0x Guard

Smart contracts security assessment

Final report

Tariff: Standard

Unite Finance

December 2021

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for the Unite Finance team. The project website is https://unitefinance.io.The audited project is a fork of the Tomb Finance Project. The purpose of the 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.

Name	Unite Finance
Audit date	2021-12-29 - 2021-12-29
Language	Solidity
Platform	Harmony

Contracts checked

Name	Address
TaxOracle.sol	
Unite.sol	
UShareRewardPool.sol	
UBond.sol	
UShare.sol	
Oracle.sol	
Treasury.sol	
Boardroom.sol	

Procedure

We perform our audit according to the following procedure:

Automated analysis

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

Manual audit

- Comparing the project to the Tomb Finance implementation

Classification of issue severity

High severity	High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.
Medium severity	Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.
Low severity	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

No issues were found

Medium severity issues

No issues were found

Low severity issues

No issues were found

Conclusion

The Unite Finance Project was compared with the Tomb Project. Unite Finance has changed the implementation of Token, Treasury and UShare contracts. The changed Token contract is not affected by the vulnerability that was discovered in the Tomb before because it doesn't contain the implementation of transfer with taxes.

In contracts Treasury and UShare were added team1Fund addresses which receive funds as well as devFund it the Tomb Finance.

Contract UShare sets state variables communityFundRewardRate, team1FundRewardRate and devFundRewardRate by calling external function setAllocations.

No serious issues were found in the audited changes.

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.

Static code analysis results

```
INFO:Detectors:
UniswapV2OracleLibrary.currentBlockTimestamp() (contracts/lib/
UniswapV2OracleLibrary.sol#13-15) uses a weak PRNG: "uint32(block.timestamp % 2 ** 32)
(contracts/lib/UniswapV2OracleLibrary.sol#14)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG
INFO:Detectors:
IERC20 is re-used:
	- contracts/interfaces/IERC20.sol#8-77
	- node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#8-77
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#name-reused
INFO:Detectors:
Reentrancy in Treasury.allocateSeigniorage() (contracts/Treasury.sol#501-541):
	External calls:
	- _updateUnitePrice() (contracts/Treasury.sol#502)
		- IOracle(kittyOracle).update() (contracts/Treasury.sol#394)
	- _sendToBoardroom(_savedForBoardroom) (contracts/Treasury.sol#532)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- IBasisAsset(kitty).mint(address(this),_amount) (contracts/Treasury.sol#460)
		- IERC20(kitty).transfer(daoFund,_daoFundSharedAmount) (contracts/Treasury.sol#465)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
		- IERC20(kitty).transfer(devFund,_devFundSharedAmount) (contracts/Treasury.sol#472)
		- IERC20(kitty).transfer(team1Fund,_team1FundSharedAmount) (contracts/
Treasury.sol#479)
		- IERC20(kitty).safeApprove(boardroom,0) (contracts/Treasury.sol#485)
		- IERC20(kitty).safeApprove(boardroom,_amount) (contracts/Treasury.sol#486)
		- IBoardroom(boardroom).allocateSeigniorage(_amount) (contracts/Treasury.sol#487)
	External calls sending eth:
	- _sendToBoardroom(_savedForBoardroom) (contracts/Treasury.sol#532)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	State variables written after the call(s):
	- seigniorageSaved = seigniorageSaved.add(_savedForBond) (contracts/Treasury.sol#535)
Reentrancy in UShareRewardPool.deposit(uint256,uint256) (contracts/distribution/
UShareRewardPool.sol#197-215):
	External calls:
```

```
	- safeUShareTransfer(_sender,_pending) (contracts/distribution/
UShareRewardPool.sol#205)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- bshare.safeTransfer(_to,_bshareBal) (contracts/distribution/
UShareRewardPool.sol#253)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
		- bshare.safeTransfer(_to,_amount) (contracts/distribution/UShareRewardPool.sol#255)
	- pool.token.safeTransferFrom(_sender,address(this),_amount) (contracts/distribution/
UShareRewardPool.sol#210)
	External calls sending eth:
	- safeUShareTransfer(_sender,_pending) (contracts/distribution/
UShareRewardPool.sol#205)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	State variables written after the call(s):
	- user.amount = user.amount.add(_amount) (contracts/distribution/
UShareRewardPool.sol#211)
	- user.rewardDebt = user.amount.mul(pool.accUSharePerShare).div(1e18) (contracts/
distribution/UShareRewardPool.sol#213)
Reentrancy in UniteGenesisRewardPool.deposit(uint256,uint256) (contracts/distribution/
UniteGenesisRewardPool.sol#196-218):
	External calls:
	- safeUniteTransfer(_sender,_pending) (contracts/distribution/
UniteGenesisRewardPool.sol#204)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- bomb.safeTransfer(_to,_bombBalance) (contracts/distribution/
UniteGenesisRewardPool.sol#256)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
		- bomb.safeTransfer(_to,_amount) (contracts/distribution/
UniteGenesisRewardPool.sol#258)
	- pool.token.safeTransferFrom(_sender,address(this),_amount) (contracts/distribution/
UniteGenesisRewardPool.sol#209)
	External calls sending eth:
	- safeUniteTransfer(_sender,_pending) (contracts/distribution/
UniteGenesisRewardPool.sol#204)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
```

```
⊠State variables written after the call(s):
⊠- user.amount = user.amount.add(_amount.mul(9900).div(10000)) (contracts/distribution/
UniteGenesisRewardPool.sol#211)
⊠- user.amount = user.amount.add(_amount) (contracts/distribution/
UniteGenesisRewardPool.sol#213)
⊠- user.rewardDebt = user.amount.mul(pool.accUnitePerShare).div(1e18) (contracts/
distribution/UniteGenesisRewardPool.sol#216)
Reentrancy in UniteRewardPool.deposit(uint256,uint256) (contracts/distribution/
UniteRewardPool.sol#201-219):
⊠External calls:
⊠- safeUniteTransfer(_sender,_pending) (contracts/distribution/UniteRewardPool.sol#209)
⊠⊠- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
⊠⊠- bomb.safeTransfer(_to,_bombBal) (contracts/distribution/UniteRewardPool.sol#257)
⊠⊠- bomb.safeTransfer(_to,_amount) (contracts/distribution/UniteRewardPool.sol#259)
⊠⊠- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
⊠- pool.token.safeTransferFrom(_sender,address(this),_amount) (contracts/distribution/
UniteRewardPool.sol#214)
⊠External calls sending eth:
⊠- safeUniteTransfer(_sender,_pending) (contracts/distribution/UniteRewardPool.sol#209)
⊠⊠- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
⊠State variables written after the call(s):
⊠- user.amount = user.amount.add(_amount) (contracts/distribution/
UniteRewardPool.sol#215)
⊠- user.rewardDebt = user.amount.mul(pool.accUnitePerShare).div(1e18) (contracts/
distribution/UniteRewardPool.sol#217)
Reentrancy in Boardroom.stake(uint256) (contracts/Boardroom.sol#203-208):
⊠External calls:
⊠- super.stake(amount) (contracts/Boardroom.sol#205)
⊠⊠- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
⊠⊠- share.safeTransferFrom(msg.sender,address(this),amount) (contracts/
Boardroom.sol#32)
⊠⊠- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
⊠External calls sending eth:
⊠- super.stake(amount) (contracts/Boardroom.sol#205)
⊠⊠- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
```

```
	State variables written after the call(s):
	- members[msg.sender].epochTimerStart = treasury.epoch() (contracts/Boardroom.sol#206)
Reentrancy in Boardroom.withdraw(uint256) (contracts/Boardroom.sol#210-216):
	External calls:
	- claimReward() (contracts/Boardroom.sol#213)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
		- kitty.safeTransfer(msg.sender,reward) (contracts/Boardroom.sol#228)
	- super.withdraw(amount) (contracts/Boardroom.sol#214)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- share.safeTransfer(msg.sender,amount) (contracts/Boardroom.sol#40)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	External calls sending eth:
	- claimReward() (contracts/Boardroom.sol#213)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	- super.withdraw(amount) (contracts/Boardroom.sol#214)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	State variables written after the call(s):
	- super.withdraw(amount) (contracts/Boardroom.sol#214)
		- _balances[msg.sender] = memberShare.sub(amount) (contracts/Boardroom.sol#39)
Reentrancy in UShareRewardPool.withdraw(uint256,uint256) (contracts/distribution/
UShareRewardPool.sol#218-235):
	External calls:
	- safeUShareTransfer(_sender,_pending) (contracts/distribution/
UShareRewardPool.sol#226)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- bshare.safeTransfer(_to,_bshareBal) (contracts/distribution/
UShareRewardPool.sol#253)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
		- bshare.safeTransfer(_to,_amount) (contracts/distribution/UShareRewardPool.sol#255)
	External calls sending eth:
	- safeUShareTransfer(_sender,_pending) (contracts/distribution/
UShareRewardPool.sol#226)
```

```
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	State variables written after the call(s):
	- user.amount = user.amount.sub(_amount) (contracts/distribution/
UShareRewardPool.sol#230)
Reentrancy in UShareRewardPool.withdraw(uint256,uint256) (contracts/distribution/
UShareRewardPool.sol#218-235):
	External calls:
	- safeUShareTransfer(_sender,_pending) (contracts/distribution/
UShareRewardPool.sol#226)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- bshare.safeTransfer(_to,_bshareBal) (contracts/distribution/
UShareRewardPool.sol#253)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
		- bshare.safeTransfer(_to,_amount) (contracts/distribution/UShareRewardPool.sol#255)
	- pool.token.safeTransfer(_sender,_amount) (contracts/distribution/
UShareRewardPool.sol#231)
	External calls sending eth:
	- safeUShareTransfer(_sender,_pending) (contracts/distribution/
UShareRewardPool.sol#226)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	State variables written after the call(s):
	- user.rewardDebt = user.amount.mul(pool.accUSharePerShare).div(1e18) (contracts/
distribution/UShareRewardPool.sol#233)
Reentrancy in UniteGenesisRewardPool.withdraw(uint256,uint256) (contracts/distribution/
UniteGenesisRewardPool.sol#221-238):
	External calls:
	- safeUniteTransfer(_sender,_pending) (contracts/distribution/
UniteGenesisRewardPool.sol#229)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- bomb.safeTransfer(_to,_bombBalance) (contracts/distribution/
UniteGenesisRewardPool.sol#256)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
		- bomb.safeTransfer(_to,_amount) (contracts/distribution/
UniteGenesisRewardPool.sol#258)
	External calls sending eth:
```

```
	- safeUniteTransfer(_sender,_pending) (contracts/distribution/
UniteGenesisRewardPool.sol#229)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	State variables written after the call(s):
	- user.amount = user.amount.sub(_amount) (contracts/distribution/
UniteGenesisRewardPool.sol#233)
Reentrancy in UniteGenesisRewardPool.withdraw(uint256,uint256) (contracts/distribution/
UniteGenesisRewardPool.sol#221-238):
	External calls:
	- safeUniteTransfer(_sender,_pending) (contracts/distribution/
UniteGenesisRewardPool.sol#229)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- bomb.safeTransfer(_to,_bombBalance) (contracts/distribution/
UniteGenesisRewardPool.sol#256)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
		- bomb.safeTransfer(_to,_amount) (contracts/distribution/
UniteGenesisRewardPool.sol#258)
	- pool.token.safeTransfer(_sender,_amount) (contracts/distribution/
UniteGenesisRewardPool.sol#234)
	External calls sending eth:
	- safeUniteTransfer(_sender,_pending) (contracts/distribution/
UniteGenesisRewardPool.sol#229)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	State variables written after the call(s):
	- user.rewardDebt = user.amount.mul(pool.accUnitePerShare).div(1e18) (contracts/
distribution/UniteGenesisRewardPool.sol#236)
Reentrancy in UniteRewardPool.withdraw(uint256,uint256) (contracts/distribution/
UniteRewardPool.sol#222-239):
	External calls:
	- safeUniteTransfer(_sender,_pending) (contracts/distribution/UniteRewardPool.sol#230)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- bomb.safeTransfer(_to,_bombBal) (contracts/distribution/UniteRewardPool.sol#257)
		- bomb.safeTransfer(_to,_amount) (contracts/distribution/UniteRewardPool.sol#259)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	External calls sending eth:
```

```
	- safeUniteTransfer(_sender,_pending) (contracts/distribution/UniteRewardPool.sol#230)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	State variables written after the call(s):
	- user.amount = user.amount.sub(_amount) (contracts/distribution/
UniteRewardPool.sol#234)
Reentrancy in UniteRewardPool.withdraw(uint256,uint256) (contracts/distribution/
UniteRewardPool.sol#222-239):
	External calls:
	- safeUniteTransfer(_sender,_pending) (contracts/distribution/UniteRewardPool.sol#230)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- bomb.safeTransfer(_to,_bombBal) (contracts/distribution/UniteRewardPool.sol#257)
		- bomb.safeTransfer(_to,_amount) (contracts/distribution/UniteRewardPool.sol#259)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	- pool.token.safeTransfer(_sender,_amount) (contracts/distribution/
UniteRewardPool.sol#235)
	External calls sending eth:
	- safeUniteTransfer(_sender,_pending) (contracts/distribution/UniteRewardPool.sol#230)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	State variables written after the call(s):
	- user.rewardDebt = user.amount.mul(pool.accUnitePerShare).div(1e18) (contracts/
distribution/UniteRewardPool.sol#237)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities
INFO:Detectors:
TaxOfficeV2.addLiquidityTaxFree(address,uint256,uint256,uint256,uint256) (contracts/
TaxOfficeV2.sol#84-129) ignores return value by
IERC20(kitty).transferFrom(msg.sender,address(this),amtUnite) (contracts/
TaxOfficeV2.sol#101)
TaxOfficeV2.addLiquidityTaxFree(address,uint256,uint256,uint256,uint256) (contracts/
TaxOfficeV2.sol#84-129) ignores return value by
IERC20(token).transferFrom(msg.sender,address(this),amtToken) (contracts/
TaxOfficeV2.sol#102)
TaxOfficeV2.addLiquidityTaxFree(address,uint256,uint256,uint256,uint256) (contracts/
TaxOfficeV2.sol#84-129) ignores return value by
IERC20(kitty).transfer(msg.sender,amtUnite.sub(resultAmtUnite)) (contracts/
TaxOfficeV2.sol#123)
TaxOfficeV2.addLiquidityTaxFree(address,uint256,uint256,uint256,uint256) (contracts/
TaxOfficeV2.sol#84-129) ignores return value by
IERC20(token).transfer(msg.sender,amtToken.sub(resultAmtToken)) (contracts/
TaxOfficeV2.sol#126)
TaxOfficeV2.addLiquidityETHTaxFree(uint256,uint256,uint256) (contracts/
TaxOfficeV2.sol#131-168) ignores return value by
IERC20(kitty).transferFrom(msg.sender,address(this),amtUnite) (contracts/
TaxOfficeV2.sol#147)
TaxOfficeV2.addLiquidityETHTaxFree(uint256,uint256,uint256) (contracts/
TaxOfficeV2.sol#131-168) ignores return value by
IERC20(kitty).transfer(msg.sender,amtUnite.sub(resultAmtUnite)) (contracts/
TaxOfficeV2.sol#165)
TaxOfficeV2.taxFreeTransferFrom(address,address,uint256) (contracts/
TaxOfficeV2.sol#178-187) ignores return value by
IERC20(kitty).transferFrom(_sender,_recipient,_amt) (contracts/TaxOfficeV2.sol#185)
Treasury._sendToBoardroom(uint256) (contracts/Treasury.sol#459-489) ignores return
value by IERC20(kitty).transfer(daoFund,_daoFundSharedAmount) (contracts/
Treasury.sol#465)
Treasury._sendToBoardroom(uint256) (contracts/Treasury.sol#459-489) ignores return
value by IERC20(kitty).transfer(devFund,_devFundSharedAmount) (contracts/
Treasury.sol#472)
Treasury._sendToBoardroom(uint256) (contracts/Treasury.sol#459-489) ignores return
value by IERC20(kitty).transfer(team1Fund,_team1FundSharedAmount) (contracts/
Treasury.sol#479)
UShare.governanceRecoverUnsupported(IERC20,uint256,address) (contracts/
UShare.sol#141-147) ignores return value by _token.transfer(_to,_amount) (contracts/
UShare.sol#146)
Unite.governanceRecoverUnsupported(IERC20,uint256,address) (contracts/Unite.sol#65-71)
ignores return value by _token.transfer(_to,_amount) (contracts/Unite.sol#70)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-
transfer
```

```
INFO:Detectors:
Treasury.allocateSeigniorage() (contracts/Treasury.sol#501-541) performs a
multiplication on the result of a division:
-_seigniorage = kittySupply.mul(_percentage).div(1e18) (contracts/Treasury.sol#524)
-_savedForBoardroom = _seigniorage.mul(seigniorageExpansionFloorPercent).div(10000)
(contracts/Treasury.sol#525)
UShareRewardPool.pendingShare(uint256,address) (contracts/distribution/
UShareRewardPool.sol#152-163) performs a multiplication on the result of a division:
-_bshareReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint) (contracts/
distribution/UShareRewardPool.sol#159)
-accUSharePerShare = accUSharePerShare.add(_bshareReward.mul(1e18).div(tokenSupply))
(contracts/distribution/UShareRewardPool.sol#160)
UShareRewardPool.updatePool(uint256) (contracts/distribution/
UShareRewardPool.sol#174-194) performs a multiplication on the result of a division:
-_bshareReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint) (contracts/
distribution/UShareRewardPool.sol#190)
-pool.accUSharePerShare =
pool.accUSharePerShare.add(_bshareReward.mul(1e18).div(tokenSupply)) (contracts/
distribution/UShareRewardPool.sol#191)
UniteGenesisRewardPool.pendingUNITE(uint256,address) (contracts/distribution/
UniteGenesisRewardPool.sol#151-162) performs a multiplication on the result of a
division:
-_bombReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint) (contracts/
distribution/UniteGenesisRewardPool.sol#158)
-accUnitePerShare = accUnitePerShare.add(_bombReward.mul(1e18).div(tokenSupply))
(contracts/distribution/UniteGenesisRewardPool.sol#159)
UniteGenesisRewardPool.updatePool(uint256) (contracts/distribution/
UniteGenesisRewardPool.sol#173-193) performs a multiplication on the result of a
division:
-_bombReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint) (contracts/
distribution/UniteGenesisRewardPool.sol#189)
-pool.accUnitePerShare =
pool.accUnitePerShare.add(_bombReward.mul(1e18).div(tokenSupply)) (contracts/
distribution/UniteGenesisRewardPool.sol#190)
UniteRewardPool.pendingUNITE(uint256,address) (contracts/distribution/
UniteRewardPool.sol#156-167) performs a multiplication on the result of a division:
-_bombReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint) (contracts/
distribution/UniteRewardPool.sol#163)
-accUnitePerShare = accUnitePerShare.add(_bombReward.mul(1e18).div(tokenSupply))
(contracts/distribution/UniteRewardPool.sol#164)
UniteRewardPool.updatePool(uint256) (contracts/distribution/
UniteRewardPool.sol#178-198) performs a multiplication on the result of a division:
-_bombReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint) (contracts/
distribution/UniteRewardPool.sol#194)
-pool.accUnitePerShare =
pool.accUnitePerShare.add(_bombReward.mul(1e18).div(tokenSupply)) (contracts/
distribution/UniteRewardPool.sol#195)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply
INFO:Detectors:
```

```
UShareRewardPool.updatePool(uint256) (contracts/distribution/
UShareRewardPool.sol#174-194) uses a dangerous strict equality:
	- tokenSupply == 0 (contracts/distribution/UShareRewardPool.sol#180)
UniteGenesisRewardPool.updatePool(uint256) (contracts/distribution/
UniteGenesisRewardPool.sol#173-193) uses a dangerous strict equality:
	- tokenSupply == 0 (contracts/distribution/UniteGenesisRewardPool.sol#179)
UniteRewardPool.updatePool(uint256) (contracts/distribution/
UniteRewardPool.sol#178-198) uses a dangerous strict equality:
	- tokenSupply == 0 (contracts/distribution/UniteRewardPool.sol#184)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-
strict-equalities
INFO:Detectors:
Reentrancy in Treasury.buyBonds(uint256,uint256) (contracts/Treasury.sol#404-431):
	External calls:
	- IBasisAsset(kitty).burnFrom(msg.sender,_kittyAmount) (contracts/Treasury.sol#424)
	- IBasisAsset(bbond).mint(msg.sender,_bondAmount) (contracts/Treasury.sol#425)
	State variables written after the call(s):
	- epochSupplyContractionLeft = epochSupplyContractionLeft.sub(_kittyAmount) (contracts/
Treasury.sol#427)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1
INFO:Detectors:
Treasury.setSupplyTiersEntry(uint8,uint256) (contracts/Treasury.sol#297-308) contains a
tautology or contradiction:
	- require(bool,string)(_index >= 0,Index has to be higher than 0) (contracts/
Treasury.sol#298)
Treasury.setMaxExpansionTiersEntry(uint8,uint256) (contracts/Treasury.sol#310-316)
contains a tautology or contradiction:
	- require(bool,string)(_index >= 0,Index has to be higher than 0) (contracts/
Treasury.sol#311)
Treasury._calculateMaxSupplyExpansionPercent(uint256) (contracts/Treasury.sol#491-499)
contains a tautology or contradiction:
	- tierId >= 0 (contracts/Treasury.sol#492)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-
contradiction
INFO:Detectors:
Treasury.getUniteUpdatedPrice().price (contracts/Treasury.sol#157) is a local variable
never initialized
Treasury.getUnitePrice().price (contracts/Treasury.sol#149) is a local variable never
initialized
FixedPoint.mul(FixedPoint.uq112x112,uint256).z (contracts/lib/FixedPoint.sol#44) is a
```

```
local variable never initialized
Treasury.allocateSeigniorage()._savedForBond (contracts/Treasury.sol#513) is a local
variable never initialized
UniswapV2Library.getAmountsOut(address,uint256,address[]).i (contracts/lib/
UniswapV2Library.sol#97) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-
local-variables
INFO:Detectors:
TaxOfficeV2._approveTokenIfNeeded(address,address) (contracts/TaxOfficeV2.sol#193-197)
ignores return value by IERC20(_token).approve(_router,type()(uint256).max) (contracts/
TaxOfficeV2.sol#195)
Treasury.getUnitePrice() (contracts/Treasury.sol#148-154) ignores return value by
IOracle(kittyOracle).consult(kitty,1e18) (contracts/Treasury.sol#149-153)
Treasury.getUniteUpdatedPrice() (contracts/Treasury.sol#156-162) ignores return value
by IOracle(kittyOracle).twap(kitty,1e18) (contracts/Treasury.sol#157-161)
Treasury.buyBonds(uint256,uint256) (contracts/Treasury.sol#404-431) ignores return
value by IBasisAsset(bbond).mint(msg.sender,_bondAmount) (contracts/Treasury.sol#425)
Treasury._sendToBoardroom(uint256) (contracts/Treasury.sol#459-489) ignores return
value by IBasisAsset(kitty).mint(address(this),_amount) (contracts/Treasury.sol#460)
Treasury.allocateSeigniorage() (contracts/Treasury.sol#501-541) ignores return value by
IBasisAsset(kitty).mint(address(this),_savedForBond) (contracts/Treasury.sol#536)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
```

```
INFO:Detectors:
Boardroom.setOperator(address) (contracts/Boardroom.sol#138-140) should emit an event
for:
- operator = _operator (contracts/Boardroom.sol#139)
Treasury.setOperator(address) (contracts/Treasury.sol#275-277) should emit an event
for:
- operator = _operator (contracts/Treasury.sol#276)
Treasury.setBoardroom(address) (contracts/Treasury.sol#279-281) should emit an event
for:
- boardroom = _boardroom (contracts/Treasury.sol#280)
UShareRewardPool.setOperator(address) (contracts/distribution/
UShareRewardPool.sol#260-262) should emit an event for:
- operator = _operator (contracts/distribution/UShareRewardPool.sol#261)
UniteGenesisRewardPool.setOperator(address) (contracts/distribution/
UniteGenesisRewardPool.sol#263-265) should emit an event for:
- operator = _operator (contracts/distribution/UniteGenesisRewardPool.sol#264)
```

```
UniteRewardPool.setOperator(address) (contracts/distribution/
UniteRewardPool.sol#264-266) should emit an event for:
⊠- operator = _operator (contracts/distribution/UniteRewardPool.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
access-control
INFO:Detectors:
Boardroom.setLockUp(uint256,uint256) (contracts/Boardroom.sol#142-146) should emit an
event for:
⊠- withdrawLockupEpochs = _withdrawLockupEpochs (contracts/Boardroom.sol#144)
⊠- rewardLockupEpochs = _rewardLockupEpochs (contracts/Boardroom.sol#145)
Treasury.setUnitePriceCeiling(uint256) (contracts/Treasury.sol#287-290) should emit an
event for:
⊠- kittyPriceCeiling = _kittyPriceCeiling (contracts/Treasury.sol#289)
Treasury.setMaxSupplyExpansionPercents(uint256) (contracts/Treasury.sol#292-295) should
emit an event for:
⊠- maxSupplyExpansionPercent = _maxSupplyExpansionPercent (contracts/Treasury.sol#294)
Treasury.setBondDepletionFloorPercent(uint256) (contracts/Treasury.sol#318-321) should
emit an event for:
⊠- bondDepletionFloorPercent = _bondDepletionFloorPercent (contracts/Treasury.sol#320)
Treasury.setMaxDebtRatioPercent(uint256) (contracts/Treasury.sol#328-331) should emit
an event for:
⊠- maxDebtRatioPercent = _maxDebtRatioPercent (contracts/Treasury.sol#330)
Treasury.setBootstrap(uint256,uint256) (contracts/Treasury.sol#333-338) should emit an
event for:
⊠- bootstrapEpochs = _bootstrapEpochs (contracts/Treasury.sol#336)
⊠- bootstrapSupplyExpansionPercent = _bootstrapSupplyExpansionPercent (contracts/
Treasury.sol#337)
Treasury.setExtraFunds(address,uint256,address,uint256,address,uint256) (contracts/
Treasury.sol#340-360) should emit an event for:
⊠- daoFundSharedPercent = _daoFundSharedPercent (contracts/Treasury.sol#355)
⊠- devFundSharedPercent = _devFundSharedPercent (contracts/Treasury.sol#357)
⊠- team1FundSharedPercent = _team1FundSharedPercent (contracts/Treasury.sol#359)
Treasury.setMaxDiscountRate(uint256) (contracts/Treasury.sol#362-364) should emit an
event for:
⊠- maxDiscountRate = _maxDiscountRate (contracts/Treasury.sol#363)
Treasury.setMaxPremiumRate(uint256) (contracts/Treasury.sol#366-368) should emit an
event for:
⊠- maxPremiumRate = _maxPremiumRate (contracts/Treasury.sol#367)
Treasury.setDiscountPercent(uint256) (contracts/Treasury.sol#370-373) should emit an
event for:
⊠- discountPercent = _discountPercent (contracts/Treasury.sol#372)
```

```
Treasury.setPremiumThreshold(uint256) (contracts/Treasury.sol#375-379) should emit an
event for:
- premiumThreshold = _premiumThreshold (contracts/Treasury.sol#378)
Treasury.setPremiumPercent(uint256) (contracts/Treasury.sol#381-384) should emit an
event for:
- premiumPercent = _premiumPercent (contracts/Treasury.sol#383)
Treasury.setMintingFactorForPayingDebt(uint256) (contracts/Treasury.sol#386-389) should
emit an event for:
- mintingFactorForPayingDebt = _mintingFactorForPayingDebt (contracts/
Treasury.sol#388)
UShareRewardPool.add(uint256,IERC20,bool,uint256) (contracts/distribution/
UShareRewardPool.sol#85-123) should emit an event for:
- totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/distribution/
UShareRewardPool.sol#121)
UShareRewardPool.set(uint256,uint256) (contracts/distribution/
UShareRewardPool.sol#126-135) should emit an event for:
- totalAllocPoint = totalAllocPoint.sub(pool.allocPoint).add(_allocPoint) (contracts/
distribution/UShareRewardPool.sol#130-132)
UniteGenesisRewardPool.add(uint256,IERC20,bool,uint256) (contracts/distribution/
UniteGenesisRewardPool.sol#94-124) should emit an event for:
- totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/distribution/
UniteGenesisRewardPool.sol#122)
UniteGenesisRewardPool.set(uint256,uint256) (contracts/distribution/
UniteGenesisRewardPool.sol#127-134) should emit an event for:
- totalAllocPoint = totalAllocPoint.sub(pool.allocPoint).add(_allocPoint) (contracts/
distribution/UniteGenesisRewardPool.sol#131)
UniteRewardPool.add(uint256,IERC20,bool,uint256) (contracts/distribution/
UniteRewardPool.sol#89-119) should emit an event for:
- totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/distribution/
UniteRewardPool.sol#117)
UniteRewardPool.set(uint256,uint256) (contracts/distribution/
UniteRewardPool.sol#122-129) should emit an event for:
- totalAllocPoint = totalAllocPoint.sub(pool.allocPoint).add(_allocPoint) (contracts/
distribution/UniteRewardPool.sol#126)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic
INFO:Detectors:
Boardroom.setOperator(address)._operator (contracts/Boardroom.sol#138) lacks a zero-
check on :
- operator = _operator (contracts/Boardroom.sol#139)
Timelock.constructor(address,uint256).admin_ (contracts/Timelock.sol#56) lacks a zero-
```

```
check on :
⊠⊠- admin = admin_ (contracts/Timelock.sol#60)
Timelock.setPendingAdmin(address).pendingAdmin_ (contracts/Timelock.sol#83) lacks a
zero-check on :
⊠⊠- pendingAdmin = pendingAdmin_ (contracts/Timelock.sol#85)
Timelock.executeTransaction(address,uint256,string,bytes,uint256).target (contracts/
Timelock.sol#123) lacks a zero-check on :
⊠⊠- (success,returnData) = target.call{value: value}(callData) (contracts/
Timelock.sol#147)
Treasury.initialize(address,address,address,address,address,uint256)._kitty (contracts/
Treasury.sol#232) lacks a zero-check on :
⊠⊠- kitty = _kitty (contracts/Treasury.sol#239)
Treasury.initialize(address,address,address,address,address,uint256)._bbond (contracts/
Treasury.sol#233) lacks a zero-check on :
⊠⊠- bbond = _bbond (contracts/Treasury.sol#240)
Treasury.initialize(address,address,address,address,address,uint256)._bshare (contracts/
Treasury.sol#234) lacks a zero-check on :
⊠⊠- bshare = _bshare (contracts/Treasury.sol#241)
Treasury.initialize(address,address,address,address,address,uint256)._kittyOracle
(contracts/Treasury.sol#235) lacks a zero-check on :
⊠⊠- kittyOracle = _kittyOracle (contracts/Treasury.sol#242)
Treasury.initialize(address,address,address,address,address,uint256)._boardroom
(contracts/Treasury.sol#236) lacks a zero-check on :
⊠⊠- boardroom = _boardroom (contracts/Treasury.sol#243)
Treasury.setOperator(address)._operator (contracts/Treasury.sol#275) lacks a zero-check
on :
⊠⊠- operator = _operator (contracts/Treasury.sol#276)
Treasury.setBoardroom(address)._boardroom (contracts/Treasury.sol#279) lacks a zero-
check on :
⊠⊠- boardroom = _boardroom (contracts/Treasury.sol#280)
Treasury.setUniteOracle(address)._kittyOracle (contracts/Treasury.sol#283) lacks a zero-
check on :
⊠⊠- kittyOracle = _kittyOracle (contracts/Treasury.sol#284)
UShare.setTreasuryFund(address)._communityFund (contracts/UShare.sol#67) lacks a zero-
check on :
⊠⊠- communityFund = _communityFund (contracts/UShare.sol#69)
UShareRewardPool.setOperator(address)._operator (contracts/distribution/
UShareRewardPool.sol#260) lacks a zero-check on :
⊠⊠- operator = _operator (contracts/distribution/UShareRewardPool.sol#261)
UniteGenesisRewardPool.setOperator(address)._operator (contracts/distribution/
UniteGenesisRewardPool.sol#263) lacks a zero-check on :
		- operator = _operator (contracts/distribution/UniteGenesisRewardPool.sol#264)
UniteRewardPool.setOperator(address)._operator (contracts/distribution/
UniteRewardPool.sol#264) lacks a zero-check on :
		- operator = _operator (contracts/distribution/UniteRewardPool.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
address-validation
INFO:Detectors:
Modifier Migrations.restricted() (contracts/Migrations.sol#13-15) does not always
execute _; or revertReference: https://github.com/crytic/slither/wiki/Detector-
Documentation#incorrect-modifier
INFO:Detectors:
Distributor.distribute() (contracts/Distributor.sol#14-18) has external calls inside a
loop: distributors[i].distribute() (contracts/Distributor.sol#16)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-
a-loop
INFO:Detectors:
Variable 'Treasury.getUnitePrice().price (contracts/Treasury.sol#149)' in
Treasury.getUnitePrice() (contracts/Treasury.sol#148-154) potentially used before
declaration: uint256(price) (contracts/Treasury.sol#150)
Variable 'Treasury.getUniteUpdatedPrice().price (contracts/Treasury.sol#157)' in
Treasury.getUniteUpdatedPrice() (contracts/Treasury.sol#156-162) potentially used
before declaration: uint256(price) (contracts/Treasury.sol#158)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-
declaration-usage-of-local-variables
INFO:Detectors:
Reentrancy in Treasury.allocateSeigniorage() (contracts/Treasury.sol#501-541):
	External calls:
	- _updateUnitePrice() (contracts/Treasury.sol#502)
		- IOracle(kittyOracle).update() (contracts/Treasury.sol#394)
	State variables written after the call(s):
	- _mse = _calculateMaxSupplyExpansionPercent(kittySupply).mul(1e14) (contracts/
Treasury.sol#515)
		- maxSupplyExpansionPercent = maxExpansionTiers[tierId] (contracts/Treasury.sol#494)
	- previousEpochUnitePrice = getUnitePrice() (contracts/Treasury.sol#503)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2
INFO:Detectors:
Reentrancy in Treasury._sendToBoardroom(uint256) (contracts/Treasury.sol#459-489):
	External calls:
	- IBasisAsset(kitty).mint(address(this),_amount) (contracts/Treasury.sol#460)
	- IERC20(kitty).transfer(daoFund,_daoFundSharedAmount) (contracts/Treasury.sol#465)
```

```
Event emitted after the call(s):
- DaoFundFunded(now,_daoFundSharedAmount) (contracts/Treasury.sol#466)
Reentrancy in Treasury._sendToBoardroom(uint256) (contracts/Treasury.sol#459-489):
External calls:
- IBasisAsset(kitty).mint(address(this),_amount) (contracts/Treasury.sol#460)
- IERC20(kitty).transfer(daoFund,_daoFundSharedAmount) (contracts/Treasury.sol#465)
- IERC20(kitty).transfer(devFund,_devFundSharedAmount) (contracts/Treasury.sol#472)
Event emitted after the call(s):
- DevFundFunded(now,_devFundSharedAmount) (contracts/Treasury.sol#473)
Reentrancy in Treasury._sendToBoardroom(uint256) (contracts/Treasury.sol#459-489):
External calls:
- IBasisAsset(kitty).mint(address(this),_amount) (contracts/Treasury.sol#460)
- IERC20(kitty).transfer(daoFund,_daoFundSharedAmount) (contracts/Treasury.sol#465)
- IERC20(kitty).transfer(devFund,_devFundSharedAmount) (contracts/Treasury.sol#472)
- IERC20(kitty).transfer(team1Fund,_team1FundSharedAmount) (contracts/
Treasury.sol#479)
Event emitted after the call(s):
- TeamFundFunded(now,_team1FundSharedAmount) (contracts/Treasury.sol#480)
Reentrancy in Treasury._sendToBoardroom(uint256) (contracts/Treasury.sol#459-489):
External calls:
- IBasisAsset(kitty).mint(address(this),_amount) (contracts/Treasury.sol#460)
- IERC20(kitty).transfer(daoFund,_daoFundSharedAmount) (contracts/Treasury.sol#465)
- IERC20(kitty).transfer(devFund,_devFundSharedAmount) (contracts/Treasury.sol#472)
- IERC20(kitty).transfer(team1Fund,_team1FundSharedAmount) (contracts/
Treasury.sol#479)
- IERC20(kitty).safeApprove(boardroom,0) (contracts/Treasury.sol#485)
- IERC20(kitty).safeApprove(boardroom,_amount) (contracts/Treasury.sol#486)
- IBoardroom(boardroom).allocateSeigniorage(_amount) (contracts/Treasury.sol#487)
Event emitted after the call(s):
- BoardroomFunded(now,_amount) (contracts/Treasury.sol#488)
Reentrancy in Boardroom.allocateSeigniorage(uint256) (contracts/Boardroom.sol#233-246):
External calls:
- kitty.safeTransferFrom(msg.sender,address(this),amount) (contracts/
Boardroom.sol#244)
Event emitted after the call(s):
- RewardAdded(msg.sender,amount) (contracts/Boardroom.sol#245)
Reentrancy in Treasury.allocateSeigniorage() (contracts/Treasury.sol#501-541):
External calls:
- _updateUnitePrice() (contracts/Treasury.sol#502)
- IOracle(kittyOracle).update() (contracts/Treasury.sol#394)
- _sendToBoardroom(kittySupply.mul(bootstrapSupplyExpansionPercent).div(10000))
```

```
(contracts/Treasury.sol#507)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- IBasisAsset(kitty).mint(address(this),_amount) (contracts/Treasury.sol#460)
		- IERC20(kitty).transfer(daoFund,_daoFundSharedAmount) (contracts/Treasury.sol#465)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
		- IERC20(kitty).transfer(devFund,_devFundSharedAmount) (contracts/Treasury.sol#472)
		- IERC20(kitty).transfer(team1Fund,_team1FundSharedAmount) (contracts/
Treasury.sol#479)
		- IERC20(kitty).safeApprove(boardroom,0) (contracts/Treasury.sol#485)
		- IERC20(kitty).safeApprove(boardroom,_amount) (contracts/Treasury.sol#486)
		- IBoardroom(boardroom).allocateSeigniorage(_amount) (contracts/Treasury.sol#487)
	External calls sending eth:
	- _sendToBoardroom(kittySupply.mul(bootstrapSupplyExpansionPercent).div(10000))
(contracts/Treasury.sol#507)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	Event emitted after the call(s):
	- BoardroomFunded(now,_amount) (contracts/Treasury.sol#488)
		- _sendToBoardroom(kittySupply.mul(bootstrapSupplyExpansionPercent).div(10000))
(contracts/Treasury.sol#507)
	- DaoFundFunded(now,_daoFundSharedAmount) (contracts/Treasury.sol#466)
		- _sendToBoardroom(kittySupply.mul(bootstrapSupplyExpansionPercent).div(10000))
(contracts/Treasury.sol#507)
	- DevFundFunded(now,_devFundSharedAmount) (contracts/Treasury.sol#473)
		- _sendToBoardroom(kittySupply.mul(bootstrapSupplyExpansionPercent).div(10000))
(contracts/Treasury.sol#507)
	- TeamFundFunded(now,_team1FundSharedAmount) (contracts/Treasury.sol#480)
		- _sendToBoardroom(kittySupply.mul(bootstrapSupplyExpansionPercent).div(10000))
(contracts/Treasury.sol#507)
Reentrancy in Treasury.allocateSeigniorage() (contracts/Treasury.sol#501-541):
	External calls:
	- _updateUnitePrice() (contracts/Treasury.sol#502)
		- IOracle(kittyOracle).update() (contracts/Treasury.sol#394)
	- _sendToBoardroom(_savedForBoardroom) (contracts/Treasury.sol#532)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- IBasisAsset(kitty).mint(address(this),_amount) (contracts/Treasury.sol#460)
		- IERC20(kitty).transfer(daoFund,_daoFundSharedAmount) (contracts/Treasury.sol#465)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
		- IERC20(kitty).transfer(devFund,_devFundSharedAmount) (contracts/Treasury.sol#472)
		- IERC20(kitty).transfer(team1Fund,_team1FundSharedAmount) (contracts/
Treasury.sol#479)
		- IERC20(kitty).safeApprove(boardroom,0) (contracts/Treasury.sol#485)
		- IERC20(kitty).safeApprove(boardroom,_amount) (contracts/Treasury.sol#486)
		- IBoardroom(boardroom).allocateSeigniorage(_amount) (contracts/Treasury.sol#487)
	External calls sending eth:
	- _sendToBoardroom(_savedForBoardroom) (contracts/Treasury.sol#532)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	Event emitted after the call(s):
	- BoardroomFunded(now,_amount) (contracts/Treasury.sol#488)
		- _sendToBoardroom(_savedForBoardroom) (contracts/Treasury.sol#532)
	- DaoFundFunded(now,_daoFundSharedAmount) (contracts/Treasury.sol#466)
		- _sendToBoardroom(_savedForBoardroom) (contracts/Treasury.sol#532)
	- DevFundFunded(now,_devFundSharedAmount) (contracts/Treasury.sol#473)
		- _sendToBoardroom(_savedForBoardroom) (contracts/Treasury.sol#532)
	- TeamFundFunded(now,_team1FundSharedAmount) (contracts/Treasury.sol#480)
		- _sendToBoardroom(_savedForBoardroom) (contracts/Treasury.sol#532)
Reentrancy in Treasury.allocateSeigniorage() (contracts/Treasury.sol#501-541):
	External calls:
	- _updateUnitePrice() (contracts/Treasury.sol#502)
		- IOracle(kittyOracle).update() (contracts/Treasury.sol#394)
	- _sendToBoardroom(_savedForBoardroom) (contracts/Treasury.sol#532)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- IBasisAsset(kitty).mint(address(this),_amount) (contracts/Treasury.sol#460)
		- IERC20(kitty).transfer(daoFund,_daoFundSharedAmount) (contracts/Treasury.sol#465)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
		- IERC20(kitty).transfer(devFund,_devFundSharedAmount) (contracts/Treasury.sol#472)
		- IERC20(kitty).transfer(team1Fund,_team1FundSharedAmount) (contracts/
Treasury.sol#479)
		- IERC20(kitty).safeApprove(boardroom,0) (contracts/Treasury.sol#485)
		- IERC20(kitty).safeApprove(boardroom,_amount) (contracts/Treasury.sol#486)
		- IBoardroom(boardroom).allocateSeigniorage(_amount) (contracts/Treasury.sol#487)
	- IBasisAsset(kitty).mint(address(this),_savedForBond) (contracts/Treasury.sol#536)
	External calls sending eth:
	- _sendToBoardroom(_savedForBoardroom) (contracts/Treasury.sol#532)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
Event emitted after the call(s):
- TreasuryFunded(now,_savedForBond) (contracts/Treasury.sol#537)
Reentrancy in Treasury.buyBonds(uint256,uint256) (contracts/Treasury.sol#404-431):
External calls:
- IBasisAsset(kitty).burnFrom(msg.sender,_kittyAmount) (contracts/Treasury.sol#424)
- IBasisAsset(bbond).mint(msg.sender,_bondAmount) (contracts/Treasury.sol#425)
- _updateUnitePrice() (contracts/Treasury.sol#428)
- IOracle(kittyOracle).update() (contracts/Treasury.sol#394)
Event emitted after the call(s):
- BoughtBonds(msg.sender,_kittyAmount,_bondAmount) (contracts/Treasury.sol#430)
Reentrancy in Boardroom.claimReward() (contracts/Boardroom.sol#222-231):
External calls:
- kitty.safeTransfer(msg.sender,reward) (contracts/Boardroom.sol#228)
Event emitted after the call(s):
- RewardPaid(msg.sender,reward) (contracts/Boardroom.sol#229)
Reentrancy in SimpleERCFund.deposit(address,uint256,string) (contracts/
SimpleERCFund.sol#14-21):
External calls:
- IERC20(token).safeTransferFrom(msg.sender,address(this),amount) (contracts/
SimpleERCFund.sol#19)
Event emitted after the call(s):
- Deposit(msg.sender,now,reason) (contracts/SimpleERCFund.sol#20)
Reentrancy in UShareRewardPool.deposit(uint256,uint256) (contracts/distribution/
UShareRewardPool.sol#197-215):
External calls:
- safeUShareTransfer(_sender,_pending) (contracts/distribution/
UShareRewardPool.sol#205)
- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
- bshare.safeTransfer(_to,_bshareBal) (contracts/distribution/
UShareRewardPool.sol#253)
- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
- bshare.safeTransfer(_to,_amount) (contracts/distribution/UShareRewardPool.sol#255)
External calls sending eth:
- safeUShareTransfer(_sender,_pending) (contracts/distribution/
UShareRewardPool.sol#205)
- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
Event emitted after the call(s):
	- RewardPaid(_sender,_pending) (contracts/distribution/UShareRewardPool.sol#206)
Reentrancy in UShareRewardPool.deposit(uint256,uint256) (contracts/distribution/
UShareRewardPool.sol#197-215):
	External calls:
	- safeUShareTransfer(_sender,_pending) (contracts/distribution/
UShareRewardPool.sol#205)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- bshare.safeTransfer(_to,_bshareBal) (contracts/distribution/
UShareRewardPool.sol#253)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
		- bshare.safeTransfer(_to,_amount) (contracts/distribution/UShareRewardPool.sol#255)
	- pool.token.safeTransferFrom(_sender,address(this),_amount) (contracts/distribution/
UShareRewardPool.sol#210)
	External calls sending eth:
	- safeUShareTransfer(_sender,_pending) (contracts/distribution/
UShareRewardPool.sol#205)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	Event emitted after the call(s):
	- Deposit(_sender,_pid,_amount) (contracts/distribution/UShareRewardPool.sol#214)
Reentrancy in UniteGenesisRewardPool.deposit(uint256,uint256) (contracts/distribution/
UniteGenesisRewardPool.sol#196-218):
	External calls:
	- safeUniteTransfer(_sender,_pending) (contracts/distribution/
UniteGenesisRewardPool.sol#204)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- bomb.safeTransfer(_to,_bombBalance) (contracts/distribution/
UniteGenesisRewardPool.sol#256)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
		- bomb.safeTransfer(_to,_amount) (contracts/distribution/
UniteGenesisRewardPool.sol#258)
	External calls sending eth:
	- safeUniteTransfer(_sender,_pending) (contracts/distribution/
UniteGenesisRewardPool.sol#204)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	Event emitted after the call(s):
```

```
- RewardPaid(_sender,_pending) (contracts/distribution/UniteGenesisRewardPool.sol#205)
Reentrancy in UniteGenesisRewardPool.deposit(uint256,uint256) (contracts/distribution/
UniteGenesisRewardPool.sol#196-218):
External calls:
- safeUniteTransfer(_sender,_pending) (contracts/distribution/
UniteGenesisRewardPool.sol#204)
- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
- bomb.safeTransfer(_to,_bombBalance) (contracts/distribution/
UniteGenesisRewardPool.sol#256)
- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
- bomb.safeTransfer(_to,_amount) (contracts/distribution/
UniteGenesisRewardPool.sol#258)
- pool.token.safeTransferFrom(_sender,address(this),_amount) (contracts/distribution/
UniteGenesisRewardPool.sol#209)
External calls sending eth:
- safeUniteTransfer(_sender,_pending) (contracts/distribution/
UniteGenesisRewardPool.sol#204)
- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
Event emitted after the call(s):
- Deposit(_sender,_pid,_amount) (contracts/distribution/
UniteGenesisRewardPool.sol#217)
Reentrancy in UniteRewardPool.deposit(uint256,uint256) (contracts/distribution/
UniteRewardPool.sol#201-219):
External calls:
- safeUniteTransfer(_sender,_pending) (contracts/distribution/UniteRewardPool.sol#209)
- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
- bomb.safeTransfer(_to,_bombBal) (contracts/distribution/UniteRewardPool.sol#257)
- bomb.safeTransfer(_to,_amount) (contracts/distribution/UniteRewardPool.sol#259)
- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
External calls sending eth:
- safeUniteTransfer(_sender,_pending) (contracts/distribution/UniteRewardPool.sol#209)
- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
Event emitted after the call(s):
- RewardPaid(_sender,_pending) (contracts/distribution/UniteRewardPool.sol#210)
Reentrancy in UniteRewardPool.deposit(uint256,uint256) (contracts/distribution/
UniteRewardPool.sol#201-219):
```

```
External calls:
- safeUniteTransfer(_sender,_pending) (contracts/distribution/UniteRewardPool.sol#209)
- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
- bomb.safeTransfer(_to,_bombBal) (contracts/distribution/UniteRewardPool.sol#257)
- bomb.safeTransfer(_to,_amount) (contracts/distribution/UniteRewardPool.sol#259)
- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
- pool.token.safeTransferFrom(_sender,address(this),_amount) (contracts/distribution/
UniteRewardPool.sol#214)
External calls sending eth:
- safeUniteTransfer(_sender,_pending) (contracts/distribution/UniteRewardPool.sol#209)
- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
Event emitted after the call(s):
- Deposit(_sender,_pid,_amount) (contracts/distribution/UniteRewardPool.sol#218)
Reentrancy in UShareRewardPool.emergencyWithdraw(uint256) (contracts/distribution/
UShareRewardPool.sol#238-246):
External calls:
- pool.token.safeTransfer(msg.sender,_amount) (contracts/distribution/
UShareRewardPool.sol#244)
Event emitted after the call(s):
- EmergencyWithdraw(msg.sender,_pid,_amount) (contracts/distribution/
UShareRewardPool.sol#245)
Reentrancy in UniteGenesisRewardPool.emergencyWithdraw(uint256) (contracts/distribution/
UniteGenesisRewardPool.sol#241-249):
External calls:
- pool.token.safeTransfer(msg.sender,_amount) (contracts/distribution/
UniteGenesisRewardPool.sol#247)
Event emitted after the call(s):
- EmergencyWithdraw(msg.sender,_pid,_amount) (contracts/distribution/
UniteGenesisRewardPool.sol#248)
Reentrancy in UniteRewardPool.emergencyWithdraw(uint256) (contracts/distribution/
UniteRewardPool.sol#242-250):
External calls:
- pool.token.safeTransfer(msg.sender,_amount) (contracts/distribution/
UniteRewardPool.sol#248)
Event emitted after the call(s):
- EmergencyWithdraw(msg.sender,_pid,_amount) (contracts/distribution/
UniteRewardPool.sol#249)
Reentrancy in Timelock.executeTransaction(address,uint256,string,bytes,uint256)
```

```
(contracts/Timelock.sol#122-153):
	External calls:
	- (success,returnData) = target.call{value: value}(callData) (contracts/
Timelock.sol#147)
	Event emitted after the call(s):
	- ExecuteTransaction(txHash,target,value,signature,data,eta) (contracts/
Timelock.sol#150)
Reentrancy in Treasury.redeemBonds(uint256,uint256) (contracts/Treasury.sol#433-457):
	External calls:
	- IBasisAsset(bbond).burnFrom(msg.sender,_bondAmount) (contracts/Treasury.sol#451)
	- IERC20(kitty).safeTransfer(msg.sender,_kittyAmount) (contracts/Treasury.sol#452)
	- _updateUnitePrice() (contracts/Treasury.sol#454)
		- IOracle(kittyOracle).update() (contracts/Treasury.sol#394)
	Event emitted after the call(s):
	- RedeemedBonds(msg.sender,_kittyAmount,_bondAmount) (contracts/Treasury.sol#456)
Reentrancy in Boardroom.stake(uint256) (contracts/Boardroom.sol#203-208):
	External calls:
	- super.stake(amount) (contracts/Boardroom.sol#205)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- share.safeTransferFrom(msg.sender,address(this),amount) (contracts/
Boardroom.sol#32)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	External calls sending eth:
	- super.stake(amount) (contracts/Boardroom.sol#205)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	Event emitted after the call(s):
	- Staked(msg.sender,amount) (contracts/Boardroom.sol#207)
Reentrancy in Boardroom.withdraw(uint256) (contracts/Boardroom.sol#210-216):
	External calls:
	- claimReward() (contracts/Boardroom.sol#213)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
		- kitty.safeTransfer(msg.sender,reward) (contracts/Boardroom.sol#228)
	- super.withdraw(amount) (contracts/Boardroom.sol#214)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
```

```
		- share.safeTransfer(msg.sender,amount) (contracts/Boardroom.sol#40)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	External calls sending eth:
	- claimReward() (contracts/Boardroom.sol#213)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	- super.withdraw(amount) (contracts/Boardroom.sol#214)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	Event emitted after the call(s):
	- Withdrawn(msg.sender,amount) (contracts/Boardroom.sol#215)
Reentrancy in SimpleERCFund.withdraw(address,uint256,address,string) (contracts/
SimpleERCFund.sol#23-31):
	External calls:
	- IERC20(token).safeTransfer(to,amount) (contracts/SimpleERCFund.sol#29)
	Event emitted after the call(s):
	- Withdrawal(msg.sender,to,now,reason) (contracts/SimpleERCFund.sol#30)
Reentrancy in UShareRewardPool.withdraw(uint256,uint256) (contracts/distribution/
UShareRewardPool.sol#218-235):
	External calls:
	- safeUShareTransfer(_sender,_pending) (contracts/distribution/
UShareRewardPool.sol#226)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- bshare.safeTransfer(_to,_bshareBal) (contracts/distribution/
UShareRewardPool.sol#253)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
		- bshare.safeTransfer(_to,_amount) (contracts/distribution/UShareRewardPool.sol#255)
	External calls sending eth:
	- safeUShareTransfer(_sender,_pending) (contracts/distribution/
UShareRewardPool.sol#226)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
	Event emitted after the call(s):
	- RewardPaid(_sender,_pending) (contracts/distribution/UShareRewardPool.sol#227)
Reentrancy in UShareRewardPool.withdraw(uint256,uint256) (contracts/distribution/
UShareRewardPool.sol#218-235):
	External calls:
	- safeUShareTransfer(_sender,_pending) (contracts/distribution/
UShareRewardPool.sol#226)
```

```
⊠⊠- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
⊠⊠- bshare.safeTransfer(_to,_bshareBal) (contracts/distribution/
UShareRewardPool.sol#253)
⊠⊠- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
⊠⊠- bshare.safeTransfer(_to,_amount) (contracts/distribution/UShareRewardPool.sol#255)
⊠- pool.token.safeTransfer(_sender,_amount) (contracts/distribution/
UShareRewardPool.sol#231)
⊠External calls sending eth:
⊠- safeUShareTransfer(_sender,_pending) (contracts/distribution/
UShareRewardPool.sol#226)
⊠⊠- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
⊠Event emitted after the call(s):
⊠- Withdraw(_sender,_pid,_amount) (contracts/distribution/UShareRewardPool.sol#234)
Reentrancy in UniteGenesisRewardPool.withdraw(uint256,uint256) (contracts/distribution/
UniteGenesisRewardPool.sol#221-238):
⊠External calls:
⊠- safeUniteTransfer(_sender,_pending) (contracts/distribution/
UniteGenesisRewardPool.sol#229)
⊠⊠- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
⊠⊠- bomb.safeTransfer(_to,_bombBalance) (contracts/distribution/
UniteGenesisRewardPool.sol#256)
⊠⊠- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
⊠⊠- bomb.safeTransfer(_to,_amount) (contracts/distribution/
UniteGenesisRewardPool.sol#258)
⊠External calls sending eth:
⊠- safeUniteTransfer(_sender,_pending) (contracts/distribution/
UniteGenesisRewardPool.sol#229)
⊠⊠- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
⊠Event emitted after the call(s):
⊠- RewardPaid(_sender,_pending) (contracts/distribution/UniteGenesisRewardPool.sol#230)
Reentrancy in UniteGenesisRewardPool.withdraw(uint256,uint256) (contracts/distribution/
UniteGenesisRewardPool.sol#221-238):
⊠External calls:
⊠- safeUniteTransfer(_sender,_pending) (contracts/distribution/
UniteGenesisRewardPool.sol#229)
```

```
⊠⊠- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
⊠⊠- bomb.safeTransfer(_to,_bombBalance) (contracts/distribution/
UniteGenesisRewardPool.sol#256)
⊠⊠- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
⊠⊠- bomb.safeTransfer(_to,_amount) (contracts/distribution/
UniteGenesisRewardPool.sol#258)
⊠- pool.token.safeTransfer(_sender,_amount) (contracts/distribution/
UniteGenesisRewardPool.sol#234)
⊠External calls sending eth:
⊠- safeUniteTransfer(_sender,_pending) (contracts/distribution/
UniteGenesisRewardPool.sol#229)
⊠⊠- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
⊠Event emitted after the call(s):
⊠- Withdraw(_sender,_pid,_amount) (contracts/distribution/
UniteGenesisRewardPool.sol#237)
Reentrancy in UniteRewardPool.withdraw(uint256,uint256) (contracts/distribution/
UniteRewardPool.sol#222-239):
⊠External calls:
⊠- safeUniteTransfer(_sender,_pending) (contracts/distribution/UniteRewardPool.sol#230)
⊠⊠- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
⊠⊠- bomb.safeTransfer(_to,_bombBal) (contracts/distribution/UniteRewardPool.sol#257)
⊠⊠- bomb.safeTransfer(_to,_amount) (contracts/distribution/UniteRewardPool.sol#259)
⊠⊠- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
⊠External calls sending eth:
⊠- safeUniteTransfer(_sender,_pending) (contracts/distribution/UniteRewardPool.sol#230)
⊠⊠- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
⊠Event emitted after the call(s):
⊠- RewardPaid(_sender,_pending) (contracts/distribution/UniteRewardPool.sol#231)
Reentrancy in UniteRewardPool.withdraw(uint256,uint256) (contracts/distribution/
UniteRewardPool.sol#222-239):
⊠External calls:
⊠- safeUniteTransfer(_sender,_pending) (contracts/distribution/UniteRewardPool.sol#230)
⊠⊠- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
⊠⊠- bomb.safeTransfer(_to,_bombBal) (contracts/distribution/UniteRewardPool.sol#257)
⊠⊠- bomb.safeTransfer(_to,_amount) (contracts/distribution/UniteRewardPool.sol#259)
⊠⊠- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
⊠- pool.token.safeTransfer(_sender,_amount) (contracts/distribution/
UniteRewardPool.sol#235)
⊠External calls sending eth:
⊠- safeUniteTransfer(_sender,_pending) (contracts/distribution/UniteRewardPool.sol#230)
⊠⊠- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#119)
⊠Event emitted after the call(s):
⊠- Withdraw(_sender,_pid,_amount) (contracts/distribution/UniteRewardPool.sol#238)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3
INFO:Detectors:
TaxOfficeV2.addLiquidityTaxFree(address,uint256,uint256,uint256,uint256) (contracts/
TaxOfficeV2.sol#84-129) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- amtUnite.sub(resultAmtUnite) > 0 (contracts/TaxOfficeV2.sol#122)
⊠- amtToken.sub(resultAmtToken) > 0 (contracts/TaxOfficeV2.sol#125)
TaxOfficeV2.addLiquidityETHTaxFree(uint256,uint256,uint256) (contracts/
TaxOfficeV2.sol#131-168) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- amtUnite.sub(resultAmtUnite) > 0 (contracts/TaxOfficeV2.sol#164)
Timelock.queueTransaction(address,uint256,string,bytes,uint256) (contracts/
Timelock.sol#90-105) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- require(bool,string)(eta >=
getBlockTimestamp().add(delay),Timelock::queueTransaction: Estimated execution block
must satisfy delay.) (contracts/Timelock.sol#98)
Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/
Timelock.sol#122-153) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- require(bool,string)(getBlockTimestamp() >= eta,Timelock::executeTransaction:
Transaction hasn't surpassed time lock.) (contracts/Timelock.sol#133)
⊠- require(bool,string)(getBlockTimestamp() <=
eta.add(GRACE_PERIOD),Timelock::executeTransaction: Transaction is stale.) (contracts/
Timelock.sol#134)
UShare.unclaimedTreasuryFund() (contracts/UShare.sol#84-89) uses timestamp for
comparisons
⊠Dangerous comparisons:
⊠- _now > endTime (contracts/UShare.sol#86)
```

```
- communityFundLastClaimed >= _now (contracts/UShare.sol#87)
UShare.unclaimedDevFund() (contracts/UShare.sol#91-96) uses timestamp for comparisons
Dangerous comparisons:
- _now > endTime (contracts/UShare.sol#93)
- devFundLastClaimed >= _now (contracts/UShare.sol#94)
UShare.unclaimedTeam1Fund() (contracts/UShare.sol#98-103) uses timestamp for
comparisons
Dangerous comparisons:
- _now > endTime (contracts/UShare.sol#100)
- team1FundLastClaimed >= _now (contracts/UShare.sol#101)
UShareRewardPool.constructor(address,uint256) (contracts/distribution/
UShareRewardPool.sol#59-70) uses timestamp for comparisons
Dangerous comparisons:
- _poolStartTime == 0 || _poolStartTime < block.timestamp (contracts/distribution/
UShareRewardPool.sol#63)
UShareRewardPool.checkPoolDuplicate(IERC20) (contracts/distribution/
UShareRewardPool.sol#77-82) uses timestamp for comparisons
Dangerous comparisons:
- pid < length (contracts/distribution/UShareRewardPool.sol#79)
- require(bool,string)(poolInfo[pid].token != _token,UShareRewardPool: existing pool?)
(contracts/distribution/UShareRewardPool.sol#80)
UShareRewardPool.add(uint256,IERC20,bool,uint256) (contracts/distribution/
UShareRewardPool.sol#85-123) uses timestamp for comparisons
Dangerous comparisons:
- block.timestamp < poolStartTime (contracts/distribution/UShareRewardPool.sol#95)
- _lastRewardTime == 0 (contracts/distribution/UShareRewardPool.sol#97)
- _lastRewardTime < poolStartTime (contracts/distribution/UShareRewardPool.sol#100)
- _lastRewardTime == 0 || _lastRewardTime < block.timestamp (contracts/distribution/
UShareRewardPool.sol#106)
- _isStarted = (_lastRewardTime <= poolStartTime) || (_lastRewardTime <=
block.timestamp) (contracts/distribution/UShareRewardPool.sol#110-112)
UShareRewardPool.getGeneratedReward(uint256,uint256) (contracts/distribution/
UShareRewardPool.sol#138-149) uses timestamp for comparisons
Dangerous comparisons:
- _fromTime >= _toTime (contracts/distribution/UShareRewardPool.sol#139)
- _toTime >= poolEndTime (contracts/distribution/UShareRewardPool.sol#140)
- _fromTime >= poolEndTime (contracts/distribution/UShareRewardPool.sol#141)
- _fromTime <= poolStartTime (contracts/distribution/UShareRewardPool.sol#142)
- _toTime <= poolStartTime (contracts/distribution/UShareRewardPool.sol#145)
- _fromTime <= poolStartTime (contracts/distribution/UShareRewardPool.sol#146)
UShareRewardPool.pendingShare(uint256,address) (contracts/distribution/
UShareRewardPool.sol#152-163) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- block.timestamp > pool.lastRewardTime && tokenSupply != 0 (contracts/distribution/
UShareRewardPool.sol#157)
UShareRewardPool.massUpdatePools() (contracts/distribution/
UShareRewardPool.sol#166-171) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- pid < length (contracts/distribution/UShareRewardPool.sol#168)
UShareRewardPool.updatePool(uint256) (contracts/distribution/
UShareRewardPool.sol#174-194) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- block.timestamp <= pool.lastRewardTime (contracts/distribution/
UShareRewardPool.sol#176)
UShareRewardPool.governanceRecoverUnsupported(IERC20,uint256,address) (contracts/
distribution/UShareRewardPool.sol#264-275) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- block.timestamp < poolEndTime + 7776000 (contracts/distribution/
UShareRewardPool.sol#265)
UniteGenesisRewardPool.constructor(address,address,uint256) (contracts/distribution/
UniteGenesisRewardPool.sol#68-79) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- require(bool,string)(block.timestamp < _poolStartTime,late) (contracts/distribution/
UniteGenesisRewardPool.sol#73)
UniteGenesisRewardPool.checkPoolDuplicate(IERC20) (contracts/distribution/
UniteGenesisRewardPool.sol#86-91) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- pid < length (contracts/distribution/UniteGenesisRewardPool.sol#88)
⊠- require(bool,string)(poolInfo[pid].token != _token,UniteGenesisPool: existing pool?)
(contracts/distribution/UniteGenesisRewardPool.sol#89)
UniteGenesisRewardPool.add(uint256,IERC20,bool,uint256) (contracts/distribution/
UniteGenesisRewardPool.sol#94-124) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- block.timestamp < poolStartTime (contracts/distribution/
UniteGenesisRewardPool.sol#104)
⊠- _lastRewardTime == 0 (contracts/distribution/UniteGenesisRewardPool.sol#106)
⊠- _lastRewardTime < poolStartTime (contracts/distribution/
UniteGenesisRewardPool.sol#109)
⊠- _lastRewardTime == 0 || _lastRewardTime < block.timestamp (contracts/distribution/
UniteGenesisRewardPool.sol#115)
⊠- _isStarted = (_lastRewardTime <= poolStartTime) || (_lastRewardTime <=
block.timestamp) (contracts/distribution/UniteGenesisRewardPool.sol#119)
```

```
UniteGenesisRewardPool.getGeneratedReward(uint256,uint256) (contracts/distribution/
UniteGenesisRewardPool.sol#137-148) uses timestamp for comparisons
	Dangerous comparisons:
	- _fromTime >= _toTime (contracts/distribution/UniteGenesisRewardPool.sol#138)
	- _toTime >= poolEndTime (contracts/distribution/UniteGenesisRewardPool.sol#139)
	- _toTime <= poolStartTime (contracts/distribution/UniteGenesisRewardPool.sol#144)
UniteGenesisRewardPool.pendingUNITE(uint256,address) (contracts/distribution/
UniteGenesisRewardPool.sol#151-162) uses timestamp for comparisons
	Dangerous comparisons:
	- block.timestamp > pool.lastRewardTime && tokenSupply != 0 (contracts/distribution/
UniteGenesisRewardPool.sol#156)
UniteGenesisRewardPool.massUpdatePools() (contracts/distribution/
UniteGenesisRewardPool.sol#165-170) uses timestamp for comparisons
	Dangerous comparisons:
	- pid < length (contracts/distribution/UniteGenesisRewardPool.sol#167)
UniteGenesisRewardPool.updatePool(uint256) (contracts/distribution/
UniteGenesisRewardPool.sol#173-193) uses timestamp for comparisons
	Dangerous comparisons:
	- block.timestamp <= pool.lastRewardTime (contracts/distribution/
UniteGenesisRewardPool.sol#175)
UniteGenesisRewardPool.governanceRecoverUnsupported(IERC20,uint256,address) (contracts/
distribution/UniteGenesisRewardPool.sol#267-282) uses timestamp for comparisons
	Dangerous comparisons:
	- block.timestamp < poolEndTime + 7776000 (contracts/distribution/
UniteGenesisRewardPool.sol#272)
UniteRewardPool.constructor(address,uint256) (contracts/distribution/
UniteRewardPool.sol#60-74) uses timestamp for comparisons
	Dangerous comparisons:
	- require(bool,string)(block.timestamp < _poolStartTime,late) (contracts/distribution/
UniteRewardPool.sol#61)
UniteRewardPool.checkPoolDuplicate(IERC20) (contracts/distribution/
UniteRewardPool.sol#81-86) uses timestamp for comparisons
	Dangerous comparisons:
	- pid < length (contracts/distribution/UniteRewardPool.sol#83)
	- require(bool,string)(poolInfo[pid].token != _token,UniteRewardPool: existing pool?)
(contracts/distribution/UniteRewardPool.sol#84)
UniteRewardPool.add(uint256,IERC20,bool,uint256) (contracts/distribution/
UniteRewardPool.sol#89-119) uses timestamp for comparisons
	Dangerous comparisons:
	- block.timestamp < poolStartTime (contracts/distribution/UniteRewardPool.sol#99)
	- _lastRewardTime == 0 (contracts/distribution/UniteRewardPool.sol#101)
```

```
⊠- _lastRewardTime < poolStartTime (contracts/distribution/UniteRewardPool.sol#104)
⊠- _lastRewardTime == 0 || _lastRewardTime < block.timestamp (contracts/distribution/
UniteRewardPool.sol#110)
⊠- _isStarted = (_lastRewardTime <= poolStartTime) || (_lastRewardTime <=
block.timestamp) (contracts/distribution/UniteRewardPool.sol#114)
UniteRewardPool.getGeneratedReward(uint256,uint256) (contracts/distribution/
UniteRewardPool.sol#132-153) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- _toTime >= epochEndTimes[epochId - 1] (contracts/distribution/
UniteRewardPool.sol#134)
UniteRewardPool.pendingUNITE(uint256,address) (contracts/distribution/
UniteRewardPool.sol#156-167) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- block.timestamp > pool.lastRewardTime && tokenSupply != 0 (contracts/distribution/
UniteRewardPool.sol#161)
UniteRewardPool.massUpdatePools() (contracts/distribution/UniteRewardPool.sol#170-175)
uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- pid < length (contracts/distribution/UniteRewardPool.sol#172)
UniteRewardPool.updatePool(uint256) (contracts/distribution/
UniteRewardPool.sol#178-198) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- block.timestamp <= pool.lastRewardTime (contracts/distribution/
UniteRewardPool.sol#180)
UniteRewardPool.governanceRecoverUnsupported(IERC20,uint256,address) (contracts/
distribution/UniteRewardPool.sol#268-283) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- block.timestamp < epochEndTimes[1] + 2592000 (contracts/distribution/
UniteRewardPool.sol#273)
UniswapV2OracleLibrary.currentCumulativePrices(address) (contracts/lib/
UniswapV2OracleLibrary.sol#18-42) uses timestamp for comparisons
⊠Dangerous comparisons:
⊠- blockTimestampLast != blockTimestamp (contracts/lib/UniswapV2OracleLibrary.sol#33)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp
INFO:Detectors:
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/
Address.sol#26-35) uses assembly
⊠- INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/
utils/Address.sol#171-188) uses assembly
```

```
- INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#180-183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
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']
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/GSN/Context.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/Math.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20Burnable.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#3)
- >=0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/ReentrancyGuard.sol#3)
- 0.6.12 (contracts/Boardroom.sol#3)
- 0.6.12 (contracts/DummyToken.sol#3)
- 0.6.12 (contracts/Oracle.sol#3)
- ^0.6.0 (contracts/SimpleERCFund.sol#3)
- 0.6.12 (contracts/TaxOffice.sol#3)
- 0.6.12 (contracts/TaxOfficeV2.sol#3)
- 0.6.12 (contracts/TaxOracle.sol#3)
- 0.6.12 (contracts/Timelock.sol#3)
- 0.6.12 (contracts/Treasury.sol#3)
- 0.6.12 (contracts/UBond.sol#3)
- 0.6.12 (contracts/UShare.sol#3)
- 0.6.12 (contracts/Unite.sol#3)
- 0.6.12 (contracts/distribution/UShareRewardPool.sol#3)
- 0.6.12 (contracts/distribution/UniteGenesisRewardPool.sol#3)
- 0.6.12 (contracts/distribution/UniteRewardPool.sol#3)
- ^0.6.0 (contracts/interfaces/IBasisAsset.sol#3)
- 0.6.12 (contracts/interfaces/IBoardroom.sol#3)
- 0.6.12 (contracts/interfaces/IERC20.sol#3)
- 0.6.12 (contracts/interfaces/IOracle.sol#3)
- ^0.6.0 (contracts/interfaces/ISimpleERCFund.sol#3)
- 0.6.12 (contracts/interfaces/ITaxable.sol#3)
- 0.6.12 (contracts/interfaces/ITreasury.sol#3)
- ^0.6.0 (contracts/interfaces/IUniswapV2Pair.sol#3)
- 0.6.12 (contracts/interfaces/IUniswapV2Router.sol#3)
- 0.6.12 (contracts/interfaces/IWrappedEth.sol#3)
```

```
	- ^0.6.0 (contracts/lib/Babylonian.sol#3)
	- ^0.6.0 (contracts/lib/FixedPoint.sol#3)
	- 0.6.12 (contracts/lib/SafeMath8.sol#3)
	- ^0.6.0 (contracts/lib/UniswapV2Library.sol#3)
	- ^0.6.0 (contracts/lib/UniswapV2OracleLibrary.sol#3)
	- 0.6.12 (contracts/owner/Operator.sol#3)
	- 0.6.12 (contracts/utils/ContractGuard.sol#3)
	- ^0.6.0 (contracts/utils/Epoch.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-
pragma-directives-are-used
INFO:Detectors:
Different versions of Solidity is used:
	- Version used: ['0.6.12', '^0.6.0']
	- 0.6.12 (contracts/Distributor.sol#3)
	- ^0.6.0 (contracts/interfaces/IDistributor.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-
pragma-directives-are-used
INFO:Detectors:
Treasury._calculateMaxSupplyExpansionPercent(uint256) (contracts/Treasury.sol#491-499)
has costly operations inside a loop:
	- maxSupplyExpansionPercent = maxExpansionTiers[tierId] (contracts/Treasury.sol#494)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-
operations-inside-a-loop
```

Guard