RUGDOG

SMART CONTRACT SECURITY AUDIT

Final report

Plan: Complex

Unite Finance

April 2022

rugdog.net

the@rugdog.net

CONTENTS

INTRODUCTION

The report has been prepared for Unite Finance.

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

CONTRACTS CHECKED

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

AUDIT PROCESS

We perform our audit according to the following procedure:

- An audit with the assistance of available automated tools
- An audit performed manually by auditors
- Comparing the original Tomb Finance code to the code of the reviewed project

OVERVIEW OF RELEVANCE LEVELS

High relevance

Issues of high relevance may lead to losses of users' funds as well as changes of ownership of a contract or possible issues with the logic of the contract.
High-relevance issues require immediate attention and a response from the team.

Medium relevance

While issues of medium relevance don't pose as high a risk as the high-relevance ones do, they can be just as easily exploited by the team or a malicious user, causing a contract failure and damaging the project's reputation in the process. Usually, these issues can be fixed if the contract is redeployed.
Medium-relevance issues require a response from the team.

Low relevance

Issues of low relevance don't pose high risks since they can't cause damage to the functionality of the contract. However, it's still recommended to consider fixing them.

ISSUES

High relevance issues

No high relevance issues found

Medium relevance issues

No medium relevance issues found

Low relevance issues

No low relevance issues found

CONCLUSION

The project has been compared to the initial Tomb Finance code.
From that, it's become clear that the implementation of the Token has been changed, as well as Treasury and Ushare contracts.

The issue existing in the Token contract in the Tomb code, is not present in the reviewed contract, since it doesn't include the transfers with taxes.

team1Fund addresses have been added to both Treasury and UShare contracts. Their main role is to receive funds similar to what the devFund does in Tomb Finance.

State variables communityFundRewardRate, team1FundRewardRate, and devFundRewardRate are set in the UShare contract by calling an external setAllocations function.

No issues have been found in the contracts.

DISCLAIMER

The published report is subject to the RugDog terms and conditions (which include without limitation: description of services, confidentiality, disclaimer, and limitation of liability), that are stated in the Services Agreement along with the scope of services, terms, and conditions provided to the Company in connection with the Agreement. This report compiled and published under the terms set in the Agreement will only be used by the Company under the circumstances, permitted by the terms and conditions specified in the Agreement.
This report will not be transmitted, disclosed, or referenced by any person for any purpose without written consent from RugDog.

This report is by no means an "endorsement" or an "attempt to discredit" any particular project or team, nor it should be viewed as such.
This report is not, nor should be viewed as an indication of the economical value or value of any other kinds of the "product" or "asset" described in the report or created by the same or any other team.
This report does not provide or reflect any warranty or guarantee regarding the reviewed project or technology. With this report, RugDog does not provide any indication of the technologies proprietors, business, business model, or legal compliance.

This report should not be used in any way as financial or investment advice, it should not be used for making investments or other decisions.
This report performed and published by RugDog, represents a detailed assessment with an intent to help the customer improve their code and its quality as well as reduce the risks presented by Decentralized Finance and accompanying technology.

STATIC CODE ANALYSIS RESULT

```
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.
```

```
- 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(1e16) (contracts/
distribution/UShareRewardPool.sol#213)
Reentrancy in UniteGenesisRewardPool.deposit(uint256,uint256) (contracts/distribution/
UniteGenesisRewardPool.sol#196-216):
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/
contract/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(1e16) (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(1e16) (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#216-235):
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#216-235):
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/
```

```
- 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(1e16) (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
```

```
9):
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(1e16) (contracts/
distribution/UniteRewardPool.sol#237)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#reentrancyvulnerabilities
INFO:Detectors:
TaxOficeV2.addLiquidityTaxFree(address,uint256,uint256,uint256,uint256) (contracts/
TaxOficeV2.sol#84-129) ignores return value by
IERC20(kitty).transferFrom(msg.sender,address(this),amtUnite) (contracts/
TaxOficeV2.sol#101)
TaxOficeV2.addLiquidityTaxFree(address,uint256,uint256,uint256,uint256) (contracts/
TaxOficeV2.sol#84-129) ignores return value by
IERC20(token).transferFrom(msg.sender,address(this),amtToken) (contracts/
TaxOficeV2.sol#102)
TaxOficeV2.addLiquidityTaxFree(address,uint256,uint256,uint256,uint256) (contracts/
TaxOficeV2.sol#84-129) ignores return value by
IERC20(kitty).transfer(msg.sender,amtUnite.sub(resultAmtUnite)) (contracts/
TaxOficeV2.sol#123)
TaxOficeV2.addLiquidityTaxFree(address,uint256,uint256,uint256
TaxOficeV2.sol#84-129) ignores return value by
IERC20(token).transfer(msg.sender,amtToken.sub(resultAmtToken)) (contracts/
TaxOficeV2.sol#126)
TaxOficeV2.addLiquidityETHTaxFree(uint256,uint256,uint256) (contracts/
TaxOficeV2.sol#131-168) ignores return value by
IERC20(kitty).transferFrom(msg.sender,address(this),amtUnite) (contracts/
TaxOficeV2.sol#147)
TaxOficeV2.addLiquidityETHTaxFree(uint256,uint256,uint256) (contracts/
TaxOficeV2.sol#131-168) ignores return value by
IERC20(kitty).transfer(msg.sender,amtUnite.sub(resultAmtUnite)) (contracts/
TaxOficeV2.sol#165)
TaxOficeV2.taxFreeTransferFrom(address,address,uint256) (contracts/
TaxOficeV2.sol#178-167) ignores return value by
IERC20(kitty).transferFrom(_sender,_recipient,_amt) (contracts/TaxOficeV2.sol#165)
```

```
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#uncheckedtransfer
```

```
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#
```

```
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-
eventsaccess-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#316-321) 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-
eventsarithmetic
INFO:Detectors:
Boardroom.setOperator(address)._operator (contracts/Boardroom.sol#138) lacks a
zerocheck 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
zerocheck on :
- boardroom = _boardroom (contracts/Treasury.sol#280)
Treasury.setUniteOracle(address)._kittyOracle (contracts/Treasury.sol#283) lacks a
zerocheck on :
- kittyOracle = _kittyOracle (contracts/Treasury.sol#284)
UShare.setTreasuryFund(address)._communityFund (contracts/UShare.sol#67) lacks a
zerocheck 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-
zeroaddress-validation
INFO:Detectors:
Modifer Migrations.restricted() (contracts/Migrations.sol#13-15) does not always
execute _; or revertReference: https://github.com/crytic/slither/wiki/
DetectorDocumentation#incorrect-modifer
INFO:Detectors:
Distributor.distribute() (contracts/Distributor.sol#14-16) has external calls inside a
loop: distributors[i].distribute() (contracts/Distributor.sol#16)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-
insidea-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#predeclaration-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#reentrancyvulnerabilities-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):
```

RUGDOG

WOOF!

rugdog.net

the@rugdog.net