www.cyberunit.tech

Smart Contract Audit – Tosdis Finance

Date: April 16,2021

Report for: Tosdis Finance

By: CyberUnit.Tech

www.cyberunit.tech

This document may contain confidential information about IT systems and the customer's intellectual property and information about potential vulnerabilities and exploitation methods.

The report contains confidential information. This information can be used internally by the customer. The customer can release the information after fixing all vulnerabilities.

Document

Name	Tosdis Finance
Link	https://kovan.etherscan.io/address/0xf1e4ede634da80646facfe6b4c176dc0f05c95b6#code
Date	15/04/21

www.cyberunit.tech

Table of contents

Introduction

This report presents the Customer` s smart contract's security assessment findings and its code review conducted between April 6 – April 15 2021

Scope

The scope of the project is Tosdis smart contract, which can be found by the link below:

https://kovan.etherscan.io/address/0xf1e4ede634da80646facfe6b4c176dc0f05c95b6#code

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the widely known vulnerabilities that are considered (the complete list includes them but does not limit them):

- Reentrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Compiler version not fixed
- Unchecked external call – Unchecked math
- Unsafe type inference
- Implicit visibility level

Executive Summary

Our team performed an analysis of code functionality, manual audit, and automated checks with Slither and remix IDE (see Appendix B pic 1-2). All issues found during automated analysis reviewed have been manually, and application vulnerabilities are

presented in the Audit overview section. A general overview is presented in the AS-IS section, and you can find all found issues in the Audit overview section.

We found two low and one medium issue in a smart contract.

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also significantly do not impact smart contract execution, e.g., public access to crucial functions.
Medium	Medium-level vulnerabilities are essential to fix; however, they can't lead to tokens loss.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc., code snippets that can't significantly impact execution.
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations, and info statements can't affect smart contract execution and can be ignored.

AS-IS overview

IDOMaster contract consists of the next smart contracts:

1. Address.sol, Context.sol, ERC20Burnable.sol, Ownable.sol, Pausable.sol, SafeERC20.sol, SafeMath.sol, Whitelist.sol, ERC20.sol, ReentrancyGuard.sol
2. IERC20.sol, IidoMaster.sol, IStakingPool.sol, IUniswapV2Pair.sol contracts – interfaces
3. IDOMaster.sol, IDOPool.sol

Contracts from point 1 were compared to original "Openzeppelin" templates no logic differences were found. They are considered secure.

Contracts from point 2 are Helpers Interfaces that include header files.

Contracts from point 3 The IDOmaster classes implementing the IDOmaster protocol will be detailed in the report.

IDOMaster AS-IS overview

IDOMaster.sol contract inherits the class – Ownable

IDOMaster contract init function:

SetFreeToken function was called with the following parameters:

- uint256(_newFeeToken)

setFeeAmount function was called with following parameters:

- uint256(_newFeeAmount)

SetFreeWallet function was called with the following parameters:

- address(payable _newFeeWallet)

SetBurnPercent function was called with the following parameters:

- uint256(_newBurnPercent)
- uint256(_newDivider)

setFeeFundsPercent function was called with the following parameters:

- uint256(_feeFundsPercent)

setFeeFundsPercent function was called with the following parameters:

- IStakingPool(_disStakingPool)
- IStakingPool(_lpDisStakingPool)
- IUniswapV2Pair(_lpUniswapV2Pair)
- bool(_disReserve0)

setFeeFundsPercent function was called with the following parameters:

- uint256(_vipDisAmount)
- uint256(_vipPercent)
- uint256(_holdersDisAmount)
- uint256(_holdersPercent)
- uint256(_publicDisAmount)
- uint256(_publicPercent)

createIDO function was called with the following parameters:

- uint256(_tokenPrice)
- ERC20(_rewardToken)
- uint256(_startTimestamp)
- uint256(_finishTimestamp)
- uint256(_startClaimTimestamp)
- uint256(_minEthPayment)
- uint256(_maxEthPayment)

- uint256(_maxDistributedTokenAmount)
- bool(_hasWhitelisting)
- bool(_enableTierSystem)

isContract function was called with the following parameters:

- address(_addr)

getMaxEthPayment function was called with the following parameters:

- address(user)
- uint256(maxEthPayment)

getFullDisBalance function was called with the following parameters:

- address(user)

getFeeWallet function was called without parameters.

IDOMaster Audit overview

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

1. Use of strict equalities that an attacker can easily manipulate. (see Appendix A pic. 1 for evidence)

Low

2. Different versions of Solidity are used in Version used: ['0.7.3', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0'] (see Appendix A pic. 2 for evidence)

IDOPool AS-IS overview

IDOPool.sol contract inherits the class Ownable, Pausable, Whitelist, ReentrancyGuard.

IDOPool.sol contract init functions:

getTokenAmount function was called with the following parameters:

- address(ethAmount)

claimFor function was called with the following parameters:

- address[](memory _addresses)

proccessClaim function was called with the following parameters:

- address(_receiver)

pay function was called without parameters.

claim function was called without parameters.

withdrawFunds function was called without parameters.

withdrawNotSoldTokens function was called without parameters.

ReentrancyGuard.sol contract init functions:

nonReentrant function was called without parameters.

IDOPool Audit overview

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

1. Different versions of Solidity are used in Version used: ['0.7.3', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0'] (see Appendix A pic. 3 for evidence)

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, a high-level description of functionality was presented in the report's As-is overview section.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

The overall quality of the reviewed contracts is secured. Security engineers found two low and one medium vulnerability, which couldn't have any significant security impact. And we can provide best practice for used:

The creation of IDOpool in the body of the contract is costly in our realities. We suggest using a short proxy contract that refers to IDOpool, while the constructor can be moved into the contract's init function where you want to place a check to prevent double execution. We would also recommend using create2 to initialize the proxy contract.

Disclaimers

Disclaimer

The smart contracts given for audit had been analyzed following the best industry practices at the date of this report, concerning: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on the security of the code. It can also not be considered a sufficient assessment regarding the code's utility and safety, bug-free status, or any other contract statements. While we have done our best to conduct the analysis and produce this report, it is important to note that you should not rely on this report only – we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, programming language, and other software related to the smart contract can have their vulnerabilities leading to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.

Appendix A. Evidences

Pic 1. Dangerous strict equalities:

```
160                 _minEthPayment,
161                 _maxEthPayment,
162                 _maxDistributedTokenAmount,
163                 _hasWhitelisting,
164                 _enableTierSystem
165             );
166
167         idoPool.transferOwnership(msg.sender);
168
169         _rewardToken.safeTransferFrom(
170             msg.sender,
171             address(idoPool),
172             _maxDistributedTokenAmount
173         );
174
175         require( _rewardToken.balanceOf(address(idoPool)) == _maxDistributedTokenAmount,  "Unsupported token");
176
```

Pic 2. Different versions:

```
Different versions of Solidity is used in :
        - Version used: ['0.7.3', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0']
        - >=0.6.0<0.8.0 (ERC20.sol#3)
        - 0.7.3 (IDOMaster.sol#2)
        - 0.7.3 (IDOPool.sol#1)
        - >=0.6.0<0.8.0 (ReentrancyGuard.sol#3)
        - >=0.6.0<0.8.0 (interfaces/IERC20.sol#3)
        - 0.7.3 (interfaces/IStakingPool.sol#1)
        - 0.7.3 (interfaces/IUniswapV2Pair.sol#3)
        - 0.7.3 (interfaces/IidoMaster.sol#1)
        - >=0.6.2<0.8.0 (lib/Address.sol#3)
        - >=0.6.0<0.8.0 (lib/Context.sol#3)
        - >=0.6.0<0.8.0 (lib/ERC20Burnable.sol#3)
        - >=0.6.0<0.8.0 (lib/Ownable.sol#3)
        - 0.7.3 (lib/Pausable.sol#1)
        - >=0.6.0<0.8.0 (lib/SafeERC20.sol#3)
        - >=0.6.0<0.8.0 (lib/SafeMath.sol#3)
        - 0.7.3 (lib/Whitelist.sol#1)
```

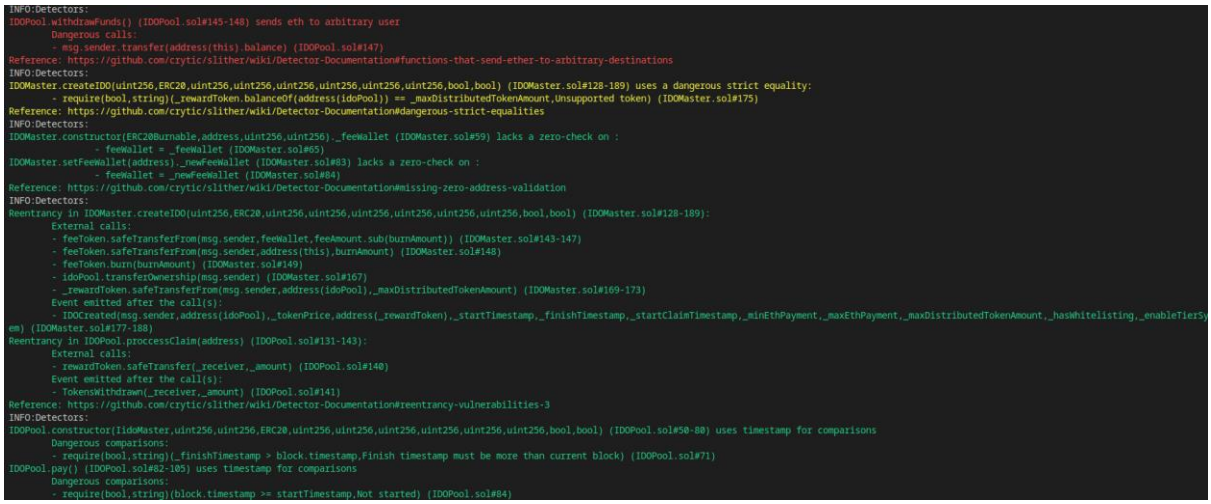
Pic 3. Different versions:

```
Different versions of Solidity is used in :
        - Version used: ['0.7.3', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0']
        - >=0.6.0<0.8.0 (ERC20.sol#3)
        - 0.7.3 (IDOPool.sol#1)
        - >=0.6.0<0.8.0 (ReentrancyGuard.sol#3)
        - >=0.6.0<0.8.0 (interfaces/IERC20.sol#3)
        - 0.7.3 (interfaces/IidoMaster.sol#1)
        - >=0.6.2<0.8.0 (lib/Address.sol#3)
        - >=0.6.0<0.8.0 (lib/Context.sol#3)
        - >=0.6.0<0.8.0 (lib/Ownable.sol#3)
        - 0.7.3 (lib/Pausable.sol#1)
        - >=0.6.0<0.8.0 (lib/SafeERC20.sol#3)
        - >=0.6.0<0.8.0 (lib/SafeMath.sol#3)
        - 0.7.3 (lib/Whitelist.sol#1)
```

www.cyberunit.tech

Appendix B. Automated tools reports

Pic 1. IDOMaster Slither automated report:



```
INFO:Detectors:
IDOPool.withdrawFunds() (IDOPool.sol#145-148) sends eth to arbitrary user
        Dangerous calls:
        - msg.sender.transfer(address(this).balance) (IDOPool.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
IDOMaster.createIDO(uint256,ERC20,uint256,uint256,uint256,uint256,uint256,uint256,bool,bool) (IDOMaster.sol#128-189) uses a dangerous strict equality:
        - require(bool,string)(_rewardToken.balanceOf(address(idoPool)) == _maxDistributedTokenAmount,Unsupported token) (IDOMaster.sol#175)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
IDOMaster.constructor(ERC20Burnable,address,uint256,uint256)._feeWallet (IDOMaster.sol#59) lacks a zero-check on :
                - feeWallet = _feeWallet (IDOMaster.sol#65)
IDOMaster.setFeeWallet(address)._newFeeWallet (IDOMaster.sol#83) lacks a zero-check on :
                - feeWallet = _newFeeWallet (IDOMaster.sol#84)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in IDOMaster.createIDO(uint256,ERC20,uint256,uint256,uint256,uint256,uint256,uint256,bool,bool) (IDOMaster.sol#128-189):
        External calls:
        - feeToken.safeTransferFrom(msg.sender,feeWallet,feeAmount.sub(burnAmount)) (IDOMaster.sol#143-147)
        - feeToken.safeTransferFrom(msg.sender,address(this),burnAmount) (IDOMaster.sol#148)
        - feeToken.burn(burnAmount) (IDOMaster.sol#149)
        - idoPool.transferOwnership(msg.sender) (IDOMaster.sol#167)
        - _rewardToken.safeTransferFrom(msg.sender,address(idoPool),_maxDistributedTokenAmount) (IDOMaster.sol#169-173)
        Event emitted after the call(s):
        - IDOCreated(msg.sender,address(idoPool),_tokenPrice,address(_rewardToken),_startTimestamp,_finishTimestamp,_startClaimTimestamp,_minEthPayment,_maxEthPayment,_maxDistributedTokenAmount,_hasWhitelisting,_enableTierSy
em) (IDOMaster.sol#177-188)
Reentrancy in IDOPool.proccessClaim(address) (IDOPool.sol#131-143):
        External calls:
        - rewardToken.safeTransfer(_receiver,_amount) (IDOPool.sol#140)
        Event emitted after the call(s):
        - TokensWithdrawn(_receiver,_amount) (IDOPool.sol#141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
IDOPool.constructor(IidoMaster,uint256,uint256,ERC20,uint256,uint256,uint256,uint256,uint256,bool,bool) (IDOPool.sol#50-80) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_finishTimestamp > block.timestamp,Finish timestamp must be more than current block) (IDOPool.sol#71)
IDOPool.pay() (IDOPool.sol#82-105) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= startTimestamp,Not started) (IDOPool.sol#84)
```

```
        - require(bool,string)(block.timestamp < finishTimestamp,Ended) (IDOPool.sol#85)
IDOPool.proccessClaim(address) (IDOPool.sol#131-143) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > startClaimTimestamp,Distribution not started) (IDOPool.sol#134)
IDOPool.withdrawNotSoldTokens() (IDOPool.sol#151-155) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > finishTimestamp,Withdraw allowed after finish time) (IDOPool.sol#152)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
IDOMaster.isContract(address) (IDOMaster.sol#191-197) uses assembly
        - INLINE ASM (IDOMaster.sol#193-195)
Address.isContract(address) (lib/Address.sol#26-35) uses assembly
        - INLINE ASM (lib/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (lib/Address.sol#147-164) uses assembly
        - INLINE ASM (lib/Address.sol#156-159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used in :
        - Version used: ['0.7.3', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0']
        - >=0.6.0<0.8.0 (ERC20.sol#3)
        - 0.7.3 (IDOMaster.sol#2)
        - 0.7.3 (IDOPool.sol#1)
        - >=0.6.0<0.8.0 (ReentrancyGuard.sol#3)
        - >=0.6.0<0.8.0 (interfaces/IERC20.sol#3)
        - 0.7.3 (interfaces/IStakingPool.sol#1)
        - 0.7.3 (interfaces/IUniswapV2Pair.sol#3)
        - 0.7.3 (interfaces/IidoMaster.sol#1)
        - >=0.6.2<0.8.0 (lib/Address.sol#3)
        - >=0.6.0<0.8.0 (lib/Context.sol#3)
        - >=0.6.0<0.8.0 (lib/ERC20Burnable.sol#3)
        - >=0.6.0<0.8.0 (lib/Ownable.sol#3)
        - 0.7.3 (lib/Pausable.sol#1)
        - >=0.6.0<0.8.0 (lib/SafeERC20.sol#3)
        - >=0.6.0<0.8.0 (lib/SafeMath.sol#3)
        - 0.7.3 (lib/Whitelist.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Pragma version>=0.6.0<0.8.0 (ERC20.sol#3) is too complex
Pragma version0.7.3 (IDOMaster.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.7.3 (IDOPool.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version>=0.6.0<0.8.0 (ReentrancyGuard.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (interfaces/IERC20.sol#3) is too complex
```

```
Pragma version0.7.3 (interfaces/IStakingPool.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.7.3 (interfaces/IUniswapV2Pair.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.7.3 (interfaces/IidoMaster.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version>=0.6.2<0.8.0 (lib/Address.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (lib/Context.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (lib/ERC20Burnable.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (lib/Ownable.sol#3) is too complex
Pragma version0.7.3 (lib/Pausable.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version>=0.6.0<0.8.0 (lib/SafeERC20.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (lib/SafeMath.sol#3) is too complex
Pragma version0.7.3 (lib/Whitelist.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
solc-0.7.3 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (lib/Address.sol#53-59):
        - (success) = recipient.call{value: amount}() (lib/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (lib/Address.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (lib/Address.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (lib/Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (lib/Address.sol#143)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
IDOMaster (IDOMaster.sol#13-243) should inherit from IidoMaster (interfaces/IidoMaster.sol#4-13)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance
INFO:Detectors:
Parameter IDOMaster.setFeeToken(address)._newFeeToken (IDOMaster.sol#70) is not in mixedCase
Parameter IDOMaster.setFeeAmount(uint256)._newFeeAmount (IDOMaster.sol#77) is not in mixedCase
Parameter IDOMaster.setFeeWallet(address)._newFeeWallet (IDOMaster.sol#83) is not in mixedCase
Parameter IDOMaster.setBurnPercent(uint256,uint256)._newBurnPercent (IDOMaster.sol#89) is not in mixedCase
Parameter IDOMaster.setBurnPercent(uint256,uint256)._newDivider (IDOMaster.sol#89) is not in mixedCase
Parameter IDOMaster.setFeeFundsPercent(uint256)._feeFundsPercent (IDOMaster.sol#101) is not in mixedCase
Parameter IDOMaster.setTierPools(IStakingPool,IStakingPool,IUniswapV2Pair,bool)._disStakingPool (IDOMaster.sol#110) is not in mixedCase
Parameter IDOMaster.setTierPools(IStakingPool,IStakingPool,IUniswapV2Pair,bool)._lpDisStakingPool (IDOMaster.sol#110) is not in mixedCase
Parameter IDOMaster.setTierPools(IStakingPool,IStakingPool,IUniswapV2Pair,bool)._lpUniswapV2Pair (IDOMaster.sol#110) is not in mixedCase
Parameter IDOMaster.setTierPools(IStakingPool,IStakingPool,IUniswapV2Pair,bool)._disReserve0 (IDOMaster.sol#110) is not in mixedCase
Parameter IDOMaster.setTier(uint256,uint256,uint256,uint256,uint256,uint256)._vipDisAmount (IDOMaster.sol#117) is not in mixedCase
Parameter IDOMaster.setTier(uint256,uint256,uint256,uint256,uint256,uint256)._vipPercent (IDOMaster.sol#117) is not in mixedCase
Parameter IDOMaster.setTier(uint256,uint256,uint256,uint256,uint256,uint256)._holdersDisAmount (IDOMaster.sol#117) is not in mixedCase
Parameter IDOMaster.setTier(uint256,uint256,uint256,uint256,uint256,uint256)._holdersPercent (IDOMaster.sol#117) is not in mixedCase
```

```
Parameter IDOMaster.setTier(uint256,uint256,uint256,uint256,uint256,uint256)._publicDisAmount (IDOMaster.sol#117) is not in mixedCase
Parameter IDOMaster.setTier(uint256,uint256,uint256,uint256,uint256,uint256)._publicPercent (IDOMaster.sol#117) is not in mixedCase
Parameter IDOMaster.createIDO(uint256,ERC20,uint256,uint256,uint256,uint256,uint256,uint256,bool,bool)._tokenPrice (IDOMaster.sol#129) is not in mixedCase
Parameter IDOMaster.createIDO(uint256,ERC20,uint256,uint256,uint256,uint256,uint256,uint256,bool,bool)._rewardToken (IDOMaster.sol#130) is not in mixedCase
Parameter IDOMaster.createIDO(uint256,ERC20,uint256,uint256,uint256,uint256,uint256,uint256,bool,bool)._startTimestamp (IDOMaster.sol#131) is not in mixedCase
Parameter IDOMaster.createIDO(uint256,ERC20,uint256,uint256,uint256,uint256,uint256,uint256,bool,bool)._finishTimestamp (IDOMaster.sol#132) is not in mixedCase
Parameter IDOMaster.createIDO(uint256,ERC20,uint256,uint256,uint256,uint256,uint256,uint256,bool,bool)._startClaimTimestamp (IDOMaster.sol#133) is not in mixedCase
Parameter IDOMaster.createIDO(uint256,ERC20,uint256,uint256,uint256,uint256,uint256,uint256,bool,bool)._minEthPayment (IDOMaster.sol#134) is not in mixedCase
Parameter IDOMaster.createIDO(uint256,ERC20,uint256,uint256,uint256,uint256,uint256,uint256,bool,bool)._maxEthPayment (IDOMaster.sol#135) is not in mixedCase
Parameter IDOMaster.createIDO(uint256,ERC20,uint256,uint256,uint256,uint256,uint256,uint256,bool,bool)._maxDistributedTokenAmount (IDOMaster.sol#136) is not in mixedCase
Parameter IDOMaster.createIDO(uint256,ERC20,uint256,uint256,uint256,uint256,uint256,uint256,bool,bool)._hasWhitelisting (IDOMaster.sol#137) is not in mixedCase
Parameter IDOMaster.createIDO(uint256,ERC20,uint256,uint256,uint256,uint256,uint256,uint256,bool,bool)._enableTierSystem (IDOMaster.sol#138) is not in mixedCase
Parameter IDOMaster.isContract(address)._addr (IDOMaster.sol#191) is not in mixedCase
Parameter IDOPool.claimFor(address[])._addresses (IDOPool.sol#118) is not in mixedCase
Parameter IDOPool.proccessClaim(address)._receiver (IDOPool.sol#132) is not in mixedCase
Parameter Whitelist.add(address[])._addresses (lib/Whitelist.sol#27) is not in mixedCase
Parameter Whitelist.remove(address)._address (lib/Whitelist.sol#36) is not in mixedCase
Parameter Whitelist.isWhitelisted(address)._address (lib/Whitelist.sol#49) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (lib/Context.sol#21)" inContext (lib/Context.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (ERC20.sol#64-66)
symbol() should be declared external:
        - ERC20.symbol() (ERC20.sol#72-74)
decimals() should be declared external:
        - ERC20.decimals() (ERC20.sol#89-91)
totalSupply() should be declared external:
        - ERC20.totalSupply() (ERC20.sol#96-98)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (ERC20.sol#103-105)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (ERC20.sol#115-118)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (ERC20.sol#134-137)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (ERC20.sol#152-156)
```

Pic 2. IDOPool Slither automated report:

```
INFO:Detectors:
IDOPool.withdrawFunds() (IDOPool.sol#145-148) sends eth to arbitrary user
        Dangerous calls:
        - msg.sender.transfer(address(this).balance) (IDOPool.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in IDOPool.proccessClaim(address) (IDOPool.sol#131-143):
        External calls:
        - rewardToken.safeTransfer(_receiver,_amount) (IDOPool.sol#140)
        Event emitted after the call(s):
        - TokensWithdrawn(_receiver,_amount) (IDOPool.sol#141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
IDOPool.constructor(IidoMaster,uint256,uint256,ERC20,uint256,uint256,uint256,uint256,uint256,uint256,bool,bool) (IDOPool.sol#50-80) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_finishTimestamp > block.timestamp,Finish timestamp must be more than current block) (IDOPool.sol#71)
IDOPool.pay() (IDOPool.sol#82-105) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= startTimestamp,Not started) (IDOPool.sol#84)
        - require(bool,string)(block.timestamp < finishTimestamp,Ended) (IDOPool.sol#85)
IDOPool.proccessClaim(address) (IDOPool.sol#131-143) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > startClaimTimestamp,Distribution not started) (IDOPool.sol#134)
IDOPool.withdrawNotSoldTokens() (IDOPool.sol#151-155) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > finishTimestamp,Withdraw allowed after finish time) (IDOPool.sol#152)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (lib/Address.sol#26-35) uses assembly
        - INLINE ASM (lib/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (lib/Address.sol#147-164) uses assembly
        - INLINE ASM (lib/Address.sol#156-159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used in :
        - Version used: ['0.7.3', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0']
        - >=0.6.0<0.8.0 (ERC20.sol#3)
        - 0.7.3 (IDOPool.sol#1)
        - >=0.6.0<0.8.0 (ReentrancyGuard.sol#3)
        - >=0.6.0<0.8.0 (interfaces/IERC20.sol#3)
        - 0.7.3 (interfaces/IidoMaster.sol#1)
        - >=0.6.2<0.8.0 (lib/Address.sol#3)
```

```
        - >=0.6.0<0.8.0 (lib/Context.sol#3)
        - >=0.6.0<0.8.0 (lib/Ownable.sol#3)
        - 0.7.3 (lib/Pausable.sol#1)
        - >=0.6.0<0.8.0 (lib/SafeERC20.sol#3)
        - >=0.6.0<0.8.0 (lib/SafeMath.sol#3)
        - 0.7.3 (lib/Whitelist.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Pragma version>=0.6.0<0.8.0 (ERC20.sol#3) is too complex
Pragma version0.7.3 (IDOPool.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version>=0.6.0<0.8.0 (ReentrancyGuard.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (interfaces/IERC20.sol#3) is too complex
Pragma version0.7.3 (interfaces/IidoMaster.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version>=0.6.2<0.8.0 (lib/Address.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (lib/Context.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (lib/Ownable.sol#3) is too complex
Pragma version0.7.3 (lib/Pausable.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version>=0.6.0<0.8.0 (lib/SafeERC20.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (lib/SafeMath.sol#3) is too complex
Pragma version0.7.3 (lib/Whitelist.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
solc-0.7.3 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (lib/Address.sol#53-59):
        - (success) = recipient.call{value: amount}() (lib/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (lib/Address.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (lib/Address.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (lib/Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (lib/Address.sol#143)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter IDOPool.claimFor(address[])._addresses (IDOPool.sol#118) is not in mixedCase
Parameter IDOPool.proccessClaim(address)._receiver (IDOPool.sol#132) is not in mixedCase
Parameter Whitelist.add(address[])._addresses (lib/Whitelist.sol#27) is not in mixedCase
```

```
Parameter Whitelist.remove(address)._address (lib/Whitelist.sol#36) is not in mixedCase
Parameter Whitelist.isWhitelisted(address)._address (lib/Whitelist.sol#49) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (lib/Context.sol#21)" inContext (lib/Context.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (ERC20.sol#64-66)
symbol() should be declared external:
        - ERC20.symbol() (ERC20.sol#72-74)
decimals() should be declared external:
        - ERC20.decimals() (ERC20.sol#89-91)
totalSupply() should be declared external:
        - ERC20.totalSupply() (ERC20.sol#96-98)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (ERC20.sol#103-105)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (ERC20.sol#115-118)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (ERC20.sol#123-125)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (ERC20.sol#134-137)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (ERC20.sol#152-156)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (ERC20.sol#170-173)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (ERC20.sol#189-192)
owner() should be declared external:
        - Ownable.owner() (lib/Ownable.sol#35-37)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (lib/Ownable.sol#54-57)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (lib/Ownable.sol#63-67)
pause() should be declared external:
        - Pausable.pause() (lib/Pausable.sol#36-39)
unpause() should be declared external:
        - Pausable.unpause() (lib/Pausable.sol#44-47)
add(address[]) should be declared external:
        - Whitelist.add(address[]) (lib/Whitelist.sol#27-34)
```

Appendix C. Automated tools GAS usage reports

```
[
  '0xDC97F226d29D73b667CD541d5E347d2bF807DC36',
  '0xa6ce54356b1B0948CC27d5b32b9fe5c83aa01dE6',
  '0x3AD68874e1075184E60Aebfc67b61EB013B91407',
  '0xb2bf860b9Fe144Bc29b983Ad6e429333432aCF85',
  '0x737EDF4A7da70f62f7e396EB201C2900Dd28e997',
  '0x4f0A3Be9d3F96d19cb8209Babdaf2c5C773a7BF4',
  '0x7904BD25646792Ef8133b12092eA773A3fc680Bd',
  '0xd00C65b975e192cAF33AE4464acabCEc05d2bd40',
  '0x2D1f41481e890E5D9ECdC65AB390fB616Bef60a9',
  '0xB625Dd8F44b61163E0Ec04A584e9fCfE2499864F'
]

  Contract: Infrastructure
    around:
      ✓ testFarming: success way

·---------------------------------------|---------------------------|-------------|-----------------------------·
|  Solc version: 0.7.3+commit.9bfce1f6  ·  Optimizer enabled: true  ·  Runs: 200  ·  Block limit: 6718946 gas  |
·.....................................|...........................|.............|.............................·
|  Methods                              ·              129 gwei/gas               ·       2332.58 usd/eth       |
·.....................|.....................|.............|.............|.............|...............|..............·
|  Contract            ·  Method             ·  Min        ·  Max        ·  Avg        ·  # calls      ·  usd (avg)  |
·.....................|.....................|.............|.............|.............|...............|..............·
|  IDOMaster           ·  createIDO          ·          -  ·          -  ·    1785744  ·            1  ·     537.34  |
·.....................|.....................|.............|.............|.............|...............|..............·
|  Migrations          ·  setCompleted       ·          -  ·          -  ·      27288  ·            1  ·       8.21  |
·.....................|.....................|.............|.............|.............|...............|..............·
|  MockERC20           ·  approve            ·          -  ·          -  ·      43964  ·            1  ·      13.23  |
·.....................|.....................|.............|.............|.............|...............|..............·
|  MockERC20           ·  mint               ·          -  ·          -  ·      65687  ·            1  ·      19.77  |
·.....................|.....................|.............|.............|.............|...............|..............·
|  Deployments                                 ·                                         ·  % of limit  ·             |
·.....................|.....................|.............|.............|.............|...............|..............·
|  IDOMaster                                   ·          -  ·          -  ·    3183103  ·       47.4 %  ·     957.80  |
·.....................|.....................|.............|.............|.............|...............|..............·
|  Migrations                                  ·          -  ·          -  ·     165046  ·        2.5 %  ·      49.66  |
·.....................|.....................|.............|.............|.............|...............|..............·
|  MockERC20                                   ·          -  ·          -  ·     533039  ·        7.9 %  ·     160.39  |
·---------------------------------------|-------------|-------------|-------------|---------------|--------------·

  1 passing (8s)
```