Smart Contract Security Audit

Smart Contract	Audited by	Dated
Mr.Mint	Esp Softtech	05/05/2022

Table of Contents

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**: Binance Smart Chain

- **Contract Address**: 0xcbb352de4241f3967c281e81446d72942f50e0f0

- **MR. MINT Smart Contract**: MR. MINT Binance smart chain-based
  Binance verified audited contract enables the use of Binance Coin BEP-20 for use in investment industry. The Longevity and stability of MR. MINT will ultimately bring crypto users, miners, investors under single platform to transfer, refer and fees income with unlimited opportunities making use of decentralization becomes a perfect platform which operates effortlessly 24x7 all the days of year!

Executive Summary

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

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 1 high, 0 medium, 1 low, 0 very low-level issues in all solidity files of the contract

The files: Mr.Mint sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
Mr. Mint.sol	0xb9373ba92f1cbff8da1c824f317da2e7270f9d3a7cc9e1df35825ed09bfd7f4c	0xcbb352de4241f3967c281e81446d72942f50e0f0

- **Contract**: Mr.Mint
- **Inherit**: Context, IBEP20, Ownable, SafeMath
- **Observation**: Passed including security check
- **Test Report**: Passed
- **Score**: Passed
- **Conclusion**: Passed

Function	Test Result	Type / Return Type	Score
totalSupply	✔	Read / Public	**Passed**
decimals	✔	Read / Public	**Passed**
symbol	✔	Read / Public	**Passed**
name	✔	Read / Public	**Passed**
getOwner	✔	Read / Public	**Passed**
balanceOf	✔	Read / Public	**Passed**
transfer	✔	Read / Public	**Passed**
allowance	✔	Read / Public	**Passed**
approve	✔	Read / Public	**Passed**
transferFrom	✔	Read / Public	**Passed**
_msgSender	✔	Read / Public	**Passed**

_msgData	✔	Read / Public	**Passed**
add	✔	Read / Public	**Passed**
sub	✔	Read / Public	**Passed**
getHold_amount	✔	Read / Public	**Passed**
unhold	✔	Read / Public	**Passed**
hold	✔	Read / Public	**Passed**
transferAnyBSC20Token	✔	Read / Public	**Passed**
registrationExt	✔	Read / Public	**Passed**
sellToken	✔	Read / Public	**Passed**
buyToken	✔	Read / Public	**Passed**
destroyBlackFunds	✔	Read / Public	**Passed**
reclaimToken	✔	Read / Public	**Passed**
isFreeze	✔	Read / Public	**Passed**
unFreeze	✔	Read / Public	**Passed**
freeze	✔	Read / Public	**Passed**
freezeAccount	✔	Read / Public	**Passed**
freezeList	✔	Read / Public	**Passed**
getBlackListStatus	✔	Read / Public	**Passed**
addBlackList	✔	Read / Public	**Passed**
removeBlackList	✔	Read / Public	**Passed**
isBlackListed	✔	Read / Public	**Passed**

mul	✓	Read / Public	**Passed**
div	✓	Read / Public	**Passed**
mod	✓	Read / Public	**Passed**
owner	✓	Read / Public	**Passed**
renounceOwnership	✓	Read / Public	**Passed**
transferOwnership	✓	Read / Public	**Passed**
_transferOwnership	✓	Read / Public	**Passed**
setLiquidityFeeWallet	✓	Read / Public	**Passed**
setLiquidityFeePercent	✓	Read / Public	**Passed**
setDevelopmentFeeWallet	✓	Read / Public	**Passed**
setDevelopmentFeePercent	✓	Read / Public	**Passed**
setBuyBackLockFeeWallet	✓	Read / Public	**Passed**
setBuyBackLockFeePercent	✓	Read / Public	**Passed**
excludeFromFee	✓	Read / Public	**Passed**
includeInFee	✓	Read / Public	**Passed**
decreaseAllowance	✓	Read / Public	**Passed**
calculateFee	✓	Read / Public	**Passed**
_transfer	✓	Read / Public	**Passed**
_burn	✓	Read / Public	**Passed**
_approve	✓	Read / Public	**Passed**
burn	✓	Read / 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	Front running.	**Passed**
6	Timestamp dependence.	**Passed**
7	Integer Overflow and Underflow.	**Passed**
8	DoS with Revert.	**Passed**
9	DoS with block gas limit.	**Passed**
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**
18	Design Logic.	**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 High severity vulnerabilities were found

No Medium severity vulnerabilities were found.

#Burn function is public

Description
Any user can burn his token directly with function which is not a good practice.   Which may impact on totalSupply.

Remediation
This should be done by admin or owner of smart contract. If this is a project   functionality then this finding can be ignored.

High:
No High severity vulnerabilities were found

Medium:
No Medium severity vulnerabilities were found.

Low:

#Showing wallet address in smart contract

Description
In constructor 3 wallet addresses has been exposed which are used for different transfers. Showing wallet addresses in smart contract is not a good practice which can cause know attacks on wallet addresses and transferring methods.

Remediation
Avoid inserting wallet addresses in smart contract. It can be added while deploying.

Status: **Acknowledged**

Very Low:
No Very Low severity vulnerabilities were found.

Notes:

None

Automatic Testing

1- Check for Security
Critical - 1, High - 0, Medium - 0, Low - 1

2- SOLIDITY STATIC ANALYSIS

3- SOLIDITY UNIT TESTING

SOLIDITY UNIT TESTING

Test your smart contract in Solidity.

Select directory to load and generate test files.

Test directory:

tests   Create

Generate   How to use...

▶ Run   ■ Stop

☑ Select all

☑ tests/4_Ballot_test.sol
☑ tests/audit_test.sol

Progress: 1 finished (of 2)

PASS BallotTest (tests/4_Ballot_test.sol)

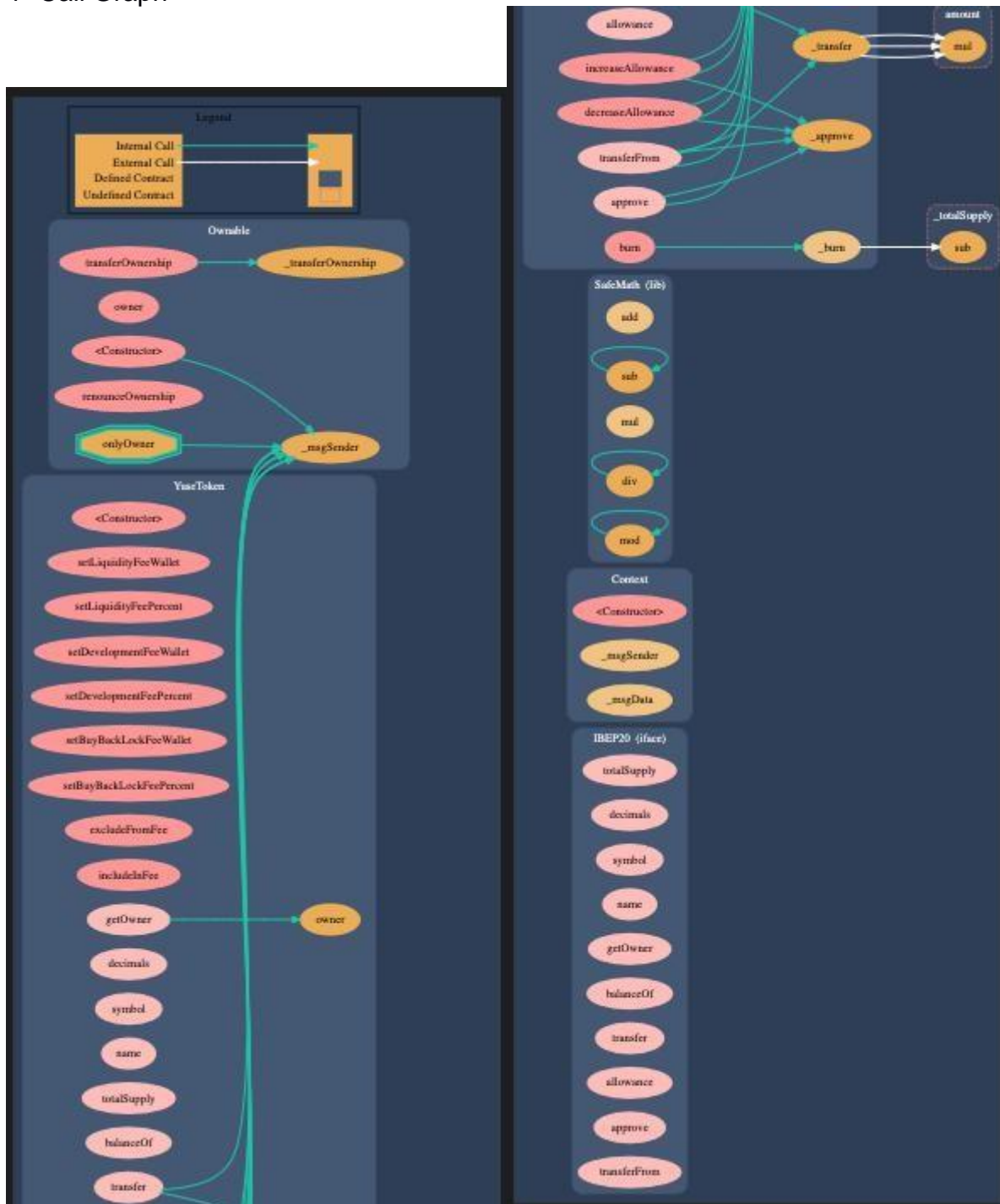✓ Check winning proposal

✓ Check winnin proposal with return value
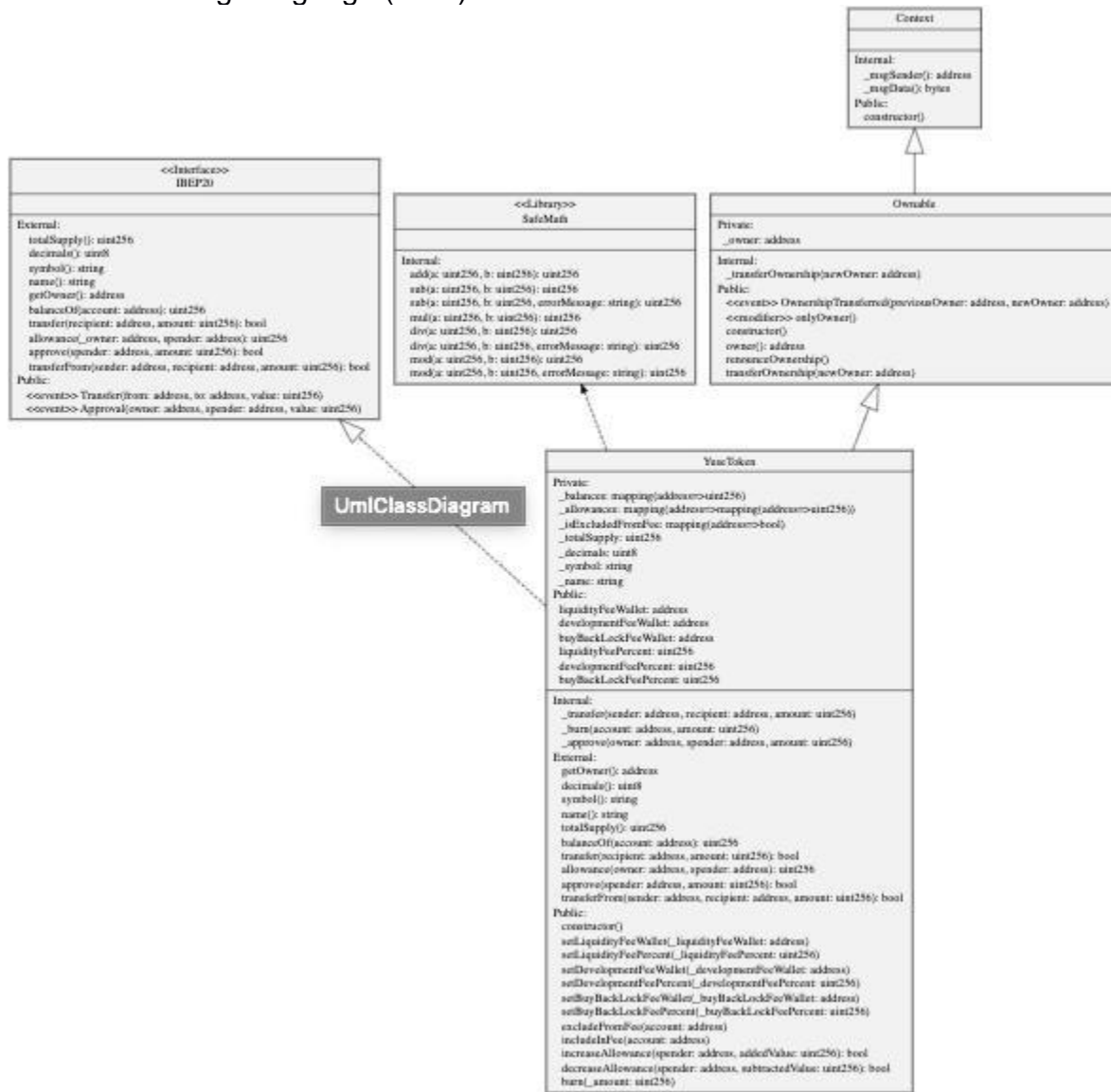
Result for tests/4_Ballot_test.sol
Passed: 2
Failed: 0
Time Taken: 0.53s

4- Call Graph

Unified Modeling Language (UML)

Context

Internal:
  _msgSender(): address
  _msgData(): bytes
Public:
  constructor()

<<Interface>>
IBEP20

External:
  totalSupply(): uint256
  decimals(): uint8
  symbol(): string
  name(): string
  getOwner(): address
  balanceOf(account: address): uint256
  transfer(recipient: address, amount: uint256): bool
  allowance(_owner: address, spender: address): uint256
  approve(spender: address, amount uint256): bool
  transferFrom(sender: address, recipient: address, amount: uint256): bool
Public:
  <<event>> Transfer(from: address, to: address, value: uint256)
  <<event>> Approval(owner: address, spender: address, value: uint256)

<<Library>>
SafeMath

Internal:
  add(a: uint256, b: uint256): uint256
  sub(a: uint256, b: uint256): uint256
  sub(a: uint256, b: uint256, errorMessage: string): uint256
  mul(a: uint256, b: uint256): uint256
  div(a: uint256, b: uint256): uint256
  div(a: uint256, b: uint256, errorMessage: string): uint256
  mod(a: uint256, b: uint256): uint256
  mod(a: uint256, b: uint256, errorMessage: string): uint256

Ownable

Private:
  _owner: address
Internal:
  _transferOwnership(newOwner: address)
Public:
  <<event>> OwnershipTransferred(previousOwner: address, newOwner: address)
  <<modifier>> onlyOwner()
  constructor()
  owner(): address
  renounceOwnership()
  transferOwnership(newOwner: address)

UmlClassDiagram

YusoToken

Private:
  _balances: mapping(address=>uint256)
  _allowances: mapping(address=>mapping(address=>uint256))
  _isExcludedFromFee: mapping(address=>bool)
  _totalSupply: uint256
  _decimals: uint8
  _symbol: string
  _name: string
Public:
  liquidityFeeWallet: address
  developmentFeeWallet: address
  buyBackLockFeeWallet: address
  liquidityFeePercent: uint256
  developmentFeePercent: uint256
  buyBackLockFeePercent: uint256

Internal:
  _transfer(sender: address, recipient: address, amount: uint256)
  _burn(account: address, amount: uint256)
  _approve(owner: address, spender: address, amount: uint256)
External:
  getOwner(): address
  decimals(): uint8
  symbol(): string
  name(): string
  totalSupply(): uint256
  balanceOf(account: address): uint256
  transfer(recipient: address, amount: uint256): bool
  allowance(owner: address, spender: address): uint256
  approve(spender: address, amount: uint256): bool
  transferFrom(sender: address, recipient: address, amount: uint256): bool
Public:
  constructor()
  setLiquidityFeeWallet(_liquidityFeeWallet: address)
  setLiquidityFeePercent(_liquidityFeePercent: uint256)
  setDevelopmentFeeWallet(_developmentFeeWallet: address)
  setDevelopmentFeePercent(_developmentFeePercent: uint256)
  setBuyBackLockFeeWallet(_buyBackLockFeeWallet: address)
  setBuyBackLockFeePercent(_buyBackLockFeePercent: uint256)
  excludeFromFee(account: address)
  includeInFee(account: address)
  increaseAllowance(spender: address, addedValue: uint256): bool
  decreaseAllowance(spender: address, subtractedValue: uint256): bool
  burn(_amount: uint256)

Functions Signature

```
Sighash   |   Function Signature
========================
39509351  =>  increaseAllowance(address,uint256)
18160ddd  =>  totalSupply()
313ce567  =>  decimals()
95d89b41  =>  symbol()
06fdde03  =>  name()
893d20e8  =>  getOwner()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
b745d336  =>  div(uint256,uint256,string)
f43f523a  =>  mod(uint256,uint256)
71af23e8  =>  mod(uint256,uint256,string)
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
d29d44ee  =>  _transferOwnership(address)
686cbb28  =>  setLiquidityFeeWallet(address)
8ee88c53  =>  setLiquidityFeePercent(uint256)
cd50e5bf  =>  setDevelopmentFeeWallet(address)
4680ff35  =>  setDevelopmentFeePercent(uint256)
eb1897c5  =>  setBuyBackLockFeeWallet(address)
c023c93f  =>  setBuyBackLockFeePercent(uint256)
437823ec  =>  excludeFromFee(address)
ea2f0b37  =>  includeInFee(address)
a457c2d7  =>  decreaseAllowance(address,uint256)
99a5d747  =>  calculateFee(uint256)
30e0789e  =>  _transfer(address,address,uint256)
6161eb18  =>  _burn(address,uint256)
104e81ff  =>  _approve(address,address,uint256)
42966c68  =>  burn(uint256)
```

Automatic General Report

Files Description Table

File Name	SHA-1 Hash
/Users/Downloads/sol/audit.sol a4c60c13b0a09699192da9849eb48cfac02f1bbd	

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IBEP20	Interface			
└	totalSupply	External ❗	|NO❗	
└	decimals	External ❗	|NO❗	
└	symbol	External ❗	|NO❗	
└	name	External ❗	|NO❗	
└	getOwner	External ❗	|NO❗	
└	balanceOf	External ❗	|NO❗	
└	transfer	External ❗	u |NO❗	
└	allowance	External ❗	|NO❗	
└	approve	External ❗	u |NO❗	
└	transferFrom	External ❗	u |NO❗	
Context	Implementation			
└	<Constructor>	Public ❗	u |NO❗	
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		

Ownable	Implementation	Context		
└	<Constructor>	Public ❗️	🛑	NO❗️
└	owner	Public ❗️		NO❗️
└	renounceOwnership	Public ❗️	🛑	onlyOwner
└	transferOwnership	Public ❗️	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	
Mr.Mint	Implementation	IBEP20, Ownable		
└	<Constructor>	Public ❗️	🛑	NO❗️
└	setLiquidityFeeWallet	Public ❗️	🛑	onlyOwner
└	setLiquidityFeePercent	Public ❗️	🛑	onlyOwner
└	setDevelopmentFeeWallet	Public ❗️	🛑	onlyOwner
└	setDevelopmentFeePercent	Public ❗️	🛑	onlyOwner
└	setBuyBackLockFeeWallet	Public ❗️	🛑	onlyOwner
└	setBuyBackLockFeePercent	Public ❗️	🛑	onlyOwner
└	excludeFromFee	Public ❗️	🛑	onlyOwner
└	includeInFee	Public ❗️	🛑	onlyOwner
└	getOwner	External ❗️		NO❗️
└	decimals	External ❗️		NO❗️
└	symbol	External ❗️		NO❗️
└	name	External ❗️		NO❗️
└	totalSupply	External ❗️		NO❗️
└	balanceOf	External ❗️		NO❗️
└	transfer	External ❗️	🛑	NO❗️
└	allowance	External ❗️		NO❗️
└	approve	External ❗️	🛑	NO❗️
└	transferFrom	External ❗️	🛑	NO❗️
└	increaseAllowance	Public ❗️	🛑	NO❗️
└	decreaseAllowance	Public ❗️	🛑	NO❗️
└	_transfer	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	burn	Public ❗️	🛑	NO❗️

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 **Secure**.

✔ Well structured code.
✔ High severity issues were not found.
✔ No Contract Ownership Renounced.

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 Esp Softtech and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Esp Softtech) owe no duty of care towards you or any other person, nor does Esp Softtech make any warranty or representation to any person on the accuracy or completeness of the report. Esp Softtech's does not have any control on admin wallets or funds involved The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Esp Softtech 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, Esp Softtech hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Esp Softtech, 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. Esp Softtech involved just to audit smart contract. 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.