SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment

19. January, 2022

For

Disclaimer

SolidProof.io reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc’...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing 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. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof’s position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	18. January 2022	• Layout project • Automated- /Manual-Security Testing • Summary
	19. January	Finished

Network

Binance Smart Chain (BEP20)

Website

https://talkaboat.online/

Telegram

https://t.me/talkaboat

Twitter

https://twitter.com/talkaboat

Description

The Aboat ecosystem maps a metaverse in which content creators and the community are enabled to interact with each other in completely new ways. The focus is on online entertainment such as podcasts, - and video streaming, as well as gaming. The latter will be the bridge to the metaverse and will allow for a whole new sense of play and belonging.

Project Engagement

During the 13th of January 2022, **Talkaboat Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo

Contract Link

v1.0

- Github
    - https://github.com/Talkaboat/smart-contracts/tree/master/contracts
    - Commit: 3be279a37d6b5d208c33ac7ffad9fd48b9fafdf0

Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon aspossible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
    i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
    ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.

2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:

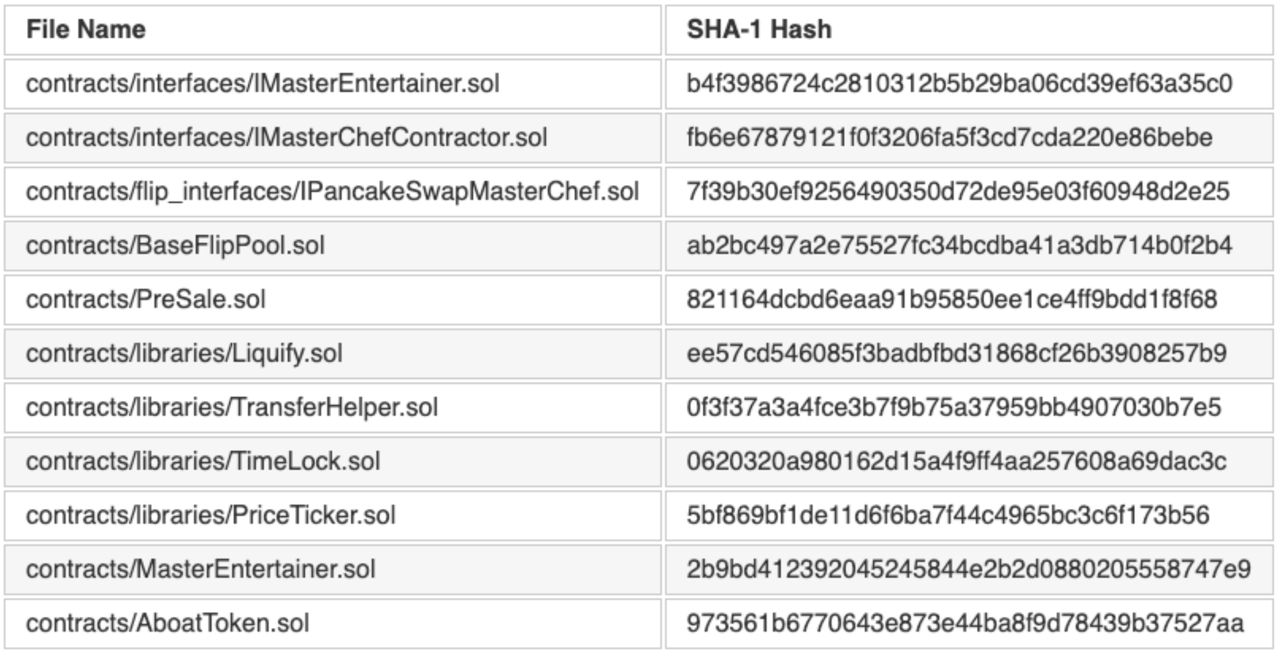
Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	7
@openzeppelin/contracts/security/ReentrancyGuard.sol	3
@openzeppelin/contracts/token/ERC20/ERC20.sol	6
@openzeppelin/contracts/token/ERC20/IERC20.sol	8
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	3
@openzeppelin/contracts/utils/Address.sol	7
@openzeppelin/contracts/utils/math/SafeMath.sol	7
@uniswap/v2-core/contracts/interfaces/IUniswapV2Factory.sol	2
@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol	5
@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol	5

Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

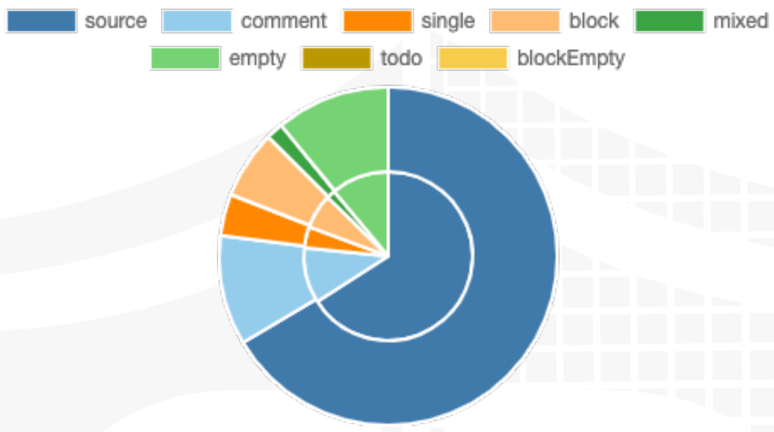


File Name	SHA-1 Hash
contracts/interfaces/IMasterEntertainer.sol	b4f3986724c2810312b5b29ba06cd39ef63a35c0
contracts/interfaces/IMasterChefContractor.sol	fb6e67879121f0f3206fa5f3cd7cda220e86bebe
contracts/flip_interfaces/IPancakeSwapMasterChef.sol	7f39b30ef9256490350d72de95e03f60948d2e25
contracts/BaseFlipPool.sol	ab2bc497a2e75527fc34bcdba41a3db714b0f2b4
contracts/PreSale.sol	821164dcbd6eaa91b95850ee1ce4ff9bdd1f8f68
contracts/libraries/Liquify.sol	ee57cd546085f3badbfbd31868cf26b3908257b9
contracts/libraries/TransferHelper.sol	0f3f37a3a4fce3b7f9b75a37959bb4907030b7e5
contracts/libraries/TimeLock.sol	0620320a980162d15a4f9ff4aa257608a69dac3c
contracts/libraries/PriceTicker.sol	5bf869bf1de11d6f6ba7f44c4965bc3c6f173b56
contracts/MasterEntertainer.sol	2b9bd412392045245844e2b2d0880205558747e9
contracts/AboatToken.sol	973561b6770643e873e44ba8f9d78439b37527aa

Metrics

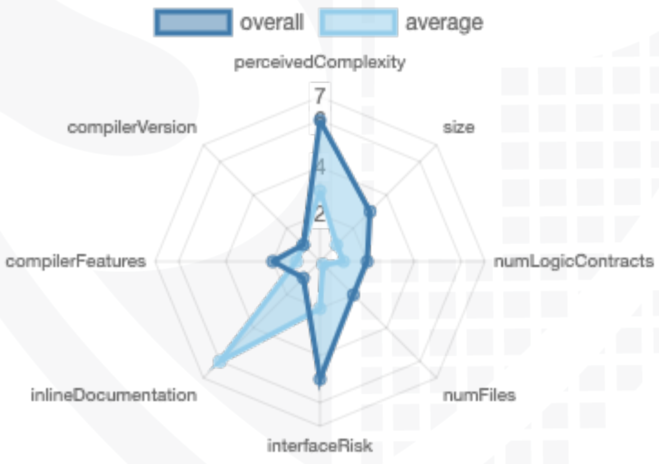
Source Lines

v1.0



Risk Level

v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	4	1	3	3

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

Version	Public	Payable
1.0	113	4

Version	External	Internal	Private	Pure	View
1.0	30	128	5	1	32

State Variables

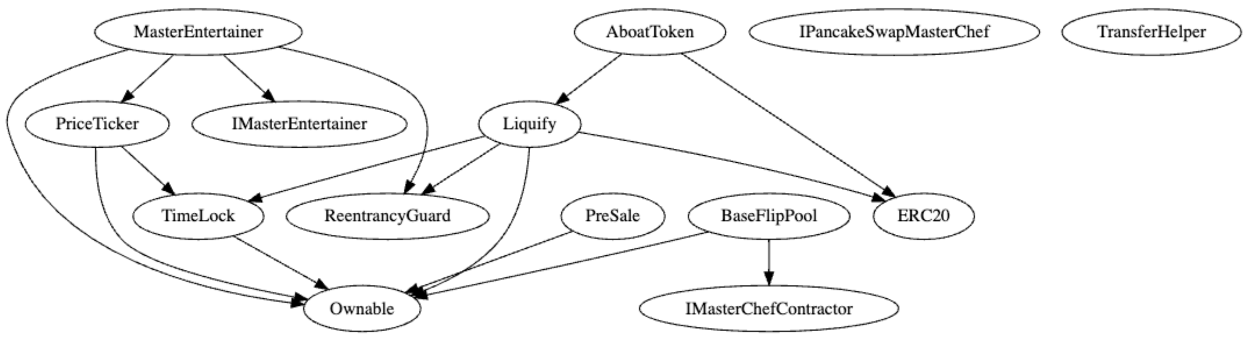
Version	Total	Public
1.0	78	71

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`^0.8.7`		`yes`		

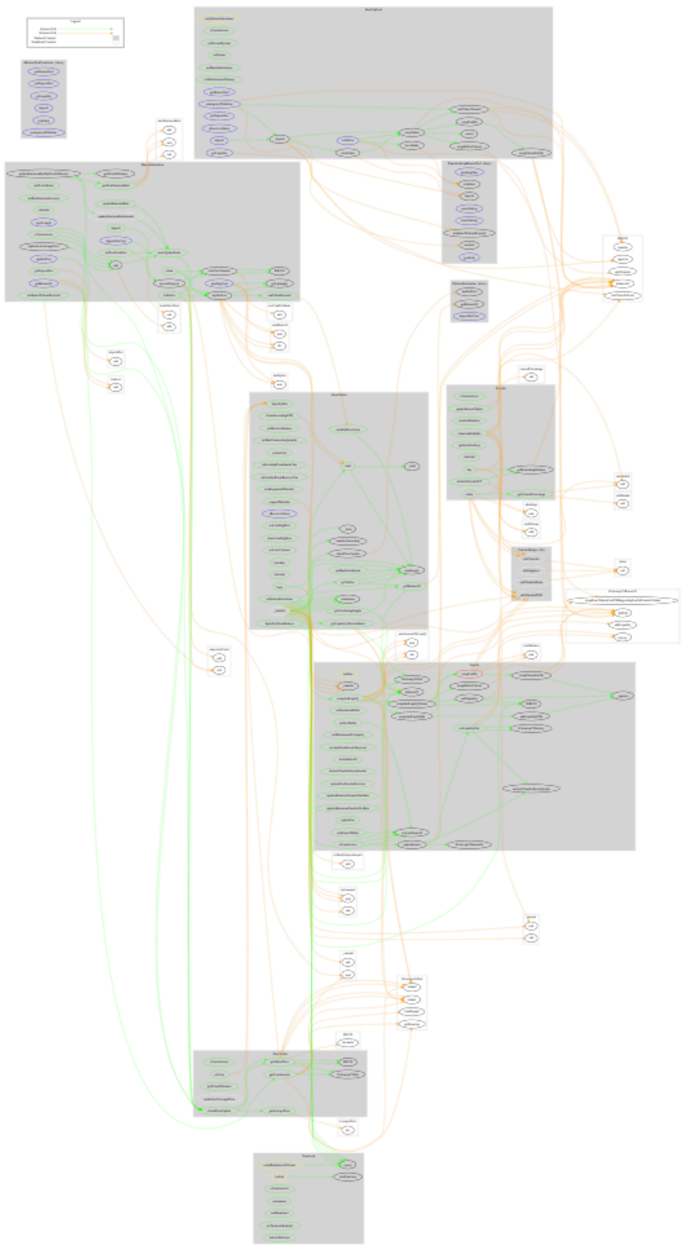
Inheritance Graph

v1.0



Call Graph

v1.0



Scope of Work/Verify Claims

The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

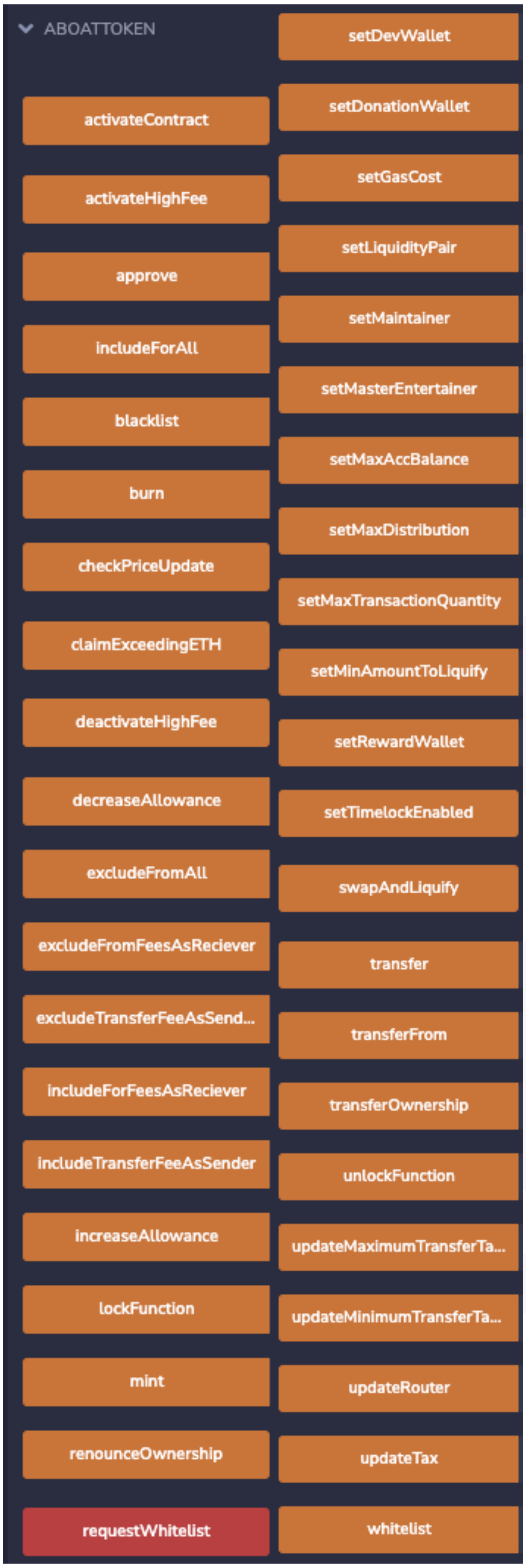
We will verify the following claims:

1. Correct implementation of Token standard
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. External Approve function is restricted
6. Overall checkup (Smart Contract Security)

Correct implementation of Token standard

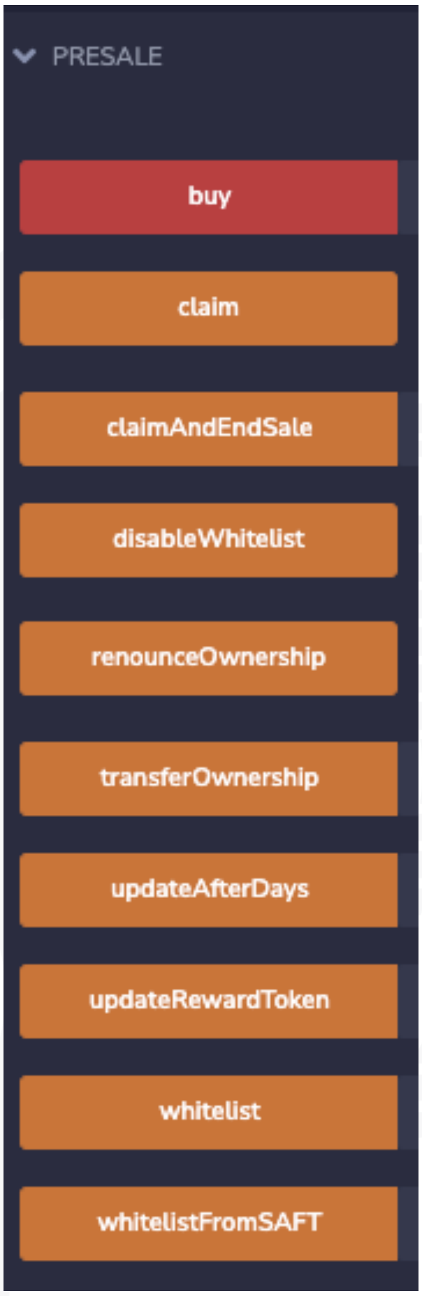
Function	Description	Exist	Tested	Verified
TotalSupply	provides information about the total token supply	✓	✓	✓
BalanceOf	provides account balance of the owner's account	✓	✓	✓
Transfer	executes transfers of a specified number of tokens to a specified address	✓	✓	✓
TransferFrom	executes transfers of a specified number of tokens from a specified address	✓	✓	✓
Approve	allow a spender to withdraw a set number of tokens from a specified account	✓	✓	✓
Allowance	returns a set number of tokens from a spender to the owner	✓	✓	✓

Write functions of contract v1.0



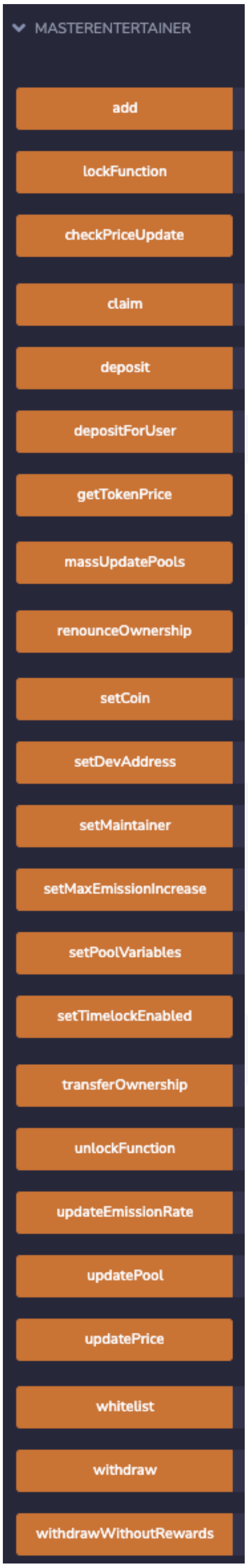
ABOATTOKEN

activateContract
activateHighFee
approve
includeForAll
blacklist
burn
checkPriceUpdate
claimExceedingETH
deactivateHighFee
decreaseAllowance
excludeFromAll
excludeFromFeesAsReciever
excludeTransferFeeAsSend...
includeForFeesAsReciever
includeTransferFeeAsSender
increaseAllowance
lockFunction
mint
renounceOwnership
requestWhitelist
setDevWallet
setDonationWallet
setGasCost
setLiquidityPair
setMaintainer
setMasterEntertainer
setMaxAccBalance
setMaxDistribution
setMaxTransactionQuantity
setMinAmountToLiquify
setRewardWallet
setTimelockEnabled
swapAndLiquify
transfer
transferFrom
transferOwnership
unlockFunction
updateMaximumTransferTa...
updateMinimumTransferTa...
updateRouter
updateTax
whitelist



PRESALE

buy
claim
claimAndEndSale
disableWhitelist
renounceOwnership
transferOwnership
updateAfterDays
updateRewardToken
whitelist
whitelistFromSAFT


MASTERENTERTAINER

add

lockFunction

checkPriceUpdate

claim

deposit

depositForUser

getTokenPrice

massUpdatePools

renounceOwnership

setCoin

setDevAddress

setMaintainer

setMaxEmissionIncrease

setPoolVariables

setTimelockEnabled

transferOwnership

unlockFunction

updateEmissionRate

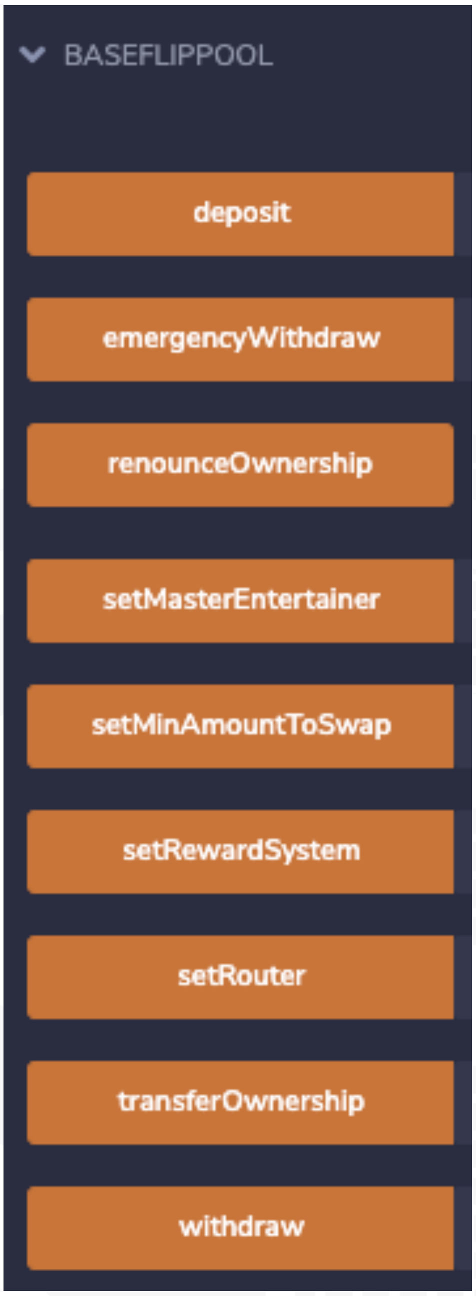
updatePool

updatePrice

whitelist

withdraw

withdrawWithoutRewards


BASEFLIPPOOL

deposit

emergencyWithdraw

renounceOwnership

setMasterEntertainer

setMinAmountToSwap

setRewardSystem

setRouter

transferOwnership

withdraw

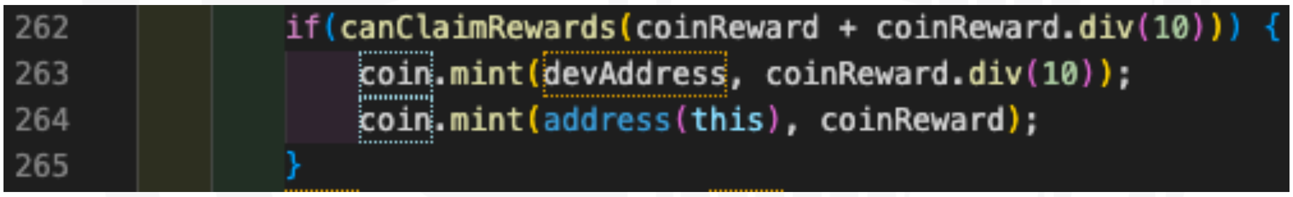
Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✗
Max / Total Supply	600.000.000.000		

Comments:

v1.0

- Deployer can mint new tokens
    - As long as total supply + minting amount is lower equal than maxDistributor
        - maxDistributor can be set without any limitations
- MasterEntertainer can mint new tokens



```
        if(canClaimRewards(coinReward + coinReward.div(10))) {
            coin.mint(devAddress, coinReward.div(10));
            coin.mint(address(this), coinReward);
        }
```

Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock	✓	✓	✗
Deployer cannot burn	✓	✓	✗

Comments:

v1.0

- OnlyMaintainerOrOwner can burn tokens
- OnlyMaintainerOrOwner can lock user funds by
    - Blacklist address with blacklist function
    - Setting maxAccBalance to 0
    - Setting maxTxQuantity to 0

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	–	–	–

External approve function is restricted

Name	Exist	Tested	Status
External Approve cannot be called without restriction	–	–	–

Overall checkup (Smart Contract Security)

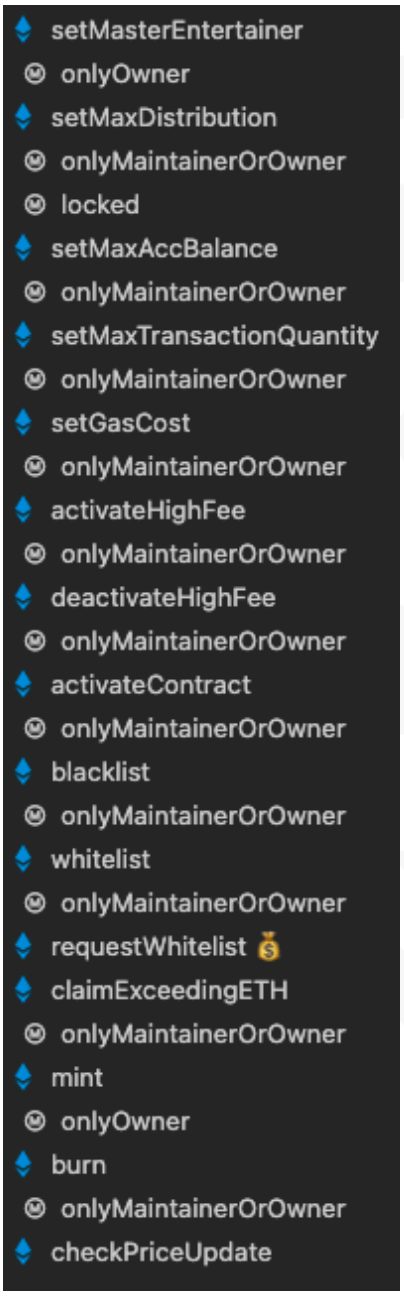
Tested	Verified
✓	✓

Legend

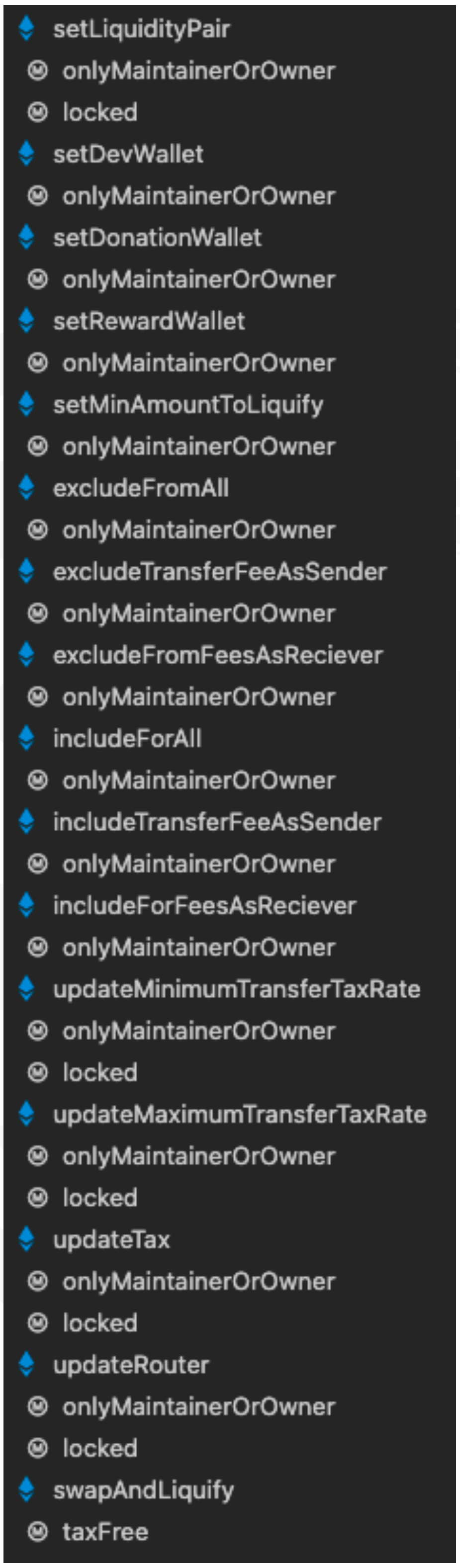
Attribute	Symbol
Verfified / Checked	✓
Partly Verified	⚑
Unverified / Not checked	✗
Not available	–

Modifiers and public functions

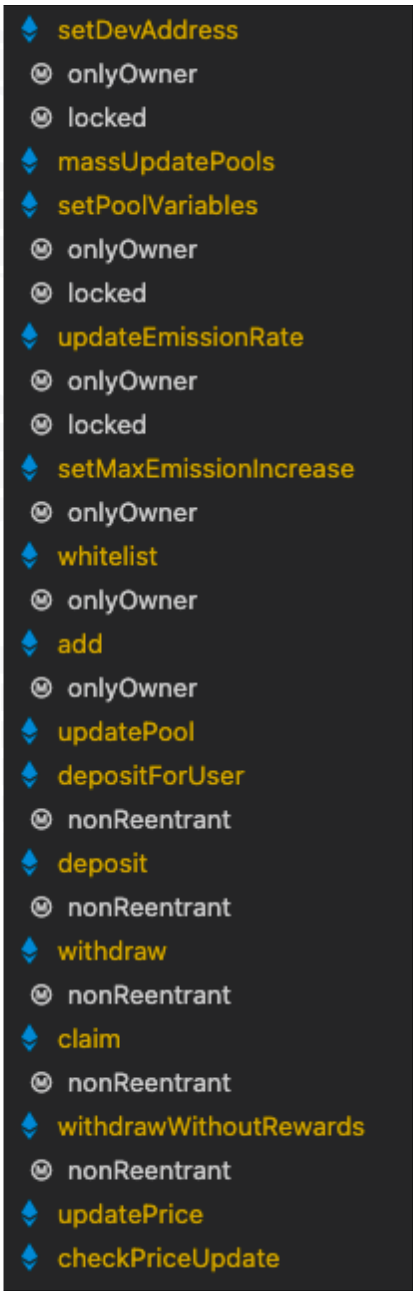
AboatToken


```
setMasterEntertainer
  onlyOwner
setMaxDistribution
  onlyMaintainerOrOwner
  locked
setMaxAccBalance
  onlyMaintainerOrOwner
setMaxTransactionQuantity
  onlyMaintainerOrOwner
setGasCost
  onlyMaintainerOrOwner
activateHighFee
  onlyMaintainerOrOwner
deactivateHighFee
  onlyMaintainerOrOwner
activateContract
  onlyMaintainerOrOwner
blacklist
  onlyMaintainerOrOwner
whitelist
  onlyMaintainerOrOwner
requestWhitelist 💰
claimExceedingETH
  onlyMaintainerOrOwner
mint
  onlyOwner
burn
  onlyMaintainerOrOwner
checkPriceUpdate
```

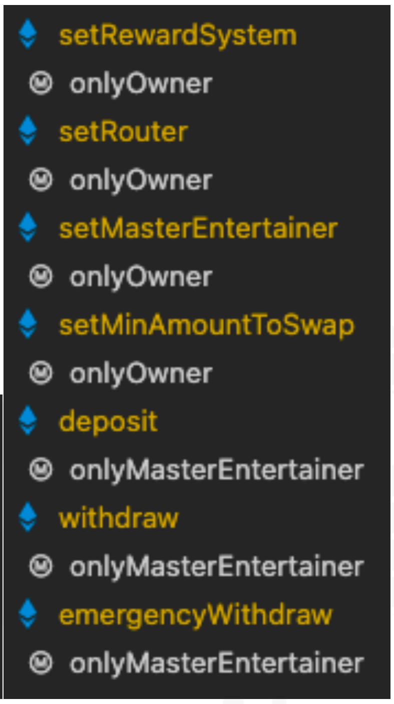
Liquify


```
setLiquidityPair
  onlyMaintainerOrOwner
  locked
setDevWallet
  onlyMaintainerOrOwner
setDonationWallet
  onlyMaintainerOrOwner
setRewardWallet
  onlyMaintainerOrOwner
setMinAmountToLiquify
  onlyMaintainerOrOwner
excludeFromAll
  onlyMaintainerOrOwner
excludeTransferFeeAsSender
  onlyMaintainerOrOwner
excludeFromFeesAsReciever
  onlyMaintainerOrOwner
includeForAll
  onlyMaintainerOrOwner
includeTransferFeeAsSender
  onlyMaintainerOrOwner
includeForFeesAsReciever
  onlyMaintainerOrOwner
updateMinimumTransferTaxRate
  onlyMaintainerOrOwner
  locked
updateMaximumTransferTaxRate
  onlyMaintainerOrOwner
  locked
updateTax
  onlyMaintainerOrOwner
  locked
updateRouter
  onlyMaintainerOrOwner
  locked
swapAndLiquify
  taxFree
```

MasterEntertainer

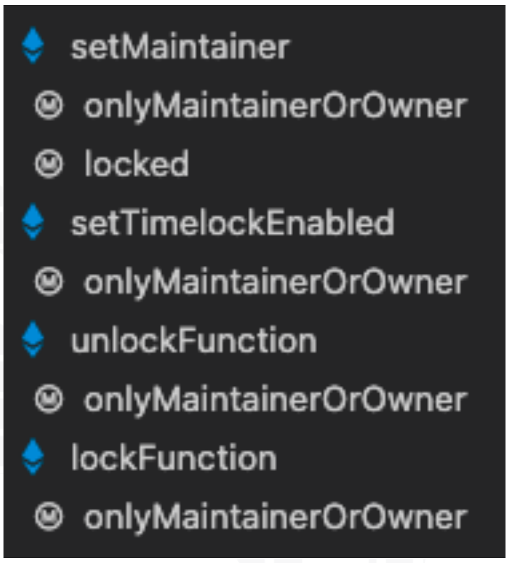

```
setDevAddress
  onlyOwner
  locked
massUpdatePools
setPoolVariables
  onlyOwner
  locked
updateEmissionRate
  onlyOwner
  locked
setMaxEmissionIncrease
  onlyOwner
whitelist
  onlyOwner
add
  onlyOwner
updatePool
depositForUser
  nonReentrant
deposit
  nonReentrant
withdraw
  nonReentrant
claim
  nonReentrant
withdrawWithoutRewards
  nonReentrant
updatePrice
checkPriceUpdate
```

BaseFlipPool



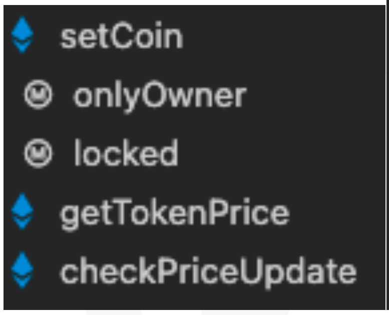
setRewardSystem
onlyOwner
setRouter
onlyOwner
setMasterEntertainer
onlyOwner
setMinAmountToSwap
onlyOwner
deposit
onlyMasterEntertainer
withdraw
onlyMasterEntertainer
emergencyWithdraw
onlyMasterEntertainer

TimeLock



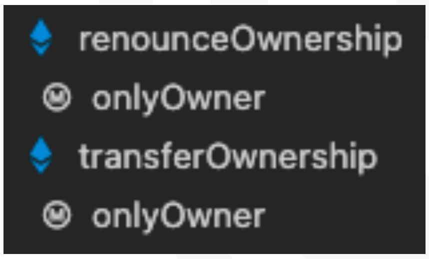
setMaintainer
onlyMaintainerOrOwner
locked
setTimelockEnabled
onlyMaintainerOrOwner
unlockFunction
onlyMaintainerOrOwner
lockFunction
onlyMaintainerOrOwner

PriceTicker



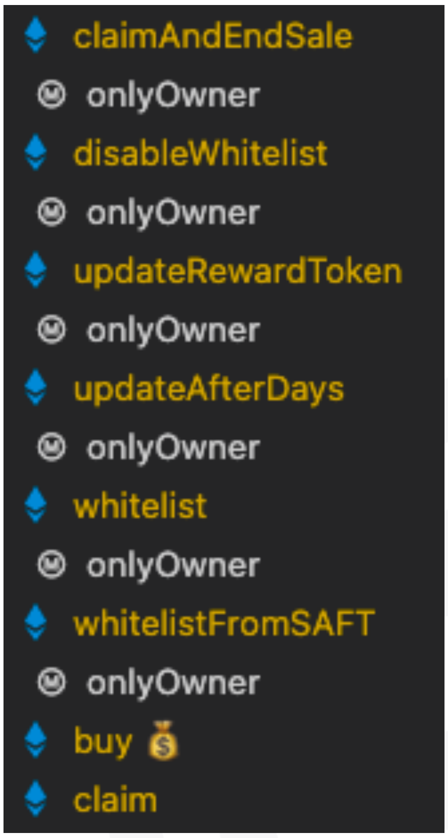
setCoin
onlyOwner
locked
getTokenPrice
checkPriceUpdate

Ownable



renounceOwnership
onlyOwner
transferOwnership
onlyOwner

Presale



claimAndEndSale
onlyOwner
disableWhitelist
onlyOwner
updateRewardToken
onlyOwner
updateAfterDays
onlyOwner
whitelist
onlyOwner
whitelistFromSAFT
onlyOwner
buy 💰
claim

Comments

- Deployer can set following state variables without any limitations
    - AboatToken
        - maxAccBalance
        - maxTxQuantity
        - gasCost
    - Liquify
        - _minAmountToLiquify
    - MasterEntertainer
        - maxEmissionIncrease
    - BaseFlipPool
        - minAmountToSwap
    - Presale
        - afterDays

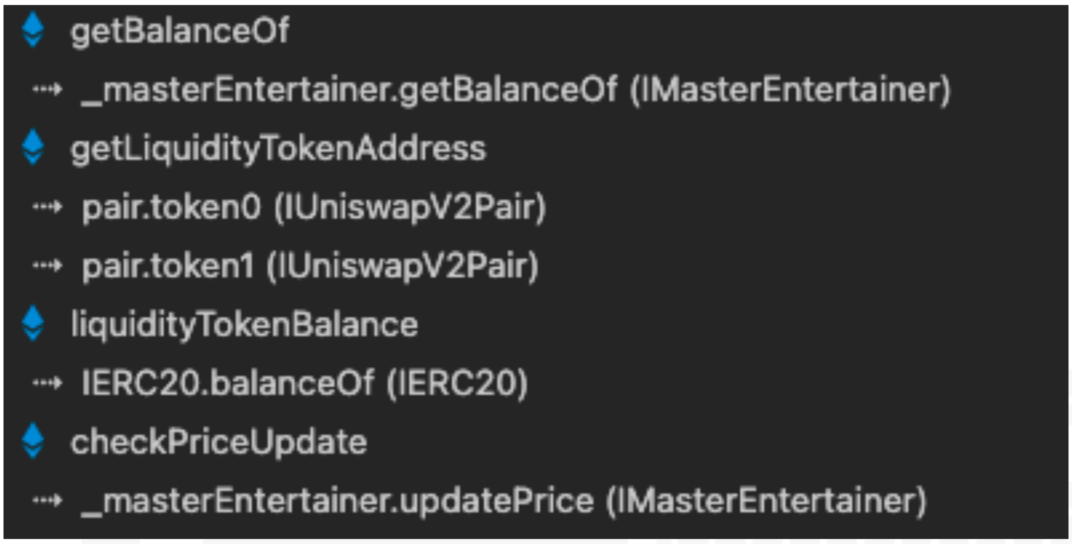
- Deployer can enable/disable following state variables
    - AboatToken
        - isHighFeeActive
        - blacklisted
    - Liquify
        - _excludedFromFeesAsSender
        - _excludedFromFeesAsReciever
    - MasterEntertainer
        - whitelisted
    - TimeLock
        - isLockEnabled
    - Presale
        - whitelisted

- depositUser can only be called from whitelisted addresses in MasterEntertainer
- Everybody can call updatePrice function in MasterEntertainer

Please check if an OnlyOwner or similar restrictive modifier has been forgotten.

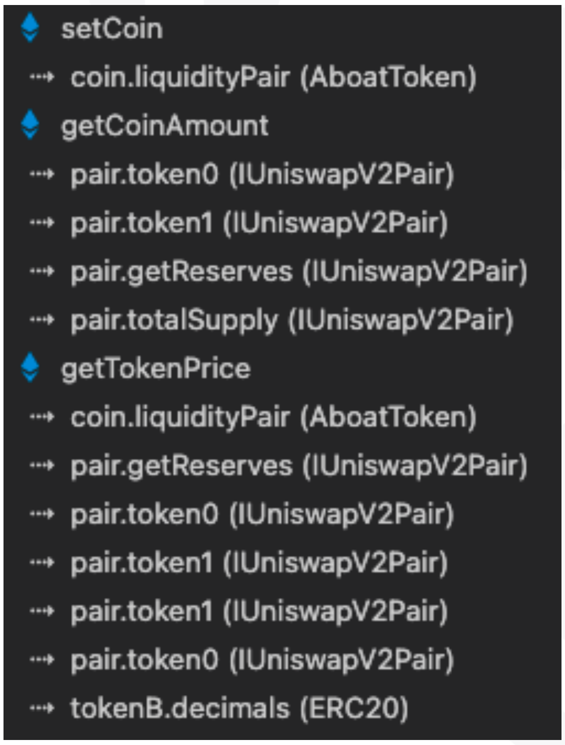
External Calls from functions

AboatToken



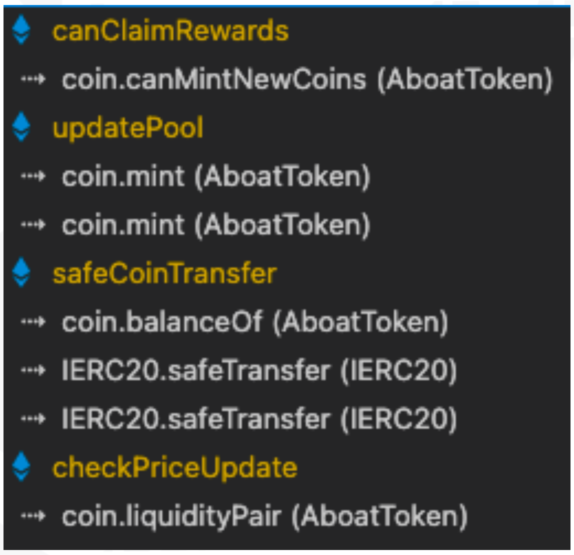
```
getBalanceOf
  ⇢ _masterEntertainer.getBalanceOf (IMasterEntertainer)
getLiquidityTokenAddress
  ⇢ pair.token0 (IUniswapV2Pair)
  ⇢ pair.token1 (IUniswapV2Pair)
liquidityTokenBalance
  ⇢ IERC20.balanceOf (IERC20)
checkPriceUpdate
  ⇢ _masterEntertainer.updatePrice (IMasterEntertainer)
```

PriceTicker



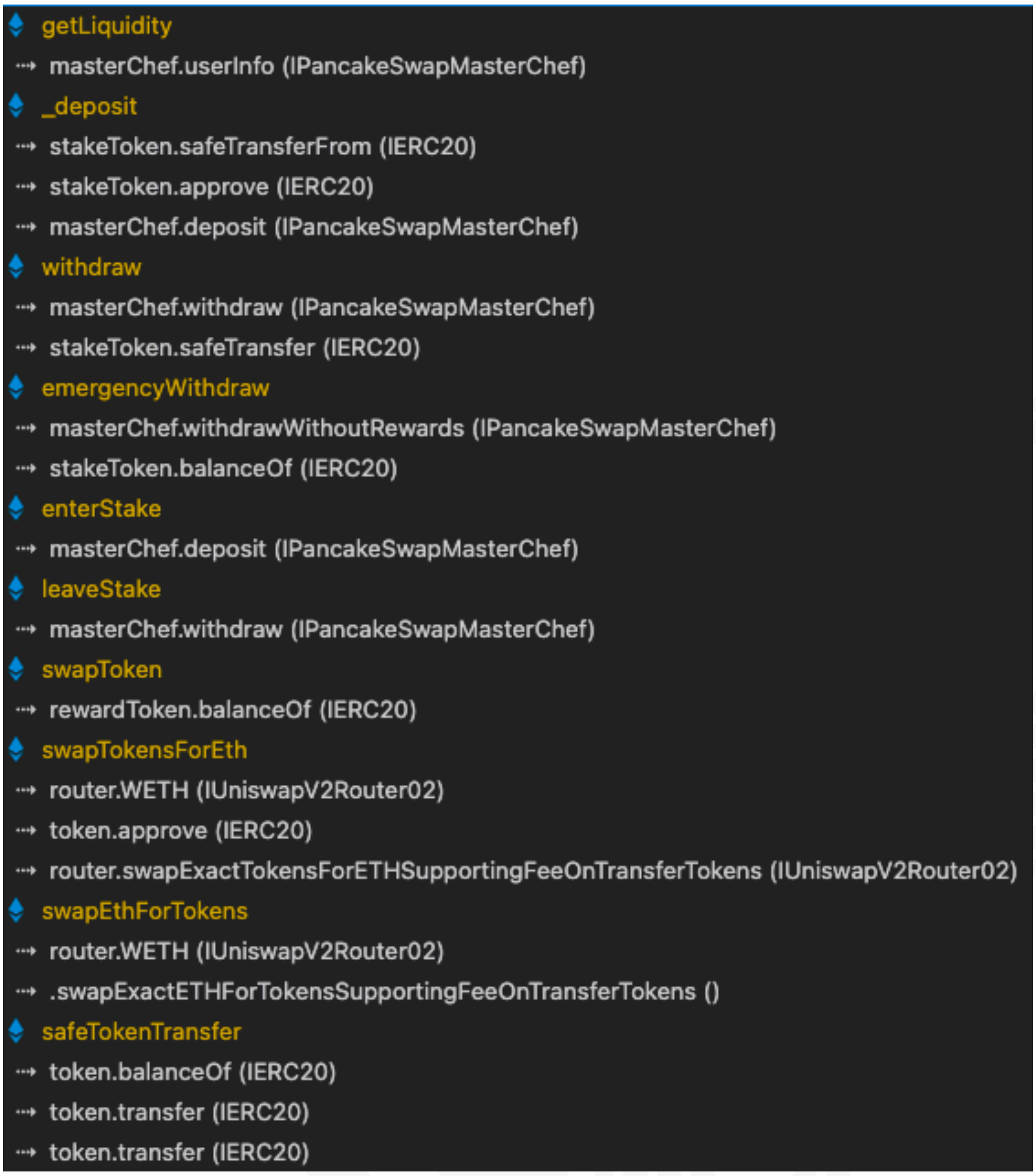
```
setCoin
  ⇢ coin.liquidityPair (AboatToken)
getCoinAmount
  ⇢ pair.token0 (IUniswapV2Pair)
  ⇢ pair.token1 (IUniswapV2Pair)
  ⇢ pair.getReserves (IUniswapV2Pair)
  ⇢ pair.totalSupply (IUniswapV2Pair)
getTokenPrice
  ⇢ coin.liquidityPair (AboatToken)
  ⇢ pair.getReserves (IUniswapV2Pair)
  ⇢ pair.token0 (IUniswapV2Pair)
  ⇢ pair.token1 (IUniswapV2Pair)
  ⇢ pair.token1 (IUniswapV2Pair)
  ⇢ pair.token0 (IUniswapV2Pair)
  ⇢ tokenB.decimals (ERC20)
```

MasterEntertainer



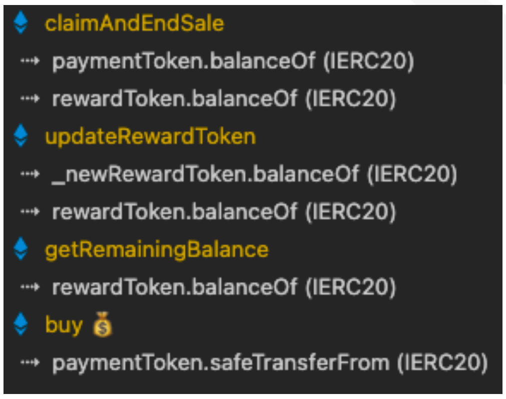
```
canClaimRewards
  ⇢ coin.canMintNewCoins (AboatToken)
updatePool
  ⇢ coin.mint (AboatToken)
  ⇢ coin.mint (AboatToken)
safeCoinTransfer
  ⇢ coin.balanceOf (AboatToken)
  ⇢ IERC20.safeTransfer (IERC20)
  ⇢ IERC20.safeTransfer (IERC20)
checkPriceUpdate
  ⇢ coin.liquidityPair (AboatToken)
```

BaseFlipPool



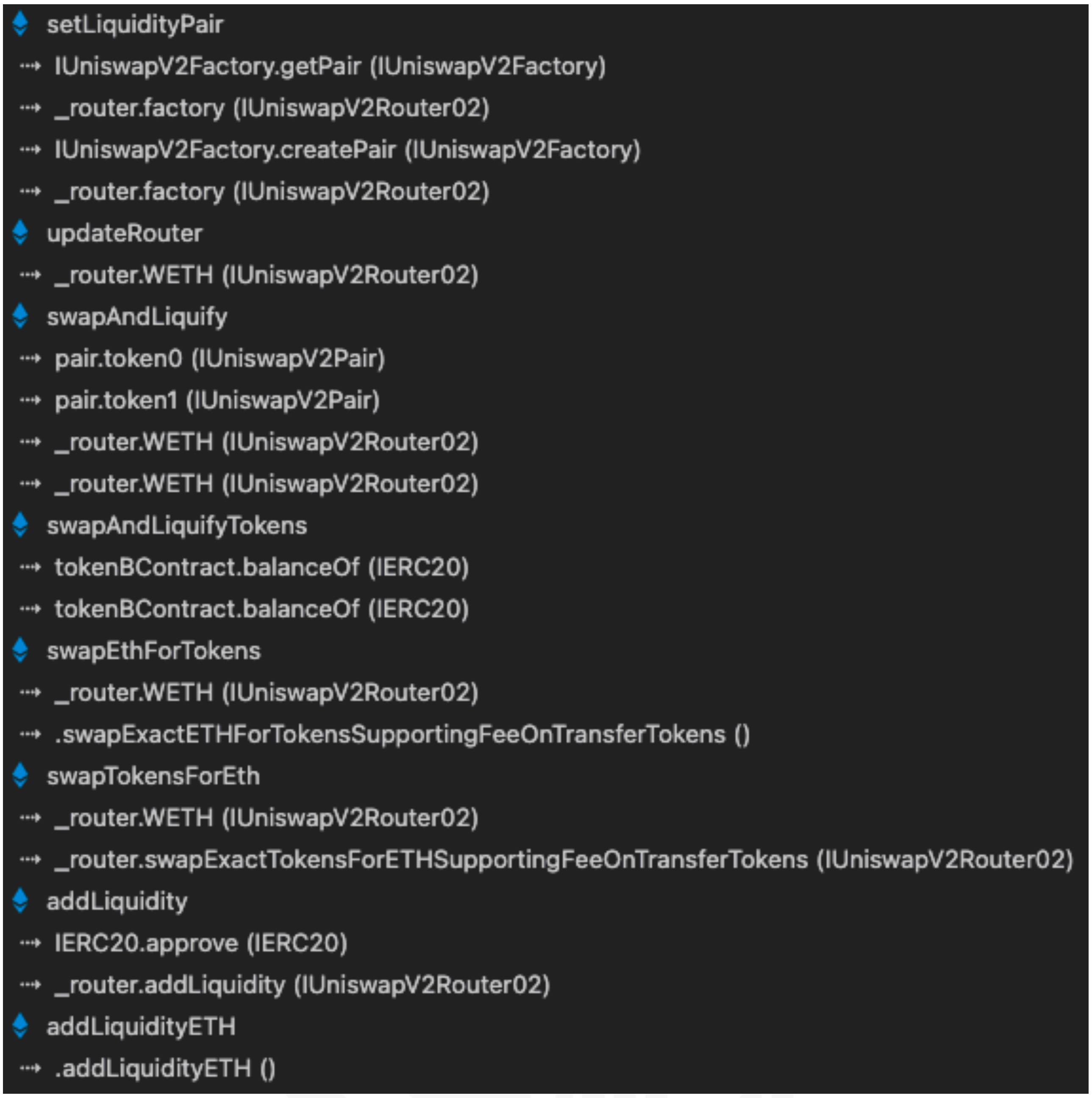
```
getLiquidity
 ⇢ masterChef.userInfo (IPancakeSwapMasterChef)
_deposit
 ⇢ stakeToken.safeTransferFrom (IERC20)
 ⇢ stakeToken.approve (IERC20)
 ⇢ masterChef.deposit (IPancakeSwapMasterChef)
withdraw
 ⇢ masterChef.withdraw (IPancakeSwapMasterChef)
 ⇢ stakeToken.safeTransfer (IERC20)
emergencyWithdraw
 ⇢ masterChef.withdrawWithoutRewards (IPancakeSwapMasterChef)
 ⇢ stakeToken.balanceOf (IERC20)
enterStake
 ⇢ masterChef.deposit (IPancakeSwapMasterChef)
leaveStake
 ⇢ masterChef.withdraw (IPancakeSwapMasterChef)
swapToken
 ⇢ rewardToken.balanceOf (IERC20)
swapTokensForEth
 ⇢ router.WETH (IUniswapV2Router02)
 ⇢ token.approve (IERC20)
 ⇢ router.swapExactTokensForETHSupportingFeeOnTransferTokens (IUniswapV2Router02)
swapEthForTokens
 ⇢ router.WETH (IUniswapV2Router02)
 ⇢ .swapExactETHForTokensSupportingFeeOnTransferTokens ()
safeTokenTransfer
 ⇢ token.balanceOf (IERC20)
 ⇢ token.transfer (IERC20)
 ⇢ token.transfer (IERC20)
```

Presale



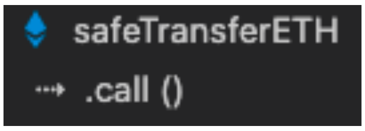
```
claimAndEndSale
 ⇢ paymentToken.balanceOf (IERC20)
 ⇢ rewardToken.balanceOf (IERC20)
updateRewardToken
 ⇢ _newRewardToken.balanceOf (IERC20)
 ⇢ rewardToken.balanceOf (IERC20)
getRemainingBalance
 ⇢ rewardToken.balanceOf (IERC20)
buy 💰
 ⇢ paymentToken.safeTransferFrom (IERC20)
```

Liquify



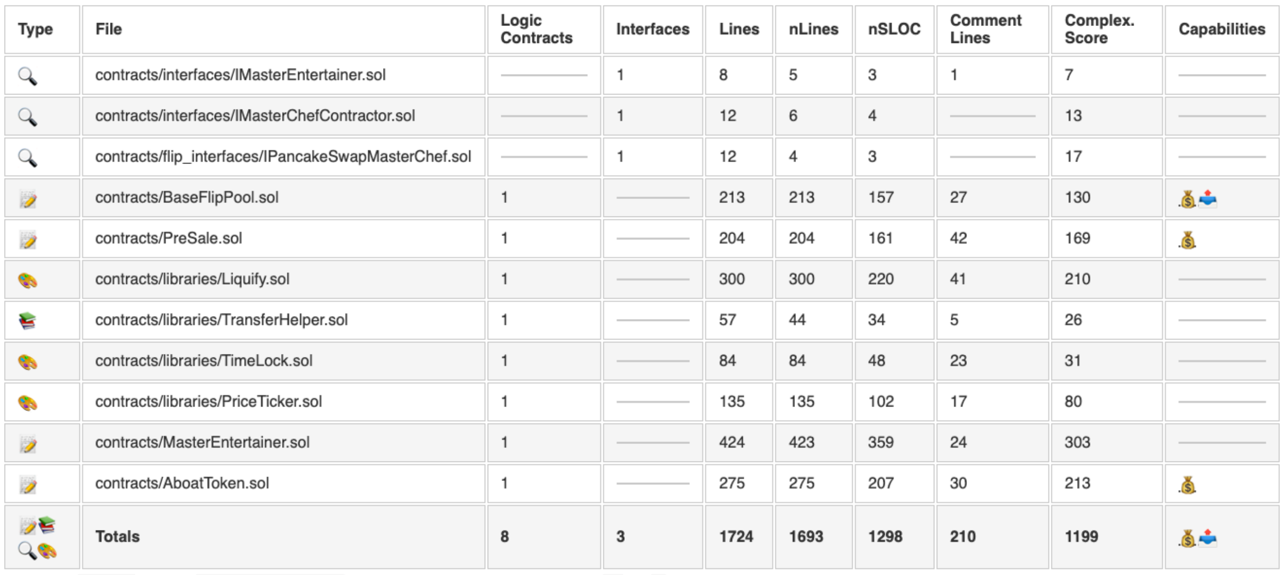
```
setLiquidityPair
  ⇢ IUniswapV2Factory.getPair (IUniswapV2Factory)
  ⇢ _router.factory (IUniswapV2Router02)
  ⇢ IUniswapV2Factory.createPair (IUniswapV2Factory)
  ⇢ _router.factory (IUniswapV2Router02)
updateRouter
  ⇢ _router.WETH (IUniswapV2Router02)
swapAndLiquify
  ⇢ pair.token0 (IUniswapV2Pair)
  ⇢ pair.token1 (IUniswapV2Pair)
  ⇢ _router.WETH (IUniswapV2Router02)
  ⇢ _router.WETH (IUniswapV2Router02)
swapAndLiquifyTokens
  ⇢ tokenBContract.balanceOf (IERC20)
  ⇢ tokenBContract.balanceOf (IERC20)
swapEthForTokens
  ⇢ _router.WETH (IUniswapV2Router02)
  ⇢ .swapExactETHForTokensSupportingFeeOnTransferTokens ()
swapTokensForEth
  ⇢ _router.WETH (IUniswapV2Router02)
  ⇢ _router.swapExactTokensForETHSupportingFeeOnTransferTokens (IUniswapV2Router02)
addLiquidity
  ⇢ IERC20.approve (IERC20)
  ⇢ _router.addLiquidity (IUniswapV2Router02)
addLiquidityETH
  ⇢ .addLiquidityETH ()
```

TransferHelper



```
safeTransferETH
  ⇢ .call ()
```

Source Units in Scope
v1.0



Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
🔍	contracts/interfaces/IMasterEntertainer.sol	————	1	8	5	3	1	7	————
🔍	contracts/interfaces/IMasterChefContractor.sol	————	1	12	6	4	————	13	————
🔍	contracts/flip_interfaces/IPancakeSwapMasterChef.sol	————	1	12	4	3	————	17	————
📝	contracts/BaseFlipPool.sol	1	————	213	213	157	27	130	💰📥
📝	contracts/PreSale.sol	1	————	204	204	161	42	169	💰
🎨	contracts/libraries/Liquify.sol	1	————	300	300	220	41	210	————
📚	contracts/libraries/TransferHelper.sol	1	————	57	44	34	5	26	————
🎨	contracts/libraries/TimeLock.sol	1	————	84	84	48	23	31	————
🎨	contracts/libraries/PriceTicker.sol	1	————	135	135	102	17	80	————
📝	contracts/MasterEntertainer.sol	1	————	424	423	359	24	303	————
📝	contracts/AboatToken.sol	1	————	275	275	207	30	213	💰
📝📚🔍🎨	**Totals**	**8**	**3**	**1724**	**1693**	**1298**	**210**	**1199**	💰📥

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
nSLOC	normalized source lines of code (only source-code lines; no comments, no blank lines)
Comment Lines	lines containing single or block comments
Complexity Score	a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Audit Results

AUDIT PASSED

Critical issues

No critical issues

High issues

No high issues

Medium issues

Issue	File	Type	Line	Description
#1	AboatToken	Reentrancy vulnerabilities	225	Apply the [`check-effects-interactions pattern`](http://solidity.readthedocs.io/en/v0.4.21/security-considerations.html#re-entrancy).or nonReentrant modifier from OpenZeppelin

Low issues

Issue	File	Type	Line	Description
#1	All	A floating pragma is set	1 or 2 or 3	The current pragma Solidity directive is „“^0.8.7””.
#2	BaseFlipPool	Missing Zero Address Validation (missing-zero-check)	52, 74, 64	Check that the address is not zero
#3	PriceTicker	Missing Zero Address Validation (missing-zero-check)	49	Check that the address is not zero
#4	Liquify	Missing Zero Address Validation (missing-zero-check)	81	Check that the address is not zero

#5	MasterEntertainer	Missing Zero Address Validation (missing-zero-check)	85, 101	Check that the address is not zero
#6	Presale	Missing Events Arithmetic	103, 113	Emit an event for critical parameter changes
#7	Presale	Out of gas	108, 114	Loop is used without any limitation. If there is no limitation and a long list of addresses, the function will be aborted
#8	MasterEntertainer	Wrong percentage or error message	114	If the basis points are 100 => 1% (file: Liquify, line 24) 10000 is not 10%, it is 100%

Informational issues

Issue	File	Type	Line	Description
#1	AboatToken	State variables that could be declared constant (constable-states)	40	Add the `constant` attributes to state variables that never change
#2	Presale	State variables that could be declared constant (constable-states)	31, 29, 30, 28	Add the `constant` attributes to state variables that never change
#3	TransferHelper	Functions that are not used	7	Remove unused functions
#4	IMasterchefContractor, IPancakeSwapMasterChef	SPDX-License is missing	-	Provide a SPDX-License to all source files
#5	Liquify	Wrong message	97, 102	Wrong require error message
#6	Presale	Costly operations in a loop	119	Use a local variable to hold the loop computation result - soldTokens

#7	Liquify	Misspelling issues	See lines in description	Change following variables - _excludedFromFeesAsReciever to excludedFromFeesAsReceiver line: 46 - tokensIntoLiqudity to tokensIntoLiquidity line: 57 - ChangedLiqudityPair to ChangedLiquidityPair line: 56 - excludeFromFeesAsReciever to excludeFromFeesAsReceiver line: 123 - Liquidity to liquidity line: 192 Make sure to change variables, functions etc. everywhere else if you want to change them
#8	AboatToken	Misspelling issues	See lines in description	Change following variables - MasteEntertainerTransferred to MasterEntertainerTransferred line: 53 - isExcludedFromRecieverTax to isExcludedFromReceiverTax line: 114 - _excludedFromFeesAsReciever to _excludedFromFeesAsReceiver line: 115, 233, 235, 237 Make sure to change variables, functions etc. everywhere else if you want to change them
#9	BaseFlipPool	Unused function parameter	96	Remove function parameter

#10	MasterEntertainer	Wrong visibility order	188, 389, 393	Visbility modifier “external/public” should come before other modifiers
#11	Presale	Wrong comment or value	29	40% or 10%? Change one or the other

Commented Code exist

There are some instances of code being commented out in the following files that should be removed:

File	Line	Comment
TransferHelper	12	// bytes4(keccak256(bytes('approve(address,uint256)')));
	26	// bytes4(keccak256(bytes('transfer(address,uint256)')));
	41	// bytes4(keccak256(bytes('transferFrom(address,address,uint256)')));

Recommendation

Remove the commented code, or address them properly.

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information https://docs.soliditylang.org/en/v0.5.10/natspec-format.html) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variables, functions etc. do.

19. January 2022:

- Read whole report for more information

SWC Attacks

ID	Title	Relationships	Status
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SWC-133	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	**PASSED**
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	**PASSED**
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	**PASSED**
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	**PASSED**
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	**PASSED**
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	**PASSED**
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	**PASSED**
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	**PASSED**
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	**PASSED**
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	**PASSED**

SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	NOT PASSED
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	**PASSED**
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	**PASSED**
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	**NOT PASSED**
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	**PASSED**
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	**PASSED**
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	**PASSED**

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY