Guard

Smart contracts security assessment

Final report

Tariff: Standard

AiOMB Protocol

February 2023

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for **AiOMB Protocol**.

The reviewed project is a Tomb Finance fork, allowing users to farm the main AiOMB (AIO) and the share AiShare (GDS) tokens. AIO and GDS tokens are ERC20-standard tokens with taxes on on transfers.

Both rewards pool (a.k.a. farms) contracts may charge a fee of up to 2% for each deposit.

The code is available at the AiOMB/AiOMB Github repo and was audited in the af31957 commit.

Name	AiOMB Protocol
Audit date	2023-02-14 - 2023-02-17
Language	Solidity
Platform	Arbitrum Network

Contracts checked

Name	Address
AiOMB	
AiShare	
ContractGuard	
AiBond	
BoardRoom	
Treasury	
GenesisRewardPool	
ShareRewardPool	
Oracle	

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) of all the issues found by the tools

Manual audit

- Manually analyze smart contracts for security vulnerabilities
- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
Unencrypted Private Data On-Chain	passed
Code With No Effects	passed
Message call with hardcoded gas amount	passed
Typographical Error	passed
DoS With Block Gas Limit	passed
Presence of unused variables	passed
Incorrect Inheritance Order	passed
Requirement Violation	passed
Weak Sources of Randomness from Chain Attributes	passed
Shadowing State Variables	passed

Incorrect Constructor Name	passed
Block values as a proxy for time	passed
Authorization through tx.origin	passed
DoS with Failed Call	passed
Delegatecall to Untrusted Callee	passed
Use of Deprecated Solidity Functions	passed
Assert Violation	passed
State Variable Default Visibility	passed
Reentrancy	passed
Unprotected SELFDESTRUCT Instruction	passed
Unprotected Ether Withdrawal	passed
Unchecked Call Return Value	passed
Floating Pragma	passed
Outdated Compiler Version	passed
Integer Overflow and Underflow	passed
Function Default Visibility	passed

Classification of issue severity

High severity High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.

Medium severity Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.

Low severity	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

No issues were found

Medium severity issues

No issues were found

Low severity issues

1. Lack of events (AiOMB)

Status: Open

There's a general lack of events emitted in governance functions, which complicates tracking the history of changes in crucial system parameters.

2. Unused code (AiOMB)

Status: Open

The SafeMath8 library is not in use. It's also outdated since it should be compiled with pre-0.8 pragma versions to avoid double-spending gas on overflow checks.

3. Public open function (AiOMB)

Status: Open

The `setPairAiShare()` initializer function is open for public use. While the deployed contract is initialized correctly, any possible future code reuse may face difficulties.

4. Gas optimization (AiOMB)

Status: Open

1. There're multiple unnecessary readings of data from blockchain in the `start()` function:

`BUSD.balanceOf(address(this))` and `balanceOf(address(this))` should be read once to local variables.

2. The `mint()` function contains a double call of the contract's balance before and after with an unclear purpose. The internal `_mint()` function inherited from the ERC20 contract has its own safety checks.

3. Unnecessary multiplication by `MULTIPLIER` is performed in the `transfer()` and `transferFrom()` functions. It doesn't improve the accuracy of calculations, the divisor should be set to 100 instead.

5. Parameters of addLiquidity (AiOMB)
Status: Open

There's a `addLiquidity` call for a Uniswap-like router in the `start()` function that may constantly fail since the pair is already created and the amounts are fixed. Also, the deadline parameter is used incorrectly since it can't be calculated on-chain: router contract checks deadline is not smaller than the current time, i.e. setting a deadline to `block.timestamp` or greater will always pass and setting it lower than `block.timestamp` will always revert.

6. Lack of events (AiShare)
Status: Open

There's a general lack of events emitted in governance functions, which complicates tracking the history of changes in crucial system parameters.

7. Gas optimization (AiShare)
Status: Open

1. Unnecessary multiplication by `MULTIPLIER` is performed in the `transfer()` and `transferFrom()` functions. It doesn't improve the accuracy of calculations, the divisor should be set to 100 instead.

8. ContractGuard doesn't prevent re-entrancy (ContractGuard)
Status: Open

The ContractGuard contract is designed to prevent multiple calls in the same block but it doesn't

prevent re-entrancy, i.e. multiple calls in the same transaction.

```
    modifier onlyOneBlock() {
        require(!checkSameOriginReentranted(), "ContractGuard: one block, one
function");
        require(!checkSameSenderReentranted(), "ContractGuard: one block, one
function");

        _;

        _status[block.number][tx.origin] = true;
        _status[block.number][msg.sender] = true;
    }
```

9. Typos (Treasury)
Status: Open

Typos in 'upto'.

10. Gas optimization (Treasury)
Status: Open

1. Checking the `targetPrice` from parameters of the `redeemBonds()` function seems unnecessary: the gas-wise way is to receive a `bondRate` parameter and check it against `getBondPremiumRate()` directly.

2. Excessive data is read from the blockchain in the `redeemBonds()` function: `getBondPremiumRate()` should receive already in-memory values of `nativePrice` and `nativePriceCeiling`.

3. Redundant code in the `redeemBonds()` function: `require(_rate > 0)` is always passed as it's already checked that `nativePrice > nativePriceCeiling`.

11. Validation in the initialize() function (Treasury)
Status: Open

The input parameters of the `initialize()` function aren't checked in any way. The `nativePriceOne` variable is set to 1e18 regardless of the actual `decimals()` value of the native

token.

12. Gas optimization (GenesisRewardPool)
Status: Open

Pool duplication check is ineffective, it should be performed via mapping from the token address. The other way is to allow duplicated pools by storing individual pools balances in `PoolInfo` structure, i.e. the `updatePool()` function should not check the `pool.token.balanceOf(address(this))` but read the pool balance from the structure.

13. Contract doesn't support tokens with transfer fees (GenesisRewardPool)
Status: Open

Actual transfer amounts aren't checked so the owner must not add pools with tokens with transfer commissions unless this contract is excluded from such fees (see AiOMB and AiShare contracts).

14. Possible block gas limit problem (GenesisRewardPool)
Status: Open

Actual transfer amounts aren't checked so the owner must not add pools with tokens with transfer commissions unless this contract is excluded from such fees (see AiOMB and AiShare contracts).

15. Gas optimization (ShareRewardPool)
Status: Open

Pool duplication check is ineffective, it should be performed via mapping from the token address. The other way is to allow duplicated pools by storing individual pools balances in `PoolInfo` structure, i.e. the `updatePool()` function should not check the `pool.token.balanceOf(address(this))` but read the pool balance from the structure.

16. Contract doesn't support tokens with transfer fees (ShareRewardPool)
Status: Open

Actual transfer amounts aren't checked so the owner must not add pools with tokens with transfer commissions unless this contract is excluded from such fees (see AiOMB and AiShare contracts).

17. Possible block gas limit problem (ShareRewardPool)
Status: Open

Actual transfer amounts aren't checked so the owner must not add pools with tokens with transfer

commissions unless this contract is excluded from such fees (see AiOMB and AiShare contracts).

18. Gas optimization (Oracle)
Status: Open

The `getCurrentEpoch()` function and `epoch` variable of the Epoch contract have unclear functionality.

19. Typos (Oracle)
Status: Open

Typos in 'substraction'.

Conclusion

AiOMB Protocol AiOMB, AiShare, ContractGuard, AiBond, BoardRoom, Treasury, GenesisRewardPool, ShareRewardPool, Oracle contracts were audited. 19 low severity issues were found.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without 0xGuard prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts 0xGuard to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.
OxGuard retains exclusive publishing rights for the results of this audit on its website and social networks.

Static code analysis

```
UniswapV2OracleLibrary.currentBlockTimestamp() (contracts/Oracle.sol#407-409) uses a
weak PRNG: "uint32(block.timestamp % 2 ** 32) (contracts/Oracle.sol#408)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Treasury._sendToBoardroom(uint256) (contracts/Treasury.sol#462-485) ignores return
value by IERC20(native).transfer(daoFund,_daoFundSharedAmount) (contracts/
Treasury.sol#468)
Treasury._sendToBoardroom(uint256) (contracts/Treasury.sol#462-485) ignores return
value by IERC20(native).transfer(devFund,_devFundSharedAmount) (contracts/
Treasury.sol#475)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-
transfer

GenesisRewardPool.pending(uint256,address) (contracts/GenesisRewardPool.sol#147-158)
performs a multiplication on the result of a division:
        - _reward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint)
(contracts/GenesisRewardPool.sol#154)
        - accTokenPerShare = accTokenPerShare.add(_reward.mul(1e18).div(tokenSupply))
(contracts/GenesisRewardPool.sol#155)
GenesisRewardPool.updatePool(uint256) (contracts/GenesisRewardPool.sol#169-189)
performs a multiplication on the result of a division:
        - _reward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint)
(contracts/GenesisRewardPool.sol#185)
        - pool.accTokenPerShare =
pool.accTokenPerShare.add(_reward.mul(1e18).div(tokenSupply)) (contracts/
GenesisRewardPool.sol#186)
ShareRewardPool.pendingShare(uint256,address) (contracts/ShareRewardPool.sol#169-180)
performs a multiplication on the result of a division:
        - _sharesReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint)
(contracts/ShareRewardPool.sol#176)
        - accTokensPerShare =
accTokensPerShare.add(_sharesReward.mul(1e18).div(tokenSupply)) (contracts/
ShareRewardPool.sol#177)
ShareRewardPool.updatePool(uint256) (contracts/ShareRewardPool.sol#191-211) performs a
multiplication on the result of a division:
        - _sharesReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint)
(contracts/ShareRewardPool.sol#207)
```

```
        - pool.accTokensPerShare =
pool.accTokensPerShare.add(_sharesReward.mul(1e18).div(tokenSupply)) (contracts/
ShareRewardPool.sol#208)
Treasury.allocateSeigniorage() (contracts/Treasury.sol#497-537) performs a
multiplication on the result of a division:
        - _seigniorage = nativeSupply.mul(_percentage).div(1e18) (contracts/
Treasury.sol#520)
        - _savedForBoardroom =
_seigniorage.mul(seigniorageExpansionFloorPercent).div(10000) (contracts/
Treasury.sol#521)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply

GenesisRewardPool.updatePool(uint256) (contracts/GenesisRewardPool.sol#169-189) uses a
dangerous strict equality:
        - tokenSupply == 0 (contracts/GenesisRewardPool.sol#175)
ShareRewardPool.updatePool(uint256) (contracts/ShareRewardPool.sol#191-211) uses a
dangerous strict equality:
        - tokenSupply == 0 (contracts/ShareRewardPool.sol#197)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-
strict-equalities

Reentrancy in Treasury.buyBonds(uint256,uint256) (contracts/Treasury.sol#407-434):
        External calls:
        - IBasisAsset(native).burnFrom(msg.sender,_nativeAmount) (contracts/
Treasury.sol#427)
        - IBasisAsset(bond).mint(msg.sender,_bondAmount) (contracts/Treasury.sol#428)
        State variables written after the call(s):
        - epochSupplyContractionLeft = epochSupplyContractionLeft.sub(_nativeAmount)
(contracts/Treasury.sol#430)
        Treasury.epochSupplyContractionLeft (contracts/Treasury.sol#40) can be used in
cross function reentrancies:
        - Treasury.buyBonds(uint256,uint256) (contracts/Treasury.sol#407-434)
        - Treasury.checkEpoch() (contracts/Treasury.sol#112-119)
        - Treasury.epochSupplyContractionLeft (contracts/Treasury.sol#40)
        - Treasury.getBurnableNativeLeft() (contracts/Treasury.sol#172-184)
Reentrancy in AiOMB.start() (contracts/AiOMB.sol#68-74):
        External calls:
        - BUSD.approve(address(uniswapV2Router),BUSD.balanceOf(address(this)))
(contracts/AiOMB.sol#71)
        - uniswapV2Router.addLiquidity(address(this),address(BUSD),balanceOf(address(thi
```

```
s)),BUSD.balanceOf(address(this)),balanceOf(address(this)),BUSD.balanceOf(address(this))
,msg.sender,block.timestamp + 60) (contracts/AiOMB.sol#72)
        State variables written after the call(s):
        - started = true (contracts/AiOMB.sol#73)
        AiOMB.started (contracts/AiOMB.sol#37) can be used in cross function
reentrancies:
        - AiOMB.start() (contracts/AiOMB.sol#68-74)
        - AiOMB.started (contracts/AiOMB.sol#37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1

Treasury.setSupplyTiersEntry(uint8,uint256) (contracts/Treasury.sol#303-314) contains a
tautology or contradiction:
        - require(bool,string)(_index >= 0,Index has to be higher than 0) (contracts/
Treasury.sol#304)
Treasury.setMaxExpansionTiersEntry(uint8,uint256) (contracts/Treasury.sol#316-322)
contains a tautology or contradiction:
        - require(bool,string)(_index >= 0,Index has to be higher than 0) (contracts/
Treasury.sol#317)
Treasury._calculateMaxSupplyExpansionPercent(uint256) (contracts/Treasury.sol#487-495)
contains a tautology or contradiction:
        - tierId >= 0 (contracts/Treasury.sol#488)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-
contradiction

Treasury.getNativePrice().price (contracts/Treasury.sol#152) is a local variable never
initialized
Treasury.allocateSeigniorage()._savedForBond (contracts/Treasury.sol#509) is a local
variable never initialized
Treasury.getNativeUpdatedPrice().price (contracts/Treasury.sol#160) is a local variable
never initialized
AiOMB._getPrice()._price (contracts/AiOMB.sol#119) is a local variable never
initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-
local-variables

FixedPoint.mul(FixedPoint.uq112x112,uint256).z (contracts/Oracle.sol#275) is a local
variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-
local-variables
```

```
AiOMB.start() (contracts/AiOMB.sol#68-74) ignores return value by
BUSD.approve(address(uniswapV2Router),BUSD.balanceOf(address(this))) (contracts/
AiOMB.sol#71)
AiOMB.start() (contracts/AiOMB.sol#68-74) ignores return value by uniswapV2Router.addLiq
uidity(address(this),address(BUSD),balanceOf(address(this)),BUSD.balanceOf(address(this)
),balanceOf(address(this)),BUSD.balanceOf(address(this)),msg.sender,block.timestamp +
60) (contracts/AiOMB.sol#72)
AiOMB._getPrice() (contracts/AiOMB.sol#118-124) ignores return value by
IOracle(oracle).consult(address(this),1e18) (contracts/AiOMB.sol#119-123)
Treasury.getNativePrice() (contracts/Treasury.sol#151-157) ignores return value by
IOracle(nativeOracle).consult(native,1e18) (contracts/Treasury.sol#152-156)
Treasury.getNativeUpdatedPrice() (contracts/Treasury.sol#159-165) ignores return value
by IOracle(nativeOracle).twap(native,1e18) (contracts/Treasury.sol#160-164)
Treasury.buyBonds(uint256,uint256) (contracts/Treasury.sol#407-434) ignores return
value by IBasisAsset(bond).mint(msg.sender,_bondAmount) (contracts/Treasury.sol#428)
Treasury._sendToBoardroom(uint256) (contracts/Treasury.sol#462-485) ignores return
value by IBasisAsset(native).mint(address(this),_amount) (contracts/Treasury.sol#463)
Treasury.allocateSeigniorage() (contracts/Treasury.sol#497-537) ignores return value by
IBasisAsset(native).mint(address(this),_savedForBond) (contracts/Treasury.sol#532)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

AiShare.setAdmin(address) (contracts/AiShare.sol#156-158) should emit an event for:
        - admin = _admin (contracts/AiShare.sol#157)
Boardroom.setOperator(address) (contracts/Boardroom.sol#105-107) should emit an event
for:
        - operator = _operator (contracts/Boardroom.sol#106)
GenesisRewardPool.setOperator(address) (contracts/GenesisRewardPool.sol#269-271) should
emit an event for:
        - operator = _operator (contracts/GenesisRewardPool.sol#270)
ShareRewardPool.setOperator(address) (contracts/ShareRewardPool.sol#295-297) should
emit an event for:
        - operator = _operator (contracts/ShareRewardPool.sol#296)
Treasury.setOperator(address) (contracts/Treasury.sol#281-283) should emit an event
for:
        - operator = _operator (contracts/Treasury.sol#282)
Treasury.setBoardroom(address) (contracts/Treasury.sol#285-287) should emit an event
for:
        - boardroom = _boardroom (contracts/Treasury.sol#286)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
access-control
```

```
Boardroom.setLockUp(uint256,uint256) (contracts/Boardroom.sol#109-113) should emit an
event for:
        - withdrawLockupEpochs = _withdrawLockupEpochs (contracts/Boardroom.sol#111)
        - rewardLockupEpochs = _rewardLockupEpochs (contracts/Boardroom.sol#112)
GenesisRewardPool.add(uint256,IERC20,bool,uint256,uint16) (contracts/
GenesisRewardPool.sol#92-118) should emit an event for:
        - totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/
GenesisRewardPool.sol#116)
GenesisRewardPool.set(uint256,uint256,uint16) (contracts/GenesisRewardPool.sol#121-130)
should emit an event for:
        - totalAllocPoint = totalAllocPoint.sub(pool.allocPoint).add(_allocPoint)
(contracts/GenesisRewardPool.sol#126)
ShareRewardPool.add(uint256,IERC20,bool,uint256,uint16) (contracts/
ShareRewardPool.sol#98-139) should emit an event for:
        - totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/
ShareRewardPool.sol#137)
ShareRewardPool.set(uint256,uint256,uint16) (contracts/ShareRewardPool.sol#142-152)
should emit an event for:
        - totalAllocPoint = totalAllocPoint.sub(pool.allocPoint).add(_allocPoint)
(contracts/ShareRewardPool.sol#146-148)
Treasury.setNativePriceCeiling(uint256) (contracts/Treasury.sol#293-296) should emit an
event for:
        - nativePriceCeiling = _nativePriceCeiling (contracts/Treasury.sol#295)
Treasury.setMaxSupplyExpansionPercents(uint256) (contracts/Treasury.sol#298-301) should
emit an event for:
        - maxSupplyExpansionPercent = _maxSupplyExpansionPercent (contracts/
Treasury.sol#300)
Treasury.setBondDepletionFloorPercent(uint256) (contracts/Treasury.sol#324-327) should
emit an event for:
        - bondDepletionFloorPercent = _bondDepletionFloorPercent (contracts/
Treasury.sol#326)
Treasury.setMaxDebtRatioPercent(uint256) (contracts/Treasury.sol#334-337) should emit
an event for:
        - maxDebtRatioPercent = _maxDebtRatioPercent (contracts/Treasury.sol#336)
Treasury.setBootstrap(uint256,uint256) (contracts/Treasury.sol#339-344) should emit an
event for:
        - bootstrapEpochs = _bootstrapEpochs (contracts/Treasury.sol#342)
        - bootstrapSupplyExpansionPercent = _bootstrapSupplyExpansionPercent (contracts/
Treasury.sol#343)
Treasury.setExtraFunds(address,uint256,address,uint256) (contracts/
Treasury.sol#346-360) should emit an event for:
```

```
        - daoFundSharedPercent = _daoFundSharedPercent (contracts/Treasury.sol#357)
        - devFundSharedPercent = _devFundSharedPercent (contracts/Treasury.sol#359)
Treasury.setMaxDiscountRate(uint256) (contracts/Treasury.sol#362-364) should emit an
event for:
        - maxDiscountRate = _maxDiscountRate (contracts/Treasury.sol#363)
Treasury.setMaxPremiumRate(uint256) (contracts/Treasury.sol#366-368) should emit an
event for:
        - maxPremiumRate = _maxPremiumRate (contracts/Treasury.sol#367)
Treasury.setDiscountPercent(uint256) (contracts/Treasury.sol#370-373) should emit an
event for:
        - discountPercent = _discountPercent (contracts/Treasury.sol#372)
Treasury.setPremiumThreshold(uint256) (contracts/Treasury.sol#375-379) should emit an
event for:
        - premiumThreshold = _premiumThreshold (contracts/Treasury.sol#378)
Treasury.setPremiumPercent(uint256) (contracts/Treasury.sol#381-384) should emit an
event for:
        - premiumPercent = _premiumPercent (contracts/Treasury.sol#383)
Treasury.setMintingFactorForPayingDebt(uint256) (contracts/Treasury.sol#386-389) should
emit an event for:
        - mintingFactorForPayingDebt = _mintingFactorForPayingDebt (contracts/
Treasury.sol#388)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic

AiOMB.constructor(address,address,address,address,address,address,address,uint256)._BOND
 (contracts/AiOMB.sol#76) lacks a zero-check on :
                - BOND = _BOND (contracts/AiOMB.sol#96)
AiOMB.constructor(address,address,address,address,address,address,address,uint256)._gene
sisAddress (contracts/AiOMB.sol#76) lacks a zero-check on :
                - genesisAddress = _genesisAddress (contracts/AiOMB.sol#97)
AiOMB.constructor(address,address,address,address,address,address,address,uint256)._trea
sury (contracts/AiOMB.sol#76) lacks a zero-check on :
                - treasury = _treasury (contracts/AiOMB.sol#98)
AiOMB.constructor(address,address,address,address,address,address,address,uint256)._boar
droom (contracts/AiOMB.sol#76) lacks a zero-check on :
                - boardroom = _boardroom (contracts/AiOMB.sol#99)
AiOMB.constructor(address,address,address,address,address,address,address,uint256)._shar
eRewardPool (contracts/AiOMB.sol#76) lacks a zero-check on :
                - shareRewardPool = _shareRewardPool (contracts/AiOMB.sol#100)
AiOMB.setPairAiShare(address)._pairAiShare (contracts/AiOMB.sol#145) lacks a zero-check
on :
```

```
                - PairAiShare = _pairAiShare (contracts/AiOMB.sol#147)
AiOMB.setAdmin(address)._admin (contracts/AiOMB.sol#150) lacks a zero-check on :
                - admin = _admin (contracts/AiOMB.sol#151)
AiShare.constructor(address,address,address,address,address,address,address,address,uint
256,address,address)._BOND (contracts/AiShare.sol#81) lacks a zero-check on :
                - BOND = _BOND (contracts/AiShare.sol#83)
AiShare.constructor(address,address,address,address,address,address,address,address,uint
256,address,address)._BUSD (contracts/AiShare.sol#81) lacks a zero-check on :
                - BUSD = _BUSD (contracts/AiShare.sol#85)
AiShare.constructor(address,address,address,address,address,address,address,address,uint
256,address,address)._genesisAddress (contracts/AiShare.sol#81) lacks a zero-check on :
                - genesisAddress = _genesisAddress (contracts/AiShare.sol#101)
AiShare.constructor(address,address,address,address,address,address,address,address,uint
256,address,address)._treasury (contracts/AiShare.sol#81) lacks a zero-check on :
                - treasury = _treasury (contracts/AiShare.sol#102)
AiShare.constructor(address,address,address,address,address,address,address,address,uint
256,address,address)._boardroom (contracts/AiShare.sol#81) lacks a zero-check on :
                - boardroom = _boardroom (contracts/AiShare.sol#103)
AiShare.constructor(address,address,address,address,address,address,address,address,uint
256,address,address)._shareRewardPool (contracts/AiShare.sol#81) lacks a zero-check
on :
                - shareRewardPool = _shareRewardPool (contracts/AiShare.sol#104)
AiShare.setAdmin(address)._admin (contracts/AiShare.sol#156) lacks a zero-check on :
                - admin = _admin (contracts/AiShare.sol#157)
Boardroom.setOperator(address)._operator (contracts/Boardroom.sol#105) lacks a zero-
check on :
                - operator = _operator (contracts/Boardroom.sol#106)
GenesisRewardPool.setOperator(address)._operator (contracts/GenesisRewardPool.sol#269)
lacks a zero-check on :
                - operator = _operator (contracts/GenesisRewardPool.sol#270)
ShareRewardPool.setFeeAddress(address)._feeAddress (contracts/ShareRewardPool.sol#290)
lacks a zero-check on :
                - feeAddress = _feeAddress (contracts/ShareRewardPool.sol#291)
ShareRewardPool.setOperator(address)._operator (contracts/ShareRewardPool.sol#295)
lacks a zero-check on :
                - operator = _operator (contracts/ShareRewardPool.sol#296)
Treasury.initialize(address,address,address,address,address,address,uint256)._native
(contracts/Treasury.sol#235) lacks a zero-check on :
                - native = _native (contracts/Treasury.sol#243)
Treasury.initialize(address,address,address,address,address,address,uint256)._bond
(contracts/Treasury.sol#236) lacks a zero-check on :
```

```
                - bond = _bond (contracts/Treasury.sol#244)
Treasury.initialize(address,address,address,address,address,address,uint256)._share
(contracts/Treasury.sol#237) lacks a zero-check on :
                - share = _share (contracts/Treasury.sol#245)
Treasury.initialize(address,address,address,address,address,address,uint256)._nativeOrac
le (contracts/Treasury.sol#238) lacks a zero-check on :
                - nativeOracle = _nativeOracle (contracts/Treasury.sol#246)
Treasury.initialize(address,address,address,address,address,address,uint256)._boardroom
(contracts/Treasury.sol#239) lacks a zero-check on :
                - boardroom = _boardroom (contracts/Treasury.sol#247)
Treasury.setOperator(address)._operator (contracts/Treasury.sol#281) lacks a zero-check
on :
                - operator = _operator (contracts/Treasury.sol#282)
Treasury.setBoardroom(address)._boardroom (contracts/Treasury.sol#285) lacks a zero-
check on :
                - boardroom = _boardroom (contracts/Treasury.sol#286)
Treasury.setNativeOracle(address)._nativeOracle (contracts/Treasury.sol#289) lacks a
zero-check on :
                - nativeOracle = _nativeOracle (contracts/Treasury.sol#290)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
address-validation

GenesisRewardPool.updatePool(uint256) (contracts/GenesisRewardPool.sol#169-189) has
external calls inside a loop: tokenSupply = pool.token.balanceOf(address(this))
(contracts/GenesisRewardPool.sol#174)
ShareRewardPool.updatePool(uint256) (contracts/ShareRewardPool.sol#191-211) has
external calls inside a loop: tokenSupply = pool.token.balanceOf(address(this))
(contracts/ShareRewardPool.sol#196)
Treasury.getNativeCirculatingSupply() (contracts/Treasury.sol#397-405) has external
calls inside a loop: balanceExcluded =
balanceExcluded.add(nativeErc20.balanceOf(excludedFromTotalSupply[entryId])) (contracts/
Treasury.sol#402)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-
a-loop

Variable 'AiOMB._getPrice()._price (contracts/AiOMB.sol#119)' in AiOMB._getPrice()
(contracts/AiOMB.sol#118-124) potentially used before declaration: uint256(_price)
(contracts/AiOMB.sol#120)
Variable 'Treasury.getNativePrice().price (contracts/Treasury.sol#152)' in
Treasury.getNativePrice() (contracts/Treasury.sol#151-157) potentially used before
declaration: uint256(price) (contracts/Treasury.sol#153)
```

```
Variable 'Treasury.getNativeUpdatedPrice().price (contracts/Treasury.sol#160)' in
Treasury.getNativeUpdatedPrice() (contracts/Treasury.sol#159-165) potentially used
before declaration: uint256(price) (contracts/Treasury.sol#161)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-
declaration-usage-of-local-variables

Reentrancy in Treasury.allocateSeigniorage() (contracts/Treasury.sol#497-537):
        External calls:
        - _updateNativePrice() (contracts/Treasury.sol#498)
                - IOracle(nativeOracle).update() (contracts/Treasury.sol#394)
        State variables written after the call(s):
        - _mse = _calculateMaxSupplyExpansionPercent(nativeSupply).mul(1e14) (contracts/
Treasury.sol#511)
                - maxSupplyExpansionPercent = maxExpansionTiers[tierId] (contracts/
Treasury.sol#490)
        - previousEpochNativePrice = getNativePrice() (contracts/Treasury.sol#499)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2

Reentrancy in Treasury._sendToBoardroom(uint256) (contracts/Treasury.sol#462-485):
        External calls:
        - IBasisAsset(native).mint(address(this),_amount) (contracts/Treasury.sol#463)
        - IERC20(native).transfer(daoFund,_daoFundSharedAmount) (contracts/
Treasury.sol#468)
        Event emitted after the call(s):
        - DaoFundFunded(block.timestamp,_daoFundSharedAmount) (contracts/
Treasury.sol#469)
Reentrancy in Treasury._sendToBoardroom(uint256) (contracts/Treasury.sol#462-485):
        External calls:
        - IBasisAsset(native).mint(address(this),_amount) (contracts/Treasury.sol#463)
        - IERC20(native).transfer(daoFund,_daoFundSharedAmount) (contracts/
Treasury.sol#468)
        - IERC20(native).transfer(devFund,_devFundSharedAmount) (contracts/
Treasury.sol#475)
        Event emitted after the call(s):
        - DevFundFunded(block.timestamp,_devFundSharedAmount) (contracts/
Treasury.sol#476)
Reentrancy in Treasury._sendToBoardroom(uint256) (contracts/Treasury.sol#462-485):
        External calls:
        - IBasisAsset(native).mint(address(this),_amount) (contracts/Treasury.sol#463)
        - IERC20(native).transfer(daoFund,_daoFundSharedAmount) (contracts/
Treasury.sol#468)
```

```
        - IERC20(native).transfer(devFund,_devFundSharedAmount) (contracts/
Treasury.sol#475)
        - IERC20(native).safeApprove(boardroom,0) (contracts/Treasury.sol#481)
        - IERC20(native).safeApprove(boardroom,_amount) (contracts/Treasury.sol#482)
        - IBoardroom(boardroom).allocateSeigniorage(_amount) (contracts/
Treasury.sol#483)
        Event emitted after the call(s):
        - BoardroomFunded(block.timestamp,_amount) (contracts/Treasury.sol#484)
Reentrancy in Boardroom.allocateSeigniorage(uint256) (contracts/Boardroom.sol#200-213):
        External calls:
        - native.safeTransferFrom(msg.sender,address(this),amount) (contracts/
Boardroom.sol#211)
        Event emitted after the call(s):
        - RewardAdded(msg.sender,amount) (contracts/Boardroom.sol#212)
Reentrancy in Treasury.buyBonds(uint256,uint256) (contracts/Treasury.sol#407-434):
        External calls:
        - IBasisAsset(native).burnFrom(msg.sender,_nativeAmount) (contracts/
Treasury.sol#427)
        - IBasisAsset(bond).mint(msg.sender,_bondAmount) (contracts/Treasury.sol#428)
        - _updateNativePrice() (contracts/Treasury.sol#431)
                - IOracle(nativeOracle).update() (contracts/Treasury.sol#394)
        Event emitted after the call(s):
        - BoughtBonds(msg.sender,_nativeAmount,_bondAmount) (contracts/
Treasury.sol#433)
Reentrancy in Boardroom.claimReward() (contracts/Boardroom.sol#189-198):
        External calls:
        - native.safeTransfer(msg.sender,reward) (contracts/Boardroom.sol#195)
        Event emitted after the call(s):
        - RewardPaid(msg.sender,reward) (contracts/Boardroom.sol#196)
Reentrancy in GenesisRewardPool.emergencyWithdraw(uint256) (contracts/
GenesisRewardPool.sol#241-249):
        External calls:
        - pool.token.safeTransfer(msg.sender,_amount) (contracts/
GenesisRewardPool.sol#247)
        Event emitted after the call(s):
        - EmergencyWithdraw(msg.sender,_pid,_amount) (contracts/
GenesisRewardPool.sol#248)
Reentrancy in ShareRewardPool.emergencyWithdraw(uint256) (contracts/
ShareRewardPool.sol#268-276):
        External calls:
        - pool.token.safeTransfer(msg.sender,_amount) (contracts/
ShareRewardPool.sol#274)
```

```
        Event emitted after the call(s):
        - EmergencyWithdraw(msg.sender,_pid,_amount) (contracts/
ShareRewardPool.sol#275)
Reentrancy in Treasury.redeemBonds(uint256,uint256) (contracts/Treasury.sol#436-460):
        External calls:
        - IBasisAsset(bond).burnFrom(msg.sender,_bondAmount) (contracts/
Treasury.sol#454)
        - IERC20(native).safeTransfer(msg.sender,_nativeAmount) (contracts/
Treasury.sol#455)
        - _updateNativePrice() (contracts/Treasury.sol#457)
                - IOracle(nativeOracle).update() (contracts/Treasury.sol#394)
        Event emitted after the call(s):
        - RedeemedBonds(msg.sender,_nativeAmount,_bondAmount) (contracts/
Treasury.sol#459)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3

AiOMB.setTradingStartTime(uint256) (contracts/AiOMB.sol#218-221) uses timestamp for
comparisons
        Dangerous comparisons:
        - require(bool,string)(tradingStartTime > block.timestamp,Trading has already
stared.) (contracts/AiOMB.sol#219)
AiOMB._beforeTokenTransfer(address,address,uint256) (contracts/AiOMB.sol#223-227) uses
timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(tradingStartTime <= block.timestamp,Trading hasn't
started yet.) (contracts/AiOMB.sol#226)
AiShare.unclaimedTreasuryFund() (contracts/AiShare.sol#183-188) uses timestamp for
comparisons
        Dangerous comparisons:
        - _now > endTime (contracts/AiShare.sol#185)
        - communityFundLastClaimed >= _now (contracts/AiShare.sol#186)
AiShare.unclaimedDevFund() (contracts/AiShare.sol#190-195) uses timestamp for
comparisons
        Dangerous comparisons:
        - _now > endTime (contracts/AiShare.sol#192)
        - devFundLastClaimed >= _now (contracts/AiShare.sol#193)
GenesisRewardPool.init(address,uint256) (contracts/GenesisRewardPool.sol#63-77) uses
timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp < _poolStartTime,late) (contracts/
GenesisRewardPool.sol#64)
```

```
GenesisRewardPool.checkPoolDuplicate(IERC20) (contracts/GenesisRewardPool.sol#84-89)
uses timestamp for comparisons
        Dangerous comparisons:
        - pid < length (contracts/GenesisRewardPool.sol#86)
        - require(bool,string)(poolInfo[pid].token != _token,GenesisPool: existing
pool?) (contracts/GenesisRewardPool.sol#87)
GenesisRewardPool.add(uint256,IERC20,bool,uint256,uint16) (contracts/
GenesisRewardPool.sol#92-118) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp < poolStartTime (contracts/GenesisRewardPool.sol#98)
        - _lastRewardTime == 0 (contracts/GenesisRewardPool.sol#100)
        - _lastRewardTime < poolStartTime (contracts/GenesisRewardPool.sol#103)
        - _lastRewardTime == 0 || _lastRewardTime < block.timestamp (contracts/
GenesisRewardPool.sol#109)
        - _isStarted = (_lastRewardTime <= poolStartTime) || (_lastRewardTime <=
block.timestamp) (contracts/GenesisRewardPool.sol#113)
GenesisRewardPool.getGeneratedReward(uint256,uint256) (contracts/
GenesisRewardPool.sol#133-144) uses timestamp for comparisons
        Dangerous comparisons:
        - _fromTime >= _toTime (contracts/GenesisRewardPool.sol#134)
        - _toTime >= poolEndTime (contracts/GenesisRewardPool.sol#135)
        - _toTime <= poolStartTime (contracts/GenesisRewardPool.sol#140)
GenesisRewardPool.pending(uint256,address) (contracts/GenesisRewardPool.sol#147-158)
uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > pool.lastRewardTime && tokenSupply != 0 (contracts/
GenesisRewardPool.sol#152)
GenesisRewardPool.massUpdatePools() (contracts/GenesisRewardPool.sol#161-166) uses
timestamp for comparisons
        Dangerous comparisons:
        - pid < length (contracts/GenesisRewardPool.sol#163)
GenesisRewardPool.updatePool(uint256) (contracts/GenesisRewardPool.sol#169-189) uses
timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp <= pool.lastRewardTime (contracts/GenesisRewardPool.sol#171)
ShareRewardPool.init(address,uint256) (contracts/ShareRewardPool.sol#69-82) uses
timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp < _poolStartTime,late) (contracts/
ShareRewardPool.sol#70)
ShareRewardPool.checkPoolDuplicate(IERC20) (contracts/ShareRewardPool.sol#90-95) uses
```

```
timestamp for comparisons
        Dangerous comparisons:
        - pid < length (contracts/ShareRewardPool.sol#92)
        - require(bool,string)(poolInfo[pid].token != _token,ShareRewardPool: existing
pool?) (contracts/ShareRewardPool.sol#93)
ShareRewardPool.add(uint256,IERC20,bool,uint256,uint16) (contracts/
ShareRewardPool.sol#98-139) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp < poolStartTime (contracts/ShareRewardPool.sol#110)
        - _lastRewardTime == 0 (contracts/ShareRewardPool.sol#112)
        - _lastRewardTime < poolStartTime (contracts/ShareRewardPool.sol#115)
        - _lastRewardTime == 0 || _lastRewardTime < block.timestamp (contracts/
ShareRewardPool.sol#121)
        - _isStarted = (_lastRewardTime <= poolStartTime) || (_lastRewardTime <=
block.timestamp) (contracts/ShareRewardPool.sol#125-127)
ShareRewardPool.getGeneratedReward(uint256,uint256) (contracts/
ShareRewardPool.sol#155-166) uses timestamp for comparisons
        Dangerous comparisons:
        - _fromTime >= _toTime (contracts/ShareRewardPool.sol#156)
        - _toTime >= poolEndTime (contracts/ShareRewardPool.sol#157)
        - _toTime <= poolStartTime (contracts/ShareRewardPool.sol#162)
ShareRewardPool.pendingShare(uint256,address) (contracts/ShareRewardPool.sol#169-180)
uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > pool.lastRewardTime && tokenSupply != 0 (contracts/
ShareRewardPool.sol#174)
ShareRewardPool.massUpdatePools() (contracts/ShareRewardPool.sol#183-188) uses
timestamp for comparisons
        Dangerous comparisons:
        - pid < length (contracts/ShareRewardPool.sol#185)
ShareRewardPool.updatePool(uint256) (contracts/ShareRewardPool.sol#191-211) uses
timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp <= pool.lastRewardTime (contracts/ShareRewardPool.sol#193)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp

UniswapV2OracleLibrary.currentCumulativePrices(address) (contracts/Oracle.sol#412-436)
uses timestamp for comparisons
        Dangerous comparisons:
        - blockTimestampLast != blockTimestamp (contracts/Oracle.sol#427)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp

AiOMB.isContract(address) (contracts/AiOMB.sol#126-132) uses assembly
        - INLINE ASM (contracts/AiOMB.sol#128-130)
AiShare.isContract(address) (contracts/AiShare.sol#137-143) uses assembly
        - INLINE ASM (contracts/AiShare.sol#139-141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

AiOMB.setWhiteList(address) (contracts/AiOMB.sol#135-142) compares to a boolean
constant:
        -require(bool,string)(isContract(_WhiteList) == true,only contracts can be
whitelisted) (contracts/AiOMB.sol#136)
AiOMB.transferFrom(address,address,uint256) (contracts/AiOMB.sol#186-199) compares to a
boolean constant:
        -whitelist[sender] == true || whitelist[recipient] == true (contracts/
AiOMB.sol#187)
AiOMB.transfer(address,uint256) (contracts/AiOMB.sol#201-216) compares to a boolean
constant:
        -whitelist[_msgSender()] == true || whitelist[recipient] == true (contracts/
AiOMB.sol#202)
AiShare.setWhiteList(address) (contracts/AiShare.sol#146-154) compares to a boolean
constant:
        -require(bool,string)(isContract(_WhiteList) == true,only contracts can be
whitelisted) (contracts/AiShare.sol#147)
AiShare.transferFrom(address,address,uint256) (contracts/AiShare.sol#213-230) compares
to a boolean constant:
        -whitelist[sender] == true || whitelist[recipient] == true (contracts/
AiShare.sol#214)
AiShare.transfer(address,uint256) (contracts/AiShare.sol#232-248) compares to a boolean
constant:
        -whitelist[_msgSender()] == true || whitelist[recipient] == true (contracts/
AiShare.sol#233)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-
equality

Different versions of Solidity are used:
        - Version used: ['0.6.12', '>=0.6.0<0.8.0', '^0.6.0']
        - 0.6.12 (contracts/Oracle.sol#530)
        - 0.6.12 (contracts/Oracle.sol#650)
        - >=0.6.0<0.8.0 (contracts/Oracle.sol#3)
```

```
        - >=0.6.0<0.8.0 (contracts/Oracle.sol#439)
        - >=0.6.0<0.8.0 (contracts/Oracle.sol#463)
        - ^0.6.0 (contracts/Oracle.sol#217)
        - ^0.6.0 (contracts/Oracle.sol#236)
        - ^0.6.0 (contracts/Oracle.sol#309)
        - ^0.6.0 (contracts/Oracle.sol#399)
        - ^0.6.0 (contracts/Oracle.sol#568)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-
pragma-directives-are-used

GenesisRewardPool.updatePool(uint256) (contracts/GenesisRewardPool.sol#169-189) has
costly operations inside a loop:
        - totalAllocPoint = totalAllocPoint.add(pool.allocPoint) (contracts/
GenesisRewardPool.sol#181)
ShareRewardPool.updatePool(uint256) (contracts/ShareRewardPool.sol#191-211) has costly
operations inside a loop:
        - totalAllocPoint = totalAllocPoint.add(pool.allocPoint) (contracts/
ShareRewardPool.sol#203)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-
operations-inside-a-loop

AiOMB._getPrice() (contracts/AiOMB.sol#118-124) is never used and should be removed
Babylonian.sqrt(uint256) (contracts/libraries/Babylonian.sol#6-18) is never used and
should be removed
SafeMath8.add(uint8,uint8) (contracts/libraries/SafeMath8.sol#16-21) is never used and
should be removed
SafeMath8.div(uint8,uint8) (contracts/libraries/SafeMath8.sol#90-92) is never used and
should be removed
SafeMath8.div(uint8,uint8,string) (contracts/libraries/SafeMath8.sol#106-112) is never
used and should be removed
SafeMath8.mod(uint8,uint8) (contracts/libraries/SafeMath8.sol#126-128) is never used
and should be removed
SafeMath8.mod(uint8,uint8,string) (contracts/libraries/SafeMath8.sol#142-145) is never
used and should be removed
SafeMath8.mul(uint8,uint8) (contracts/libraries/SafeMath8.sol#64-76) is never used and
should be removed
SafeMath8.sub(uint8,uint8) (contracts/libraries/SafeMath8.sol#33-35) is never used and
should be removed
SafeMath8.sub(uint8,uint8,string) (contracts/libraries/SafeMath8.sol#47-52) is never
used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
Context._msgData() (contracts/Oracle.sol#456-459) is never used and should be removed
FixedPoint.decode(FixedPoint.uq112x112) (contracts/Oracle.sol#288-290) is never used
and should be removed
FixedPoint.div(FixedPoint.uq112x112,uint112) (contracts/Oracle.sol#267-270) is never
used and should be removed
FixedPoint.encode(uint112) (contracts/Oracle.sol#257-259) is never used and should be
removed
FixedPoint.encode144(uint144) (contracts/Oracle.sol#262-264) is never used and should
be removed
FixedPoint.reciprocal(FixedPoint.uq112x112) (contracts/Oracle.sol#298-301) is never
used and should be removed
FixedPoint.sqrt(FixedPoint.uq112x112) (contracts/Oracle.sol#304-306) is never used and
should be removed
SafeMath.div(uint256,uint256) (contracts/Oracle.sol#135-138) is never used and should
be removed
SafeMath.div(uint256,uint256,string) (contracts/Oracle.sol#190-193) is never used and
should be removed
SafeMath.mod(uint256,uint256) (contracts/Oracle.sol#152-155) is never used and should
be removed
SafeMath.mod(uint256,uint256,string) (contracts/Oracle.sol#210-213) is never used and
should be removed
SafeMath.mul(uint256,uint256) (contracts/Oracle.sol#116-121) is never used and should
be removed
SafeMath.sub(uint256,uint256,string) (contracts/Oracle.sol#170-173) is never used and
should be removed
SafeMath.tryAdd(uint256,uint256) (contracts/Oracle.sol#24-28) is never used and should
be removed
SafeMath.tryDiv(uint256,uint256) (contracts/Oracle.sol#60-63) is never used and should
be removed
SafeMath.tryMod(uint256,uint256) (contracts/Oracle.sol#70-73) is never used and should
be removed
SafeMath.tryMul(uint256,uint256) (contracts/Oracle.sol#45-53) is never used and should
be removed
SafeMath.trySub(uint256,uint256) (contracts/Oracle.sol#35-38) is never used and should
be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.17 (contracts/AiBond.sol#3) necessitates a version too recent to be
trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/AiOMB.sol#3) necessitates a version too recent to be
```

```
trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/AiShare.sol#3) necessitates a version too recent to be
trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/Boardroom.sol#3) necessitates a version too recent to
be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/GenesisRewardPool.sol#3) necessitates a version too
recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/ShareRewardPool.sol#3) necessitates a version too
recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/Treasury.sol#3) necessitates a version too recent to
be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/interfaces/IBasisAsset.sol#3) necessitates a version
too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/interfaces/IBoardroom.sol#3) necessitates a version
too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/interfaces/IOracle.sol#3) necessitates a version too
recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/interfaces/ITreasury.sol#3) necessitates a version too
recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/interfaces/IUniswapV2Factory.sol#2) necessitates a
version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/interfaces/IUniswapV2Pair.sol#2) necessitates a
version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/interfaces/IUniswapV2Router01.sol#2) necessitates a
version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/interfaces/IUniswapV2Router02.sol#2) necessitates a
version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version>=0.8.17 (contracts/libraries/Babylonian.sol#3) necessitates a version
too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/libraries/ContractGuard.sol#3) necessitates a version
too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/libraries/Operator.sol#3) necessitates a version too
recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/libraries/SafeMath8.sol#3) necessitates a version too
recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/libraries/ShareWrapper.sol#3) necessitates a version
too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity
```

```
Pragma version>=0.6.0<0.8.0 (contracts/Oracle.sol#3) is too complex
Pragma version^0.6.0 (contracts/Oracle.sol#217) allows old versions
Pragma version^0.6.0 (contracts/Oracle.sol#236) allows old versions
Pragma version^0.6.0 (contracts/Oracle.sol#309) allows old versions
Pragma version^0.6.0 (contracts/Oracle.sol#399) allows old versions
Pragma version>=0.6.0<0.8.0 (contracts/Oracle.sol#439) is too complex
Pragma version>=0.6.0<0.8.0 (contracts/Oracle.sol#463) is too complex
Pragma version^0.6.0 (contracts/Oracle.sol#568) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity

AiBond (contracts/AiBond.sol#11-41) should inherit from IBasisAsset (contracts/
interfaces/IBasisAsset.sol#5-18)
AiOMB (contracts/AiOMB.sol#18-230) should inherit from IBasisAsset (contracts/
interfaces/IBasisAsset.sol#5-18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-
inheritance

Parameter AiOMB.isContract(address)._addr (contracts/AiOMB.sol#126) is not in mixedCase
Parameter AiOMB.setWhiteList(address)._WhiteList (contracts/AiOMB.sol#135) is not in
mixedCase
Parameter AiOMB.setPairAiShare(address)._pairAiShare (contracts/AiOMB.sol#145) is not
in mixedCase
Parameter AiOMB.setAdmin(address)._admin (contracts/AiOMB.sol#150) is not in mixedCase
Parameter AiOMB.setOracle(address)._oracle (contracts/AiOMB.sol#154) is not in
mixedCase
Parameter AiOMB.setTaxCollectorAddress(address)._taxCollectorAddress (contracts/
AiOMB.sol#160) is not in mixedCase
Parameter AiOMB.setTaxRate(uint256)._taxRate (contracts/AiOMB.sol#166) is not in
mixedCase
Parameter AiOMB.setTradingStartTime(uint256)._tradingStartTime (contracts/
AiOMB.sol#218) is not in mixedCase
Variable AiOMB.PairAiShare (contracts/AiOMB.sol#36) is not in mixedCase
Variable AiOMB.BOND (contracts/AiOMB.sol#42) is not in mixedCase
Variable AiOMB.BUSD (contracts/AiOMB.sol#43) is not in mixedCase
Variable AiOMB.PairWBNB (contracts/AiOMB.sol#44) is not in mixedCase
Variable AiOMB.PairBUSD (contracts/AiOMB.sol#45) is not in mixedCase
Parameter AiShare.isContract(address)._addr (contracts/AiShare.sol#137) is not in
mixedCase
Parameter AiShare.setWhiteList(address)._WhiteList (contracts/AiShare.sol#146) is not
in mixedCase
```

```
Parameter AiShare.setAdmin(address)._admin (contracts/AiShare.sol#156) is not in
mixedCase
Parameter AiShare.setDevFund(address)._devFund (contracts/AiShare.sol#160) is not in
mixedCase
Parameter AiShare.setCommunityFund(address)._communityFund (contracts/AiShare.sol#167)
is not in mixedCase
Parameter AiShare.setTaxCollectorAddress(address)._taxCollectorAddress (contracts/
AiShare.sol#173) is not in mixedCase
Parameter AiShare.setTaxRate(uint256)._taxRate (contracts/AiShare.sol#178) is not in
mixedCase
Variable AiShare.BOND (contracts/AiShare.sol#43) is not in mixedCase
Variable AiShare.AIO (contracts/AiShare.sol#44) is not in mixedCase
Variable AiShare.BUSD (contracts/AiShare.sol#45) is not in mixedCase
Variable AiShare.PairWBNB (contracts/AiShare.sol#46) is not in mixedCase
Variable AiShare.PairBUSD (contracts/AiShare.sol#47) is not in mixedCase
Variable AiShare.PairAIO (contracts/AiShare.sol#48) is not in mixedCase
Parameter Boardroom.initialize(IERC20,IERC20,ITreasury)._native (contracts/
Boardroom.sol#89) is not in mixedCase
Parameter Boardroom.initialize(IERC20,IERC20,ITreasury)._share (contracts/
Boardroom.sol#89) is not in mixedCase
Parameter Boardroom.initialize(IERC20,IERC20,ITreasury)._treasury (contracts/
Boardroom.sol#89) is not in mixedCase
Parameter Boardroom.setOperator(address)._operator (contracts/Boardroom.sol#105) is not
in mixedCase
Parameter Boardroom.setLockUp(uint256,uint256)._withdrawLockupEpochs (contracts/
Boardroom.sol#109) is not in mixedCase
Parameter Boardroom.setLockUp(uint256,uint256)._rewardLockupEpochs (contracts/
Boardroom.sol#109) is not in mixedCase
Parameter GenesisRewardPool.init(address,uint256)._token (contracts/
GenesisRewardPool.sol#63) is not in mixedCase
Parameter GenesisRewardPool.init(address,uint256)._poolStartTime (contracts/
GenesisRewardPool.sol#63) is not in mixedCase
Parameter GenesisRewardPool.checkPoolDuplicate(IERC20)._token (contracts/
GenesisRewardPool.sol#84) is not in mixedCase
Parameter GenesisRewardPool.add(uint256,IERC20,bool,uint256,uint16)._allocPoint
(contracts/GenesisRewardPool.sol#92) is not in mixedCase
Parameter GenesisRewardPool.add(uint256,IERC20,bool,uint256,uint16)._token (contracts/
GenesisRewardPool.sol#92) is not in mixedCase
Parameter GenesisRewardPool.add(uint256,IERC20,bool,uint256,uint16)._withUpdate
(contracts/GenesisRewardPool.sol#92) is not in mixedCase
Parameter GenesisRewardPool.add(uint256,IERC20,bool,uint256,uint16)._lastRewardTime
```

```
(contracts/GenesisRewardPool.sol#92) is not in mixedCase
Parameter GenesisRewardPool.add(uint256,IERC20,bool,uint256,uint16)._depositFeeBP
(contracts/GenesisRewardPool.sol#92) is not in mixedCase
Parameter GenesisRewardPool.set(uint256,uint256,uint16)._pid (contracts/
GenesisRewardPool.sol#121) is not in mixedCase
Parameter GenesisRewardPool.set(uint256,uint256,uint16)._allocPoint (contracts/
GenesisRewardPool.sol#121) is not in mixedCase
Parameter GenesisRewardPool.set(uint256,uint256,uint16)._depositFeeBP (contracts/
GenesisRewardPool.sol#121) is not in mixedCase
Parameter GenesisRewardPool.getGeneratedReward(uint256,uint256)._fromTime (contracts/
GenesisRewardPool.sol#133) is not in mixedCase
Parameter GenesisRewardPool.getGeneratedReward(uint256,uint256)._toTime (contracts/
GenesisRewardPool.sol#133) is not in mixedCase
Parameter GenesisRewardPool.pending(uint256,address)._pid (contracts/
GenesisRewardPool.sol#147) is not in mixedCase
Parameter GenesisRewardPool.pending(uint256,address)._user (contracts/
GenesisRewardPool.sol#147) is not in mixedCase
Parameter GenesisRewardPool.updatePool(uint256)._pid (contracts/
GenesisRewardPool.sol#169) is not in mixedCase
Parameter GenesisRewardPool.deposit(uint256,uint256)._pid (contracts/
GenesisRewardPool.sol#192) is not in mixedCase
Parameter GenesisRewardPool.deposit(uint256,uint256)._amount (contracts/
GenesisRewardPool.sol#192) is not in mixedCase
Parameter GenesisRewardPool.withdraw(uint256,uint256)._pid (contracts/
GenesisRewardPool.sol#221) is not in mixedCase
Parameter GenesisRewardPool.withdraw(uint256,uint256)._amount (contracts/
GenesisRewardPool.sol#221) is not in mixedCase
Parameter GenesisRewardPool.emergencyWithdraw(uint256)._pid (contracts/
GenesisRewardPool.sol#241) is not in mixedCase
Parameter GenesisRewardPool.safeTransfer(address,uint256)._to (contracts/
GenesisRewardPool.sol#252) is not in mixedCase
Parameter GenesisRewardPool.safeTransfer(address,uint256)._amount (contracts/
GenesisRewardPool.sol#252) is not in mixedCase
Parameter GenesisRewardPool.setFeeAddress(address)._feeAddress (contracts/
GenesisRewardPool.sol#263) is not in mixedCase
Parameter GenesisRewardPool.setOperator(address)._operator (contracts/
GenesisRewardPool.sol#269) is not in mixedCase
Parameter ShareRewardPool.init(address,uint256)._share (contracts/
ShareRewardPool.sol#69) is not in mixedCase
Parameter ShareRewardPool.init(address,uint256)._poolStartTime (contracts/
ShareRewardPool.sol#69) is not in mixedCase
```

```
Parameter ShareRewardPool.checkPoolDuplicate(IERC20)._token (contracts/
ShareRewardPool.sol#90) is not in mixedCase
Parameter ShareRewardPool.add(uint256,IERC20,bool,uint256,uint16)._allocPoint
(contracts/ShareRewardPool.sol#99) is not in mixedCase
Parameter ShareRewardPool.add(uint256,IERC20,bool,uint256,uint16)._token (contracts/
ShareRewardPool.sol#100) is not in mixedCase
Parameter ShareRewardPool.add(uint256,IERC20,bool,uint256,uint16)._withUpdate
(contracts/ShareRewardPool.sol#101) is not in mixedCase
Parameter ShareRewardPool.add(uint256,IERC20,bool,uint256,uint16)._lastRewardTime
(contracts/ShareRewardPool.sol#102) is not in mixedCase
Parameter ShareRewardPool.add(uint256,IERC20,bool,uint256,uint16)._depositFeeBP
(contracts/ShareRewardPool.sol#103) is not in mixedCase
Parameter ShareRewardPool.set(uint256,uint256,uint16)._pid (contracts/
ShareRewardPool.sol#142) is not in mixedCase
Parameter ShareRewardPool.set(uint256,uint256,uint16)._allocPoint (contracts/
ShareRewardPool.sol#142) is not in mixedCase
Parameter ShareRewardPool.set(uint256,uint256,uint16)._depositFeeBP (contracts/
ShareRewardPool.sol#142) is not in mixedCase
Parameter ShareRewardPool.getGeneratedReward(uint256,uint256)._fromTime (contracts/
ShareRewardPool.sol#155) is not in mixedCase
Parameter ShareRewardPool.getGeneratedReward(uint256,uint256)._toTime (contracts/
ShareRewardPool.sol#155) is not in mixedCase
Parameter ShareRewardPool.pendingShare(uint256,address)._pid (contracts/
ShareRewardPool.sol#169) is not in mixedCase
Parameter ShareRewardPool.pendingShare(uint256,address)._user (contracts/
ShareRewardPool.sol#169) is not in mixedCase
Parameter ShareRewardPool.updatePool(uint256)._pid (contracts/ShareRewardPool.sol#191)
is not in mixedCase
Parameter ShareRewardPool.deposit(uint256,uint256)._pid (contracts/
ShareRewardPool.sol#214) is not in mixedCase
Parameter ShareRewardPool.deposit(uint256,uint256)._amount (contracts/
ShareRewardPool.sol#214) is not in mixedCase
Parameter ShareRewardPool.withdraw(uint256,uint256)._pid (contracts/
ShareRewardPool.sol#248) is not in mixedCase
Parameter ShareRewardPool.withdraw(uint256,uint256)._amount (contracts/
ShareRewardPool.sol#248) is not in mixedCase
Parameter ShareRewardPool.emergencyWithdraw(uint256)._pid (contracts/
ShareRewardPool.sol#268) is not in mixedCase
Parameter ShareRewardPool.safeShareTransfer(address,uint256)._to (contracts/
ShareRewardPool.sol#279) is not in mixedCase
Parameter ShareRewardPool.safeShareTransfer(address,uint256)._amount (contracts/
```

```
ShareRewardPool.sol#279) is not in mixedCase
Parameter ShareRewardPool.setFeeAddress(address)._feeAddress (contracts/
ShareRewardPool.sol#290) is not in mixedCase
Parameter ShareRewardPool.setOperator(address)._operator (contracts/
ShareRewardPool.sol#295) is not in mixedCase
Parameter
Treasury.initialize(address,address,address,address,address,address,uint256)._native
(contracts/Treasury.sol#235) is not in mixedCase
Parameter
Treasury.initialize(address,address,address,address,address,address,uint256)._bond
(contracts/Treasury.sol#236) is not in mixedCase
Parameter
Treasury.initialize(address,address,address,address,address,address,uint256)._share
(contracts/Treasury.sol#237) is not in mixedCase
Parameter Treasury.initialize(address,address,address,address,address,address,uint256)._
nativeOracle (contracts/Treasury.sol#238) is not in mixedCase
Parameter
Treasury.initialize(address,address,address,address,address,address,uint256)._boardroom
(contracts/Treasury.sol#239) is not in mixedCase
Parameter
Treasury.initialize(address,address,address,address,address,address,uint256)._genesis
(contracts/Treasury.sol#240) is not in mixedCase
Parameter
Treasury.initialize(address,address,address,address,address,address,uint256)._startTime
(contracts/Treasury.sol#241) is not in mixedCase
Parameter Treasury.setOperator(address)._operator (contracts/Treasury.sol#281) is not
in mixedCase
Parameter Treasury.setBoardroom(address)._boardroom (contracts/Treasury.sol#285) is not
in mixedCase
Parameter Treasury.setNativeOracle(address)._nativeOracle (contracts/Treasury.sol#289)
is not in mixedCase
Parameter Treasury.setNativePriceCeiling(uint256)._nativePriceCeiling (contracts/
Treasury.sol#293) is not in mixedCase
Parameter Treasury.setMaxSupplyExpansionPercents(uint256)._maxSupplyExpansionPercent
(contracts/Treasury.sol#298) is not in mixedCase
Parameter Treasury.setSupplyTiersEntry(uint8,uint256)._index (contracts/
Treasury.sol#303) is not in mixedCase
Parameter Treasury.setSupplyTiersEntry(uint8,uint256)._value (contracts/
Treasury.sol#303) is not in mixedCase
Parameter Treasury.setMaxExpansionTiersEntry(uint8,uint256)._index (contracts/
Treasury.sol#316) is not in mixedCase
```

```
Parameter Treasury.setMaxExpansionTiersEntry(uint8,uint256)._value (contracts/
Treasury.sol#316) is not in mixedCase
Parameter Treasury.setBondDepletionFloorPercent(uint256)._bondDepletionFloorPercent
(contracts/Treasury.sol#324) is not in mixedCase
Parameter Treasury.setMaxSupplyContractionPercent(uint256)._maxSupplyContractionPercent
(contracts/Treasury.sol#329) is not in mixedCase
Parameter Treasury.setMaxDebtRatioPercent(uint256)._maxDebtRatioPercent (contracts/
Treasury.sol#334) is not in mixedCase
Parameter Treasury.setBootstrap(uint256,uint256)._bootstrapEpochs (contracts/
Treasury.sol#339) is not in mixedCase
Parameter Treasury.setBootstrap(uint256,uint256)._bootstrapSupplyExpansionPercent
(contracts/Treasury.sol#339) is not in mixedCase
Parameter Treasury.setExtraFunds(address,uint256,address,uint256)._daoFund (contracts/
Treasury.sol#347) is not in mixedCase
Parameter Treasury.setExtraFunds(address,uint256,address,uint256)._daoFundSharedPercent
(contracts/Treasury.sol#348) is not in mixedCase
Parameter Treasury.setExtraFunds(address,uint256,address,uint256)._devFund (contracts/
Treasury.sol#349) is not in mixedCase
Parameter Treasury.setExtraFunds(address,uint256,address,uint256)._devFundSharedPercent
(contracts/Treasury.sol#350) is not in mixedCase
Parameter Treasury.setMaxDiscountRate(uint256)._maxDiscountRate (contracts/
Treasury.sol#362) is not in mixedCase
Parameter Treasury.setMaxPremiumRate(uint256)._maxPremiumRate (contracts/
Treasury.sol#366) is not in mixedCase
Parameter Treasury.setDiscountPercent(uint256)._discountPercent (contracts/
Treasury.sol#370) is not in mixedCase
Parameter Treasury.setPremiumThreshold(uint256)._premiumThreshold (contracts/
Treasury.sol#375) is not in mixedCase
Parameter Treasury.setPremiumPercent(uint256)._premiumPercent (contracts/
Treasury.sol#381) is not in mixedCase
Parameter Treasury.setMintingFactorForPayingDebt(uint256)._mintingFactorForPayingDebt
(contracts/Treasury.sol#386) is not in mixedCase
Parameter Treasury.buyBonds(uint256,uint256)._nativeAmount (contracts/Treasury.sol#407)
is not in mixedCase
Parameter Treasury.redeemBonds(uint256,uint256)._bondAmount (contracts/
Treasury.sol#436) is not in mixedCase
Parameter Treasury.boardroomSetOperator(address)._operator (contracts/Treasury.sol#540)
is not in mixedCase
Parameter Treasury.boardroomSetLockUp(uint256,uint256)._withdrawLockupEpochs (contracts/
Treasury.sol#544) is not in mixedCase
Parameter Treasury.boardroomSetLockUp(uint256,uint256)._rewardLockupEpochs (contracts/
```

```
Treasury.sol#544) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (contracts/interfaces/IUniswapV2Pair.sol#33)
is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (contracts/interfaces/IUniswapV2Pair.sol#35)
is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (contracts/interfaces/
IUniswapV2Pair.sol#66) is not in mixedCase
Function IUniswapV2Router01.WETH() (contracts/interfaces/IUniswapV2Router01.sol#7) is
not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions

Struct FixedPoint.uq112x112 (contracts/Oracle.sol#242-244) is not in CapWords
Struct FixedPoint.uq144x112 (contracts/Oracle.sol#248-250) is not in CapWords
Parameter Epoch.setPeriod(uint256)._period (contracts/Oracle.sol#640) is not in
mixedCase
Parameter Epoch.setEpoch(uint256)._epoch (contracts/Oracle.sol#645) is not in mixedCase
Parameter Oracle.consult(address,uint256)._token (contracts/Oracle.sol#715) is not in
mixedCase
Parameter Oracle.consult(address,uint256)._amountIn (contracts/Oracle.sol#715) is not
in mixedCase
Parameter Oracle.twap(address,uint256)._token (contracts/Oracle.sol#724) is not in
mixedCase
Parameter Oracle.twap(address,uint256)._amountIn (contracts/Oracle.sol#724) is not in
mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions

Redundant expression "this (contracts/Oracle.sol#457)" inContext (contracts/
Oracle.sol#451-460)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-
statements

Variable Treasury.setExtraFunds(address,uint256,address,uint256)._daoFundSharedPercent
(contracts/Treasury.sol#348) is too similar to
Treasury.setExtraFunds(address,uint256,address,uint256)._devFundSharedPercent
(contracts/Treasury.sol#350)
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256
,address,uint256).amountADesired (contracts/interfaces/IUniswapV2Router01.sol#12) is
too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,u
int256,address,uint256).amountBDesired (contracts/interfaces/IUniswapV2Router01.sol#13)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

Variable UniswapV2OracleLibrary.currentCumulativePrices(address).price0Cumulative (contracts/Oracle.sol#416) is too similar to UniswapV2OracleLibrary.currentCumulativePrices(address).price1Cumulative (contracts/Oracle.sol#417)
Variable Oracle.price0Average (contracts/Oracle.sol#670) is too similar to Oracle.price1Average (contracts/Oracle.sol#671)
Variable Oracle.update().price0Cumulative (contracts/Oracle.sol#694) is too similar to Oracle.update().price1Cumulative (contracts/Oracle.sol#694)
Variable Oracle.price0CumulativeLast (contracts/Oracle.sol#668) is too similar to Oracle.price1CumulativeLast (contracts/Oracle.sol#669)
Variable Oracle.update().price0Cumulative (contracts/Oracle.sol#694) is too similar to Oracle.twap(address,uint256).price1Cumulative (contracts/Oracle.sol#725)
Variable Oracle.twap(address,uint256).price0Cumulative (contracts/Oracle.sol#725) is too similar to Oracle.update().price1Cumulative (contracts/Oracle.sol#694)
Variable Oracle.twap(address,uint256).price0Cumulative (contracts/Oracle.sol#725) is too similar to Oracle.twap(address,uint256).price1Cumulative (contracts/Oracle.sol#725)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

Treasury.initialize(address,address,address,address,address,address,uint256) (contracts/Treasury.sol#234-279) uses literals with too many digits:
        - supplyTiers = (0,100000000000000000000,150000000000000000000,250000000000000000000,350000000000000000000,600000000000000000000,2500000000000000000000,5000000000000000000000,10000000000000000000000) (contracts/Treasury.sol#256)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

GenesisRewardPool.runningTime (contracts/GenesisRewardPool.sol#54) should be constant
GenesisRewardPool.tokenPerSecond (contracts/GenesisRewardPool.sol#53) should be constant
ShareRewardPool.runningTime (contracts/ShareRewardPool.sol#59) should be constant
ShareRewardPool.sharesPerSecond (contracts/ShareRewardPool.sol#58) should be constant
Treasury.deadAddress (contracts/Treasury.sol#49) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

AiOMB.rewardPoolDistributed (contracts/AiOMB.sol#30) should be immutable
AiShare.communityFundRewardRate (contracts/AiShare.sol#31) should be immutable
```

```
AiShare.devFundRewardRate (contracts/AiShare.sol#32) should be immutable
AiShare.endTime (contracts/AiShare.sol#30) should be immutable
AiShare.rewardPoolDistributed (contracts/AiShare.sol#63) should be immutable
AiShare.startTime (contracts/AiShare.sol#29) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-immutable

Oracle.pair (contracts/Oracle.sol#664) should be immutable
Oracle.token0 (contracts/Oracle.sol#662) should be immutable
Oracle.token1 (contracts/Oracle.sol#663) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-immutable
. analyzed (43 contracts with 84 detectors), 337 result(s) found
```

0x Guard