0x Guard

Smart contracts security assessment

Final report

Tariff: Standard

Argo Finance

March 2022

0xguard.com

hello@0xguard.com

Contents

1. Introduction 3
2. Contracts checked 3
3. Procedure 4
4. Classification of issue severity 4
5. Issues 5
6. Conclusion 6
7. Disclaimer 7
8. Slither output 8

Introduction

This report has been prepared for the Argo Finance team upon their request.

The audited project is a fork of the Tomb Finance Project.

The purpose of this audit was to ensure that no issues were introduced with the changes to the original code and that known vulnerabilities (e.g. circumventing the protocol's fee system) are fixed prior to deployment.

Further details about Argo Finance are available at the official website: http://argofinance.io/

Name	Argo Finance
Audit date	2022-03-09 - 2022-03-11
Language	Solidity
Platform	Metis

Contracts checked

Name	Address
TridentRewardPool	https://github.com/argo-finance/argo-contracts/blob/bd5c6b8ea8affb3d5ba674249ac5cab8493c2ef5/contracts/TridentRewardPool.sol
Trident	https://github.com/argo-finance/argo-contracts/blob/bd5c6b8ea8affb3d5ba674249ac5cab8493c2ef5/contracts/Trident.sol
Argo	https://github.com/argo-finance/argo-contracts/blob/bd5c6b8ea8affb3d5ba674249ac5cab8493c2ef5/contracts/Argo.sol
ArgoGenesisRewardPool	https://github.com/argo-finance/argo-contracts/blob/bd5c6b8ea8affb3d5ba674249ac5cab8493c2ef5/contracts/ArgoGenesisRewardPool.sol

Cove	https://github.com/argo-finance/argo-contracts/blob/bd5c6b8ea8affb3d5ba674249ac5cab8493c2ef5/contracts/Cove.sol
Treasury	https://github.com/argo-finance/argo-contracts/blob/bd5c6b8ea8affb3d5ba674249ac5cab8493c2ef5/contracts/Treasury.sol
Oar	https://github.com/argo-finance/argo-contracts/blob/bd5c6b8ea8affb3d5ba674249ac5cab8493c2ef5/contracts/Oar.sol

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) all the issues found by the tools

Manual audit

- Comparing the project to the Tomb Finance implementation

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

No issues were found

Conclusion

The Argo Finance Project was compared with the Tomb Project. Argo Finance has changed the implementation of ArgoGenesisRewardPool and Token contracts.

The changed Token contract is not affected by the vulnerability that was discovered in the Tomb before because it doesn't contain the implementation of transfer with taxes.

In the TridentRewardPool reward distribution was changed. Removed community fund from Trident contract. The Argo contract removed all tax-related components and added initial distribution. The ArgoGenesisRewardPool contract added support of several tokens.

No issues were found with the changes.

Disclaimer

Slither output

```
UniswapV2OracleLibrary.currentBlockTimestamp() (contracts/lib/
UniswapV2OracleLibrary.sol#13-15) uses a weak PRNG: "uint32(block.timestamp % 2 ** 32)
(contracts/lib/UniswapV2OracleLibrary.sol#14)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Argo._operator (contracts/Argo.sol#32) shadows:
        - Operator._operator (contracts/owner/Operator.sol#9)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-
shadowing

Argo.governanceRecoverUnsupported(IERC20,uint256,address) (contracts/Argo.sol#114-120)
ignores return value by _token.transfer(_to,_amount) (contracts/Argo.sol#119)
Treasury._sendToMasonry(uint256) (contracts/Treasury.sol#461-484) ignores return value
by IERC20(ARGO).transfer(daoFund,_daoFundSharedAmount) (contracts/Treasury.sol#467)
Treasury._sendToMasonry(uint256) (contracts/Treasury.sol#461-484) ignores return value
by IERC20(ARGO).transfer(devFund,_devFundSharedAmount) (contracts/Treasury.sol#474)
Trident.governanceRecoverUnsupported(IERC20,uint256,address) (contracts/
Trident.sol#83-89) ignores return value by _token.transfer(_to,_amount) (contracts/
Trident.sol#88)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-
transfer

ArgoGenesisRewardPool.pendingARGO(uint256,address) (contracts/
ArgoGenesisRewardPool.sol#167-178) performs a multiplication on the result of a
division:
        -_argoReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint)
(contracts/ArgoGenesisRewardPool.sol#174)
        -accArgoPerShare = accArgoPerShare.add(_argoReward.mul(1e18).div(tokenSupply))
(contracts/ArgoGenesisRewardPool.sol#175)
ArgoGenesisRewardPool.updatePool(uint256) (contracts/ArgoGenesisRewardPool.sol#189-209)
performs a multiplication on the result of a division:
        -_argoReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint)
(contracts/ArgoGenesisRewardPool.sol#205)
        -pool.accArgoPerShare =
pool.accArgoPerShare.add(_argoReward.mul(1e18).div(tokenSupply)) (contracts/
ArgoGenesisRewardPool.sol#206)
Treasury.allocateSeigniorage() (contracts/Treasury.sol#496-536) performs a
```

```
multiplication on the result of a division:
        -_seigniorage = ARGOSupply.mul(_percentage).div(1e18) (contracts/
Treasury.sol#519)
        -_savedForGarden =
_seigniorage.mul(seigniorageExpansionFloorPercent).div(10000) (contracts/
Treasury.sol#520)
TridentRewardPool.pendingShare(uint256,address) (contracts/
TridentRewardPool.sol#148-159) performs a multiplication on the result of a division:
        -_tridentReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint)
(contracts/TridentRewardPool.sol#155)
        -accTridentPerShare =
accTridentPerShare.add(_tridentReward.mul(1e18).div(tokenSupply)) (contracts/
TridentRewardPool.sol#156)
TridentRewardPool.updatePool(uint256) (contracts/TridentRewardPool.sol#170-190)
performs a multiplication on the result of a division:
        -_tridentReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint)
(contracts/TridentRewardPool.sol#186)
        -pool.accTridentPerShare =
pool.accTridentPerShare.add(_tridentReward.mul(1e18).div(tokenSupply)) (contracts/
TridentRewardPool.sol#187)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply

ArgoGenesisRewardPool.updatePool(uint256) (contracts/ArgoGenesisRewardPool.sol#189-209)
uses a dangerous strict equality:
        - tokenSupply == 0 (contracts/ArgoGenesisRewardPool.sol#195)
TridentRewardPool.updatePool(uint256) (contracts/TridentRewardPool.sol#170-190) uses a
dangerous strict equality:
        - tokenSupply == 0 (contracts/TridentRewardPool.sol#176)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-
strict-equalities

Reentrancy in Treasury.buyBonds(uint256,uint256) (contracts/Treasury.sol#406-433):
        External calls:
        - IBasisAsset(ARGO).burnFrom(msg.sender,_ARGOAmount) (contracts/
Treasury.sol#426)
        - IBasisAsset(bud).mint(msg.sender,_bondAmount) (contracts/Treasury.sol#427)
        State variables written after the call(s):
        - epochSupplyContractionLeft = epochSupplyContractionLeft.sub(_ARGOAmount)
(contracts/Treasury.sol#429)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1
```

```
Treasury.setSupplyTiersEntry(uint8,uint256) (contracts/Treasury.sol#302-313) contains a
tautology or contradiction:
        - require(bool,string)(_index >= 0,Index has to be higher than 0) (contracts/
Treasury.sol#303)
Treasury.setMaxExpansionTiersEntry(uint8,uint256) (contracts/Treasury.sol#315-321)
contains a tautology or contradiction:
        - require(bool,string)(_index >= 0,Index has to be higher than 0) (contracts/
Treasury.sol#316)
Treasury._calculateMaxSupplyExpansionPercent(uint256) (contracts/Treasury.sol#486-494)
contains a tautology or contradiction:
        - tierId >= 0 (contracts/Treasury.sol#487)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-
contradiction

Argo._getArgoPrice()._price (contracts/Argo.sol#48) is a local variable never
initialized
Treasury.allocateSeigniorage()._savedForBond (contracts/Treasury.sol#508) is a local
variable never initialized
Treasury.getARGOUpdatedPrice().price (contracts/Treasury.sol#163) is a local variable
never initialized
FixedPoint.mul(FixedPoint.uq112x112,uint256).z (contracts/lib/FixedPoint.sol#44) is a
local variable never initialized
Treasury.getARGOPrice().price (contracts/Treasury.sol#155) is a local variable never
initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-
local-variables

Argo._getArgoPrice() (contracts/Argo.sol#47-53) ignores return value by
IOracle(argoOracle).consult(address(this),1e18) (contracts/Argo.sol#48-52)
Treasury.getARGOPrice() (contracts/Treasury.sol#154-160) ignores return value by
IOracle(ARGOOracle).consult(ARGO,1e18) (contracts/Treasury.sol#155-159)
Treasury.getARGOUpdatedPrice() (contracts/Treasury.sol#162-168) ignores return value by
IOracle(ARGOOracle).twap(ARGO,1e18) (contracts/Treasury.sol#163-167)
Treasury.buyBonds(uint256,uint256) (contracts/Treasury.sol#406-433) ignores return
value by IBasisAsset(bud).mint(msg.sender,_bondAmount) (contracts/Treasury.sol#427)
Treasury._sendToMasonry(uint256) (contracts/Treasury.sol#461-484) ignores return value
by IBasisAsset(ARGO).mint(address(this),_amount) (contracts/Treasury.sol#462)
Treasury.allocateSeigniorage() (contracts/Treasury.sol#496-536) ignores return value by
IBasisAsset(ARGO).mint(address(this),_savedForBond) (contracts/Treasury.sol#531)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

```
ArgoGenesisRewardPool.setOperator(address) (contracts/
ArgoGenesisRewardPool.sol#289-291) should emit an event for:
	- operator = _operator (contracts/ArgoGenesisRewardPool.sol#290)
Cove.setOperator(address) (contracts/Cove.sol#138-140) should emit an event for:
	- operator = _operator (contracts/Cove.sol#139)
Treasury.setOperator(address) (contracts/Treasury.sol#280-282) should emit an event
for:
	- operator = _operator (contracts/Treasury.sol#281)
Treasury.setGarden(address) (contracts/Treasury.sol#284-286) should emit an event for:
	- garden = _garden (contracts/Treasury.sol#285)
TridentRewardPool.setOperator(address) (contracts/TridentRewardPool.sol#256-258) should
emit an event for:
	- operator = _operator (contracts/TridentRewardPool.sol#257)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
access-control

ArgoGenesisRewardPool.add(uint256,IERC20,bool,uint256) (contracts/
ArgoGenesisRewardPool.sol#100-138) should emit an event for:
	- totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/
ArgoGenesisRewardPool.sol#136)
ArgoGenesisRewardPool.set(uint256,uint256) (contracts/
ArgoGenesisRewardPool.sol#141-150) should emit an event for:
	- totalAllocPoint = totalAllocPoint.sub(pool.allocPoint).add(_allocPoint)
(contracts/ArgoGenesisRewardPool.sol#145-147)
Cove.setLockUp(uint256,uint256) (contracts/Cove.sol#142-146) should emit an event for:
	- withdrawLockupEpochs = _withdrawLockupEpochs (contracts/Cove.sol#144)
	- rewardLockupEpochs = _rewardLockupEpochs (contracts/Cove.sol#145)
Treasury.setARGOPriceCeiling(uint256) (contracts/Treasury.sol#292-295) should emit an
event for:
	- ARGOPriceCeiling = _ARGOPriceCeiling (contracts/Treasury.sol#294)
Treasury.setMaxSupplyExpansionPercents(uint256) (contracts/Treasury.sol#297-300) should
emit an event for:
	- maxSupplyExpansionPercent = _maxSupplyExpansionPercent (contracts/
Treasury.sol#299)
Treasury.setBondDepletionFloorPercent(uint256) (contracts/Treasury.sol#323-326) should
emit an event for:
	- bondDepletionFloorPercent = _bondDepletionFloorPercent (contracts/
Treasury.sol#325)
Treasury.setMaxDebtRatioPercent(uint256) (contracts/Treasury.sol#333-336) should emit
an event for:
        - maxDebtRatioPercent = _maxDebtRatioPercent (contracts/Treasury.sol#335)
Treasury.setBootstrap(uint256,uint256) (contracts/Treasury.sol#338-343) should emit an
event for:
        - bootstrapEpochs = _bootstrapEpochs (contracts/Treasury.sol#341)
        - bootstrapSupplyExpansionPercent = _bootstrapSupplyExpansionPercent (contracts/
Treasury.sol#342)
Treasury.setExtraFunds(address,uint256,address,uint256) (contracts/
Treasury.sol#345-359) should emit an event for:
        - daoFundSharedPercent = _daoFundSharedPercent (contracts/Treasury.sol#356)
        - devFundSharedPercent = _devFundSharedPercent (contracts/Treasury.sol#358)
Treasury.setMaxDiscountRate(uint256) (contracts/Treasury.sol#361-363) should emit an
event for:
        - maxDiscountRate = _maxDiscountRate (contracts/Treasury.sol#362)
Treasury.setMaxPremiumRate(uint256) (contracts/Treasury.sol#365-367) should emit an
event for:
        - maxPremiumRate = _maxPremiumRate (contracts/Treasury.sol#366)
Treasury.setDiscountPercent(uint256) (contracts/Treasury.sol#369-372) should emit an
event for:
        - discountPercent = _discountPercent (contracts/Treasury.sol#371)
Treasury.setPremiumThreshold(uint256) (contracts/Treasury.sol#374-378) should emit an
event for:
        - premiumThreshold = _premiumThreshold (contracts/Treasury.sol#377)
Treasury.setPremiumPercent(uint256) (contracts/Treasury.sol#380-383) should emit an
event for:
        - premiumPercent = _premiumPercent (contracts/Treasury.sol#382)
Treasury.setMintingFactorForPayingDebt(uint256) (contracts/Treasury.sol#385-388) should
emit an event for:
        - mintingFactorForPayingDebt = _mintingFactorForPayingDebt (contracts/
Treasury.sol#387)
TridentRewardPool.add(uint256,IERC20,bool,uint256) (contracts/
TridentRewardPool.sol#81-119) should emit an event for:
        - totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/
TridentRewardPool.sol#117)
TridentRewardPool.set(uint256,uint256) (contracts/TridentRewardPool.sol#122-131) should
emit an event for:
        - totalAllocPoint = totalAllocPoint.sub(pool.allocPoint).add(_allocPoint)
(contracts/TridentRewardPool.sol#126-128)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic

ArgoGenesisRewardPool.constructor(address,address,address,address,address,address,addres
s)._xtethys (contracts/ArgoGenesisRewardPool.sol#70) lacks a zero-check on :
                - xtethys = _xtethys (contracts/ArgoGenesisRewardPool.sol#78)
ArgoGenesisRewardPool.setOperator(address)._operator (contracts/
ArgoGenesisRewardPool.sol#289) lacks a zero-check on :
                - operator = _operator (contracts/ArgoGenesisRewardPool.sol#290)
Cove.setOperator(address)._operator (contracts/Cove.sol#138) lacks a zero-check on :
                - operator = _operator (contracts/Cove.sol#139)
Treasury.initialize(address,address,address,address,address)._ARGO (contracts/
Treasury.sol#238) lacks a zero-check on :
                - ARGO = _ARGO (contracts/Treasury.sol#244)
Treasury.initialize(address,address,address,address,address)._bud (contracts/
Treasury.sol#239) lacks a zero-check on :
                - bud = _bud (contracts/Treasury.sol#245)
Treasury.initialize(address,address,address,address,address)._petal (contracts/
Treasury.sol#240) lacks a zero-check on :
                - petal = _petal (contracts/Treasury.sol#246)
Treasury.initialize(address,address,address,address,address)._ARGOOracle (contracts/
Treasury.sol#241) lacks a zero-check on :
                - ARGOOracle = _ARGOOracle (contracts/Treasury.sol#247)
Treasury.initialize(address,address,address,address,address)._garden (contracts/
Treasury.sol#242) lacks a zero-check on :
                - garden = _garden (contracts/Treasury.sol#248)
Treasury.setOperator(address)._operator (contracts/Treasury.sol#280) lacks a zero-check
on :
                - operator = _operator (contracts/Treasury.sol#281)
Treasury.setGarden(address)._garden (contracts/Treasury.sol#284) lacks a zero-check
on :
                - garden = _garden (contracts/Treasury.sol#285)
Treasury.setARGOOracle(address)._ARGOOracle (contracts/Treasury.sol#288) lacks a zero-
check on :
                - ARGOOracle = _ARGOOracle (contracts/Treasury.sol#289)
TridentRewardPool.setOperator(address)._operator (contracts/TridentRewardPool.sol#256)
lacks a zero-check on :
                - operator = _operator (contracts/TridentRewardPool.sol#257)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
address-validation

ArgoGenesisRewardPool.updatePool(uint256) (contracts/ArgoGenesisRewardPool.sol#189-209)
has external calls inside a loop: tokenSupply = pool.token.balanceOf(address(this))
(contracts/ArgoGenesisRewardPool.sol#194)
Treasury.getARGOCirculatingSupply() (contracts/Treasury.sol#396-404) has external calls
```

```
inside a loop: balanceExcluded =
balanceExcluded.add(ARGOErc20.balanceOf(excludedFromTotalSupply[entryId])) (contracts/
Treasury.sol#401)
TridentRewardPool.updatePool(uint256) (contracts/TridentRewardPool.sol#170-190) has
external calls inside a loop: tokenSupply = pool.token.balanceOf(address(this))
(contracts/TridentRewardPool.sol#175)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-
a-loop

Variable 'Argo._getArgoPrice()._price (contracts/Argo.sol#48)' in Argo._getArgoPrice()
(contracts/Argo.sol#47-53) potentially used before declaration: uint256(_price)
(contracts/Argo.sol#49)
Variable 'Treasury.getARGOPrice().price (contracts/Treasury.sol#155)' in
Treasury.getARGOPrice() (contracts/Treasury.sol#154-160) potentially used before
declaration: uint256(price) (contracts/Treasury.sol#156)
Variable 'Treasury.getARGOUpdatedPrice().price (contracts/Treasury.sol#163)' in
Treasury.getARGOUpdatedPrice() (contracts/Treasury.sol#162-168) potentially used before
declaration: uint256(price) (contracts/Treasury.sol#164)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-
declaration-usage-of-local-variables

Reentrancy in Treasury.allocateSeigniorage() (contracts/Treasury.sol#496-536):
        External calls:
        - _updateARGOPrice() (contracts/Treasury.sol#497)
                - IOracle(ARGOOracle).update() (contracts/Treasury.sol#393)
        State variables written after the call(s):
        - _mse = _calculateMaxSupplyExpansionPercent(ARGOSupply).mul(1e14) (contracts/
Treasury.sol#510)
                - maxSupplyExpansionPercent = maxExpansionTiers[tierId] (contracts/
Treasury.sol#489)
        - previousEpochARGOPrice = getARGOPrice() (contracts/Treasury.sol#498)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2

Reentrancy in Treasury._sendToMasonry(uint256) (contracts/Treasury.sol#461-484):
        External calls:
        - IBasisAsset(ARGO).mint(address(this),_amount) (contracts/Treasury.sol#462)
        - IERC20(ARGO).transfer(daoFund,_daoFundSharedAmount) (contracts/
Treasury.sol#467)
        Event emitted after the call(s):
        - DaoFundFunded(block.timestamp,_daoFundSharedAmount) (contracts/
Treasury.sol#468)
```

```
Reentrancy in Treasury._sendToMasonry(uint256) (contracts/Treasury.sol#461-484):
        External calls:
        - IBasisAsset(ARGO).mint(address(this),_amount) (contracts/Treasury.sol#462)
        - IERC20(ARGO).transfer(daoFund,_daoFundSharedAmount) (contracts/
Treasury.sol#467)
        - IERC20(ARGO).transfer(devFund,_devFundSharedAmount) (contracts/
Treasury.sol#474)
        Event emitted after the call(s):
        - DevFundFunded(block.timestamp,_devFundSharedAmount) (contracts/
Treasury.sol#475)
Reentrancy in Treasury._sendToMasonry(uint256) (contracts/Treasury.sol#461-484):
        External calls:
        - IBasisAsset(ARGO).mint(address(this),_amount) (contracts/Treasury.sol#462)
        - IERC20(ARGO).transfer(daoFund,_daoFundSharedAmount) (contracts/
Treasury.sol#467)
        - IERC20(ARGO).transfer(devFund,_devFundSharedAmount) (contracts/
Treasury.sol#474)
        - IERC20(ARGO).safeApprove(garden,0) (contracts/Treasury.sol#480)
        - IERC20(ARGO).safeApprove(garden,_amount) (contracts/Treasury.sol#481)
        - IMasonry(garden).allocateSeigniorage(_amount) (contracts/Treasury.sol#482)
        Event emitted after the call(s):
        - GardenFunded(block.timestamp,_amount) (contracts/Treasury.sol#483)
Reentrancy in Cove.allocateSeigniorage(uint256) (contracts/Cove.sol#233-250):
        External calls:
        - argo.safeTransferFrom(msg.sender,address(this),amount) (contracts/
Cove.sol#248)
        Event emitted after the call(s):
        - RewardAdded(msg.sender,amount) (contracts/Cove.sol#249)
Reentrancy in Treasury.buyBonds(uint256,uint256) (contracts/Treasury.sol#406-433):
        External calls:
        - IBasisAsset(ARGO).burnFrom(msg.sender,_ARGOAmount) (contracts/
Treasury.sol#426)
        - IBasisAsset(bud).mint(msg.sender,_bondAmount) (contracts/Treasury.sol#427)
        - _updateARGOPrice() (contracts/Treasury.sol#430)
                - IOracle(ARGOOracle).update() (contracts/Treasury.sol#393)
        Event emitted after the call(s):
        - BoughtBonds(msg.sender,_ARGOAmount,_bondAmount) (contracts/Treasury.sol#432)
Reentrancy in Cove.claimReward() (contracts/Cove.sol#222-231):
        External calls:
        - argo.safeTransfer(msg.sender,reward) (contracts/Cove.sol#228)
        Event emitted after the call(s):
```

```
        - RewardPaid(msg.sender,reward) (contracts/Cove.sol#229)
Reentrancy in ArgoGenesisRewardPool.emergencyWithdraw(uint256) (contracts/
ArgoGenesisRewardPool.sol#267-275):
        External calls:
        - pool.token.safeTransfer(msg.sender,_amount) (contracts/
ArgoGenesisRewardPool.sol#273)
        Event emitted after the call(s):
        - EmergencyWithdraw(msg.sender,_pid,_amount) (contracts/
ArgoGenesisRewardPool.sol#274)
Reentrancy in TridentRewardPool.emergencyWithdraw(uint256) (contracts/
TridentRewardPool.sol#234-242):
        External calls:
        - pool.token.safeTransfer(msg.sender,_amount) (contracts/
TridentRewardPool.sol#240)
        Event emitted after the call(s):
        - EmergencyWithdraw(msg.sender,_pid,_amount) (contracts/
TridentRewardPool.sol#241)
Reentrancy in Treasury.redeemBonds(uint256,uint256) (contracts/Treasury.sol#435-459):
        External calls:
        - IBasisAsset(bud).burnFrom(msg.sender,_bondAmount) (contracts/
Treasury.sol#453)
        - IERC20(ARGO).safeTransfer(msg.sender,_ARGOAmount) (contracts/
Treasury.sol#454)
        - _updateARGOPrice() (contracts/Treasury.sol#456)
                - IOracle(ARGOOracle).update() (contracts/Treasury.sol#393)
        Event emitted after the call(s):
        - RedeemedBonds(msg.sender,_ARGOAmount,_bondAmount) (contracts/
Treasury.sol#458)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3

ArgoGenesisRewardPool.checkPoolDuplicate(IERC20) (contracts/
ArgoGenesisRewardPool.sol#92-97) uses timestamp for comparisons
        Dangerous comparisons:
        - pid < length (contracts/ArgoGenesisRewardPool.sol#94)
        - require(bool,string)(poolInfo[pid].token != _token,ArgoGenesisPool: existing
pool?) (contracts/ArgoGenesisRewardPool.sol#95)
ArgoGenesisRewardPool.add(uint256,IERC20,bool,uint256) (contracts/
ArgoGenesisRewardPool.sol#100-138) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp < poolStartTime (contracts/ArgoGenesisRewardPool.sol#110)
        - _lastRewardTime == 0 (contracts/ArgoGenesisRewardPool.sol#112)
        - _lastRewardTime < poolStartTime (contracts/ArgoGenesisRewardPool.sol#115)
        - _lastRewardTime == 0 || _lastRewardTime < block.timestamp (contracts/
ArgoGenesisRewardPool.sol#121)
        - _isStarted = (_lastRewardTime <= poolStartTime) || (_lastRewardTime <=
block.timestamp) (contracts/ArgoGenesisRewardPool.sol#125-127)
ArgoGenesisRewardPool.getGeneratedReward(uint256,uint256) (contracts/
ArgoGenesisRewardPool.sol#153-164) uses timestamp for comparisons
        Dangerous comparisons:
        - _fromTime >= _toTime (contracts/ArgoGenesisRewardPool.sol#154)
        - _toTime >= poolEndTime (contracts/ArgoGenesisRewardPool.sol#155)
        - _fromTime >= poolEndTime (contracts/ArgoGenesisRewardPool.sol#156)
        - _fromTime <= poolStartTime (contracts/ArgoGenesisRewardPool.sol#157)
        - _toTime <= poolStartTime (contracts/ArgoGenesisRewardPool.sol#160)
        - _fromTime <= poolStartTime (contracts/ArgoGenesisRewardPool.sol#161)
ArgoGenesisRewardPool.pendingARGO(uint256,address) (contracts/
ArgoGenesisRewardPool.sol#167-178) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > pool.lastRewardTime && tokenSupply != 0 (contracts/
ArgoGenesisRewardPool.sol#172)
ArgoGenesisRewardPool.massUpdatePools() (contracts/ArgoGenesisRewardPool.sol#181-186)
uses timestamp for comparisons
        Dangerous comparisons:
        - pid < length (contracts/ArgoGenesisRewardPool.sol#183)
ArgoGenesisRewardPool.updatePool(uint256) (contracts/ArgoGenesisRewardPool.sol#189-209)
uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp <= pool.lastRewardTime (contracts/
ArgoGenesisRewardPool.sol#191)
ArgoGenesisRewardPool.governanceRecoverUnsupported(IERC20,uint256,address) (contracts/
ArgoGenesisRewardPool.sol#293-304) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp < poolEndTime + 7776000 (contracts/
ArgoGenesisRewardPool.sol#294)
Trident.unclaimedDevFund() (contracts/Trident.sol#51-56) uses timestamp for comparisons
        Dangerous comparisons:
        - _now > endTime (contracts/Trident.sol#53)
        - devFundLastClaimed >= _now (contracts/Trident.sol#54)
TridentRewardPool.checkPoolDuplicate(IERC20) (contracts/TridentRewardPool.sol#73-78)
uses timestamp for comparisons
        Dangerous comparisons:
```

```
        - pid < length (contracts/TridentRewardPool.sol#75)
        - require(bool,string)(poolInfo[pid].token != _token,TridentRewardPool:
existing pool?) (contracts/TridentRewardPool.sol#76)
TridentRewardPool.add(uint256,IERC20,bool,uint256) (contracts/
TridentRewardPool.sol#81-119) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp < poolStartTime (contracts/TridentRewardPool.sol#91)
        - _lastRewardTime == 0 (contracts/TridentRewardPool.sol#93)
        - _lastRewardTime < poolStartTime (contracts/TridentRewardPool.sol#96)
        - _lastRewardTime == 0 || _lastRewardTime < block.timestamp (contracts/
TridentRewardPool.sol#102)
        - _isStarted = (_lastRewardTime <= poolStartTime) || (_lastRewardTime <=
block.timestamp) (contracts/TridentRewardPool.sol#106-108)
TridentRewardPool.getGeneratedReward(uint256,uint256) (contracts/
TridentRewardPool.sol#134-145) uses timestamp for comparisons
        Dangerous comparisons:
        - _fromTime >= _toTime (contracts/TridentRewardPool.sol#135)
        - _toTime >= poolEndTime (contracts/TridentRewardPool.sol#136)
        - _fromTime >= poolEndTime (contracts/TridentRewardPool.sol#137)
        - _fromTime <= poolStartTime (contracts/TridentRewardPool.sol#138)
        - _toTime <= poolStartTime (contracts/TridentRewardPool.sol#141)
        - _fromTime <= poolStartTime (contracts/TridentRewardPool.sol#142)
TridentRewardPool.pendingShare(uint256,address) (contracts/
TridentRewardPool.sol#148-159) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > pool.lastRewardTime && tokenSupply != 0 (contracts/
TridentRewardPool.sol#153)
TridentRewardPool.massUpdatePools() (contracts/TridentRewardPool.sol#162-167) uses
timestamp for comparisons
        Dangerous comparisons:
        - pid < length (contracts/TridentRewardPool.sol#164)
TridentRewardPool.updatePool(uint256) (contracts/TridentRewardPool.sol#170-190) uses
timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp <= pool.lastRewardTime (contracts/TridentRewardPool.sol#172)
TridentRewardPool.governanceRecoverUnsupported(IERC20,uint256,address) (contracts/
TridentRewardPool.sol#260-271) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp < poolEndTime + 7776000 (contracts/TridentRewardPool.sol#261)
UniswapV2OracleLibrary.currentCumulativePrices(address) (contracts/lib/
UniswapV2OracleLibrary.sol#18-42) uses timestamp for comparisons
```

```
        Dangerous comparisons:
        - blockTimestampLast != blockTimestamp (contracts/lib/
UniswapV2OracleLibrary.sol#33)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp

ArgoGenesisRewardPool.updatePool(uint256) (contracts/ArgoGenesisRewardPool.sol#189-209)
has costly operations inside a loop:
        - totalAllocPoint = totalAllocPoint.add(pool.allocPoint) (contracts/
ArgoGenesisRewardPool.sol#201)
TridentRewardPool.updatePool(uint256) (contracts/TridentRewardPool.sol#170-190) has
costly operations inside a loop:
        - totalAllocPoint = totalAllocPoint.add(pool.allocPoint) (contracts/
TridentRewardPool.sol#182)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-
operations-inside-a-loop

Argo._getArgoPrice() (contracts/Argo.sol#47-53) is never used and should be removed
Babylonian.sqrt(uint256) (contracts/lib/Babylonian.sol#6-18) is never used and should
be removed
FixedPoint.decode(FixedPoint.uq112x112) (contracts/lib/FixedPoint.sol#57-59) is never
used and should be removed
FixedPoint.div(FixedPoint.uq112x112,uint112) (contracts/lib/FixedPoint.sol#36-39) is
never used and should be removed
FixedPoint.encode(uint112) (contracts/lib/FixedPoint.sol#26-28) is never used and
should be removed
FixedPoint.encode144(uint144) (contracts/lib/FixedPoint.sol#31-33) is never used and
should be removed
FixedPoint.reciprocal(FixedPoint.uq112x112) (contracts/lib/FixedPoint.sol#67-70) is
never used and should be removed
FixedPoint.sqrt(FixedPoint.uq112x112) (contracts/lib/FixedPoint.sol#73-75) is never
used and should be removed
SafeMath8.add(uint8,uint8) (contracts/lib/SafeMath8.sol#29-34) is never used and should
be removed
SafeMath8.div(uint8,uint8) (contracts/lib/SafeMath8.sol#103-105) is never used and
should be removed
SafeMath8.div(uint8,uint8,string) (contracts/lib/SafeMath8.sol#119-125) is never used
and should be removed
SafeMath8.mod(uint8,uint8) (contracts/lib/SafeMath8.sol#139-141) is never used and
should be removed
SafeMath8.mod(uint8,uint8,string) (contracts/lib/SafeMath8.sol#155-158) is never used
```

```
and should be removed
SafeMath8.mul(uint8,uint8) (contracts/lib/SafeMath8.sol#77-89) is never used and should
be removed
SafeMath8.sub(uint8,uint8) (contracts/lib/SafeMath8.sol#46-48) is never used and should
be removed
SafeMath8.sub(uint8,uint8,string) (contracts/lib/SafeMath8.sol#60-65) is never used and
should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (contracts/Argo.sol#3) allows old versions
Pragma version^0.8.0 (contracts/ArgoGenesisRewardPool.sol#3) allows old versions
Pragma version^0.8.0 (contracts/Cove.sol#3) allows old versions
Pragma version^0.8.0 (contracts/Oar.sol#3) allows old versions
Pragma version^0.8.0 (contracts/Oracle.sol#3) allows old versions
Pragma version^0.8.0 (contracts/Treasury.sol#3) allows old versions
Pragma version^0.8.0 (contracts/Trident.sol#3) allows old versions
Pragma version^0.8.0 (contracts/TridentRewardPool.sol#3) allows old versions
Pragma version^0.8.0 (contracts/interfaces/IBasisAsset.sol#3) allows old versions
Pragma version^0.8.0 (contracts/interfaces/IMasonry.sol#3) allows old versions
Pragma version^0.8.0 (contracts/interfaces/IOracle.sol#3) allows old versions
Pragma version^0.8.0 (contracts/interfaces/ITreasury.sol#3) allows old versions
Pragma version^0.8.0 (contracts/interfaces/IUniswapV2Pair.sol#3) allows old versions
Pragma version^0.8.0 (contracts/lib/Babylonian.sol#3) allows old versions
Pragma version^0.8.0 (contracts/lib/FixedPoint.sol#3) allows old versions
Pragma version^0.8.0 (contracts/lib/SafeMath8.sol#3) allows old versions
Pragma version^0.8.0 (contracts/lib/UniswapV2OracleLibrary.sol#3) allows old versions
Pragma version^0.8.0 (contracts/owner/Operator.sol#3) allows old versions
Pragma version^0.8.0 (contracts/utils/ContractGuard.sol#3) allows old versions
Pragma version^0.8.0 (contracts/utils/Epoch.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity

Argo (contracts/Argo.sol#15-121) should inherit from IBasisAsset (contracts/interfaces/
IBasisAsset.sol#5-17)
Oar (contracts/Oar.sol#9-37) should inherit from IBasisAsset (contracts/interfaces/
IBasisAsset.sol#5-17)
Oracle (contracts/Oracle.sol#15-94) should inherit from IOracle (contracts/interfaces/
IOracle.sol#5-12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-
inheritance
```

```
Parameter Argo.setArgoOracle(address)._argoOracle (contracts/Argo.sol#55) is not in
mixedCase
Parameter Argo.distributeInitialOffering(address)._offeringContract (contracts/
Argo.sol#93) is not in mixedCase
Parameter Argo.distributeReward(address)._genesisPool (contracts/Argo.sol#106) is not
in mixedCase
Parameter Argo.governanceRecoverUnsupported(IERC20,uint256,address)._token (contracts/
Argo.sol#115) is not in mixedCase
Parameter Argo.governanceRecoverUnsupported(IERC20,uint256,address)._amount (contracts/
Argo.sol#116) is not in mixedCase
Parameter Argo.governanceRecoverUnsupported(IERC20,uint256,address)._to (contracts/
Argo.sol#117) is not in mixedCase
Parameter ArgoGenesisRewardPool.checkPoolDuplicate(IERC20)._token (contracts/
ArgoGenesisRewardPool.sol#92) is not in mixedCase
Parameter ArgoGenesisRewardPool.add(uint256,IERC20,bool,uint256)._allocPoint (contracts/
ArgoGenesisRewardPool.sol#101) is not in mixedCase
Parameter ArgoGenesisRewardPool.add(uint256,IERC20,bool,uint256)._token (contracts/
ArgoGenesisRewardPool.sol#102) is not in mixedCase
Parameter ArgoGenesisRewardPool.add(uint256,IERC20,bool,uint256)._withUpdate (contracts/
ArgoGenesisRewardPool.sol#103) is not in mixedCase
Parameter ArgoGenesisRewardPool.add(uint256,IERC20,bool,uint256)._lastRewardTime
(contracts/ArgoGenesisRewardPool.sol#104) is not in mixedCase
Parameter ArgoGenesisRewardPool.set(uint256,uint256)._pid (contracts/
ArgoGenesisRewardPool.sol#141) is not in mixedCase
Parameter ArgoGenesisRewardPool.set(uint256,uint256)._allocPoint (contracts/
ArgoGenesisRewardPool.sol#141) is not in mixedCase
Parameter ArgoGenesisRewardPool.getGeneratedReward(uint256,uint256)._fromTime
(contracts/ArgoGenesisRewardPool.sol#153) is not in mixedCase
Parameter ArgoGenesisRewardPool.getGeneratedReward(uint256,uint256)._toTime (contracts/
ArgoGenesisRewardPool.sol#153) is not in mixedCase
Parameter ArgoGenesisRewardPool.pendingARGO(uint256,address)._pid (contracts/
ArgoGenesisRewardPool.sol#167) is not in mixedCase
Parameter ArgoGenesisRewardPool.pendingARGO(uint256,address)._user (contracts/
ArgoGenesisRewardPool.sol#167) is not in mixedCase
Parameter ArgoGenesisRewardPool.updatePool(uint256)._pid (contracts/
ArgoGenesisRewardPool.sol#189) is not in mixedCase
Parameter ArgoGenesisRewardPool.deposit(uint256,uint256)._pid (contracts/
ArgoGenesisRewardPool.sol#212) is not in mixedCase
Parameter ArgoGenesisRewardPool.deposit(uint256,uint256)._amount (contracts/
ArgoGenesisRewardPool.sol#212) is not in mixedCase
Parameter ArgoGenesisRewardPool.withdraw(uint256,uint256)._pid (contracts/
ArgoGenesisRewardPool.sol#247) is not in mixedCase
Parameter ArgoGenesisRewardPool.withdraw(uint256,uint256)._amount (contracts/
ArgoGenesisRewardPool.sol#247) is not in mixedCase
Parameter ArgoGenesisRewardPool.emergencyWithdraw(uint256)._pid (contracts/
ArgoGenesisRewardPool.sol#267) is not in mixedCase
Parameter ArgoGenesisRewardPool.safeArgoTransfer(address,uint256)._to (contracts/
ArgoGenesisRewardPool.sol#278) is not in mixedCase
Parameter ArgoGenesisRewardPool.safeArgoTransfer(address,uint256)._amount (contracts/
ArgoGenesisRewardPool.sol#278) is not in mixedCase
Parameter ArgoGenesisRewardPool.setOperator(address)._operator (contracts/
ArgoGenesisRewardPool.sol#289) is not in mixedCase
Parameter
ArgoGenesisRewardPool.governanceRecoverUnsupported(IERC20,uint256,address)._token
(contracts/ArgoGenesisRewardPool.sol#293) is not in mixedCase
Parameter Cove.initialize(IERC20,IERC20,ITreasury)._argo (contracts/Cove.sol#119) is
not in mixedCase
Parameter Cove.initialize(IERC20,IERC20,ITreasury)._trident (contracts/Cove.sol#120) is
not in mixedCase
Parameter Cove.initialize(IERC20,IERC20,ITreasury)._treasury (contracts/Cove.sol#121)
is not in mixedCase
Parameter Cove.setOperator(address)._operator (contracts/Cove.sol#138) is not in
mixedCase
Parameter Cove.setLockUp(uint256,uint256)._withdrawLockupEpochs (contracts/
Cove.sol#142) is not in mixedCase
Parameter Cove.setLockUp(uint256,uint256)._rewardLockupEpochs (contracts/Cove.sol#142)
is not in mixedCase
Parameter Cove.governanceRecoverUnsupported(IERC20,uint256,address)._token (contracts/
Cove.sol#252) is not in mixedCase
Parameter Cove.governanceRecoverUnsupported(IERC20,uint256,address)._amount (contracts/
Cove.sol#252) is not in mixedCase
Parameter Cove.governanceRecoverUnsupported(IERC20,uint256,address)._to (contracts/
Cove.sol#252) is not in mixedCase
Parameter Oracle.consult(address,uint256)._token (contracts/Oracle.sol#73) is not in
mixedCase
Parameter Oracle.consult(address,uint256)._amountIn (contracts/Oracle.sol#73) is not in
mixedCase
Parameter Oracle.twap(address,uint256)._token (contracts/Oracle.sol#82) is not in
mixedCase
Parameter Oracle.twap(address,uint256)._amountIn (contracts/Oracle.sol#82) is not in
mixedCase
Parameter Treasury.initialize(address,address,address,address,address)._ARGO (contracts/
Treasury.sol#238) is not in mixedCase
Parameter Treasury.initialize(address,address,address,address,address)._bud (contracts/
Treasury.sol#239) is not in mixedCase
Parameter Treasury.initialize(address,address,address,address,address)._petal
(contracts/Treasury.sol#240) is not in mixedCase
Parameter Treasury.initialize(address,address,address,address,address)._ARGOOracle
(contracts/Treasury.sol#241) is not in mixedCase
Parameter Treasury.initialize(address,address,address,address,address)._garden
(contracts/Treasury.sol#242) is not in mixedCase
Parameter Treasury.setOperator(address)._operator (contracts/Treasury.sol#280) is not
in mixedCase
Parameter Treasury.setGarden(address)._garden (contracts/Treasury.sol#284) is not in
mixedCase
Parameter Treasury.setARGOOracle(address)._ARGOOracle (contracts/Treasury.sol#288) is
not in mixedCase
Parameter Treasury.setARGOPriceCeiling(uint256)._ARGOPriceCeiling (contracts/
Treasury.sol#292) is not in mixedCase
Parameter Treasury.setMaxSupplyExpansionPercents(uint256)._maxSupplyExpansionPercent
(contracts/Treasury.sol#297) is not in mixedCase
Parameter Treasury.setSupplyTiersEntry(uint8,uint256)._index (contracts/
Treasury.sol#302) is not in mixedCase
Parameter Treasury.setSupplyTiersEntry(uint8,uint256)._value (contracts/
Treasury.sol#302) is not in mixedCase
Parameter Treasury.setMaxExpansionTiersEntry(uint8,uint256)._index (contracts/
Treasury.sol#315) is not in mixedCase
Parameter Treasury.setMaxExpansionTiersEntry(uint8,uint256)._value (contracts/
Treasury.sol#315) is not in mixedCase
Parameter Treasury.setBondDepletionFloorPercent(uint256)._bondDepletionFloorPercent
(contracts/Treasury.sol#323) is not in mixedCase
Parameter Treasury.setMaxSupplyContractionPercent(uint256)._maxSupplyContractionPercent
(contracts/Treasury.sol#328) is not in mixedCase
Parameter Treasury.setMaxDebtRatioPercent(uint256)._maxDebtRatioPercent (contracts/
Treasury.sol#333) is not in mixedCase
Parameter Treasury.setBootstrap(uint256,uint256)._bootstrapEpochs (contracts/
Treasury.sol#338) is not in mixedCase
Parameter Treasury.setBootstrap(uint256,uint256)._bootstrapSupplyExpansionPercent
(contracts/Treasury.sol#338) is not in mixedCase
Parameter Treasury.setExtraFunds(address,uint256,address,uint256)._daoFund (contracts/
Treasury.sol#346) is not in mixedCase
Parameter Treasury.setExtraFunds(address,uint256,address,uint256)._daoFundSharedPercent
(contracts/Treasury.sol#347) is not in mixedCase
```

```
Parameter Treasury.setExtraFunds(address,uint256,address,uint256)._devFund (contracts/
Treasury.sol#348) is not in mixedCase
Parameter Treasury.setExtraFunds(address,uint256,address,uint256)._devFundSharedPercent
(contracts/Treasury.sol#349) is not in mixedCase
Parameter Treasury.setMaxDiscountRate(uint256)._maxDiscountRate (contracts/
Treasury.sol#361) is not in mixedCase
Parameter Treasury.setMaxPremiumRate(uint256)._maxPremiumRate (contracts/
Treasury.sol#365) is not in mixedCase
Parameter Treasury.setDiscountPercent(uint256)._discountPercent (contracts/
Treasury.sol#369) is not in mixedCase
Parameter Treasury.setPremiumThreshold(uint256)._premiumThreshold (contracts/
Treasury.sol#374) is not in mixedCase
Parameter Treasury.setPremiumPercent(uint256)._premiumPercent (contracts/
Treasury.sol#380) is not in mixedCase
Parameter Treasury.setMintingFactorForPayingDebt(uint256)._mintingFactorForPayingDebt
(contracts/Treasury.sol#385) is not in mixedCase
Parameter Treasury.buyBonds(uint256,uint256)._ARGOAmount (contracts/Treasury.sol#406)
is not in mixedCase
Parameter Treasury.redeemBonds(uint256,uint256)._bondAmount (contracts/
Treasury.sol#435) is not in mixedCase
Parameter Treasury.governanceRecoverUnsupported(IERC20,uint256,address)._token
(contracts/Treasury.sol#539) is not in mixedCase
Parameter Treasury.governanceRecoverUnsupported(IERC20,uint256,address)._amount
(contracts/Treasury.sol#540) is not in mixedCase
Parameter Treasury.governanceRecoverUnsupported(IERC20,uint256,address)._to (contracts/
Treasury.sol#541) is not in mixedCase
Parameter Treasury.gardenSetOperator(address)._operator (contracts/Treasury.sol#550) is
not in mixedCase
Parameter Treasury.gardenSetLockUp(uint256,uint256)._withdrawLockupEpochs (contracts/
Treasury.sol#554) is not in mixedCase
Parameter Treasury.gardenSetLockUp(uint256,uint256)._rewardLockupEpochs (contracts/
Treasury.sol#554) is not in mixedCase
Parameter Treasury.gardenGovernanceRecoverUnsupported(address,uint256,address)._token
(contracts/Treasury.sol#563) is not in mixedCase
Parameter Treasury.gardenGovernanceRecoverUnsupported(address,uint256,address)._amount
(contracts/Treasury.sol#564) is not in mixedCase
Parameter Treasury.gardenGovernanceRecoverUnsupported(address,uint256,address)._to
(contracts/Treasury.sol#565) is not in mixedCase
Variable Treasury.ARGO (contracts/Treasury.sol#50) is not in mixedCase
Variable Treasury.ARGOOracle (contracts/Treasury.sol#55) is not in mixedCase
Variable Treasury.ARGOPriceOne (contracts/Treasury.sol#58) is not in mixedCase
Variable Treasury.ARGOPriceCeiling (contracts/Treasury.sol#59) is not in mixedCase
Parameter Trident.setDevFund(address)._devFund (contracts/Trident.sol#45) is not in
mixedCase
Parameter Trident.distributeReward(address)._farmingIncentiveFund (contracts/
Trident.sol#72) is not in mixedCase
Parameter Trident.governanceRecoverUnsupported(IERC20,uint256,address)._token
(contracts/Trident.sol#84) is not in mixedCase
Parameter Trident.governanceRecoverUnsupported(IERC20,uint256,address)._amount
(contracts/Trident.sol#85) is not in mixedCase
Parameter Trident.governanceRecoverUnsupported(IERC20,uint256,address)._to (contracts/
Trident.sol#86) is not in mixedCase
Parameter TridentRewardPool.checkPoolDuplicate(IERC20)._token (contracts/
TridentRewardPool.sol#73) is not in mixedCase
Parameter TridentRewardPool.add(uint256,IERC20,bool,uint256)._allocPoint (contracts/
TridentRewardPool.sol#82) is not in mixedCase
Parameter TridentRewardPool.add(uint256,IERC20,bool,uint256)._token (contracts/
TridentRewardPool.sol#83) is not in mixedCase
Parameter TridentRewardPool.add(uint256,IERC20,bool,uint256)._withUpdate (contracts/
TridentRewardPool.sol#84) is not in mixedCase
Parameter TridentRewardPool.add(uint256,IERC20,bool,uint256)._lastRewardTime (contracts/
TridentRewardPool.sol#85) is not in mixedCase
Parameter TridentRewardPool.set(uint256,uint256)._pid (contracts/
TridentRewardPool.sol#122) is not in mixedCase
Parameter TridentRewardPool.set(uint256,uint256)._allocPoint (contracts/
TridentRewardPool.sol#122) is not in mixedCase
Parameter TridentRewardPool.getGeneratedReward(uint256,uint256)._fromTime (contracts/
TridentRewardPool.sol#134) is not in mixedCase
Parameter TridentRewardPool.getGeneratedReward(uint256,uint256)._toTime (contracts/
TridentRewardPool.sol#134) is not in mixedCase
Parameter TridentRewardPool.pendingShare(uint256,address)._pid (contracts/
TridentRewardPool.sol#148) is not in mixedCase
Parameter TridentRewardPool.pendingShare(uint256,address)._user (contracts/
TridentRewardPool.sol#148) is not in mixedCase
Parameter TridentRewardPool.updatePool(uint256)._pid (contracts/
TridentRewardPool.sol#170) is not in mixedCase
Parameter TridentRewardPool.deposit(uint256,uint256)._pid (contracts/
TridentRewardPool.sol#193) is not in mixedCase
Parameter TridentRewardPool.deposit(uint256,uint256)._amount (contracts/
TridentRewardPool.sol#193) is not in mixedCase
Parameter TridentRewardPool.withdraw(uint256,uint256)._pid (contracts/
TridentRewardPool.sol#214) is not in mixedCase
```

```
Parameter TridentRewardPool.withdraw(uint256,uint256)._amount (contracts/
TridentRewardPool.sol#214) is not in mixedCase
Parameter TridentRewardPool.emergencyWithdraw(uint256)._pid (contracts/
TridentRewardPool.sol#234) is not in mixedCase
Parameter TridentRewardPool.safeTridentTransfer(address,uint256)._to (contracts/
TridentRewardPool.sol#245) is not in mixedCase
Parameter TridentRewardPool.safeTridentTransfer(address,uint256)._amount (contracts/
TridentRewardPool.sol#245) is not in mixedCase
Parameter TridentRewardPool.setOperator(address)._operator (contracts/
TridentRewardPool.sol#256) is not in mixedCase
Parameter TridentRewardPool.governanceRecoverUnsupported(IERC20,uint256,address)._token
(contracts/TridentRewardPool.sol#260) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (contracts/interfaces/IUniswapV2Pair.sol#31)
is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (contracts/interfaces/IUniswapV2Pair.sol#33)
is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (contracts/interfaces/
IUniswapV2Pair.sol#52) is not in mixedCase
Struct FixedPoint.uq112x112 (contracts/lib/FixedPoint.sol#11-13) is not in CapWords
Struct FixedPoint.uq144x112 (contracts/lib/FixedPoint.sol#17-19) is not in CapWords
Parameter Epoch.setPeriod(uint256)._period (contracts/utils/Epoch.sol#79) is not in
mixedCase
Parameter Epoch.setEpoch(uint256)._epoch (contracts/utils/Epoch.sol#84) is not in
mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions

Variable Oracle.price0Average (contracts/Oracle.sol#30) is too similar to
Oracle.price1Average (contracts/Oracle.sol#31)
Variable Oracle.twap(address,uint256).price0Cumulative (contracts/Oracle.sol#83) is too
similar to Oracle.twap(address,uint256).price1Cumulative (contracts/Oracle.sol#83)
Variable Oracle.twap(address,uint256).price0Cumulative (contracts/Oracle.sol#83) is too
similar to Oracle.update().price1Cumulative (contracts/Oracle.sol#52)
Variable Oracle.update().price0Cumulative (contracts/Oracle.sol#52) is too similar to
Oracle.update().price1Cumulative (contracts/Oracle.sol#52)
Variable Oracle.price0CumulativeLast (contracts/Oracle.sol#28) is too similar to
Oracle.price1CumulativeLast (contracts/Oracle.sol#29)
Variable Oracle.update().price0Cumulative (contracts/Oracle.sol#52) is too similar to
Oracle.twap(address,uint256).price1Cumulative (contracts/Oracle.sol#83)
Variable Treasury.setExtraFunds(address,uint256,address,uint256)._daoFundSharedPercent
(contracts/Treasury.sol#347) is too similar to
```

```
Treasury.setExtraFunds(address,uint256,address,uint256)._devFundSharedPercent
(contracts/Treasury.sol#349)
Variable UniswapV2OracleLibrary.currentCumulativePrices(address).price0Cumulative
(contracts/lib/UniswapV2OracleLibrary.sol#22) is too similar to
UniswapV2OracleLibrary.currentCumulativePrices(address).price1Cumulative (contracts/lib/
UniswapV2OracleLibrary.sol#23)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-
are-too-similar

Treasury.initialize(address,address,address,address,address) (contracts/
Treasury.sol#237-278) uses literals with too many digits:
        - supplyTiers = (0,500000000000000000000000,1000000000000000000000000,1500000000
000000000000000,2000000000000000000000000,5000000000000000000000000,10000000000000000000
000000,20000000000000000000000000,50000000000000000000000000) (contracts/
Treasury.sol#255)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-
digits

ArgoGenesisRewardPool.argoPerSecond (contracts/ArgoGenesisRewardPool.sol#56) should be
constant
ArgoGenesisRewardPool.runningTime (contracts/ArgoGenesisRewardPool.sol#57) should be
constant
TridentRewardPool.runningTime (contracts/TridentRewardPool.sol#51) should be constant
TridentRewardPool.tSharePerSecond (contracts/TridentRewardPool.sol#50) should be
constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-constant

setArgoOracle(address) should be declared external:
        - Argo.setArgoOracle(address) (contracts/Argo.sol#55-58)
mint(address,uint256) should be declared external:
        - Argo.mint(address,uint256) (contracts/Argo.sol#66-72)
set(uint256,uint256) should be declared external:
        - ArgoGenesisRewardPool.set(uint256,uint256) (contracts/
ArgoGenesisRewardPool.sol#141-150)
deposit(uint256,uint256) should be declared external:
        - ArgoGenesisRewardPool.deposit(uint256,uint256) (contracts/
ArgoGenesisRewardPool.sol#212-244)
withdraw(uint256,uint256) should be declared external:
        - ArgoGenesisRewardPool.withdraw(uint256,uint256) (contracts/
ArgoGenesisRewardPool.sol#247-264)
```

```
emergencyWithdraw(uint256) should be declared external:
	- ArgoGenesisRewardPool.emergencyWithdraw(uint256) (contracts/
ArgoGenesisRewardPool.sol#267-275)
initialize(IERC20,IERC20,ITreasury) should be declared external:
	- Cove.initialize(IERC20,IERC20,ITreasury) (contracts/Cove.sol#118-136)
rewardPerShare() should be declared external:
	- Cove.rewardPerShare() (contracts/Cove.sol#190-192)
mint(address,uint256) should be declared external:
	- Oar.mint(address,uint256) (contracts/Oar.sol#21-27)
isInitialized() should be declared external:
	- Treasury.isInitialized() (contracts/Treasury.sol#144-146)
getARGOUpdatedPrice() should be declared external:
	- Treasury.getARGOUpdatedPrice() (contracts/Treasury.sol#162-168)
getReserve() should be declared external:
	- Treasury.getReserve() (contracts/Treasury.sol#171-173)
getBurnableARGOLeft() should be declared external:
	- Treasury.getBurnableARGOLeft() (contracts/Treasury.sol#175-187)
getRedeemableBonds() should be declared external:
	- Treasury.getRedeemableBonds() (contracts/Treasury.sol#189-198)
initialize(address,address,address,address,address) should be declared external:
	- Treasury.initialize(address,address,address,address,address) (contracts/
Treasury.sol#237-278)
set(uint256,uint256) should be declared external:
	- TridentRewardPool.set(uint256,uint256) (contracts/
TridentRewardPool.sol#122-131)
deposit(uint256,uint256) should be declared external:
	- TridentRewardPool.deposit(uint256,uint256) (contracts/
TridentRewardPool.sol#193-211)
withdraw(uint256,uint256) should be declared external:
	- TridentRewardPool.withdraw(uint256,uint256) (contracts/
TridentRewardPool.sol#214-231)
emergencyWithdraw(uint256) should be declared external:
	- TridentRewardPool.emergencyWithdraw(uint256) (contracts/
TridentRewardPool.sol#234-242)
isOperator() should be declared external:
	- Operator.isOperator() (contracts/owner/Operator.sol#27-29)
transferOperator(address) should be declared external:
	- Operator.transferOperator(address) (contracts/owner/Operator.sol#31-33)
getCurrentEpoch() should be declared external:
	- Epoch.getCurrentEpoch() (contracts/utils/Epoch.sol#57-59)
getPeriod() should be declared external:
        - Epoch.getPeriod() (contracts/utils/Epoch.sol#61-63)
getStartTime() should be declared external:
        - Epoch.getStartTime() (contracts/utils/Epoch.sol#65-67)
getLastEpochTime() should be declared external:
        - Epoch.getLastEpochTime() (contracts/utils/Epoch.sol#69-71)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external
. analyzed (32 contracts with 77 detectors), 290 result(s) found
```

0x Guard