Smart Contract
Security Audit
V1

Pulse Zen Token Smart Contract

19/8/2022

https://saferico.com/

business@saferico.com
https://t.me/SFI_ANN

–

Table of Contents

Table of Contents

Background

Project Information

- Token Information
- Executive Summary

File and Function Level Report

- **File in Scope:**

Issues Checking Status

- Severity Definitions
- Audit Findings

Automatic testing

- Testing proves
- Inheritance graph
- Call graph

Unified Modeling Language (UML)

Functions signature

Automatic general report

Conclusion

Disclaimer

Background

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Project Information

- **Platform**: Ethereum
- **Contract Address**: 0x5760da7a9e649CBE1b9cFCb88917914E4be0c6c9

Token Information

- Name: PLSZENS2
- Total Supply: 880,000,000
- Holders:
- Total transactions:

Contracts address deployed to test net (ETH)

Pulse Zen Token smart contract on Eth test net by the auditor to test every function (ETH Test Net)

https://rinkeby.etherscan.io/address/0x5760da7a9e649cbe1b9cfcb88917914e4be0c6c9

Executive Summary

According to our assessment, the customer`s solidity smart contract is **Secured**.

Well Secured	
Secured	✓
Poor Secured	
Insecure	

Automated checks are with remix IDE. All issues were performed by the team, which included the analysis of code functionality, manual audit found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the audit overview section. The general overview is presented in the Project Information section and all issues found are located in the audit overview section.

Team found 0 critical, 0 high, 0 medium, 3 low, 0 very low-level issues and 2 note in all solidity files of the contract

The files:

PulseZen.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
PulseZen.sol	**d1132d47426e428a43c0595ceca3b95946c9b61a216e34f39e9003326e5f8912**	0x5760da7a9e649CBE1b9cFCb88917914E4be0c6c9

- Contract: PLSZEN
- Inherit: Context, IERC20, Ownable
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
name	✓	Read / public	**Passed**
symbol	✓	Read / public	**Passed**
decimals	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
allowance	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
Owner	✓	Read / public	**Passed**
isExcludedFromFees	✓	Read / public	**Passed**
totalFees	✓	Read / public	**Passed**
tokenFromReflection	✓	Read / public	**Passed**
isExcludedFromReward	✓	Read / public	**Passed**
reflectionFromToken	✓	Read / public	**Passed**
_charityFee	✓	Read / public	**Passed**

_burnFee	✓	Read / public	**Passed**
_taxFee	✓	Read / public	**Passed**
_walletLimits	✓	Read / public	**Passed**
checkWalletLimit	✓	Read / public	**Passed**
getCirculatingSupply	✓	Read / public	**Passed**
DEAD	✓	Read / public	**Passed**
isWalletLimitExempt	✓	Read / public	**Passed**
uniswapV2Router	✓	Read / public	**Passed**
uniswapV2Pair	✓	Read / public	**Passed**
ZERO	✓	Read / public	**Passed**
approve	✓	Write / public	**Passed**
transferFrom	✓	Write / public	**Passed**
increaseAllowance	✓	Write / public	**Passed**
transfer	✓	Write / public	**Passed**
decreaseAllowance	✓	Write / public	**Passed**
deliver	✓	Write / public	**Passed**
enableWalletLimit	✓	Write / public	**Passed**
excludeFromFees	✓	Write / public	**Passed**
setBurnAmount	✓	Write / public	**Passed**
includeInReward	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**
transferOwnership	✓	Write / public	**Passed**
setCharityFeePercent	✓	Write / public	**Passed**
excludeFromReward	✓	Write / public	**Passed**
setCharityWallet	✓	Write / public	**Passed**
setRefFeePercent	✓	Write / public	**Passed**
setWalletLimit	✓	Write / public	**Passed**
setWalletLimitExempt	✓	Write / public	**Passed**

Issues Checking Status

No.	Issue Description	Checking Status
1	Compiler warnings.	**Passed**
2	Race conditions and Reentrancy. Cross-function race conditions.	**Passed**
3	Possible delays in data delivery.	**Passed**
4	Oracle calls.	**Passed**
5	Design Logic.	**Passed**
6	Timestamp dependence.	**Passed**
7	Integer Overflow and Underflow.	**Passed**
8	DoS with Revert.	**Passed**
9	DoS with block gas limit.	**Passed with notes**
10	Methods execution permissions.	**Passed**
11	Economy model. If application logic is based on an incorrect economic model, the application would not function correctly and participants would incur financial losses. This type of issue is most often found in bonus rewards systems, Staking and Farming contracts, Vault and Vesting contracts, etc.	**Passed**
12	The impact of the exchange rate on the logic.	**Passed**
13	Private user data leaks.	**Passed**
14	Malicious Event log.	**Passed**
15	Scoping and Declarations.	**Passed**
16	Uninitialized storage pointers.	**Passed**
17	Arithmetic accuracy.	**Passed**

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 have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can’t lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can’t have significant impact on execution
Note	Lowest-level vulnerabilities, code style violations and info statements can’t affect smart contract execution and can be ignored.

Audit Findings

Critical:

No Critical severity vulnerabilities were found.

High:

No High severity vulnerabilities were found.

Medium:

No Medium severity vulnerabilities were found.

Low:

#Pragam version not fixed

Description

It is a good practice to lock the solidity version for a live deployment (use 0.8.15 instead of ^0.8.4). contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors.

Remediation

Remove the ^ sign to lock the pragma version.

Status: Closed. Fixed in version2.

#Missing zero address validation

Description

When the owner wants to change charity wallet, he has to check for the zero address to make, he didn't add the zero address. Otherwise, charity fees will be burn fees.

```
function setCharityWallet(address _adr) external onlyOwner() {
        _charityWalletAddress = _adr;
    }
```

Remediation

Use the require statement to check for zero addresses.

Status: Closed. Fixed in version2.

#Owner privileges (In the period when the owner isn't renounced)

Description

The owner can add limits of holding the token to any wallet.
The owner can change all fees.
The owner can include / exclude any address from Fees or Reward.

```
function excludeFromFee(address account) public onlyOwner {
        _isExcludedFromFee[account] = true;
    }
    function includeInFee(address account) public onlyOwner {
        _isExcludedFromFee[account] = false;
    }
    function setRefFeePercent(uint256 taxFee) external onlyOwner() {
        _taxFee = taxFee;
    }
    function setCharityFeePercent(uint256 charityFee) external onlyOwner() {
        _charityFee = charityFee;
    }
    function setBurnAmount(uint256 burnAmount) external onlyOwner() {
        _burnFee = burnAmount;
    }
    function setWalletLimit(uint256 _amount) external onlyOwner() {
        _walletLimit = _amount;
    }
    function setWalletLimitExempt(address _adr, bool _status) external onlyOwner()
{
        isWalletLimitExempt[_adr] = _status;
    }

    function enableWalletLimit(bool _status) external onlyOwner() {
        checkWalletLimit = _status;
    }

    function setCharityWallet(address _adr) external onlyOwner() {
        _charityWalletAddress = _adr;

    }
```

Remediation

Make these functions internal in next version or the team should announce the investors before change the fees and give them time if they want to use the old fees.

P.S: This issue is common to the majority of rewards smart contracts.

Status: Acknowledged.

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

#Naming Conventions

Description

The contract follows a consistent naming convention where we are private variables with leading"_" and public variables without it. But we have missed to comply to the condition for certain variable names "__taxFees" which is public.

Remediation

Remove "_" from external variable names and add it to private variable names.

Status: Acknowledged

Constant calculations in the contract

Description

recalculated initialization will save 2847 units of gas in deployment

```
uint256 private _tTotal = 880000000 * 10**9;
```

Recommendation

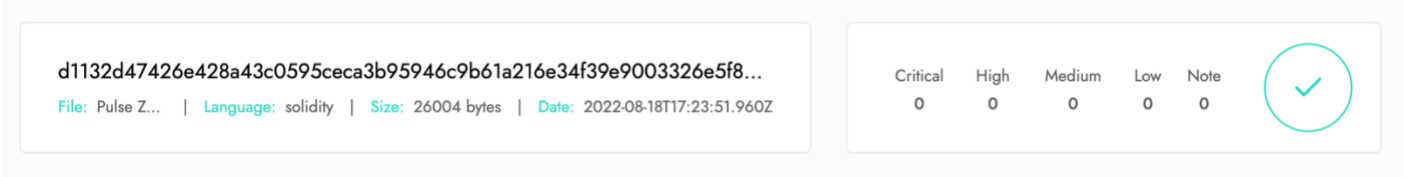
Replace the initialization as

```
uint256 private _tTotal = 880000000000000000;
```

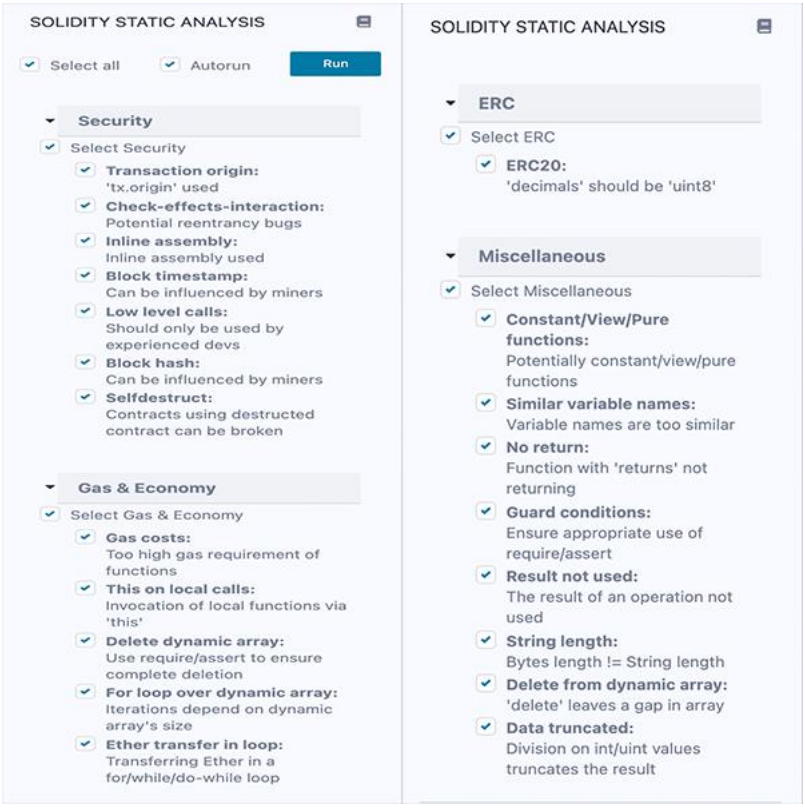
Status Acknowledged.

Automatic Testing

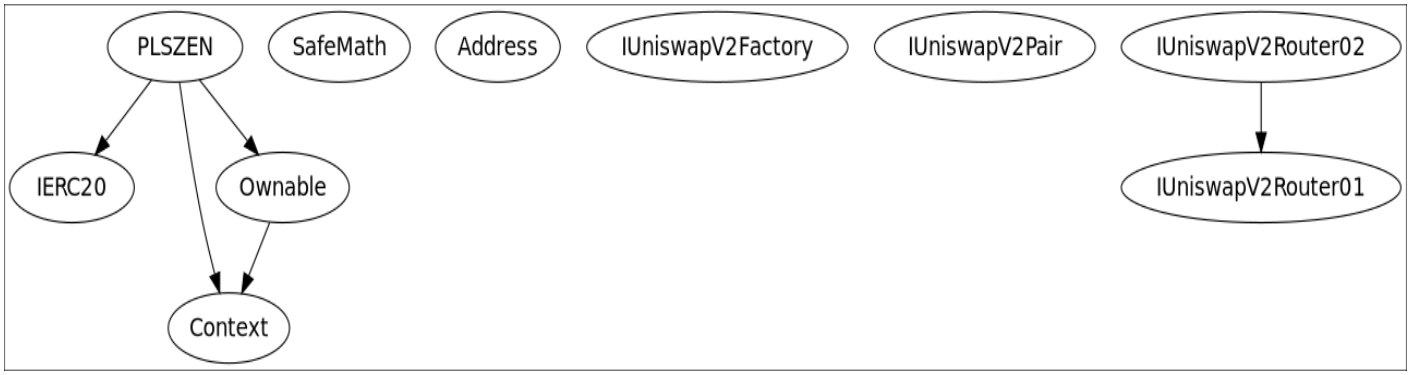
1- Check for security



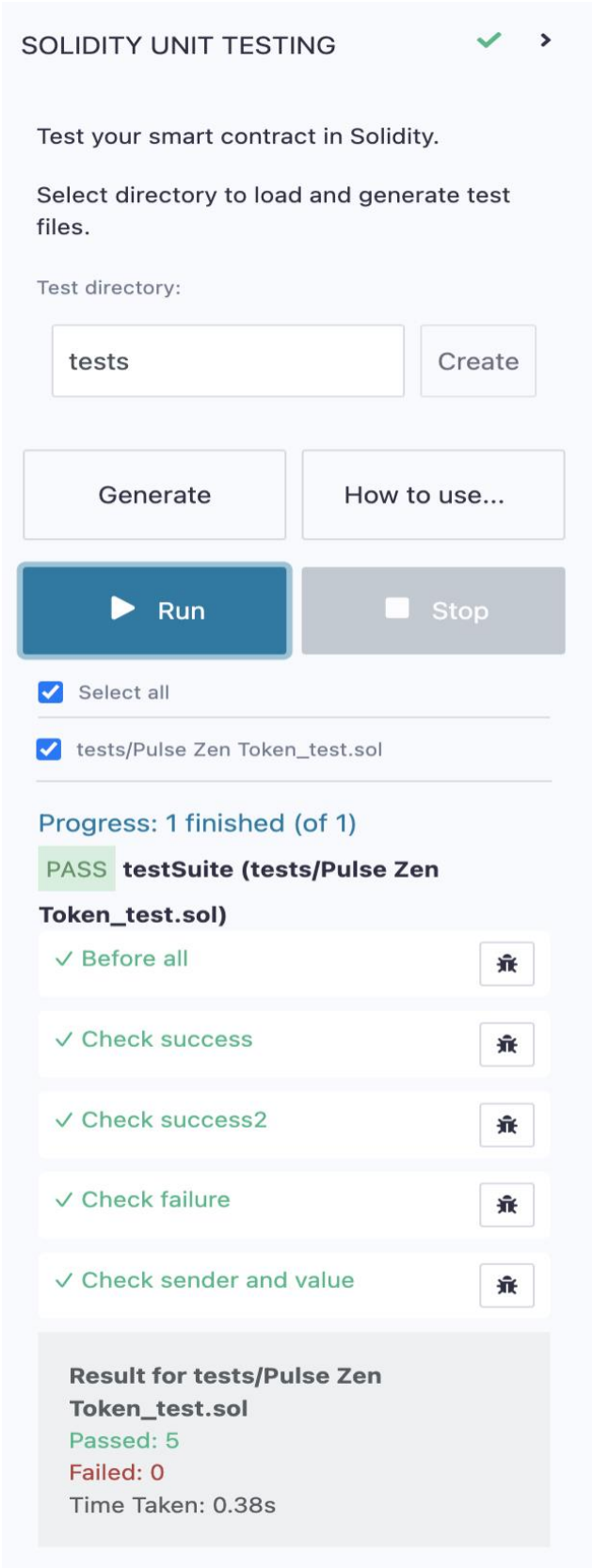
2- SOLIDITY STATIC ANALYSIS



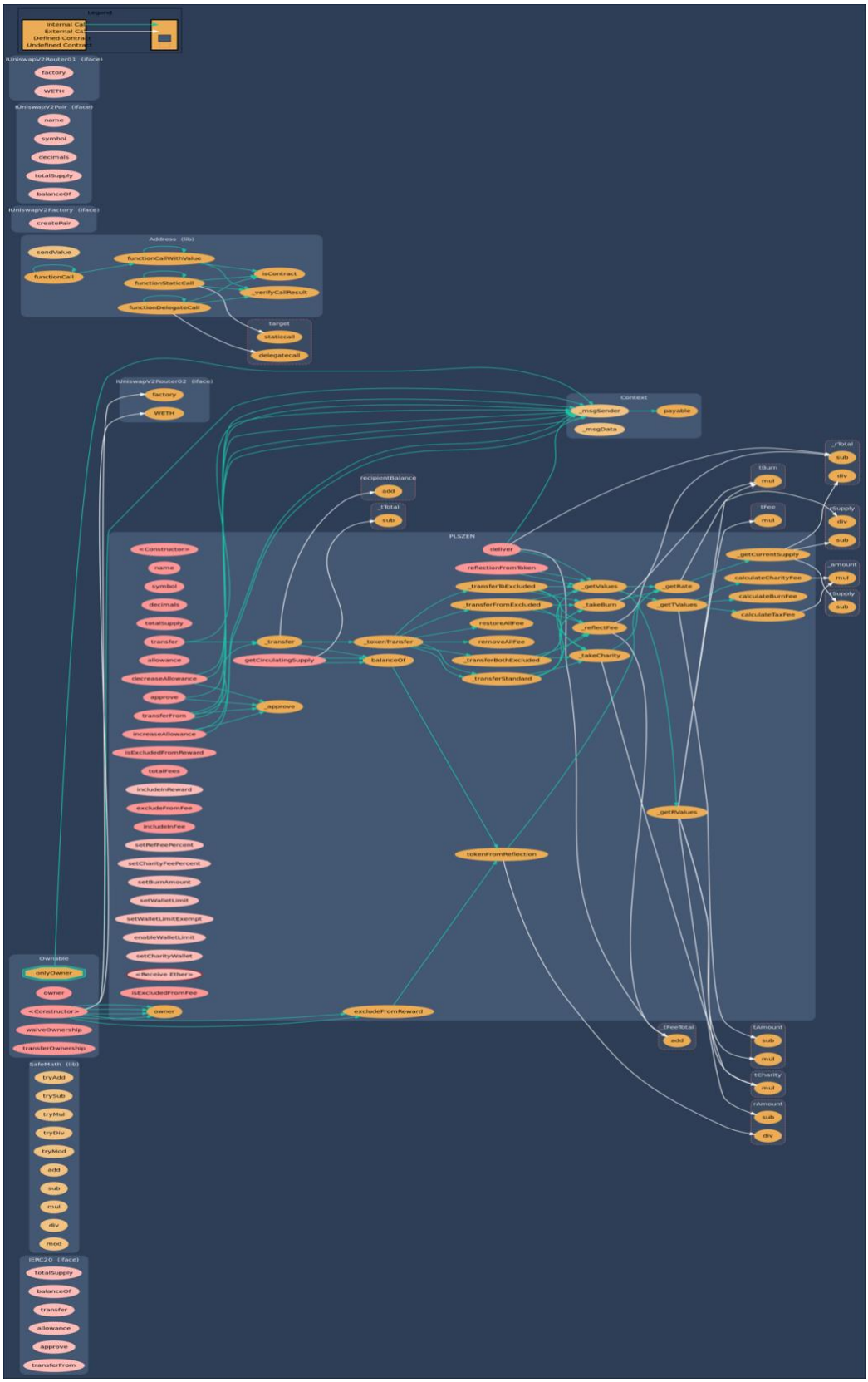
3- Inheritance graph



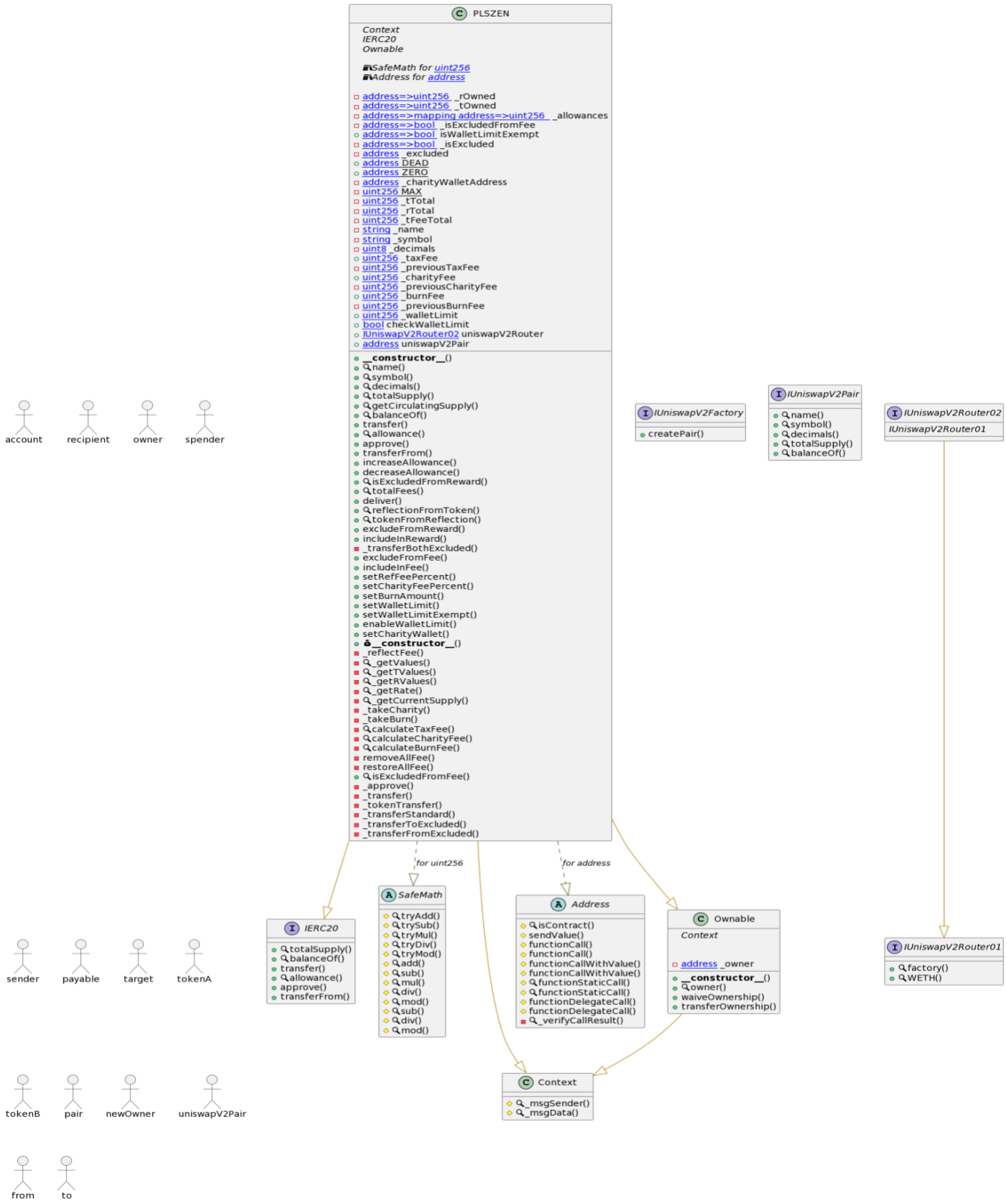
4- SOLIDITY UNIT TESTING



5- Call graph


Legend
IUniswapV2Router01 (iface)
factory
WETH
IUniswapV2Pair (iface)
name
symbol
decimals
totalSupply
balanceOf
IUniswapV2Factory (iface)
createPair
Address (lib)
sendValue
functionCallWithValue
functionCall
functionStaticCall
isContract
_verifyCallResult
functionDelegateCall
target
staticcall
delegatecall
IUniswapV2Router02 (iface)
factory
WETH
Context
_msgSender
payable
_msgData
recipientBalance
add
_tTotal
sub
PLSZEN
<Constructor>
name
symbol
decimals
totalSupply
transfer
allowance
decreaseAllowance
approve
transferFrom
increaseAllowance
isExcludedFromReward
totalFees
includeInReward
excludeFromFee
includeInFee
setReffeePercent
setCharityFeePercent
setBurnAmount
setWalletLimit
setWalletLimitExempt
enableWalletLimit
setCharityWallet
<Receive Ether>
isExcludedFromFee
_transfer
getCirculatingSupply
_tokenTransfer
balanceOf
_approve
deliver
reflectionFromToken
_transferToExcluded
_transferFromExcluded
restoreAllFee
removeAllFee
_transferBothExcluded
_transferStandard
_getValues
_takeBurn
_reflectFee
_takeCharity
_getRate
_getTValues
_getRValues
tokenFromReflection
excludeFromReward
_getCurrentSupply
calculateCharityFee
calculateBurnFee
calculateTaxFee
tBurn
mul
tFee
mul
_rTotal
sub
div
tSupply
div
sub
_amount
mul
tSupply
sub
Ownable
onlyOwner
owner
<Constructor>
owner
waiveOwnership
transferOwnership
tFeeTotal
add
tAmount
sub
mul
tCharity
mul
rAmount
sub
div
SafeMath (lib)
tryAdd
trySub
tryMul
tryDiv
tryMod
add
sub
mul
div
mod
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom

Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
11902160  =>  _getTValues(uint256)
16279055  =>  isContract(address)
39509351  =>  increaseAllowance(address,uint256)
75128141  =>  calculateTaxFee(uint256)
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
884557bf  =>  tryAdd(uint256,uint256)
a29962b1  =>  trySub(uint256,uint256)
6281efa4  =>  tryMul(uint256,uint256)
736ecb18  =>  tryDiv(uint256,uint256)
38dc0867  =>  tryMod(uint256,uint256)
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
f43f523a  =>  mod(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
b745d336  =>  div(uint256,uint256,string)
71af23e8  =>  mod(uint256,uint256,string)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
24a084df  =>  sendValue(address,uint256)
a0b5ffb0  =>  functionCall(address,bytes)
241b5886  =>  functionCall(address,bytes,string)
2a011594  =>  functionCallWithValue(address,bytes,uint256)
d525ab8a  =>  functionCallWithValue(address,bytes,uint256,string)
c21d36f3  =>  functionStaticCall(address,bytes)
dbc40fb9  =>  functionStaticCall(address,bytes,string)
ee33b7e2  =>  functionDelegateCall(address,bytes)
57387df0  =>  functionDelegateCall(address,bytes,string)
18c2c6a2  =>  _verifyCallResult(bool,bytes,string)
c9c65396  =>  createPair(address,address)
06fdde03  =>  name()
95d89b41  =>  symbol()
313ce567  =>  decimals()
c45a0155  =>  factory()
ad5c4648  =>  WETH()
8da5cb5b  =>  owner()
914eb66a  =>  waiveOwnership()
f2fde38b  =>  transferOwnership(address)
2b112e49  =>  getCirculatingSupply()
a457c2d7  =>  decreaseAllowance(address,uint256)
88f82020  =>  isExcludedFromReward(address)
13114a9d  =>  totalFees()
3bd5d173  =>  deliver(uint256)
4549b039  =>  reflectionFromToken(uint256,bool)
2d838119  =>  tokenFromReflection(uint256)
52390c02  =>  excludeFromReward(address)
3685d419  =>  includeInReward(address)
6ff6cdf4  =>  _transferBothExcluded(address,address,uint256)
```

```
437823ec  =>  excludeFromFee(address)
ea2f0b37  =>  includeInFee(address)
28f235ba  =>  setRefFeePercent(uint256)
af41063b  =>  setCharityFeePercent(uint256)
cc43f3d3  =>  setBurnAmount(uint256)
f1d5f517  =>  setWalletLimit(uint256)
d04a33d1  =>  setWalletLimitExempt(address,bool)
4974d88f  =>  enableWalletLimit(bool)
30563bd7  =>  setCharityWallet(address)
184d894e  =>  _reflectFee(uint256,uint256)
d4780e36  =>  _getValues(uint256)
65c63d72  =>  _getRValues(uint256,uint256,uint256,uint256,uint256)
94e10784  =>  _getRate()
97a9d560  =>  _getCurrentSupply()
81ef5eb0  =>  _takeCharity(uint256)
9520e9d4  =>  _takeBurn(uint256)
0b2d25ca  =>  calculateCharityFee(uint256)
a40d0e3c  =>  calculateBurnFee()
301370af  =>  removeAllFee()
e7e3e3a7  =>  restoreAllFee()
5342acb4  =>  isExcludedFromFee(address)
104e81ff  =>  _approve(address,address,uint256)
30e0789e  =>  _transfer(address,address,uint256)
b09bbc79  =>  _tokenTransfer(address,address,uint256,bool)
2852df65  =>  _transferStandard(address,address,uint256)
16f1cc83  =>  _transferToExcluded(address,address,uint256)
c7d9be66  =>  _transferFromExcluded(address,address,uint256)
```

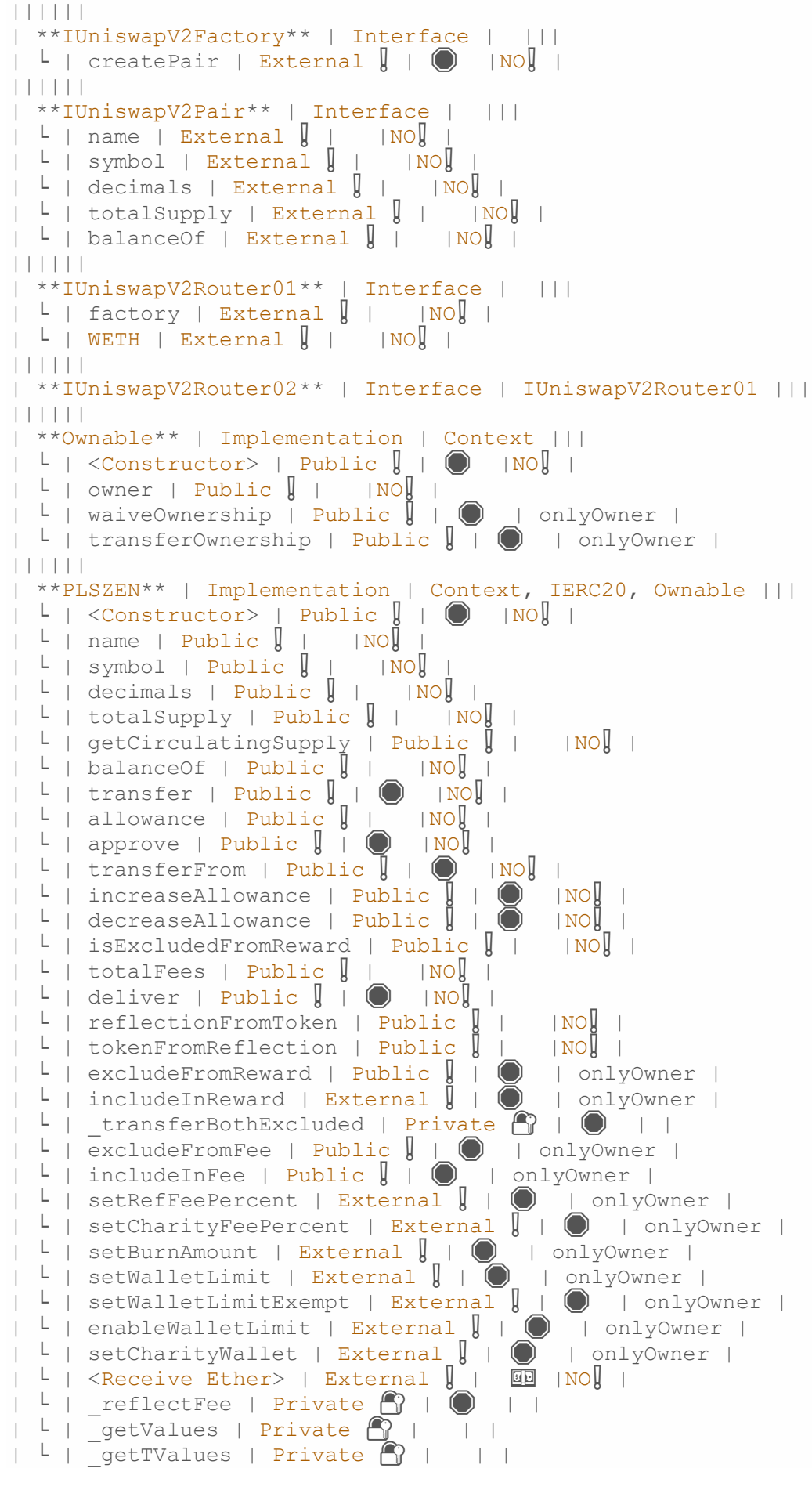
Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/Pulse Zen Token.sol	9d90352c075dbb6d498b88031602f770ed0e8cf4

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
SafeMath	Library			
└	tryAdd	Internal 🔒		
└	trySub	Internal 🔒		
└	tryMul	Internal 🔒		
└	tryDiv	Internal 🔒		
└	tryMod	Internal 🔒		
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	sub	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Address	Library			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionStaticCall	Internal 🔒		
└	functionStaticCall	Internal 🔒		
└	functionDelegateCall	Internal 🔒	🛑	
└	functionDelegateCall	Internal 🔒	🛑	
└	_verifyCallResult	Private 🔐		


IUniswapV2Factory	Interface			
└	createPair	External		NO
IUniswapV2Pair	Interface			
└	name	External		NO
└	symbol	External		NO
└	decimals	External		NO
└	totalSupply	External		NO
└	balanceOf	External		NO
IUniswapV2Router01	Interface			
└	factory	External		NO
└	WETH	External		NO
IUniswapV2Router02	Interface	IUniswapV2Router01		
Ownable	Implementation	Context		
└	<Constructor>	Public		NO
└	owner	Public		NO
└	waiveOwnership	Public		onlyOwner
└	transferOwnership	Public		onlyOwner
PLSZEN	Implementation	Context, IERC20, Ownable		
└	<Constructor>	Public		NO
└	name	Public		NO
└	symbol	Public		NO
└	decimals	Public		NO
└	totalSupply	Public		NO
└	getCirculatingSupply	Public		NO
└	balanceOf	Public		NO
└	transfer	Public		NO
└	allowance	Public		NO
└	approve	Public		NO
└	transferFrom	Public		NO
└	increaseAllowance	Public		NO
└	decreaseAllowance	Public		NO
└	isExcludedFromReward	Public		NO
└	totalFees	Public		NO
└	deliver	Public		NO
└	reflectionFromToken	Public		NO
└	tokenFromReflection	Public		NO
└	excludeFromReward	Public		onlyOwner
└	includeInReward	External		onlyOwner
└	_transferBothExcluded	Private		
└	excludeFromFee	Public		onlyOwner
└	includeInFee	Public		onlyOwner
└	setRefFeePercent	External		onlyOwner
└	setCharityFeePercent	External		onlyOwner
└	setBurnAmount	External		onlyOwner
└	setWalletLimit	External		onlyOwner
└	setWalletLimitExempt	External		onlyOwner
└	enableWalletLimit	External		onlyOwner
└	setCharityWallet	External		onlyOwner
└	<Receive Ether>	External		NO
└	_reflectFee	Private		
└	_getValues	Private		
└	_getTValues	Private		

└	_getRValues	Private 🔐			
└	_getRate	Private 🔐			
└	_getCurrentSupply	Private 🔐			
└	_takeCharity	Private 🔐	🛑		
└	_takeBurn	Private 🔐	🛑		
└	calculateTaxFee	Private 🔐			
└	calculateCharityFee	Private 🔐			
└	calculateBurnFee	Private 🔐			
└	removeAllFee	Private 🔐	🛑		
└	restoreAllFee	Private 🔐	🛑		
└	isExcludedFromFee	Public ❗		NO❗	
└	_approve	Private 🔐	🛑		
└	_transfer	Private 🔐	🛑		
└	_tokenTransfer	Private 🔐	🛑		
└	_transferStandard	Private 🔐	🛑		
└	_transferToExcluded	Private 🔐	🛑		
└	_transferFromExcluded	Private 🔐	🛑		

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Conclusion

The contracts are written systematically. Team found no critical issues. So, it is good to go for production.

Since possible test cases can be unlimited and developer level documentation (code flow diagram with function level description) not provided, for such an extensive smart contract protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan Everything.

Security state of the reviewed contract is “Secured”.

✓ No mint function.
✓ No volatile code.
✓ No high severity issues were found.

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against the team on the basis of what it says or doesn't say, or how team produced it, and it is important for you to conduct your own independent investigations before making any decisions. team go into more detail on this in the below disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Saferico and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Saferico s) owe no duty of care towards you or any other person, nor does Saferico make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Saferico hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Saferico hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Saferico, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.