d5b5c304487217711121ae9f558595ebcb1c5989cb553083b9f64d5bff7ba3
IVA	projects/volmex/volmex-amm/contracts/interfaces/IVolmexAMM.sol	112f599ca40bb7fb7bd413524381868ea6e3c184da3993ac46d35a7bf747e7c9
IVO	projects/volmex/volmex-amm/contracts/interfaces/IVolmexOracle.sol	6156f30456a3ccf4db3be34d860ebea3b040ed8a48b2ce77000e842b2da4396e
IVP	projects/volmex/volmex-amm/contracts/interfaces/IVolmexProtocol.sol	24fa5d7ac43d2d8adb87150b1d209760de51453bdec127438bf133a3a7bdf34d
IVR	projects/volmex/volmex-amm/contracts/interfaces/IVolmexRepricer.sol	4dfc1685c22710f725f82e93f3ab1520a6c22bd9337d3037b4c914c65b970694
BPD	projects/volmex/volmex-amm/contracts/libs/BokkyPooBahsDateTimeLibrary/BokkyPooBahsDateTimeLibrary.sol	db2fa55473d8f5a60c3ed089f16a3740f4608692af3c0150f5459aafe0cd4c3f
EIP	projects/volmex/volmex-amm/contracts/libs/tokens/EIP20NonStandardInterface.sol	f9e6198d302e07d8b548afa371e179f4303dbc7f0e8275e0e924558b681da422
TCK	projects/volmex/volmex-amm/contracts/libs/tokens/Token.sol	f1883581661a7ced88763412ec272ff52a6572933f29501ad486cd783ce6bfdb
TMG	projects/volmex/volmex-amm/contracts/libs/tokens/TokenMetadataGenerator.sol	aae490d368d425f06682d4c4d39074a03b055049a6d4ad490891150fbf3f9935
DFC	projects/volmex/volmex-amm/contracts/libs/DynamicFee.sol	896d757b44b1e0057eaf2926089e09a19bd5f17ee879333d1be35430ca0da74f
CCK	projects/volmex/volmex-amm/contracts/maths/Const.sol	4f0af797c7ea269f0dda0775c1a2c1707494551649e6a9b012c329572bb23e28

ID	File	SHA256 Checksum
MCK	projects/volmex/volmex-amm/contracts/maths/Math.sol	ebbdc13be4ae4f66ddeb12dd62a47c772cb87eaed20a9965ccd1eae2c304f245
NCK	projects/volmex/volmex-amm/contracts/maths/Num.sol	2dc34f01807eb3268eb0cf98fef0d6e2a0e509aa3665fe4cea3669ae66371069
NEC	projects/volmex/volmex-amm/contracts/maths/NumExtra.sol	c8bf5236f5075582c964219c090f2c38210cd049839f42a34546e69394778c76
VOC	projects/volmex/volmex-amm/contracts/oracles/VolmexOracle.sol	5793c62a486bf4f336e6e690ec2a51a361e8071321002ce054d8985545c32b6e
VSE	projects/volmex/volmex-amm/contracts/protocol/library/VolmexSafeERC20.sol	d6754809a761b797ab563eff5e29267b51cfec26fe75758fc63c2ad5e0500d67
VPT	projects/volmex/volmex-amm/contracts/protocol/VolmexPositionToken.sol	a6459015dc936ef3021988a2623e4f1aca959f420ea066b9c23c74d5e7262116
VPC	projects/volmex/volmex-amm/contracts/protocol/VolmexProtocol.sol	247f49a321be371b06ab4cdb1a1f3f3a1eb861f8bbef65948293b074bd90b852
VRC	projects/volmex/volmex-amm/contracts/repricers/VolmexRepricer.sol	abca43f8ca6d1626d33588cbca0e2c0fa2c23519d2e0c4da5276823763765288
VAM	projects/volmex/volmex-amm/contracts/VolmexAMM.sol	2baad59527d6e75fafd415691a05833c34ffe51658836d80395c0448eb6d00c4
VAR	projects/volmex/volmex-amm/contracts/VolmexAMMRegistry.sol	90446ffaef8964e97d474dd5053736bcbdad19b4cc3dcee387758ac50872350c
VAV	projects/volmex/volmex-amm/contracts/VolmexAMMView.sol	dd7b0c10b078a1a8682e2f6b9236dcff9aa911a73485f48c4ef3659cb1c6c188
VCC	projects/volmex/volmex-amm/contracts/VolmexController.sol	66e038b0f4d3b7ecbcca39de9577e1df687754af17789ba580f98ae5b471b715
IEI	projects/volmex/volmex-amm/contracts/interfaces/IEIP20NonStandard.sol	7d6520e07b2f889db35b2491999c87b270ea8024c95bd7d1c7cbc8bd0ea4a1aa
IEC	projects/volmex/volmex-amm/contracts/interfaces/IERC20.sol	dca510b1b2c32c98356ad1cf1ce3053727e842663bd80ef9ee2547fbaf53212d
IEM	projects/volmex/volmex-amm/contracts/interfaces/IERC20Modified.sol	b5888d562e118c497c8a64fe30217d3de48c7966f3c09786b342354247642840
IFR	projects/volmex/volmex-amm/contracts/interfaces/IFlashLoanReceiver.sol	a29d44de19cf906e1cec27573676fcc706f0150d644458d3d332ab128991faa0

ID	File	SHA256 Checksum
IPC	projects/volmex/volmex-amm/contracts/interfaces/IPausablePool.sol	649eea159e14779fee5aed31fdb62339e8c43f1f0747a8a3b54c413160a5df98
IVC	projects/volmex/volmex-amm/contracts/interfaces/IVolmexController.sol	3d7d753d10180347c530d9ce9fd7b606ea3ba54f1f5e6f5ba686dbe21b5a619c
IVK	projects/volmex/volmex-amm/contracts/interfaces/IVolmexOracle.sol	4e0605d610e45254a8633c4dc56d1ad73652fb22b6370ff764def570916e956a
IPK	projects/volmex/volmex-amm/contracts/interfaces/IVolmexPool.sol	8be88ed8dea67a2c9bf0eb9b65a716f9d015f840f6c635575167eee73b2fdd3a
IVV	projects/volmex/volmex-amm/contracts/interfaces/IVolmexPoolView.sol	247f9e12dca004337a93bbb2adc0787c1ede2aa31a43e125033c527cf07d385f
ICK	projects/volmex/volmex-amm/contracts/interfaces/IVolmexProtocol.sol	a1f11c14daa558681f24daa10ae11c98485d70ecf2dd3f0f46e0f6cf7698c182
IRC	projects/volmex/volmex-amm/contracts/interfaces/IVolmexRepricer.sol	4f320251d70ea4d2e6364372bf671e29b1427176efbdedbc637612b1a39f2578
TCP	projects/volmex/volmex-amm/contracts/libs/tokens/Token.sol	8ac9044f0c6024e88ca15dd16032eddb5e35fea76be3b9b95ea65ff288b2f7e5
TMC	projects/volmex/volmex-amm/contracts/libs/tokens/TokenMetadataGenerator.sol	98d70821ef9673f27ce2e95627118f88aa06d0c2123d8c78fd2549bd6dd94eee
CCP	projects/volmex/volmex-amm/contracts/maths/Const.sol	f444e4dc82ab20fa0c5a21a4979ec4d267e3c00b7c731863dbdc9c93f30e12e1
MCP	projects/volmex/volmex-amm/contracts/maths/Math.sol	6cec196aef75ed0ceabe3a28fb65408aba76864f07b305ecede3178395761e99
NCP	projects/volmex/volmex-amm/contracts/maths/Num.sol	37a0c3f04ecb983c7a7318122bd6ac6f11d29c22298227d7ffddde651e37ef7e
NEK	projects/volmex/volmex-amm/contracts/maths/NumExtra.sol	90e4e1c0e815faf28b0ad4ea3ac7632f3d1ef8ec86cdce9d78244b985a9e6343
VOK	projects/volmex/volmex-amm/contracts/oracles/VolmexOracle.sol	496f46aebed94d1c0ce9f1b767603b5816ce2a89db2356c4b2c415eef9504bd8
TCT	projects/volmex/volmex-amm/contracts/protocol/TestCollateralToken.sol	b926af85900b341898df7a4044d2f9237cf77ab432efc70cabd435e3e2cd45c1

ID	File	SHA256 Checksum
VPK	projects/volmex/volmex-amm/contracts/protocol/VolmexPositionToken.sol	6bdf7189fdf9bdc492d17d3f36214f9d7040cb28504eb173713da75a30d9bfd5
VPP	projects/volmex/volmex-amm/contracts/protocol/VolmexProtocol.sol	f32c300071fd3a2fcba43b8b0b78544bbb75deba46b17817dd5b58019921e17c
VPW	projects/volmex/volmex-amm/contracts/protocol/VolmexProtocolWithPrecision.sol	4f1e6a584f9a8fed4b23564120ea448d77e445271773f111934b641e0e09b421
VRK	projects/volmex/volmex-amm/contracts/repricers/VolmexRepricer.sol	6b6b448905eefe88c47cba898df3da7af3af9673d46838a4ab1c9e3d2e48d1fc
VCK	projects/volmex/volmex-amm/contracts/VolmexController.sol	c27ddd1a4b6e019551b42be77085a8bc43af9ff7ae7d801223da602a4147ca08
VCP	projects/volmex/volmex-amm/contracts/VolmexPool.sol	03ba75a899df5fbf81e6109715cda91d4d32f48c5ffb15ceaac711c2c1e82a2a
VPV	projects/volmex/volmex-amm/contracts/VolmexPoolView.sol	4ba3bf2a8d62b6bf510fbb3368df67855aa457dbdd592a84cb597772b87eeaab

Review Notes

External Dependencies

There are a few depending injection contracts or addresses in the current project:

- `stablecoin`, `_pool` and `_protocol` for the contract `VolmexController`;
- `_pool` for the contract `VolmexAMMView`;
- `_newPool` for the contract `VolmexAMMRegistry`;
- `protocol` and `repricer` for the contract `VolmexAMM`;
- `oracle` and `protocol` for the contract `VolmexRepricer`;
- `volatilityToken`, `inverseVolatilityToken` and `collateral` for the contract `VolmexProtocol`.

We assume these contracts or addresses are valid and non-vulnerable actors and implement proper logic to collaborate with the current project.

Privilledged Functions

To set up the project correctly, improve overall project quality and preserve upgradability, the following roles are adopted in the codebase.

In the contract `VolmexPositionToken`, the role `VOLMEX_PROTOCOL_ROLE` has the authority over the following functions:

- `mint()` to mint new tokens to arbitrary accounts;
- `burn()` to burn new tokens from arbitrary accounts;
- `pause()` to pause the whole contract;
- `unpause()` to unpause the whole contract.

In the contract `VolmexProtocol`, the role `owner` has the authority over the following functions:

- `toggleActive()` to toggle the active variable;
- `updateMinimumCollQty()` to update the `minimumCollateralQty`;
- `updateVolatilityToken()` to update the volatility token;
- `settle()` to settle the contract, preventing new minting and providing individual token redemption;
- `recoverTokens()` to recover tokens accidentally sent to this contract;
- `updateFees()` to update the percentage of `issuanceFees` and `redeemFees`;
- `claimAccumulatedFees()` to safely transfer the accumulated fees to owner;
- `togglePause()` to Pause/unpause volmex position token.

In the contract `VolmexOrale`, the role `owner` has the authority over the following functions:

- `updateVolatilityTokenPrice()` to update volatility token price by index;
- `updateVolatilityTokenPriceBySymbol()` to update volatility token price by symbol;
- `addVolatilityTokenPrice()` to update volatility token price.

In the contract `VolmexAMMRegistry`, the role `owner` has the authority over the following functions:

- `pausePool()` to pause the pool;
- `unpausePool()` to unpause the pool;
- `collect()` to transfer token in AMM to the owner.

In the contract `VolmexController`, the role `owner` has the authority over the following functions:

- `setPoolAndProtocol()` to set pool and protocol;
- `updateMinCollateralQty()` to update minimum collateral quantity.

In the contract `Owner`, the role `owner` has the authority over the following functions:

- `transferOwnership()` to transfer ownership;
- `renounceOwnership()` to renounce ownership.

In the contract `VolmexAMM`, the role `owner` has the authority over the following functions:

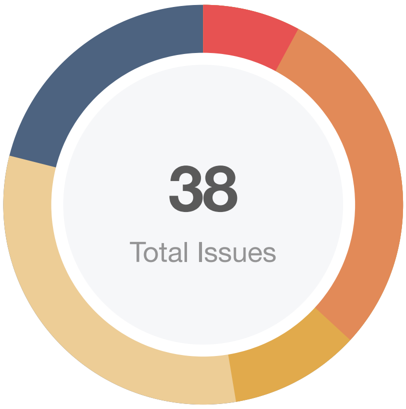
- `setController()` to set controller of the AMM;
- `finalize()` to finalize the pool.
- `updateFlashLoanPremium()` to update the flash loan premium percent;
- `pause()` to pause the contract;
- `unpause()` to unpause the contract, if paused.

The role `controller` has the authority over the following functions:

- `flashLoan()` to make flashloan;
- `joinPool()` to add liquidity to the pool;
- `exitPool()` to remove liquidity from the pool;
- `swapExactAmountIn()` to swap the pool asset.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of timelock contract.

Findings



🟥 **Critical**	**3** (7.89%)
🟧 **Major**	**11** (28.95%)
🟨 **Medium**	**4** (10.53%)
🟨 **Minor**	**12** (31.58%)
🟦 **Informational**	**8** (21.05%)
🟩 **Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Incompatibility With Deflationary Token	Volatile Code	● Minor	ⓘ Acknowledged
GLOBAL-02	Potential Integer Overflow and Underflow	Mathematical Operations	● Minor	✓ Resolved
GLOBAL-03	Front Running Risk	Logical Issue	● Minor	ⓘ Acknowledged
VAM-01	Missing Access Control	Logical Issue	● Critical	✓ Resolved
VAM-02	Incorrect Fee Caculation	Logical Issue	● Critical	✓ Resolved
VAM-03	Incorrect Usage of Memory	Logical Issue	● Critical	✓ Resolved
VAM-04	Centralization Risk	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
VAM-05	Potential Reentrancy Attack	Logical Issue	● Medium	✓ Resolved
VAM-06	Lack of Handling Return Value	Logical Issue	● Minor	ⓘ Acknowledged
VAM-07	Lack of Input Validation	Volatile Code	● Minor	✓ Resolved
VAM-08	Redundant Condition	Coding Style	● Informational	✓ Resolved
VAM-09	Typo in Comment	Coding Style	● Informational	✓ Resolved
VAM-10	Lack of Event Emissions for Significant Transaction	Coding Style	● Informational	✓ Resolved

ID	Title	Category	Severity	Status
VAR-01	Centralization Risk	**Centralization / Privilege**	● **Major**	✓ Resolved
VAR-02	Missing Access Control	Logical Issue	● Major	✓ Resolved
VCC-01	Centralization Risk	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
VCC-02	Incorrect Parameter in `_approveAssets()`	Logical Issue	● Major	✓ Resolved
VCC-03	Potential Redemption Failure	Logical Issue	● Major	✓ Resolved
VCC-04	Lack of Handling Return Value	Logical Issue	● Minor	ⓘ Acknowledged
VCC-05	Lack of Check for Protocols	Volatile Code	● Minor	✓ Resolved
VCC-06	Lack of Event Emissions for Significant Transaction	Coding Style	● Informational	✓ Resolved
VCC-07	Logic of Setting Minimum Token Amount Out	Logical Issue	● Medium	ⓘ Acknowledged
VCK-01	Unchecked Token Decimals	Logical Issue	● Major	✓ Resolved
VCK-02	Lack of Input Validation on `_tokenIn` and `_isInverse`	Logical Issue	● Medium	✓ Resolved
VCK-03	Lack of Input Validation	Volatile Code	● Informational	✓ Resolved
VCK-04	Lack of Handling Return Value	Logical Issue	● Minor	ⓘ Acknowledged
VCK-05	Preview for Swap and Burn	Logical Issue	● Informational	✓ Resolved
VCP-01	Unsafe Casting from `uint256` to `int256`	Mathematical Operations	● Minor	ⓘ Acknowledged
VCP-02	Users Unable to Join Pool When `poolTotal = 0`	Logical Issue	● Minor	ⓘ Acknowledged
VOC-01	Centralization Risk	Logical Issue	● Major	ⓘ Acknowledged
VOC-02	Inaccurate Error Message	Logical Issue	● Informational	✓ Resolved

ID	Title	Category	Severity	Status
VPC-01	Centralization Risk	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
VPC-02	Potential Reentrancy Attack	Logical Issue	● Medium	ⓘ Acknowledged
VPC-03	Lack of Handling Return Value	Logical Issue	● Minor	ⓘ Acknowledged
VPP-01	Potentially Incorrect Decimal Assumption	Logical Issue	● Major	✓ Resolved
VPP-02	Lack of Handling Return Value	Logical Issue	● Minor	ⓘ Acknowledged
VPP-03	Incomplete Function	Logical Issue	● Informational	✓ Resolved
VPT-01	Centralization Risk	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged

GLOBAL-01 | Incompatibility With Deflationary Token

Category	Severity	Location	Status
Volatile Code	● Minor	Global	ⓘ Acknowledged

Description

When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee. As a result, an inconsistency in the amount will occur and the transaction may fail due to the validation checks.

For example, calling the function `VolmexController.swapCollateralToVolatility()` will swap `_amount` stablecoin to volatility token. However, the function does not check the incoming balance of the stablecoin. If the stablecoin is a deflationary token and the user input 100 tokens. The contract will only receive 90 tokens, but the contract will try to swap 100 stablecoin to the corresponding volatility token. This will cause unexpected errors.

Recommendation

We advise the client to regulate the set of tokens supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

Alleviation

[**Volmex Team**]: We only use DAI and USDC.

GLOBAL-02 | Potential Integer Overflow And Underflow

Category	Severity	Location	Status
Mathematical Operations	● Minor	Global	✓ Resolved

Description

Integer overflow and underflow might happen in integer operations if the Solidity version is lower than 0.8.0 and the `SafeMath` library is not used. The following contracts are vulnerable to integer overflow and underflow:

- `DynamicFee`
- `VolmexAMM`
- `VolmexProtocol`
- `VolmexOracle`

Recommendation

We advise the client to use OpenZeppelin's SafeMath library for all of the mathematical operations or use Solidity 0.8.x.

Alleviation

The Volmex team heeded our advice and resolved this issue in the commit f1b9cb2a87dba73c603f2bb9d4affdefc76e5647 by setting the compiler version to v0.8.11.

GLOBAL-03 | Front Running Risk

Category	Severity	Location	Status
Logical Issue	● Minor	Global	ⓘ Acknowledged

Description

The function `initialize()` is used to init the whole contract. This function can only be called once and should be called immediately after deployment by the deployer. However Ethereum does not support to execute multiple transactions together. Thus hacker may front run the `initialize()` call and init the contract maliciously.

Recommendation

We recommend checking the execution result of calling the function `initialize()`.

Alleviation

[**Volmex Team**]: We deploy our contracts using OpenZeppelin upgrades plugin so our contracts are deployed and initialized in a single transaction.

VAM-01 | Missing Access Control

Category	Severity	Location	Status
Logical Issue	● Critical	projects/volmex/volmex-amm/contracts/VolmexAMM.sol (base): 465	✓ Resolved

Description

The function `finalize()` finalizes the volatility tokens and updates the state variable `_finalized`. The comments of the aforementioned function indicate that this function should only be called by the role `controller`. However, this function does not have proper access controls.

Recommendation

We recommend enforcing modifier `onlyController()` to the aforementioned function.

Alleviation

The Volmex team heeded our advice and resolved this issue by adding the modifier `onlyController()` to the function `finalize()`. The fixing is reflected in the commit `2d44b46e46a511d72ef09cfd0d990bc7ba5d525e`.

VAM-02 | Incorrect Fee Caculation

Category	Severity	Location	Status
Logical Issue	● Critical	projects/volmex/volmex-amm/contracts/VolmexAMM.sol (base): 892	Resolved

Description

The fee is calculated with the following code:

```
890  fee =
891                      _baseFee +
892                      (((_feeAmp) * (_spow3(_expEnd) - _spow3(expStart))) * iBONE) /
893                      (3 * (_expEnd - expStart));
```

In the caculation `(((_feeAmp) ∗ (_spow3(_expEnd) − _spow3(expStart))) ∗ iBONE) /(3 ∗ (_expEnd − expStart))`, an execess `iBONE` is multiplied. All elements in the caculaiton has the correct decimal. Thus there is no need to multiply one more `iBONE`.

Recommendation

We recommend removing the multiplier `iBONE`.

Alleviation

The Volmex team heeded our advice and resolved this issue by removing the redundant `iBONE` in the commit cf47e11b270ad9a8407312de16b9bb937acd813e.

VAM-03 | Incorrect Usage Of Memory

Category	Severity	Location	Status
Logical Issue	● Critical	projects/volmex/volmex-amm/contracts/VolmexAMM.sol (base): 725, 727	✓ Resolved

Description

The function `_updateLeverages()` is used to update the leverage in the record. However, the input variables `inToken` and `outToken` are declared as `memory`:

```
724      function _updateLeverages(
725          Record memory inToken,
726          uint256 tokenAmountIn,
727          Record memory outToken,
728          uint256 tokenAmountOut
729      ) internal pure {
730          ...
```

Although `inToken` and `outToken` are updated in the function, `inRecord` and `outRecord`, which are in the `storage` type, are not updated accordingly:

```
627          Record storage inRecord = _records[tokenIn];
628          Record storage outRecord = _records[tokenOut];
```

```
641          _updateLeverages(inRecord, tokenAmountIn, outRecord, tokenAmountOut);
```

This may lead to an incorrect output amount during swapping.

Recommendation

We recommend using `storage` instead of `memory` type for the aforementioned function parameters.

Alleviation

The Volmex team heeded our advice and resolved this issue in the commit 26c4b1f1fe666015c2d5a986bbe03388b9c318c2.

VAM-04 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	projects/volmex/volmex-amm/contracts/VolmexAMM.sol (base): 272, 306, 342, 380, 250, 260, 958, 965	ⓘ Acknowledged

Description

In the contract `VolmexAMM`, the role `owner` has the authority over the following functions:

- `setController()` to set controller of the AMM;
- `updateFlashLoanPremium()` to update the flash loan premium percent;
- `pause()` to pause the contract;
- `unpause()` to unpause the contract, if paused.

The role `controller` has the authority over the following functions:

- `flashLoan()` to make flashloan;
- `joinPool()` to add liquidity to the pool;
- `exitPool()` to remove liquidity from the pool;
- `swapExactAmountIn()` to swap the pool asset.

[0x5b5961e2da9f83738de98c0716adde34fb641049 Update]:

The role `controller` has the authority over the following functions:

- `finalize()` to finalize the pool.

Any compromise to the `owner` and `controller` accounts may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `owner` and `controller` accounts' private keys to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[**Volmex Team**]: We will be shifting privileged operations to Volmex core Multisig immediately and eventually moving to governance.

VAM-05 | Potential Reentrancy Attack

Category	Severity	Location	Status
Logical Issue	● Medium	projects/volmex/volmex-amm/contracts/VolmexAMM.sol (base): 272	✓ Resolved

Description

A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the external call resolved the effects.

The function `VolmexAMM.flashLoan()` has external calls before state updates, so it is vulnerable to reentrancy attacks.

Recommendation

We recommend applying the modifier `_lock_` for the aforementioned function to prevent reentrancy attacks.

Alleviation

The Volmex team heeded our advice and resolved this issue in the commit 85ec5ef226d2f0fec0bd18f20de0391b23f82e0c.

VAM-06 | Lack Of Handling Return Value

Category	Severity	Location	Status
Logical Issue	● Minor	projects/volmex/volmex-amm/contracts/VolmexAMM.sol (base): 279, 291, 326, 675, 835	ⓘ Acknowledged

Description

Functions `transfer()`, `transferFrom()` and `_pullUnderlying()` are not void-returning functions. Ignoring their return values, especially when their first return value represents the status if the transaction is executed successfully, might cause unexpected exceptions.

Recommendation

We recommend handling return values of the functions at the aforementioned line before continuing processing.

Alleviation

[**Volmex Team**]: This has been fixed for `_pullUnderlying()` and doesn't need to be changed for `transfer()` and `trasnferFrom()`, because the system only uses volatility tokens deployed by Volmex Labs.

[**CertiK**]: For `_pullUnderlying()`, the return value is still not handled; for `transfer()` and `trasnferFrom()`, the return value handling would not be required if they are guaranteed to be reverted upon failure. However, considering possible contract updates, we would still recommend handling return values of the aforementioned functions.

VAM-07 | Lack Of Input Validation

Category	Severity	Location	Status
Volatile Code	● Minor	projects/volmex/volmex-amm/contracts/VolmexAMM.sol (base): 260	✓ Resolved

Description

The input parameter `_premium` updates the flashloan fee and it is not properly validated. It should never be greater than 10000.

Recommendation

We recommend enforcing appropriate range for the input parameter `_premium`.

Alleviation

The Volmex team heeded our advice and resolved this issue in the commit 9a63474659bcecbabf176872165a7713dc4d1b7d

VAM-08 | Redundant Condition

Category	Severity	Location	Status
Coding Style	● Informational	projects/volmex/volmex-amm/contracts/VolmexAMM.sol (base): 499	✓ Resolved

Description

The variable `collateralDecimals` is an `uint256` and it is always a non-negative number. Thus the condition `collateralDecimals >= 0` is always true.

Recommendation

We recommend removing the aforementioned condition.

Alleviation

The Volmex team heeded our advice and resolved this issue in the commit 3fd73694aff440f80d7e594457784fae2ef7c1e1.

VAM-09 | Typo In Comment

Category	Severity	Location	Status
Coding Style	● Informational	projects/volmex/volmex-amm/contracts/VolmexAMM.sol (base): 956	✓ Resolved

Description

The comment `@notice Used to puase the contract` contains a typo, namely `puase`.

Recommendation

We recommending fixing the typo by using `pause`.

Alleviation

The Volmex team heeded our advice and resolved this issue in the commit 46fff53eca003ed15daf8ea0035fd6b83e802299.

VAM-10 | Lack Of Event Emissions For Significant Transaction

Category	Severity	Location	Status
Coding Style	● Informational	projects/volmex/volmex-amm/contracts/VolmexAMM.sol (base): 260	✓ Resolved

Description

The following function affects the status of sensitive state variables and should be able to emit events as notifications:

- `updateFlashLoanPremium()` to update the flash loan premium percent.

Recommendation

Consider adding events for sensitive actions and emit them in the aforementioned function.

Alleviation

The Volmex team heeded our advice and resolved this issue in the commit adc844b7bbd16c7d6dd3a2eeb3ddb1e89ba0d0e9.

VAR-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	projects/volmex/volmex-amm/contracts/VolmexAMMRegistry.sol (base): 39, 43, 47	✓ Resolved

Description

In the contract `VolmexAMMRegistry`, the role `owner` has the authority over the following functions:

- `pausePool()` to pause the pool;
- `unpausePool()` to unpause the pool;
- `collect()` to transfer token in AMM to the owner.

Any compromise to the `owner` account may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

In the codebase with the commit hash `2d44b46e46a511d72ef09cfd0d990bc7ba5d525e`, the contract `VolmexAMMRegistry` has been removed.

VAR-02 | Missing Access Control

Category	Severity	Location	Status
Logical Issue	● Major	projects/volmex/volmex-amm/contracts/VolmexAMMRegistry.sol (base): 30	✓ Resolved

Description

The function `registerNewPool()` registers new pools. Calling this function updates state variables `index`, `_isPool` and `_pools`. Anyone can call this function to register new pools. We would like to check with the Volmex team whether this is an intended design.

Recommendation

We recommend setting a proper access control for the function `registerNewPool()` if it is not an intended design.

Alleviation

In the codebase with the commit hash `2d44b46e46a511d72ef09cfd0d990bc7ba5d525e`, the contract `VolmexAMMRegistry` has been removed.

VCC-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	projects/volmex/volmex-amm/contracts/VolmexController.sol (base): 63, 72	ⓘ Acknowledged

Description

In the contract `VolmexController`, the role `owner` has the authority over the following functions:

- `setPoolAndProtocol()` to set pool and protocol;
- `updateMinCollateralQty()` to update minimum collateral quantity.

[0x5b5961e2da9f83738de98c0716adde34fb641049 Update]: the role `owner` has the authority over the following functions:

- `addPool()` to add a pool;
- `addStableCoin()` to add a stable coin;
- `addProtocol()` to add a Volmex protocol;
- `pausePool()` to pause a pool;
- `unpausePool()` to unpause a pool;
- `collect()` to collect pool share from the controller;
- `finalizePool()` to finalize a pool.

[0x21b4bb30a7d2121a7f5a173b5b53d2927d74dc03 Update]: the role `owner` has the authority over the following functions:

- `updateAdminFee()` to update the admin fee;
- `updateVolatilityIndex()` to update the volatility index.

Any compromise to the `VolmexController` account may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[**Volmex Team**] (2022/01/07): We will be shifting privileged operations to Volmex core Multisig immediately and eventually moving to governance.

VCC-02 | Incorrect Parameter In `_approveAssets()`

Category	Severity	Location	Status
Logical Issue	● Major	projects/volmex/volmex-amm/contracts/VolmexController.sol (base): 267~268	✓ Resolved

Description

In the function `addLiquidity()`, the contract approves tokens `volatilityToken` and `inverseVolatilityToken` and then add them to the pool. However, the approvals do not guarantee sufficient allowances before executing `_pool.joinPool(_poolAmountOut, _maxAmountsIn, msg.sender)`:

```
267         _approveAssets(_protocol.volatilityToken(), _maxAmountsIn[0], msg.sender, address(_pool));
268         _approveAssets(_protocol.inverseVolatilityToken(), _maxAmountsIn[1], msg.sender, address(_pool));
```

When calling `_approveAssets()` and the allowance is not large enough, the function does not update the allowance from `_owner` to `_spender`. Instead, it updates the allowance from the contract address to `_spender`:

```
    function _approveAssets(
        IERC20Modified _token,
        uint256 _amount,
        address _owner,
        address _spender
    ) internal {
        uint256 _allowance = _token.allowance(_owner, _spender);

        if (_amount <= _allowance) return;

        _token.approve(_spender, _amount);
    }
```

Thus caller's assets might not be able to be transferred to the pool due to insufficient allowance.

Recommendation

We advise the Volmex team to reconsider the workflow of adding liquidity. If the allowance should be guaranteed by users, the lines 267 and 268 are unnecessary; if the allowance should be controlled by the contract, users should be required to send funds to the contract and `_pool` should not pull funds from `msg.sender`.

Alleviation

The Volmex team heeded our advice and resolved this issue by removing unnecessary approvals. The fixing is reflected in the commit `2d44b46e46a511d72ef09cfd0d990bc7ba5d525e`.

VCC-03 | Potential Redemption Failure

Category	Severity	Location	Status
Logical Issue	● Major	projects/volmex/volmex-amm/contracts/VolmexController.sol (base): 188, 221	✓ Resolved

Description

To redeem the collateral, the protocol requires the ratio of the volatility token and inverse volatility token to be 1.

For example, the user input the volatility token. Then the controller will swap half of the volatility token to inverse volatility token and try to redeem the collateral with the amount of inverse volatility token.

However, if the amount of the volatility token is less than the amount of the inverse volatility token, the redeem may fail, because the redeem process requires the token ratio to be 1 and there is not enough volatility token.

Moreover, since `tokenAmountOut` is not guaranteed to be less than `_amount >> 1`, the functions with the operation `(_amount >> 1).sub(tokenAmountOut)` will revert if `tokenAmountOut` is large.

Recommendation

We recommend checking `_amount` and `tokenAmountOut` to set a reasonable amount for redemption.

Alleviation

In the codebase with the commit hash `2d44b46e46a511d72ef09cfd0d990bc7ba5d525e`, the aforementioned logic has been removed.

VCC-04 | Lack Of Handling Return Value

Category	Severity	Location	Status
Logical Issue	● Minor	projects/volmex/volmex-amm/contracts/VolmexController.sol (base): 93, 159, 170, 207, 326, 352, 380	ⓘ Acknowledged

Description

The functions `approve()`, `transferFrom()` and `transfer()` are not void-returning functions. Ignoring their return values, especially when their first return value represents the status if the transaction is executed successfully, might cause unexpected exceptions.

Recommendation

We recommend handling return values of function `approve()`, `transferFrom()` and `transfer()` at the aforementioned line before continuing processing.

Alleviation

[**Volmex Team**]: Since we use standard OpenZeppellin tokens, we don't need to implement this because they throw errors instead of returning a false value whenever happens.

[**CertiK**]: For `approve()`, `transfer()` and `trasnferFrom()`, the return value handling would not be required if they are guaranteed to be reverted upon failure. However, considering possible contract updates, we would still recommend handling return values of the aforementioned functions.

VCC-05 | Lack Of Check For Protocols

Category	Severity	Location	Status
Volatile Code	Minor	projects/volmex/volmex-amm/contracts/VolmexController.sol (base): 54, 63	Resolved

Description

The contract `VolmexController` works with the token `stablecoin`. When setting a new protocol, the contract does not check whether the new protocol accepts `stablecoin` as collateral. If the new protocol does not support `stablecoin`, this may cause failure when the contract `VolmexController` calls `protocol.collateralize()`:

```
        stablecoin.transferFrom(msg.sender, address(this), _amount);
        _approveAssets(stablecoin, _amount, address(this), address(_protocol));

        _protocol.collateralize(_amount);
```

Recommendation

We recommend checking whether the new protocol accepts `stablecoin` as collateral when adding protocols.

Alleviation

The Volmex team heeded our advice and resolved this issue in the commit 67474857cd6e6f5244db3173230c435dd81ebb56

VCC-06 | Lack Of Event Emissions For Significant Transaction

Category	Severity	Location	Status
Coding Style	● Informational	projects/volmex/volmex-amm/contracts/VolmexController.sol (base): 63, 72	✓ Resolved

Description

The following functions affect the status of sensitive state variables and should be able to emit events as notifications:

- `setPoolAndProtocol()` to set pool and protocol;
- `updateMinCollateralQty()` to update minimum collateral quantity.

Recommendation

Consider adding events for sensitive actions and emit them in the aforementioned functions.

Alleviation

The Volmex team heeded our advice and resolved this issue by adding the event `PoolAdded` and removing the function `updateMinCollateralQty()`. The update is reflected in the commit `2d44b46e46a511d72ef09cfd0d990bc7ba5d525e`.

VCC-07 | Logic Of Setting Minimum Token Amount Out

Category	Severity	Location	Status
Logical Issue	● Medium	projects/volmex/volmex-amm/contracts/VolmexController.sol (base): 113 , 122, 167, 245	ⓘ Acknowledged

Description

When calling the function `_swap()`, the minimum token amount out is set to 1/10 or 1/2 with the following code:

```
108             tokenAmountOut = _swap(
109                 _pool,
110                 address(_protocol.volatilityToken()),
111                 volatilityAmount,
112                 address(_protocol.inverseVolatilityToken()),
113                 volatilityAmount >> 1
114             );
```

```
117             tokenAmountOut = _swap(
118                 _pool,
119                 address(_protocol.inverseVolatilityToken()),
120                 volatilityAmount,
121                 address(_protocol.volatilityToken()),
122                 volatilityAmount >> 1
123             );
```

```
162             tokenAmountOut = _swap(
163                 _pool,
164                 address(_protocol.volatilityToken()),
165                 _amount >> 1,
166                 address(_protocol.inverseVolatilityToken()),
167                 _amount.div(10)
```

```
173             tokenAmountOut = _swap(
174                 _pool,
175                 address(_protocol.inverseVolatilityToken()),
176                 _amount >> 1,
177                 address(_protocol.volatilityToken()),
178                 _amount.div(10)
179             );
```

```
        uint256 tokenAmount = _swap(
            _pool,
            address(_tokenIn),
            _amountIn >> 1,
            _pool.getPrimaryDerivativeAddress() == address(_tokenIn)
                ? _pool.getComplementDerivativeAddress()
                : _pool.getPrimaryDerivativeAddress(),
            _amountIn.div(10)
        );
```

```
        uint256 tokenAmountOut = _swap(
            _pool,
            _pool.getPrimaryDerivativeAddress() == address(_tokenOut)
                ? _pool.getComplementDerivativeAddress()
                : _pool.getPrimaryDerivativeAddress(),
            _volatilityAmount >> 1,
            _tokenOut,
            _volatilityAmount.div(10)
        );
```

A sandwich attack might happen when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (before the transaction is attacked) a transaction to purchase one of the assets and make profits by backrunning (after the transaction is attacked) a transaction to sell the asset.

The following functions are called without setting appropriate restrictions on slippage or minimum output amount, so transactions triggering these functions are vulnerable to sandwich attacks, especially when the input amount is large:

- `swapCollateralToVolatility()`
- `swapAssets()`

On the other hand, when the volatility is large, the token amount out may be less than 1/10 or 1/2. For example, in Oct. 2021, the volatility index is 80. This means the price of volatility is 80 and the inverse volatility is 170. Then with 100 volatility tokens, the output is 47, which is less than half of the input amount. In this scenario, the swap may fail.

We would like to double-check with the Volmex Team whether the minimum token amount out is set appropriately.

Recommendation

We recommend setting a proper minimum token out amount when performing token swaps.

Alleviation

In the codebase with the commit hash `2d44b46e46a511d72ef09cfd0d990bc7ba5d525e`, the project triggers `pool.getTokenAmountOut()` to provide minimum token out amounts. If `pool` refers to the contract `VolmexPool`, the function `pool.getTokenAmountOut()` might be affected by short-time balance changes of the `pool` account, so the swaps are still vulnerable to sandwich attacks.

VCK-01 | Unchecked Token Decimals

Category	Severity	Location	Status
Logical Issue	● Major	projects/volmex/volmex-amm/contracts/VolmexController.sol (2022/1/13): 286	✓ Resolved

Description

The stablecoins may have decimal precision other than 18 digits, therefore, the mapping of `precisionRatios` is designed to track the precision ratios.

In the implementations of `VolmexController.swapCollateralToVolatility()`, the decimal is not checked before passing it into `VolmexProtocol.collateralize()`.

Recommendation

The audit team recommend adding decimal checking before calling `VolmexProtocol.collateralize()`.

Alleviation

[**Volmex Team**]: _protocol is retrieved using both poolIndex and stableCoinIndex.

The protocol contract deployed with decimals less than 18 is: https://github.com/volmexfinance/volmex-amm/blob/master/contracts/protocol/VolmexProtocolWithPrecision.sol

- It handles token decimal precision during collateralize

Also, we have used precision ratio in our calculations within AMM using which helps us calculate current return values for different decimal stable coins: https://github.com/volmexfinance/volmex-amm/blob/2d44b46e46a511d72ef09cfd0d990 bc7ba5d525e/contracts/VolmexController.sol#L778

Unit tests for 6 decimal USDC: https://github.com/volmexfinance/volmex-amm/blob/2d44b46e46a511d72ef09cfd0d990 bc7ba5d525e/test/Controller.test.ts#L708

VCK-02 | Lack Of Input Validation On `_tokenIn` And `_isInverse`

Category	Severity	Location	Status
Logical Issue	● Medium	projects/volmex/volmex-amm/contracts/VolmexController.sol (2022/1/13): 688~693	✓ Resolved

Description

The function `VolmexController.getVolatilityToCollateral()` fails to check whether the `_tokenIn` matches the bool `_isInverse`.

If these two inputs don't match, the `fee` and `amount` calculated from `VolmexController._getSwappedAssetAmount()` would not be correct since the `swapAmount` on Line 848 that calculated from `_volatilityAmountToSwap()` is singly based on the assumption that `_isInverse` is provided correctly. Furthermore, the fee is singly relying on the `_tokenIn`. The mismatch of the `_tokenIn` and the bool `_isInverse` would result in incorrect fees and swap amounts.

Recommendation

The audit team recommend adding input validations on `_tokenIn` and `_isInverse`.

Alleviation

The Volmex team heeded our advice and resolved this issue by calculating the boolean inside the function. The fixing is reflected in the commit `3cfee515d7ea11c674dc40175e6083fb8f2b1e3f`

VCK-03 | Lack Of Input Validation

Category	Severity	Location	Status
Volatile Code	● Informational	projects/volmex/volmex-amm/contracts/VolmexController.sol (2022/1/13): 591~597, 616~622	✓ Resolved

Description

The input parameters `_assetToken` and `_poolIndex` in the function `makeFlashLoan()` lack validations to check whether the `_assetToken` is supported by the pool with `_poolIndex` to borrow from. In addition, it fails to check the existence of the pool.

Similarly, the function `swap()` also lacks input validations on the parameters, such as `_poolIndex`, `_tokenIn`, and `_tokenOut`.

Recommendation

The audit team recommend enforcing appropriate validation for the aforementioned input parameters.

Alleviation

[**Volmex Team**]: These validations will be added on the frontend to reduce execution costs, during normal successful execution scenarios. If a user still inputs an incorrect param, it will result in a failed tx anyway.

VCK-04 | Lack Of Handling Return Value

Category	Severity	Location	Status
Logical Issue	● Minor	projects/volmex/volmex-amm/contracts/VolmexController.sol (2022/1/13): 282, 409, 468, 641, 798, 811	ⓘ Acknowledged

Description

Functions `transfer()`, `transferFrom()`, and `approve()` are not void-returning functions. Ignoring their return values, especially when their first return value represents the status if the transaction is executed successfully, might cause unexpected exceptions.

Recommendation

The auditing team recommend handling return values of the functions `transfer()`, `transferFrom()`, and `approve()` at the aforementioned line before continuing processing.

Alleviation

[**Volmex Team**]: Since we use standard OpenZeppellin tokens, we don’t need to implement this.

VCK-05 | Preview For Swap And Burn

Category	Severity	Location	Status
Logical Issue	● Informational	projects/volmex/volmex-amm/contracts/VolmexController.sol (2022/1/13): 359~363	✓ Resolved

Description

In the function `swapVolatilityToCollateral()`, a user would swap a certain amount of volatility token A to `Q` amount of the other volatility token B, and provide the same amount `Q` of A token, then redeem these pair of tokens to stablecoins. Users may not be able to know the right amount of in-token to swap before actually swapping, since he/she needs to have at least Q amount of in-token after swapping for redeeming.

The following code snippets show the token flows involved in this function.

```
        (swapAmounts[1], fees[0]) = _pool.swapExactAmountIn(
            address(_tokenIn),
            swapAmounts[0],
            isInverse ? _pool.tokens(0) : _pool.tokens(1),
            swapAmounts[1],
            msg.sender,
            true
        );
```

```
        _tokenIn.transferFrom(msg.sender, address(this), swapAmounts[1]);
        _protocol.redeem(swapAmounts[1]);
```

```
        _transferAsset(stableCoin, collateralAmount, msg.sender);
```

The users only provide `swapAmounts[0]` in-token to the pool, and the pool transfer `swapAmounts[1]` out-token to the controller. Then, `swapAmounts[1]` in-token was transferred to the controller, but the `VolmexProtocol.redeem()` function would burn `swapAmounts[1]` amount of both volatility tokens, where users may not be able to know that the protocol needs `swapAmounts[1]` tokens from them before the swapping. We would like to discuss whether this is the intended design and how to alleviate such situations.

Recommendation

N/A

Alleviation

[**Volmex Team**]: CertiK team's understanding is correct. It is impossible to determine the value Q before a swap because there are more than 2 unknown variables in the calculations. The user is expecting Collateral as a return for X Volatility token, so the swap getVolatilityToCollateral estimates the minimum amount of collateral returned for a given amount of Volatility token. During the swap, the complete amount of volatility is not swapped some immaterial amount remains, therefore the volatility token amount provided is the maximum token amount In. This will be notified to users on the UI.

VCP-01 | Unsafe Casting From `uint256` To `int256`

Category	Severity	Location	Status
Mathematical Operations	● Minor	projects/volmex/volmex-amm/contracts/VolmexPool.sol (2022/1/13) : 587~592, 675~680, 698~702	ⓘ Acknowledged

Description

The linked statements cast a `uint256` value to an `int256` without evaluating its bounds.

Recommendation

The audit team advise a safe casting operation to be performed by ensuring the result is still positive as big numbers will cause an underflow to occur here, thereby causing the system to misbehave.

Alleviation

[**Volmex Team**]: The values being converted to `int256` will be well within the bounds as a number greater than 2^255 would be required to break it.

VCP-02 | Users Unable To Join Pool When `poolTotal = 0`

Category	Severity	Location	Status
Logical Issue	● Minor	projects/volmex/volmex-amm/contracts/VolmexPool.sol (2022/1/13): 338 ~339	ⓘ Acknowledged

Description

When the share once reaches 0, i.e., `poolTotal = 0`, no one can ever join pool since the function reverts due to the division by zero in the function `_div()`.

Recommendation

The audit team recommend the Volmex team to revisit logic behind the `joinPool()` function to ensure that it is indeed reflecting the design.

Alleviation

[**Volmex Team**]: This case is handled in `finalize` method. Without finalizing no one can call the `joinPool` as well. If there is any other scenario, please provide a test case.

[**CertiK**]: `poolTotal` can be decreased to 0 or a really small number when the users exit the pool. Although in `finalize` method the initial shares are minted to the owner, it doesn't guarantee the `poolTotal` remains above 0. If the owner exit the pool, it is possible for the total share to go to 0.

In addition, when the total share of the pool is small, the chance of being manipulated by the flashloan attack increases. It is recommended that there should be a minimum balance of the pool to reduce the effect from flashloan attacks.

VOC-01 | Centralization Risk

Category	Severity	Location	Status
Logical Issue	● Major	projects/volmex/volmex-amm/contracts/oracles/VolmexOracle.sol (base): 70, 95, 110	ⓘ Acknowledged

Description

In the contract `VolmexOrale`, the role `owner` has the authority over the following functions:

- `updateVolatilityTokenPrice()` to update volatility token price by index;
- `updateVolatilityTokenPriceBySymbol()` to update volatility token price by symbol;
- `addVolatilityTokenPrice()` to update volatility token price.

[ox5b5961e2da9f83738de98c0716adde34fb641049 Update]: the role `owner` has the authority over the following functions:

- `updateBatchVolatilityTokenPrice()` to update a batch of the volatility tokens;
- `updateIndexBySymbol()` to update the index of a token symbol;
- `addVolatilityIndex()` to add a volatility index.

Any compromise to the `owner` account may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[**Volmex Team**]: We will be shifting privileged operations to Volmex core Multisig immediately and eventually moving to governance.

VOC-02 | Inaccurate Error Message

Category	Severity	Location	Status
Logical Issue	● Informational	projects/volmex/volmex-amm/contracts/oracles/VolmexOracle.sol (base): 37	✓ Resolved

Description

The error message in belowing `require` statement is inaccurate:

```
35          require(
36              _volatilityTokenPrice > 0 && _volatilityTokenPrice < 250,
37              'VolmexOracle: _volatilityTokenPrice should be greater than 0'
38          );
```

Recommendation

We recommend updating the error message to 'VolmexOracle: _volatilityTokenPrice should be greater than 0 and less than 250'.

Alleviation

The Volmex team heeded our advice and resolved this issue by providing a correct error message in the aforementioned `require` statement. The fixing is reflected in the commit `2d44b46e46a511d72ef09cfd0d990bc7ba5d525e`.

VPC-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	projects/volmex/volmex-amm/contracts/protocol/VolmexProtocol.sol (base): 147, 156, 174, 284, 302, 320, 339, 353	ⓘ Acknowledged

Description

In the contract `VolmexProtocol`, the role `owner` has the authority over the following functions:

- `toggleActive()` to toggle the active variable;
- `updateMinimumCollQty()` to update the `minimumCollateralQty`;
- `updateVolatilityToken()` to update the volatility token;
- `settle()` to settle the contract, preventing new minting and providing individual token redemption;
- `recoverTokens()` to recover tokens accidentally sent to this contract;
- `updateFees()` to update the percentage of `issuanceFees` and `redeemFees`;
- `claimAccumulatedFees()` to safely transfer the accumulated fees to owner;
- `togglePause()` to Pause/unpause volmex position token.

Any compromise to the `owner` account may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[**Volmex Team**]: We will be shifting privileged operations to Volmex core Multisig immediately and eventually moving to governance.

VPC-02 | Potential Reentrancy Attack

Category	Severity	Location	Status
Logical Issue	● Medium	projects/volmex/volmex-amm/contracts/protocol/VolmexProtocol.sol (base): 194, 370	ⓘ Acknowledged

Description

A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the external call resolved the effects.

The functions `VolmexProtocol.collateralize()` and `VolmexProtocol._redeem()` have external calls before state updates, so they are vulnerable to reentrancy attacks.

Recommendation

We recommend applying OpenZeppelin ReentrancyGuard library - nonReentrant modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation

[**Volmex Team**]: There is no re-entrancy here because the interaction happens only with the Volmex core contracts.

[**CertiK**]: Interactions with `collateral`, `volatilityToken` and `inverseVolatilityToken` are considered as external calls because their addresses are not determined until initialization. It would be recommended to exclude the possibilities of reentrancy attacks at the level of implementation.

VPC-03 | Lack Of Handling Return Value

Category	Severity	Location	Status
Logical Issue	● Minor	projects/volmex/volmex-amm/contracts/protocol/VolmexProtocol.sol (base) : 208, 311, 343, 389	ⓘ Acknowledged

Description

Functions `transfer()` and `transferFrom()` are not void-returning functions. Ignoring their return values, especially when their first return value represents the status if the transaction is executed successfully, might cause unexpected exceptions.

Recommendation

We recommend handling return values of the functions `transfer()` and `transferFrom()` at the aforementioned line before continuing processing.

Alleviation

[**Volmex Team**]: Since we use standard OpenZeppellin tokens, we don't need to implement this.

[**CertiK**]: For `transfer()` and `trasnferFrom()`, the return value handling would not be required if they are guaranteed to be reverted upon failure. However, considering possible contract updates, we would still recommend handling return values of the aforementioned functions.

VPP-01 | Potentially Incorrect Decimal Assumption

Category	Severity	Location	Status
Logical Issue	● Major	projects/volmex/volmex-amm/contracts/protocol/VolmexProtocol.sol (2022/1/13): 191~196	✓ Resolved

Description

The functions `VolmexProtocol.collateralize()` and `VolmexProtocol.redeem()` defined here assumes that the collateral quantity will hold `18` decimal places, which may not be the case i.e. for some stablecoins, wrapped Bitcoin implementations, and more.

In the controller, before passing the collateral quantity into `VolmexController.collateralize()` or `VolmexProtocol.redeem()`, the collateral quantity is checked and scaled by the decimal precision in `VolmexController._calculateAssetQuantity()`. However, the `VolmexProtocol.collateralize()` and `VolmexProtocol.redeem()` can be called directly without the decimal checks in `VolmexController`, which may cause erraneous decimals.

Recommendation

The audit team strongly recommend the decimals of the token to be assimilated in the codebase by querying the `decimals` member and storing it in an `immutable` contract level variable that is consequently used in the calculations.

Alleviation

[**Volmex Team**]: Check comments for VCK-01, it covers this as well.

VPP-02 | Lack Of Handling Return Value

Category	Severity	Location	Status
Logical Issue	● Minor	projects/volmex/volmex-amm/contracts/protocol/VolmexProtocol.sol (2022/1/13): 205, 308, 340, 386	ⓘ Acknowledged

Description

Functions `transferFrom()` and `transfer()` are not void-returning functions. Ignoring their return values, especially when their first return value represents the status if the transaction is executed successfully, might cause unexpected exceptions.

Recommendation

The audit team recommend handling return values of the functions `transferFrom()` and `transfer()` at the aforementioned lines before continuing processing.

VPP-03 | Incomplete Function

Category	Severity	Location	Status
Logical Issue	● Informational	projects/volmex/volmex-amm/contracts/protocol/VolmexProtocol.sol (2022/1/13): 362~365	✓ Resolved

Description

The function `VolmexProtocol.upgradeTo()` is incomplete, which is suggested in its comments.

```
362     function upgradeTo(address newImplementation) external virtual {
363         // _authorizeUpgrade(newImplementation);
364         // _upgradeToAndCallSecure(newImplementation, bytes(""), false);
365     }
```

Recommendation

The audit team recommend implementing the full logic of this function.

Alleviation

The Volmex team heeded our advice and resolved this issue by removing the aforementioned function. The fixing is reflected in the commit `16cd0c9d7c895d8ccedc32f92c2c1e1e3f5f2a2e`.

VPT-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	projects/volmex/volmex-amm/contracts/protocol/VolmexPositionToken.sol (base): 51, 64, 81, 98	ⓘ Acknowledged

Description

In the contract `VolmexPositionToken`, the role `VOLMEX_PROTOCOL_ROLE` has the authority over the following functions:

- `mint()` to mint new tokens to arbitrary accounts;
- `burn()` to burn new tokens from arbitrary accounts;
- `pause()` to pause the whole contract;
- `unpause()` to unpause the whole contract.

Any compromise to the `VOLMEX_PROTOCOL_ROLE` account may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `VOLMEX_PROTOCOL_ROLE` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[**Volmex Team**]: We will be shifting privileged operations to Volmex core Multisig immediately and eventually moving to governance.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

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

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK’s prior written consent in each instance.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is 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 assessment. This report does 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.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

CERTIK