# Olympus V2 Protocol Audit

Enclosed is the V2 Audit from Omniscia. A URL for an HTML version is also listed below.

### Source Code:

https://github.com/OlympusDAO/olympus-contracts/tree/182cfdb29f781c418819fc699c71760ec1dca7e3

### **URL for Audit Content:**

https://omniscia.io/olympus-dao-protocol-v2/

### Important Notes:

All findings in this Audit have been alleviated either via code changes or commentary within this audit. There are two Major findings that are listed as not addressed; however, in both cases the Olympus Engineering team deployed alleviations in the final, live contracts.

# **Protocol V Security Audit**

We were tasked with performing a second round audit on the version 2 implementation of the Olympus DAO protocol composed of a complex system architecture involving a triple token system, an LP-based bond system, and utility contracts for incentivizing the use of all three token types.

Over the course of the audit, we were able to pinpoint potentially harmful arbitrage opportunities that can arise in the conversion between the three tokens as well as a potential under-pricing flaw in the bond creation mechanism that if exploited could cause a bond to be priced at a very low value and thus cause a significant evaluation of an otherwise small deposit.

In addition to logical flaws, we identified several optimizations that can be applied to the codebase that we urge the Olympus DAO team to consider. Overall, the codebase appears to be at an unpolished state and can be significantly improved in terms of styling, consistency, and documentation. For the former, we advise a linting plugin to be enforced on the codebase to greatly increase its readability.

Another important point that should be raised about the codebase is the over-reliance on good faith of the various authorized operators in the protocol. As an example, the terms of a bond are not validated and permit arbitrary values for all terms whilst they are only set by the guardian of the protocol. As we have expressed in some of the exhibits, we advise the Olympus DAO team to attempt to further decentralize the operation of the protocol by introducing new sanitization checks restricting the authorative actions of the privileged roles of the system.

Files in Scope	Repository	Commit(s)
Address.sol (ADD)	olympus-contracts 💭	61f3d44487, 21fe403ed7, 182cfdb29f
BondTeller.sol (BTR)	olympus-contracts	61f3d44487, 21fe403ed7, 182cfdb29f

Files in Scope	Repository	Commit(s)
BondDepository.sol (BDY)	olympus-contracts 다	61f3d44487, 21fe403ed7, 182cfdb29f
Counters.sol (COU)	olympus-contracts	61f3d44487, 21fe403ed7, 182cfdb29f
ERC20.sol (ERC)	olympus-contracts 🏳	61f3d44487, 21fe403ed7, 182cfdb29f
ERC20Permit.sol (ERP)	olympus-contracts 🏳	61f3d44487, 21fe403ed7, 182cfdb29f
EnumerableSet.sol (EST)	olympus-contracts	61f3d44487, 21fe403ed7, 182cfdb29f
FullMath.sol (FMH)	olympus-contracts 🏳	61f3d44487, 21fe403ed7, 182cfdb29f
FixedPoint.sol (FPT)	olympus-contracts 다	61f3d44487, 21fe403ed7, 182cfdb29f
Guardable.sol (GUA)	olympus-contracts 다	61f3d44487, 21fe403ed7, 182cfdb29f
Governable.sol (GOV)	olympus-contracts 다	61f3d44487, 21fe403ed7, 182cfdb29f
GovernorAlpha.sol (GAA)	olympus-contracts 🖓	61f3d44487, 21fe403ed7, 182cfdb29f
GovernorOHMegaDelegate.sol (GOH)	olympus-contracts 🖓	61f3d44487, 21fe403ed7, 182cfdb29f

Files in Scope	Repository	Commit(s)
GovernorOHMegaDelegator.sol (GOM)	olympus-contracts 다	61f3d44487, 21fe403ed7, 182cfdb29f
GovernorOHMegaInterfaces.sol (GOI)	olympus-contracts 🏳	61f3d44487, 21fe403ed7, 182cfdb29f
ManagerOwnable.sol (MOE)	olympus-contracts 🏳	61f3d44487, 21fe403ed7, 182cfdb29f
Ownable.sol (OWN)	olympus-contracts 🏳	61f3d44487, 21fe403ed7, 182cfdb29f
OlympusERC20.sol (OER)	olympus-contracts 🏳	61f3d44487, 21fe403ed7, 182cfdb29f
OlympusTokenMigrator.sol (OTM)	olympus-contracts 🏳	61f3d44487, 21fe403ed7, 182cfdb29f
Staking.sol (STA)	olympus-contracts 🏳	61f3d44487, 21fe403ed7, 182cfdb29f
SafeMath.sol (SMH)	olympus-contracts 다	61f3d44487, 21fe403ed7, 182cfdb29f
SafeERC20.sol (SER)	olympus-contracts 다	61f3d44487, 21fe403ed7, 182cfdb29f
StakingDistributor.sol (SDR)	olympus-contracts 다	61f3d44487, 21fe403ed7, 182cfdb29f
StandardBondingCalculator.sol (SBC)	olympus-contracts 🛱	61f3d44487, 21fe403ed7, 182cfdb29f

Files in Scope	Repository	Commit(s)
Timelock.sol (TIM)	olympus-contracts 🦙	61f3d44487, 21fe403ed7, 182cfdb29f
Treasury.sol (TRE)	olympus-contracts	61f3d44487, 21fe403ed7, 182cfdb29f
VaultOwned.sol (VOD)	olympus-contracts	61f3d44487, 21fe403ed7, 182cfdb29f
gOHM.sol (OHM)	olympus-contracts	61f3d44487, 21fe403ed7, 182cfdb29f
sOlympusERC20.sol (OEC)	olympus-contracts	61f3d44487, 21fe403ed7, 182cfdb29f

During the audit, we filtered and validated a total of **10 findings utilizing static analysis** tools as well as identified a total of **73 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.

The list below covers each segment of the audit in depth and links to the respective chapter of the report:

- **Compilation**
- Static Analysis
- Manual Review
- / Code Style

# 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:



The hardhat tool automatically selects between Solidity versions 0.8.9, 0.7.5, and 0.5.16 based on the version specified within the hardhat.config.ts file as well as the pragma statement of the contract being currently compiled.

The project contains discrepancies with regards to the Solidity version used, however, they are located in external dependencies of the project and as such can be safely ignored.

The Olympus DAO team has locked the pragma statements to 0.7.5, 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 >

# **■** Omniscia Olympus DAO Audit

# **Static Analysis**

The execution of our static analysis toolkit identified **420 potential issues** within the codebase of which **407 were ruled out to be false positives** or negligible findings.

The remaining **13 issues** were validated and grouped and formalized into the **10 exhibits** that follow:

ID	Severity	Addressed	Title
BTR-01S	Major	Yes	Inexistent Initialization of Member
BTR-02S	Informational	Yes	Improper Inheritence
SDR-01S	Informational	Yes	Improper Inheritence
SDR-02S	Informational	Yes	Undocumented Value Literal
TRE-01S	Minor	Yes	Improper Usage of EIP-20 Transfer
TRE-02S	Informational	Yes	Literal Equality of bool Variables
VOD-01S	Minor	Yes	Inexistent Validation of Address Argument
VOD-02S	Informational	Yes	Inexistent Emission of Event
OEC-01S	Informational	Yes	Illegible Numeric Literal
OEC-02S	Informational	Yes	Improper Inheritence

Compilation

# **BondTeller Static Analysis Findings**

#### ON THIS PAGE

BTR-01S: Inexistent Initialization of Member

BTR-02S: Improper Inheritence

# **BTR-01S: Inexistent Initialization of Member**

Туре	Severity	Location
Logical Fault	Major ●	BondTeller.sol:L57

## **Description:**

The policy member of the contract is never initialized.

# **Example:**

```
contracts/BondTeller.sol

SOL

Copy

57  address public policy;

58

59  /* ======== CONSTRUCTOR ====== */
60
61  constructor(
62   address _depository,
63   address _staking,
64   address _treasury,
65   address _OHM,
66   address _SOHM,
67   address _gOHM
68 ) {
69   require(_depository != address(0));
```

```
depository = _depository;

require(_staking != address(0));

staking = IStaking(_staking);

require(_treasury != address(0));

treasury = ITreasury(_treasury);

require(_OHM != address(0));

OHM = IERC20(_OHM);

require(_sOHM != address(0));

sOHM = IERC20(_sOHM);

require(_gOHM != address(0));

gOHM = IgOHM(_gOHM);

IERC20(_OHM).approve(_staking, le27); // saves gas

leaddress(0)
```

### **Recommendation:**

We advise it to be initialized to ensure the **setFEReward** function can be invoked.

### **Alleviation:**

The policy member has now been removed from the contract, thereby nullifying this exhibit.

View Fix on GitHub

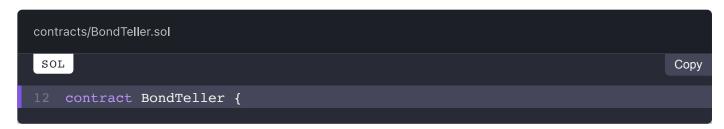
# **BTR-02S: Improper Inheritence**

Туре	Severity	Location
Code Style	Informational •	BondTeller.sol:L12

# **Description:**

The BondTeller contract complies with the ITeller interface of the codebase yet does not inherit it.

# **Example:**



### **Recommendation:**

We advise the contract to properly inherit it ensuring consistency and maintainability across the codebase.

### Alleviation:

The contract now properly inherits the ITeller interface.

View Fix on GitHub



NEXT > StakingDistributor.sol (SDR-S)



# **StakingDistributor Static Analysis Findings**

#### ON THIS PAGE

SDR-01S: Improper Inheritence

SDR-02S: Undocumented Value Literal

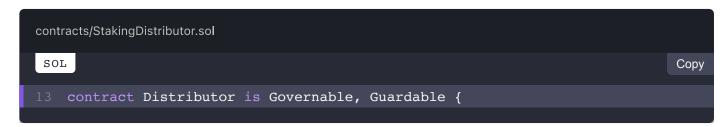
# **SDR-01S: Improper Inheritence**

Туре	Severity	Location
Code Style	Informational •	StakingDistributor.sol:L12

# **Description:**

The Distributor contract complies with the IDistributor interface of the codebase yet does not inherit it.

# <sup>ര</sup> Example:



### **Recommendation:**

We advise the contract to properly inherit it ensuring consistency and maintainability across the codebase.

#### Alleviation:

The contract now properly inherits the <code>IDistributor</code> interface.

View Fix on GitHub

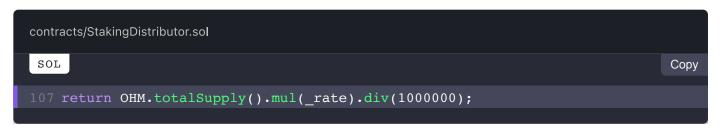
# **SDR-02S: Undocumented Value Literal**

Туре	Severity	Location
Code Style	Informational •	StakingDistributor.sol:L107

## **Description:**

The value literal 1000000 is meant to be used as the rate divisor for a particular reward distribution, however, it is undocumented and unclearly depicted.

### **Example:**



### **Recommendation:**

We advise the special underscore ( ) separator to be applied to it (i.e. 1000000 would become 1\_000\_000) and we advise the value to be set to a contract-level constant as it will be useful for other exhibits and general logic checks of the codebase.

### **Alleviation:**

The numeric literal was relocated to a contract-level <u>immutable</u> declaration thereby alleviating this exhibit.

View Fix on GitHub

PREV
BondTeller.sol (BTR-S)

# **Treasury Static Analysis Findings**

#### ON THIS PAGE

TRE-01S: Improper Usage of EIP-20 Transfer

TRE-02S: Literal Equality of bool Variables

# **TRE-01S: Improper Usage of EIP-20 Transfer**

Туре	Severity	Location
Standard Conformity	Minor •	Treasury.sol:L160

## **Description:**

The **EIP-20** standard denotes that callers MUST NOT assume that **false** is never returned in **transfer** invocations and should be able to gracefully handle the returned **bool** of the function invocation.

# **Example:**

```
contracts/Treasury.sol

SOL

Copy

160 IERC20(_token).transfer(msg.sender, _amount);
```

### **Recommendation:**

As certain tokens are not compliant with the standard, we advise the usage of a wrapper library such as SafeERC20 of OpenZeppelin that opportunistically evaluates the yielded bool if it exists.

#### **Alleviation:**

The linked EIP-20 transfer call is now properly wrapped in its safe-prefixed equivalent by OpenZeppelin's safeERC20 library.

View Fix on GitHub

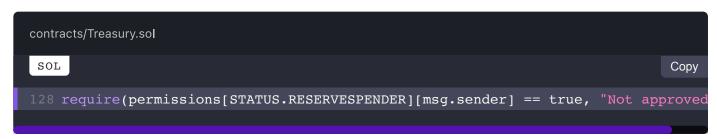
# TRE-02S: Literal Equality of **bool** Variables

Туре	Severity	Location
Code Style	Informational •	Treasury.sol:L128

## **Description:**

The linked statement performs a direct comparison between a bool variable and a bool literal.

# **Example:**



### **Recommendation:**

We advise the bool variable to be utilized directly either in its normal or negated (!) form, depending on the bool literal it was being compared to.

### **Alleviation:**

The bool variable is now utilized directly in the require check.

View Fix on GitHub

StakingDistributor.sol (SDR-S)

VaultOwned.sol (VOD-S)

# **VaultOwned Static Analysis Findings**

#### ON THIS PAGE

VOD-01S: Inexistent Validation of Address Argument

VOD-02S: Inexistent Emission of Event

# **VOD-01S: Inexistent Validation of Address Argument**

Туре	Severity	Location
Input Sanitization	Minor •	VaultOwned.sol:L10-L14

## **Description:**

The linked function contains an address argument that is not properly sanitized against the zero-address.

# <sup>ര</sup> Example:

```
contracts/types/VaultOwned.sol

SOL

10 function setVault( address vault_ ) external onlyOwner() returns ( bool ) {
11   _vault = vault_;
12
13   return true;
14 }
```

### **Recommendation:**

We advise it to be sanitized so to avoid potential misconfigurations of the contract.

### Alleviation:

The Olympus DAO team considered this exhibit but opted not to apply any remediation for it.

View Fix on GitHub

# **VOD-02S: Inexistent Emission of Event**

Туре	Severity	Location
Input Sanitization	Informational •	VaultOwned.sol:L10-L14

## **Description:**

The linked function adjusts a sensitive contract variable without emitting a corresponding event.

### **Example:**

### **Recommendation:**

We advise an event to be coded for the action and emitted whenever it is executed to ensure off-chain observers of the contracts can properly sync their data points.

#### Alleviation:

The Olympus DAO team considered this exhibit but opted not to apply any remediation for it.

View Fix on GitHub

sOlympusERC20.sol (OEC-S)

# sOlympusERC20 Static Analysis Findings

#### ON THIS PAGE

OEC-01S: Illegible Numeric Literal

OEC-02S: Improper Inheritence

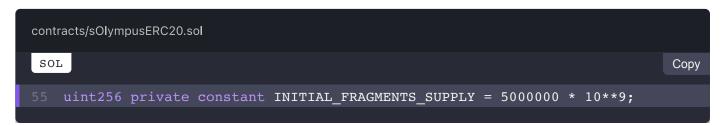
# **OEC-01S: Illegible Numeric Literal**

Туре	Severity	Location
Code Style	Informational •	sOlympusERC20.sol:L55

## **Description:**

The linked variable contains a numeric literal with too many digits and no separator.

# **Example:**



### **Recommendation:**

We advise the special numeric separator ( ) to be used to discern per thousand units (i.e. 10000 becomes 10 000), increasing the legibility of the codebase.

### Alleviation:

The underscore ( numeric separator was properly introduced to the linked variable.

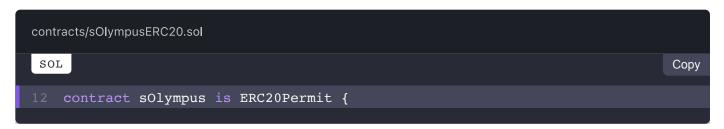
# **OEC-02S: Improper Inheritence**

Туре	Severity	Location
Code Style	Informational •	sOlympusERC20.sol:L12

## **Description:**

The solympus contract complies with the Isohm interface of the codebase yet does not inherit it.

# **Example:**



### **Recommendation:**

We advise the contract to properly inherit it ensuring consistency and maintainability across the codebase.

### Alleviation:

The now properly inherits the ISOHM interface.

View Fix on GitHub

VaultOwned.sol (VOD-S)

NEXT > BondDepository.sol (BDY-M)

**Manual Review** 

# **Manual Review**

A **thorough line-by-line review** was conducted on the codebase to identify potential malfunctions and vulnerabilities in the version 2 iteration of the Olympus DAO protocol.

As the project at hand implements a complex system architecture of a three token system and a bond pricing mechanism, 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 certain misconceptions** within the system which could have had **severe ramifications** to its overall operation when exploited under the right circumstances, however, they were conveyed ahead of time to the Olympus DAO 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 mathematical operations surrounding the pricing of debt ratios utilizing the undocumented decode112with18 function.

A total of **73 findings** were identified over the course of the manual review of which **39 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
BDY-01M	Major	No	Improper Bond Price Assumption
BDY-02M	Medium	No	Inexistent Validation of Terms

# **BondDepository Manual Review Findings**

#### ON THIS PAGE

BDY-01M: Improper Bond Price Assumption

BDY-02M: Inexistent Validation of Terms

BDY-03M: Inexplicable Optional Value of Decay

# **BDY-01M: Improper Bond Price Assumption**

Туре	Severity	Location
Logical Fault	Major •	BondDepository.sol:L328-L341

# **Description:**

The bondPrice adjusts the on-chain minimum price of a bond to of the price of the bond is exceeded, however, the debtratio variable relies on the total supply of OHM which is fully manipulateable and thus can cause the price of a bond to increase above the minimum temporarily by taking advantage of the debtratio reliance on OHM.totalSupply.

# **Example:**

```
contracts/BondDepository.sol

SOL

Copy

328 /**
329 * @notice calculate current bond price and remove floor if above
330 * @param _BID uint
331 * @return price_ uint
332 */
333 function _bondPrice(uint256 _BID) internal returns (uint256 price_) {
334    Bond memory info = bonds[_BID];
335    price_ = info.terms.controlVariable.mul(debtRatio(_BID)).add(1000000000).di
```

```
if (price_ < info.terms.minimumPrice) {</pre>
       price = info.terms.minimumPrice;
      } else if (info.terms.minimumPrice != 0) {
        bonds[ BID].terms.minimumPrice = 0;
341 }
348 function bondPriceInUSD(uint256 _BID) public view returns (uint256 price_) {
     Bond memory bond = bonds[_BID];
if (address(bond.calculator) != address(0)) {
       price = bondPrice(_BID).mul(bond.calculator.markdown(address(bond.princi
     } else {
       price = bondPrice(_BID).mul(10**IERC20Metadata(address(bond.principal))
355 }
364 function debtRatio(uint256 _BID) public view returns (uint256 debtRatio ) {
     debtRatio = FixedPoint.fraction(currentDebt( BID).mul(1e9), OHM.totalSuppl
366 }
```

#### **Recommendation:**

We advise the function to not adjust the minimum price as it can lead to the pricing of a bond becoming less-than-minimum under the right circumstances.

#### Alleviation:

The Olympus DAO team considered this exhibit but decided to retain the current behaviour of the code in place.

# **BDY-02M: Inexistent Validation of Terms**

Туре	Severity	Location
Input Sanitization	Medium •	BondDepository.sol:L97-L129

## **Description:**

All arguments of the setTerms function are not sanitized and as such can be arbitrarily set to values that may be illogical or result in exploits, such as a conclusion that is beyond the expiration of the bond and other similar issues.

### **Example:**

```
contracts/BondDepository.sol
SOL
                                                                          Copy
110 function setTerms(
111 uint256 id,
112 uint256 controlVariable,
113 bool fixedTerm,
114 uint256 vestingTerm,
115 uint256 expiration,
116 uint256 conclusion,
117 uint256 minimumPrice,
118 uint256 maxPayout,
```

### **Recommendation:**

We strongly recommend some form of input sanitization to be enforced on the bond terms as in the current state the guardian has significant control over the protocol's normal operation.

### Alleviation:

The Olympus DAO team considered this exhibit but decided to retain the current behaviour of the code in place.

# **BDY-03M: Inexplicable Optional Value of Decay**

Туре	Severity	Location
Logical Fault	Medium •	BondDepository.sol:L390-L402

## **Description:**

The debtDecay function relies on the vestingTerm variable of the Term struct which is meant to be an optional variable based on the documentation of the struct.

### **Example:**

### **Recommendation:**

We advise its purpose to be better defined in the struct declaration as bonds without a fixed term are possible, causing decay to misbehave.

#### Alleviation:

The Olympus DAO team considered this exhibit but decided to retain the current behaviour of

the code in place.

PREV
sOlympusERC20.sol (OEC-S)

BondTeller.sol (BTR-M)

# **BondTeller Manual Review Findings**

#### ON THIS PAGE

BTR-01M: Confusion of Value Denominations

BTR-02M: Artificial Inflation Mechanism

BTR-03M: Inexistent Redemption of FEO Fees

BTR-04M: Inexistent Validation of Non-Zero Redemption

# **BTR-01M: Confusion of Value Denominations**

Туре	Severity	Location
Logical Fault	Major •	BondTeller.sol:L232-L245

# **Description:**

The percentvestedFor function assumes the bond.vested value to be a timestamp yet it represents a block.

# **Example:**

```
contracts/BondTeller.sol

SOL

Copy

232 /**
233 * @notice calculate how far into vesting a depositor is
234 * @param _bonder address
235 * @param _index uint
236 * @return percentVested_ uint
237 */
238 function percentVestedFor(address _bonder, uint256 _index) public view return
239    Bond memory bond = bonderInfo[_bonder][_index];
240
```

```
uint256 timeSince = block.timestamp.sub(bond.created);

uint256 term = bond.vested.sub(bond.created);

and percentVested = timeSince.mul(1e9).div(term);

are percentVested = timeSince.mul(1e9).div(term).div(term).div(term).div(term).div(term).div(term).div(term).div(term).div(term).div(term).div(term).div(term).div(term).div(term).div(term).div(term).div(term).div(term).div(term).div(term).div(term).div(te
```

### **Recommendation:**

We strongly recommend the function to be adjusted as it should be inoperable in its current state.

### **Alleviation:**

After discussion with the Olympus DAO team, both units are meant to represent a timestamp and as such this exhibit can be considered null.

View Fix on GitHub

### BTR-02M: Artificial Inflation Mechanism

Туре	Severity	Location
Logical Fault	Medium •	BondTeller.sol:L104, L110

### **Description:**

The newBond function is meant to award bond creators with a fereward that is added to the original payout of a bond, however, this action introduces a secondary level of inflation that can also be redirected to the user themselves and could result in a significant issue if fereward is close to the minimum payout as users are incentivized to break their bonds into multiple smaller ones and set the feo as themselves.

```
OHM.approve(address(staking), _payout); // approve staking payout

staking.stake(_payout, address(this), true, true);

FERS[_feo] = FERS[_feo].add(feReward); // FE operator takes fee

index_ = bonderInfo[_bonder].length;

// store bond & stake payout

bonderInfo[_bonder].push(Bond({principal: _principal, principalPaid: _principal})

bonderInfo[_bonder].push(Bond({principal: _principal, principalPaid: _principal})
```

We strongly recommend this aspect of the protocol to be revised. Some potential solutions would be to retain an address of whitelisted front end operators that rewards can be re-directed to and having the fereward be based on a percentage rather than a static value.

### Alleviation:

The Olympus DAO team considered this exhibit but opted not to apply a remediation for it as they consider the current mechanism sufficiently secure.

# **BTR-03M: Inexistent Redemption of FEO Fees**

Туре	Severity	Location
Logical Fault	Minor •	BondTeller.sol:L110

### **Description:**

The front-end operator (FEO) fees are stored within the contract and are minted, however, there is no way to claim them by the FEOs.

```
contracts/BondTeller.sol
SOL
   function newBond(
       address bonder,
       address principal,
       uint256 principalPaid,
      uint256 payout,
       uint256 _expires,
       address feo
103 ) external onlyDepository returns (uint256 index ) {
       treasury.mint(address(this), payout.add(feReward));
       OHM.approve(address(staking), payout); // approve staking payout
       staking.stake( payout, address(this), true, true);
```

```
110    FERs[_feo] = FERs[_feo].add(feReward); // FE operator takes fee
111
112    index_ = bonderInfo[_bonder].length;
113
114    // store bond & stake payout
115    bonderInfo[_bonder].push(Bond({principal: _principal, principalPaid: _pri
116 }
```

We advise some form of pull-pattern to be applied where FEOs are able to invoke a function to retrieve all fees they have accumulated.

### **Alleviation:**

A fee redemption mechanism was introduced in the codebase in the form of the getReward function that transfers the necessary OHM outward to the user.

# **BTR-04M: Inexistent Validation of Non-Zero Redemption**

Туре	Severity	Location
Input Sanitization	Minor •	BondTeller.sol:L148

### **Description:**

The redeem function should validate that a non-zero redemption is being performed.

```
contracts/BondTeller.sol
SOL
                                                                               Copy
136 function redeem(address bonder, uint256[] memory indexes) public returns (u
       uint256 dues;
        for (uint256 i = 0; i < indexes.length; i++) {</pre>
            Bond memory info = bonderInfo[ bonder][ indexes[i]];
            if (pendingFor( bonder, indexes[i]) != 0) {
                bonderInfo[ bonder][ indexes[i]].redeemed = block.timestamp; // m
                dues = dues.add(info.payout);
            }
        }
        dues = gOHM.balanceFrom(dues);
        emit Redeemed( bonder, dues);
        pay( bonder, dues);
       return dues;
153 }
```

We advise this to be done so by ensuring that dues is non-zero beyond conversion.

### **Alleviation:**

The Olympus DAO team considered this exhibit but decided to retain the current behaviour of the code in place.





# **ERC20 Manual Review Findings**

ON THIS PAGE

ERC-01M: Non-Standard Mint Implementation

# **ERC-01M: Non-Standard Mint Implementation**

Туре	Severity	Location
Standard Conformity	Minor •	ERC20.sol:L95-L101

### **Description:**

The \_mint function of the ERC20 implementation invokes the \_beforeTokenTransfer hook and the Transfer event with the from argument being the address(this), an adaptation that will cause off-chain explorers to misbehave.

### **Example:**

```
contracts/types/ERC20.sol

SOL

SOL

95 function _mint(address account_, uint256 ammount_) internal virtual {
96    require(account_ != address(0), "ERC20: mint to the zero address");
97    _beforeTokenTransfer(address( this ), account_, ammount_);
98    _totalSupply = _totalSupply.add(ammount_);
99    _balances[account_] = _balances[account_].add(ammount_);
100    emit Transfer(address( this ), account_, ammount_);
101 }
```

### **Recommendation:**

We advise both instances to be replaced by the canonical address(0) to enable proper

interfacing of block explorers.

### **Alleviation:**

<

The hook and event instances now properly use the zero address instead of the self address to indicate the origin of the "transfer" operation, thereby alleviating this exhibit.

PREV
BondTeller.sol (BTR-M)

NEXT
ERC20Permit.sol (ERP-M)

# **ERC20Permit Manual Review Findings**

#### ON THIS PAGE

ERP-01M: Insecure Elliptic Curve Recovery Mechanism

ERP-02M: Cross-Chain Signature Replay Attack Susceptibility

# **ERP-01M: Insecure Elliptic Curve Recovery Mechanism**

Туре	Severity	Location
Language Specific	Medium •	ERC20Permit.sol:L55

### **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).

```
contracts/types/ERC20Permit.sol

SOL

Copy

35 /**
36 * @dev See {IERC2612Permit-permit}.

37 *
38 */
39 function permit(
40 address owner,
41 address spender,
42 uint256 amount,
43 uint256 deadline,
```

```
bytes32 r,
bytes32 s

7 ) public virtual override {
    require(block.timestamp <= deadline, "Permit: expired deadline");

bytes32 hashStruct =
    keccak256(abi.encode(PERMIT_TYPEHASH, owner, spender, amount, _nonces

bytes32 _hash = keccak256(abi.encodePacked(uint16(0x1901), DOMAIN_SEPARAT

bytes32 _hash = keccak256(abi.encodePacked(uint16(0x1901), DOMAIN_SEPARAT

address signer = ecrecover(_hash, v, r, s);

require(signer != address(0) && signer == owner, "ZeroSwapPermit: Invalid

nonces[owner].increment();
approve(owner, spender, amount);
}</pre>
```

### Alleviation:

The code was adjusted to instead utilize the ECDSA library from OpenZeppelin directly that imposes the necessary sanitizations on the v and s arguments of the ecrecover function.

# **ERP-02M: Cross-Chain Signature Replay Attack Susceptibility**

Туре	Severity	Location
Language Specific	Minor •	ERC20Permit.sol:L19-L33

### **Description:**

The DOMAIN\_SEPARATOR used in conjunction with the **EIP-712** standard is calculated once during the **constructor** of the contract. As this calculation involves the execution context's **chainid**, blockchain forks will allow signatures to be replayed across chains as the **chainid** is consequently not validated.

### **Example:**

```
contracts/types/ERC20Permit.sol
SOL
                                                                                 Copy
    constructor() {
        uint256 chainID;
        assembly {
            chainID := chainid()
        }
        DOMAIN SEPARATOR = keccak256(abi.encode(
            keccak256("EIP712Domain(string name, string version, uint256 chainId, ad
            keccak256(bytes(name())),
            keccak256(bytes("1")), // Version
            chainID,
            address(this)
        ));
33 }
```

### **Recommendation:**

We advise a caching mechanism to be imposed here instead whereby the <code>DOMAIN\_SEPARATOR</code> is stored to an <code>immutable</code> contract-level variable and is utilized only when the cached <code>chainid</code>

(also stored in an immutable variable) matches the current execution context's chainid. A reference implementation of this paradigm can be observed at the draft-EIP712.sol implementation of OpenZeppelin.

### **Alleviation:**

The code took inspiration from the draft-EIP712.sol implementation of OpenZeppelin and now uses a caching system that dynamically calculates the separator as needed depending on the evaluation of the current chainID, thereby alleviating this exhibit.

View Fix on GitHub

PREV
ERC20.sol (ERC-M)

NEXT FixedPoint.sol (FPT-M)

# **FixedPoint Manual Review Findings**

ON THIS PAGE

FPT-01M: Potentially Invalid Implementation

# **FPT-01M: Potentially Invalid Implementation**

Туре	Severity	Location
Logical Fault	Major ●	FixedPoint.sol:L110-L113

### **Description:**

The decode112with18 function is non-standard, utilizes a value literal and cannot be validated as to its purpose.

### <sup>ര</sup> Example:

```
contracts/libraries/FixedPoint.sol

SOL

Copy

110 function decode112with18(uq112x112 memory self) internal pure returns (uint)
111

112 return uint(self._x) / 5192296858534827;
113 }
```

### **Recommendation:**

We advise it to be extensively documented as in its current state it is ambiguous. This finding will be adjusted accordingly when the code segment has been properly documented.

### Alleviation:

The Olympus DAO team has not provided a response for this exhibit yet.

PREV
ERC20Permit.sol (ERP-M)

Sovernable.sol (GOV-M)

# **Governable Manual Review Findings**

#### ON THIS PAGE

GOV-01M: Improper Governor Renouncation

GOV-02M: Incorrect Event Emitted

GOV-03M: Potentially Restrictive Functionality

# **GOV-01M: Improper Governor Renouncation**

Туре	Severity	Location
Logical Fault	Medium •	Governable.sol:L31-L34

### **Description:**

The renounceGovernor function deletes the current governor in place, however, the newGovernor remains especially so when the pullGovernor function is invoked once as it does not delete the previous entry. This would permit governorship to be re-instated even after it has been renounced on a particular contract.

## **Example:**

```
contracts/types/Governable.sol

SOL

Copy

32 function renounceGovernor() public virtual override onlyGovernor() {
        emit GovernorPushed( _governor, address(0) );
        _governor = address(0);
    }

35 }
```

### **Recommendation:**

We strongly recommend the <u>newGovernor</u> to also be deleted when <u>renounceGovernor</u> is invoked to prevent improper state transitions.

# **Alleviation:**

The new governor is properly deleted when governorship is renounced.

### **GOV-02M: Incorrect Event Emitted**

Туре	Severity	Location
Logical Fault	Minor •	Governable.sol:L33

### **Description:**

The renounceGovernor function emits the GovernorPushed event when the one in place is renounced, however, it should emit the GovernorPulled event instead.

```
contracts/types/Governable.sol
SOL
                                                                            Copy
16 constructor () {
     governor = msg.sender;
       emit GovernorPulled( address(0), governor );
   function governor() public view override returns (address) {
       return governor;
   modifier onlyGovernor() {
       require( governor == msg.sender, "Governable: caller is not the governor
       _;
   function renounceGovernor() public virtual override onlyGovernor() {
       emit GovernorPushed( governor, address(0));
       governor = address(0);
   function pushGovernor( address newGovernor ) public virtual override onlyGov
       require( newGovernor_ != address(0), "Governable: new governor is the zer
       emit GovernorPushed( governor, newGovernor );
```

We advise it to do so as the latter event is utilized in functions like pullGovernor and the constructor.

### **Alleviation:**

The GovernorPulled event is now properly emitted within the renounceGovernor function.

# **GOV-03M: Potentially Restrictive Functionality**

Туре	Severity	Location
Logical Fault	Minor •	Governable.sol:L38

### **Description:**

The pushGovernor function validates that the newGovernor being set is not the zero address, however, in doing so it will prevent a pending governor to be overwritten with a zero entry.

### **Example:**

```
contracts/types/Governable.sol

SOL

37 function pushGovernor( address newGovernor_ ) public virtual override onlyGov
38    require( newGovernor_ != address(0), "Governable: new governor is the zer
39    emit GovernorPushed( _governor, newGovernor_ );
40    __newGovernor = newGovernor_;
41 }
```

### **Recommendation:**

We advise this check to either be omitted entirely or only be evaluated when <a href="mewGovernor">newGovernor</a> is itself equal to the zero-address as otherwise, an incorrectly set <a href="mewGovernor">newGovernor</a> will not be able to be deleted.

### Alleviation:

The restrictive require check is now omitted from the codebase.



NEXT >

GovernorAlpha.sol (GAA-M)

# **GovernorAlpha Manual Review Findings**

#### ON THIS PAGE

GAA-01M: Improper Percentage Documented

# **GAA-01M: Improper Percentage Documented**

Туре	Severity	Location
Logical Fault	Informational •	GovernorAlpha.sol:L14

### **Description:**

The proposalThresholdPercent is meant to showcase the percentage of voting power required to make a proposal, however, it is incorrectly documented as 1.00% when in reality it is 10% when the divisor of the getVotesFromPercentOfsOHMSupply function is taken into account.

### **Example:**

```
contracts/governance/GovernorAlpha.sol

SOL

Copy

12 /// @notice The maximum setable proposal threshold percent
13 /// @notice change from original contract

14 function proposalThresholdPercent() public pure returns (uint) { return 10000
```

### **Recommendation:**

We advise the documentation or value itself to be properly remediated depending on the canonical value of the proposal threshold.

### **Alleviation:**

The value was corrected to the 1.00% indicated by its corresponding comments.

View Fix on GitHub

Governable.sol (GOV-M)

NEXT

Guardable.sol (GUA-M)

# **Guardable Manual Review Findings**

#### ON THIS PAGE

**GUA-01M: Improper Guardian Renouncation** 

GUA-02M: Incorrect Event Emitted

GUA-03M: Potentially Restrictive Functionality

# **GUA-01M: Improper Guardian Renouncation**

Туре	Severity	Location
Logical Fault	Medium •	Guardable.sol:L31-L34

### **Description:**

The renounceGuardian function deletes the current guardian in place, however, the newGuardian remains especially so when the pullGuardian function is invoked once as it does not delete the previous entry. This would permit guardianship to be re-instated even after it has been renounced on a particular contract.

## **Example:**

```
contracts/types/Guardable.sol

SOL

Copy

31 function renounceGuardian() public virtual override onlyGuardian() {
    emit GuardianPushed( _guardian, address(0));

    _guardian = address(0);

34 }
```

### **Recommendation:**

We strongly recommend the <a>newGuardian</a> to also be deleted when <a>renounceGuardian</a> is invoked to prevent improper state transitions.

# **Alleviation:**

The new guardian is properly deleted when guardianship is renounced.

### **GUA-02M: Incorrect Event Emitted**

Туре	Severity	Location
Logical Fault	Minor •	Guardable.sol:L32

### **Description:**

The renounceGuardian function emits the GuardianPushed event when the one in place is renounced, however, it should emit the GuardianPulled event instead.

```
contracts/types/Guardable.sol
SOL
                                                                             Copy
   constructor () {
           quardian = msq.sender;
            emit GuardianPulled( address(0), _guardian );
        function quardian() public view override returns (address) {
            return guardian;
        }
       modifier onlyGuardian() {
            require( guardian == msg.sender, "Guardable: caller is not the guard
        function renounceGuardian() public virtual override onlyGuardian() {
            emit GuardianPushed( guardian, address(0));
           guardian = address(0);
        function pushGuardian( address newGuardian ) public virtual override onl
            require( newGuardian != address(0), "Guardable: new guardian is the
            emit GuardianPushed( _guardian, newGuardian_ );
            newGuardian = newGuardian ;
```

```
40  }
41
42  function pullGuardian() public virtual override {
43     require( msg.sender == _newGuardian, "Guardable: must be new guardian
44     emit GuardianPulled( _guardian, _newGuardian );
45     __guardian = _newGuardian;
46  }
47 }
```

We advise it to do so as the latter event is utilized in functions like pullGuardian and the constructor.

### **Alleviation:**

The GuardianPulled event is now properly emitted within the renounceGuardian function.

# **GUA-03M: Potentially Restrictive Functionality**

Туре	Severity	Location
Logical Fault	Minor ●	Guardable.sol:L37

### **Description:**

The pushGuardian function validates that the newGuardian being set is not the zero address, however, in doing so it will prevent a pending guardian to be overwritten with a zero entry.

### **Example:**

```
contracts/types/Guardable.sol

SOL

SOL

Copy

36 function pushGuardian( address newGuardian_ ) public virtual override onlyGua

37     require( newGuardian_ != address(0), "Guardable: new guardian is the zero

38     emit GuardianPushed( _guardian, newGuardian_ );

39     _newGuardian = newGuardian_;

40 }
```

### **Recommendation:**

We advise this check to either be omitted entirely or only be evaluated when <a href="mewGuardian">newGuardian</a> is itself equal to the zero-address as otherwise, an incorrectly set <a href="mewGuardian">newGuardian</a> will not be able to be deleted.

### Alleviation:

The restrictive require check is now omitted from the codebase.

NEXT >

OlympusTokenMigrator.sol (OTM-M)

# OlympusTokenMigrator Manual Review Findings

#### **ON THIS PAGE**

OTM-01M: Improper Integration w/ Uniswap V2

OTM-02M: Improper Evaluation of Token Balance

OTM-03M: Ungraceful Mint Handling

OTM-04M: Potential of Repeat Invocation

# OTM-01M: Improper Integration w/ Uniswap V2

Туре	Severity	Location
Logical Fault	Major •	OlympusTokenMigrator.sol:L266-L274, L281-L290

# **Description:**

The way the migration of the token works can cause the migration to either completely halt or cause the liquidity position to significantly diminish in value should the governor address be a contract that can be actuated from anyone, such as a Timelock forked from Compound.

```
contracts/migration/OlympusTokenMigrator.sol

SOL

Copy

249 /**

250 * @notice Migrate LP and pair with new OHM

251 */

252 function migrateLP(

253 address pair,

254 bool sushi,
```

```
address token
256 ) external onlyGovernor {
        uint256 oldLPAmount = IERC20(pair).balanceOf(address(oldTreasury));
        oldTreasury.manage(pair, oldLPAmount);
        IUniswapV2Router router = sushiRouter;
        if (!sushi) {
            router = uniRouter;
        }
        IERC20(pair).approve(address(router), oldLPAmount);
        (uint256 amountA, uint256 amountB) = router.removeLiquidity(
            token,
            address(oldOHM),
            oldLPAmount,
            0,
            0,
            address(this),
            1000000000000
        );
        newTreasury.mint(address(this), amountB);
        IERC20(token).approve(address(router), amountA);
        newOHM.approve(address(router), amountB);
        router.addLiquidity(
            token,
            address(newOHM),
            amountA,
            amountB,
            amountA,
            amountB,
            address(newTreasury),
            100000000000
        );
291 }
```

We strongly recommend the migration procedure to be revised. In the current state, it specifies the expected output amounts as  $\boxed{0}$  which can cause an arbitreur to significantly skew the pair, diminish the LP position one-sidedly (i.e. towards OHM) and cause the liquidity removal to be in

the native token only. This can cause the protocol to crash due to the artificial inflation of OHM's price which can be performed with the help of flash loans if for example the governor is a <code>Timelock</code> implementation relying on a <code>GovernorAlpha</code> to actuate it. Additionally, the liquidity provision is also performed incorrectly as it specifies the amounts that should at minimum be set within the pair to be equal to the amounts provided. This case is only true when the pair has not been created before. Should <code>newOHM</code> units circulate in the market before this point, it would be possible for someone to race the transaction, create the pair with miniscule amounts and thus cause the migration to be impossible. As a last note, the current <code>block.timestamp</code> can and should be passed in as the expiry argument instead of the literal <code>10000000000000</code> which is meaningless.

### **Alleviation:**

Minimum arguments were properly added to the migrateLP function and the numeric literal was substituted for the current block.timestamp, thereby alleviating this exhibit in full.

# **OTM-02M: Improper Evaluation of Token Balance**

Туре	Severity	Location
Logical Fault	Medium •	OlympusTokenMigrator.sol:L364

### **Description:**

```
The _migrateToken function is utilizing the tokenValue yielded by the newTreasury implementation yet is comparing it with the excessReserves of the oldTreasury.
```

```
contracts/migration/OlympusTokenMigrator.sol
SOL
                                                                               Copy
360 function migrateToken(address token, bool deposit) internal {
        uint256 balance = IERC20(token).balanceOf(address(oldTreasury));
        uint256 excessReserves = oldTreasury.excessReserves();
        uint256 tokenValue = newTreasury.tokenValue(token, balance);
        if (tokenValue > excessReserves) {
            tokenValue = excessReserves;
            balance = excessReserves * 10**9;
        oldTreasury.manage(token, balance);
        if (deposit) {
            IERC20(token).safeApprove(address(newTreasury), balance);
            newTreasury.deposit(balance, token, tokenValue);
        } else {
            IERC20(token).transfer(address(newTreasury), balance);
379 }
```

We strongly recommend the valueOf implementation of the legacy treasury to be utilized instead as it is currently incorrectly evaluating the maximum value that can be retrieved from oldTreasury.

### **Alleviation:**

The code now properly uttilizes the legacy value of function to properly identify how many funds can be managed from the legacy treasury.

# **OTM-03M: Ungraceful Mint Handling**

Туре	Severity	Location
Logical Fault	Medium •	OlympusTokenMigrator.sol:L217, L219

### **Description:**

The contract contains logic blocks that indicate the possibility of oldohm being minted beyond migration is realistic and simply prohibits swaps for it, however, a mint event and corresponding transfer to the migrator contract is unaccounted for and can cause defund to be inoperable.

### **Example:**

```
contracts/migration/OlympusTokenMigrator.sol
SOL
                                                                               Copy
209 function defund(address reserve) external onlyGovernor {
        require(ohmMigrated && timelockEnd < block.number && timelockEnd != 0);
        oldwsOHM.unwrap(oldwsOHM.balanceOf(address(this)));
        uint256 amountToUnstake = oldsOHM.balanceOf(address(this));
        oldsOHM.approve(address(oldStaking), amountToUnstake);
        oldStaking.unstake(amountToUnstake, false);
        uint256 balance = oldOHM.balanceOf(address(this));
        oldSupply = oldSupply.sub(balance);
        uint256 amountToWithdraw = balance.mul(1e9);
        oldOHM.approve(address(oldTreasury), amountToWithdraw);
        oldTreasury.withdraw(amountToWithdraw, reserve);
        IERC20(reserve).safeTransfer(address(newTreasury), IERC20(reserve).balance
        emit Defunded(balance);
227 }
```

S ....

We strongly recommend the code to gracefully handle such an event by containing an if block that nullifies oldsupply if the balance exceeds it.

### **Alleviation:**

The code now properly handles an instance of the balance exceeding the oldsupply in accordance to our recommendation.

# **OTM-04M: Potential of Repeat Invocation**

Туре	Severity	Location
Logical Fault	Minor •	OlympusTokenMigrator.sol:L229-L234

### **Description:**

The startTimelock function can be invoked multiple times.

### **Example:**

```
contracts/migration/OlympusTokenMigrator.sol

SOL

229 // start timelock to send backing to new treasury
230 function startTimelock() external onlyGovernor {
231    timelockEnd = block.number.add(timelockLength);
232
233    emit TimelockStarted(block.number, timelockEnd);
234 }
```

### **Recommendation:**

It should only be invoke-able once and as such should introduce a require check that ensures timelockEnd is equal to 0 at the beginning.

### **Alleviation:**

The require check we recommended was properly added to the codebase.

View Fix on GitHub

C PREV Guardable.sol (GUA-M)

OWITADIC.301 (OWIT-IVI)

# **Ownable Manual Review Findings**

### ON THIS PAGE

OWN-01M: Improper Ownership Renouncation

OWN-02M: Incorrect Events Emitted

OWN-03M: Potentially Restrictive Functionality

### **OWN-01M: Improper Ownership Renouncation**

Туре	Severity	Location
Logical Fault	Medium •	Ownable.sol:L31-L34

### **Description:**

The renounceManagement function deletes the current owner in place, however, the newOwner remains especially so when the pullManagement function is invoked once as it does not delete the previous entry. This would permit ownership to be re-instated even after it has been renounced on a particular contract.

### **Example:**

```
contracts/types/Ownable.sol

SOL

28 function renounceManagement() public virtual override onlyOwner() {
29    emit OwnershipPushed( _owner, address(0) );
30    _owner = address(0);
31 }
```

### **Recommendation:**

We strongly recommend the \_newOwner to also be deleted when renounceManagement is invoked to prevent improper state transitions.

### **Alleviation:**

The new owner is properly deleted when ownership is renounced.

### **OWN-02M: Incorrect Events Emitted**

Туре	Severity	Location
Logical Fault	Minor •	Ownable.sol:L16

### **Description:**

```
The constructor and renounceManagement function of the Ownable contract incorrectly emit the OwnershipPushed event instead of the OwnershipPulled One.
```

```
contracts/types/Ownable.sol
SOL
                                                                             Copy
   event OwnershipPushed(address indexed previousOwner, address indexed newOwner
   event OwnershipPulled(address indexed previousOwner, address indexed newOwner
   constructor () {
       owner = msg.sender;
       emit OwnershipPushed( address(0), owner );
   function owner() public view override returns (address) {
       return owner;
   modifier onlyOwner() {
        require( owner == msg.sender, "Ownable: caller is not the owner" );
   }
   function renounceManagement() public virtual override onlyOwner() {
        emit OwnershipPushed( owner, address(0));
       owner = address(0);
   function pushManagement( address newOwner_ ) public virtual override onlyOwne
       require( newOwner != address(0), "Ownable: new owner is the zero address
```

```
as a semit OwnershipPushed( _owner, newOwner_ );
as a _newOwner = newOwner_;
as a semit OwnershipPushed( _owner, newOwner owner);
as a semit OwnershipPulled( _owner, _newOwner owner);
as a semit OwnershipPulled( _owner, _newOwner);
as a semit Owne
```

We advise the latter to be utilized as it is the canonical one when an \_owner entry is written in pullManagement.

### **Alleviation:**

The OwnershipPulled event is now properly emitted within the renounceOwnership function.

### **OWN-03M: Potentially Restrictive Functionality**

Туре	Severity	Location
Logical Fault	Minor •	Ownable.sol:L34

### **Description:**

The pushManagement function validates that the newOwner\_being set is not the zero address, however, in doing so it will prevent a pending owner to be overwritten with a zero entry.

### **Example:**

```
contracts/types/Ownable.sol

SOL

Copy

33 function pushManagement( address newOwner_ ) public virtual override onlyOwned

34     require( newOwner_ != address(0), "Ownable: new owner is the zero address

35     emit OwnershipPushed( _owner, newOwner_ );

36     _newOwner = newOwner_;

37 }
```

### **Recommendation:**

We advise this check to either be omitted entirely or only be evaluated when <a href="mewowner">newowner</a> is itself equal to the zero-address as otherwise, an incorrectly set <a href="mewowner">newowner</a> will not be able to be deleted.

### Alleviation:

The restrictive require check is now omitted from the codebase.

NEXT >

**StakingDistributor.sol (SDR-M)** 



# Staking Distributor Manual Review **Findings**

### ON THIS PAGE

SDR-01M: Improper Accumulation of Rewards

SDR-02M: Ungraceful Handling of High Adjustment Rates

SDR-03M: Inexistent Validation of Entry Validity

SDR-04M: Inexistent Validation of Reward Rate

### **SDR-01M: Improper Accumulation of Rewards**

Туре	Severity	Location
Logical Fault	Medium •	StakingDistributor.sol:L110-L123

### **Description:**

The nextrewardfor function does not properly accumulate rewards if multiple ones are specified for a particular recipient.

```
contracts/StakingDistributor.sol
SOL
115 function nextRewardFor(address _recipient) public view returns (uint256) {
        uint256 reward;
```

We advise the function to properly sum the results of nextRewardAt invocations to ensure it operates as intended.

### **Alleviation:**

Rewards are now properly accumulated in the reward entry.

### **SDR-02M: Ungraceful Handling of High Adjustment Rates**

Туре	Severity	Location
Mathematical Operations	Medium ●	StakingDistributor.sol:L90

### **Description:**

The adjustment.rate is meant to represent a step-by-step reduction or increase of the reward rate for a particular recipient, however, there can be a case where the <code>info[\_index].rate</code> is smaller than the step which would render the adjust operation impossible and thus cause the full distribute hook to fail.

### **Example:**

### **Recommendation:**

We advise the reduction of a particular rate to be gracefully handled whereby if the reduction is greater than the current rate the rate should be set to zero.

### **Alleviation:**

The Olympus DAO team considered this exhibit but decided to retain the current behaviour of the code in place.

### **SDR-03M: Inexistent Validation of Entry Validity**

Туре	Severity	Location
Input Sanitization	Minor •	StakingDistributor.sol:L142-L147, L156-L169

### **Description:**

The removeRecipient and setAdjustment functions do not actually validate whether there is an existing entry in the \_index they are operating in, leading to adjustments for inexistent entries / future ones or removal of inexistent entries.

```
contracts/StakingDistributor.sol
SOL
                                                                               Copy
142 function removeRecipient(uint256 index, address recipient) external {
        require(msg.sender == governor() || msg.sender == guardian(), "Caller is
      require(_recipient == info[_index].recipient);
       info[ index].recipient = address(0);
       info[ index].rate = 0;
156 function setAdjustment(
       uint256 index,
       bool add,
```

We advise both functions to properly validate that an info entry exists by evaluating its recipient.

### Alleviation:

Both functions now properly validate the existance of an entry by ensuring that the info[\_index].recipient member is non-zero.

### SDR-04M: Inexistent Validation of Reward Rate

Туре	Severity	Location
Input Sanitization	Minor •	StakingDistributor.sol:L134

### **Description:**

The addrecipient function does not validate that the \_rewardRate set does not exceed the maximum achievable which is 1000000.

### **Example:**

### **Recommendation:**

We advise such validation to be imposed to prevent arbitrarily high reward rates.

### Alleviation:

The function now properly validate that the <u>rewardRate</u> set is at most equivalent to the newly declared <u>rateDenominator</u>.

Ownable.sol (OWN-M)

NEXT

StandardBondingCalculator.sol (SBC-M)

>

# StandardBondingCalculator Manual Review Findings

### ON THIS PAGE

SBC-01M: Inexistent Validation of Pair Tokens

SBC-02M: Incorrect Usage of SafeMath Library

### SBC-01M: Inexistent Validation of Pair Tokens

Туре	Severity	Location
Logical Fault	Major ●	StandardBondingCalculator.sol:L50-L60

### **Description:**

The markdown function incorrectly assumes that if the token of a pair is not the OHM address, token will be so which may not be the case.

```
contracts/StandardBondingCalculator.sol

SOL

Copy

function markdown( address _pair ) external view override returns ( uint ) {
    ( uint reserve0, uint reserve1, ) = IUniswapV2Pair( _pair ).getReserves()

uint reserve;

if ( IUniswapV2Pair( _pair ).token0() == address( OHM ) ) {
    reserve = reserve1;

} else {
    reserve = reserve0;

} reserve = reserve0;

}

return reserve.mul( 2 * ( 10 ** IERC20Metadata(address(OHM)).decimals() )
```

We advise a require check to be introduced in the else chain of the if clause that mandates token1 to be the OHM address.

### **Alleviation:**

A require check was introduced in the else case that mandates token1 to be equivalent to ohm thereby alleviating this exhibit.

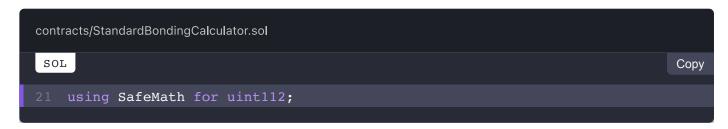
### SBC-02M: Incorrect Usage of SafeMath Library

Туре	Severity	Location
Language Specific	Minor •	StandardBondingCalculator.sol:L5, L21

### **Description:**

The using SafeMath for uint112 statement is ineffectual as all SafeMath operations that will be performed on the uint112 data type will indirectly cast the value to a uint256 and yield the uint256 result which if casted to a uint112 can still overflow.

### **Example:**



### **Recommendation:**

We advise either the SafeMath library implementation to be expanded to support the uint112 data type or the using statement to be omitted should it be considered unnecessary in the codebase and replaced by uint256 casts to uint112 variables that are used in these calculations.

### **Alleviation:**

The ineffectual using \* for statement was omitted from the codebase.

View Fix on GitHub

PREV
StakingDistributor.sol (SDR-M)



reasury.sol(IKE-M)

# **Treasury Manual Review Findings**

### ON THIS PAGE

TRE-01M: Insecure Management of Reserve & Liquidity Tokens

TRE-02M: Weak Debt Position Validation

TRE-03M: Improperly Valid Case

TRE-04M: Inexistent Validation of Token Status

TRE-05M: Potentially Unsafe Primitive Evaluation

# TRE-01M: Insecure Management of Reserve & Liquidity Tokens

Туре	Severity	Location
Logical Fault	Major •	Treasury.sol:L242, L244, L246, L248, L270, L333

### **Description:**

The registry arrays of reserve and liquidity tokens are improperly maintained which can cause huge discrepancies to the totalReserves measured in the auditReserves function. As an example, duplicate entries within a single array can cause a particular reserve to be calculated twice whereas an asset being a reserve and liquidity token at the same time will also cause an incorrect duplicate measurement of its balance value.

```
contracts/Treasury.sol

SOL

Copy

254 /**

255 * @notice enable permission from queue

256 * @param _status STATUS
```

```
260 function enable(
       STATUS _status,
      address _address,
       address _calculator
264 ) external onlyOwner {
       require(onChainGoverned, "OCG Not Enabled: Use queueTimelock");
       if (_status == STATUS.SOHM) {
           sOHM = IERC20(_address);
     } else {
            registry[ status].push( address);
           permissions[_status][_address] = true;
           if (_status == STATUS.LIQUIDITYTOKEN) {
                bondCalculator[_address] = _calculator;
           }
       emit Permissioned( address, status, true);
279 }
```

We advise all actions modifying those arrays to properly check for duplicates by preventing resetting the same permission for an address to true.

### Alleviation:

Both registry adjustment code blocks were updated to properly evaluate whether duplicate entries exist as well as to delete any previously set state in case the same token is being set between a liquidity and reserve token and vice versa.

### TRE-02M: Weak Debt Position Validation

Туре	Severity	Location
Logical Fault	Major •	Treasury.sol:L140-L163

### **Description:**

The incurpebt function mandates that the caller has a sufficient balance of SOHM to create the debt position, however, SOHM is a freely transferrable asset that should only be used as a data point when it cannot be transferred.

```
contracts/Treasury.sol
SOL
                                                                              Copy
145 function incurDebt(uint256 amount, address token) external override {
        require(permissions[STATUS.DEBTOR][msg.sender], "Not approved");
        require(permissions[STATUS.RESERVETOKEN][ token], "Not accepted");
       uint256 value = tokenValue( token, _amount);
       require(value != 0);
       uint256 availableDebt = sOHM.balanceOf(msg.sender).sub(debtorBalance[msg.
        require(value <= availableDebt, "Exceeds debt limit");
        debtorBalance[msg.sender] = debtorBalance[msg.sender].add(value);
        totalDebt = totalDebt.add(value);
        totalReserves = totalReserves.sub(value);
        IERC20( token).transfer(msg.sender, amount);
```

```
162 emit CreateDebt(msg.sender, _token, _amount, value);

163 }
```

We advise the **SOHM** to either be held in custody or for some other similar mechanism to be put in place as the current debt mechanism is circumventable.

### **Alleviation:**

The debt management system has now been built-in the **SOHM** implementation which in turn prevents transfers that would reduce the holder's balance below the required debt threshold.

### **TRE-03M: Improperly Valid Case**

Туре	Severity	Location
Mathematical Operations	Medium •	Treasury.sol:L150, L380-L386

### **Description:**

The tokenvalue function should not yield a value of o under any circumstances as it will result in no-ops when utilized in mathematical operations and can cause the system to misbehave in case i.e. a token has more decimals than OHM is utilized in the evaluation that is not a LIQUIDITYTOKEN.

### **Example:**

### **Recommendation:**

We strongly recommend the first linked require check to be relocated to the tokenValue function itself as no zero evaluations should be considered "valid".

### Alleviation:

The Olympus DAO team considered this exhibit but decided to retain the current behaviour of the code in place.

### TRE-04M: Inexistent Validation of Token Status

Туре	Severity	Location
Logical Fault	Medium •	Treasury.sol:L208-L210

### **Description:**

The manage function does not properly validate the permission status of the input \_token which can improperly manipulate the totalReserves i.e. with no-op function implementations.

```
contracts/Treasury.sol
SOL
                                                                              Copy
205 function manage(address token, uint256 amount) external override {
        if (permissions[STATUS.LIQUIDITYTOKEN][ token]) {
            require(permissions[STATUS.LIQUIDITYMANAGER][msg.sender], "Not approv
            require(permissions[STATUS.RESERVEMANAGER][msg.sender], "Not approved
        uint256 value = tokenValue(_token, _amount);
        require(value <= excessReserves(), "Insufficient reserves");</pre>
        totalReserves = totalReserves.sub(value);
        IERC20( token).safeTransfer(msg.sender, amount);
        emit ReservesManaged( token, amount);
220 }
```

Necommendation.

We advise the else branch of the first clause in manage to impose a require check ensuring that the permission of the token specified is as a RESERVETOKEN.

### **Alleviation:**

The totalReserves value is now adjusted solely when the input token falls under either the liquidity or reserve token category, thereby alleviating this exhibit.

### **TRE-05M: Potentially Unsafe Primitive Evaluation**

Туре	Severity	Location
Standard Conformity	Minor •	Treasury.sol:L380-L386

### **Description:**

The evaluation of a particular token's value when the token is not a **LIQUIDITYTOKEN** is performed by a simple decimal-based conversion.

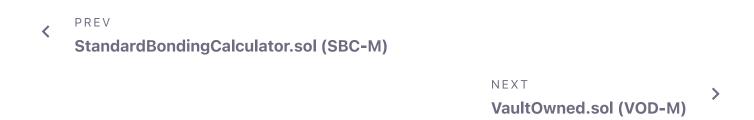
### **Example:**

### **Recommendation:**

Apart from considering these tokens equal in value, it also directly relies on the presence of the decimals operator on the token which at times may not be available as it is an OPTIONAL member of the EIP-20 standard. We advise this particular trait to be considered carefully in the overall system as checks can be imposed at the inclusion level to prevent non-compliant tokens from being added.

### **Alleviation:**

The Olympus DAO team considered this exhibit but decided to retain the current behaviour of the code in place.



# **VaultOwned Manual Review Findings**

### ON THIS PAGE

VOD-01M: Centralized Sensitive Functionality

### **VOD-01M: Centralized Sensitive Functionality**

Туре	Severity	Location
Logical Fault	Medium •	VaultOwned.sol:L10-L14

### **Description:**

The setvault function sets the current vault in place for onlyvault modifier enforcement, however, it can be invoked an arbitrary number of times, can override the current vault in place and does not contain an override keyword meaning that it is meant to be invoked by EOAs or similar actors.

### **Example:**

```
contracts/types/VaultOwned.sol
SOL
                                                                                 Copy
    function setVault( address vault ) external onlyOwner() returns ( bool ) {
      vault = vault ;
      return true;
14 }
```

### Recommendation:

We advise it to potentially disallow over-writing the vault in place once it has been set once as otherwise, it can become a single point of failure for the system. Additionally, we advise the returned bool to be omitted given that it is always true and is a non-standard function.

### **Alleviation:**

The Olympus DAO team considered this exhibit but decided to retain the current behaviour of the code in place.





## **gOHM Manual Review Findings**

ON THIS PAGE

OHM-01M: Improper State Control of Migration

### **OHM-01M: Improper State Control of Migration**

Туре	Severity	Location
Logical Fault	Medium •	gOHM.sol:L120-L134

### **Description:**

The migrate function, as its documentation states, should only be invoke-able once during the contract migration, however, the logical checks it enforces allow it to be invoked and thus set very sensitive contract variables an arbitrary number of times.

```
contracts/governance/gOHM.sol

SOL

Copy

120 /**
121 * @notice transfer mint rights from migrator to staking

122 * @notice can only be done once, at the time of contract migration

123 * @param _staking address

124 * @param _sOHM address

125 */

126 function migrate(address _staking, address _sOHM) external override onlyAppro

127     require(_staking != approved);

128

129     require(_staking != address(0));

130     approved = _staking;

131
```

```
require(_sOHM != address(0));

sohm = IsOHM(_sOHM);

134 }
```

We strongly recommend the function to ensure that **SOHM** has not been previously set as otherwise, there is no protection preventing re-setting those variables.

### **Alleviation:**

A dedicated migrated flag has been introduced to the codebase and is being used as a flag to indicate whether migrate has been invoked before thereby preventing its re-execution and alleviating this exhibit.

View Fix on GitHub

NEXT > sOlympusERC20.sol (OEC-M)

# sOlympusERC20 Manual Review Findings

### ON THIS PAGE

OEC-01M: Potentially Incorrect Extrapolation of Rebase

### **OEC-01M: Potentially Incorrect Extrapolation of Rebase**

Туре	Severity	Location
Logical Fault	Medium •	sOlympusERC20.sol:L122

### **Description:**

The rebase function will extrapolate the rebaseAmount should a non-zero amount of circulatingSupply be returned as the calculation performed multiplies the profit by the totalSupply and divides it by the circulatingSupply, the latter of which is guaranteed to be greater than the former thus causing the profit to be increased.

We advise this trait to be carefully assessed and if desired to be properly documented as it can cause disproportionate profits to be calculated.

### Alleviation:

The Olympus DAO team considered this exhibit, identified it as desired behaviour but opted not to apply any remediation for it.

```
PREV
gOHM.sol (OHM-M)
```

```
NEXT

BondDepository.sol (BDY-C)
```

ID	Severity	Addressed	Title
BDY-03M	Medium	No	Inexplicable Optional Value of Decay
BTR-01M	Major	Yes	Confusion of Value Denominations
BTR-02M	Medium	No	Artificial Inflation Mechanism
BTR-03M	Minor	Yes	Inexistent Redemption of FEO Fees
BTR-04M	Minor	No	Inexistent Validation of Non-Zero Redemption
ERC-01M	Minor	Yes	Non-Standard Mint Implementation
ERP-01M	Medium	Yes	Insecure Elliptic Curve Recovery Mechanism
ERP-02M	Minor	Yes	Cross-Chain Signature Replay Attack Susceptibility
FPT-01M	Major	No	Potentially Invalid Implementation
GOV-01M	Medium	Yes	Improper Governor Renouncation
GOV-02M	Minor	Yes	Incorrect Event Emitted
GOV-03M	Minor	Yes	Potentially Restrictive Functionality
GAA-01M	Informational	Yes	Improper Percentage Documented
GUA-01M	Medium	Yes	Improper Guardian Renouncation
GUA-02M	Minor	Yes	Incorrect Event Emitted
GUA-03M	Minor	Yes	Potentially Restrictive Functionality
OTM-01M	Major	Yes	Improper Integration w/ Uniswap V2
OTM-02M	Medium	Yes	Improper Evaluation of Token Balance
ULM-USM	Madium	Voc	Ungraceful Mint Handling

ID	Severity	Addressed	Title
OTM-04M	Minor	Yes	Potential of Repeat Invocation
OWN-01M	Medium	Yes	Improper Ownership Renouncation
OWN-02M	Minor	Yes	Incorrect Events Emitted
OWN-03M	Minor	Yes	Potentially Restrictive Functionality
SDR-01M	Medium	Yes	Improper Accumulation of Rewards
SDR-02M	Medium	No	Ungraceful Handling of High Adjustment Rates
SDR-03M	Minor	Yes	Inexistent Validation of Entry Validity
SDR-04M	Minor	Yes	Inexistent Validation of Reward Rate
SBC-01M	Major	Yes	Inexistent Validation of Pair Tokens
SBC-02M	Minor	Yes	Incorrect Usage of SafeMath Library
TRE-01M	Major	Yes	Insecure Management of Reserve & Liquidity Tokens
TRE-02M	Major	Yes	Weak Debt Position Validation
TRE-03M	Medium	No	Improperly Valid Case
TRE-04M	Medium	Yes	Inexistent Validation of Token Status
TRE-05M	Minor	No	Potentially Unsafe Primitive Evaluation
VOD-01M	Medium	No	Centralized Sensitive Functionality
OHM-01M	Medium	Yes	Improper State Control of Migration
OEC-01M	Medium	No	Potentially Incorrect Extrapolation of Rebase

PREV
Static Analysis

NEXT >

# **Code Style**

During the manual portion of the audit, we identified **34 optimizations** that can be applied to the codebase that will decrease the gas-cost associated with the execution of a particular function and generally ensure that the project complies with the latest best practices and standards in Solidity.

Additionally, this section of the audit contains any opinionated adjustments we believe the code should make to make it more legible as well as truer to its purpose.

These optimizations are enumerated below:

ID	Severity	Addressed	Title
BDY-01C	Informational	No	Improper Error Messages
BDY-02C	Informational	No	Inexistent Error Messages
BDY-03C	Informational	No	Potentially Misleading USD Evaluation
BDY-04C	Informational	No	Redundant Explicit Zero-Value Assignment
BTR-01C	Informational	Yes	Inexistent Variable Visibility Specifiers
ERC-01C	Informational	Yes	Variable Mutability Specifier
ERP-01C	Informational	Yes	Improper Error Name
ERP-02C	Informational	Yes	Redundant Visibility Specifiers
ERP-03C	Informational	Yes	Suboptimal Code Style
FMH-01C	Informational	Yes	Outdated Implementation
GOV-01C	Informational	Yes	Inexistent Deletion of Pending Governor

# **BondDepository Code Style Findings**

#### ON THIS PAGE

BDY-01C: Improper Error Messages

BDY-02C: Inexistent Error Messages

BDY-03C: Potentially Misleading USD Evaluation

BDY-04C: Redundant Explicit Zero-Value Assignment

## **BDY-01C: Improper Error Messages**

Туре	Severity	Location
Code Style	Informational •	BondDepository.sol:L187, L191

## **Description:**

The linked error messages state that the bond has concluded, however, the same error will arise if the <code>capacity</code> is simply exceeded rather than <code>0</code> and would otherwise succeed with a <code>payout</code> / <code>amount</code> less than the limit.

```
contracts/BondDepository.sol

SOL

Copy

184 // ensure there is remaining capacity for bond

185 if (info.capacityIsPayout) {

186    // capacity in payout terms

187    require(info.capacity >= payout, "Bond concluded");

188    info.capacity = info.capacity.sub(payout);

189 } else {

190    // capacity in principal terms

191    require(info.capacity >= _amount, "Bond concluded");
```

```
info.capacity = info.capacity.sub(_amount);

193 }
```

## **Recommendation:**

We advise the error message to be more descriptive to aid users in adjusting the variables correctly to ensure their action succeeds.

## **Alleviation:**

The Olympus DAO team considered this exhibit but opted not to apply any remediation for it.

## **BDY-02C: Inexistent Error Messages**

Туре	Severity	Location
Code Style	Informational •	BondDepository.sol:L68, L70, L144, L145

## **Description:**

The linked require checks contain no descriptive error messages.

## **Example:**

```
contracts/BondDepository.sol

SOL

Copy

67  constructor(address _OHM, address _treasury) {
    require(_OHM != address(0));
    OHM = IERC20(_OHM);
    require(_treasury != address(0));
    treasury = ITreasury(_treasury);
    }
```

#### **Recommendation:**

We advise them to be set so to aid in the debugging of the application and to also enable more accurate validation of the require condition purposes.

#### Alleviation:

The Olympus DAO team considered this exhibit but opted not to apply any remediation for it.

## **BDY-03C: Potentially Misleading USD Evaluation**

Туре	Severity	Location
Code Style	Informational •	BondDepository.sol:L343-L355

## **Description:**

The on-chain USD evaluation of a bond is solely used for events, however, it appears to be incorrect as its comments indicate that a DAI value is calculated whereas the principal's paired assets are simply used with no relation to the actual DAI token.

## **Example:**

```
contracts/BondDepository.sol

SOL

Copy

343 /**
344 * @notice converts bond price to DAI value
345 * @param _BID uint
346 * @return price_ uint
347 */
348 function bondPriceInUSD(uint256 _BID) public view returns (uint256 price_) {
349    Bond memory bond = bonds[_BID];
350    if (address(bond.calculator) != address(0)) {
351        price_ = bondPrice(_BID).mul(bond.calculator.markdown(address(bond.princi));
352    } else {
353        price_ = bondPrice(_BID).mul(10**IERC20Metadata(address(bond.principal));
354    }
355 }
```

#### **Recommendation:**

We advise either the code or the comments surrounding it to be revised to better illustrate the function's purpose.

#### **Alleviation:**

The Olympus DAO team considered this exhibit but opted not to apply any remediation for it.

## **BDY-04C: Redundant Explicit Zero-Value Assignment**

Туре	Severity	Location
Gas Optimization	Informational •	BondDepository.sol:L89

#### **Description:**

The addBond function redundantly assigns the Terms memory terms pointer to an uninitialized Terms entry.

## **Example:**

```
contracts/BondDepository.sol
SOL
    * @param capacityIsPayout bool
   function addBond(
   address principal,
    address calculator,
    uint256 capacity,
    bool capacityIsPayout
88 ) external onlyGuardian returns (uint256 id ) {
     Terms memory terms = Terms({controlVariable: 0, fixedTerm: false, vestingTe
     bonds[IDs.length] = Bond({principal: IERC20( principal), calculator: IBondi
    id = IDs.length;
     IDs.push( principal);
95 }
```

#### **Recommendation:**

We advise the assignment to be omitted and the pointer to be used in the next statement directly as it points to an uninitialized Terms struct when declared.

## **Alleviation:**

The Olympus DAO team considered this exhibit but opted not to apply any remediation for it.





# **BondTeller Code Style Findings**

ON THIS PAGE

BTR-01C: Inexistent Variable Visibility Specifiers

## **BTR-01C: Inexistent Variable Visibility Specifiers**

Туре	Severity	Location
Code Style	Informational •	BondTeller.sol:L44-L49

## **Description:**

The linked variables have no visibility specifier explicitly set.

## **Example:**

```
contracts/BondTeller.sol

SOL

Copy

44 address depository; // contract where users deposit bonds
45 IStaking immutable staking; // contract to stake payout
46 ITreasury immutable treasury;
47 IERC20 immutable OHM;
48 IERC20 immutable sOHM; // payment token
49 IgOHM immutable gOHM;
```

#### **Recommendation:**

We advise one to be set so to avoid potential compilation discrepancies in the future as the current compiler behaviour is to assign a specifier automatically.

#### **Alleviation:**

All variables were set to internal thereby alleviating this exhibit.

View Fix on GitHub

PREV
BondDepository.sol (BDY-C)

ERC20.sol (ERC-C)

# **ERC20 Code Style Findings**

ON THIS PAGE

ERC-01C: Variable Mutability Specifier

## **ERC-01C: Variable Mutability Specifier**

Туре	Severity	Location
Gas Optimization	Informational •	ERC20.sol:L26, L31

## **Description:**

As the ERC20 token implementation is used internally, the \_decimals member is only set once during the constructor of the contract.

## **Example:**

## **Recommendation:**

We advise it to be set as immutable greatly opimizing its read access gas cost.

## **Alleviation:**

The \_decimals member was properly set as immutable.

View Fix on GitHub

PREV
BondTeller.sol (BTR-C)

NEXT

ERC20Permit.sol (ERP-C)



# **ERC20Permit Code Style Findings**

#### ON THIS PAGE

**ERP-01C: Improper Error Name** 

ERP-02C: Redundant Visibility Specifiers

ERP-03C: Suboptimal Code Style

## **ERP-01C: Improper Error Name**

Туре	Severity	Location
Code Style	Informational •	ERC20Permit.sol:L56

## **Description:**

The error name references another contract that does not exist in the codebase.

## **Example:**



#### **Recommendation:**

We advise the error to be renamed.

#### Alleviation:

The error message has been corrected to utilize the contract's name instead.

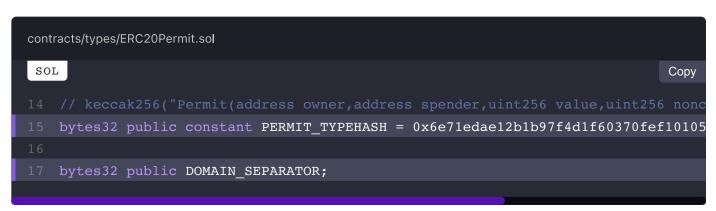
## **ERP-02C: Redundant Visibility Specifiers**

Туре	Severity	Location
Gas Optimization	Informational •	ERC20Permit.sol:L15, L17

## **Description:**

The linked variables are meant to be internally available constant variables.

## **Example:**



#### **Recommendation:**

We advise them to be set as private or internal to reduce the codebase bloat and bytecode size of the contract.

#### Alleviation:

Visibility specifiers were adjusted for both variables to ensure they are no longer exposed publicly.

View Fix on GitHub

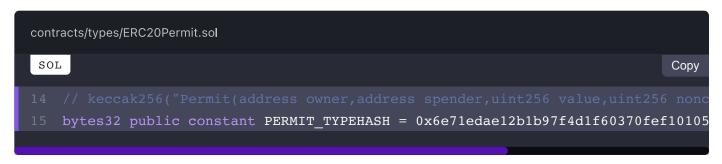
## **ERP-03C: Suboptimal Code Style**

Туре	Severity	Location
Code Style	Informational •	ERC20Permit.sol:L14, L15

#### **Description:**

The PERMIT\_TYPEHASH variable has the hash declared as its assignment and the keccak256 instruction as its comment.

## **Example:**



#### **Recommendation:**

We advise these to be reversed as the one-way association of keccak256 is not guaranteed from a hash to its value but rather from a value to its hash. Additionally, validation of the values would have less implications as the keccak256 instruction is guaranteed to be performed properly. To note, assigning a constant to a keccak256 instruction's result does not affect gas cost as the value is pre-calculated by the compiler.

#### **Alleviation:**

The typehash is now properly assigned to the evaluation of a keccak256 instruction and is set as immutable to benefit from the gas optimization of evaluating the expression once.

View Fix on GitHub

NEXT

FullMath.sol (FMH-C)

>

# **FullMath Code Style Findings**

ON THIS PAGE

FMH-01C: Outdated Implementation

## **FMH-01C: Outdated Implementation**

Туре	Severity	Location
Gas Optimization	Informational •	FullMath.sol:L33-L44

## **Description:**

The FullMath contract is a copy of the homonymous Uniswap V2 library and implements an outdated version of the muldiv function which does not contain the optimization of h == 0 as present in the Uniswap V2 equivalent.

```
contracts/libraries/FullMath.sol

SOL

Copy

33  function mulDiv(
34     uint256 x,
35     uint256 d,
36     uint256 d

37  ) internal pure returns (uint256) {
38     (uint256 l, uint256 h) = fullMul(x, y);
39     uint256 mm = mulmod(x, y, d);
40     if (mm > l) h -= 1;
41     l -= mm;
42     require(h < d, 'FullMath::mulDiv: overflow');
43     return fullDiv(l, h, d);
44 }</pre>
```

## **Recommendation:**

We advise the optimization to be applied to reduce the gas cost involved with the library.

## **Alleviation:**

The optimization of Uniswap V2 was properly integrated into the codebase.

View Fix on GitHub

PREV
ERC20Permit.sol (ERP-C)

NEXT
Governable.sol (GOV-C)

# **Governable Code Style Findings**

ON THIS PAGE

GOV-01C: Inexistent Deletion of Pending Governor

## **GOV-01C: Inexistent Deletion of Pending Governor**

Туре	Severity	Location
Gas Optimization	Informational •	Governable.sol:L43-L47

## **Description:**

The \_newGovernor entry should be deleted when it is consumed by the pullGovernor function.

## **Example:**

```
contracts/types/Governable.sol

SOL

43 function pullGovernor() public virtual override {
44    require( msg.sender == _newGovernor, "Governable: must be new governor to
45    emit GovernorPulled( _governor, _newGovernor );
46    _governor = _newGovernor;
47 }
```

## **Recommendation:**

We advise it to be deleted so to ensure a consistent contract state.

#### Alleviation:

The new governor is now properly deleted when the pullGovernor function concludes.

View Fix on GitHub

FullMath.sol (FMH-C)

NEXT > Guardable.sol (GUA-C)

# **Guardable Code Style Findings**

ON THIS PAGE

GUA-01C: Inexistent Deletion of Pending Guardian

## **GUA-01C: Inexistent Deletion of Pending Guardian**

Туре	Severity	Location
Gas Optimization	Informational •	Guardable.sol:L42-L46

## **Description:**

The \_newGuardian entry should be deleted when it is consumed by the pullGuardian function.

## **Example:**

```
contracts/types/Guardable.sol

SOL

Copy

42 function pullGuardian() public virtual override {
    require( msg.sender == _newGuardian, "Guardable: must be new guardian to
    emit GuardianPulled( _guardian, _newGuardian );
    _guardian = _newGuardian;
46 }
```

## **Recommendation:**

We advise it to be deleted so to ensure a consistent contract state.

#### Alleviation:

The new guardian is now properly deleted when the pullGuardian function concludes.

View Fix on GitHub

Governable.sol (GOV-C)

ManagerOwnable.sol (MOE-C)

# ManagerOwnable Code Style Findings

ON THIS PAGE

MOE-01C: Redundant Implementation

## **MOE-01C: Redundant Implementation**

Туре	Severity	Location
Code Style	Informational •	ManagerOwnable.sol:L7-L10

## **Description:**

The ManagerOwnable contract is redundant as it declares a new modifier labelled onlyManager that is exactly the same as the onlyOwner modifier, inclusive of the error messages.

## **Example:**

## **Recommendation:**

We advise the implementation to be omitted from the codebase entirely.

## **Alleviation:**

The contract is no longer part of the codebase rendering this exhibit null.

View Fix on GitHub

Guardable.sol (GUA-C)

NEXT
OlympusERC20.sol (OER-C)

# OlympusERC20 Code Style Findings

ON THIS PAGE

**OER-01C: Incorrect Function Visibility** 

## **OER-01C: Incorrect Function Visibility**

Туре	Severity	Location
Code Style	Informational •	OlympusERC20.sol:L34

## **Description:**

The \_burnFrom function is incorrectly available externally by its public modifier.

## **Recommendation:**

We advise it to be set to internal to properly illustrate its purpose and avoid potential circumventions of the burnFrom function in the future.

## **Alleviation:**

The visibility specifier of the \_burnFrom function was adjusted according to our recommendation.

View Fix on GitHub

>

PREV
ManagerOwnable.sol (MOE-C)

NEXT
OlympusTokenMigrator.sol (OTM-C)

# OlympusTokenMigrator Code Style Findings

#### **ON THIS PAGE**

OTM-01C: Inexistent Error Messages

OTM-02C: Multiple Top-Level Declarations

OTM-03C: Redundant & Confusing Comparisons

## **OTM-01C: Inexistent Error Messages**

Туре	Severity	Location
Code Style	Informational •	OlympusTokenMigrator.sol:L99, L101, L103, L105, L107, L109, L111, L210, L238, L239, L328, L330, L332

## **Description:**

The linked require checks have no explicit error messages defined.

```
contracts/migration/OlympusTokenMigrator.sol

SOL

Copy

208 // withdraw backing of migrated OHM
209 function defund(address reserve) external onlyGovernor {

210     require(ohmMigrated && timelockEnd < block.number && timelockEnd != 0);

211     oldwsOHM.unwrap(oldwsOHM.balanceOf(address(this)));

212

213     uint256 amountToUnstake = oldsOHM.balanceOf(address(this));

214     oldsOHM.approve(address(oldStaking), amountToUnstake);

215     oldStaking.unstake(amountToUnstake, false);

216</pre>
```

```
uint256 balance = oldOHM.balanceOf(address(this));

218

219     oldSupply = oldSupply.sub(balance);

220

221     uint256 amountToWithdraw = balance.mul(le9);

222     oldOHM.approve(address(oldTreasury), amountToWithdraw);

223     oldTreasury.withdraw(amountToWithdraw, reserve);

224     IERC20(reserve).safeTransfer(address(newTreasury), IERC20(reserve).balance

225

226     emit Defunded(balance);

227 }
```

#### **Recommendation:**

We advise them to be set so to aid in the validation of the conditionals as well as in debugging the application.

#### Alleviation:

Error messages were included to all require checks across the contract.

View Fix on GitHub

## **OTM-02C: Multiple Top-Level Declarations**

Туре	Severity	Location
Code Style	Informational •	OlympusTokenMigrator.sol:L17-L44, L46-L50

## **Description:**

The linked interface implementations should be relocated to dedicated files to not pollute the top-level of the contract file.

```
contracts/migration/OlympusTokenMigrator.sol
                                                                                 Сору
SOL
    interface IUniswapV2Router {
        function addLiquidity(
            address tokenA,
            address tokenB,
            uint256 amountADesired,
            uint256 amountBDesired,
            uint256 amountAMin,
            uint256 amountBMin,
            address to,
            uint256 deadline
            external
            returns (
                uint256 amountA,
                uint256 amountB,
                uint256 liquidity
            );
        function removeLiquidity(
            address tokenA,
            address tokenB,
            uint256 liquidity,
            uint256 amountAMin,
            uint256 amountBMin,
```

```
address to,
uint256 deadline
) external returns (uint256 amountA, uint256 amountB);

44 }

45 

46 interface IStakingV1 {

47 function unstake(uint256 _amount, bool _trigger) external;

48

49 function index() external view returns (uint256);

50 }

51 
52 contract OlympusTokenMigrator is OlympusAccessControlled {
```

#### **Recommendation:**

We advise them to be relocated to the interfaces sub-folder, potentially under an external second-level subfolder, to ensure that the code structure of the system is maintainable.

#### **Alleviation:**

All required interfaces were split to their dedicated files and are now properly imported to the codebase.

View Fix on GitHub

## **OTM-03C: Redundant & Confusing Comparisons**

Туре	Severity	Location
Gas Optimization	Informational •	OlympusTokenMigrator.sol:L137, L182

## **Description:**

The if-else structure within migrate and send evaluate all states of the enum redundantly which can also cause ambiguous behaviour if the compiler does not enforce the value range of the TYPE enum due to a compiler issue given that it is internally represented by a uint8.

```
contracts/migration/OlympusTokenMigrator.sol
SOL
                                                                              Copy
118 enum TYPE {
       UNSTAKED,
       STAKED,
       WRAPPED
122 }
125 function migrate(
      uint256 amount,
      TYPE from,
      TYPE to
129 ) external {
      uint256 sAmount = amount;
       uint256 wAmount = oldwsOHM.sOHMTowOHM( amount);
        if ( from == TYPE.UNSTAKED) {
            oldOHM.safeTransferFrom(msg.sender, address(this), amount);
        } else if ( from == TYPE.STAKED) {
            oldsOHM.safeTransferFrom(msg.sender, address(this), amount);
        } else if ( from == TYPE.WRAPPED) {
            oldwsOHM.safeTransferFrom(msg.sender, address(this), amount);
            wAmount = amount;
```

#### **Recommendation:**

We advise the last else if branch to be converted to an else branch to ensure transfer of funds is performed at all times from the user to the contract and vice versa for the migration to occur.

#### **Alleviation:**

The if-else optimization was only applied to the first linked segment thereby partially alleviating this exhibit.

```
PREV
OlympusERC20.sol (OER-C)
```

NEXT
Ownable.sol (OWN-C)

## **Ownable Code Style Findings**

ON THIS PAGE

OWN-01C: Inexistent Deletion of Pending Owner

## **OWN-01C: Inexistent Deletion of Pending Owner**

Туре	Severity	Location
Gas Optimization	Informational •	Ownable.sol:L39-L43

## **Description:**

The \_newOwner entry should be deleted when it is consumed by the pullManagement function.

## **Example:**

```
contracts/types/Ownable.sol

SOL

SOL

Copy

39 function pullManagement() public virtual override {
40     require( msg.sender == _newOwner, "Ownable: must be new owner to pull");
41     emit OwnershipPulled( _owner, _newOwner );
42     _owner = _newOwner;
43 }
```

#### **Recommendation:**

We advise it to be deleted so to ensure a consistent contract state.

#### Alleviation:

The new owner is now properly deleted when the pullManagement function concludes.

PREV
OlympusTokenMigrator.sol (OTM-C)

SafeMath.sol (SMH-C)

# SafeMath Code Style Findings

ON THIS PAGE

SMH-01C: Inefficient Implementation

## **SMH-01C: Inefficient Implementation**

Туре	Severity	Location
Gas Optimization	Informational •	SafeMath.sol:L115-L127

## **Description:**

The sqqrt implementation does not efficiently calculate the root of the provided argument as it wraps operations unnecessarily (i.e. divisions with non-zero value literals).

#### **Recommendation:**

We advise a more efficient square root algorithm to be implemented instead, such as the optimized Babylonian method by Uniswap.

#### **Alleviation:**

The Olympus DAO team considered this exhibit but opted not to apply any remediation for it.

Ownable.sol (OWN-C)

NEXT
StakingDistributor.sol (SDR-C)

## Staking Distributor Code Style Findings

#### ON THIS PAGE

SDR-01C: Inexistent Error Messages

SDR-02C: Inexistent Variable Visibility Specifiers

### **SDR-01C: Inexistent Error Messages**

Туре	Severity	Location
Code Style	Informational •	StakingDistributor.sol:L48, L50, L52, L133, L144

#### **Description:**

The linked require checks contain no descriptive error messages.

### **Example:**

```
contracts/StakingDistributor.sol
SOL
                                                                                Copy
   constructor(
        address _treasury,
       address _ohm,
       address staking
    ) {
        require(_treasury != address(0));
        treasury = ITreasury(_treasury);
        require( ohm != address(0));
        OHM = IERC20(ohm);
        require(_staking != address(0));
        staking = _staking;
54 }
```

#### **Recommendation:**

We advise them to be set so to aid in the debugging of the application and to also enable more accurate validation of the require condition purposes.

### **Alleviation:**

Error messages were introduced in all linked require checks.

### **SDR-02C: Inexistent Variable Visibility Specifiers**

Туре	Severity	Location
Code Style	Informational •	StakingDistributor.sol:L21, L22, L23

#### **Description:**

The linked variables have no visibility specifier explicitly set.

#### **Example:**



#### **Recommendation:**

We advise one to be set so to avoid potential compilation discrepancies in the future as the current compiler behaviour is to assign a specifier automatically.

#### Alleviation:

Proper visibility specifiers were set for all linked variables.

View Fix on GitHub

SafeMath.sol (SMH-C)



# StandardBondingCalculator Code Style **Findings**

#### ON THIS PAGE

SBC-01C: Inexistent Error Message

SBC-02C: Inexistent Variable Visibility Specifier

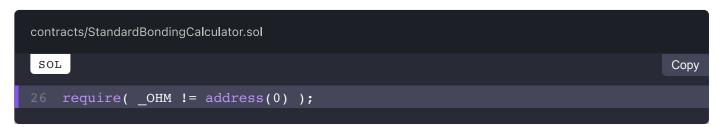
### **SBC-01C: Inexistent Error Message**

Туре	Severity	Location
Code Style	Informational •	StandardBondingCalculator.sol:L26

#### **Description:**

The linked require check contains no descriptive error message.

### **Example:**



#### **Recommendation:**

We advise one to be set so to aid in the debugging of the application and to also enable more accurate validation of the require condition's purpose.

#### Alleviation:

Error messages were introduced in all linked require checks.

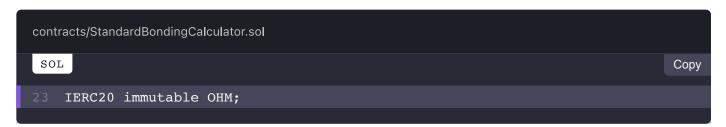
### **SBC-02C: Inexistent Variable Visibility Specifier**

Туре	Severity	Location
Code Style	Informational •	StandardBondingCalculator.sol:L23

#### **Description:**

The linked variable has no visibility specifier explicitly set.

#### **Example:**



#### **Recommendation:**

We advise one to be set so to avoid potential compilation discrepancies in the future as the current compiler behaviour is to assign a specifier automatically.

#### Alleviation:

The internal visibility specifier was properly introduced to the linked variable.

View Fix on GitHub

PREV
StakingDistributor.sol (SDR-C)

NEXT > Timelock.sol (TIM-C)

## **Timelock Code Style Findings**

ON THIS PAGE

TIM-01C: Redundant Implementation

### **TIM-01C: Redundant Implementation**

Туре	Severity	Location
Language Specific	Informational •	Timelock.sol:L53-L64

#### **Description:**

The sqrrt implementation present within the in-file declared safeMath function is redundant and overly convoluted by wrapping each statement with its safeMath equivalent, which at times is unnecessary such as when dividing with non-zero value literals.

#### **Example:**

```
contracts/governance/Timelock.sol

SOL

Copy

53 function sqrrt(uint256 a) internal pure returns (uint c) {
54    if (a > 3) {
55        c = a;
56        uint b = add( div( a, 2), 1 );
57        while (b < c) {
58             c = b;
59             b = div( add( div( a, b ), b), 2 );
60        }
61    } else if (a != 0) {
62             c = 1;
63    }
64 }</pre>
```

#### **Recommendation:**

We advise the implementation to be entirely omitted to also ensure that source code match analysis detects the Timelock contract as being an identical copy of Compound's implementation.

#### **Alleviation:**

The sqrrt function was safely omitted from the codebase.

View Fix on GitHub

PREV
StandardBondingCalculator.sol (SBC-C)

NEXT > Treasury.sol (TRE-C)

## **Treasury Code Style Findings**

#### ON THIS PAGE

TRE-01C: Improper Failure Enforcement

TRE-02C: Improper Permitted Execution Flow

TRE-03C: Inexistent Error Messages

TRE-04C: Inexistent Variable Visibility Specifier

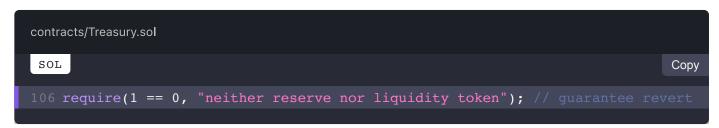
### **TRE-01C: Improper Failure Enforcement**

Туре	Severity	Location
Code Style	Informational •	Treasury.sol:L106

#### **Description:**

The require check linked performs a guaranteed-to-fail check (1 == 0) to illustrate the error message that accompanies it.

### **Example:**



#### **Recommendation:**

We advise a revert to be used directly instead that accepts the error message directly.

#### **Alleviation:**

THE TEQUITE CHOCK Was Substituted for a TEVELL CHOCK as recommended.

### **TRE-02C: Improper Permitted Execution Flow**

Туре	Severity	Location
Code Style	Informational •	Treasury.sol:L353-L362

### **Description:**

The enableOnchainGovernance function should not be invoke-able if the onchainGoverned status has already been set.

#### **Example:**

```
contracts/Treasury.sol

SOL

Copy

353 /**
354 * @notice disables timelocked functions
355 */
356 function enableOnChainGovernance() external onlyOwner {

357     if (onChainGovernanceTimelock != 0 && onChainGovernanceTimelock <= block.
358          onChainGoverned = true;
359     } else {
360          onChainGovernanceTimelock = block.number.add(blocksNeededForQueue.mul.)
361     }
362 }</pre>
```

#### **Recommendation:**

We advise this to be enforced by introducing a require check that prevents this scenario at the top of the function.

#### Alleviation:

The function now properly validates that onchainGoverned has not been set already.

### **TRE-03C: Inexistent Error Messages**

Туре	Severity	Location
Code Style	Informational •	Treasury.sol:L81, L150, L305, L321

#### **Description:**

The linked require checks contain no descriptive error messages.

#### **Example:**

```
contracts/Treasury.sol

SOL

80  constructor(address _OHM, uint256 _timelock) {
81     require(_OHM != address(0));
82     OHM = IOHMERC20(_OHM);
83

84     blocksNeededForQueue = _timelock;
85 }
```

#### **Recommendation:**

We advise them to be set so to aid in the debugging of the application and to also enable more accurate validation of the require condition purposes.

#### **Alleviation:**

Error messages were introduced in all linked require checks.

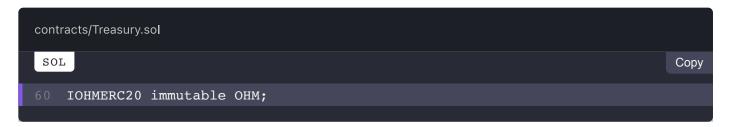
### **TRE-04C: Inexistent Variable Visibility Specifier**

Туре	Severity	Location
Code Style	Informational •	Treasury.sol:L60

#### **Description:**

The linked variable has no visibility specifier explicitly set.

#### **Example:**



#### **Recommendation:**

We advise one to be set so to avoid potential compilation discrepancies in the future as the current compiler behaviour is to assign a specifier automatically.

#### Alleviation:

The public visibility specifier was explicitly set to the linked variable.

View Fix on GitHub

PREV
Timelock.sol (TIM-C)

gOHM.sol (OHM-C)

## **gOHM Code Style Findings**

ON THIS PAGE

OHM-01C: Inexistent Error Messages

### **OHM-01C: Inexistent Error Messages**

Туре	Severity	Location
Code Style	Informational •	gOHM.sol:L59, L127, L129, L132

#### **Description:**

The linked require checks contain no descriptive error messages.

### **Example:**

```
contracts/governance/gOHM.sol

SOL

SOL

Copy

58  constructor(address _migrator) {
    require(_migrator != address(0));
    approved = _migrator;
    61 }
```

#### **Recommendation:**

We advise them to be set so to aid in the debugging of the application and to also enable more accurate validation of the require condition purposes.

#### **Alleviation:**

Error messages were introduced in all linked require checks.

PREV
Treasury.sol (TRE-C)

NEXT
sOlympusERC20.sol (OEC-C)

## sOlympusERC20 Code Style Findings

#### ON THIS PAGE

OEC-01C: Deprecated Representation Style

OEC-02C: Inefficient Code Structure
OEC-03C: Inexistent Error Messages

OEC-04C: Inexistent Variable Visibility Specifiers

OEC-05C: Redundant Event Argument

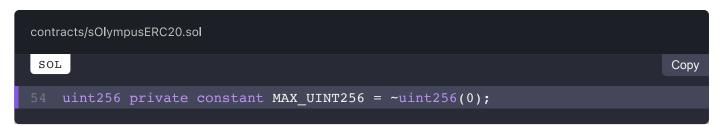
### **OEC-01C: Deprecated Representation Style**

Туре	Severity	Location
Code Style	Informational •	sOlympusERC20.sol:L54

### **Description:**

The maximum of uint256 representation in use (~uint256(0)) has been deprecated in favor of the special type operator.

### **Example:**



#### **Recommendation:**

We advise the operator to be used to instead assign the maximum (type(uint256).max).

#### Alleviation:

The representation of the maximum was adjusted according to our recommendation.

#### **OEC-02C: Inefficient Code Structure**

Туре	Severity	Location
Gas Optimization	Informational •	sOlympusERC20.sol:L174, L175, L185, L186, L191, L192, L198-L203, L210-L213

#### **Description:**

All approve style functions can internally use the approve function.

#### **Example:**

```
contracts/sOlympusERC20.sol
SOL
                                                                              Copy
184 function approve( address spender, uint256 value ) public override returns (b
        allowedValue[ msg.sender ][ spender ] = value;
         emit Approval( msg.sender, spender, value );
        return true;
188 }
190 function increaseAllowance( address spender, uint256 addedValue ) public over
        _allowedValue[ msg.sender ][ spender ] = _allowedValue[ msg.sender ][ spe
        emit Approval( msg.sender, spender, _allowedValue[ msg.sender ][ spender
        return true;
194 }
196 function decreaseAllowance( address spender, uint256 subtractedValue ) public
        uint256 oldValue = allowedValue[ msg.sender ][ spender ];
        if (subtractedValue >= oldValue) {
            allowedValue[ msg.sender ][ spender ] = 0;
        } else {
            allowedValue[ msg.sender ][ spender ] = oldValue.sub( subtractedValu
        emit Approval( msg.sender, spender, _allowedValue[ msg.sender ][ spender
        return true;
205 }
```

```
208
209 // called in a permit
210 function _approve( address owner, address spender, uint256 value ) internal o
211 _allowedValue[owner][spender] = value;
212 emit Approval( owner, spender, value );
213 }
```

#### **Recommendation:**

We advise them to do so to signficantly reduce the bytecode size of the contract.

#### **Alleviation:**

The code was refactored to properly utilize \_approve in all instances possible.

### **OEC-03C: Inexistent Error Messages**

Туре	Severity	Location
Code Style	Informational •	sOlympusERC20.sol:L80, L81, L86, L87, L88, L94, L96

#### **Description:**

The linked require checks contain no descriptive error messages.

#### **Example:**

```
contracts/sOlympusERC20.sol
SOL
                                                                             Copy
   function setIndex( uint _INDEX ) external {
        require( msg.sender == initializer );
       require( INDEX == 0 );
       INDEX = gonsForBalance( INDEX );
   }
   function setgOHM( address _gOHM ) external {
        require( msg.sender == initializer );
       require( address( gOHM ) == address(0) );
       require( gOHM != address(0));
       gOHM = IgOHM( gOHM );
   function initialize( address stakingContract ) external {
        require( msg.sender == initializer );
        require( stakingContract != address(0) );
       stakingContract = stakingContract;
       gonBalances[ stakingContract ] = TOTAL GONS;
       emit Transfer( address(0x0), stakingContract, totalSupply );
       emit LogStakingContractUpdated( stakingContract );
       initializer = address(0);
```

#### **Recommendation:**

We advise them to be set so to aid in the debugging of the application and to also enable more accurate validation of the require conditions.

#### **Alleviation:**

Error messages were introduced in all linked require checks.

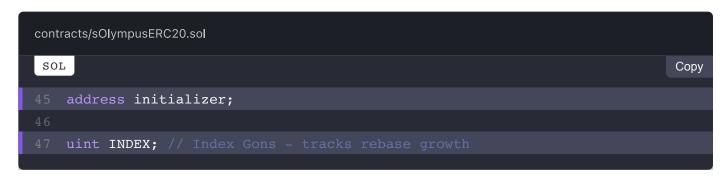
### **OEC-04C: Inexistent Variable Visibility Specifiers**

Туре	Severity	Location
Code Style	Informational •	sOlympusERC20.sol:L45, L47

#### **Description:**

The linked variables have no visibility specifier explicitly set.

#### **Example:**



#### **Recommendation:**

We advise one to be set so to avoid potential compilation discrepancies in the future as the current compiler behaviour is to assign a specifier automatically.

#### **Alleviation:**

The Olympus DAO team considered this exhibit but opted not to apply any remediation for it.

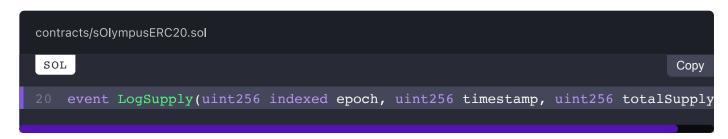
### **OEC-05C: Redundant Event Argument**

Туре	Severity	Location
Code Style	Informational •	sOlympusERC20.sol:L20, L118, L159

#### **Description:**

The current timestamp that a LogSupply event emits is already attached to each event emittance by the blockchain itself.

#### **Example:**

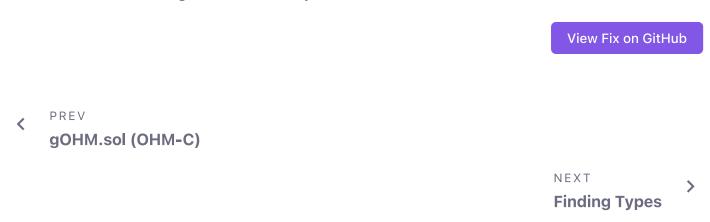


#### **Recommendation:**

We advise the timestamp member to be omitted from the event as it is redundant.

#### Alleviation:

The redundant event argument was safely omitted from the codebase.



ID	Severity	Addressed	Title
GUA-01C	Informational	Yes	Inexistent Deletion of Pending Guardian
MOE-01C	Informational	Yes	Redundant Implementation
OER-01C	Informational	Yes	Incorrect Function Visibility
OTM-01C	Informational	Yes	Inexistent Error Messages
OTM-02C	Informational	Yes	Multiple Top-Level Declarations
OTM-03C	Informational	No	Redundant & Confusing Comparisons
OWN-01C	Informational	Yes	Inexistent Deletion of Pending Owner
SMH-01C	Informational	No	Inefficient Implementation
SDR-01C	Informational	Yes	Inexistent Error Messages
SDR-02C	Informational	Yes	Inexistent Variable Visibility Specifiers
SBC-01C	Informational	Yes	Inexistent Error Message
SBC-02C	Informational	Yes	Inexistent Variable Visibility Specifier
TIM-01C	Informational	Yes	Redundant Implementation
TRE-01C	Informational	Yes	Improper Failure Enforcement
TRE-02C	Informational	Yes	Improper Permitted Execution Flow
TRE-03C	Informational	Yes	Inexistent Error Messages
TRE-04C	Informational	Yes	Inexistent Variable Visibility Specifier
OHM-01C	Informational	Yes	Inexistent Error Messages
OEC-01C	Informational	Yes	Deprecated Representation Style

ID	Severity	Addressed	Title
OEC-02C	Informational	Yes	Inefficient Code Structure
OEC-03C	Informational	Yes	Inexistent Error Messages
OEC-04C	Informational	No	Inexistent Variable Visibility Specifiers
OEC-05C	Informational	Yes	Redundant Event Argument

PREV
Manual Review

BondTeller.sol (BTR-S)

## **Finding Types**

#### ON THIS PAGE

**External Call Validation** 

**Input Sanitization** 

Indeterminate Code

Language Specific

Code Style

**Gas Optimization** 

**Standard Conformity** 

**Mathematical Operations** 

Logical Fault

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 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.



