



Smart Contract Security Assessment

09-3-2022

Prepared for
Ovix

Online Report
[Ovix-lending-protocol](#)

Lending Protocol Security Audit

Audit Overview

We were tasked with performing an audit on the Ovix decentralized lending and borrowing protocol based on the Compound implementation.

The codebase has undergone several adjustments from the original Compound implementation to accommodate for the additional features the Ovix team desired. Namely, the `pragma` versions have been upgraded no longer necessitating the use of `SafeMath` with several statements have been wrapped in `unchecked` code blocks and the borrow and supply tracking mechanisms of the `OToken` (ex-`CToken`) implementations have been upgraded to support second-level tracking via interactions with the `BoostManager` contract.

Over the course of the audit, we identified a significant flaw in the `BoostManager` contract that causes boosted balances to be miscalculated.

VoteController.sol Scope

Furthermore, we pinpointed several discrepancies in the `VoteController` contract in comparison to the Vyper implementation by Curve Finance. We requested supplemental information from the Ovix team to identify what the delta is of the `VoteController` in relation to the original Curve.fi implementation.

In conjunction with the material provided to us by Ovix and our own analysis of the codebase, we deduced that the contract cannot be audited as or considered a "fork" as it has been re-written to adapt to a completely different system (Compound vs Curve.fi) with multiple original notions being removed and replaced by new ones that are Ovix or Compound related.












To this end, the `VoteController` contract cannot be considered as audited in full as we only evaluated the difference between the original implementation and the new one and we strongly advise the Ovix team to procure a dedicated audit for the `VoteController` as we cannot vouch for its safety.

Post-Audit Conclusion

The Ovix team provided a PR (Pull Request) in the original GitHub repository in scope of the audit that contained commits meant to address the exhibits outlined in the report.



We advise them to re-visit certain exhibits that have been marked as not-addressed or partially addressed to ensure the outputs of the audit have been correctly assimilated in the codebase as desired.




Contracts Assessed

| Files in Scope | Repository | Commit(s) |
|------------------------------|-----------------------------------------------------------------------------------------------------|------------------------|
| BoostManager.sol (BMR) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| CarefulMath.sol (CMH) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| Comptroller.sol (COM) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| ComptrollerStorage.sol (CSE) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| Exponential.sol (EXP) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| ErrorReporter.sol (ERR) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| ExponentialNoError.sol (ENE) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| JumpRateModel.sol (JRM) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| Maximillion.sol (MAX) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| Ovix.sol (OVI) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| OErc20.sol (OE0) | Ovix-protocol  | 016c904860, fd6151c0d2 |

| Files in Scope | Repository | Commit(s) |
|----------------------------------|-----------------------------------------------------------------------------------------------------|------------------------|
| OMatic.sol (OMC) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| OToken.sol (OTN) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| OErc20Storage.sol (OES) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| OTokenStorage.sol (OTS) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| OvixChainlinkOracle.sol (OCO) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| OvixChainlinkOracleV2.sol (OCV) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| PriceOracle.sol (POE) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| SafeMath.sol (SMH) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| TransparentProxy.sol (TPY) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| Unitroller.sol (UNI) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| UnitrollerAdminStorage.sol (UAS) | Ovix-protocol  | 016c904860, fd6151c0d2 |
| VoteController.sol (VCR) | Ovix-protocol  | 016c904860, fd6151c0d2 |

Audit Synopsis

| Severity | Identified | Alleviated | Partially Alleviated | Acknowledged |
|---------------------------------------------------------------------------------------------------|------------|------------|----------------------|--------------|
|  Unknown | 1 | 0 | 0 | 1 |
|  Informational | 36 | 29 | 2 | 5 |

| Severity | Identified | Alleviated | Partially Alleviated | Acknowledged |
|------------------------------------------------------------------------------------------|------------|------------|----------------------|--------------|
|  Minor | 12 | 7 | 0 | 5 |
|  Medium | 12 | 3 | 2 | 7 |
|  Major | 0 | 0 | 0 | 0 |

During the audit, we filtered and validated a total of **8 findings utilizing static analysis** tools as well as identified a total of **53 findings during the manual review** of the codebase. We strongly recommend that any minor severity or higher findings are dealt with promptly prior to the project's launch as they introduce potential misbehaviours of the system as well as exploits.

Compilation

The project utilizes `hardhat` as its development pipeline tool, containing an array of tests and scripts coded in TypeScript.

To compile the project, the `compile` command needs to be issued via the `npx` CLI tool to `hardhat`:

```
BASH
```

```
npx hardhat compile
```

The `hardhat` tool automatically selects Solidity version `0.8.4` based on the version specified within the `hardhat.config.ts` file.

The project contains discrepancies with regards to the Solidity version used, however, they are solely contained in dependencies and can thus be ignored.

The Ovox team has locked the `pragma` statements to `0.8.4` (`=0.8.4`), the same version utilized for our static analysis as well as optimizational review of the codebase.

During compilation with the `hardhat` pipeline, no errors were identified that relate to the syntax or bytecode size of the contracts.

Static Analysis

The execution of our static analysis toolkit identified **648 potential issues** within the codebase of which **639 were ruled out to be false positives** or negligible findings.

The remaining **9 issues** were validated and grouped and formalized into the **8 exhibits** that follow:

| ID | Severity | Addressed | Title |
|---------|---------------|-----------|------------------------------------------------|
| BMR-01S | Minor | Yes | Inexistent Sanitization of Input Addresses |
| COM-01S | Informational | Yes | Inexistent Event Emissions |
| COM-02S | Informational | Yes | Literal Equality of <code>bool</code> Variable |
| COM-03S | Minor | Yes | Inexistent Sanitization of Input Addresses |
| MAX-01S | Minor | Yes | Deprecated Native Asset Transfer |
| OCV-01S | Informational | Yes | Inexistent Event Emissions |
| VCR-01S | Informational | Yes | Redundant Constructor Implementation |
| VCR-02S | Informational | Yes | Redundant Variable Assignments |

Manual Review

A **thorough line-by-line review** was conducted on the codebase to identify potential malfunctions and vulnerabilities in the lending and borrowing protocol of OviX.











As the project at hand implements a decentralized lending and borrowing protocol, intricate care was put into ensuring that the **flow of funds within the system conforms to the specifications and restrictions** laid forth within the protocol's specification.

We validated that **all state transitions of the system occur within sane criteria** and that all rudimentary formulas within the system execute as expected. We **pinpointed a significant flaw** within the system's balance boosting mechanism which could have had **severe ramifications** to its overall operation, however, it was conveyed ahead of time to the OviX team to be **promptly remediated**.

Additionally, the system was investigated for any other commonly present attack vectors such as re-entrancy attacks, mathematical truncations, logical flaws and **ERC / EIP** standard inconsistencies. The documentation of the project was satisfactory to a certain extent, however, we strongly recommend the documentation of the project to be expanded at certain complex points such as the exact way the `VotingController` contract is meant to implement its slope functions akin to Curve's implementation.

A total of **53 findings** were identified over the course of the manual review of which **22 findings** concerned the behaviour and security of the system. The non-security related findings, such as optimizations, are included in the separate **Code Style** chapter.

The finding table below enumerates all these security / behavioural findings:

| ID | Severity | Addressed | Title |
|---------|---------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|---------------------------------------------|
| BMR-01M |  Unknown |  Acknowledged | Incorrect Multiplier Utilization |
| BMR-02M |  Minor |  Acknowledged | Improper Prohibition of Base Initialization |
| BMR-03M |  Minor |  Yes | Inexplicable Capability of Re-Invocation |
| BMR-04M |  Medium |  Acknowledged | Flash-Loan Prone Balance Measurements |
| COM-01M |  Minor |  No | Inexistent Retroactive Reward System |

| ID | Severity | Addressed | Title |
|---------|--------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|-----------------------------------------------------------|
| COM-02M |  Minor |  Yes | Inexistent Sanitization of Market Addition |
| COM-03M |  Medium |  Acknowledged | Overly Centralized Reward Control |
| OE0-01M |  Medium |  Acknowledged | Improper Invocation of EIP-20 <code>transfer</code> |
| OVI-01M |  Medium |  Yes | Insecure Elliptic Curve Recovery Mechanism |
| OVI-02M |  Medium |  Acknowledged | Race-Prone Nonce System |
| OCO-01M |  Minor |  Nullified | Potentially Restrictive Token Support |
| OCO-02M |  Medium |  Nullified | Authorative Control of Asset Prices |
| OCO-03M |  Medium |  Nullified | Permittance of Feed Overriding |
| OCV-01M |  Minor |  Acknowledged | Improper Staleness Limit |
| OCV-02M |  Minor |  Acknowledged | Potentially Restrictive Token Support |
| OCV-03M |  Medium |  Acknowledged | Authorative Control of Asset Prices |
| OCV-04M |  Medium |  Partial | Misconstrued Data Staleness System |
| OCV-05M |  Medium |  Acknowledged | Permittance of Feed Overriding |
| UNI-01M |  Medium |  No | Inexplicable Introduction of Pending Administrator Bypass |
| VCR-01M |  Minor |  Acknowledged | Potentially Overly Centralized Protocol Functionality |
| VCR-02M |  Minor |  Yes | Truncation of Reward Speed Achieved |
| VCR-03M |  Medium |  Partial | Improper Market Removal Methodology |

BoostManager Static Analysis Findings

BMR-01S: Inexistent Sanitization of Input Addresses

| Type | Severity | Location |
|--------------------|-----------------------------|--------------------------|
| Input Sanitization | <div><div></div>Minor</div> | BoostManager.sol:L38-L40 |

Description:

The linked function(s) accept `address` arguments yet do not properly sanitize them.

Example:

contracts/vote-escrow/BoostManager.sol

SOL

```
37 function initialize(  
38     IERC20 ve,  
39     IComptroller _comptroller,  
40     address _owner  
41 ) external {
```

Recommendation:

We advise some basic sanitization to be put in place by ensuring that each `address` specified is non-zero.

Alleviation:

The 0vix team added the recommended `address` sanitizations.

Comptroller Static Analysis Findings

COM-01S: Inexistent Event Emissions

| Type | Severity | Location |
|-------------------|-------------------------------------|------------------------------------------|
| Language Specific | <div><div></div>Informational</div> | Comptroller.sol:L1891-L1893, L1898-L1900 |

Description:

The linked functions adjust sensitive contract variables yet do not emit an event for it.

Example:

contracts/Comptroller.sol

SOL

```
1891function setVixAddress(address newVixAddress) public onlyAdmin {
1892    vixAddress = newVixAddress;
1893}
```

Recommendation:

We advise an `event` to be declared and correspondingly emitted for each function to ensure off-chain processes can properly react to this system adjustment.

Alleviation:

The Ovix team implemented the recommended `event`s.

COM-02S: Literal Equality of `bool` Variable

| Type | Severity | Location |
|------------------|-------------------------------------|-----------------------|
| Gas Optimization | <div><div></div>Informational</div> | Comptroller.sol:L1872 |

Description:

The linked `bool` comparison is performed between a variable and a `bool` literal.

Example:

contracts/Comptroller.sol

SOL

1872guardianPaused[address(oToken)].borrow == true &&


Recommendation:

We advise the `bool` variable to be utilized directly either in its negated (`!`) or original form.

Alleviation:

The Ovox team changed the linked statement to directly utilize the `bool` variable.

COM-03S: Inexistent Sanitization of Input Addresses

| Type | Severity | Location |
|--------------------|-----------------------------------------------------------------------------------------|------------------------------|
| Input Sanitization |  Minor | Comptroller.sol:L1891, L1898 |

Description:

The linked function(s) accept `address` arguments yet do not properly sanitize them.

Example:

```
contracts/Comptroller.sol

SOL

1888/**
1889 * @notice Set the OVIX token address
1890 */
1891function setVixAddress(address newVixAddress) public onlyAdmin {
1892    vixAddress = newVixAddress;
1893}
```

Recommendation:

We advise some basic sanitization to be put in place by ensuring that each `address` specified is non-zero.

Alleviation:

The Ovox team added the recommended `address` sanitizations.

Maximillion Static Analysis Findings

MAX-01S: Deprecated Native Asset Transfer

| Type | Severity | Location |
|-------------------|-----------------------------|---------------------|
| Language Specific | <div><div></div>Minor</div> | Maximillion.sol:L43 |

Description:

The linked statement performs a low-level native asset transfer via the `transfer` function exposed by the `address payable` data type.

Example:

contracts/Maximillion.sol

SOL

43 payable(msg.sender).transfer(received - borrows);

Recommendation:

As new EIPs such as **EIP-2930** are introduced to the blockchain, gas costs can change and the `transfer` instruction of Solidity specifies a fixed gas stipend that is prone to failure should such changes be integrated to the blockchain the contract is deployed in. We advise alternative ways of transferring assets to be utilized instead, such as OpenZeppelin's Address.sol library and in particular the `sendValue` method exposed by it.

Alleviation:

The Ovix team applied the recommended fix.

OvixChainlinkOracleV2 Static Analysis Findings

OCV-01S: Inexistent Event Emissions

| Type | Severity | Location |
|-------------------|-------------------------------------|------------------------------------------------|
| Language Specific | <div><div></div>Informational</div> | OvixChainlinkOracleV2.sol:L143-L145, L154-L156 |

Description:

The linked functions adjust sensitive contract variables yet do not emit an event for it.

Example:

```
contracts/chainlink/OvixChainlinkOracleV2.sol
SOL
143 function setValidPeriod(uint256 period) external onlyAdmin {
144     validPeriod = period;
145 }
```

Recommendation:

We advise an `event` to be declared and correspondingly emitted for each function to ensure off-chain processes can properly react to this system adjustment.

Alleviation:

The Ovix team implemented the recommended `events`.

VoteController Static Analysis Findings

VCR-01S: Redundant Constructor Implementation

| Type | Severity | Location |
|-------------------|-------------------------------------|-------------------------|
| Language Specific | <div><div></div>Informational</div> | VoteController.sol:L174 |

Description:

The linked `constructor` definition is entirely redundant as it executes no statements.

Example:

```
contracts/vote-escrow/VoteController.sol
SOL
174 constructor() {}
```

Recommendation:

We advise the implementation to be omitted from the codebase optimizing its deployment cost.

Alleviation:

The 0vix team applied the recommended fix.

VCR-02S: Redundant Variable Assignments

| Type | Severity | Location |
|------------------|-------------------------------------|-----------------------------|
| Gas Optimization | <div><div></div>Informational</div> | VoteController.sol:L48, L80 |

Description:

The linked variables are assigned to redundantly to the default value of each relevant data type (i.e. `uint256` assigned to `0`, `address` assigned to `address(0)` etc.).

Example:

contracts/vote-escrow/VoteController.sol

SOL

48 uint256 public totalEmissions = 0; // in wei

Recommendation:

We advise the assignments to be safely omitted optimizing the codebase.

Alleviation:

The 0vix team applied the recommended fix.

BoostManager Manual Review Findings

BMR-01M: Incorrect Multiplier Utilization

| Type | Severity | Location |
|-------------------------|-------------------------------|----------------------------------------|
| Mathematical Operations | <div><div></div>Unknown</div> | BoostManager.sol:L240, L243-L245, L247 |

Description:

The `MULTIPLIER` of the protocol is defined as `10**18` (`1e18`) yet it is utilized as if it represented `0.1` (`0.1e18`) as a multiplier of "2.5" is represented as `25 * MULTIPLIER` instead of `25 * MULTIPLIER / 10`, thereby causing all multiplier boosts to be incorrect.

Example:

contracts/vote-escrow/BoostManager.sol

SOL

```
228 function calcBoostedBalance(  
229     address user,  
230     uint256 boosterBasis,  
231     uint256 balance  
232 ) internal view returns (uint256) {  
233     if (veBalances[user] == 0 || boosterBasis == 0) return balance;  
234  
235     uint256 minVe = (boosterBasis * balance) / MULTIPLIER;  
236  
237     uint256 booster;  
238  
239     if (veBalances[user] >= minVe) {  
240         booster = 25 * MULTIPLIER; // = 2,5  
241     } else {  
242         booster =  
243             ((15 * MULTIPLIER * veBalances[user]) / minVe) +  
244             10 *  
245             MULTIPLIER; // 1.5 * veBalance / minVe + 1;  
246     }  
247     return ((balance * booster) / (10 * MULTIPLIER));  
248 }
```

Recommendation:

We advise this trait of the system to be corrected to ensure proper boost balances are tracked by the contract.

Alleviation:

The Ovix team has considered this exhibit and assessed that the boost balances are correctly calculated in the system. We advised the Ovix team to instead use different multiplicants for the `minVe` and `booster` calculations as they are currently confusing (one using `MULTIPLIER` and the other using `10 * MULTIPLIER`), however, the Ovix team opted to retain the current code in place.

BMR-02M: Improper Prohibition of Base Initialization

| Type | Severity | Location |
|---------------|-----------------------------|--------------------------|
| Logical Fault | <div><div></div>Minor</div> | BoostManager.sol:L33-L35 |

Description:

The `constructor` of the contract accepts an input argument instead of setting the value of `init` directly to `true`.

Example:

```
contracts/vote-escrow/BoostManager.sol
SOL
33 constructor(bool _init) {
34     init = _init;
35 }
```


Recommendation:

We advise the `constructor` to remove the input argument and set the value directly as advised to prevent initializations of the logic implementation.

Alleviation:

The Ovox team has considered this exhibit but opted not to apply a remediation for it in the current iteration of the codebase.

BMR-03M: Inexplicable Capability of Re-Invocation

| Type | Severity | Location |
|------------------------|-----------------------------------------------------------------------------------------|----------------------------|
| Centralization Concern |  Minor | BoostManager.sol:L303-L306 |

Description:

The `veOVIX` implementation can be arbitrarily set by the owner multiple times.

Example:

```
contracts/vote-escrow/BoostManager.sol

SOL

303 function setVeOVIX(IERC20 ve) external onlyOwner {
304     veOVIX = ve;
305     emit VeOVIXUpdated(veOVIX);
306 }
```


Recommendation:

We advise the capability of re-setting the value to be removed from the codebase and allowing to only be set once to avoid the system being compromised by a future update.

Alleviation:

The 0vix team added the recommended `address` sanitizations, removing the aforementioned capability from the contract owner.

BMR-04M: Flash-Loan Prone Balance Measurements

| Type | Severity | Location |
|---------------|------------------------------------------------------------------------------------------|---------------------------------------|
| Logical Fault |  Medium | BoostManager.sol:L250-L262, L264-L276 |

Description:

The two linked functions are prone to flash loan manipulation as they rely on spot evaluations of borrowed and supplied assets to the protocol.

Example:

contracts/vote-escrow/BoostManager.sol

SOL

```
250 function boostedSupplyBalanceOf(address market, address user)
251     public
252     view
253     returns (uint256)
254 {
255     return (
256         calcBoostedBalance(
257             user,
258             supplyBoosterBasis[market][user],
259             IOToken(market).balanceOf(user)
260         )
261     );
262 }
263
264 function boostedBorrowBalanceOf(address market, address user)
265     public
266     view
267     returns (uint256)
268 {
269     return (
270         calcBoostedBalance(
271             user,
272             borrowBoosterBasis[market][user],
273             IOToken(market).borrowBalanceStored(user)
274         )
275     );
276 }
```

Recommendation:


We advise these functions to not be relied on for any form of governance or reward implementation as they will significantly compromise either system.

Alleviation:

The Ovix team has considered this exhibit but opted not to apply a remediation for it in the current iteration of the codebase.

Comptroller Manual Review Findings

COM-01M: Inexistent Retroactive Reward System

| Type | Severity | Location |
|---------------|-----------------------------------------------------------------------------------------|------------------------------------------|
| Logical Fault |  Minor | Comptroller.sol:L1585-L1587, L1636-L1638 |

Description:

The reward system of OVix is not retro-active in contrast to the Compound implementation.

Example:

```
contracts/Comptroller.sol
SOL
1636 if (borrowerIndex == 0 && borrowIndex >= 0) {
1637     borrowerIndex = borrowIndex;
1638 }
```

Recommendation:

We advise the system to be made retroactive by updating the logic of an "initial" index as per the Compound implementation.

Alleviation:

The OVix team has considered this exhibit but applied a improper fix, i.e. keeping the "initial" index to zero (0) instead of using the default initial market index value of 10^{*36} ($1e36$).

COM-02M: Inexistent Sanitization of Market Addition

| Type | Severity | Location |
|--------------------|-----------------------------|-----------------------|
| Input Sanitization | <div><div></div>Minor</div> | Comptroller.sol:L1265 |

Description:

The linked statement pushes a new `oToken` to the `allMarkets` array without validating if it already exists as the original implementation performs.

Example:

```
contracts/Comptroller.sol
SOL
1265allMarkets.push(oToken);
```

Recommendation:

We advise duplicates to be avoided by iterating through all markets and ensuring the newly added one doesn't already exist.

Alleviation:

The Ovox team applied the recommended fix, disallowing for duplicated values to the `allMarkets` array.

COM-03M: Overly Centralized Reward Control

| Type | Severity | Location |
|------------------------|------------------------------|------------------------------------------|
| Centralization Concern | <div><div></div>Medium</div> | Comptroller.sol:L1891-L1893, L1898-L1900 |

Description:

The administrator of the system has newly introduced administrative functionalities that permit them to adjust the boost-tracking contract as well as the reward token itself, rendering the project highly centralized.

Example:

contracts/Comptroller.sol

SOL

```
1888/**
1889 * @notice Set the OVIX token address
1890 */
1891function setVixAddress(address newVixAddress) public onlyAdmin {
1892    vixAddress = newVixAddress;
1893}
1894
1895/**
1896 * @notice Set the booster manager address
1897 */
1898function setBoostManager(address newBoostManager) public onlyAdmin {
1899    boostManager = IBoostManager(newBoostManager);
1900}
```

Recommendation:

We advise this trait of the system to be re-assessed as it currently acts as a significant Single-Point-of-Failure (SPoF) for the system.

Alleviation:

The Ovox team has considered this exhibit but opted not to apply a remediation for it in the current iteration of the codebase.

OErc20 Manual Review Findings

OE0-01M: Improper Invocation of EIP-20 `transfer`

| Type | Severity | Location |
|---------------------|------------------------------|-----------------|
| Standard Conformity | <div><div></div>Medium</div> | OErc20.sol:L150 |

Description:

The linked statement does not properly validate the returned `bool` of the **EIP-20** standard `transfer` function. As the **standard dictates**, callers **must not** assume that `false` is never returned.

Example:

```
contracts/otokens/OErc20.sol
SOL
150 token.transfer(admin, balance);
```

Recommendation:

Since not all standardized tokens are **EIP-20** compliant (such as Tether / USDT), we advise a safe wrapper library to be utilized instead such as `SafeERC20` by OpenZeppelin to opportunistically validate the returned `bool` only if it exists.

Alleviation:

The 0vix team has considered this exhibit but opted not to apply a remediation for it in the current iteration of the codebase.

Ovix Manual Review Findings

OVI-01M: Insecure Elliptic Curve Recovery Mechanism

| Type | Severity | Location |
|-------------------|------------------------------|---------------------|
| Language Specific | <div><div></div>Medium</div> | Ovix.sol:L182, L286 |

Description:

The `ecrecover` function is a low-level cryptographic function that should be utilized after appropriate sanitizations have been enforced on its arguments, namely on the `s` and `v` values. This is due to the inherent trait of the curve to be symmetrical on the x-axis and thus permitting signatures to be replayed with the same `x` value (`r`) but a different `y` value (`s`).

Example:

contracts/governance/Ovix.sol

SOL

```
145 function permit(  
146     address owner,  
147     address spender,  
148     uint rawAmount,  
149     uint deadline,  
150     uint8 v,  
151     bytes32 r,  
152     bytes32 s  
153 ) external {  
154     uint96 amount;  
155     if (rawAmount == type(uint).max) {  
156         amount = type(uint96).max;  
157     } else {  
158         amount = safe96(rawAmount, "O::permit: amount exceeds 96 bits");  
159     }  
160  
161     bytes32 domainSeparator = keccak256(  
162         abi.encode(  
163             DOMAIN_TYPEHASH,  
164             keccak256(bytes(name)),  
165             getChainId(),
```

```

166         address(this)
167     )
168 };
169 bytes32 structHash = keccak256(
170     abi.encode(
171         PERMIT_TYPEHASH,
172         owner,
173         spender,
174         rawAmount,
175         nonces[owner]++,
176         deadline
177     )
178 );
179 bytes32 digest = keccak256(
180     abi.encodePacked("\x19\x01", domainSeparator, structHash)
181 );
182 address signatory = ecrecover(digest, v, r, s);

```

Recommendation:

We advise them to be sanitized by ensuring that v is equal to either 27 or 28 ($v \in \{27, 28\}$) and to ensure that s is existent in the lower half order of the elliptic curve ($0 < s < \text{secp256k1n} \div 2 + 1$) by ensuring it is less than `0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5D576E7357A4501DDFE92F46681B20A1`. A reference implementation of those checks can be observed in the **ECDSA** library of OpenZeppelin and the rationale behind those restrictions exists within **Appendix F of the Yellow Paper**.

Alleviation:

The OviX team applied the recommended fix, removing the possibility of signature malleability.

OVI-02M: Race-Prone Nonce System

| Type | Severity | Location |
|-------------------|----------|---------------------|
| Language Specific | Medium | Ovix.sol:L175, L289 |

Description:

The `nonce` system of the default Compound implementation has been re-purposed for the **EIP-2612** `permit` function of the contract thereby using the same `nonce` for two different purposes.

Example:

```
contracts/governance/Ovix.sol

SOL

169 bytes32 structHash = keccak256(
170     abi.encode(
171         PERMIT_TYPEHASH,
172         owner,
173         spender,
174         rawAmount,
175         nonces[owner]++,
176         deadline
177     )
178 );
```

Recommendation:

We advise separate `nonce` systems to be used for either function as currently race conditions can manifest whereby a vote delegation is not consumed until a `permit` operation using the same `nonce` is detected and thereby prohibited by consuming the vote delegation (and vice-versa).

Alleviation:

The Ovix team has considered this exhibit but opted not to apply a remediation for it in the current iteration of the codebase.

OvixChainlinkOracle Manual Review Findings

OCO-01M: Potentially Restrictive Token Support

| Type | Severity | Location |
|---------------------|-----------------------------|----------------------------------|
| Standard Conformity | <div><div></div>Minor</div> | OvixChainlinkOracle.sol:L44, L55 |

Description:

The linked statement prohibits valid **EIP-20** tokens with more than **18** decimals from ever being supported by the system.

Example:

contracts/chainlink/OvixChainlinkOracle.sol

SOL

44 uint decimalDelta = uint(18).sub(uint(token.decimals()));

Recommendation:

We advise a proper delta evaluation system to be introduced to the codebase that divides by **10** to the power of the decimal delta in case the decimals exceed **18**.

Alleviation:

The Ovix team has opted to remove the linked contract from codebase altogether.

OCO-02M: Authorative Control of Asset Prices

| Type | Severity | Location |
|------------------------|------------------------------|------------------------------------------|
| Centralization Concern | <div><div></div>Medium</div> | OvixChainlinkOracle.sol:L64-L68, L70-L73 |

Description:

The linked functions permit the administrator of the system to arbitrarily set prices for assets that are supported by it.

Example:

```
contracts/chainlink/OvixChainlinkOracle.sol
SOL
64 function setUnderlyingPrice(IOToken oToken, uint underlyingPriceMantissa) e
65     address asset = address(OErc20(address(oToken)).underlying());
66     emit PricePosted(asset, prices[asset], underlyingPriceMantissa, underly
67     prices[asset] = underlyingPriceMantissa;
68 }
```


Recommendation:

We advise this trait of the system to be re-evaluated and potentially prohibited as apart from allowing the owner to manipulate prices permits stale prices from being set within the protocol.

Alleviation:

The Ovix team has opted to remove the linked contract from codebase altogether.

OCO-03M: Permittance of Feed Overriding

| Type | Severity | Location |
|------------------------|------------------------------------------------------------------------------------------|---------------------------------|
| Centralization Concern |  Medium | OvixChainlinkOracle.sol:L75-L79 |

Description:

The `setFeed` function permits an already set feed to be replaced, thereby affecting all pending transactions of the protocol.

Example:

```
contracts/chainlink/OvixChainlinkOracle.sol

SOL

75 function setFeed(string calldata symbol, address feed) external onlyAdmin {
76     require(feed != address(0) && feed != address(this), "invalid feed address");
77     emit FeedSet(feed, symbol);
78     feeds[keccak256(abi.encodePacked(symbol))] = IAggregatorV2V3(feed);
79 }
```

Recommendation:

We advise the function to disallow overwriting existing feed entries. Alternatively, we advise any feed adjustment to be accompanied by a multi-hour cooldown after which the change is applied to allow for proper community due diligence.

Alleviation:

The Ovix team has opted to remove the linked contract from codebase altogether.

OvixChainlinkOracleV2 Manual Review Findings

OCV-01M: Improper Staleness Limit

| Type | Severity | Location |
|--------------------|-----------------------------|--------------------------------|
| Input Sanitization | <div><div></div>Minor</div> | OvixChainlinkOracleV2.sol:L108 |

Description:

The linked `require` check ensures that the `updatedAt` variable specified can at most be 3 seconds in the future, however, this limit is inexcusably small as the blockchain that the project will be deployed to is Polygon with a median processing time of 2.1 seconds which when coupled with the time required to retrieve a particular price and submit it to the blockchain would always exceed the 3 seconds specified.

Example:

contracts/chainlink/OvixChainlinkOracleV2.sol

SOL

```
102 if (block.timestamp > updatedAt) {
103     // reject stale price
104     // validPeriod can be set to 5 mins
105     require(block.timestamp - updatedAt < validPeriod, "bad updatedAt");
106 } else {
107     // reject future timestamp (< 3s is allowed)
108     require(updatedAt - block.timestamp < 3, "bad updatedAt");
109     updatedAt = block.timestamp;
110 }
```

Recommendation:

We advise future price measurements to be either prohibited or have `updatedAt` set to `block.timestamp` without any logical checks as the current checks are ineffectual.

Alleviation:

The OviX team has considered this exhibit but opted not to apply a remediation for it in the current iteration of the codebase.

OCV-02M: Potentially Restrictive Token Support

| Type | Severity | Location |
|---------------------|-----------------------------|------------------------------------|
| Standard Conformity | <div><div></div>Minor</div> | OvixChainlinkOracleV2.sol:L64, L79 |

Description:

The linked statement prohibits valid **EIP-20** tokens with more than **18** decimals from ever being supported by the system.

Example:

```
contracts/chainlink/OvixChainlinkOracleV2.sol
SOL
64  uint decimalDelta = uint(18).sub(uint(token.decimals()));
```


Recommendation:

We advise a proper delta evaluation system to be introduced to the codebase that divides by **10** to the power of the decimal delta in case the decimals exceed **18**.

Alleviation:

The Ovix team has considered this exhibit but opted not to apply a remediation for it in the current iteration of the codebase.

OCV-03M: Authorative Control of Asset Prices

| Type | Severity | Location |
|------------------------|------------------------------------------------------------------------------------------|------------------------------------|
| Centralization Concern |  Medium | OvixChainlinkOracleV2.sol:L96-L114 |

Description:

The linked function permits the administrator of the system to arbitrarily set prices for assets that are supported by it.

Example:

contracts/chainlink/OvixChainlinkOracleV2.sol

SOL

```
96 function setUnderlyingPrice(  
97     address oToken,  
98     uint underlyingPriceMantissa,  
99     uint256 updatedAt  
100 ) external onlyAdmin {  
101     require(underlyingPriceMantissa > 0, "bad price");  
102     if (block.timestamp > updatedAt) {  
103         // reject stale price  
104         // validPeriod can be set to 5 mins  
105         require(block.timestamp - updatedAt < validPeriod, "bad updatedAt")  
106     } else {  
107         // reject future timestamp (< 3s is allowed)  
108         require(updatedAt - block.timestamp < 3, "bad updatedAt");  
109         updatedAt = block.timestamp;  
110     }  
111  
112     emit PricePosted(oToken, prices[oToken].price, underlyingPriceMantissa,  
113     prices[oToken] = PriceData(underlyingPriceMantissa, updatedAt);  
114 }
```


Recommendation:

We advise this trait of the system to be re-evaluated and potentially prohibited as apart from allowing the owner to manipulate prices permits stale prices from being set within the protocol.

Alleviation:

The Ovox team has considered this exhibit but opted not to apply a remediation for it in the current iteration of the codebase.

OCV-04M: Misconstrued Data Staleness System

| Type | Severity | Location |
|---------------|------------------------------------------------------------------------------------------|-------------------------------------------|
| Logical Fault |  Medium | OvixChainlinkOracleV2.sol:L59, L105, L108 |

Description:

The data staleness system utilized by the contract relies entirely on input arguments rather than evaluating data staleness on use.

Example:

contracts/chainlink/OvixChainlinkOracleV2.sol

SOL

```
58  if (prices[address(oToken)].price != 0) {
59      price = prices[address(oToken)].price;
60  } else {
61      price = getChainlinkPrice(getFeed(address(oToken)));
62  }
```


Recommendation:

We advise data staleness to be validated on use by ensuring that whenever `prices` is utilized that its `updatedAt` argument is within a specified window from the current `block.timestamp` and to otherwise fetch an updated price from the Chainlink oracle.

Alleviation:

The Ovix team has considered this exhibit but applied a non-optimized fix, as it should check the stored price against a valid period in the `if` clause instead.

OCV-05M: Permittance of Feed Overriding

| Type | Severity | Location |
|------------------------|------------------------------------------------------------------------------------------|-------------------------------------|
| Centralization Concern |  Medium | OvixChainlinkOracleV2.sol:L116-L129 |

Description:

The `setFeed` function permits an already set feed to be replaced, thereby affecting all pending transactions of the protocol.

Example:

contracts/chainlink/OvixChainlinkOracleV2.sol

SOL

```
116 function setFeed(  
117     address oToken,  
118     address feed,  
119     uint256 heartbeat  
120 ) external onlyAdmin {  
121     require(  
122         feed != address(0) && feed != address(this),  
123         "invalid feed address"  
124     );  
125     heartbeats[IAggregatorV2V3(feed)] = heartbeat;  
126     feeds[oToken] = IAggregatorV2V3(feed);  
127     emit FeedSet(feed, oToken);  
128     emit HeartbeatSet(feed, heartbeat);  
129 }
```

Recommendation:

We advise the function to disallow overwriting existing feed entries. Alternatively, we advise any feed adjustment to be accompanied by a multi-hour cooldown after which the change is applied to allow for proper community due diligence.

Alleviation:

The Ovix team has considered this exhibit but opted not to apply a remediation for it in the current iteration of the codebase.

Unitroller Manual Review Findings

UNI-01M: Inexplicable Introduction of Pending Administrator Bypass

| Type | Severity | Location |
|---------------|------------------------------|--------------------------|
| Logical Fault | <div><div></div>Medium</div> | Unitroller.sol:L131-L136 |

Description:

The `setAdmin` function bypasses the pending administrator scheme imposed by the Compound system and assigns a new administrator directly.

Example:

contracts/Unitroller.sol

SOL

```
131 function setAdmin(address _admin) public {
132     if(msg.sender != admin) revert("Unauthorized");
133     address oldAdmin = admin;
134     admin = _admin;
135     emit NewAdmin(oldAdmin, admin);
136 }
```

Recommendation:

We advise this trait of the system to be omitted from it as it serves no purpose and diminishes the security of the system.

Alleviation:

The Ovox team has considered this exhibit but applied a improper fix, as the `setAdmin` function emits an incorrect event with incorrect `address` values.

VoteController Manual Review Findings

VCR-01M: Potentially Overly Centralized Protocol Functionality

| Type | Severity | Location |
|------------------------|-----------------------------|------------------------------|
| Centralization Concern | <div><div></div>Minor</div> | VoteController.sol:L640-L645 |

Description:

The `totalEmissions` the protocol performs are in complete control of the system's administrator.

Example:

```
contracts/vote-escrow/VoteController.sol
SOL
640 function setTotalEmissions(uint256 _totalEmissions) external onlyAdmin {
641     uint256 oldEmissions = totalEmissions;
642     totalEmissions = _totalEmissions;
643
644     emit TotalEmissionsChanged(oldEmissions, totalEmissions);
645 }
```


Recommendation:

We advise this trait to be re-evaluated as a more decentralized method may be more optimal such as a governance vote setting them.

Alleviation:

The Ovox team has considered this exhibit but opted not to apply a remediation for it in the current iteration of the codebase.

VCR-02M: Truncation of Reward Speed Achieved

| Type | Severity | Location |
|-------------------------|-----------------------------------------------------------------------------------------|-------------------------|
| Mathematical Operations |  Minor | VoteController.sol:L677 |

Description:

The reward speed adjustment for the Compound system performed by the `VoteController` attempts to divide the `reward` expected to two equal parts by dividing by `2`, thus causing truncation.

Impact:

The cumulative reward speed set will be less than the desired one.

Example:

```
contracts/vote-escrow/VoteController.sol

SOL

673 address[] memory addrs = new address[] (1);
674 addrs[0] = addr;
675
676 uint256[] memory rewards = new uint256[] (1);
677 rewards[0] = reward/2;
678
679 // current implementation doesn't differentiate supply and borrow reward sp
680 comp._setRewardSpeeds(addrs, rewards, rewards);
```


Recommendation:

We advise the `_setRewardSpeeds` function to be invoked with the second argument being a `rewards` array with the `reward / 2` entry and the third argument to be a `uint256` array with a `reward - rewards[0]` entry, accounting for any truncation that may occur.

Alleviation:

The OviX team applied a fix that alleviates the aforementioned potential truncation.

VCR-03M: Improper Market Removal Methodology

| Type | Severity | Location |
|---------------|------------------------------------------------------------------------------------------|-----------------------------------------|
| Logical Fault |  Medium | VoteController.sol:L307-L321, L337-L366 |

Description:

The removal of a market will cause a corrupt system state as the `fixedRewardWeights` and `sumWeights` entries are not properly synchronized.

Impact:

The corrupt system state will ultimately cause incorrect rewards to be updated.

Example:

contracts/vote-escrow/VoteController.sol

SOL

```
337 /**
338  * @notice Sets percentage of the emission community can vote upon
339  * @param _markets The struct containing market's address and its fixed wei
340  */
341 function setFixedRewardWeights(Market[] memory _markets) public onlyAdmin {
342     uint256 sumWeights = 0;
343
344     for (uint256 i = 0; i < markets.length(); i++) {
345         sumWeights += fixedRewardWeights[markets.at(i)];
346     }
347
348     for (uint256 i = 0; i < _markets.length; i++) {
349         require(
350             markets.contains(_markets[i].market),
351             "Market is not in the list"
352         );
353
354         uint256 oldWeight = fixedRewardWeights[_markets[i].market];
355         fixedRewardWeights[_markets[i].market] = _markets[i].weight;
356         sumWeights = sumWeights - oldWeight + _markets[i].weight;
357
358         emit FixedWeightChanged(
359             _markets[i].market,
360             oldWeight,
361             _markets[i].weight
362         );
363     }
364 }
```

```
362     },  
363     }  
364  
365     require(sumWeights <= HUNDRED_PERCENT, "New weight(s) too high");  
366 }
```

Recommendation:

We advise the removal of a market to properly synchronize those and all relevant system variables that such an action should, such as the `timeWeight` of an address.

Alleviation:

The Ovix team applied the recommended fix, although a gap of corrupted values still exists between a market removal and the market weights update. We advise the Ovix team to revisit this exhibit and ensure no corrupt data entries remain in the contract.

BoostManager Code Style Findings

BMR-01C: Redundant Ternary Operator

| Type | Severity | Location |
|------------------|-------------------------------------|----------------------|
| Gas Optimization | <div><div></div>Informational</div> | BoostManager.sol:L74 |

Description:

The linked ternary operator is redundant as the result of its evaluation can be yielded by the function directly.

Example:

contracts/vote-escrow/BoostManager.sol

SOL

74 return veBalances[user] == 0 ? false : true;

Recommendation:

We advise the result of the evaluation in its negated form (**!=**) to be yielded by the function instead.

Alleviation:

The 0vix team changed the linked statement to an inequality conditional.

Comptroller Code Style Findings

COM-01C: Inexistent Visibility Specifier

| Type | Severity | Location |
|------------|-------------------------------------|----------------------|
| Code Style | <div><div></div>Informational</div> | Comptroller.sol:L137 |

Description:

The linked variable has no visibility specifier explicitly set.

Example:

```
contracts/Comptroller.sol
SOL
137 address vixAddress;
```

Recommendation:

We advise one to be set so to avoid potential compilation discrepancies in the future as the current behaviour is for the compiler to assign one automatically which may deviate between `pragma` versions.

Alleviation:

The 0vix team applied the recommended fix.

COM-02C: Mislabeledled Local Variable

| Type | Severity | Location |
|------------|-------------------------------------|-------------------------------------|
| Code Style | <div><div></div>Informational</div> | Comptroller.sol:L1513, L1544, L1675 |

Description:

The linked variable is labelled as `deltaBlocks` while it represents a delta between timestamps.

Example:

contracts/Comptroller.sol

SOL

1675uint256 deltaBlocks = timestamp - lastContributorTimestamp[contributor];


Recommendation:

We advise the variable to be aptly renamed.

Alleviation:

The Ovix team applied the recommended fix.

COM-03C: Redundantly Named Function Arguments

| Type | Severity | Location |
|------------------|-------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Gas Optimization |  Informational | Comptroller.sol:L329, L332, L359-L362, L365-L368, L441-L442, L447-L448, L530-L532, L535-L537, L555-L557, L560-L562, L582-L586, L589-L593, L612, L617, L678-L683, L686-L691, L712, L718, L751-L755, L758-L762, L810-L813, L816-L819 |

Description:

The linked function arguments are redundantly given an explicit name and raise compiler issues unless the redundant referencing statements exist in the code as linked.

Example:

```
contracts/Comptroller.sol
SOL
326 function mintAllowed(
327     address oToken,
328     address minter,
329     uint256 mintAmount
330 ) external override returns (uint256) {
331     // Pausing is a very serious situation - we revert to sound the alarms
332     mintAmount; // not used yet
```

Recommendation:

We advise the code to instead omit the names from the function signature declarations to avoid the compilation error entirely and still comply with the relevant interfaces of the system.

Alleviation:

The Ovox team has considered this exhibit but opted not to apply a remediation for it in the current iteration of the codebase.

COM-04C: Variable Mutability Specifier (Immutable)

| Type | Severity | Location |
|------------------|-------------------------------------|----------------------|
| Gas Optimization | <div><div></div>Informational</div> | Comptroller.sol:L146 |

Description:

The linked variable is assigned to only once during the contract's `constructor`.

Example:

contracts/Comptroller.sol

SOL

```
145 constructor() {  
146     admin = msg.sender;  
147 }
```

Recommendation:

We advise it to be set as `immutable` greatly optimizing its read-access gas cost.

Alleviation:

The 0vix team has considered this exhibit but opted not to apply a remediation for it in the current iteration of the codebase.

ExponentialNoError Code Style Findings

ENE-01C: Deprecated Numeric Representation

| Type | Severity | Location |
|------------|-------------------------------------|---------------------------------|
| Code Style | <div><div></div>Informational</div> | ExponentialNoError.sol:L78, L83 |

Description:

The linked numeric representations are meant to represent the limit of their corresponding type (`uint224` and `uint32` respectively) but do so in a deprecated way.

Example:

```
contracts/libraries/ExponentialNoError.sol
SOL
77 require(n < 2**224, "safe224 overflow");
78     return uint224(n);
79 }
80
81 function safe32(uint n) pure internal returns (uint32) {
82     require(n < 2**32, "safe32 overflow");
83     return uint32(n);
84 }
```

Recommendation:

We advise the comparisons to be adjusted to inclusive ones (`<=`) and the literals to be replaced by the `type(uintXX).max` representation of the respective type.

Alleviation:

The Ovox team applied the recommended fix.

JumpRateModel Code Style Findings

JRM-01C: Documentation Typo

| Type | Severity | Location |
|------------|-------------------------------------|---------------------------------------|
| Code Style | <div><div></div>Informational</div> | JumpRateModel.sol:L76, L80, L95, L100 |

Description:

The linked documentation lines contain a typographic error.

Example:

contracts/interest-rate-models/JumpRateModel.sol

SOL

```
75  /**
76   * @notice Calculates the current borrow rate per timestmp, with the error
77   * @param cash The amount of cash in the market
78   * @param borrows The amount of borrows in the market
79   * @param reserves The amount of reserves in the market
80   * @return The borrow rate percentage per timestmp as a mantissa (scaled by
81   */
```


Recommendation:

We advise it to be corrected.

Alleviation:

The Ovox team has considered this exhibit but opted not to apply a remediation for it in the current iteration of the codebase.

JRM-02C: Redundant Usage of SafeMath

| Type | Severity | Location |
|-------------------|-------------------------------------------------------------------------------------------------|---------------------------|
| Language Specific |  Informational | JumpRateModel.sol:L2, L12 |

Description:

The linked contract utilizes `SafeMath` when it is compiled with a `pragma` version of `0.8.x`.

Example:

```
contracts/interest-rate-models/JumpRateModel.sol
SOL
2  pragma solidity 0.8.4;
3
4  import "../interfaces/IInterestRateModel.sol";
5  import "../libraries/SafeMath.sol";
6
7  /**
8   * @title 0VIX's JumpRateModel Contract
9   * @author 0VIX
10  */
11  contract JumpRateModel is IInterestRateModel {
12      using SafeMath for uint;
```

Recommendation:

Given that safe arithmetics are toggled on by default in these versions, we advise the usage of `SafeMath` to be omitted from the codebase as it incurs an additional gas cost at verbosol benefit.

Alleviation:

The 0vix team applied the recommended fix.

JRM-03C: Variable Mutability Specifiers (Immutable)

| Type | Severity | Location |
|------------------|-------------------------------------|--------------------------------------|
| Gas Optimization | <div><div></div>Informational</div> | JumpRateModel.sol:L51, L52, L53, L54 |

Description:

The linked variables are assigned to only once during the contract's `constructor`.

Example:

contracts/interest-rate-models/JumpRateModel.sol

SOL

```
50 constructor(uint baseRatePerYear, uint multiplierPerYear, uint jumpMultipli
51     baseRatePerTimestamp = baseRatePerYear.mul(1e18).div(timestampsPerYear)
52     multiplierPerTimestamp = multiplierPerYear.mul(1e18).div(timestampsPerY
53     jumpMultiplierPerTimestamp = jumpMultiplierPerYear.mul(1e18).div(timest
54     kink = kink_;
55
56     emit NewInterestParams(baseRatePerTimestamp, multiplierPerTimestamp, ju
57 }
```

Recommendation:

We advise them to be set as `immutable` greatly optimizing their read-access gas cost.

Alleviation:

The Ovox team applied the recommended fix.

Maximillion Code Style Findings

MAX-01C: Variable Mutability Specifier (Immutable)

| Type | Severity | Location |
|------------------|-------------------------------------|---------------------|
| Gas Optimization | <div><div></div>Informational</div> | Maximillion.sol:L20 |

Description:

The linked variable is assigned to only once during the contract's `constructor`.

Example:

contracts/Maximillion.sol

SOL

```
19  constructor(OMatic oMatic_) {
20      oMatic = oMatic_;
21  }
```

Recommendation:

We advise it to be set as `immutable` greatly optimizing its read-access gas cost.

Alleviation:

The 0vix team applied the recommended fix.

OToken Code Style Findings

OTN-01C: Inefficient Event Arguments

| Type | Severity | Location |
|------------------|-------------------------------------|------------------|
| Gas Optimization | <div><div></div>Informational</div> | OToken.sol:L1818 |

Description:

The linked events are meant to emit the previous and new value of a storage variable being adjusted, however, to do so they redundantly use a local variable.

Example:

contracts/otokens/abstract/OToken.sol

SOL

```
1814function setAdmin(address payable _admin) public {
1815    require(msg.sender == admin, "Unauthorized");
1816    address oldAdmin = admin;
1817    admin = _admin;
1818    emit NewAdmin(oldAdmin, admin);
1819}
```


Recommendation:

We advise the emission to occur prior to the assignment of each storage variable by setting the first argument of the event as the existing storage value and the second argument as the input argument of the function.

Alleviation:

The Ovox team applied the recommended fix.

OTN-02C: Inexplicable Flag Introduction

| Type | Severity | Location |
|------------------|-------------------------------------------------------------------------------------------------|-----------------|
| Gas Optimization |  Informational | OToken.sol:L961 |

Description:

The original Compound codebase of the borrowable tokens did not allow the specification of a "maximum" `redeemAmountIn` flag and thus only performed a `divScalarByExpTruncate` instruction. In the new implementation by OVix, the flag is present and allows a `mulScalarTruncate` operation to be performed instead that is a one-to-one equivalent of specifying `redeemTokensIn` as `type(uint256).max`.

Example:

contracts/otokens/abstract/OToken.sol

SOL

```
930 /* If redeemTokensIn > 0: */
931 if (redeemTokensIn > 0) {
932     /*
933      * We calculate the exchange rate and the amount of underlying to be re
934      * redeemTokens = redeemTokensIn
935      * redeemAmount = redeemTokensIn x exchangeRateCurrent
936      */
937     if (redeemTokensIn == type(uint256).max) {
938         vars.redeemTokens = accountTokens[redeemer];
939     } else {
940         vars.redeemTokens = redeemTokensIn;
941     }
942
943     (vars.mathErr, vars.redeemAmount) = mulScalarTruncate(
944         Exp({mantissa: vars.exchangeRateMantissa}),
945         vars.redeemTokens
946     );
947     if (vars.mathErr != MathError.NO_ERROR) {
948         return
949             failOpaque(
950                 Error.MATH_ERROR,
951                 FailureInfo.REDEEM_EXCHANGE_TOKENS_CALCULATION_FAILED,
952                 uint256(vars.mathErr)
953             );
954     }
955 } else {
```

```

956     /*
957     * We get the current exchange rate and calculate the amount to be rede
958     * redeemTokens = redeemAmountIn / exchangeRate
959     * redeemAmount = redeemAmountIn
960     */
961     if (redeemAmountIn == type(uint256).max) {
962         vars.redeemTokens = accountTokens[redeemer];
963
964         (vars.mathErr, vars.redeemAmount) = mulScalarTruncate(
965             Exp({mantissa: vars.exchangeRateMantissa}),
966             vars.redeemTokens
967         );
968         if (vars.mathErr != MathError.NO_ERROR) {
969             return
970                 failOpaque(
971                     Error.MATH_ERROR,
972                     FailureInfo
973                         .REDEEM_EXCHANGE_TOKENS_CALCULATION_FAILED,
974                     uint256(vars.mathErr)
975                 );
976         }
977     } else {
978         vars.redeemAmount = redeemAmountIn;
979
980         (vars.mathErr, vars.redeemTokens) = divScalarByExpTruncate(
981             redeemAmountIn,
982             Exp({mantissa: vars.exchangeRateMantissa})
983         );
984         if (vars.mathErr != MathError.NO_ERROR) {
985             return
986                 failOpaque(
987                     Error.MATH_ERROR,
988                     FailureInfo
989                         .REDEEM_EXCHANGE_AMOUNT_CALCULATION_FAILED,
990                     uint256(vars.mathErr)
991                 );
992         }
993     }
994 }

```

Recommendation:

Given that the functionality is already exposed by the codebase, we advise the flag to be omitted to reduce the complexity of the redemption amount based `redeemFresh` code flow.

Alleviation:

The Ovox team applied the recommended fix.

Ovix Code Style Findings

OVI-01C: Deprecated Numeric Representation

| Type | Severity | Location |
|------------|-------------------------------------|---------------------|
| Code Style | <div><div></div>Informational</div> | Ovix.sol:L463, L472 |

Description:

The linked numeric representations are meant to represent the limit of their corresponding type (`uint32` and `uint96` respectively) but do so in a deprecated way.

Example:

contracts/governance/Ovix.sol

SOL

```
458 function safe32(uint n, string memory errorMessage)
459     internal
460     pure
461     returns (uint32)
462 {
463     require(n < 2**32, errorMessage);
464     return uint32(n);
465 }
466
467 function safe96(uint n, string memory errorMessage)
468     internal
469     pure
470     returns (uint96)
471 {
472     require(n < 2**96, errorMessage);
473     return uint96(n);
474 }
```

Recommendation:

We advise the comparisons to be adjusted to inclusive ones (`<=`) and the literals to be replaced by the `type(uintXX).max` representation of the respective type.

Alleviation:

The OviX team partially applied the recommended fix, as the last unsigned integer of a said type is not included.

OVI-02C: Inefficient Casting Operations

| Type | Severity | Location |
|-------------------------|-------------------------------------|---------------------|
| Mathematical Operations | <div><div></div>Informational</div> | Ovix.sol:L464, L473 |

Description:

The linked casting operations are inefficient as their safety is guaranteed by the `require` checks that precede them.

Example:

contracts/governance/Ovix.sol

SOL

```
458 function safe32(uint n, string memory errorMessage)
459     internal
460     pure
461     returns (uint32)
462 {
463     require(n < 2**32, errorMessage);
464     return uint32(n);
465 }
```


Recommendation:

We advise them to be performed within an `unchecked` code block optimizing their gas cost.

Alleviation:

The Ovix team has considered this exhibit but opted not to apply a remediation for it in the current iteration of the codebase.

OVI-03C: Variable Mutability Specifier (Immutable)

| Type | Severity | Location |
|------------------|-------------------------------------------------------------------------------------------------|--------------|
| Gas Optimization |  Informational | Ovix.sol:L94 |

Description:

The linked variable is assigned to only once during the contract's `constructor`.

Example:

```
contracts/governance/Ovix.sol

SOL

86  constructor(
87      address account,
88      string memory _name,
89      string memory _symbol,
90      uint _supply
91  ) {
92      name = _name;
93      symbol = _symbol;
94      totalSupply = _supply * 1 ether;
95      balances[account] = uint96(totalSupply);
96      emit Transfer(address(0), account, totalSupply);
97  }
```

Recommendation:

We advise it to be set as `immutable` greatly optimizing its read-access gas cost.

Alleviation:

The Ovix team has considered this exhibit but opted not to apply a remediation for it in the current iteration of the codebase.

OvixChainlinkOracle Code Style Findings

OCO-01C: Inefficient Event Arguments

| Type | Severity | Location |
|------------------|-------------------------------------|-----------------------------|
| Gas Optimization | <div><div></div>Informational</div> | OvixChainlinkOracle.sol:L97 |

Description:

The linked event is meant to emit the previous and new value of a storage variable being adjusted, however, to do so it redundantly uses a local variable.

Example:

```
contracts/chainlink/OvixChainlinkOracle.sol
SOL
93 function setAdmin(address newAdmin) external onlyAdmin {
94     address oldAdmin = admin;
95     admin = newAdmin;
96
97     emit NewAdmin(oldAdmin, newAdmin);
98 }
```


Recommendation:

We advise the emission to occur prior to the assignment of the storage variable by setting the first argument of the event as the existing storage value and the second argument as the input argument of the function.

Alleviation:

The Ovix team has opted to remove the linked contract from codebase altogether.

OCO-02C: Inefficient **mapping** Lookups

| Type | Severity | Location |
|------------------|-------------------------------------------------------------------------------------------------|----------------------------------|
| Gas Optimization |  Informational | OvixChainlinkOracle.sol:L38, L39 |

Description:

The linked statements perform key-based lookup operations on **mapping** declarations from storage multiple times for the same key redundantly.

Example:

contracts/chainlink/OvixChainlinkOracle.sol

SOL

```
38  if (prices[address(token)] != 0) {  
39      price = prices[address(token)];  
40  } else {  
41      price = getChainlinkPrice(getFeed(token.symbol()));  
42  }
```

Recommendation:

As the lookups internally perform an expensive **keccak256** operation, we advise the lookups to be cached wherever possible to a single local declaration that either holds the value of the **mapping** in case of primitive types or holds a **storage** pointer to the **struct** contained.

Alleviation:

The Ovix team has opted to remove the linked contract from codebase altogether.

OCO-03C: Inexistent Visibility Specifier

| Type | Severity | Location |
|------------|-------------------------------------|-----------------------------|
| Code Style | <div><div></div>Informational</div> | OvixChainlinkOracle.sol:L13 |

Description:

The linked variable has no visibility specifier explicitly set.

Example:

contracts/chainlink/OvixChainlinkOracle.sol

SOL

13 string nativeSymbol;

Recommendation:

We advise one to be set so to avoid potential compilation discrepancies in the future as the current behaviour is for the compiler to assign one automatically which may deviate between `pragma` versions.

Alleviation:

The Ovix team has opted to remove the linked contract from codebase altogether.

OCO-04C: Misleading Behaviour Comment

| Type | Severity | Location |
|------------|----------------------------|---------------------------------|
| Code Style | Informational | OvixChainlinkOracle.sol:L45-L50 |

Description:

The linked comment indicates that the conditional that follows protects the `price` from being multiplied by `0`, however, the operation `10**0` would yield a multiplication with `1` and thus not affect the code's operation.

Example:

```
contracts/chainlink/OvixChainlinkOracle.sol
SOL
45  // Ensure that we don't multiply the result by 0
46  if (decimalDelta > 0) {
47      return price.mul(10**decimalDelta);
48  } else {
49      return price;
50  }
```

Recommendation:

While a gas benefit is observed by the current code structure, we advise a simple ternary operator to be utilized instead unless the decimal delta exhibit is applied in which case only the comment should be removed.

Alleviation:

The Ovix team has opted to remove the linked contract from codebase altogether.

OCO-05C: Redundant Usage of SafeMath

| Type | Severity | Location |
|-------------------|----------------------------|---------------------------------|
| Language Specific | Informational | OvixChainlinkOracle.sol:L2, L11 |

Description:

The linked contract utilizes `SafeMath` when it is compiled with a `pragma` version of `0.8.x`.

Example:

```
contracts/chainlink/OvixChainlinkOracle.sol
SOL
2  pragma solidity 0.8.4;
3
4  import "../PriceOracle.sol";
5  import "../otokens/OErc20.sol";
6  import "../otokens/interfaces/IEIP20.sol";
7  import "../libraries/SafeMath.sol";
8  import "../interfaces/IAggregatorV2V3.sol";
9
10 contract OvixChainlinkOracle is PriceOracle {
11     using SafeMath for uint;
```

Recommendation:

Given that safe arithmetics are toggled on by default in these versions, we advise the usage of `SafeMath` to be omitted from the codebase as it incurs an additional gas cost at verbosal benefit.

Alleviation:

The Ovix team has opted to remove the linked contract from codebase altogether.

OvixChainlinkOracleV2 Code Style Findings

OCV-01C: Inefficient Event Arguments

| Type | Severity | Location |
|------------------|-------------------------------------|--------------------------------|
| Gas Optimization | <div><div></div>Informational</div> | OvixChainlinkOracleV2.sol:L151 |

Description:

The linked event is meant to emit the previous and new value of a storage variable being adjusted, however, to do so it redundantly uses a local variable.

Example:

```
contracts/chainlink/OvixChainlinkOracleV2.sol
SOL
147 function setAdmin(address newAdmin) external onlyAdmin {
148     address oldAdmin = admin;
149     admin = newAdmin;
150
151     emit NewAdmin(oldAdmin, newAdmin);
152 }
```

Recommendation:

We advise the emission to occur prior to the assignment of the storage variable by setting the first argument of the event as the existing storage value and the second argument as the input argument of the function.

Alleviation:

The Ovix team applied the recommended fix.

OCV-02C: Inefficient Initialization of Contract

| Type | Severity | Location |
|------------------|-------------------------------------|------------------------------------------|
| Gas Optimization | <div><div></div>Informational</div> | OvixChainlinkOracleV2.sol:L40, L154-L156 |

Description:

The linked setter function is invoked from the `constructor` of the contract and inefficiently applies access control.

Example:

contracts/chainlink/OvixChainlinkOracleV2.sol

SOL

```
154 function setOMatic(address _oMatic) public onlyAdmin {
155     oMatic = _oMatic;
156 }
```


Recommendation:

We advise the code within `setOMatic` to be refactored to an `internal` / `private` function that is consequently utilized by both the `constructor` and the `public` facing function.

Alleviation:

The Ovix team applied the recommended fix.

OCV-03C: Inefficient **mapping** Lookups

| Type | Severity | Location |
|------------------|-------------------------------------------------------------------------------------------------|------------------------------------------------------------|
| Gas Optimization |  Informational | OvixChainlinkOracleV2.sol:L58, L59, L112, L113, L135, L136 |

Description:

The linked statements perform key-based lookup operations on **mapping** declarations from storage multiple times for the same key redundantly.

Example:

contracts/chainlink/OvixChainlinkOracleV2.sol

SOL

```
131 function setHeartbeat(address oToken, uint256 heartbeat)
132     external
133     onlyAdmin
134 {
135     heartbeats[feeds[oToken]] = heartbeat;
136     emit HeartbeatSet(address(feeds[oToken]), heartbeat);
137 }
```

Recommendation:

As the lookups internally perform an expensive **keccak256** operation, we advise the lookups to be cached wherever possible to a single local declaration that either holds the value of the **mapping** in case of primitive types or holds a **storage** pointer to the **struct** contained.

Alleviation:

The Ovix team applied the recommended fix, although in only one of the many exhibits.

OCV-04C: Inexistent Visibility Specifier

| Type | Severity | Location |
|------------|-------------------------------------|-------------------------------|
| Code Style | <div><div></div>Informational</div> | OvixChainlinkOracleV2.sol:L22 |

Description:

The linked variable has no visibility specifier explicitly set.

Example:

```
contracts/chainlink/OvixChainlinkOracleV2.sol
SOL
22 mapping(IAggregatorV2V3 => uint256) heartbeats;
```

Recommendation:

We advise one to be set so to avoid potential compilation discrepancies in the future as the current behaviour is for the compiler to assign one automatically which may deviate between `pragma` versions.

Alleviation:

The Ovix team applied the recommended fix.

OCV-05C: Redundant Usage of SafeMath

| Type | Severity | Location |
|-------------------|----------------------------|-----------------------------------|
| Language Specific | Informational | OvixChainlinkOracleV2.sol:L2, L11 |

Description:

The linked contract utilizes `SafeMath` when it is compiled with a `pragma` version of `0.8.x`.

Example:

```
contracts/chainlink/OvixChainlinkOracleV2.sol
SOL
2  pragma solidity 0.8.4;
3
4  import "../PriceOracle.sol";
5  import "../otokens/OErc20.sol";
6  import "../otokens/interfaces/IEIP20.sol";
7  import "../libraries/SafeMath.sol";
8  import "../interfaces/IAggregatorV2V3.sol";
9
10 contract OvixChainlinkOracleV2 is PriceOracle {
11     using SafeMath for uint;
```

Recommendation:

Given that safe arithmetics are toggled on by default in these versions, we advise the usage of `SafeMath` to be omitted from the codebase as it incurs an additional gas cost at verbosal benefit.

Alleviation:

The Ovix team applied the recommended fix.

Unitroller Code Style Findings

UNI-01C: Deprecated `if-revert` Pattern

| Type | Severity | Location |
|------------|-------------------------------------|---------------------|
| Code Style | <div><div></div>Informational</div> | Unitroller.sol:L132 |

Description:

The linked statement evaluates a conditional in an `if` clause and performs a `revert` statement after it with a textual argument.

Example:

contracts/Unitroller.sol

SOL

```
132 if(msg.sender != admin) revert("Unauthorized");
```

Recommendation:

We advise either a `require` check to be introduced or a proper `revert` to be set that uses a custom `error` defined at the contract level, the former of which we advise.

Alleviation:

The Ovix team applied the recommended fix.

VoteController Code Style Findings

VCR-01C: Illegible Numeric Value Representations

| Type | Severity | Location |
|------------|-------------------------------------|------------------------------|
| Code Style | <div><div></div>Informational</div> | VoteController.sol:L45, L195 |

Description:

The linked representations of numeric literals are sub-optimally represented decreasing the legibility of the codebase.

Example:

contracts/vote-escrow/VoteController.sol

SOL

45 uint256 public constant HUNDRED_PERCENT = 10000;

Recommendation:

To properly illustrate each value's purpose, we advise the following guidelines to be followed. For values meant to depict fractions with a base of `1e18`, we advise fractions to be utilized directly (i.e. `1e17` becomes `0.1e18`) as they are supported. For values meant to represent a percentage base, we advise each value to utilize the underscore (`_`) separator to discern the percentage decimal (i.e. `10000` becomes `100_00`, `300` becomes `3_00` and so on). Finally, for large numeric values we simply advise the underscore character to be utilized again to represent them (i.e. `1000000` becomes `1_000_000`).

Alleviation:

The Ovox team applied the recommended fix.

VCR-02C: Inefficient Loop Limit Evaluations

| Type | Severity | Location |
|------------------|-------------------------------------|-------------------------------------|
| Gas Optimization | <div><div></div>Informational</div> | VoteController.sol:L344, L648, L665 |

Description:

The linked `for` loops evaluate their limit inefficiently on each iteration.

Example:

contracts/vote-escrow/VoteController.sol

SOL

344 for (uint256 i = 0; i < markets.length(); i++) {

Recommendation:

We advise the statements within the `for` loop limits to be relocated outside to a local variable declaration that is consequently utilized for the evaluations to significantly reduce the codebase's gas cost. We should note the same optimization is applicable for storage reads present in those limits as they are newly read on each iteration (i.e. `length` members of arrays in storage).

Alleviation:

The Ovix team applied the recommended fix.

VCR-03C: Inexistent Error Messages

| Type | Severity | Location |
|------------|-------------------------------------|-------------------------------------|
| Code Style | <div><div></div>Informational</div> | VoteController.sol:L183, L184, L185 |

Description:

The linked `require` checks have no error messages explicitly defined.

Example:

contracts/vote-escrow/VoteController.sol

SOL

183 `require(_votingEscrow != address(0));`

Recommendation:

We advise each to be set so to increase the legibility of the codebase and aid in validating the `require` checks' conditions.

Alleviation:

The 0vix team applied the recommended fix.

VCR-04C: Repetitive Value Literal

| Type | Severity | Location |
|------------|-------------------------------------|-------------------------------|
| Code Style | <div><div></div>Informational</div> | VoteController.sol:L231, L265 |

Description:

The linked value literal is repeated across the codebase multiple times.

Example:

contracts/vote-escrow/VoteController.sol

SOL

231 for (uint256 i = 0; i < 500; i++) {


Recommendation:

We advise it to be set to a `constant` variable instead optimizing the legibility of the codebase.

Alleviation:

The 0vix team applied the recommended fix.

VCR-05C: Same-Purpose Variables

| Type | Severity | Location |
|------------------|-------------------------------------------------------------------------------------------------|-----------------------------|
| Gas Optimization |  Informational | VoteController.sol:L30, L90 |

Description:

The linked variables indirectly achieve the same purpose, illustrating whether a particular market is part of the votable pool.

Example:

contracts/vote-escrow/VoteController.sol

SOL

```
286 /**
287  * @notice Add market `addr` essentially making it votable; manual fixedWei
288  * @dev admin only
289  * @param addr Market address
290  */
291 function addMarket(address addr) external onlyAdmin {
292     require(!isVotable[addr], "Cannot add the same market twice");
293     require(comp.isMarket(addr), "address is not an 0vix market");
294     isVotable[addr] = true;
295
296     markets.add(addr);
297
298     uint256 nextTime = ((block.timestamp + PERIOD) / PERIOD) * PERIOD;
299
300     if (timeTotal == 0) timeTotal = nextTime;
301
302     timeWeight[addr] = nextTime;
303
304     emit NewMarket(addr);
305 }
306
307 /**
308  * @notice Remove market `addr` essentially making it non-votable; manual f
309  * @dev admin only
310  * @param addr Market address
311  */
312 function removeMarket(address addr) external onlyAdmin {
313     require(isVotable[addr], "Market doesn't exist");
314     isVotable[addr] = false;
315 }
```

```
315  
316     markets.remove(addr);  
317  
318     // todo test what happens with market's lists (e.g. timeWeight[addr]) w  
319  
320     emit MarketRemoved(addr);  
321 }
```

Recommendation:

We advise either of the two to be utilized, preferably the `markets` enumerable set, to optimize the system's purpose.

Alleviation:

The Oviz team applied the recommended fix and removed the redundant state variable.

Finding Types

A description of each finding type included in the report can be found below and is linked by each respective finding. A full list of finding types Omniscia has defined will be viewable at the central audit methodology we will publish soon.

External Call Validation

Many contracts that interact with DeFi contain a set of complex external call executions that need to happen in a particular sequence and whose execution is usually taken for granted whereby it is not always the case. External calls should always be validated, either in the form of `require` checks imposed at the contract-level or via more intricate mechanisms such as invoking an external getter-variable and ensuring that it has been properly updated.

Input Sanitization

As there are no inherent guarantees to the inputs a function accepts, a set of guards should always be in place to sanitize the values passed in to a particular function.

Indeterminate Code

These types of issues arise when a linked code segment may not behave as expected, either due to mistyped code, convoluted `if` blocks, overlapping functions / variable names and other ambiguous statements.

Language Specific

Language specific issues arise from certain peculiarities that the Solidity language boasts that discerns it from other conventional programming languages. For example, the EVM is a 256-bit machine meaning that operations on less-than-256-bit types are more costly for the EVM in terms of gas costs, meaning that loops utilizing a `uint8` variable because their limit will never exceed the 8-bit range actually cost more than redundantly using a `uint256` variable.

Code Style

An official Solidity style guide exists that is constantly under development and is adjusted on each new Solidity release, designating how the overall look and feel of a codebase should be. In these types of findings, we identify whether a project conforms to a particular naming convention and whether that convention is consistent within the codebase and legible. In case of inconsistencies, we point them out under this category. Additionally, variable shadowing falls under this category as well which is identified when a local-level variable contains the same name as a contract-level variable that is present in the inheritance chain of the local execution level's context.

Gas Optimization

Gas optimization findings relate to ways the codebase can be optimized to reduce the gas cost involved with interacting with it to various degrees. These types of findings are completely optional and are pointed out for the benefit of the project's developers.

Standard Conformity

These types of findings relate to incompatibility between a particular standard's implementation and the project's implementation, oftentimes causing significant issues in the usability of the contracts.

Mathematical Operations

In Solidity, math generally behaves differently than other programming languages due to the constraints of the EVM. A prime example of this difference is the truncation of values during a division which in turn leads to loss of precision and can cause systems to behave incorrectly when dealing with percentages and proportion calculations.

Logical Fault

This category is a bit broad and is meant to cover implementations that contain flaws in the way they are implemented, either due to unimplemented functionality, unaccounted-for edge cases or similar extraordinary scenarios.

Centralization Concern

This category covers all findings that relate to a significant degree of centralization present in the project and as such the potential of a Single-Point-of-Failure (SPoF) for the project that we urge them to re-consider and potentially omit.

Reentrant Call

This category relates to findings that arise from re-entrant external calls (such as EIP-721 minting operations) and revolve around the inapplicacy of the Checks-Effects-Interactions (CEI) pattern, a pattern that dictates checks (`require` statements etc.) should occur before effects (local storage updates) and interactions (external calls) should be performed last.

Disclaimer

The following disclaimer applies to all versions of the audit report produced (preliminary / public / private) and is in effect for all past, current, and future audit reports that are produced and hosted under Omniscia:

IMPORTANT TERMS & CONDITIONS REGARDING OUR SECURITY AUDITS/REVIEWS/REPORTS AND ALL PUBLIC/PRIVATE CONTENT/DELIVERABLES

Omniscia ("Omniscia") has conducted an independent security review to verify the integrity of and highlight any vulnerabilities, bugs or errors, intentional or unintentional, that may be present in the codebase that were provided for the scope of this Engagement.

Blockchain technology and the cryptographic assets it supports are nascent technologies. This makes them extremely volatile assets. Any assessment report obtained on such volatile and nascent assets may include unpredictable results which may lead to positive or negative outcomes.

In some cases, services provided may be reliant on a variety of third parties. This security review does not constitute endorsement, agreement or acceptance for the Project and technology that was reviewed. Users relying on this security review should not consider this as having any merit for financial advice or technological due diligence in any shape, form or nature.

The veracity and accuracy of the findings presented in this report relate solely to the proficiency, competence, aptitude and discretion of our auditors. Omniscia and its employees make no guarantees, nor assurance that the contracts are free of exploits, bugs, vulnerabilities, deprecation of technologies or any system / economical / mathematical malfunction.

This audit report shall not be printed, saved, disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Omniscia.

All the information/opinions/suggestions provided in this report does not constitute financial or investment advice, nor should it be used to signal that any person reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report.

Information in this report is provided 'as is'. Omniscia is under no covenant to the completeness, accuracy or solidity of the contracts reviewed. Omniscia's goal is to help reduce the attack vectors/surface and the high level of variance associated with utilizing new and consistently changing technologies.

Omniscia in no way claims any guarantee, warranty or assurance of security or functionality of the technology that was in scope for this security review.

In no event will Omniscia, its partners, employees, agents or any parties related to the design/creation of this security review be ever liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this security review.

Cryptocurrencies and all other technologies directly or indirectly related to cryptocurrencies are not standardized, highly prone to malfunction and extremely speculative by nature. No due diligence and/or safeguards may be insufficient and users should exercise maximum caution when participating and/or investing in this nascent industry.

The preparation of this security review has made all reasonable attempts to provide clear and actionable recommendations to the Project team (the "client") with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts in scope for this engagement.

All services, the security reports, discussions, work product, attack vectors description or any other materials, products or results of this security review engagement is provided "as is" and "as available" and with all faults, uncertainty and defects without warranty or guarantee of any kind.

Omniscia will assume no liability or responsibility for delays, errors, mistakes, or any inaccuracies of content, suggestions, materials or for any loss, delay, damage of any kind which arose as a result of this engagement/security review.

Omniscia will assume no liability or responsibility for any personal injury, property damage, of any kind whatsoever that resulted in this engagement and the customer having access to or use of the products, engineers, services, security report, or any other other materials.

For avoidance of doubt, this report, its content, access, and/or usage thereof, including any associated services or materials, shall not be considered or relied upon as any form of financial, investment, tax, legal, regulatory, or any other type of advice.