Smart Contract
Security Audit

Smart Contract	Audited by	Dated
Mr. Mint	EspSofttech	05/05/2022

Table of Contents

Background

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions asintended.
- Identifypotentialsecurityissueswiththesmartcontract.

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 SmartChain
- **Contract Address**: 0x0Dea8E1Bf6bf41D6ffd45A13b24A5Ffbe8f2e902
- **MR. MINT SmartContract**: MR. MINT Binance smartchain-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 ofyear!

Executive Summary

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

Well Secured	✓
Secured	
Poor Secured	
Insecure	

Automated checks have been done with remix IDE. ALI issues were performed by the team, including the analysis of code functionality and manual audit review during the automated analysis. Applicable vulnerabilities, if any, are presented in the Audit Overview section. A general overview is presented in the Project Information section.

Team found 0 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	0xe14dff3bc7a3383ed16da8414ca3e7ce45c2f4084179be3e1757c8844a2bac66	0x0Dea8E1Bf6bf41D6ffd45A13b24A5Ffbe8f2e902

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

Function	Test Result	Type / Return Type	Score
allowance	✓	Read / Public	**Passed**
balanceOf	✓	Read / Public	**Passed**
buyBackLockFeePercent	✓	Read / Public	**Passed**
buyBackLockFeeWallet	✓	Read / Public	**Passed**
decimals	✓	Read / Public	**Passed**
developmentFeePercent	✓	Read / Public	**Passed**
developmentFeeWallet	✓	Read / Public	**Passed**
freezelist	✓	Read / Public	**Passed**
getBlackListStatus	✓	Read / Public	**Passed**
getHold_amount	✓	Read / Public	**Passed**
getOwner	✓	Read / Public	**Passed**

isBlackListed	✓	Read / Public	Passed
isFreeze	✓	Read / Public	Passed
liquidityFeePercent	✓	Read / Public	Passed
liquidityFeeWallet	✓	Read / Public	Passed
name	✓	Read / Public	Passed
owner	✓	Read / Public	Passed
symbol	✓	Read / Public	Passed
addBlackList	✓	Write / Public	Passed
approve	✓	Write / Public	Passed
burn	✓	Write / Public	Passed
buyToken	✓	Write / Public	Passed
decreaseAllowance	✓	Write / Public	Passed
destroyBlackFunds	✓	Write / Public	Passed
excludeFromFee	✓	Write / Public	Passed
freeze	✓	Write / Public	Passed
hold	✓	Write / Public	Passed
includeInFee	✓	Write / Public	Passed
increaseAllowance	✓	Write / Public	Passed
mint	✓	Write / Public	Passed
reclaimToken	✓	Write / Public	Passed
registrationExt	✓	Write / Public	Passed

removeBlackList	✓	Write / Public	Passed
renounceOwnership	✓	Write / Public	Passed
sellToken	✓	Write / Public	Passed
setBuyBackLockFeePercent	✓	Write / Public	Passed
setBuyBackLockFeeWallet	✓	Write / Public	Passed
setDevelopmentFeePercent	✓	Write / Public	Passed
setDevelopmentFeeWallet	✓	Write / Public	Passed
setLiquidityFeePercent	✓	Write / Public	Passed
setLiquidityFeeWallet	✓	Write / Public	Passed
transfer	✓	Write / Public	Passed
transferAnyBSC20Token	✓	Write / Public	Passed
tarnsferFrom	✓	Write / Public	Passed
transferOwernship	✓	Write / Public	Passed
unFreeze	✓	Write / Public	Passed
unhold	✓	Write / Public	Passed
withdraw	✓	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	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, Vaultand 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 token 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 token 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 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 addressess 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 & Fixed**

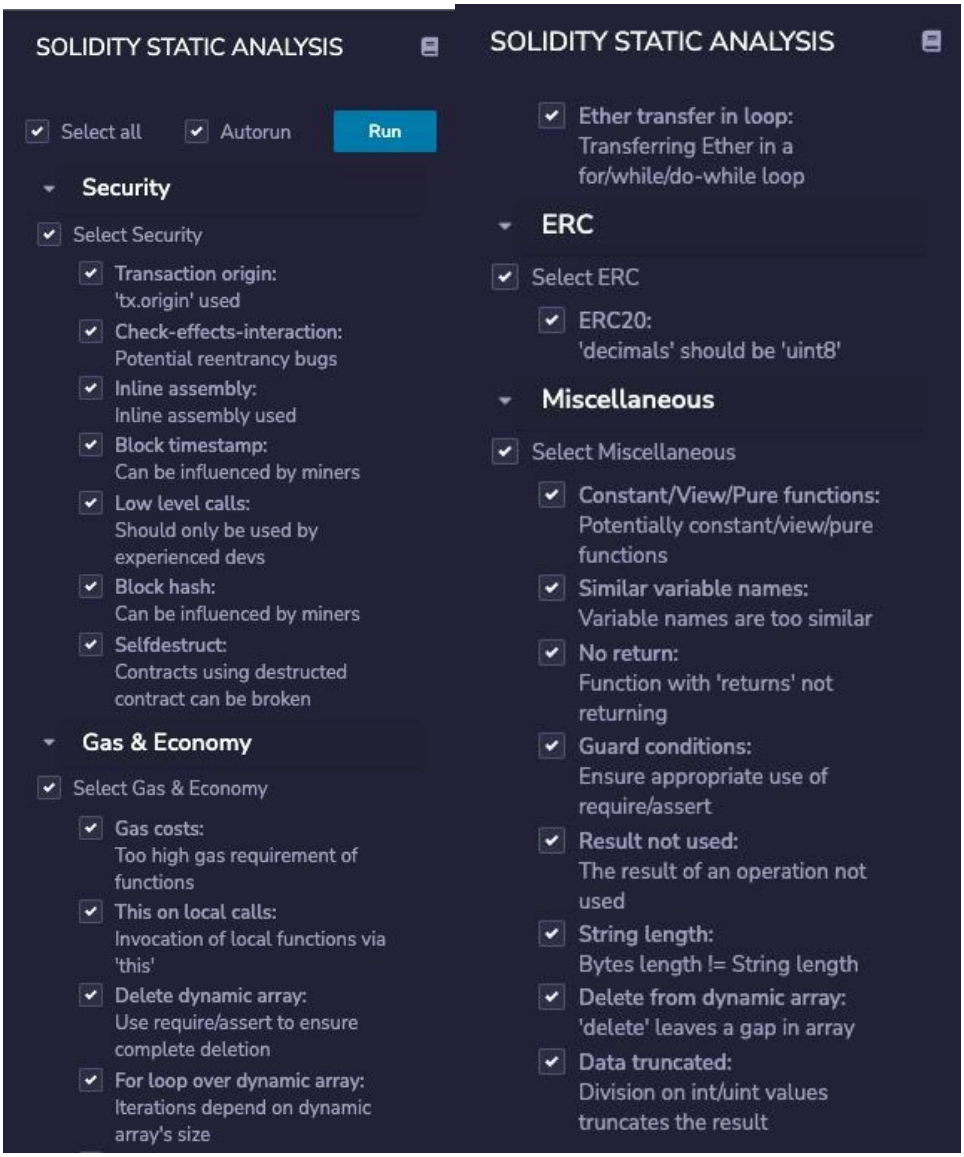
Very Low:

No Very Low severity vulnerabilities were found.

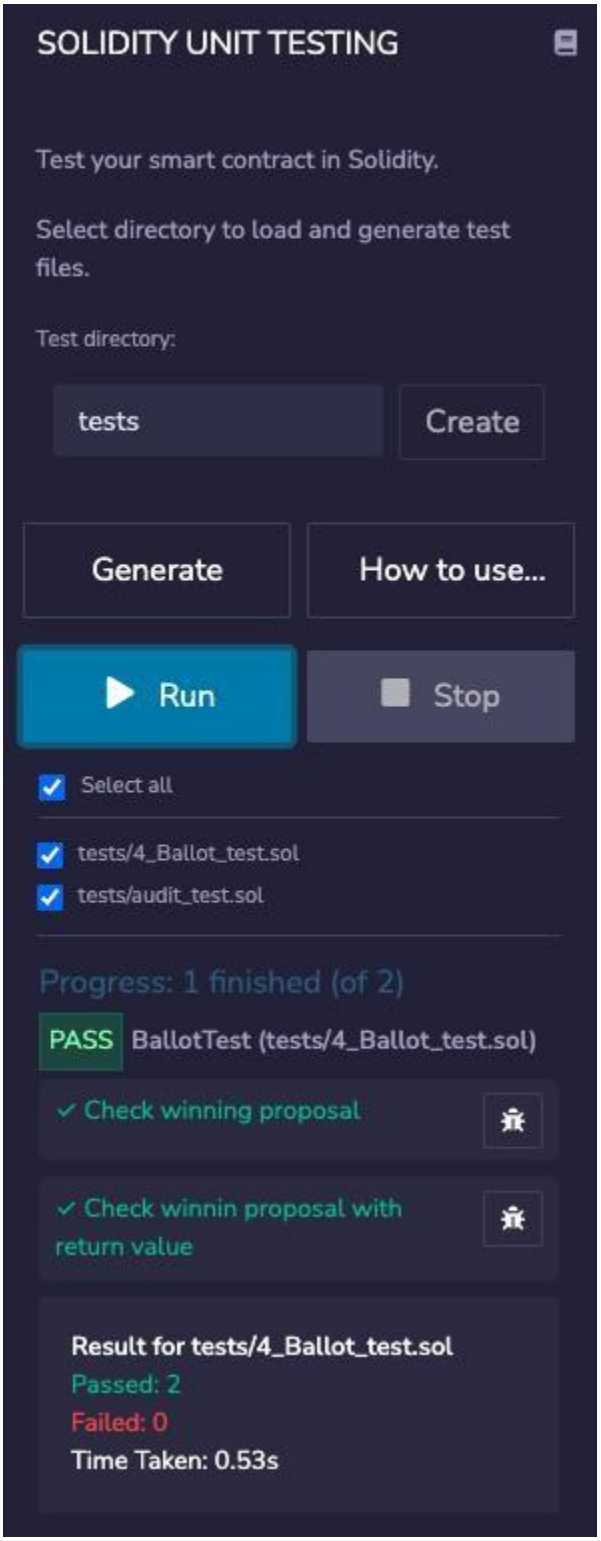
Automatic Testing

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

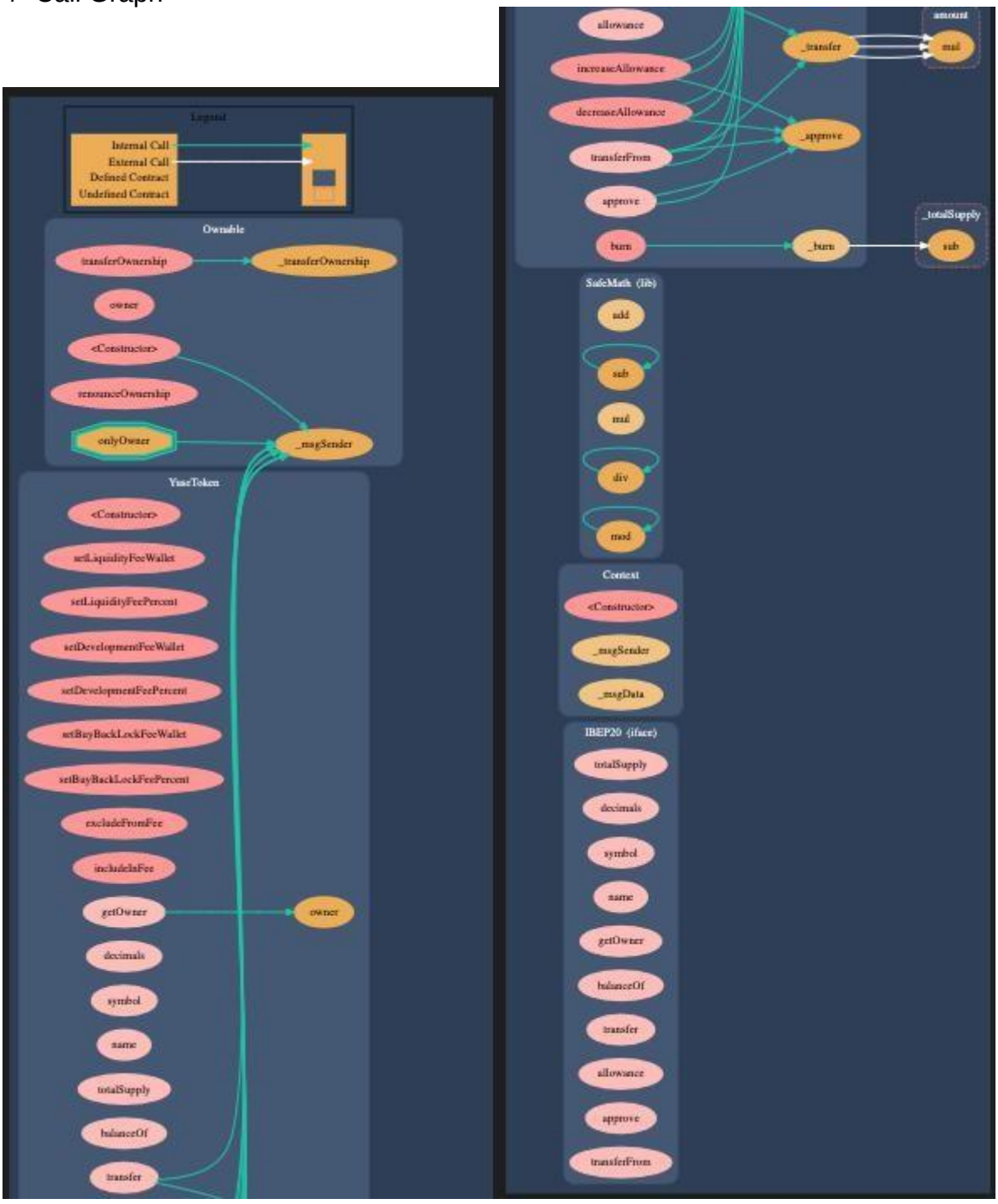
2- SOLIDITY STATICANALYSIS



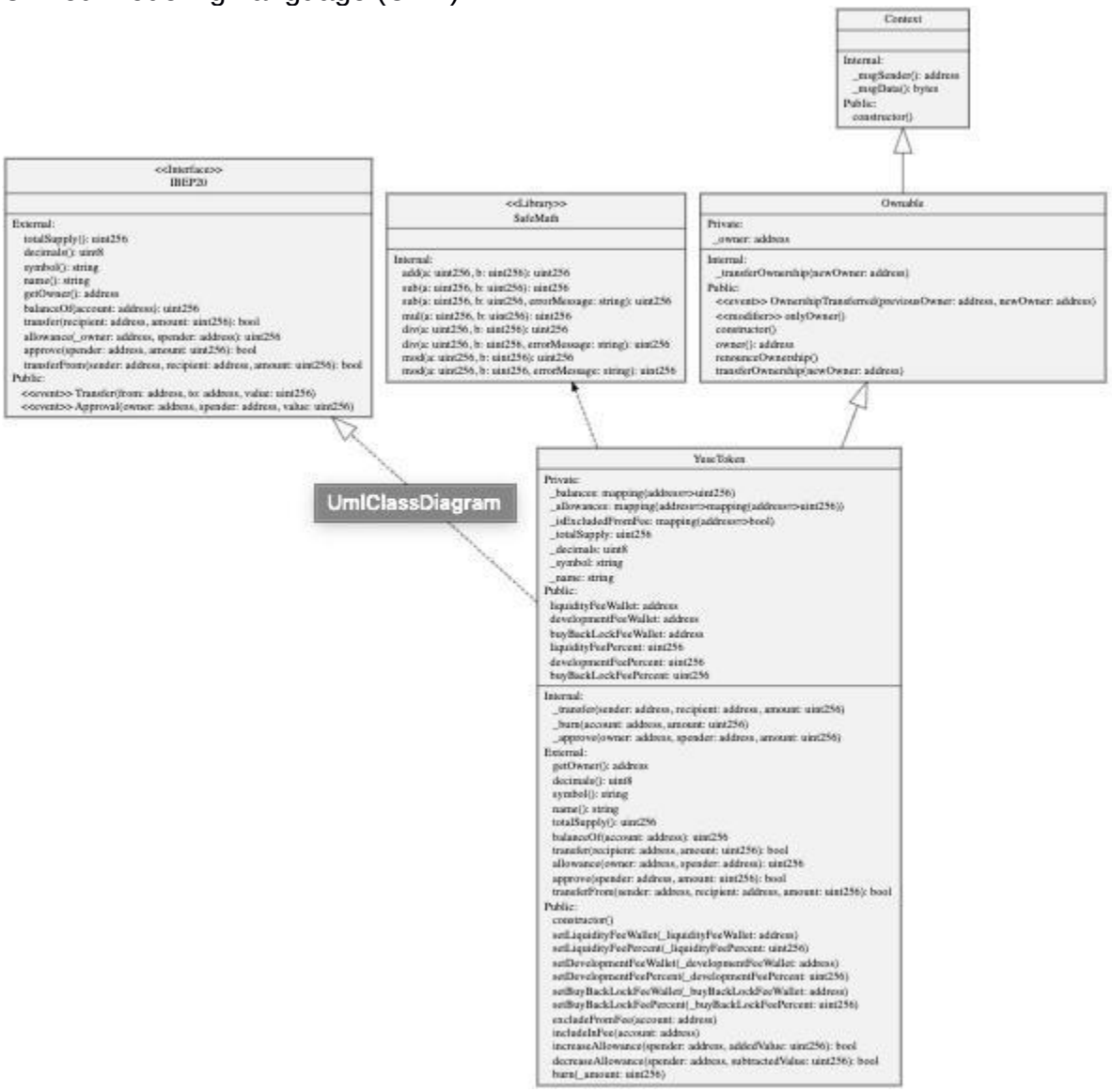
3- SOLIDITY UNITTESTING



4- Call Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Ownable
transferOwnership
_transferOwnership
owner
<Constructor>
renounceOwnership
onlyOwner
_msgSender
YuseToken
<Constructor>
setLiquidityFeeWallet
setLiquidityFeePercent
setDevelopmentFeeWallet
setDevelopmentFeePercent
setBuyBackLockFeeWallet
setBuyBackLockFeePercent
excludeFromFee
includeInFee
getOwner
owner
decimals
symbol
name
totalSupply
balanceOf
transfer
allowance
increaseAllowance
decreaseAllowance
transferFrom
approve
burn
_transfer
_approve
_burn
amount
mul
_totalSupply
sub
SafeMath (lib)
add
sub
mul
div
mod
Context
<Constructor>
_msgSender
_msgData
IBEP20 (iface)
totalSupply
decimals
symbol
name
getOwner
balanceOf
transfer
allowance
approve
transferFrom

Unified Modeling Language (UML)



Functions Signature

```
Sighash   |   FunctionSignature
========================
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

FileName	SHA-1Hash
/Users/Downloads/sol/audit.sol	a4c60c13b0a09699192da9849eb48cfac02f1bbd

Contracts Description Table

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

Legend

Symbol	Meaning
u	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 flowdiagramwithfunctionleveldescription) notprovided, forsuchanextensivesmart contractprotocol, weprovidenosuchguaranteeoffutureoutcomes.Wehaveusedall thelateststatictoolsandmanualobservationstocovermaximumpossibletestcases to scan Everything.

Security state of the reviewed contract is **Secure**.

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