



Smart contracts security assessment

Final report

[Tariff: Standard](#)

Ghost DeFi

April 2022



0xguard.com



hello@0xguard.com

Contents

1. Introduction	3
2. Contracts checked	3
3. Procedure	4
4. Classification of issue severity	5
5. Issues	5
6. Conclusion	7
7. Disclaimer	8
8. Slither output	9

Introduction

This report has been prepared for the Ghost DeFi team upon their request.

The audited project is a fork of the Tomb Finance Project. The code is available in the Github [repository](#). The code was checked in [69b79eb](#) commit.

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 Ghost DeFi are available at the official website: <https://www.ghostdefi.io/>.

Name	Ghost DeFi
Audit date	2022-04-06 - 2022-04-11
Language	Solidity
Platform	Fantom Network

Contracts checked

Name	Address
GhostToken	https://github.com/ghostdefiio/ghostdefi-contracts/ blob/69b79ebcb16c3947cf6ee282e0ad79a731c1c9bf/ GhostToken.sol
GhostShares	https://github.com/ghostdefiio/ghostdefi-contracts/ blob/69b79ebcb16c3947cf6ee282e0ad79a731c1c9bf/ GhostShares.sol

GBond	https://github.com/ghostdefiio/ghostdefi-contracts/ blob/69b79ebcb16c3947cf6ee282e0ad79a731c1c9bf/ GBond.sol
GshareRewardPool	https://github.com/ghostdefiio/ghostdefi-contracts/ blob/69b79ebcb16c3947cf6ee282e0ad79a731c1c9bf/ GshareRewardPool.sol
Oracle	https://github.com/ghostdefiio/ghostdefi-contracts/ blob/69b79ebcb16c3947cf6ee282e0ad79a731c1c9bf/ Oracle.sol
Masonry	https://github.com/ghostdefiio/ghostdefi-contracts/ blob/69b79ebcb16c3947cf6ee282e0ad79a731c1c9bf/ Masonry.sol
Treasury	https://github.com/ghostdefiio/ghostdefi-contracts/ blob/69b79ebcb16c3947cf6ee282e0ad79a731c1c9bf/ Treasury.sol
Multiple contracts	

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

1. Contract ownership (PARTIALLY FIXED) (Multiple contracts)

1) The projects owner can change the `taxRate`, `buyTax`, `sellTax` in GhostToken up to 100% in the `setTaxRate()`, `setBuyTax()`, `setSellTax()` functions if owner change the `taxOffice` in the `setTaxOffice()` function to a compromised address.

2) Operator can change `taxTiersTwaps` and `taxTiersRate` up to 100% in GhostToken in `setTaxTiersTwap()` and `setTaxTiersRate()` functions.

3) Operator can change the addresses of funds in the Treasury contract using the function

`setExtraFunds()` function. The `daoFund` can be withdrawn if the operator account is compromised.

Recommendation: There are a large number of functions with the `onlyOperator()` modifier, there is a possibility that the operator can be compromised. It is recommended to create multiple roles for different kinds of functions to reduce the operator's problem. It is also recommended to add a time delay to the especially important set functions using the [TimelockController](#). We also recommend that you look through the entire codebase to find functions that are dangerous for you as the owner of the project (mainly set functions), if there are any, then add a call to them via multisig wallet. This will help avoid the issue of owner compromise.

Update: The Ghost token was deployed at address [0x4F0d07209529C714eA5CaFad90ed1494A3bB800a](#). The ownership of the token and the taxOffice were set to a zero address (renounced) after deployment. The points 1 and 2 are now resolved.

Low severity issues

1. Few events (Multiple contracts)

Many set functions from contracts are missing events when changing important values in the contract.

Recommendation: Create events for these set functions.

Conclusion

0 high, 1 medium, 1 low severity issues were found.

The Ghost DeFi Project was compared with the Tomb Project. Ghost DeFi has changed the implementation of **GhostToken** contract.

The **transfer()** function has been overloaded in the **GhostToken** contract, and fees have been added for buying and selling on Uniswap-like exchanges.

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.

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.

Slither output

UniswapV2OracleLibrary.currentBlockTimestamp() (contracts/Oracle.sol#348-350) uses a weak PRNG: "uint32(block.timestamp % 2 ** 32) (contracts/Oracle.sol#349)"

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#weak-prng>

Reentrancy in GshareRewardPool.deposit(uint256,uint256) (contracts/GshareRewardPool.sol#764-782):

External calls:

- [- safeTShareTransfer(_sender,_pending) (contracts/GshareRewardPool.sol#772)
- [- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/GshareRewardPool.sol#561)
- [- tshare.safeTransfer(_to,_tshareBal) (contracts/GshareRewardPool.sol#820)
- [- (success,returndata) = target.call{value: value}(data) (contracts/GshareRewardPool.sol#123)
- [- tshare.safeTransfer(_to,_amount) (contracts/GshareRewardPool.sol#822)
- [- pool.token.safeTransferFrom(_sender,address(this),_amount) (contracts/GshareRewardPool.sol#777)

External calls sending eth:

- [- safeTShareTransfer(_sender,_pending) (contracts/GshareRewardPool.sol#772)
- [- (success,returndata) = target.call{value: value}(data) (contracts/GshareRewardPool.sol#123)

State variables written after the call(s):

- [- user.amount = user.amount.add(_amount) (contracts/GshareRewardPool.sol#778)
- [- user.rewardDebt = user.amount.mul(pool.accTSharePerShare).div(1e18) (contracts/GshareRewardPool.sol#780)

Reentrancy in GshareRewardPool.withdraw(uint256,uint256) (contracts/GshareRewardPool.sol#785-802):

External calls:

- [- safeTShareTransfer(_sender,_pending) (contracts/GshareRewardPool.sol#793)
- [- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/GshareRewardPool.sol#561)
- [- tshare.safeTransfer(_to,_tshareBal) (contracts/GshareRewardPool.sol#820)
- [- (success,returndata) = target.call{value: value}(data) (contracts/GshareRewardPool.sol#123)
- [- tshare.safeTransfer(_to,_amount) (contracts/GshareRewardPool.sol#822)

External calls sending eth:

- [- safeTShareTransfer(_sender,_pending) (contracts/GshareRewardPool.sol#793)
- [- (success,returndata) = target.call{value: value}(data) (contracts/GshareRewardPool.sol#123)

☒ State variables written after the call(s):

☒ - user.amount = user.amount.sub(_amount) (contracts/GshareRewardPool.sol#797)

Reentrancy in GshareRewardPool.withdraw(uint256,uint256) (contracts/GshareRewardPool.sol#785-802):

☒ External calls:

☒ - safeTShareTransfer(_sender,_pending) (contracts/GshareRewardPool.sol#793)

☒☒ - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/GshareRewardPool.sol#561)

☒☒ - tshare.safeTransfer(_to,_tshareBal) (contracts/GshareRewardPool.sol#820)

☒☒ - (success, returndata) = target.call{value: value}(data) (contracts/GshareRewardPool.sol#123)

☒☒ - tshare.safeTransfer(_to,_amount) (contracts/GshareRewardPool.sol#822)

☒ - pool.token.safeTransfer(_sender,_amount) (contracts/GshareRewardPool.sol#798)

☒ External calls sending eth:

☒ - safeTShareTransfer(_sender,_pending) (contracts/GshareRewardPool.sol#793)

☒☒ - (success, returndata) = target.call{value: value}(data) (contracts/GshareRewardPool.sol#123)

☒ State variables written after the call(s):

☒ - user.rewardDebt = user.amount.mul(pool.accTSharePerShare).div(1e18) (contracts/GshareRewardPool.sol#800)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities>

Reentrancy in Masonry.stake(uint256) (contracts/Masonry.sol#840-845):

☒ External calls:

☒ - super.stake(amount) (contracts/Masonry.sol#842)

☒☒ - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/Masonry.sol#631)

☒☒ - share.safeTransferFrom(msg.sender,address(this),amount) (contracts/Masonry.sol#670)

☒☒ - (success, returndata) = target.call{value: value}(data) (contracts/Masonry.sol#193)

☒ External calls sending eth:

☒ - super.stake(amount) (contracts/Masonry.sol#842)

☒☒ - (success, returndata) = target.call{value: value}(data) (contracts/Masonry.sol#193)

☒ State variables written after the call(s):

☒ - masons[msg.sender].epochTimerStart = treasury.epoch() (contracts/Masonry.sol#843)

Reentrancy in Masonry.withdraw(uint256) (contracts/Masonry.sol#847-853):

☒ External calls:

☒ - claimReward() (contracts/Masonry.sol#850)

☒☒ - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/Masonry.sol#631)

☒☒ - (success, returndata) = target.call{value: value}(data) (contracts/Masonry.sol#193)

```

❏- tomb.safeTransfer(msg.sender,reward) (contracts/Masonry.sol#865)
❏- super.withdraw(amount) (contracts/Masonry.sol#851)
❏- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
  (contracts/Masonry.sol#631)
❏- share.safeTransfer(msg.sender,amount) (contracts/Masonry.sol#678)
❏- (success, returndata) = target.call{value: value}(data) (contracts/Masonry.sol#193)
❏External calls sending eth:
❏- claimReward() (contracts/Masonry.sol#850)
❏- (success, returndata) = target.call{value: value}(data) (contracts/Masonry.sol#193)
❏- super.withdraw(amount) (contracts/Masonry.sol#851)
❏- (success, returndata) = target.call{value: value}(data) (contracts/Masonry.sol#193)
❏State variables written after the call(s):
❏- super.withdraw(amount) (contracts/Masonry.sol#851)
❏- _balances[msg.sender] = masonShare.sub(amount) (contracts/Masonry.sol#677)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

```

Reentrancy in Treasury.allocateSeigniorage() (contracts/Treasury.sol#1435-1475):

```

❏External calls:
❏- _updateTombPrice() (contracts/Treasury.sol#1436)
❏- IOracle(tombOracle).update() (contracts/Treasury.sol#1332)
❏- _sendToMasonry(_savedForMasonry) (contracts/Treasury.sol#1466)
❏- IBasisAsset(tomb).mint(address(this),_amount) (contracts/Treasury.sol#1401)
❏- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
  (contracts/Treasury.sol#896)
❏- IERC20(tomb).transfer(daoFund,_daoFundSharedAmount) (contracts/Treasury.sol#1406)
❏- (success, returndata) = target.call{value: value}(data) (contracts/Treasury.sol#458)
❏- IERC20(tomb).transfer(devFund,_devFundSharedAmount) (contracts/Treasury.sol#1413)
❏- IERC20(tomb).safeApprove(masonry,0) (contracts/Treasury.sol#1419)
❏- IERC20(tomb).safeApprove(masonry,_amount) (contracts/Treasury.sol#1420)
❏- IMasonry(masonry).allocateSeigniorage(_amount) (contracts/Treasury.sol#1421)
❏External calls sending eth:
❏- _sendToMasonry(_savedForMasonry) (contracts/Treasury.sol#1466)
❏- (success, returndata) = target.call{value: value}(data) (contracts/Treasury.sol#458)
❏State variables written after the call(s):
❏- seigniorageSaved = seigniorageSaved.add(_savedForBond) (contracts/Treasury.sol#1469)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

```

```

GShare.governanceRecoverUnsupported(IERC20,uint256,address) (contracts/
GhostShares.sol#905-911) ignores return value by _token.transfer(_to,_amount)

```

(contracts/GhostShares.sol#910)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer>

Ghost.governanceRecoverUnsupported(IERC20,uint256,address) (contracts/GhostToken.sol#1331-1337) ignores return value by `_token.transfer(_to,_amount)` (contracts/GhostToken.sol#1336)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer>

Treasury._sendToMasonry(uint256) (contracts/Treasury.sol#1400-1423) ignores return value by `IERC20(tomb).transfer(daoFund,_daoFundSharedAmount)` (contracts/Treasury.sol#1406)

Treasury._sendToMasonry(uint256) (contracts/Treasury.sol#1400-1423) ignores return value by `IERC20(tomb).transfer(devFund,_devFundSharedAmount)` (contracts/Treasury.sol#1413)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer>

GshareRewardPool.pendingShare(uint256,address) (contracts/GshareRewardPool.sol#719-730) performs a multiplication on the result of a division:

❑ `_tshareReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint)` (contracts/GshareRewardPool.sol#726)

❑ `_accTSharePerShare = accTSharePerShare.add(_tshareReward.mul(1e18).div(tokenSupply))` (contracts/GshareRewardPool.sol#727)

GshareRewardPool.updatePool(uint256) (contracts/GshareRewardPool.sol#741-761) performs a multiplication on the result of a division:

❑ `_tshareReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint)` (contracts/GshareRewardPool.sol#757)

❑ `_pool.accTSharePerShare = pool.accTSharePerShare.add(_tshareReward.mul(1e18).div(tokenSupply))` (contracts/GshareRewardPool.sol#758)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply>

Treasury.allocateSeigniorage() (contracts/Treasury.sol#1435-1475) performs a multiplication on the result of a division:

❑ `_seigniorage = tombSupply.mul(_percentage).div(1e18)` (contracts/Treasury.sol#1458)

❑ `_savedForMasonry = _seigniorage.mul(seigniorageExpansionFloorPercent).div(10000)` (contracts/Treasury.sol#1459)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply>

GshareRewardPool.updatePool(uint256) (contracts/GshareRewardPool.sol#741-761) uses a dangerous strict equality:

☒- tokenSupply == 0 (contracts/GshareRewardPool.sol#747)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities>

Reentrancy in Treasury.buyBonds(uint256,uint256) (contracts/Treasury.sol#1345-1372):

☒External calls:

☒- IBasisAsset(tomb).burnFrom(msg.sender,_tombAmount) (contracts/Treasury.sol#1365)

☒- IBasisAsset(tbond).mint(msg.sender,_bondAmount) (contracts/Treasury.sol#1366)

☒State variables written after the call(s):

☒- epochSupplyContractionLeft = epochSupplyContractionLeft.sub(_tombAmount) (contracts/Treasury.sol#1368)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1>

Ghost.setTaxTiersTwap(uint8,uint256) (contracts/GhostToken.sol#1105-1116) contains a tautology or contradiction:

☒- require(bool,string)(_index >= 0,Index has to be higher than 0) (contracts/GhostToken.sol#1106)

Ghost.setTaxTiersRate(uint8,uint256) (contracts/GhostToken.sol#1118-1123) contains a tautology or contradiction:

☒- require(bool,string)(_index >= 0,Index has to be higher than 0) (contracts/GhostToken.sol#1119)

Ghost._updateTaxRate(uint256) (contracts/GhostToken.sol#1137-1147) contains a tautology or contradiction:

☒- tierId >= 0 (contracts/GhostToken.sol#1139)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contradiction>

Treasury.setSupplyTiersEntry(uint8,uint256) (contracts/Treasury.sol#1241-1252) contains a tautology or contradiction:

☒- require(bool,string)(_index >= 0,Index has to be higher than 0) (contracts/Treasury.sol#1242)

Treasury.setMaxExpansionTiersEntry(uint8,uint256) (contracts/Treasury.sol#1254-1260) contains a tautology or contradiction:

☒- require(bool,string)(_index >= 0,Index has to be higher than 0) (contracts/Treasury.sol#1255)

Treasury._calculateMaxSupplyExpansionPercent(uint256) (contracts/Treasury.sol#1425-1433) contains a tautology or contradiction:

❌- tierId >= 0 (contracts/Treasury.sol#1426)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contradiction>

Ghost._getTombPrice()._price (contracts/GhostToken.sol#1130) 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#303) is a local variable never initialized

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables>

Treasury.getTombUpdatedPrice().price (contracts/Treasury.sol#1101) is a local variable never initialized

Treasury.getTombPrice().price (contracts/Treasury.sol#1093) is a local variable never initialized

Treasury.allocateSeigniorage()._savedForBond (contracts/Treasury.sol#1447) is a local variable never initialized

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables>

Ghost._getTombPrice() (contracts/GhostToken.sol#1129-1135) ignores return value by IOracle(tombOracle).consult(address(this),1e18) (contracts/GhostToken.sol#1130-1134)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return>

Treasury.getTombPrice() (contracts/Treasury.sol#1092-1098) ignores return value by IOracle(tombOracle).consult(tomb,1e18) (contracts/Treasury.sol#1093-1097)

Treasury.getTombUpdatedPrice() (contracts/Treasury.sol#1100-1106) ignores return value by IOracle(tombOracle).twap(tomb,1e18) (contracts/Treasury.sol#1101-1105)

Treasury.buyBonds(uint256,uint256) (contracts/Treasury.sol#1345-1372) ignores return value by IBasisAsset(tbond).mint(msg.sender,_bondAmount) (contracts/Treasury.sol#1366)

Treasury._sendToMasonry(uint256) (contracts/Treasury.sol#1400-1423) ignores return value by IBasisAsset(tomb).mint(address(this),_amount) (contracts/Treasury.sol#1401)

Treasury.allocateSeigniorage() (contracts/Treasury.sol#1435-1475) ignores return value by IBasisAsset(tomb).mint(address(this),_savedForBond) (contracts/Treasury.sol#1470)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return>

GshareRewardPool.setOperator(address) (contracts/GshareRewardPool.sol#827-829) should emit an event for:

☒- operator = _operator (contracts/GshareRewardPool.sol#828)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control>

Masonry.setOperator(address) (contracts/Masonry.sol#775-777) should emit an event for:

☒- operator = _operator (contracts/Masonry.sol#776)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control>

Treasury.setOperator(address) (contracts/Treasury.sol#1219-1221) should emit an event for:

☒- operator = _operator (contracts/Treasury.sol#1220)

Treasury.setMasonry(address) (contracts/Treasury.sol#1223-1225) should emit an event for:

☒- masonry = _masonry (contracts/Treasury.sol#1224)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control>

Ghost.setTaxRate(uint256) (contracts/GhostToken.sol#1173-1177) should emit an event for:

☒- taxRate = _taxRate (contracts/GhostToken.sol#1176)

Ghost.setBuyTax(uint256) (contracts/GhostToken.sol#1179-1183) should emit an event for:

☒- buyTax = _taxRate (contracts/GhostToken.sol#1182)

Ghost.setSellTax(uint256) (contracts/GhostToken.sol#1185-1189) should emit an event for:

☒- sellTax = _taxRate (contracts/GhostToken.sol#1188)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic>

GshareRewardPool.add(uint256,IERC20,bool,uint256) (contracts/GshareRewardPool.sol#652-690) should emit an event for:

☒- totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/GshareRewardPool.sol#688)

GshareRewardPool.set(uint256,uint256) (contracts/GshareRewardPool.sol#693-702) should emit an event for:

☒- totalAllocPoint = totalAllocPoint.sub(pool.allocPoint).add(_allocPoint) (contracts/GshareRewardPool.sol#697-699)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic>

Masonry.setLockUp(uint256,uint256) (contracts/Masonry.sol#779-783) should emit an event for:

☒- withdrawLockupEpochs = _withdrawLockupEpochs (contracts/Masonry.sol#781)

☒- rewardLockupEpochs = _rewardLockupEpochs (contracts/Masonry.sol#782)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic>

Treasury.setTombPriceCeiling(uint256) (contracts/Treasury.sol#1231-1234) should emit an event for:

☒- tombPriceCeiling = _tombPriceCeiling (contracts/Treasury.sol#1233)

Treasury.setMaxSupplyExpansionPercents(uint256) (contracts/Treasury.sol#1236-1239) should emit an event for:

☒- maxSupplyExpansionPercent = _maxSupplyExpansionPercent (contracts/Treasury.sol#1238)

Treasury.setBondDepletionFloorPercent(uint256) (contracts/Treasury.sol#1262-1265) should emit an event for:

☒- bondDepletionFloorPercent = _bondDepletionFloorPercent (contracts/Treasury.sol#1264)

Treasury.setMaxDebtRatioPercent(uint256) (contracts/Treasury.sol#1272-1275) should emit an event for:

☒- maxDebtRatioPercent = _maxDebtRatioPercent (contracts/Treasury.sol#1274)

Treasury.setBootstrap(uint256,uint256) (contracts/Treasury.sol#1277-1282) should emit an event for:

☒- bootstrapEpochs = _bootstrapEpochs (contracts/Treasury.sol#1280)

☒- bootstrapSupplyExpansionPercent = _bootstrapSupplyExpansionPercent (contracts/Treasury.sol#1281)

Treasury.setExtraFunds(address,uint256,address,uint256) (contracts/Treasury.sol#1284-1298) should emit an event for:

☒- daoFundSharedPercent = _daoFundSharedPercent (contracts/Treasury.sol#1295)

☒- devFundSharedPercent = _devFundSharedPercent (contracts/Treasury.sol#1297)

Treasury.setMaxDiscountRate(uint256) (contracts/Treasury.sol#1300-1302) should emit an event for:

☒- maxDiscountRate = _maxDiscountRate (contracts/Treasury.sol#1301)

Treasury.setMaxPremiumRate(uint256) (contracts/Treasury.sol#1304-1306) should emit an event for:

☒- maxPremiumRate = _maxPremiumRate (contracts/Treasury.sol#1305)

Treasury.setDiscountPercent(uint256) (contracts/Treasury.sol#1308-1311) should emit an event for:

☒- discountPercent = _discountPercent (contracts/Treasury.sol#1310)

Treasury.setPremiumThreshold(uint256) (contracts/Treasury.sol#1313-1317) should emit an event for:

☒- premiumThreshold = _premiumThreshold (contracts/Treasury.sol#1316)
 Treasury.setPremiumPercent(uint256) (contracts/Treasury.sol#1319-1322) should emit an event for:
 ☒- premiumPercent = _premiumPercent (contracts/Treasury.sol#1321)
 Treasury.setMintingFactorForPayingDebt(uint256) (contracts/Treasury.sol#1324-1327) should emit an event for:
 ☒- mintingFactorForPayingDebt = _mintingFactorForPayingDebt (contracts/Treasury.sol#1326)
 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic>

GShare.setTreasuryFund(address)._communityFund (contracts/GhostShares.sol#850) lacks a zero-check on :
 ☒☒- communityFund = _communityFund (contracts/GhostShares.sol#852)
 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation>

GshareRewardPool.setOperator(address)._operator (contracts/GshareRewardPool.sol#827) lacks a zero-check on :
 ☒☒- operator = _operator (contracts/GshareRewardPool.sol#828)
 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation>

Masonry.setOperator(address)._operator (contracts/Masonry.sol#775) lacks a zero-check on :
 ☒☒- operator = _operator (contracts/Masonry.sol#776)
 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation>

Treasury.initialize(address,address,address,address,address,uint256)._tomb (contracts/Treasury.sol#1176) lacks a zero-check on :
 ☒☒- tomb = _tomb (contracts/Treasury.sol#1183)
 Treasury.initialize(address,address,address,address,address,uint256)._tbond (contracts/Treasury.sol#1177) lacks a zero-check on :
 ☒☒- tbond = _tbond (contracts/Treasury.sol#1184)
 Treasury.initialize(address,address,address,address,address,uint256)._tshare (contracts/Treasury.sol#1178) lacks a zero-check on :
 ☒☒- tshare = _tshare (contracts/Treasury.sol#1185)
 Treasury.initialize(address,address,address,address,address,uint256)._tombOracle (contracts/Treasury.sol#1179) lacks a zero-check on :
 ☒☒- tombOracle = _tombOracle (contracts/Treasury.sol#1186)

Treasury.initialize(address,address,address,address,address,uint256)._masonry (contracts/Treasury.sol#1180) lacks a zero-check on :
 ☒- masonry = _masonry (contracts/Treasury.sol#1187)
 Treasury.setOperator(address)._operator (contracts/Treasury.sol#1219) lacks a zero-check on :
 ☒- operator = _operator (contracts/Treasury.sol#1220)
 Treasury.setMasonry(address)._masonry (contracts/Treasury.sol#1223) lacks a zero-check on :
 ☒- masonry = _masonry (contracts/Treasury.sol#1224)
 Treasury.setTombOracle(address)._tombOracle (contracts/Treasury.sol#1227) lacks a zero-check on :
 ☒- tombOracle = _tombOracle (contracts/Treasury.sol#1228)
 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation>

GshareRewardPool.updatePool(uint256) (contracts/GshareRewardPool.sol#741-761) has external calls inside a loop: tokenSupply = pool.token.balanceOf(address(this)) (contracts/GshareRewardPool.sol#746)
 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop>

Treasury.getTombCirculatingSupply() (contracts/Treasury.sol#1335-1343) has external calls inside a loop: balanceExcluded = balanceExcluded.add(tombErc20.balanceOf(excludedFromTotalSupply[entryId])) (contracts/Treasury.sol#1340)
 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop>

Variable 'Ghost._getTombPrice()._price (contracts/GhostToken.sol#1130)' in Ghost._getTombPrice() (contracts/GhostToken.sol#1129-1135) potentially used before declaration: uint256(_price) (contracts/GhostToken.sol#1131)
 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables>

Variable 'Treasury.getTombPrice().price (contracts/Treasury.sol#1093)' in Treasury.getTombPrice() (contracts/Treasury.sol#1092-1098) potentially used before declaration: uint256(price) (contracts/Treasury.sol#1094)
 Variable 'Treasury.getTombUpdatedPrice().price (contracts/Treasury.sol#1101)' in Treasury.getTombUpdatedPrice() (contracts/Treasury.sol#1100-1106) potentially used before declaration: uint256(price) (contracts/Treasury.sol#1102)
 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables>

declaration-usage-of-local-variables

Reentrancy in Treasury.allocateSeigniorage() (contracts/Treasury.sol#1435-1475):

External calls:

- _updateTombPrice() (contracts/Treasury.sol#1436)

- IOracle(tombOracle).update() (contracts/Treasury.sol#1332)

State variables written after the call(s):

- _mse = _calculateMaxSupplyExpansionPercent(tombSupply).mul(1e14) (contracts/Treasury.sol#1449)

- maxSupplyExpansionPercent = maxExpansionTiers[tierId] (contracts/Treasury.sol#1428)

- previousEpochTombPrice = getTombPrice() (contracts/Treasury.sol#1437)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2>

Reentrancy in GshareRewardPool.deposit(uint256,uint256) (contracts/GshareRewardPool.sol#764-782):

External calls:

- safeTShareTransfer(_sender,_pending) (contracts/GshareRewardPool.sol#772)

- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/GshareRewardPool.sol#561)

- tshare.safeTransfer(_to,_tshareBal) (contracts/GshareRewardPool.sol#820)

- (success,returndata) = target.call{value: value}(data) (contracts/GshareRewardPool.sol#123)

- tshare.safeTransfer(_to,_amount) (contracts/GshareRewardPool.sol#822)

External calls sending eth:

- safeTShareTransfer(_sender,_pending) (contracts/GshareRewardPool.sol#772)

- (success,returndata) = target.call{value: value}(data) (contracts/GshareRewardPool.sol#123)

Event emitted after the call(s):

- RewardPaid(_sender,_pending) (contracts/GshareRewardPool.sol#773)

Reentrancy in GshareRewardPool.deposit(uint256,uint256) (contracts/GshareRewardPool.sol#764-782):

External calls:

- safeTShareTransfer(_sender,_pending) (contracts/GshareRewardPool.sol#772)

- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/GshareRewardPool.sol#561)

- tshare.safeTransfer(_to,_tshareBal) (contracts/GshareRewardPool.sol#820)

- (success,returndata) = target.call{value: value}(data) (contracts/GshareRewardPool.sol#123)

- tshare.safeTransfer(_to,_amount) (contracts/GshareRewardPool.sol#822)

- pool.token.safeTransferFrom(_sender,address(this),_amount) (contracts/GshareRewardPool.sol#777)

External calls sending eth:

- safeTShareTransfer(_sender,_pending) (contracts/GshareRewardPool.sol#772)

- (success, returndata) = target.call{value: value}(data) (contracts/GshareRewardPool.sol#123)

Event emitted after the call(s):

- Deposit(_sender,_pid,_amount) (contracts/GshareRewardPool.sol#781)

Reentrancy in GshareRewardPool.emergencyWithdraw(uint256) (contracts/GshareRewardPool.sol#805-813):

External calls:

- pool.token.safeTransfer(msg.sender,_amount) (contracts/GshareRewardPool.sol#811)

Event emitted after the call(s):

- EmergencyWithdraw(msg.sender,_pid,_amount) (contracts/GshareRewardPool.sol#812)

Reentrancy in GshareRewardPool.withdraw(uint256,uint256) (contracts/GshareRewardPool.sol#785-802):

External calls:

- safeTShareTransfer(_sender,_pending) (contracts/GshareRewardPool.sol#793)

- returndata = address(token).functionCall(data, SafeERC20: low-level call failed) (contracts/GshareRewardPool.sol#561)

- tshare.safeTransfer(_to,_tshareBal) (contracts/GshareRewardPool.sol#820)

- (success, returndata) = target.call{value: value}(data) (contracts/GshareRewardPool.sol#123)

- tshare.safeTransfer(_to,_amount) (contracts/GshareRewardPool.sol#822)

External calls sending eth:

- safeTShareTransfer(_sender,_pending) (contracts/GshareRewardPool.sol#793)

- (success, returndata) = target.call{value: value}(data) (contracts/GshareRewardPool.sol#123)

Event emitted after the call(s):

- RewardPaid(_sender,_pending) (contracts/GshareRewardPool.sol#794)

Reentrancy in GshareRewardPool.withdraw(uint256,uint256) (contracts/GshareRewardPool.sol#785-802):

External calls:

- safeTShareTransfer(_sender,_pending) (contracts/GshareRewardPool.sol#793)

- returndata = address(token).functionCall(data, SafeERC20: low-level call failed) (contracts/GshareRewardPool.sol#561)

- tshare.safeTransfer(_to,_tshareBal) (contracts/GshareRewardPool.sol#820)

- (success, returndata) = target.call{value: value}(data) (contracts/GshareRewardPool.sol#123)

- tshare.safeTransfer(_to,_amount) (contracts/GshareRewardPool.sol#822)

- pool.token.safeTransfer(_sender,_amount) (contracts/GshareRewardPool.sol#798)

External calls sending eth:

- safeTShareTransfer(_sender,_pending) (contracts/GshareRewardPool.sol#793)

☒- (success, returndata) = target.call{value: value}(data) (contracts/GshareRewardPool.sol#123)

☒Event emitted after the call(s):

☒- Withdraw(_sender,_pid,_amount) (contracts/GshareRewardPool.sol#801)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3>

Reentrancy in Masonry.allocateSeigniorage(uint256) (contracts/Masonry.sol#870-887):

☒External calls:

☒- tomb.safeTransferFrom(msg.sender,address(this),amount) (contracts/Masonry.sol#885)

☒Event emitted after the call(s):

☒- RewardAdded(msg.sender,amount) (contracts/Masonry.sol#886)

Reentrancy in Masonry.claimReward() (contracts/Masonry.sol#859-868):

☒External calls:

☒- tomb.safeTransfer(msg.sender,reward) (contracts/Masonry.sol#865)

☒Event emitted after the call(s):

☒- RewardPaid(msg.sender,reward) (contracts/Masonry.sol#866)

Reentrancy in Masonry.stake(uint256) (contracts/Masonry.sol#840-845):

☒External calls:

☒- super.stake(amount) (contracts/Masonry.sol#842)

☒☒- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/Masonry.sol#631)

☒☒- share.safeTransferFrom(msg.sender,address(this),amount) (contracts/Masonry.sol#670)

☒☒- (success, returndata) = target.call{value: value}(data) (contracts/Masonry.sol#193)

☒External calls sending eth:

☒- super.stake(amount) (contracts/Masonry.sol#842)

☒☒- (success, returndata) = target.call{value: value}(data) (contracts/Masonry.sol#193)

☒Event emitted after the call(s):

☒- Staked(msg.sender,amount) (contracts/Masonry.sol#844)

Reentrancy in Masonry.withdraw(uint256) (contracts/Masonry.sol#847-853):

☒External calls:

☒- claimReward() (contracts/Masonry.sol#850)

☒☒- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/Masonry.sol#631)

☒☒- (success, returndata) = target.call{value: value}(data) (contracts/Masonry.sol#193)

☒☒- tomb.safeTransfer(msg.sender,reward) (contracts/Masonry.sol#865)

☒- super.withdraw(amount) (contracts/Masonry.sol#851)

☒☒- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/Masonry.sol#631)

☒☒- share.safeTransfer(msg.sender,amount) (contracts/Masonry.sol#678)

☒☒- (success, returndata) = target.call{value: value}(data) (contracts/Masonry.sol#193)

External calls sending eth:

```

- claimReward() (contracts/Masonry.sol#850)
- (success, returndata) = target.call{value: value}(data) (contracts/Masonry.sol#193)
- super.withdraw(amount) (contracts/Masonry.sol#851)
- (success, returndata) = target.call{value: value}(data) (contracts/Masonry.sol#193)
Event emitted after the call(s):
- Withdrawn(msg.sender, amount) (contracts/Masonry.sol#852)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

```

Reentrancy in Treasury._sendToMasonry(uint256) (contracts/Treasury.sol#1400-1423):

External calls:

```

- IBasisAsset(tomb).mint(address(this), _amount) (contracts/Treasury.sol#1401)
- IERC20(tomb).transfer(daoFund, _daoFundSharedAmount) (contracts/Treasury.sol#1406)

```

Event emitted after the call(s):

```

- DaoFundFunded(now, _daoFundSharedAmount) (contracts/Treasury.sol#1407)

```

Reentrancy in Treasury._sendToMasonry(uint256) (contracts/Treasury.sol#1400-1423):

External calls:

```

- IBasisAsset(tomb).mint(address(this), _amount) (contracts/Treasury.sol#1401)
- IERC20(tomb).transfer(daoFund, _daoFundSharedAmount) (contracts/Treasury.sol#1406)
- IERC20(tomb).transfer(devFund, _devFundSharedAmount) (contracts/Treasury.sol#1413)

```

Event emitted after the call(s):

```

- DevFundFunded(now, _devFundSharedAmount) (contracts/Treasury.sol#1414)

```

Reentrancy in Treasury._sendToMasonry(uint256) (contracts/Treasury.sol#1400-1423):

External calls:

```

- IBasisAsset(tomb).mint(address(this), _amount) (contracts/Treasury.sol#1401)
- IERC20(tomb).transfer(daoFund, _daoFundSharedAmount) (contracts/Treasury.sol#1406)
- IERC20(tomb).transfer(devFund, _devFundSharedAmount) (contracts/Treasury.sol#1413)
- IERC20(tomb).safeApprove(masonry, 0) (contracts/Treasury.sol#1419)
- IERC20(tomb).safeApprove(masonry, _amount) (contracts/Treasury.sol#1420)
- IMasonry(masonry).allocateSeigniorage(_amount) (contracts/Treasury.sol#1421)

```

Event emitted after the call(s):

```

- MasonryFunded(now, _amount) (contracts/Treasury.sol#1422)

```

Reentrancy in Treasury.allocateSeigniorage() (contracts/Treasury.sol#1435-1475):

External calls:

```

- _updateTombPrice() (contracts/Treasury.sol#1436)
- IOracle(tombOracle).update() (contracts/Treasury.sol#1332)
- _sendToMasonry(tombSupply.mul(bootstrapSupplyExpansionPercent).div(10000))
  (contracts/Treasury.sol#1441)
- IBasisAsset(tomb).mint(address(this), _amount) (contracts/Treasury.sol#1401)
- returndata = address(token).functionCall(data, SafeERC20: low-level call failed)

```

```

(contracts/Treasury.sol#896)
☒- IERC20(tomb).transfer(daoFund,_daoFundSharedAmount) (contracts/Treasury.sol#1406)
☒- (success, returndata) = target.call{value: value}(data) (contracts/Treasury.sol#458)
☒- IERC20(tomb).transfer(devFund,_devFundSharedAmount) (contracts/Treasury.sol#1413)
☒- IERC20(tomb).safeApprove(masonry,0) (contracts/Treasury.sol#1419)
☒- IERC20(tomb).safeApprove(masonry,_amount) (contracts/Treasury.sol#1420)
☒- IMasonry(masonry).allocateSeigniorage(_amount) (contracts/Treasury.sol#1421)
☒External calls sending eth:
☒- _sendToMasonry(tombSupply.mul(bootstrapSupplyExpansionPercent).div(10000))
  (contracts/Treasury.sol#1441)
☒- (success, returndata) = target.call{value: value}(data) (contracts/Treasury.sol#458)
☒Event emitted after the call(s):
☒- DaoFundFunded(now,_daoFundSharedAmount) (contracts/Treasury.sol#1407)
☒- _sendToMasonry(tombSupply.mul(bootstrapSupplyExpansionPercent).div(10000))
  (contracts/Treasury.sol#1441)
☒- DevFundFunded(now,_devFundSharedAmount) (contracts/Treasury.sol#1414)
☒- _sendToMasonry(tombSupply.mul(bootstrapSupplyExpansionPercent).div(10000))
  (contracts/Treasury.sol#1441)
☒- MasonryFunded(now,_amount) (contracts/Treasury.sol#1422)
☒- _sendToMasonry(tombSupply.mul(bootstrapSupplyExpansionPercent).div(10000))
  (contracts/Treasury.sol#1441)
Reentrancy in Treasury.allocateSeigniorage() (contracts/Treasury.sol#1435-1475):
☒External calls:
☒- _updateTombPrice() (contracts/Treasury.sol#1436)
☒- IOracle(tombOracle).update() (contracts/Treasury.sol#1332)
☒- _sendToMasonry(_savedForMasonry) (contracts/Treasury.sol#1466)
☒- IBasisAsset(tomb).mint(address(this),_amount) (contracts/Treasury.sol#1401)
☒- returndata = address(token).functionCall(data, SafeERC20: low-level call failed)
  (contracts/Treasury.sol#896)
☒- IERC20(tomb).transfer(daoFund,_daoFundSharedAmount) (contracts/Treasury.sol#1406)
☒- (success, returndata) = target.call{value: value}(data) (contracts/Treasury.sol#458)
☒- IERC20(tomb).transfer(devFund,_devFundSharedAmount) (contracts/Treasury.sol#1413)
☒- IERC20(tomb).safeApprove(masonry,0) (contracts/Treasury.sol#1419)
☒- IERC20(tomb).safeApprove(masonry,_amount) (contracts/Treasury.sol#1420)
☒- IMasonry(masonry).allocateSeigniorage(_amount) (contracts/Treasury.sol#1421)
☒External calls sending eth:
☒- _sendToMasonry(_savedForMasonry) (contracts/Treasury.sol#1466)
☒- (success, returndata) = target.call{value: value}(data) (contracts/Treasury.sol#458)
☒Event emitted after the call(s):
☒- DaoFundFunded(now,_daoFundSharedAmount) (contracts/Treasury.sol#1407)
☒- _sendToMasonry(_savedForMasonry) (contracts/Treasury.sol#1466)

```

```

☒- DevFundFunded(now,_devFundSharedAmount) (contracts/Treasury.sol#1414)
☒☒- _sendToMasonry(_savedForMasonry) (contracts/Treasury.sol#1466)
☒- MasonryFunded(now,_amount) (contracts/Treasury.sol#1422)
☒☒- _sendToMasonry(_savedForMasonry) (contracts/Treasury.sol#1466)
Reentrancy in Treasury.allocateSeigniorage() (contracts/Treasury.sol#1435-1475):
☒External calls:
☒- _updateTombPrice() (contracts/Treasury.sol#1436)
☒☒- IOracle(tombOracle).update() (contracts/Treasury.sol#1332)
☒- _sendToMasonry(_savedForMasonry) (contracts/Treasury.sol#1466)
☒☒- IBasisAsset(tomb).mint(address(this),_amount) (contracts/Treasury.sol#1401)
☒☒- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/Treasury.sol#896)
☒☒- IERC20(tomb).transfer(daoFund,_daoFundSharedAmount) (contracts/Treasury.sol#1406)
☒☒- (success,returndata) = target.call{value: value}(data) (contracts/Treasury.sol#458)
☒☒- IERC20(tomb).transfer(devFund,_devFundSharedAmount) (contracts/Treasury.sol#1413)
☒☒- IERC20(tomb).safeApprove(masonry,0) (contracts/Treasury.sol#1419)
☒☒- IERC20(tomb).safeApprove(masonry,_amount) (contracts/Treasury.sol#1420)
☒☒- IMasonry(masonry).allocateSeigniorage(_amount) (contracts/Treasury.sol#1421)
☒- IBasisAsset(tomb).mint(address(this),_savedForBond) (contracts/Treasury.sol#1470)
☒External calls sending eth:
☒- _sendToMasonry(_savedForMasonry) (contracts/Treasury.sol#1466)
☒☒- (success,returndata) = target.call{value: value}(data) (contracts/Treasury.sol#458)
☒Event emitted after the call(s):
☒- TreasuryFunded(now,_savedForBond) (contracts/Treasury.sol#1471)
Reentrancy in Treasury.buyBonds(uint256,uint256) (contracts/Treasury.sol#1345-1372):
☒External calls:
☒- IBasisAsset(tomb).burnFrom(msg.sender,_tombAmount) (contracts/Treasury.sol#1365)
☒- IBasisAsset(tbond).mint(msg.sender,_bondAmount) (contracts/Treasury.sol#1366)
☒- _updateTombPrice() (contracts/Treasury.sol#1369)
☒☒- IOracle(tombOracle).update() (contracts/Treasury.sol#1332)
☒Event emitted after the call(s):
☒- BoughtBonds(msg.sender,_tombAmount,_bondAmount) (contracts/Treasury.sol#1371)
Reentrancy in Treasury.redeemBonds(uint256,uint256) (contracts/Treasury.sol#1374-1398):
☒External calls:
☒- IBasisAsset(tbond).burnFrom(msg.sender,_bondAmount) (contracts/Treasury.sol#1392)
☒- IERC20(tomb).safeTransfer(msg.sender,_tombAmount) (contracts/Treasury.sol#1393)
☒- _updateTombPrice() (contracts/Treasury.sol#1395)
☒☒- IOracle(tombOracle).update() (contracts/Treasury.sol#1332)
☒Event emitted after the call(s):
☒- RedeemedBonds(msg.sender,_tombAmount,_bondAmount) (contracts/Treasury.sol#1397)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

```


GShare.unclaimedTreasuryFund() (contracts/GhostShares.sol#861-866) uses timestamp for comparisons

⚠Dangerous comparisons:

⊗- _now > endTime (contracts/GhostShares.sol#863)

⊗- communityFundLastClaimed >= _now (contracts/GhostShares.sol#864)

GShare.unclaimedDevFund() (contracts/GhostShares.sol#868-873) uses timestamp for comparisons

⚠Dangerous comparisons:

⊗- _now > endTime (contracts/GhostShares.sol#870)

⊗- devFundLastClaimed >= _now (contracts/GhostShares.sol#871)

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

GshareRewardPool.constructor(address,uint256) (contracts/GshareRewardPool.sol#628-637) uses timestamp for comparisons

⚠Dangerous comparisons:

⊗- require(bool,string)(block.timestamp < _poolStartTime,late) (contracts/GshareRewardPool.sol#632)

GshareRewardPool.checkPoolDuplicate(IERC20) (contracts/GshareRewardPool.sol#644-649) uses timestamp for comparisons

⚠Dangerous comparisons:

⊗- pid < length (contracts/GshareRewardPool.sol#646)

⊗- require(bool,string)(poolInfo[pid].token != _token,GshareRewardPool: existing pool?) (contracts/GshareRewardPool.sol#647)

GshareRewardPool.add(uint256,IERC20,bool,uint256) (contracts/GshareRewardPool.sol#652-690) uses timestamp for comparisons

⚠Dangerous comparisons:

⊗- block.timestamp < poolStartTime (contracts/GshareRewardPool.sol#662)

⊗- _lastRewardTime == 0 (contracts/GshareRewardPool.sol#664)

⊗- _lastRewardTime < poolStartTime (contracts/GshareRewardPool.sol#667)

⊗- _lastRewardTime == 0 || _lastRewardTime < block.timestamp (contracts/GshareRewardPool.sol#673)

⊗- _isStarted = (_lastRewardTime <= poolStartTime) || (_lastRewardTime <= block.timestamp) (contracts/GshareRewardPool.sol#677-679)

GshareRewardPool.getGeneratedReward(uint256,uint256) (contracts/GshareRewardPool.sol#705-716) uses timestamp for comparisons

⚠Dangerous comparisons:

⊗- _fromTime >= _toTime (contracts/GshareRewardPool.sol#706)

⊗- _toTime >= poolEndTime (contracts/GshareRewardPool.sol#707)

⊗- _toTime <= poolStartTime (contracts/GshareRewardPool.sol#712)

GshareRewardPool.pendingShare(uint256,address) (contracts/GshareRewardPool.sol#719-730) uses timestamp for comparisons

☒ Dangerous comparisons:

☒- block.timestamp > pool.lastRewardTime && tokenSupply != 0 (contracts/GshareRewardPool.sol#724)

GshareRewardPool.massUpdatePools() (contracts/GshareRewardPool.sol#733-738) uses timestamp for comparisons

☒ Dangerous comparisons:

☒- pid < length (contracts/GshareRewardPool.sol#735)

GshareRewardPool.updatePool(uint256) (contracts/GshareRewardPool.sol#741-761) uses timestamp for comparisons

☒ Dangerous comparisons:

☒- block.timestamp <= pool.lastRewardTime (contracts/GshareRewardPool.sol#743)

GshareRewardPool.governanceRecoverUnsupported(IERC20,uint256,address) (contracts/GshareRewardPool.sol#831-842) uses timestamp for comparisons

☒ Dangerous comparisons:

☒- block.timestamp < poolEndTime + 7776000 (contracts/GshareRewardPool.sol#832)

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

UniswapV2OracleLibrary.currentCumulativePrices(address) (contracts/Oracle.sol#353-377) uses timestamp for comparisons

☒ Dangerous comparisons:

☒- blockTimestampLast != blockTimestamp (contracts/Oracle.sol#368)

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

Address.isContract(address) (contracts/GshareRewardPool.sol#30-39) uses assembly

☒- INLINE ASM (contracts/GshareRewardPool.sol#37)

Address._verifyCallResult(bool,bytes,string) (contracts/GshareRewardPool.sol#175-192) uses assembly

☒- INLINE ASM (contracts/GshareRewardPool.sol#184-187)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage>

Address.isContract(address) (contracts/Masonry.sol#100-109) uses assembly

☒- INLINE ASM (contracts/Masonry.sol#107)

Address._verifyCallResult(bool,bytes,string) (contracts/Masonry.sol#245-262) uses assembly

☒- INLINE ASM (contracts/Masonry.sol#254-257)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage>

Address.isContract(address) (contracts/Treasury.sol#365-374) uses assembly

☒- INLINE ASM (contracts/Treasury.sol#372)

Address._verifyCallResult(bool,bytes,string) (contracts/Treasury.sol#510-527) uses assembly

☒- INLINE ASM (contracts/Treasury.sol#519-522)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage>

Different versions of Solidity is used:

☒- Version used: ['0.6.12', '>=0.6.0<0.8.0']

☒- >=0.6.0<0.8.0 (contracts/GBond.sol#11)

☒- >=0.6.0<0.8.0 (contracts/GBond.sol#228)

☒- >=0.6.0<0.8.0 (contracts/GBond.sol#308)

☒- >=0.6.0<0.8.0 (contracts/GBond.sol#335)

☒- >=0.6.0<0.8.0 (contracts/GBond.sol#405)

☒- 0.6.12 (contracts/GBond.sol#412)

☒- >=0.6.0<0.8.0 (contracts/GBond.sol#454)

☒- >=0.6.0<0.8.0 (contracts/GBond.sol#762)

☒- 0.6.12 (contracts/GBond.sol#806)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used>

Different versions of Solidity is used:

☒- Version used: ['0.6.12', '>=0.6.0<0.8.0']

☒- >=0.6.0<0.8.0 (contracts/GhostShares.sol#11)

☒- >=0.6.0<0.8.0 (contracts/GhostShares.sol#91)

☒- >=0.6.0<0.8.0 (contracts/GhostShares.sol#118)

☒- >=0.6.0<0.8.0 (contracts/GhostShares.sol#188)

☒- 0.6.12 (contracts/GhostShares.sol#195)

☒- >=0.6.0<0.8.0 (contracts/GhostShares.sol#237)

☒- >=0.6.0<0.8.0 (contracts/GhostShares.sol#454)

☒- >=0.6.0<0.8.0 (contracts/GhostShares.sol#762)

☒- 0.6.12 (contracts/GhostShares.sol#806)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used>

Different versions of Solidity is used:

☒- Version used: ['0.6.12', '>=0.6.0<0.8.0']

☒- 0.6.12 (contracts/GhostToken.sol#11)

☒- 0.6.12 (contracts/GhostToken.sol#25)

☒- >=0.6.0<0.8.0 (contracts/GhostToken.sol#187)

☒- >=0.6.0<0.8.0 (contracts/GhostToken.sol#221)

```

❑- >=0.6.0<0.8.0 (contracts/GhostToken.sol#438)
❑- >=0.6.0<0.8.0 (contracts/GhostToken.sol#518)
❑- >=0.6.0<0.8.0 (contracts/GhostToken.sol#545)
❑- >=0.6.0<0.8.0 (contracts/GhostToken.sol#615)
❑- 0.6.12 (contracts/GhostToken.sol#622)
❑- >=0.6.0<0.8.0 (contracts/GhostToken.sol#664)
❑- >=0.6.0<0.8.0 (contracts/GhostToken.sol#972)
❑- 0.6.12 (contracts/GhostToken.sol#1016)

```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used>

Different versions of Solidity is used:

```

❑- Version used: ['0.6.12', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0']
❑- >=0.6.2<0.8.0 (contracts/GshareRewardPool.sol#7)
❑- >=0.6.0<0.8.0 (contracts/GshareRewardPool.sol#199)
❑- >=0.6.0<0.8.0 (contracts/GshareRewardPool.sol#416)
❑- >=0.6.0<0.8.0 (contracts/GshareRewardPool.sol#496)
❑- 0.6.12 (contracts/GshareRewardPool.sol#573)

```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used>

Different versions of Solidity is used:

```

❑- Version used: ['0.6.12', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0', '^0.6.0']
❑- 0.6.12 (contracts/Masonry.sol#15)
❑- ^0.6.0 (contracts/Masonry.sol#31)
❑- 0.6.12 (contracts/Masonry.sol#49)
❑- >=0.6.2<0.8.0 (contracts/Masonry.sol#77)
❑- >=0.6.0<0.8.0 (contracts/Masonry.sol#269)
❑- >=0.6.0<0.8.0 (contracts/Masonry.sol#486)
❑- >=0.6.0<0.8.0 (contracts/Masonry.sol#566)
❑- 0.6.12 (contracts/Masonry.sol#643)

```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used>

Different versions of Solidity is used:

```

❑- Version used: ['0.6.12', '>=0.6.0<0.8.0', '^0.6.0']
❑- >=0.6.0<0.8.0 (contracts/Oracle.sol#7)
❑- >=0.6.0<0.8.0 (contracts/Oracle.sol#34)
❑- >=0.6.0<0.8.0 (contracts/Oracle.sol#104)
❑- 0.6.12 (contracts/Oracle.sol#111)
❑- ^0.6.0 (contracts/Oracle.sol#151)

```

- ☒- ^0.6.0 (contracts/Oracle.sol#243)
- ☒- ^0.6.0 (contracts/Oracle.sol#263)
- ☒- ^0.6.0 (contracts/Oracle.sol#339)
- ☒- >=0.6.0<0.8.0 (contracts/Oracle.sol#384)
- ☒- ^0.6.0 (contracts/Oracle.sol#599)
- ☒- 0.6.12 (contracts/Oracle.sol#687)

Reference: <https://github.com/crytic/sliether/wiki/Detector-Documentation#different-pragma-directives-are-used>

Different versions of Solidity is used:

- ☒- Version used: ['0.6.12', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0', '^0.6.0']
- ☒- 0.6.12 (contracts/Treasury.sol#15)
- ☒- 0.6.12 (contracts/Treasury.sol#53)
- ☒- ^0.6.0 (contracts/Treasury.sol#65)
- ☒- 0.6.12 (contracts/Treasury.sol#83)
- ☒- >=0.6.0<0.8.0 (contracts/Treasury.sol#111)
- ☒- >=0.6.0<0.8.0 (contracts/Treasury.sol#138)
- ☒- >=0.6.0<0.8.0 (contracts/Treasury.sol#208)
- ☒- 0.6.12 (contracts/Treasury.sol#215)
- ☒- ^0.6.0 (contracts/Treasury.sol#255)
- ☒- >=0.6.0<0.8.0 (contracts/Treasury.sol#277)
- ☒- >=0.6.2<0.8.0 (contracts/Treasury.sol#342)
- ☒- >=0.6.0<0.8.0 (contracts/Treasury.sol#534)
- ☒- >=0.6.0<0.8.0 (contracts/Treasury.sol#751)
- ☒- >=0.6.0<0.8.0 (contracts/Treasury.sol#831)
- ☒- >=0.6.0<0.8.0 (contracts/Treasury.sol#908)
- ☒- 0.6.12 (contracts/Treasury.sol#942)

Reference: <https://github.com/crytic/sliether/wiki/Detector-Documentation#different-pragma-directives-are-used>

GshareRewardPool.updatePool(uint256) (contracts/GshareRewardPool.sol#741-761) has costly operations inside a loop:

- ☒- totalAllocPoint = totalAllocPoint.add(pool.allocPoint) (contracts/GshareRewardPool.sol#753)

Reference: <https://github.com/crytic/sliether/wiki/Detector-Documentation#costly-operations-inside-a-loop>

Context._msgData() (contracts/GBond.sol#325-328) is never used and should be removed
 ERC20._setupDecimals(uint8) (contracts/GBond.sol#737-739) is never used and should be removed

SafeMath.div(uint256,uint256) (contracts/GBond.sol#143-146) is never used and should be removed

SafeMath.div(uint256,uint256,string) (contracts/GBond.sol#198-201) is never used and should be removed

SafeMath.mod(uint256,uint256) (contracts/GBond.sol#160-163) is never used and should be removed

SafeMath.mod(uint256,uint256,string) (contracts/GBond.sol#218-221) is never used and should be removed

SafeMath.mul(uint256,uint256) (contracts/GBond.sol#124-129) is never used and should be removed

SafeMath.tryAdd(uint256,uint256) (contracts/GBond.sol#32-36) is never used and should be removed

SafeMath.tryDiv(uint256,uint256) (contracts/GBond.sol#68-71) is never used and should be removed

SafeMath.tryMod(uint256,uint256) (contracts/GBond.sol#78-81) is never used and should be removed

SafeMath.tryMul(uint256,uint256) (contracts/GBond.sol#53-61) is never used and should be removed

SafeMath.trySub(uint256,uint256) (contracts/GBond.sol#43-46) is never used and should be removed

Reference: <https://github.com/crytic/sliether/wiki/Detector-Documentation#dead-code>

Ghost._getTombPrice() (contracts/GhostToken.sol#1129-1135) is never used and should be removed

Ghost._updateTaxRate(uint256) (contracts/GhostToken.sol#1137-1147) is never used and should be removed

Math.average(uint256,uint256) (contracts/GhostToken.sol#211-214) is never used and should be removed

Math.max(uint256,uint256) (contracts/GhostToken.sol#196-198) is never used and should be removed

Math.min(uint256,uint256) (contracts/GhostToken.sol#203-205) is never used and should be removed

SafeMath8.add(uint8,uint8) (contracts/GhostToken.sol#51-56) is never used and should be removed

SafeMath8.div(uint8,uint8) (contracts/GhostToken.sol#125-127) is never used and should be removed

SafeMath8.div(uint8,uint8,string) (contracts/GhostToken.sol#141-147) is never used and should be removed

SafeMath8.mod(uint8,uint8) (contracts/GhostToken.sol#161-163) is never used and should be removed

SafeMath8.mod(uint8,uint8,string) (contracts/GhostToken.sol#177-180) is never used and should be removed

SafeMath8.mul(uint8,uint8) (contracts/GhostToken.sol#99-111) is never used and should

be removed

SafeMath8.sub(uint8,uint8) (contracts/GhostToken.sol#68-70) is never used and should be removed

SafeMath8.sub(uint8,uint8,string) (contracts/GhostToken.sol#82-87) is never used and should be removed

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code>

Address.functionCall(address,bytes) (contracts/GshareRewardPool.sol#83-85) is never used and should be removed

Address.functionCallWithValue(address,bytes,uint256) (contracts/GshareRewardPool.sol#108-110) is never used and should be removed

Address.functionDelegateCall(address,bytes) (contracts/GshareRewardPool.sol#157-159) is never used and should be removed

Address.functionDelegateCall(address,bytes,string) (contracts/GshareRewardPool.sol#167-173) is never used and should be removed

Address.functionStaticCall(address,bytes) (contracts/GshareRewardPool.sol#133-135) is never used and should be removed

Address.functionStaticCall(address,bytes,string) (contracts/GshareRewardPool.sol#143-149) is never used and should be removed

Address.sendValue(address,uint256) (contracts/GshareRewardPool.sol#57-63) is never used and should be removed

SafeERC20.safeApprove(IERC20,address,uint256) (contracts/GshareRewardPool.sol#529-538) is never used and should be removed

SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (contracts/GshareRewardPool.sol#545-548) is never used and should be removed

SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (contracts/GshareRewardPool.sol#540-543) is never used and should be removed

SafeMath.sub(uint256,uint256,string) (contracts/GshareRewardPool.sol#366-369) is never used and should be removed

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code>

Babylonian.sqrt(uint256) (contracts/Oracle.sol#246-258) is never used and should be removed

FixedPoint.decode(FixedPoint.uq112x112) (contracts/Oracle.sol#316-318) is never used and should be removed

FixedPoint.div(FixedPoint.uq112x112,uint112) (contracts/Oracle.sol#295-298) is never used and should be removed

FixedPoint.encode(uint112) (contracts/Oracle.sol#285-287) is never used and should be removed

FixedPoint.encode144(uint144) (contracts/Oracle.sol#290-292) is never used and should be removed

`FixedPoint.reciprocal(FixedPoint.uq112x112)` (contracts/Oracle.sol#326-329) is never used and should be removed

`FixedPoint.sqrt(FixedPoint.uq112x112)` (contracts/Oracle.sol#332-334) is never used and should be removed

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code>

`SafeERC20.safeTransferFrom(IERC20,address,address,uint256)` (contracts/Treasury.sol#853-855) is never used and should be removed

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code>

`Pragma version>=0.6.0<0.8.0` (contracts/GBond.sol#11) is too complex

`Pragma version>=0.6.0<0.8.0` (contracts/GBond.sol#228) is too complex

`Pragma version>=0.6.0<0.8.0` (contracts/GBond.sol#308) is too complex

`Pragma version>=0.6.0<0.8.0` (contracts/GBond.sol#335) is too complex

`Pragma version>=0.6.0<0.8.0` (contracts/GBond.sol#405) is too complex

`Pragma version>=0.6.0<0.8.0` (contracts/GBond.sol#454) is too complex

`Pragma version>=0.6.0<0.8.0` (contracts/GBond.sol#762) is too complex

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity>

`Pragma version>=0.6.0<0.8.0` (contracts/GhostShares.sol#11) is too complex

`Pragma version>=0.6.0<0.8.0` (contracts/GhostShares.sol#91) is too complex

`Pragma version>=0.6.0<0.8.0` (contracts/GhostShares.sol#118) is too complex

`Pragma version>=0.6.0<0.8.0` (contracts/GhostShares.sol#188) is too complex

`Pragma version>=0.6.0<0.8.0` (contracts/GhostShares.sol#237) is too complex

`Pragma version>=0.6.0<0.8.0` (contracts/GhostShares.sol#454) is too complex

`Pragma version>=0.6.0<0.8.0` (contracts/GhostShares.sol#762) is too complex

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity>

`Pragma version>=0.6.0<0.8.0` (contracts/GhostToken.sol#187) is too complex

`Pragma version>=0.6.0<0.8.0` (contracts/GhostToken.sol#221) is too complex

`Pragma version>=0.6.0<0.8.0` (contracts/GhostToken.sol#438) is too complex

`Pragma version>=0.6.0<0.8.0` (contracts/GhostToken.sol#518) is too complex

`Pragma version>=0.6.0<0.8.0` (contracts/GhostToken.sol#545) is too complex

`Pragma version>=0.6.0<0.8.0` (contracts/GhostToken.sol#615) is too complex

`Pragma version>=0.6.0<0.8.0` (contracts/GhostToken.sol#664) is too complex

`Pragma version>=0.6.0<0.8.0` (contracts/GhostToken.sol#972) is too complex

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity>

Pragma version $\geq 0.6.2 < 0.8.0$ (contracts/GshareRewardPool.sol#7) is too complex
Pragma version $\geq 0.6.0 < 0.8.0$ (contracts/GshareRewardPool.sol#199) is too complex
Pragma version $\geq 0.6.0 < 0.8.0$ (contracts/GshareRewardPool.sol#416) is too complex
Pragma version $\geq 0.6.0 < 0.8.0$ (contracts/GshareRewardPool.sol#496) is too complex
Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity>

Pragma version $\wedge 0.6.0$ (contracts/Masonry.sol#31) allows old versions
Pragma version $\geq 0.6.2 < 0.8.0$ (contracts/Masonry.sol#77) is too complex
Pragma version $\geq 0.6.0 < 0.8.0$ (contracts/Masonry.sol#269) is too complex
Pragma version $\geq 0.6.0 < 0.8.0$ (contracts/Masonry.sol#486) is too complex
Pragma version $\geq 0.6.0 < 0.8.0$ (contracts/Masonry.sol#566) is too complex
Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity>

Pragma version $\geq 0.6.0 < 0.8.0$ (contracts/Oracle.sol#7) is too complex
Pragma version $\geq 0.6.0 < 0.8.0$ (contracts/Oracle.sol#34) is too complex
Pragma version $\geq 0.6.0 < 0.8.0$ (contracts/Oracle.sol#104) is too complex
Pragma version $\wedge 0.6.0$ (contracts/Oracle.sol#151) allows old versions
Pragma version $\wedge 0.6.0$ (contracts/Oracle.sol#243) allows old versions
Pragma version $\wedge 0.6.0$ (contracts/Oracle.sol#263) allows old versions
Pragma version $\wedge 0.6.0$ (contracts/Oracle.sol#339) allows old versions
Pragma version $\geq 0.6.0 < 0.8.0$ (contracts/Oracle.sol#384) is too complex
Pragma version $\wedge 0.6.0$ (contracts/Oracle.sol#599) allows old versions
Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity>

Pragma version $\wedge 0.6.0$ (contracts/Treasury.sol#65) allows old versions
Pragma version $\geq 0.6.0 < 0.8.0$ (contracts/Treasury.sol#111) is too complex
Pragma version $\geq 0.6.0 < 0.8.0$ (contracts/Treasury.sol#138) is too complex
Pragma version $\geq 0.6.0 < 0.8.0$ (contracts/Treasury.sol#208) is too complex
Pragma version $\wedge 0.6.0$ (contracts/Treasury.sol#255) allows old versions
Pragma version $\geq 0.6.0 < 0.8.0$ (contracts/Treasury.sol#277) is too complex
Pragma version $\geq 0.6.2 < 0.8.0$ (contracts/Treasury.sol#342) is too complex
Pragma version $\geq 0.6.0 < 0.8.0$ (contracts/Treasury.sol#534) is too complex
Pragma version $\geq 0.6.0 < 0.8.0$ (contracts/Treasury.sol#751) is too complex
Pragma version $\geq 0.6.0 < 0.8.0$ (contracts/Treasury.sol#831) is too complex
Pragma version $\geq 0.6.0 < 0.8.0$ (contracts/Treasury.sol#908) is too complex
Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity>

Low level call in Address.sendValue(address,uint256) (contracts/GshareRewardPool.sol#57-63):

☒- (success) = recipient.call{value: amount}() (contracts/GshareRewardPool.sol#61)

Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (contracts/GshareRewardPool.sol#118-125):

☒- (success, returndata) = target.call{value: value}(data) (contracts/GshareRewardPool.sol#123)

Low level call in Address.functionStaticCall(address,bytes,string) (contracts/GshareRewardPool.sol#143-149):

☒- (success, returndata) = target.staticcall(data) (contracts/GshareRewardPool.sol#147)

Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/GshareRewardPool.sol#167-173):

☒- (success, returndata) = target.delegatecall(data) (contracts/GshareRewardPool.sol#171)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls>

Low level call in Address.sendValue(address,uint256) (contracts/Masonry.sol#127-133):

☒- (success) = recipient.call{value: amount}() (contracts/Masonry.sol#131)

Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (contracts/Masonry.sol#188-195):

☒- (success, returndata) = target.call{value: value}(data) (contracts/Masonry.sol#193)

Low level call in Address.functionStaticCall(address,bytes,string) (contracts/Masonry.sol#213-219):

☒- (success, returndata) = target.staticcall(data) (contracts/Masonry.sol#217)

Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/Masonry.sol#237-243):

☒- (success, returndata) = target.delegatecall(data) (contracts/Masonry.sol#241)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls>

Low level call in Address.sendValue(address,uint256) (contracts/Treasury.sol#392-398):

☒- (success) = recipient.call{value: amount}() (contracts/Treasury.sol#396)

Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (contracts/Treasury.sol#453-460):

☒- (success, returndata) = target.call{value: value}(data) (contracts/Treasury.sol#458)

Low level call in Address.functionStaticCall(address,bytes,string) (contracts/Treasury.sol#478-484):

☒- (success, returndata) = target.staticcall(data) (contracts/Treasury.sol#482)

Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/Treasury.sol#502-508):

❌- (success, returndata) = target.delegatecall(data) (contracts/Treasury.sol#506)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls>

Parameter GShare.setTreasuryFund(address)._communityFund (contracts/GhostShares.sol#850) is not in mixedCase

Parameter GShare.setDevFund(address)._devFund (contracts/GhostShares.sol#855) is not in mixedCase

Parameter GShare.distributeReward(address)._farmingIncentiveFund (contracts/GhostShares.sol#894) is not in mixedCase

Parameter GShare.governanceRecoverUnsupported(IERC20,uint256,address)._token (contracts/GhostShares.sol#906) is not in mixedCase

Parameter GShare.governanceRecoverUnsupported(IERC20,uint256,address)._amount (contracts/GhostShares.sol#907) is not in mixedCase

Parameter GShare.governanceRecoverUnsupported(IERC20,uint256,address)._to (contracts/GhostShares.sol#908) is not in mixedCase

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions>

Parameter Ghost.isAddressExcluded(address)._address (contracts/GhostToken.sol#1101) is not in mixedCase

Parameter Ghost.setTaxTiersTwap(uint8,uint256)._index (contracts/GhostToken.sol#1105) is not in mixedCase

Parameter Ghost.setTaxTiersTwap(uint8,uint256)._value (contracts/GhostToken.sol#1105) is not in mixedCase

Parameter Ghost.setTaxTiersRate(uint8,uint256)._index (contracts/GhostToken.sol#1118) is not in mixedCase

Parameter Ghost.setTaxTiersRate(uint8,uint256)._value (contracts/GhostToken.sol#1118) is not in mixedCase

Parameter Ghost.setBurnThreshold(uint256)._burnThreshold (contracts/GhostToken.sol#1125) is not in mixedCase

Parameter Ghost.setTombOracle(address)._tombOracle (contracts/GhostToken.sol#1157) is not in mixedCase

Parameter Ghost.setTaxOffice(address)._taxOffice (contracts/GhostToken.sol#1162) is not in mixedCase

Parameter Ghost.setTaxCollectorAddress(address)._taxCollectorAddress (contracts/GhostToken.sol#1168) is not in mixedCase

Parameter Ghost.setTaxRate(uint256)._taxRate (contracts/GhostToken.sol#1173) is not in mixedCase

Parameter Ghost.setBuyTax(uint256)._taxRate (contracts/GhostToken.sol#1179) is not in mixedCase

Parameter Ghost.setSellTax(uint256)._taxRate (contracts/GhostToken.sol#1185) is not in mixedCase

Parameter Ghost.setUniswapv2Pair(address)._uniswapv2Pair (contracts/GhostToken.sol#1191) is not in mixedCase

Parameter Ghost.excludeAddress(address)._address (contracts/GhostToken.sol#1196) is not in mixedCase

Parameter Ghost.includeAddress(address)._address (contracts/GhostToken.sol#1202) is not in mixedCase

Parameter Ghost.distributeReward(address,address)._launcherWallet (contracts/GhostToken.sol#1320) is not in mixedCase

Parameter Ghost.distributeReward(address,address)._treasuryWallet (contracts/GhostToken.sol#1321) is not in mixedCase

Parameter Ghost.governanceRecoverUnsupported(IERC20,uint256,address)._token (contracts/GhostToken.sol#1332) is not in mixedCase

Parameter Ghost.governanceRecoverUnsupported(IERC20,uint256,address)._amount (contracts/GhostToken.sol#1333) is not in mixedCase

Parameter Ghost.governanceRecoverUnsupported(IERC20,uint256,address)._to (contracts/GhostToken.sol#1334) is not in mixedCase

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions>

Parameter GshareRewardPool.checkPoolDuplicate(IERC20)._token (contracts/GshareRewardPool.sol#644) is not in mixedCase

Parameter GshareRewardPool.add(uint256,IERC20,bool,uint256)._allocPoint (contracts/GshareRewardPool.sol#653) is not in mixedCase

Parameter GshareRewardPool.add(uint256,IERC20,bool,uint256)._token (contracts/GshareRewardPool.sol#654) is not in mixedCase

Parameter GshareRewardPool.add(uint256,IERC20,bool,uint256)._withUpdate (contracts/GshareRewardPool.sol#655) is not in mixedCase

Parameter GshareRewardPool.add(uint256,IERC20,bool,uint256)._lastRewardTime (contracts/GshareRewardPool.sol#656) is not in mixedCase

Parameter GshareRewardPool.set(uint256,uint256)._pid (contracts/GshareRewardPool.sol#693) is not in mixedCase

Parameter GshareRewardPool.set(uint256,uint256)._allocPoint (contracts/GshareRewardPool.sol#693) is not in mixedCase

Parameter GshareRewardPool.getGeneratedReward(uint256,uint256)._fromTime (contracts/GshareRewardPool.sol#705) is not in mixedCase

Parameter GshareRewardPool.getGeneratedReward(uint256,uint256)._toTime (contracts/GshareRewardPool.sol#705) is not in mixedCase

Parameter GshareRewardPool.pendingShare(uint256,address)._pid (contracts/GshareRewardPool.sol#719) is not in mixedCase

Parameter GshareRewardPool.pendingShare(uint256,address)._user (contracts/GshareRewardPool.sol#719) is not in mixedCase

Parameter GshareRewardPool.updatePool(uint256)._pid (contracts/GshareRewardPool.sol#741) is not in mixedCase

Parameter GshareRewardPool.deposit(uint256,uint256)._pid (contracts/GshareRewardPool.sol#764) is not in mixedCase

Parameter GshareRewardPool.deposit(uint256,uint256)._amount (contracts/GshareRewardPool.sol#764) is not in mixedCase

Parameter GshareRewardPool.withdraw(uint256,uint256)._pid (contracts/GshareRewardPool.sol#785) is not in mixedCase

Parameter GshareRewardPool.withdraw(uint256,uint256)._amount (contracts/GshareRewardPool.sol#785) is not in mixedCase

Parameter GshareRewardPool.emergencyWithdraw(uint256)._pid (contracts/GshareRewardPool.sol#805) is not in mixedCase

Parameter GshareRewardPool.safeTShareTransfer(address,uint256)._to (contracts/GshareRewardPool.sol#816) is not in mixedCase

Parameter GshareRewardPool.safeTShareTransfer(address,uint256)._amount (contracts/GshareRewardPool.sol#816) is not in mixedCase

Parameter GshareRewardPool.setOperator(address)._operator (contracts/GshareRewardPool.sol#827) is not in mixedCase

Parameter GshareRewardPool.governanceRecoverUnsupported(IERC20,uint256,address)._token (contracts/GshareRewardPool.sol#831) is not in mixedCase

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions>

Parameter Masonry.initialize(IERC20,IERC20,ITreasury)._tomb (contracts/Masonry.sol#756) is not in mixedCase

Parameter Masonry.initialize(IERC20,IERC20,ITreasury)._share (contracts/Masonry.sol#757) is not in mixedCase

Parameter Masonry.initialize(IERC20,IERC20,ITreasury)._treasury (contracts/Masonry.sol#758) is not in mixedCase

Parameter Masonry.setOperator(address)._operator (contracts/Masonry.sol#775) is not in mixedCase

Parameter Masonry.setLockUp(uint256,uint256)._withdrawLockupEpochs (contracts/Masonry.sol#779) is not in mixedCase

Parameter Masonry.setLockUp(uint256,uint256)._rewardLockupEpochs (contracts/Masonry.sol#779) is not in mixedCase

Parameter Masonry.governanceRecoverUnsupported(IERC20,uint256,address)._token (contracts/Masonry.sol#889) is not in mixedCase

Parameter Masonry.governanceRecoverUnsupported(IERC20,uint256,address)._amount (contracts/Masonry.sol#889) is not in mixedCase

Parameter `Masonry.governanceRecoverUnsupported(IERC20,uint256,address)._to` (`contracts/Masonry.sol#889`) is not in mixedCase

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions>

Function `IUniswapV2Pair.DOMAIN_SEPARATOR()` (`contracts/Oracle.sol#179`) is not in mixedCase

Function `IUniswapV2Pair.PERMIT_TYPEHASH()` (`contracts/Oracle.sol#181`) is not in mixedCase

Function `IUniswapV2Pair.MINIMUM_LIQUIDITY()` (`contracts/Oracle.sol#200`) is not in mixedCase

Struct `FixedPoint.uq112x112` (`contracts/Oracle.sol#270-272`) is not in CapWords

Struct `FixedPoint.uq144x112` (`contracts/Oracle.sol#276-278`) is not in CapWords

Parameter `Epoch.setPeriod(uint256)._period` (`contracts/Oracle.sol#673`) is not in mixedCase

Parameter `Epoch.setEpoch(uint256)._epoch` (`contracts/Oracle.sol#678`) is not in mixedCase

Parameter `Oracle.consult(address,uint256)._token` (`contracts/Oracle.sol#753`) is not in mixedCase

Parameter `Oracle.consult(address,uint256)._amountIn` (`contracts/Oracle.sol#753`) is not in mixedCase

Parameter `Oracle.twap(address,uint256)._token` (`contracts/Oracle.sol#762`) is not in mixedCase

Parameter `Oracle.twap(address,uint256)._amountIn` (`contracts/Oracle.sol#762`) is not in mixedCase

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions>

Parameter `Treasury.initialize(address,address,address,address,address,uint256)._tomb` (`contracts/Treasury.sol#1176`) is not in mixedCase

Parameter `Treasury.initialize(address,address,address,address,address,uint256)._tbond` (`contracts/Treasury.sol#1177`) is not in mixedCase

Parameter `Treasury.initialize(address,address,address,address,address,uint256)._tshare` (`contracts/Treasury.sol#1178`) is not in mixedCase

Parameter

`Treasury.initialize(address,address,address,address,address,uint256)._tombOracle` (`contracts/Treasury.sol#1179`) is not in mixedCase

Parameter `Treasury.initialize(address,address,address,address,address,uint256)._masonry` (`contracts/Treasury.sol#1180`) is not in mixedCase

Parameter

`Treasury.initialize(address,address,address,address,address,uint256)._startTime` (`contracts/Treasury.sol#1181`) is not in mixedCase

Parameter Treasury.setOperator(address)._operator (contracts/Treasury.sol#1219) is not in mixedCase

Parameter Treasury.setMasonry(address)._masonry (contracts/Treasury.sol#1223) is not in mixedCase

Parameter Treasury.setTombOracle(address)._tombOracle (contracts/Treasury.sol#1227) is not in mixedCase

Parameter Treasury.setTombPriceCeiling(uint256)._tombPriceCeiling (contracts/Treasury.sol#1231) is not in mixedCase

Parameter Treasury.setMaxSupplyExpansionPercents(uint256)._maxSupplyExpansionPercent (contracts/Treasury.sol#1236) is not in mixedCase

Parameter Treasury.setSupplyTiersEntry(uint8,uint256)._index (contracts/Treasury.sol#1241) is not in mixedCase

Parameter Treasury.setSupplyTiersEntry(uint8,uint256)._value (contracts/Treasury.sol#1241) is not in mixedCase

Parameter Treasury.setMaxExpansionTiersEntry(uint8,uint256)._index (contracts/Treasury.sol#1254) is not in mixedCase

Parameter Treasury.setMaxExpansionTiersEntry(uint8,uint256)._value (contracts/Treasury.sol#1254) is not in mixedCase

Parameter Treasury.setBondDepletionFloorPercent(uint256)._bondDepletionFloorPercent (contracts/Treasury.sol#1262) is not in mixedCase

Parameter Treasury.setMaxSupplyContractionPercent(uint256)._maxSupplyContractionPercent (contracts/Treasury.sol#1267) is not in mixedCase

Parameter Treasury.setMaxDebtRatioPercent(uint256)._maxDebtRatioPercent (contracts/Treasury.sol#1272) is not in mixedCase

Parameter Treasury.setBootstrap(uint256,uint256)._bootstrapEpochs (contracts/Treasury.sol#1277) is not in mixedCase

Parameter Treasury.setBootstrap(uint256,uint256)._bootstrapSupplyExpansionPercent (contracts/Treasury.sol#1277) is not in mixedCase

Parameter Treasury.setExtraFunds(address,uint256,address,uint256)._daoFund (contracts/Treasury.sol#1285) is not in mixedCase

Parameter Treasury.setExtraFunds(address,uint256,address,uint256)._daoFundSharedPercent (contracts/Treasury.sol#1286) is not in mixedCase

Parameter Treasury.setExtraFunds(address,uint256,address,uint256)._devFund (contracts/Treasury.sol#1287) is not in mixedCase

Parameter Treasury.setExtraFunds(address,uint256,address,uint256)._devFundSharedPercent (contracts/Treasury.sol#1288) is not in mixedCase

Parameter Treasury.setMaxDiscountRate(uint256)._maxDiscountRate (contracts/Treasury.sol#1300) is not in mixedCase

Parameter Treasury.setMaxPremiumRate(uint256)._maxPremiumRate (contracts/Treasury.sol#1304) is not in mixedCase

Parameter Treasury.setDiscountPercent(uint256)._discountPercent (contracts/

Treasury.sol#1308) is not in mixedCase
 Parameter Treasury.setPremiumThreshold(uint256)._premiumThreshold (contracts/
 Treasury.sol#1313) is not in mixedCase
 Parameter Treasury.setPremiumPercent(uint256)._premiumPercent (contracts/
 Treasury.sol#1319) is not in mixedCase
 Parameter Treasury.setMintingFactorForPayingDebt(uint256)._mintingFactorForPayingDebt
 (contracts/Treasury.sol#1324) is not in mixedCase
 Parameter Treasury.buyBonds(uint256,uint256)._tombAmount (contracts/Treasury.sol#1345)
 is not in mixedCase
 Parameter Treasury.redeemBonds(uint256,uint256)._bondAmount (contracts/
 Treasury.sol#1374) is not in mixedCase
 Parameter Treasury.governanceRecoverUnsupported(IERC20,uint256,address)._token
 (contracts/Treasury.sol#1478) is not in mixedCase
 Parameter Treasury.governanceRecoverUnsupported(IERC20,uint256,address)._amount
 (contracts/Treasury.sol#1479) is not in mixedCase
 Parameter Treasury.governanceRecoverUnsupported(IERC20,uint256,address)._to (contracts/
 Treasury.sol#1480) is not in mixedCase
 Parameter Treasury.masonrySetOperator(address)._operator (contracts/Treasury.sol#1489)
 is not in mixedCase
 Parameter Treasury.masonrySetLockUp(uint256,uint256)._withdrawLockupEpochs (contracts/
 Treasury.sol#1493) is not in mixedCase
 Parameter Treasury.masonrySetLockUp(uint256,uint256)._rewardLockupEpochs (contracts/
 Treasury.sol#1493) is not in mixedCase
 Parameter Treasury.masonryGovernanceRecoverUnsupported(address,uint256,address)._token
 (contracts/Treasury.sol#1502) is not in mixedCase
 Parameter Treasury.masonryGovernanceRecoverUnsupported(address,uint256,address)._amount
 (contracts/Treasury.sol#1503) is not in mixedCase
 Parameter Treasury.masonryGovernanceRecoverUnsupported(address,uint256,address)._to
 (contracts/Treasury.sol#1504) is not in mixedCase
 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions>

Redundant expression "this (contracts/GBond.sol#326)" inContext (contracts/
 GBond.sol#320-329)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements>

Redundant expression "this (contracts/GhostShares.sol#109)" inContext (contracts/
 GhostShares.sol#103-112)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements>

Redundant expression "this (contracts/GhostToken.sol#536)" inContext (contracts/GhostToken.sol#530-539)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements>

Redundant expression "this (contracts/Oracle.sol#25)" inContext (contracts/Oracle.sol#19-28)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements>

Redundant expression "this (contracts/Treasury.sol#129)" inContext (contracts/Treasury.sol#123-132)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements>

Variable UniswapV2OracleLibrary.currentCumulativePrices(address).price0Cumulative (contracts/Oracle.sol#357) is too similar to
UniswapV2OracleLibrary.currentCumulativePrices(address).price1Cumulative (contracts/Oracle.sol#358)

Variable Oracle.price0Average (contracts/Oracle.sol#708) is too similar to
Oracle.price1Average (contracts/Oracle.sol#709)

Variable Oracle.update().price0Cumulative (contracts/Oracle.sol#732) is too similar to
Oracle.update().price1Cumulative (contracts/Oracle.sol#732)

Variable Oracle.price0CumulativeLast (contracts/Oracle.sol#706) is too similar to
Oracle.price1CumulativeLast (contracts/Oracle.sol#707)

Variable Oracle.twap(address,uint256).price0Cumulative (contracts/Oracle.sol#763) is too similar to
Oracle.update().price1Cumulative (contracts/Oracle.sol#732)

Variable Oracle.twap(address,uint256).price0Cumulative (contracts/Oracle.sol#763) is too similar to
Oracle.twap(address,uint256).price1Cumulative (contracts/Oracle.sol#763)

Variable Oracle.update().price0Cumulative (contracts/Oracle.sol#732) is too similar to
Oracle.twap(address,uint256).price1Cumulative (contracts/Oracle.sol#763)

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

Variable Treasury.setExtraFunds(address,uint256,address,uint256)._daoFundSharedPercent (contracts/Treasury.sol#1286) is too similar to

Treasury.setExtraFunds(address,uint256,address,uint256)._devFundSharedPercent (contracts/Treasury.sol#1288)

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

[illegible]

`renounceOwnership()` should be declared external:

`transferOwnership(address)` should be declared external:

`operator()` should be declared external:

isOperator() should be declared external:

- ☒- `Operator.isOperator()` (contracts/GBond.sol#435-437)

transferOperator(address) should be declared external:

- ☒- `Operator.transferOperator(address)` (contracts/GBond.sol#439-441)

```
name() should be declared external:
```

- ERC20.name() (contracts/GBond.sol#514-516)

symbol() should be declared external:

- ☒- ERC20.symbol() (contracts/GBond.sol#522-524)

decimals() should be declared external:

- ☒- ERC20.decimals() (contracts/GBond.sol#539-541)

totalSupply() should be declared external:

☒- ERC20.totalSupply() (contracts/GBond.sol#546-548)

```
transfer(address,uint256) should be declared external:
```

- ☒- ERC20.transfer(address,uint256) (contracts/GBond.sol#565-568)

approve(address,uint256) should be declared external:

☒- ERC20.approve(address,uint256) (contracts/GBond.sol#584-587)

transferFrom(address,address,uint256) should be declared external:

- ☒- ERC20.transferFrom(address,address,uint256) (contracts/GBond.sol#602-606)

```
increaseAllowance(address,uint256) should be declared external:
```

☒- ERC20.increaseAllowance(address,uint256) (contracts/GBond.sol#620-623)

decreaseAllowance(address,uint256) should be declared external:

☒- ERC20.decreaseAllowance(address,uint256) (contracts/GBond.sol#639-642)

mint(address,uint256) should be declared external:

☒- GBond.mint(address,uint256) (contracts/GBond.sol#820-826)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external>

balanceOf(address) should be declared external:

☒- ERC20.balanceOf(address) (contracts/GhostShares.sol#553-555)

burnFrom(address,uint256) should be declared external:

☒- ERC20Burnable.burnFrom(address,uint256) (contracts/GhostShares.sol#794-799)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external>

transfer(address,uint256) should be declared external:

☒- ERC20.transfer(address,uint256) (contracts/GhostToken.sol#775-778)

☒- Ghost.transfer(address,uint256) (contracts/GhostToken.sol#1230-1254)

transferFrom(address,address,uint256) should be declared external:

☒- ERC20.transferFrom(address,address,uint256) (contracts/GhostToken.sol#812-816)

☒- Ghost.transferFrom(address,address,uint256) (contracts/GhostToken.sol#1256-1281)

isAddressExcluded(address) should be declared external:

☒- Ghost.isAddressExcluded(address) (contracts/GhostToken.sol#1101-1103)

setTaxTiersTwap(uint8,uint256) should be declared external:

☒- Ghost.setTaxTiersTwap(uint8,uint256) (contracts/GhostToken.sol#1105-1116)

setTaxTiersRate(uint8,uint256) should be declared external:

☒- Ghost.setTaxTiersRate(uint8,uint256) (contracts/GhostToken.sol#1118-1123)

setBurnThreshold(uint256) should be declared external:

☒- Ghost.setBurnThreshold(uint256) (contracts/GhostToken.sol#1125-1127)

enableAutoCalculateTax() should be declared external:

☒- Ghost.enableAutoCalculateTax() (contracts/GhostToken.sol#1149-1151)

disableAutoCalculateTax() should be declared external:

☒- Ghost.disableAutoCalculateTax() (contracts/GhostToken.sol#1153-1155)

setTombOracle(address) should be declared external:

☒- Ghost.setTombOracle(address) (contracts/GhostToken.sol#1157-1160)

setTaxOffice(address) should be declared external:

☒- Ghost.setTaxOffice(address) (contracts/GhostToken.sol#1162-1166)

setTaxCollectorAddress(address) should be declared external:

☒- Ghost.setTaxCollectorAddress(address) (contracts/GhostToken.sol#1168-1171)

setTaxRate(uint256) should be declared external:

☒- Ghost.setTaxRate(uint256) (contracts/GhostToken.sol#1173-1177)

setBuyTax(uint256) should be declared external:

☒- Ghost.setBuyTax(uint256) (contracts/GhostToken.sol#1179-1183)
 setSellTax(uint256) should be declared external:
 ☒- Ghost.setSellTax(uint256) (contracts/GhostToken.sol#1185-1189)
 setUniswapv2Pair(address) should be declared external:
 ☒- Ghost.setUniswapv2Pair(address) (contracts/GhostToken.sol#1191-1194)
 includeAddress(address) should be declared external:
 ☒- Ghost.includeAddress(address) (contracts/GhostToken.sol#1202-1206)
 mint(address,uint256) should be declared external:
 ☒- Ghost.mint(address,uint256) (contracts/GhostToken.sol#1214-1220)
 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external>

set(uint256,uint256) should be declared external:
 ☒- GshareRewardPool.set(uint256,uint256) (contracts/GshareRewardPool.sol#693-702)
 deposit(uint256,uint256) should be declared external:
 ☒- GshareRewardPool.deposit(uint256,uint256) (contracts/GshareRewardPool.sol#764-782)
 withdraw(uint256,uint256) should be declared external:
 ☒- GshareRewardPool.withdraw(uint256,uint256) (contracts/GshareRewardPool.sol#785-802)
 emergencyWithdraw(uint256) should be declared external:
 ☒- GshareRewardPool.emergencyWithdraw(uint256) (contracts/GshareRewardPool.sol#805-813)
 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external>

initialize(IERC20,IERC20,ITreasury) should be declared external:
 ☒- Masonry.initialize(IERC20,IERC20,ITreasury) (contracts/Masonry.sol#755-773)
 rewardPerShare() should be declared external:
 ☒- Masonry.rewardPerShare() (contracts/Masonry.sol#827-829)
 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external>

getCurrentEpoch() should be declared external:
 ☒- Epoch.getCurrentEpoch() (contracts/Oracle.sol#651-653)
 getPeriod() should be declared external:
 ☒- Epoch.getPeriod() (contracts/Oracle.sol#655-657)
 getStartTime() should be declared external:
 ☒- Epoch.getStartTime() (contracts/Oracle.sol#659-661)
 getLastEpochTime() should be declared external:
 ☒- Epoch.getLastEpochTime() (contracts/Oracle.sol#663-665)
 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external>

isInitialized() should be declared external:

☒- Treasury.isInitialized() (contracts/Treasury.sol#1082-1084)

getTombUpdatedPrice() should be declared external:

☒- Treasury.getTombUpdatedPrice() (contracts/Treasury.sol#1100-1106)

getReserve() should be declared external:

☒- Treasury.getReserve() (contracts/Treasury.sol#1109-1111)

getBurnableTombLeft() should be declared external:

☒- Treasury.getBurnableTombLeft() (contracts/Treasury.sol#1113-1125)

getRedeemableBonds() should be declared external:

☒- Treasury.getRedeemableBonds() (contracts/Treasury.sol#1127-1136)

initialize(address,address,address,address,address,uint256) should be declared external:

☒- Treasury.initialize(address,address,address,address,address,address,uint256) (contracts/Treasury.sol#1175-1217)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external>

. analyzed (66 contracts with 77 detectors), 395 result(s) found

