Smart Contract Audit

volmex-core

Prepared for Volmex Finance • May 2021

v210514

1. Executive Summary

In April, Volmex Finance engaged Coinspect to perform a source code review of its new protocol for volatility indices and non-custodial trading platform built on Ethereum smart contracts. The objective of the audit was to evaluate the security of the smart contracts being developed.

The assessment was conducted on contracts from the Git repository at https://github.com/volmexfinance/volmex-core obtained on **April 26**.

The code reviewed is well documented, and the tests code coverage is good.

The smart contracts reviewed are upgradable and several important parameters can be reconfigured by the contract owner after they are created. Barring the administrative role actions that could hurt the protocol user, such as the emergency settlement mechanism and other actions available to the privileged role, the current implementation does guarantee that collateral deposits are kept in the protocol contract and are available to be redeemed by returning back exactly the same amount that was minted originally.

The flash loan based / market manipulation attacks protection mechanisms in place could be bypassed by using currently available MEV tools that allow bundling a sequence of transactions. It is recommended to further improve these mechanisms before launching the protocol. After the audit, the Volmex team removed these defensive measures as they consider them no longer necessary.

The off-chain components of the platform, including the index value calculation algorithm and external data sources from which options data is pulled are not relevant to the security of the smart contracts; they were not in scope for this engagement.

The following issues were identified during the assessment:

High Risk	Medium Risk	Low Risk
1	3	0

The high risk finding reported explains how the current flash loan based attacks protections are insufficient (See VOL-004). A medium risk finding reported details an unlikely scenario where the `VolmexProtocol` contract could be abused to mint more tokens than the corresponding to the collateral actually deposited (See VOL-001). Another medium risk finding details how the protocol owner could modify the settlement price after a contract has been settled (See VOL-002). The last medium risk finding is related to flash loan based attack protections missing in a public function (See VOL-003).

In May 2021, Coinspect updated the fix status for each finding with Volmex's code updates and feedback.

The following report details the tasks performed and the vulnerabilities found during this audit as well as several suggestions aimed at improving the overall code quality and warnings regarding potential issues.

2. Introduction

The audit started on April 26 and was conducted on the Git repository at https://github.com/volmexfinance/volmex-core. The last commit reviewed during this engagement was `bf1ab0f21c9d77147ad34ca862ebdeadcb5c584b` from **April 26**:

```
commit bf1ab0f21c9d77147ad34ca862ebdeadcb5c584b
Author: Derek Barrera <derekbarrera@gmail.com>
Date:   Mon Apr 26 11:23:21 2021 -0600

   add .env
   add tests

commit a1d17b695181e1bee95d2071021353c9bac5e5ff
Author: Ayush Tiwari <ayush@volmexlabs.com>
Date:   Thu Apr 22 19:05:00 2021 +0530

   Volmex Protocol V1.
```

The scope of the audit was limited to the latest version of the following Solidity source files, shown here with their sha256sum hash:

```
39948d3e8f84a82c1d0443187eae90ed1af533ef94740cf2aa64e06834e03964  ./VolmexProtocol.sol
f00911bbf8b361a4025b467cc63bf7de0b34a9947f8832353914de07bf0ceb36  ./interfaces/IERC20Modified.sol
e3f2efcafa159a1c46f4b6b1607b4361b6c37e73452a0a6755c2ab50119b91ac  ./interfaces/IVolmexProtocol.sol
cddc6c15a8d343c2e2787783355a7722aaf05dbbcb3c3962cb354d228ff719cd  ./helper/TestCollateralToken.sol
465b6a098dac6e71c071e8b5162f025309e795e6b0fc0962dc1e0db1fdbfebc6  ./library/VolmexSafeERC20.sol
d61786bb818fc3071e25aeeceea0e2c2b188296b7743889a8037f36d50ae68bc  ./tokens/VolmexPositionToken.sol
1144248fd81ac9096a3376c9f4973cd135b3f90d22ae5efc53183083f7e229d6  ./VolmexIndexFactory.sol
```

Volmex is a protocol for volatility indices and a non-custodial trading platform built on Ethereum. The focus of this audit is their new core protocol for volatility indices which is currently being developed.

The Volmex protocol's main goal is to enable the creation of volatility indexes and tradable tokens which track volatility for assets.

Users can deposit stablecoins as collateral and receive in exchange an equal amount of volatility and inverse volatility tokens proportional to the ratio set in the corresponding `VolmexProtocol` contract. Position token holders can then profit from trading and/or providing liquidity in AMMs. The holders can redeem collateral anytime by surrendering position tokens: the same amount of volatility and inverse volatility tokens must be provided. There is an emergency mechanism, called protocol settlement, that shuts down the protocol, stops minting and allows redeems using a settlement price established by the protocol owner.

3. Assessment

All contracts are specified to be compiled with an up-to-date Solidity compiler version `0.8.2`. There are no security relevant warnings emitted by the compiler.

All the 146 existing tests passed. The **test coverage at the moment is close to 100%** for all the contracts that are deployed.

The source code is clearly documented with NatSpec format annotations with both developer and end-user oriented messages.

The audit is focused on specific concerns detailed by the client which are described below:

- Flash loans based exploits.
- Volmex protocol position tokens particular behaviors that could affect the interaction with Uniswap pools.
- Re-entrancy related attacks.

The `VolmexIndexFactory.sol` contract is responsible for instantiating new volatility indexes through the `createIndex` function, which is only callable by the contract owner. This factory constructor creates new instances of the `VolmexProtocol` and the `VolmexPositionToken` contracts. The factory itself is not upgradable.
When `createIndex` is called, new volatility and inverse volatility position tokens are “cloned” using a salt based on the token name, symbol and the corresponding Volmex protocol index in order to obtain deterministic addresses. The cloning is done by the imported OpenZeppelin project `Clones` library, which implements the EIP-1167 and relies on the `CREATE2` EVM opcode. The same procedure is used for the new cloned protocol implementation, the salt used in this case is the new `VolmexProtocol` index in the `getIndex` mapping in the factory contract. This

mechanism allows querying the `determineIndexAddress(uint256 _indexCount)` and the `determinePositionTokenAddress(uint256 _indexCount, string memory _name, string memory _symbol)` functions to obtain the protocol and position tokens respectively. The new `VolmexProtocol` contract is granted the `VOLMEX_PROTOCOL_ROLE` role in the new position tokens. Also, the transaction sender (which must be the factory contract owner) is granted the `VOLMEX_ADMIN_ROLE` role in the new position tokens. These roles are then renounced by the factory.

The `VolmexPositionTokens.sol` contract implements the volatility and inverse volatility tokens. The contracts implement the ERC20 interface, are pausable and upgradable. The `VOLMEX_PROTOCOL_ROLE` is allowed to mint, burn, pause and unpause the token.

The Volmex core protocol is implemented in the `VolmexProtocol` smart contract, which is also upgradable.

All ERC20 transfers are performed through the `VolmexSafeERC20` library wrappers `safeTransfer` and `safeTransferFrom`, which correctly handle both scenarios of compliant and non-compliant ERC20 tokens and do not return any value from transfer calls. In case of a non-reverting transfer with no return value, transfer success is assumed.

A `VolmexProtocol` contract is set to active in its constructor and the contract owner can deactivate it and reactivate it. When not active, all user operations are suspended. Additionally, the contract owner can pause and unpause the protocol's underlying position and inverse position tokens, which as explained above, pauses the transfer operations of those tokens.

Also, the protocol owner has the ability to switch those underlying position and inverse position tokens anytime, **which could prevent position token holders from being able to redeem their tokens to get back their collateral**.

When a new `VolmexProtocol` is created, it is initialized with a volatility cap ratio that is used to determine how many position tokens (and inverse position tokens) are minted for each collateral token deposited in the contract. This value can not be modified after the contract is initialized. During initialization the collateral token to be used is set as well, and can not be modified afterwards. There is also a minimum quantity of collateral tokens that are needed when calling the `collateralize` function. In this case, this value can be modified by the contract owner after initialization.

The main user accessible functions are: `collateralize` and `redeem`.

The `collateralize` function results in collateral tokens being transferred to the protocol contract and the proportional amount of positions tokens being minted to the transaction sender. The same amount of both volatility and inverse volatility tokens are minted. Concretely, the quantity of tokens minted is equal to the quantity of collateral tokens received (minus fees if any) divided by the `volatilityCapRatio` value set during protocol initialization. The accepted collateral token being deposited was set during the protocol contract initialization and is intended to be set to a stablecoin such as DAI.

The `redeem` function allows position token holders to obtain back collateral tokens by surrendering an amount of volatility tokens passed as a parameter. The same amount of volatility and inverse volatility tokens are burned, only one quantity parameter is provided. The amount of collateral being redeemed is calculated as the quantity of position tokens being surrendered multiplied by the `volatilityCapRatio` value set during protocol initialization, minus fees if any.

The contract charges issuance and redeem fees, which are initially set to zero, and can be updated anytime by the contract owner, up to a hardcoded maximum of 5% of the amount collateralized. The contract owner can claim the accumulated fees anytime, those fees are transferred to his account.

The Volmex implementation relies on a combination of two defensive rules that are enforced in order to protect the token holders from market manipulation attacks.

1. By default, the `collateralize` and `redeem` operations can only be called from EOAs (externally owned accounts) in order to prevent flash loan based attacks. **This limitation could hinder the protocol's adoption, for example, because multi-signature wallets will not be able to operate**. It was found that this protection mechanism is not enforced for redeems when the protocol is a settled state, this scenario is described in VOL-003 redeemSettled missing flash loan attacks protections.
2. Additionally, a second protection was added to prevent atomic market manipulation attacks: only one operation per address per block is allowed. This only applies to the `collateralize` and `redeem` functions, the `redeemSettled` calls are not limited. This is further detailed in VOL-003 redeemSettled missing flash loan attacks protections.

Both protection mechanisms can be bypassed by employing the flashbots framework or any similar alternative as detailed in VOL-004 Flash loan based attack protections can be bypassed.

Regarding the first rule, the contract owner can approve and revoke approval for certain contracts addresses to bypass this limitation. This is intended to whitelist known trusted smart contracts. **Special care should be taken when deciding to add a contract to the whitelist: the contract should be audited, taking into account any external calls it performs, and should also be not upgradable.**

The external owner can trigger the protocol settlement mode by calling the `settle` function. This action is irreversible. Further minting via `collateralize` calls is disabled, and redeems can only be performed through the `redeemSettled` function. The call to settle establishes a `settlementPrice` passed as a parameter to the function. **Even though the protocol can not be unsettled, it is still possible to call the `settle` function again and change the `settlementPrice` as many times as desired.** This issue is detailed in VOL-002 Settlement price can be updated after

settlement. The `settlementPrice` must be below or equal to the protocol's volatility cap ratio.

When the protocol is settled, token holders must use the `redeemSettled` function. This function differs from the regular `redeem` function in how the quantity of collateral being sent back to the user is calculated. In this scenario, the quantity of volatility tokens can differ from the quantity of inverse volatility tokens being surrendered. The `settlementPrice` set by the protocol owner is used to calculate how much collateral is obtained by the user as shown below:

```
uit256 collQtyToBeRedeemed =
    (_volatilityIndexTokenQty * settlementPrice) +
        (_inverseVolatilityIndexTokenQty * (volatilityCapRatio - settlementPrice))
```

The settlement feature is intended to allow shutting down the system and reimbursing the volatility holders, and the public documentation states it is a last resort although it is unclear in which scenarios it would be used.

The source code comment in the `onlySettled` and `onlyNotSettled` function modifiers is misleading, as the settle flag is global and not address specific as the comment states: `Used to check calling address is not settled.`

There is a function available for the contract owner that permits recovering tokens incorrectly sent to the protocol contract. Any token but the collateral token of the protocol itself can be recovered in this way.

Because of the protocol's centralized administration role and the actions allowed to that role, there are many ways the protocol owner's actions could harm the token holders. For example: changing the underlying position tokens for new ones (thus preventing redeems of older tokens), pausing position tokens transfers, triggering settlement mode with arbitrary settlement price, etc.

It is important to keep in mind that the contracts are upgradeable; this means that they are not truly autonomous and the Volmex Finance organization has full power

over the contracts and the deposited funds. Upgradability allows the organization to fix bugs and potentially mitigate ongoing attacks by pausing a contract, but at the same time it also poses a risk. Special care should be taken with the admin keys that allow access to this functionality; it is also recommended to consider resigning admin access in the future to make the contracts fully autonomous, or consider options for decentralized governance. The Volmex team has stated that administrative functions will be handled by a multi-signature contract.

The off-chain code responsible for calculating the index values representing the implied volatility of the underlying assets (e.g., ETH ,BTC) was not reviewed. Volmex Finance documentation states options data is pulled from centralized options exchanges in order to calculate the index values. These indexes will be published in the platform's website, but currently there is no direct interaction between the index values and the contracts reviewed.

4. Recommendations

The following list sums up the most important suggestions provided in this report:

1. Redesign the market manipulation attacks defense mechanisms.
2. Enforce an immutable settlement price for settled contracts.
3. Verify balances before and after `transferFrom` calls to guarantee the expected amount was transferred.
4. Implement throughout review procedures for smart contracts being whitelisted to operate with the protocol: they should be audited, taking into account any external calls they perform, and should also be not upgradable.
5. Clearly document administrative roles and their capabilities and how these actions could impact the protocol users.
6. Amend the source code comment in the `onlySettle` and `onlyNotSettled` function modifiers.

5. Summary of Findings

ID	Description	Risk	Fixed
VOL-001	Collateral token balance not verified after transferFrom	Medium	✔
VOL-002	Settlement price can be updated after settlement	Medium	✔
VOL-003	redeemSettled missing flash loan attacks protections	Medium	✔
VOL-004	Flash loan based attack protections can be bypassed	High	✔

6. Remediations

During May 2021 Coinspect verified the findings that Volmex addressed had been correctly fixed.

The following table lists the findings that were fixed with their corresponding fix status and commit:

ID	Description	Status / Commit
VOL-001	Collateral token balance not verified after transferFrom	Fixed `85daf69521fc59b592d3b6427880ad19ab5aa33d`
VOL-002	Settlement price can be updated after settlement	Fixed `85daf69521fc59b592d3b6427880ad19ab5aa33d`
VOL-003	redeemSettled missing flash loan attacks protections	Fixed `85daf69521fc59b592d3b6427880ad19ab5aa33d`
VOL-004	Flash loan based attack protections can be bypassed	Fixed `efa11242ae872b4355a7ce3d20aea0ef6d955eab`

Regarding VOL-004, the issue is considered fixed as Volmex decided to remove the protections involved.

More detailed status information can be found in each finding.

7. Findings

VOL-001	Collateral token balance not verified after transferFrom

Total Risk
Medium

Fixed
✔

Impact
High

Likelihood
Low

Location
VolmexProtocol.sol

Description

The `VolmexProtocol` contract could be abused to mint more tokens than the amount corresponding to the collateral actually deposited.

The collateralize function does not verify if the collateral token `safeTransferFrom` resulted in the expected amount being transferred, instead if the call does not revert, it trusts the total amount was transferred.

There are tokens that transfer less than the specified amount. For example, the USDT token's `transfer` and `transferFrom` functions (Tether: USDT Stablecoin | 0xdac17f958d2ee523a2206206994597c13d831ec7) deduct a fee for each transfer if a fee percentage is configured. While it is not configured to take fees right now, this could change in the future.

Similar scenarios are possible in the future depending on the stablecoins that are added as collateral in the Volmex protocol.

This is the `collateralize` function:

```
function collateralize(uint256 _collateralQty)
    external
    onlyActive
    defend
    blockLocked
    onlyNotSettled
```

```
{
    require(
        _collateralQty >= minimumCollateralQty,
        "Volmex: CollateralQty < minimum qty required"
    );

    _lockForBlock();

    collateral.safeTransferFrom(msg.sender, address(this), _collateralQty);

    ...

    uint256 qtyToBeMinted = _collateralQty / volatilityCapRatio;

    volatilityToken.mint(msg.sender, qtyToBeMinted);
    inverseVolatilityToken.mint(msg.sender, qtyToBeMinted);

    emit Collateralized(msg.sender, _collateralQty, qtyToBeMinted, fee);
}
```

Recommendation

It is advised to check the balance of the contract before and after the `transferFrom` call is performed in order to determine the exact amount that was received.

Status

This issue was fixed in commit 85daf69521fc59b592d3b6427880ad19ab5aa33d.

VOL-002	Settlement price can be updated after settlement

Total Risk	Impact	Location
Medium	High	VolmexProtocol.sol
Fixed ✔	Likelihood Low	

Description

The `VolmexProtocol` contract settlement price can be modified after the contract has been settled, which could lead to fairness issues as token holders could end up redeeming their tokens at different exchange rates.

The call to the `settle` function establishes a `settlementPrice` passed as a parameter to the function. **Even though the protocol can not be unsettled, it is still possible to call the `settle` function again and change the `settlementPrice` as many times as desired.**

Coinspect confirmed with the Volmex team this is not the intended behaviour, and that a check should be made to prevent the contract owner to resettle a contract that was already settled.

Recommendation

Only allow settling contracts that are unsettled.

Status

This issue was fixed in commit 85daf69521fc59b592d3b6427880ad19ab5aa33d.

VOL-003	redeemSettled missing flash loan attacks protections

Total Risk
Medium

Fixed
✔

Impact
High

Likelihood
Low

Location
VolmexProtocol.sol

Description

The protection mechanisms used for the functions `redeem` and `collateralize` are missing or incomplete in `redeemSettled`. As a consequence, the rules intended to protect the protocol from market manipulation attacks are not enforced in this scenario.

The function `redeemSettled` is used by token holders to redeem collateral back by surrendering position tokens at the settlement price when the contract has been settled. Because of the missing protection mechanisms this function could be invoked from smart contracts (instead of only from EOAs) and could be invoked more than one time per address per block.

Regarding the `blockLocked` modifier, there is a call to the `lockForBlock` function (responsible for storing the last block the `msg.sender` address performed a transaction) but the `blockLocked` modifier is missing in that function, rendering the `lockForBlock` call useless. The `defend` modifier was completely omitted.

Recommendation

Add the `defend` and `blockLocked` modifiers to the `redeemSettled` function.

Status

This issue was fixed in commit 85daf69521fc59b592d3b6427880ad19ab5aa33d.

VOL-004	Flash loan based attack protections can be bypassed

Total Risk	Impact	Location
High	High	VolmexProtocol.sol
Fixed ✔	Likelihood: High	

Description

Attackers can exploit the protocol, bypassing the protection mechanisms in place, by using currently available MEV tools (e.g., flashbots).

In order to prevent flash loan-based attacks and manipulation of the position tokens value in trading platforms by minting large amounts of tokens and redeeming them in the same transaction without risk, the Volmex protocol enforces two rules:

1. The public functions are only callable from EOAs. Smart contracts are banned, unless explicitly whitelisted by the protocol owner. This is enforced by the `defend` function modifier.
2. Only one transaction, per address, per block is allowed. This is enforced by the `blockLocked` function modifier:

```
/**
 * @notice Used to secure our functions from flash loans attack.
 */
modifier blockLocked() {
    require(
        blockLock[msg.sender] < block.number,
        "Volmex: Operations are locked for current block"
    );
    _;
}
```

By bundling a set of transactions from different EOAs, a miner can include these transactions in the specified order and check if the result of their executions is

profitable, avoiding the need to use a smart contract to chain a set of transactions, bypassing the first defensive mechanism.
The second mechanism by itself is not enough because the accounts are still able to transfer the position tokens to other accounts to perform the necessary transactions bypassing this limitation.

Also, these rules, besides being insufficient, can be counterproductive and could hinder user adoption, because for example they limit the ability to use multi-signature wallets and trading bots.

Recommendation

Coinspect recommends redesigning the flash loan based attacks protection mechanisms. One possible solution could be splitting the minting process, so the new tokens are only claimable after a certain period of time has passed from the collateral being deposited. The same process can be applied to the redeem operation.

Status

This issue was addressed by commit efa11242ae872b4355a7ce3d20aea0ef6d955eab.

Note this commit completely removes both protection mechanisms that were in place. After evaluating this finding and potential solutions, the Volmex team decided the defensive measures they had implemented were never really necessary.

8. Disclaimer

The information presented in this document is provided "as is" and without warranty. Source code reviews are a “point in time” analysis and as such it is possible that something in the code could have changed since the tasks reflected in this report were executed. This report should not be considered a perfect representation of the risks threatening the analyzed system.