0x Guard

Smart contracts security assessment

Final report

Tariff: Standard

SSS.CASH

April 2022

0xguard.com

hello@0xguard.com

Contents

Introduction

This report has been prepared for the SSS.CASH 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 9efeed8 commit.

Further details about SSS.CASH are available at the official website: https://www.sss.cash/.

Name	SSS.CASH
Audit date	2022-04-13 - 2022-04-14
Language	Solidity
Platform	SmartBCH

Contracts checked

Name	Address
SSS	https://github.com/SSSCASH/contractsmartbch/blob/9efeed8d3ba6b75200e31412b4951c64c5969886/sss.sol
SSHARE	https://github.com/SSSCASH/contractsmartbch/blob/9efeed8d3ba6b75200e31412b4951c64c5969886/sshare.sol
SBOND	https://github.com/SSSCASH/contractsmartbch/blob/9efeed8d3ba6b75200e31412b4951c64c5969886/sbond.sol
SShareRewardPool	https://github.com/SSSCASH/contractsmartbch/blob/9efeed8d3ba6b75200e31412b4951c64c5969886/ssharerewardpool.sol
SSSGenesisRewardPool	https://github.com/SSSCASH/contractsmartbch/blob/9efeed8d3ba6b75200e31412b4951c64c5969886/sssgenesisrewardpool.sol

Oracle	https://github.com/SSSCASH/contractsmartbch/blob/9efeed8d3ba6b75200e31412b4951c64c5969886/oracle.sol
Treasury	https://github.com/SSSCASH/contractsmartbch/blob/9efeed8d3ba6b75200e31412b4951c64c5969886/treasury.sol
TaxOffice	https://github.com/SSSCASH/contractsmartbch/blob/9efeed8d3ba6b75200e31412b4951c64c5969886/taxoffice.sol
Bonus	https://github.com/SSSCASH/contractsmartbch/blob/9efeed8d3ba6b75200e31412b4951c64c5969886/bonus.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. Tax bypass (SSS)

Tax avoidance in the Tomb project is the main problem the team faced. The problem is that there is an invariant in the `transferFrom()` function that deducts tax for the transfer of tokens, but there is also an invariant without deduction of tax that calls the `transfer()` function. Using this problem, you can bypass all tax deductions if you use only the `transfer()` function and it is possible to violate the tokenomics of the project.

Recommendation: It is recommended to overload the `transfer()` function to work with a tax or completely remove the tax functionality in contracts.

Team response: "transfer" is the standard interface of ERC20. This interface is only used for ordinary transfers in our project. The tax deduction designed in our project is tax deduction only for transactions on swap, and ordinary transfers do not deduct tax. Any swap transaction, call It must be the "transferFrom" interface, so this high-risk vulnerability does not exist, because our project requirements are like this.This is not a bug, but a normal requirement of our project.There is no tax deduction for transactions through the "transfer" interface, which is completely fine.

2. Contract ownership (Multiple contracts)

1) The projects owner can change the `taxRate` in SSS token up to 100% in the `setTaxRate()` function if the owner changes the `taxOffice` in the `setTaxOffice()` function to a compromised address.

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

3) The `governanceRecoverUnsupported()` function (found in the `SSS`, `SShare`, `SShareRewardPool`, and `SssGenesisRewardPool` contracts) can remove all tokens from the contract balance if the operator role is compromised.

Recommendation: There is 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 participation. 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 discovered, add a call to them via a multisig wallet. This helps to avoid the issue of owner compromise.

Low severity issues

1. Reentrancy attack (SShareRewardPool)

When withdrawing, some pool tokens may be subject to a reentrancy attack. The variable `user.rewardDebt` on 735L is updated after calling `pool.token.safeTransfer()`.

Recommendation: It is recommended to update the value of the `user.rewardDebt` variable before calling `pool.token.safeTransfer()`.

2. Unused variable (SSSGenesisRewardPool)

State variable `feewallet` is not used anywhere.

Recommendation: Remove all references to this variable from the contract.

3. Reentrancy attack (SSSGenesisRewardPool)

When withdrawing, some pool tokens may be subject to a reentrancy attack. The variable `user.rewardDebt` on 745L is updated after calling `pool.token.safeTransfer()`.

Recommendation: It is recommended to update the value of the `user.rewardDebt` variable before calling `pool.token.safeTransfer()`.

4. Few events (Multiple contracts)

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

Recommendation: Create events for these set functions.

Conclusion

SSS.CASH SSS, SSHARE, SBOND, SShareRewardPool, SSSGenesisRewardPool, Oracle, Treasury, TaxOffice, Bonus, Multiple contracts contracts were audited. 2 medium, 4 low severity issues were found.

The SSS.CASH Project was compared with the Tomb Project. SSS.CASH has changed the implementation of `Treasury` contract.

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#386-388) uses a
weak PRNG: "uint32(block.timestamp % 2 ** 32) (contracts/oracle.sol#387)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Reentrancy in Masonry.stake(uint256) (contracts/bonus.sol#750-755):
⊠External calls:
⊠- super.stake(amount) (contracts/bonus.sol#752)
⊠⊠- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/bonus.sol#505)
⊠⊠- share.safeTransferFrom(msg.sender,address(this),amount) (contracts/bonus.sol#580)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/bonus.sol#382)
⊠External calls sending eth:
⊠- super.stake(amount) (contracts/bonus.sol#752)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/bonus.sol#382)
⊠State variables written after the call(s):
⊠- masons[msg.sender].epochTimerStart = treasury.epoch() (contracts/bonus.sol#753)
Reentrancy in Masonry.withdraw(uint256) (contracts/bonus.sol#757-763):
⊠External calls:
⊠- claimReward() (contracts/bonus.sol#760)
⊠⊠- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/bonus.sol#505)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/bonus.sol#382)
⊠⊠- sss.safeTransfer(msg.sender,reward) (contracts/bonus.sol#775)
⊠- super.withdraw(amount) (contracts/bonus.sol#761)
⊠⊠- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/bonus.sol#505)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/bonus.sol#382)
⊠⊠- share.safeTransfer(msg.sender,amount) (contracts/bonus.sol#588)
⊠External calls sending eth:
⊠- claimReward() (contracts/bonus.sol#760)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/bonus.sol#382)
⊠- super.withdraw(amount) (contracts/bonus.sol#761)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/bonus.sol#382)
⊠State variables written after the call(s):
⊠- super.withdraw(amount) (contracts/bonus.sol#761)
⊠⊠- _balances[msg.sender] = masonShare.sub(amount) (contracts/bonus.sol#587)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities
```

```
Reentrancy in SShareRewardPool.deposit(uint256,uint256) (contracts/
sharerewardpool.sol#699-717):
⊠External calls:
⊠- safeSShareTransfer(_sender,_pending) (contracts/sharerewardpool.sol#707)
⊠⊠- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/sharerewardpool.sol#309)
⊠⊠- sshare.safeTransfer(_to,_sshareBal) (contracts/sharerewardpool.sol#755)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/
sharerewardpool.sol#186)
⊠⊠- sshare.safeTransfer(_to,_amount) (contracts/sharerewardpool.sol#757)
⊠- pool.token.safeTransferFrom(_sender,address(this),_amount) (contracts/
sharerewardpool.sol#712)
⊠External calls sending eth:
⊠- safeSShareTransfer(_sender,_pending) (contracts/sharerewardpool.sol#707)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/
sharerewardpool.sol#186)
⊠State variables written after the call(s):
⊠- user.amount = user.amount.add(_amount) (contracts/sharerewardpool.sol#713)
⊠- user.rewardDebt = user.amount.mul(pool.accSSharePerShare).div(1e18) (contracts/
sharerewardpool.sol#715)
Reentrancy in SShareRewardPool.withdraw(uint256,uint256) (contracts/
sharerewardpool.sol#720-737):
⊠External calls:
⊠- safeSShareTransfer(_sender,_pending) (contracts/sharerewardpool.sol#728)
⊠⊠- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/sharerewardpool.sol#309)
⊠⊠- sshare.safeTransfer(_to,_sshareBal) (contracts/sharerewardpool.sol#755)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/
sharerewardpool.sol#186)
⊠⊠- sshare.safeTransfer(_to,_amount) (contracts/sharerewardpool.sol#757)
⊠External calls sending eth:
⊠- safeSShareTransfer(_sender,_pending) (contracts/sharerewardpool.sol#728)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/
sharerewardpool.sol#186)
⊠State variables written after the call(s):
⊠- user.amount = user.amount.sub(_amount) (contracts/sharerewardpool.sol#732)
Reentrancy in SShareRewardPool.withdraw(uint256,uint256) (contracts/
sharerewardpool.sol#720-737):
⊠External calls:
⊠- safeSShareTransfer(_sender,_pending) (contracts/sharerewardpool.sol#728)
```

```
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/ssharerewardpool.sol#309)
		- sshare.safeTransfer(_to,_sshareBal) (contracts/sharerewardpool.sol#755)
		- (success,returndata) = target.call{value: value}(data) (contracts/
ssharerewardpool.sol#186)
		- sshare.safeTransfer(_to,_amount) (contracts/sharerewardpool.sol#757)
	- pool.token.safeTransfer(_sender,_amount) (contracts/sharerewardpool.sol#733)
	External calls sending eth:
	- safeSShareTransfer(_sender,_pending) (contracts/sharerewardpool.sol#728)
		- (success,returndata) = target.call{value: value}(data) (contracts/
sharerewardpool.sol#186)
	State variables written after the call(s):
	- user.rewardDebt = user.amount.mul(pool.accSSharePerShare).div(1e18) (contracts/
sharerewardpool.sol#735)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities

Reentrancy in SssGenesisRewardPool.deposit(uint256,uint256) (contracts/
sssgenesisrewardpool.sol#709-727):
	External calls:
	- safeSssTransfer(_sender,_pending) (contracts/sssgenesisrewardpool.sol#717)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/sssgenesisrewardpool.sol#507)
		- sss.safeTransfer(_to,_sssBalance) (contracts/sssgenesisrewardpool.sol#765)
		- sss.safeTransfer(_to,_amount) (contracts/sssgenesisrewardpool.sol#767)
		- (success,returndata) = target.call{value: value}(data) (contracts/
sssgenesisrewardpool.sol#384)
	- pool.token.safeTransferFrom(_sender,address(this),_amount) (contracts/
sssgenesisrewardpool.sol#722)
	External calls sending eth:
	- safeSssTransfer(_sender,_pending) (contracts/sssgenesisrewardpool.sol#717)
		- (success,returndata) = target.call{value: value}(data) (contracts/
sssgenesisrewardpool.sol#384)
	State variables written after the call(s):
	- user.amount = user.amount.add(_amount) (contracts/sssgenesisrewardpool.sol#723)
	- user.rewardDebt = user.amount.mul(pool.accSssPerShare).div(1e18) (contracts/
sssgenesisrewardpool.sol#725)
Reentrancy in SssGenesisRewardPool.withdraw(uint256,uint256) (contracts/
sssgenesisrewardpool.sol#730-747):
	External calls:
	- safeSssTransfer(_sender,_pending) (contracts/sssgenesisrewardpool.sol#738)
```

```
⊠⊠- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/sssgenesisrewardpool.sol#507)
⊠⊠- sss.safeTransfer(_to,_sssBalance) (contracts/sssgenesisrewardpool.sol#765)
⊠⊠- sss.safeTransfer(_to,_amount) (contracts/sssgenesisrewardpool.sol#767)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/
sssgenesisrewardpool.sol#384)
⊠External calls sending eth:
⊠- safeSssTransfer(_sender,_pending) (contracts/sssgenesisrewardpool.sol#738)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/
sssgenesisrewardpool.sol#384)
⊠State variables written after the call(s):
⊠- user.amount = user.amount.sub(_amount) (contracts/sssgenesisrewardpool.sol#742)
Reentrancy in SssGenesisRewardPool.withdraw(uint256,uint256) (contracts/
sssgenesisrewardpool.sol#730-747):
⊠External calls:
⊠- safeSssTransfer(_sender,_pending) (contracts/sssgenesisrewardpool.sol#738)
⊠⊠- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/sssgenesisrewardpool.sol#507)
⊠⊠- sss.safeTransfer(_to,_sssBalance) (contracts/sssgenesisrewardpool.sol#765)
⊠⊠- sss.safeTransfer(_to,_amount) (contracts/sssgenesisrewardpool.sol#767)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/
sssgenesisrewardpool.sol#384)
⊠- pool.token.safeTransfer(_sender,_amount) (contracts/sssgenesisrewardpool.sol#743)
⊠External calls sending eth:
⊠- safeSssTransfer(_sender,_pending) (contracts/sssgenesisrewardpool.sol#738)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/
sssgenesisrewardpool.sol#384)
⊠State variables written after the call(s):
⊠- user.rewardDebt = user.amount.mul(pool.accSssPerShare).div(1e18) (contracts/
sssgenesisrewardpool.sol#745)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities

Reentrancy in Treasury.allocateSeigniorage() (contracts/treasury.sol#1242-1282):
⊠External calls:
⊠- _updateSssPrice() (contracts/treasury.sol#1243)
⊠⊠- IOracle(sssOracle).update() (contracts/treasury.sol#1139)
⊠- _sendToMasonry(_savedForMasonry) (contracts/treasury.sol#1273)
⊠⊠- IBasisAsset(sss).mint(address(this),_amount) (contracts/treasury.sol#1208)
⊠⊠- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/treasury.sol#530)
```

```
		- (success,returndata) = target.call{value: value}(data) (contracts/treasury.sol#407)
		- IERC20(sss).transfer(daoFund,_daoFundSharedAmount) (contracts/treasury.sol#1213)
		- IERC20(sss).transfer(devFund,_devFundSharedAmount) (contracts/treasury.sol#1220)
		- IERC20(sss).safeApprove(masonry,0) (contracts/treasury.sol#1226)
		- IERC20(sss).safeApprove(masonry,_amount) (contracts/treasury.sol#1227)
		- IMasonry(masonry).allocateSeigniorage(_amount) (contracts/treasury.sol#1228)
	External calls sending eth:
	- _sendToMasonry(_savedForMasonry) (contracts/treasury.sol#1273)
		- (success,returndata) = target.call{value: value}(data) (contracts/treasury.sol#407)
	State variables written after the call(s):
	- seigniorageSaved = seigniorageSaved.add(_savedForBond) (contracts/treasury.sol#1276)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

SShare.governanceRecoverUnsupported(IERC20,uint256,address) (contracts/sshare.sol#772-778) ignores return value by _token.transfer(_to,_amount) (contracts/sshare.sol#777)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

Sss.governanceRecoverUnsupported(IERC20,uint256,address) (contracts/sss.sol#1104-1110) ignores return value by _token.transfer(_to,_amount) (contracts/sss.sol#1109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

TaxOfficeV2.addLiquidityTaxFree(address,uint256,uint256,uint256,uint256) (contracts/taxoffice.sol#604-649) ignores return value by IERC20(sss).transferFrom(msg.sender,address(this),amtSss) (contracts/taxoffice.sol#621)
TaxOfficeV2.addLiquidityTaxFree(address,uint256,uint256,uint256,uint256) (contracts/taxoffice.sol#604-649) ignores return value by IERC20(token).transferFrom(msg.sender,address(this),amtToken) (contracts/taxoffice.sol#622)
TaxOfficeV2.addLiquidityTaxFree(address,uint256,uint256,uint256,uint256) (contracts/taxoffice.sol#604-649) ignores return value by IERC20(sss).transfer(msg.sender,amtSss.sub(resultAmtSss)) (contracts/taxoffice.sol#643)
TaxOfficeV2.addLiquidityTaxFree(address,uint256,uint256,uint256,uint256) (contracts/taxoffice.sol#604-649) ignores return value by IERC20(token).transfer(msg.sender,amtToken.sub(resultAmtToken)) (contracts/taxoffice.sol#646)
TaxOfficeV2.addLiquidityETHTaxFree(uint256,uint256,uint256) (contracts/taxoffice.sol#651-688) ignores return value by
```

```
IERC20(sss).transferFrom(msg.sender,address(this),amtSss) (contracts/taxoffice.sol#667)
TaxOfficeV2.addLiquidityETHTaxFree(uint256,uint256,uint256) (contracts/
taxoffice.sol#651-688) ignores return value by
IERC20(sss).transfer(msg.sender,amtSss.sub(resultAmtSss)) (contracts/taxoffice.sol#685)
TaxOfficeV2.taxFreeTransferFrom(address,address,uint256) (contracts/
taxoffice.sol#698-707) ignores return value by
IERC20(sss).transferFrom(_sender,_recipient,_amt) (contracts/taxoffice.sol#705)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-
transfer

Treasury._sendToMasonry(uint256) (contracts/treasury.sol#1207-1230) ignores return
value by IERC20(sss).transfer(daoFund,_daoFundSharedAmount) (contracts/
treasury.sol#1213)
Treasury._sendToMasonry(uint256) (contracts/treasury.sol#1207-1230) ignores return
value by IERC20(sss).transfer(devFund,_devFundSharedAmount) (contracts/
treasury.sol#1220)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-
transfer

SShareRewardPool.pendingShare(uint256,address) (contracts/sharerewardpool.sol#654-665)
performs a multiplication on the result of a division:
⊠-_sshareReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint) (contracts/
ssharerewardpool.sol#661)
⊠-accSSharePerShare = accSSharePerShare.add(_sshareReward.mul(1e18).div(tokenSupply))
(contracts/ssharerewardpool.sol#662)
SShareRewardPool.updatePool(uint256) (contracts/ssharerewardpool.sol#676-696) performs
a multiplication on the result of a division:
⊠-_sshareReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint) (contracts/
ssharerewardpool.sol#692)
⊠-pool.accSSharePerShare =
pool.accSSharePerShare.add(_sshareReward.mul(1e18).div(tokenSupply)) (contracts/
ssharerewardpool.sol#693)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply

SssGenesisRewardPool.pendingSSS(uint256,address) (contracts/
sssgenesisrewardpool.sol#664-675) performs a multiplication on the result of a
division:
⊠-_sssReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint) (contracts/
sssgenesisrewardpool.sol#671)
⊠-accSssPerShare = accSssPerShare.add(_sssReward.mul(1e18).div(tokenSupply)) (contracts/
sssgenesisrewardpool.sol#672)
```

```
SssGenesisRewardPool.updatePool(uint256) (contracts/sssgenesisrewardpool.sol#686-706)
performs a multiplication on the result of a division:
⊠-_sssReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint) (contracts/
sssgenesisrewardpool.sol#702)
⊠-pool.accSssPerShare = pool.accSssPerShare.add(_sssReward.mul(1e18).div(tokenSupply))
(contracts/sssgenesisrewardpool.sol#703)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply

Treasury.allocateSeigniorage() (contracts/treasury.sol#1242-1282) performs a
multiplication on the result of a division:
⊠-_seigniorage = sssSupply.mul(_percentage).div(1e18) (contracts/treasury.sol#1265)
⊠-_savedForMasonry = _seigniorage.mul(seigniorageExpansionFloorPercent).div(10000)
(contracts/treasury.sol#1266)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply

SShareRewardPool.updatePool(uint256) (contracts/sshare rewardpool.sol#676-696) uses a
dangerous strict equality:
⊠- tokenSupply == 0 (contracts/sshare rewardpool.sol#682)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-
strict-equalities

SssGenesisRewardPool.updatePool(uint256) (contracts/sssgenesisrewardpool.sol#686-706)
uses a dangerous strict equality:
⊠- tokenSupply == 0 (contracts/sssgenesisrewardpool.sol#692)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-
strict-equalities

Reentrancy in Treasury.buyBonds(uint256,uint256) (contracts/treasury.sol#1152-1179):
⊠External calls:
⊠- IBasisAsset(sss).burnFrom(msg.sender,_sssAmount) (contracts/treasury.sol#1172)
⊠- IBasisAsset(sbond).mint(msg.sender,_bondAmount) (contracts/treasury.sol#1173)
⊠State variables written after the call(s):
⊠- epochSupplyContractionLeft = epochSupplyContractionLeft.sub(_sssAmount) (contracts/
treasury.sol#1175)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1

Sss.setTaxTiersTwap(uint8,uint256) (contracts/sss.sol#929-940) contains a tautology or
contradiction:
```

```
⊠- require(bool,string)(_index >= 0,Index has to be higher than 0) (contracts/
sss.sol#930)
Sss.setTaxTiersRate(uint8,uint256) (contracts/sss.sol#942-947) contains a tautology or
contradiction:
⊠- require(bool,string)(_index >= 0,Index has to be higher than 0) (contracts/
sss.sol#943)
Sss._updateTaxRate(uint256) (contracts/sss.sol#961-971) contains a tautology or
contradiction:
⊠- tierId >= 0 (contracts/sss.sol#963)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-
contradiction

Treasury.setSupplyTiersEntry(uint8,uint256) (contracts/treasury.sol#1048-1059) contains
a tautology or contradiction:
⊠- require(bool,string)(_index >= 0,Index has to be higher than 0) (contracts/
treasury.sol#1049)
Treasury.setMaxExpansionTiersEntry(uint8,uint256) (contracts/treasury.sol#1061-1067)
contains a tautology or contradiction:
⊠- require(bool,string)(_index >= 0,Index has to be higher than 0) (contracts/
treasury.sol#1062)
Treasury._calculateMaxSupplyExpansionPercent(uint256) (contracts/
treasury.sol#1232-1240) contains a tautology or contradiction:
⊠- tierId >= 0 (contracts/treasury.sol#1233)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-
contradiction

FixedPoint.mul(FixedPoint.uq112x112,uint256).z (contracts/oracle.sol#349) is a local
variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-
local-variables

Sss._getSssPrice()._price (contracts/sss.sol#954) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-
local-variables

Treasury.allocateSeigniorage()._savedForBond (contracts/treasury.sol#1254) is a local
variable never initialized
Treasury.getSssUpdatedPrice().price (contracts/treasury.sol#909) is a local variable
never initialized
Treasury.getSssPrice().price (contracts/treasury.sol#901) is a local variable never
initialized
```

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

Sss._getSssPrice() (contracts/sss.sol#953-959) ignores return value by
IOracle(sssOracle).consult(address(this),1e18) (contracts/sss.sol#954-958)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

TaxOfficeV2._approveTokenIfNeeded(address,address) (contracts/taxoffice.sol#713-717)
ignores return value by IERC20(_token).approve(_router,type()(uint256).max) (contracts/
taxoffice.sol#715)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Treasury.getSssPrice() (contracts/treasury.sol#900-906) ignores return value by
IOracle(sssOracle).consult(sss,1e18) (contracts/treasury.sol#901-905)
Treasury.getSssUpdatedPrice() (contracts/treasury.sol#908-914) ignores return value by
IOracle(sssOracle).twap(sss,1e18) (contracts/treasury.sol#909-913)
Treasury.buyBonds(uint256,uint256) (contracts/treasury.sol#1152-1179) ignores return
value by IBasisAsset(sbond).mint(msg.sender,_bondAmount) (contracts/treasury.sol#1173)
Treasury._sendToMasonry(uint256) (contracts/treasury.sol#1207-1230) ignores return
value by IBasisAsset(sss).mint(address(this),_amount) (contracts/treasury.sol#1208)
Treasury.allocateSeigniorage() (contracts/treasury.sol#1242-1282) ignores return value
by IBasisAsset(sss).mint(address(this),_savedForBond) (contracts/treasury.sol#1277)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Masonry.setOperator(address) (contracts/bonus.sol#685-687) should emit an event for:
⊠- operator = _operator (contracts/bonus.sol#686)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
access-control

SShareRewardPool.setOperator(address) (contracts/ssharerewardpool.sol#762-764) should
emit an event for:
⊠- operator = _operator (contracts/ssharerewardpool.sol#763)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
access-control

SssGenesisRewardPool.setOperator(address) (contracts/sssgenesisrewardpool.sol#772-774)
should emit an event for:
⊠- operator = _operator (contracts/sssgenesisrewardpool.sol#773)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
access-control
```

```
Treasury.setOperator(address) (contracts/treasury.sol#1026-1028) should emit an event
for:
⊠- operator = _operator (contracts/treasury.sol#1027)
Treasury.setMasonry(address) (contracts/treasury.sol#1030-1032) should emit an event
for:
⊠- masonry = _masonry (contracts/treasury.sol#1031)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
access-control

Masonry.setLockUp(uint256,uint256) (contracts/bonus.sol#689-693) should emit an event
for:
⊠- withdrawLockupEpochs = _withdrawLockupEpochs (contracts/bonus.sol#691)
⊠- rewardLockupEpochs = _rewardLockupEpochs (contracts/bonus.sol#692)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic

SShareRewardPool.add(uint256,IERC20,bool,uint256) (contracts/
ssharerewardpool.sol#587-625) should emit an event for:
⊠- totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/
ssharerewardpool.sol#623)
SShareRewardPool.set(uint256,uint256) (contracts/ssharerewardpool.sol#628-637) should
emit an event for:
⊠- totalAllocPoint = totalAllocPoint.sub(pool.allocPoint).add(_allocPoint) (contracts/
ssharerewardpool.sol#632-634)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic

Sss.setBurnThreshold(uint256) (contracts/sss.sol#949-951) should emit an event for:
⊠- burnThreshold = _burnThreshold (contracts/sss.sol#950)
Sss.setTaxRate(uint256) (contracts/sss.sol#997-1001) should emit an event for:
⊠- taxRate = _taxRate (contracts/sss.sol#1000)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic

SssGenesisRewardPool.add(uint256,IERC20,bool,uint256) (contracts/
sssgenesisrewardpool.sol#597-635) should emit an event for:
⊠- totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/
sssgenesisrewardpool.sol#633)
SssGenesisRewardPool.set(uint256,uint256) (contracts/sssgenesisrewardpool.sol#638-647)
should emit an event for:
⊠- totalAllocPoint = totalAllocPoint.sub(pool.allocPoint).add(_allocPoint) (contracts/
sssgenesisrewardpool.sol#642-644)
```

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

Treasury.setSssPriceCeiling(uint256) (contracts/treasury.sol#1038-1041) should emit an
event for:
⊠- sssPriceCeiling = _sssPriceCeiling (contracts/treasury.sol#1040)
Treasury.setMaxSupplyExpansionPercents(uint256) (contracts/treasury.sol#1043-1046)
should emit an event for:
⊠- maxSupplyExpansionPercent = _maxSupplyExpansionPercent (contracts/
treasury.sol#1045)
Treasury.setBondDepletionFloorPercent(uint256) (contracts/treasury.sol#1069-1072)
should emit an event for:
⊠- bondDepletionFloorPercent = _bondDepletionFloorPercent (contracts/
treasury.sol#1071)
Treasury.setMaxDebtRatioPercent(uint256) (contracts/treasury.sol#1079-1082) should emit
an event for:
⊠- maxDebtRatioPercent = _maxDebtRatioPercent (contracts/treasury.sol#1081)
Treasury.setBootstrap(uint256,uint256) (contracts/treasury.sol#1084-1089) should emit
an event for:
⊠- bootstrapEpochs = _bootstrapEpochs (contracts/treasury.sol#1087)
⊠- bootstrapSupplyExpansionPercent = _bootstrapSupplyExpansionPercent (contracts/
treasury.sol#1088)
Treasury.setExtraFunds(address,uint256,address,uint256) (contracts/
treasury.sol#1091-1105) should emit an event for:
⊠- daoFundSharedPercent = _daoFundSharedPercent (contracts/treasury.sol#1102)
⊠- devFundSharedPercent = _devFundSharedPercent (contracts/treasury.sol#1104)
Treasury.setMaxDiscountRate(uint256) (contracts/treasury.sol#1107-1109) should emit an
event for:
⊠- maxDiscountRate = _maxDiscountRate (contracts/treasury.sol#1108)
Treasury.setMaxPremiumRate(uint256) (contracts/treasury.sol#1111-1113) should emit an
event for:
⊠- maxPremiumRate = _maxPremiumRate (contracts/treasury.sol#1112)
Treasury.setDiscountPercent(uint256) (contracts/treasury.sol#1115-1118) should emit an
event for:
⊠- discountPercent = _discountPercent (contracts/treasury.sol#1117)
Treasury.setPremiumThreshold(uint256) (contracts/treasury.sol#1120-1124) should emit an
event for:
⊠- premiumThreshold = _premiumThreshold (contracts/treasury.sol#1123)
Treasury.setPremiumPercent(uint256) (contracts/treasury.sol#1126-1129) should emit an
event for:
⊠- premiumPercent = _premiumPercent (contracts/treasury.sol#1128)
```

```
Treasury.setMintingFactorForPayingDebt(uint256) (contracts/treasury.sol#1131-1134)
should emit an event for:
⊠- mintingFactorForPayingDebt = _mintingFactorForPayingDebt (contracts/
treasury.sol#1133)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic

Masonry.setOperator(address)._operator (contracts/bonus.sol#685) lacks a zero-check
on :
⊠⊠- operator = _operator (contracts/bonus.sol#686)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
address-validation

SShare.setTreasuryFund(address)._communityFund (contracts/sshare.sol#717) lacks a zero-
check on :
⊠⊠- communityFund = _communityFund (contracts/sshare.sol#719)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
address-validation

SShareRewardPool.setOperator(address)._operator (contracts/ssharerewardpool.sol#762)
lacks a zero-check on :
⊠⊠- operator = _operator (contracts/ssharerewardpool.sol#763)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
address-validation

SssGenesisRewardPool.setOperator(address)._operator (contracts/
sssgenesisrewardpool.sol#772) lacks a zero-check on :
⊠⊠- operator = _operator (contracts/sssgenesisrewardpool.sol#773)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
address-validation

Treasury.initialize(address,address,address,address,address,uint256)._sss (contracts/
treasury.sol#983) lacks a zero-check on :
⊠⊠- sss = _sss (contracts/treasury.sol#990)
Treasury.initialize(address,address,address,address,address,uint256)._sbond (contracts/
treasury.sol#984) lacks a zero-check on :
⊠⊠- sbond = _sbond (contracts/treasury.sol#991)
Treasury.initialize(address,address,address,address,address,uint256)._sshare (contracts/
treasury.sol#985) lacks a zero-check on :
⊠⊠- sshare = _sshare (contracts/treasury.sol#992)
Treasury.initialize(address,address,address,address,address,uint256)._sssOracle
```

```
(contracts/treasury.sol#986) lacks a zero-check on :
⊠⊠- sssOracle = _sssOracle (contracts/treasury.sol#993)
Treasury.initialize(address,address,address,address,address,uint256)._masonry
(contracts/treasury.sol#987) lacks a zero-check on :
⊠⊠- masonry = _masonry (contracts/treasury.sol#994)
Treasury.setOperator(address)._operator (contracts/treasury.sol#1026) lacks a zero-
check on :
⊠⊠- operator = _operator (contracts/treasury.sol#1027)
Treasury.setMasonry(address)._masonry (contracts/treasury.sol#1030) lacks a zero-check
on :
⊠⊠- masonry = _masonry (contracts/treasury.sol#1031)
Treasury.setSssOracle(address)._sssOracle (contracts/treasury.sol#1034) lacks a zero-
check on :
⊠⊠- sssOracle = _sssOracle (contracts/treasury.sol#1035)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
address-validation

SShareRewardPool.updatePool(uint256) (contracts/ssharerewardpool.sol#676-696) has
external calls inside a loop: tokenSupply = pool.token.balanceOf(address(this))
(contracts/ssharerewardpool.sol#681)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-
a-loop

SssGenesisRewardPool.updatePool(uint256) (contracts/sssgenesisrewardpool.sol#686-706)
has external calls inside a loop: tokenSupply = pool.token.balanceOf(address(this))
(contracts/sssgenesisrewardpool.sol#691)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-
a-loop

Treasury.getSssCirculatingSupply() (contracts/treasury.sol#1142-1150) has external
calls inside a loop: balanceExcluded =
balanceExcluded.add(sssErc20.balanceOf(excludedFromTotalSupply[entryId])) (contracts/
treasury.sol#1147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-
a-loop

Variable 'Sss._getSssPrice()._price (contracts/sss.sol#954)' in Sss._getSssPrice()
(contracts/sss.sol#953-959) potentially used before declaration: uint256(_price)
(contracts/sss.sol#955)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-
declaration-usage-of-local-variables
```

```
Variable 'Treasury.getSssPrice().price (contracts/treasury.sol#901)' in
Treasury.getSssPrice() (contracts/treasury.sol#900-906) potentially used before
declaration: uint256(price) (contracts/treasury.sol#902)
Variable 'Treasury.getSssUpdatedPrice().price (contracts/treasury.sol#909)' in
Treasury.getSssUpdatedPrice() (contracts/treasury.sol#908-914) potentially used before
declaration: uint256(price) (contracts/treasury.sol#910)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-
declaration-usage-of-local-variables

Reentrancy in Treasury.allocateSeigniorage() (contracts/treasury.sol#1242-1282):
	External calls:
	- _updateSssPrice() (contracts/treasury.sol#1243)
		- IOracle(sssOracle).update() (contracts/treasury.sol#1139)
	State variables written after the call(s):
	- _mse = _calculateMaxSupplyExpansionPercent(sssSupply).mul(1e14) (contracts/
treasury.sol#1256)
		- maxSupplyExpansionPercent = maxExpansionTiers[tierId] (contracts/treasury.sol#1235)
	- previousEpochSssPrice = getSssPrice() (contracts/treasury.sol#1244)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2

Reentrancy in Masonry.allocateSeigniorage(uint256) (contracts/bonus.sol#780-797):
	External calls:
	- sss.safeTransferFrom(msg.sender,address(this),amount) (contracts/bonus.sol#795)
	Event emitted after the call(s):
	- RewardAdded(msg.sender,amount) (contracts/bonus.sol#796)
Reentrancy in Masonry.claimReward() (contracts/bonus.sol#769-778):
	External calls:
	- sss.safeTransfer(msg.sender,reward) (contracts/bonus.sol#775)
	Event emitted after the call(s):
	- RewardPaid(msg.sender,reward) (contracts/bonus.sol#776)
Reentrancy in Masonry.stake(uint256) (contracts/bonus.sol#750-755):
	External calls:
	- super.stake(amount) (contracts/bonus.sol#752)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/bonus.sol#505)
		- share.safeTransferFrom(msg.sender,address(this),amount) (contracts/bonus.sol#580)
		- (success,returndata) = target.call{value: value}(data) (contracts/bonus.sol#382)
	External calls sending eth:
	- super.stake(amount) (contracts/bonus.sol#752)
```

```
		- (success,returndata) = target.call{value: value}(data) (contracts/bonus.sol#382)
	Event emitted after the call(s):
	- Staked(msg.sender,amount) (contracts/bonus.sol#754)
Reentrancy in Masonry.withdraw(uint256) (contracts/bonus.sol#757-763):
	External calls:
	- claimReward() (contracts/bonus.sol#760)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/bonus.sol#505)
		- (success,returndata) = target.call{value: value}(data) (contracts/bonus.sol#382)
		- sss.safeTransfer(msg.sender,reward) (contracts/bonus.sol#775)
	- super.withdraw(amount) (contracts/bonus.sol#761)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/bonus.sol#505)
		- (success,returndata) = target.call{value: value}(data) (contracts/bonus.sol#382)
		- share.safeTransfer(msg.sender,amount) (contracts/bonus.sol#588)
	External calls sending eth:
	- claimReward() (contracts/bonus.sol#760)
		- (success,returndata) = target.call{value: value}(data) (contracts/bonus.sol#382)
	- super.withdraw(amount) (contracts/bonus.sol#761)
		- (success,returndata) = target.call{value: value}(data) (contracts/bonus.sol#382)
	Event emitted after the call(s):
	- Withdrawn(msg.sender,amount) (contracts/bonus.sol#762)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3

Reentrancy in SShareRewardPool.deposit(uint256,uint256) (contracts/
ssharerewardpool.sol#699-717):
	External calls:
	- safeSShareTransfer(_sender,_pending) (contracts/ssharerewardpool.sol#707)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/ssharerewardpool.sol#309)
		- sshare.safeTransfer(_to,_sshareBal) (contracts/ssharerewardpool.sol#755)
		- (success,returndata) = target.call{value: value}(data) (contracts/
ssharerewardpool.sol#186)
		- sshare.safeTransfer(_to,_amount) (contracts/ssharerewardpool.sol#757)
	External calls sending eth:
	- safeSShareTransfer(_sender,_pending) (contracts/ssharerewardpool.sol#707)
		- (success,returndata) = target.call{value: value}(data) (contracts/
ssharerewardpool.sol#186)
	Event emitted after the call(s):
	- RewardPaid(_sender,_pending) (contracts/ssharerewardpool.sol#708)
```

```
Reentrancy in SShareRewardPool.deposit(uint256,uint256) (contracts/
sharerewardpool.sol#699-717):
	External calls:
	- safeSShareTransfer(_sender,_pending) (contracts/ssharerewardpool.sol#707)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/ssharerewardpool.sol#309)
		- sshare.safeTransfer(_to,_sshareBal) (contracts/ssharerewardpool.sol#755)
		- (success,returndata) = target.call{value: value}(data) (contracts/
ssharerewardpool.sol#186)
		- sshare.safeTransfer(_to,_amount) (contracts/ssharerewardpool.sol#757)
	- pool.token.safeTransferFrom(_sender,address(this),_amount) (contracts/
ssharerewardpool.sol#712)
	External calls sending eth:
	- safeSShareTransfer(_sender,_pending) (contracts/ssharerewardpool.sol#707)
		- (success,returndata) = target.call{value: value}(data) (contracts/
ssharerewardpool.sol#186)
	Event emitted after the call(s):
	- Deposit(_sender,_pid,_amount) (contracts/ssharerewardpool.sol#716)
Reentrancy in SShareRewardPool.emergencyWithdraw(uint256) (contracts/
ssharerewardpool.sol#740-748):
	External calls:
	- pool.token.safeTransfer(msg.sender,_amount) (contracts/ssharerewardpool.sol#746)
	Event emitted after the call(s):
	- EmergencyWithdraw(msg.sender,_pid,_amount) (contracts/ssharerewardpool.sol#747)
Reentrancy in SShareRewardPool.withdraw(uint256,uint256) (contracts/
ssharerewardpool.sol#720-737):
	External calls:
	- safeSShareTransfer(_sender,_pending) (contracts/ssharerewardpool.sol#728)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/ssharerewardpool.sol#309)
		- sshare.safeTransfer(_to,_sshareBal) (contracts/ssharerewardpool.sol#755)
		- (success,returndata) = target.call{value: value}(data) (contracts/
ssharerewardpool.sol#186)
		- sshare.safeTransfer(_to,_amount) (contracts/ssharerewardpool.sol#757)
	External calls sending eth:
	- safeSShareTransfer(_sender,_pending) (contracts/ssharerewardpool.sol#728)
		- (success,returndata) = target.call{value: value}(data) (contracts/
ssharerewardpool.sol#186)
	Event emitted after the call(s):
	- RewardPaid(_sender,_pending) (contracts/ssharerewardpool.sol#729)
Reentrancy in SShareRewardPool.withdraw(uint256,uint256) (contracts/
ssharerewardpool.sol#720-737):
	External calls:
	- safeSShareTransfer(_sender,_pending) (contracts/sharerewardpool.sol#728)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/sharerewardpool.sol#309)
		- sshare.safeTransfer(_to,_sshareBal) (contracts/sharerewardpool.sol#755)
		- (success,returndata) = target.call{value: value}(data) (contracts/
sharerewardpool.sol#186)
		- sshare.safeTransfer(_to,_amount) (contracts/sharerewardpool.sol#757)
	- pool.token.safeTransfer(_sender,_amount) (contracts/sharerewardpool.sol#733)
	External calls sending eth:
	- safeSShareTransfer(_sender,_pending) (contracts/sharerewardpool.sol#728)
		- (success,returndata) = target.call{value: value}(data) (contracts/
sharerewardpool.sol#186)
	Event emitted after the call(s):
	- Withdraw(_sender,_pid,_amount) (contracts/sharerewardpool.sol#736)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3

Reentrancy in SssGenesisRewardPool.deposit(uint256,uint256) (contracts/
sssgenesisrewardpool.sol#709-727):
	External calls:
	- safeSssTransfer(_sender,_pending) (contracts/sssgenesisrewardpool.sol#717)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/sssgenesisrewardpool.sol#507)
		- sss.safeTransfer(_to,_sssBalance) (contracts/sssgenesisrewardpool.sol#765)
		- sss.safeTransfer(_to,_amount) (contracts/sssgenesisrewardpool.sol#767)
		- (success,returndata) = target.call{value: value}(data) (contracts/
sssgenesisrewardpool.sol#384)
	External calls sending eth:
	- safeSssTransfer(_sender,_pending) (contracts/sssgenesisrewardpool.sol#717)
		- (success,returndata) = target.call{value: value}(data) (contracts/
sssgenesisrewardpool.sol#384)
	Event emitted after the call(s):
	- RewardPaid(_sender,_pending) (contracts/sssgenesisrewardpool.sol#718)
Reentrancy in SssGenesisRewardPool.deposit(uint256,uint256) (contracts/
sssgenesisrewardpool.sol#709-727):
	External calls:
	- safeSssTransfer(_sender,_pending) (contracts/sssgenesisrewardpool.sol#717)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/sssgenesisrewardpool.sol#507)
		- sss.safeTransfer(_to,_sssBalance) (contracts/sssgenesisrewardpool.sol#765)
```

```
⊠⊠- sss.safeTransfer(_to,_amount) (contracts/sssgenesisrewardpool.sol#767)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/
sssgenesisrewardpool.sol#384)
⊠- pool.token.safeTransferFrom(_sender,address(this),_amount) (contracts/
sssgenesisrewardpool.sol#722)
⊠External calls sending eth:
⊠- safeSssTransfer(_sender,_pending) (contracts/sssgenesisrewardpool.sol#717)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/
sssgenesisrewardpool.sol#384)
⊠Event emitted after the call(s):
⊠- Deposit(_sender,_pid,_amount) (contracts/sssgenesisrewardpool.sol#726)
Reentrancy in SssGenesisRewardPool.emergencyWithdraw(uint256) (contracts/
sssgenesisrewardpool.sol#750-758):
⊠External calls:
⊠- pool.token.safeTransfer(msg.sender,_amount) (contracts/sssgenesisrewardpool.sol#756)
⊠Event emitted after the call(s):
⊠- EmergencyWithdraw(msg.sender,_pid,_amount) (contracts/sssgenesisrewardpool.sol#757)
Reentrancy in SssGenesisRewardPool.withdraw(uint256,uint256) (contracts/
sssgenesisrewardpool.sol#730-747):
⊠External calls:
⊠- safeSssTransfer(_sender,_pending) (contracts/sssgenesisrewardpool.sol#738)
⊠⊠- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/sssgenesisrewardpool.sol#507)
⊠⊠- sss.safeTransfer(_to,_sssBalance) (contracts/sssgenesisrewardpool.sol#765)
⊠⊠- sss.safeTransfer(_to,_amount) (contracts/sssgenesisrewardpool.sol#767)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/
sssgenesisrewardpool.sol#384)
⊠External calls sending eth:
⊠- safeSssTransfer(_sender,_pending) (contracts/sssgenesisrewardpool.sol#738)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/
sssgenesisrewardpool.sol#384)
⊠Event emitted after the call(s):
⊠- RewardPaid(_sender,_pending) (contracts/sssgenesisrewardpool.sol#739)
Reentrancy in SssGenesisRewardPool.withdraw(uint256,uint256) (contracts/
sssgenesisrewardpool.sol#730-747):
⊠External calls:
⊠- safeSssTransfer(_sender,_pending) (contracts/sssgenesisrewardpool.sol#738)
⊠⊠- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/sssgenesisrewardpool.sol#507)
⊠⊠- sss.safeTransfer(_to,_sssBalance) (contracts/sssgenesisrewardpool.sol#765)
⊠⊠- sss.safeTransfer(_to,_amount) (contracts/sssgenesisrewardpool.sol#767)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/
sssgenesisrewardpool.sol#384)
⊠- pool.token.safeTransfer(_sender,_amount) (contracts/sssgenesisrewardpool.sol#743)
⊠External calls sending eth:
⊠- safeSssTransfer(_sender,_pending) (contracts/sssgenesisrewardpool.sol#738)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/
sssgenesisrewardpool.sol#384)
⊠Event emitted after the call(s):
⊠- Withdraw(_sender,_pid,_amount) (contracts/sssgenesisrewardpool.sol#746)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3

Reentrancy in Treasury._sendToMasonry(uint256) (contracts/treasury.sol#1207-1230):
⊠External calls:
⊠- IBasisAsset(sss).mint(address(this),_amount) (contracts/treasury.sol#1208)
⊠- IERC20(sss).transfer(daoFund,_daoFundSharedAmount) (contracts/treasury.sol#1213)
⊠Event emitted after the call(s):
⊠- DaoFundFunded(now,_daoFundSharedAmount) (contracts/treasury.sol#1214)
Reentrancy in Treasury._sendToMasonry(uint256) (contracts/treasury.sol#1207-1230):
⊠External calls:
⊠- IBasisAsset(sss).mint(address(this),_amount) (contracts/treasury.sol#1208)
⊠- IERC20(sss).transfer(daoFund,_daoFundSharedAmount) (contracts/treasury.sol#1213)
⊠- IERC20(sss).transfer(devFund,_devFundSharedAmount) (contracts/treasury.sol#1220)
⊠Event emitted after the call(s):
⊠- DevFundFunded(now,_devFundSharedAmount) (contracts/treasury.sol#1221)
Reentrancy in Treasury._sendToMasonry(uint256) (contracts/treasury.sol#1207-1230):
⊠External calls:
⊠- IBasisAsset(sss).mint(address(this),_amount) (contracts/treasury.sol#1208)
⊠- IERC20(sss).transfer(daoFund,_daoFundSharedAmount) (contracts/treasury.sol#1213)
⊠- IERC20(sss).transfer(devFund,_devFundSharedAmount) (contracts/treasury.sol#1220)
⊠- IERC20(sss).safeApprove(masonry,0) (contracts/treasury.sol#1226)
⊠- IERC20(sss).safeApprove(masonry,_amount) (contracts/treasury.sol#1227)
⊠- IMasonry(masonry).allocateSeigniorage(_amount) (contracts/treasury.sol#1228)
⊠Event emitted after the call(s):
⊠- MasonryFunded(now,_amount) (contracts/treasury.sol#1229)
Reentrancy in Treasury.allocateSeigniorage() (contracts/treasury.sol#1242-1282):
⊠External calls:
⊠- _updateSssPrice() (contracts/treasury.sol#1243)
⊠⊠- IOracle(sssOracle).update() (contracts/treasury.sol#1139)
⊠- _sendToMasonry(sssSupply.mul(bootstrapSupplyExpansionPercent).div(10000)) (contracts/
treasury.sol#1248)
```

```
		- IBasisAsset(sss).mint(address(this),_amount) (contracts/treasury.sol#1208)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/treasury.sol#530)
		- (success,returndata) = target.call{value: value}(data) (contracts/treasury.sol#407)
		- IERC20(sss).transfer(daoFund,_daoFundSharedAmount) (contracts/treasury.sol#1213)
		- IERC20(sss).transfer(devFund,_devFundSharedAmount) (contracts/treasury.sol#1220)
		- IERC20(sss).safeApprove(masonry,0) (contracts/treasury.sol#1226)
		- IERC20(sss).safeApprove(masonry,_amount) (contracts/treasury.sol#1227)
		- IMasonry(masonry).allocateSeigniorage(_amount) (contracts/treasury.sol#1228)
	External calls sending eth:
	- _sendToMasonry(sssSupply.mul(bootstrapSupplyExpansionPercent).div(10000)) (contracts/
treasury.sol#1248)
		- (success,returndata) = target.call{value: value}(data) (contracts/treasury.sol#407)
	Event emitted after the call(s):
	- DaoFundFunded(now,_daoFundSharedAmount) (contracts/treasury.sol#1214)
		- _sendToMasonry(sssSupply.mul(bootstrapSupplyExpansionPercent).div(10000))
(contracts/treasury.sol#1248)
	- DevFundFunded(now,_devFundSharedAmount) (contracts/treasury.sol#1221)
		- _sendToMasonry(sssSupply.mul(bootstrapSupplyExpansionPercent).div(10000))
(contracts/treasury.sol#1248)
	- MasonryFunded(now,_amount) (contracts/treasury.sol#1229)
		- _sendToMasonry(sssSupply.mul(bootstrapSupplyExpansionPercent).div(10000))
(contracts/treasury.sol#1248)
Reentrancy in Treasury.allocateSeigniorage() (contracts/treasury.sol#1242-1282):
	External calls:
	- _updateSssPrice() (contracts/treasury.sol#1243)
		- IOracle(sssOracle).update() (contracts/treasury.sol#1139)
	- _sendToMasonry(_savedForMasonry) (contracts/treasury.sol#1273)
		- IBasisAsset(sss).mint(address(this),_amount) (contracts/treasury.sol#1208)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/treasury.sol#530)
		- (success,returndata) = target.call{value: value}(data) (contracts/treasury.sol#407)
		- IERC20(sss).transfer(daoFund,_daoFundSharedAmount) (contracts/treasury.sol#1213)
		- IERC20(sss).transfer(devFund,_devFundSharedAmount) (contracts/treasury.sol#1220)
		- IERC20(sss).safeApprove(masonry,0) (contracts/treasury.sol#1226)
		- IERC20(sss).safeApprove(masonry,_amount) (contracts/treasury.sol#1227)
		- IMasonry(masonry).allocateSeigniorage(_amount) (contracts/treasury.sol#1228)
	External calls sending eth:
	- _sendToMasonry(_savedForMasonry) (contracts/treasury.sol#1273)
		- (success,returndata) = target.call{value: value}(data) (contracts/treasury.sol#407)
	Event emitted after the call(s):
```

```
⊠- DaoFundFunded(now,_daoFundSharedAmount) (contracts/treasury.sol#1214)
⊠⊠- _sendToMasonry(_savedForMasonry) (contracts/treasury.sol#1273)
⊠- DevFundFunded(now,_devFundSharedAmount) (contracts/treasury.sol#1221)
⊠⊠- _sendToMasonry(_savedForMasonry) (contracts/treasury.sol#1273)
⊠- MasonryFunded(now,_amount) (contracts/treasury.sol#1229)
⊠⊠- _sendToMasonry(_savedForMasonry) (contracts/treasury.sol#1273)
Reentrancy in Treasury.allocateSeigniorage() (contracts/treasury.sol#1242-1282):
⊠External calls:
⊠- _updateSssPrice() (contracts/treasury.sol#1243)
⊠⊠- IOracle(sssOracle).update() (contracts/treasury.sol#1139)
⊠- _sendToMasonry(_savedForMasonry) (contracts/treasury.sol#1273)
⊠⊠- IBasisAsset(sss).mint(address(this),_amount) (contracts/treasury.sol#1208)
⊠⊠- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/treasury.sol#530)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/treasury.sol#407)
⊠⊠- IERC20(sss).transfer(daoFund,_daoFundSharedAmount) (contracts/treasury.sol#1213)
⊠⊠- IERC20(sss).transfer(devFund,_devFundSharedAmount) (contracts/treasury.sol#1220)
⊠⊠- IERC20(sss).safeApprove(masonry,0) (contracts/treasury.sol#1226)
⊠⊠- IERC20(sss).safeApprove(masonry,_amount) (contracts/treasury.sol#1227)
⊠⊠- IMasonry(masonry).allocateSeigniorage(_amount) (contracts/treasury.sol#1228)
⊠- IBasisAsset(sss).mint(address(this),_savedForBond) (contracts/treasury.sol#1277)
⊠External calls sending eth:
⊠- _sendToMasonry(_savedForMasonry) (contracts/treasury.sol#1273)
⊠⊠- (success,returndata) = target.call{value: value}(data) (contracts/treasury.sol#407)
⊠Event emitted after the call(s):
⊠- TreasuryFunded(now,_savedForBond) (contracts/treasury.sol#1278)
Reentrancy in Treasury.buyBonds(uint256,uint256) (contracts/treasury.sol#1152-1179):
⊠External calls:
⊠- IBasisAsset(sss).burnFrom(msg.sender,_sssAmount) (contracts/treasury.sol#1172)
⊠- IBasisAsset(sbond).mint(msg.sender,_bondAmount) (contracts/treasury.sol#1173)
⊠- _updateSssPrice() (contracts/treasury.sol#1176)
⊠⊠- IOracle(sssOracle).update() (contracts/treasury.sol#1139)
⊠Event emitted after the call(s):
⊠- BoughtBonds(msg.sender,_sssAmount,_bondAmount) (contracts/treasury.sol#1178)
Reentrancy in Treasury.redeemBonds(uint256,uint256) (contracts/treasury.sol#1181-1205):
⊠External calls:
⊠- IBasisAsset(sbond).burnFrom(msg.sender,_bondAmount) (contracts/treasury.sol#1199)
⊠- IERC20(sss).safeTransfer(msg.sender,_sssAmount) (contracts/treasury.sol#1200)
⊠- _updateSssPrice() (contracts/treasury.sol#1202)
⊠⊠- IOracle(sssOracle).update() (contracts/treasury.sol#1139)
⊠Event emitted after the call(s):
```

```
⊠- RedeemedBonds(msg.sender,_sssAmount,_bondAmount) (contracts/treasury.sol#1204)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3

UniswapV2OracleLibrary.currentCumulativePrices(address) (contracts/oracle.sol#391-415)
uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- blockTimestampLast != blockTimestamp (contracts/oracle.sol#406)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp

SShare.unclaimedTreasuryFund() (contracts/sshare.sol#728-733) uses timestamp for
comparisons
⊠Dangerous comparisons:
⊠- _now > endTime (contracts/sshare.sol#730)
⊠- communityFundLastClaimed >= _now (contracts/sshare.sol#731)
SShare.unclaimedDevFund() (contracts/sshare.sol#735-740) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- _now > endTime (contracts/sshare.sol#737)
⊠- devFundLastClaimed >= _now (contracts/sshare.sol#738)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp

SShareRewardPool.constructor(address,uint256) (contracts/sharerewardpool.sol#563-572)
uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- require(bool,string)(block.timestamp < _poolStartTime,late) (contracts/
sharerewardpool.sol#567)
SShareRewardPool.checkPoolDuplicate(IERC20) (contracts/sharerewardpool.sol#579-584)
uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- pid < length (contracts/sharerewardpool.sol#581)
⊠- require(bool,string)(poolInfo[pid].token != _token,SShareRewardPool: existing pool?)
(contracts/sharerewardpool.sol#582)
SShareRewardPool.add(uint256,IERC20,bool,uint256) (contracts/
sharerewardpool.sol#587-625) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- block.timestamp < poolStartTime (contracts/sharerewardpool.sol#597)
⊠- _lastRewardTime == 0 (contracts/sharerewardpool.sol#599)
⊠- _lastRewardTime < poolStartTime (contracts/sharerewardpool.sol#602)
⊠- _lastRewardTime == 0 || _lastRewardTime < block.timestamp (contracts/
sharerewardpool.sol#608)
```

```
☒- _isStarted = (_lastRewardTime <= poolStartTime) || (_lastRewardTime <=
block.timestamp) (contracts/ssharerewardpool.sol#612-614)
SShareRewardPool.getGeneratedReward(uint256,uint256) (contracts/
ssharerewardpool.sol#640-651) uses timestamp for comparisons
☒Dangerous comparisons:
☒- _fromTime >= _toTime (contracts/ssharerewardpool.sol#641)
☒- _toTime >= poolEndTime (contracts/ssharerewardpool.sol#642)
☒- _toTime <= poolStartTime (contracts/ssharerewardpool.sol#647)
SShareRewardPool.pendingShare(uint256,address) (contracts/ssharerewardpool.sol#654-665)
uses timestamp for comparisons
☒Dangerous comparisons:
☒- block.timestamp > pool.lastRewardTime && tokenSupply != 0 (contracts/
ssharerewardpool.sol#659)
SShareRewardPool.massUpdatePools() (contracts/ssharerewardpool.sol#668-673) uses
timestamp for comparisons
☒Dangerous comparisons:
☒- pid < length (contracts/ssharerewardpool.sol#670)
SShareRewardPool.updatePool(uint256) (contracts/ssharerewardpool.sol#676-696) uses
timestamp for comparisons
☒Dangerous comparisons:
☒- block.timestamp <= pool.lastRewardTime (contracts/ssharerewardpool.sol#678)
SShareRewardPool.governanceRecoverUnsupported(IERC20,uint256,address) (contracts/
ssharerewardpool.sol#766-777) uses timestamp for comparisons
☒Dangerous comparisons:
☒- block.timestamp < poolEndTime + 7776000 (contracts/ssharerewardpool.sol#767)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp

SssGenesisRewardPool.constructor(address,uint256) (contracts/
sssgenesisrewardpool.sol#572-582) uses timestamp for comparisons
☒Dangerous comparisons:
☒- require(bool,string)(block.timestamp < _poolStartTime,late) (contracts/
sssgenesisrewardpool.sol#576)
SssGenesisRewardPool.checkPoolDuplicate(IERC20) (contracts/
sssgenesisrewardpool.sol#589-594) uses timestamp for comparisons
☒Dangerous comparisons:
☒- pid < length (contracts/sssgenesisrewardpool.sol#591)
☒- require(bool,string)(poolInfo[pid].token != _token,SssGenesisPool: existing pool?)
(contracts/sssgenesisrewardpool.sol#592)
SssGenesisRewardPool.add(uint256,IERC20,bool,uint256) (contracts/
sssgenesisrewardpool.sol#597-635) uses timestamp for comparisons
```

```
⊠Dangerous comparisons:
⊠- block.timestamp < poolStartTime (contracts/sssgenesisrewardpool.sol#607)
⊠- _lastRewardTime == 0 (contracts/sssgenesisrewardpool.sol#609)
⊠- _lastRewardTime < poolStartTime (contracts/sssgenesisrewardpool.sol#612)
⊠- _lastRewardTime == 0 || _lastRewardTime < block.timestamp (contracts/
sssgenesisrewardpool.sol#618)
⊠- _isStarted = (_lastRewardTime <= poolStartTime) || (_lastRewardTime <=
block.timestamp) (contracts/sssgenesisrewardpool.sol#622-624)
SssGenesisRewardPool.getGeneratedReward(uint256,uint256) (contracts/
sssgenesisrewardpool.sol#650-661) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- _fromTime >= _toTime (contracts/sssgenesisrewardpool.sol#651)
⊠- _toTime >= poolEndTime (contracts/sssgenesisrewardpool.sol#652)
⊠- _toTime <= poolStartTime (contracts/sssgenesisrewardpool.sol#657)
SssGenesisRewardPool.pendingSSS(uint256,address) (contracts/
sssgenesisrewardpool.sol#664-675) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- block.timestamp > pool.lastRewardTime && tokenSupply != 0 (contracts/
sssgenesisrewardpool.sol#669)
SssGenesisRewardPool.massUpdatePools() (contracts/sssgenesisrewardpool.sol#678-683)
uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- pid < length (contracts/sssgenesisrewardpool.sol#680)
SssGenesisRewardPool.updatePool(uint256) (contracts/sssgenesisrewardpool.sol#686-706)
uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- block.timestamp <= pool.lastRewardTime (contracts/sssgenesisrewardpool.sol#688)
SssGenesisRewardPool.governanceRecoverUnsupported(IERC20,uint256,address) (contracts/
sssgenesisrewardpool.sol#776-787) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- block.timestamp < poolEndTime + 7776000 (contracts/sssgenesisrewardpool.sol#777)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp

TaxOfficeV2.addLiquidityTaxFree(address,uint256,uint256,uint256,uint256) (contracts/
taxoffice.sol#604-649) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- amtSss.sub(resultAmtSss) > 0 (contracts/taxoffice.sol#642)
⊠- amtToken.sub(resultAmtToken) > 0 (contracts/taxoffice.sol#645)
TaxOfficeV2.addLiquidityETHTaxFree(uint256,uint256,uint256) (contracts/
taxoffice.sol#651-688) uses timestamp for comparisons
```

```
⊠Dangerous comparisons:
⊠- amtSss.sub(resultAmtSss) > 0 (contracts/taxoffice.sol#684)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp

Address.isContract(address) (contracts/bonus.sol#289-298) uses assembly
⊠- INLINE ASM (contracts/bonus.sol#296)
Address._verifyCallResult(bool,bytes,string) (contracts/bonus.sol#434-451) uses
assembly
⊠- INLINE ASM (contracts/bonus.sol#443-446)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Address.isContract(address) (contracts/ssharerewardpool.sol#93-102) uses assembly
⊠- INLINE ASM (contracts/ssharerewardpool.sol#100)
Address._verifyCallResult(bool,bytes,string) (contracts/ssharerewardpool.sol#238-255)
uses assembly
⊠- INLINE ASM (contracts/ssharerewardpool.sol#247-250)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Address.isContract(address) (contracts/sssgenesisrewardpool.sol#291-300) uses assembly
⊠- INLINE ASM (contracts/sssgenesisrewardpool.sol#298)
Address._verifyCallResult(bool,bytes,string) (contracts/
sssgenesisrewardpool.sol#436-453) uses assembly
⊠- INLINE ASM (contracts/sssgenesisrewardpool.sol#445-448)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Address.isContract(address) (contracts/treasury.sol#314-323) uses assembly
⊠- INLINE ASM (contracts/treasury.sol#321)
Address._verifyCallResult(bool,bytes,string) (contracts/treasury.sol#459-476) uses
assembly
⊠- INLINE ASM (contracts/treasury.sol#468-471)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

SShareRewardPool.updatePool(uint256) (contracts/ssharerewardpool.sol#676-696) has
costly operations inside a loop:
⊠- totalAllocPoint = totalAllocPoint.add(pool.allocPoint) (contracts/
ssharerewardpool.sol#688)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-
operations-inside-a-loop

SssGenesisRewardPool.updatePool(uint256) (contracts/sssgenesisrewardpool.sol#686-706)
```

```
has costly operations inside a loop:
- totalAllocPoint = totalAllocPoint.add(pool.allocPoint) (contracts/
sssgenesisrewardpool.sol#698)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-
operations-inside-a-loop

Address.functionCall(address,bytes) (contracts/bonus.sol#342-344) is never used and
should be removed
Address.functionCallWithValue(address,bytes,uint256) (contracts/bonus.sol#367-369) is
never used and should be removed
Address.functionDelegateCall(address,bytes) (contracts/bonus.sol#416-418) is never used
and should be removed
Address.functionDelegateCall(address,bytes,string) (contracts/bonus.sol#426-432) is
never used and should be removed
Address.functionStaticCall(address,bytes) (contracts/bonus.sol#392-394) is never used
and should be removed
Address.functionStaticCall(address,bytes,string) (contracts/bonus.sol#402-408) is never
used and should be removed
Address.sendValue(address,uint256) (contracts/bonus.sol#316-322) is never used and
should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (contracts/bonus.sol#473-482) is never
used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (contracts/bonus.sol#489-492)
is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (contracts/bonus.sol#484-487)
is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/bonus.sol#176-179) is never used and
should be removed
SafeMath.mod(uint256,uint256) (contracts/bonus.sol#138-141) is never used and should be
removed
SafeMath.mod(uint256,uint256,string) (contracts/bonus.sol#196-199) is never used and
should be removed
SafeMath.sub(uint256,uint256,string) (contracts/bonus.sol#156-159) is never used and
should be removed
SafeMath.tryAdd(uint256,uint256) (contracts/bonus.sol#10-14) is never used and should
be removed
SafeMath.tryDiv(uint256,uint256) (contracts/bonus.sol#46-49) is never used and should
be removed
SafeMath.tryMod(uint256,uint256) (contracts/bonus.sol#56-59) is never used and should
be removed
SafeMath.tryMul(uint256,uint256) (contracts/bonus.sol#31-39) is never used and should
```

```
be removed
SafeMath.trySub(uint256,uint256) (contracts/bonus.sol#21-24) is never used and should
be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Babylonian.sqrt(uint256) (contracts/oracle.sol#299-311) is never used and should be
removed
Context._msgData() (contracts/oracle.sol#208-211) is never used and should be removed
FixedPoint.decode(FixedPoint.uq112x112) (contracts/oracle.sol#362-364) is never used
and should be removed
FixedPoint.div(FixedPoint.uq112x112,uint112) (contracts/oracle.sol#341-344) is never
used and should be removed
FixedPoint.encode(uint112) (contracts/oracle.sol#331-333) is never used and should be
removed
FixedPoint.encode144(uint144) (contracts/oracle.sol#336-338) is never used and should
be removed
FixedPoint.reciprocal(FixedPoint.uq112x112) (contracts/oracle.sol#372-375) is never
used and should be removed
FixedPoint.sqrt(FixedPoint.uq112x112) (contracts/oracle.sol#378-380) is never used and
should be removed
SafeMath.div(uint256,uint256) (contracts/oracle.sol#122-125) is never used and should
be removed
SafeMath.mul(uint256,uint256) (contracts/oracle.sol#103-108) is never used and should
be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

ERC20._setupDecimals(uint8) (contracts/sbond.sol#538-540) is never used and should be
removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Math.average(uint256,uint256) (contracts/sss.sol#106-109) is never used and should be
removed
Math.max(uint256,uint256) (contracts/sss.sol#91-93) is never used and should be removed
Math.min(uint256,uint256) (contracts/sss.sol#98-100) is never used and should be
removed
SafeMath8.add(uint8,uint8) (contracts/sss.sol#320-325) is never used and should be
removed
SafeMath8.div(uint8,uint8) (contracts/sss.sol#394-396) is never used and should be
removed
SafeMath8.div(uint8,uint8,string) (contracts/sss.sol#410-416) is never used and should
be removed
```

```
SafeMath8.mod(uint8,uint8) (contracts/sss.sol#430-432) is never used and should be
removed
SafeMath8.mod(uint8,uint8,string) (contracts/sss.sol#446-449) is never used and should
be removed
SafeMath8.mul(uint8,uint8) (contracts/sss.sol#368-380) is never used and should be
removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

SafeMath.add(uint256,uint256) (contracts/taxoffice.sol#71-75) 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#487-489) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Low level call in Address.sendValue(address,uint256) (contracts/bonus.sol#316-322):
⊠- (success) = recipient.call{value: amount}() (contracts/bonus.sol#320)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string)
(contracts/bonus.sol#377-384):
⊠- (success,returndata) = target.call{value: value}(data) (contracts/bonus.sol#382)
Low level call in Address.functionStaticCall(address,bytes,string) (contracts/
bonus.sol#402-408):
⊠- (success,returndata) = target.staticcall(data) (contracts/bonus.sol#406)
Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/
bonus.sol#426-432):
⊠- (success,returndata) = target.delegatecall(data) (contracts/bonus.sol#430)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls

Low level call in Address.sendValue(address,uint256) (contracts/
ssharerewardpool.sol#120-126):
⊠- (success) = recipient.call{value: amount}() (contracts/ssharerewardpool.sol#124)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string)
(contracts/ssharerewardpool.sol#181-188):
⊠- (success,returndata) = target.call{value: value}(data) (contracts/
ssharerewardpool.sol#186)
Low level call in Address.functionStaticCall(address,bytes,string) (contracts/
ssharerewardpool.sol#206-212):
⊠- (success,returndata) = target.staticcall(data) (contracts/ssharerewardpool.sol#210)
Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/
ssharerewardpool.sol#230-236):
```

```
⊠- (success,returndata) = target.delegatecall(data) (contracts/
sssharerewardpool.sol#234)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls

Low level call in Address.sendValue(address,uint256) (contracts/
sssgenesisrewardpool.sol#318-324):
⊠- (success) = recipient.call{value: amount}() (contracts/sssgenesisrewardpool.sol#322)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string)
(contracts/sssgenesisrewardpool.sol#379-386):
⊠- (success,returndata) = target.call{value: value}(data) (contracts/
sssgenesisrewardpool.sol#384)
Low level call in Address.functionStaticCall(address,bytes,string) (contracts/
sssgenesisrewardpool.sol#404-410):
⊠- (success,returndata) = target.staticcall(data) (contracts/
sssgenesisrewardpool.sol#408)
Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/
sssgenesisrewardpool.sol#428-434):
⊠- (success,returndata) = target.delegatecall(data) (contracts/
sssgenesisrewardpool.sol#432)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls

Low level call in Address.sendValue(address,uint256) (contracts/treasury.sol#341-347):
⊠- (success) = recipient.call{value: amount}() (contracts/treasury.sol#345)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string)
(contracts/treasury.sol#402-409):
⊠- (success,returndata) = target.call{value: value}(data) (contracts/treasury.sol#407)
Low level call in Address.functionStaticCall(address,bytes,string) (contracts/
treasury.sol#427-433):
⊠- (success,returndata) = target.staticcall(data) (contracts/treasury.sol#431)
Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/
treasury.sol#451-457):
⊠- (success,returndata) = target.delegatecall(data) (contracts/treasury.sol#455)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls

Redundant expression "this (contracts/oracle.sol#209)" inContext (contracts/
oracle.sol#203-212)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-
statements
```

```
Redundant expression "this (contracts/sbond.sol#208)" inContext (contracts/
sbond.sol#202-211)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-
statements

Redundant expression "this (contracts/sshare.sol#209)" inContext (contracts/
sshare.sol#203-212)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-
statements

Redundant expression "this (contracts/sss.sol#11)" inContext (contracts/sss.sol#5-14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-
statements

Redundant expression "this (contracts/taxoffice.sol#207)" inContext (contracts/
taxoffice.sol#201-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-
statements

Redundant expression "this (contracts/treasury.sol#601)" inContext (contracts/
treasury.sol#595-604)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-
statements

Variable UniswapV2OracleLibrary.currentCumulativePrices(address).price0Cumulative
(contracts/oracle.sol#395) is too similar to
UniswapV2OracleLibrary.currentCumulativePrices(address).price1Cumulative (contracts/
oracle.sol#396)
Variable Oracle.price0Average (contracts/oracle.sol#601) is too similar to
Oracle.price1Average (contracts/oracle.sol#602)
Variable Oracle.update().price0Cumulative (contracts/oracle.sol#626) is too similar to
Oracle.update().price1Cumulative (contracts/oracle.sol#626)
Variable Oracle.twap(address,uint256).price0Cumulative (contracts/oracle.sol#657) is
too similar to Oracle.twap(address,uint256).price1Cumulative (contracts/oracle.sol#657)
Variable Oracle.twap(address,uint256).price0Cumulative (contracts/oracle.sol#657) is
too similar to Oracle.update().price1Cumulative (contracts/oracle.sol#626)
Variable Oracle.update().price0Cumulative (contracts/oracle.sol#626) is too similar to
Oracle.twap(address,uint256).price1Cumulative (contracts/oracle.sol#657)
Variable Oracle.price0CumulativeLast (contracts/oracle.sol#599) is too similar to
Oracle.price1CumulativeLast (contracts/oracle.sol#600)
```

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

Variable IUniswapV2Router.addLiquidity(address,address,uint256,uint256,uint256,uint256,a
ddress,uint256).amountADesired (contracts/taxoffice.sol#331) is too similar to IUniswapV2
Router.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amo
untBDesired (contracts/taxoffice.sol#332)
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#1093) is too similar to
Treasury.setExtraFunds(address,uint256,address,uint256)._devFundSharedPercent
(contracts/treasury.sol#1095)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-
are-too-similar

Treasury.initialize(address,address,address,address,address,uint256) (contracts/
treasury.sol#982-1024) uses literals with too many digits:
☒- supplyTiers = (0,500000000000000000000000,1000000000000000000000000,15000000000000000
00000000,2000000000000000000000000,5000000000000000000000000,10000000000000000000000000,
20000000000000000000000000,50000000000000000000000000) (contracts/treasury.sol#1001)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-
digits

SShareRewardPool.runningTime (contracts/ssharerewardpool.sol#555) should be constant
SShareRewardPool.sSharePerSecond (contracts/ssharerewardpool.sol#554) should be
constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-constant

SssGenesisRewardPool.feewallet (contracts/sssgenesisrewardpool.sol#538) should be
constant
SssGenesisRewardPool.runningTime (contracts/sssgenesisrewardpool.sol#563) should be
constant
SssGenesisRewardPool.sssPerSecond (contracts/sssgenesisrewardpool.sol#562) should be
constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-constant

TaxOfficeV2.sss (contracts/taxoffice.sol#546) should be constant
```

```
TaxOfficeV2.uniRouter (contracts/taxoffice.sol#548) should be constant
TaxOfficeV2.wftm (contracts/taxoffice.sol#547) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-constant

initialize(IERC20,IERC20,ITreasury) should be declared external:
⊠- Masonry.initialize(IERC20,IERC20,ITreasury) (contracts/bonus.sol#665-683)
rewardPerShare() should be declared external:
⊠- Masonry.rewardPerShare() (contracts/bonus.sol#737-739)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external

renounceOwnership() should be declared external:
⊠- Ownable.renounceOwnership() (contracts/oracle.sol#249-252)
transferOwnership(address) should be declared external:
⊠- Ownable.transferOwnership(address) (contracts/oracle.sol#258-262)
isOperator() should be declared external:
⊠- Operator.isOperator() (contracts/oracle.sol#284-286)
transferOperator(address) should be declared external:
⊠- Operator.transferOperator(address) (contracts/oracle.sol#288-290)
getCurrentEpoch() should be declared external:
⊠- Epoch.getCurrentEpoch() (contracts/oracle.sol#465-467)
getPeriod() should be declared external:
⊠- Epoch.getPeriod() (contracts/oracle.sol#469-471)
getStartTime() should be declared external:
⊠- Epoch.getStartTime() (contracts/oracle.sol#473-475)
getLastEpochTime() should be declared external:
⊠- Epoch.getLastEpochTime() (contracts/oracle.sol#477-479)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external

name() should be declared external:
⊠- ERC20.name() (contracts/sbond.sol#315-317)
symbol() should be declared external:
⊠- ERC20.symbol() (contracts/sbond.sol#323-325)
decimals() should be declared external:
⊠- ERC20.decimals() (contracts/sbond.sol#340-342)
totalSupply() should be declared external:
⊠- ERC20.totalSupply() (contracts/sbond.sol#347-349)
transfer(address,uint256) should be declared external:
⊠- ERC20.transfer(address,uint256) (contracts/sbond.sol#366-369)
```

```
approve(address,uint256) should be declared external:
⊠- ERC20.approve(address,uint256) (contracts/sbond.sol#385-388)
transferFrom(address,address,uint256) should be declared external:
⊠- ERC20.transferFrom(address,address,uint256) (contracts/sbond.sol#403-407)
increaseAllowance(address,uint256) should be declared external:
⊠- ERC20.increaseAllowance(address,uint256) (contracts/sbond.sol#421-424)
decreaseAllowance(address,uint256) should be declared external:
⊠- ERC20.decreaseAllowance(address,uint256) (contracts/sbond.sol#440-443)
operator() should be declared external:
⊠- Operator.operator() (contracts/sbond.sol#651-653)
mint(address,uint256) should be declared external:
⊠- SBond.mint(address,uint256) (contracts/sbond.sol#687-693)
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/sshare.sol#355-357)
burnFrom(address,uint256) should be declared external:
⊠- ERC20Burnable.burnFrom(address,uint256) (contracts/sshare.sol#584-589)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external

set(uint256,uint256) should be declared external:
⊠- SShareRewardPool.set(uint256,uint256) (contracts/ssharerewardpool.sol#628-637)
deposit(uint256,uint256) should be declared external:
⊠- SShareRewardPool.deposit(uint256,uint256) (contracts/ssharerewardpool.sol#699-717)
withdraw(uint256,uint256) should be declared external:
⊠- SShareRewardPool.withdraw(uint256,uint256) (contracts/ssharerewardpool.sol#720-737)
emergencyWithdraw(uint256) should be declared external:
⊠- SShareRewardPool.emergencyWithdraw(uint256) (contracts/ssharerewardpool.sol#740-748)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external

transferFrom(address,address,uint256) should be declared external:
⊠- ERC20.transferFrom(address,address,uint256) (contracts/sss.sol#570-574)
⊠- Sss.transferFrom(address,address,uint256) (contracts/sss.sol#1037-1062)
isAddressExcluded(address) should be declared external:
⊠- Sss.isAddressExcluded(address) (contracts/sss.sol#925-927)
setTaxTiersTwap(uint8,uint256) should be declared external:
⊠- Sss.setTaxTiersTwap(uint8,uint256) (contracts/sss.sol#929-940)
setTaxTiersRate(uint8,uint256) should be declared external:
```

```
⊠- Sss.setTaxTiersRate(uint8,uint256) (contracts/sss.sol#942-947)
setBurnThreshold(uint256) should be declared external:
⊠- Sss.setBurnThreshold(uint256) (contracts/sss.sol#949-951)
enableAutoCalculateTax() should be declared external:
⊠- Sss.enableAutoCalculateTax() (contracts/sss.sol#973-975)
disableAutoCalculateTax() should be declared external:
⊠- Sss.disableAutoCalculateTax() (contracts/sss.sol#977-979)
setSssOracle(address) should be declared external:
⊠- Sss.setSssOracle(address) (contracts/sss.sol#981-984)
setTaxOffice(address) should be declared external:
⊠- Sss.setTaxOffice(address) (contracts/sss.sol#986-990)
setTaxCollectorAddress(address) should be declared external:
⊠- Sss.setTaxCollectorAddress(address) (contracts/sss.sol#992-995)
setTaxRate(uint256) should be declared external:
⊠- Sss.setTaxRate(uint256) (contracts/sss.sol#997-1001)
includeAddress(address) should be declared external:
⊠- Sss.includeAddress(address) (contracts/sss.sol#1009-1013)
mint(address,uint256) should be declared external:
⊠- Sss.mint(address,uint256) (contracts/sss.sol#1021-1027)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external

set(uint256,uint256) should be declared external:
⊠- SssGenesisRewardPool.set(uint256,uint256) (contracts/
sssgenesisrewardpool.sol#638-647)
deposit(uint256,uint256) should be declared external:
⊠- SssGenesisRewardPool.deposit(uint256,uint256) (contracts/
sssgenesisrewardpool.sol#709-727)
withdraw(uint256,uint256) should be declared external:
⊠- SssGenesisRewardPool.withdraw(uint256,uint256) (contracts/
sssgenesisrewardpool.sol#730-747)
emergencyWithdraw(uint256) should be declared external:
⊠- SssGenesisRewardPool.emergencyWithdraw(uint256) (contracts/
sssgenesisrewardpool.sol#750-758)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external

setTaxTiersTwap(uint8,uint256) should be declared external:
⊠- TaxOfficeV2.setTaxTiersTwap(uint8,uint256) (contracts/taxoffice.sol#552-554)
setTaxTiersRate(uint8,uint256) should be declared external:
⊠- TaxOfficeV2.setTaxTiersRate(uint8,uint256) (contracts/taxoffice.sol#556-558)
```

```
enableAutoCalculateTax() should be declared external:
⊠- TaxOfficeV2.enableAutoCalculateTax() (contracts/taxoffice.sol#560-562)
disableAutoCalculateTax() should be declared external:
⊠- TaxOfficeV2.disableAutoCalculateTax() (contracts/taxoffice.sol#564-566)
setTaxRate(uint256) should be declared external:
⊠- TaxOfficeV2.setTaxRate(uint256) (contracts/taxoffice.sol#568-570)
setBurnThreshold(uint256) should be declared external:
⊠- TaxOfficeV2.setBurnThreshold(uint256) (contracts/taxoffice.sol#572-574)
setTaxCollectorAddress(address) should be declared external:
⊠- TaxOfficeV2.setTaxCollectorAddress(address) (contracts/taxoffice.sol#576-578)
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#890-892)
getSssUpdatedPrice() should be declared external:
⊠- Treasury.getSssUpdatedPrice() (contracts/treasury.sol#908-914)
getReserve() should be declared external:
⊠- Treasury.getReserve() (contracts/treasury.sol#917-919)
getBurnableSssLeft() should be declared external:
⊠- Treasury.getBurnableSssLeft() (contracts/treasury.sol#921-933)
getRedeemableBonds() should be declared external:
⊠- Treasury.getRedeemableBonds() (contracts/treasury.sol#935-944)
initialize(address,address,address,address,address,uint256) should be declared
external:
⊠- Treasury.initialize(address,address,address,address,address,uint256) (contracts/
treasury.sol#982-1024)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external
. analyzed (79 contracts with 77 detectors), 424 result(s) found
```

0x Guard