

Smart Contract Security Audit

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

Table of Contents

1.	Background	3
2.	Project Information	3
	Smart Contract Information	
	Executive Summary	
3.	File and Function Level Report	5
	File in Scope	
4.	Issues Checking Status	7
	Severity Definition	
	Audit Findings	
5.	Automatic Testing	11
	Testing proves	
	Inheritance graph	
	Call graph	
6.	Unified Modeling Language (UML)	14
7.	Functions Signature	15
8.	Automatic General Report	16
9.	Conclusion	18
10.	Disclaimer	19

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	0xb9373ba92f1cbff8da1c824f317d a2e7270f9d3a7cc9e1df35825ed09 bfd7f4c	0xcbb352de4241f3967c28 1e81446d72942f50e0f0

- **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

SOLIDITY STATIC ANALYSIS

☒ Select all
 ☒ Autorun
 Run

Security

☒ Select Security

- ☒ Transaction origin:
'tx.origin' used
- ☒ Check-effects-interaction:
Potential reentrancy bugs
- ☒ Inline assembly:
Inline assembly used
- ☒ Block timestamp:
Can be influenced by miners
- ☒ Low level calls:
Should only be used by experienced devs
- ☒ Block hash:
Can be influenced by miners
- ☒ Selfdestruct:
Contracts using destructed contract can be broken

Gas & Economy

☒ Select Gas & Economy

- ☒ Gas costs:
Too high gas requirement of functions
- ☒ This on local calls:
Invocation of local functions via 'this'
- ☒ Delete dynamic array:
Use require/assert to ensure complete deletion
- ☒ For loop over dynamic array:
Iterations depend on dynamic array's size

SOLIDITY STATIC ANALYSIS

☒ Ether transfer in loop:
Transferring Ether in a for/while/do-while loop

ERC

☒ Select ERC

- ☒ ERC20:
'decimals' should be 'uint8'

Miscellaneous

☒ Select Miscellaneous

- ☒ Constant/View/Pure functions:
Potentially constant/view/pure functions
- ☒ Similar variable names:
Variable names are too similar
- ☒ No return:
Function with 'returns' not returning
- ☒ Guard conditions:
Ensure appropriate use of require/assert
- ☒ Result not used:
The result of an operation not used
- ☒ String length:
Bytes length != String length
- ☒ Delete from dynamic array:
'delete' leaves a gap in array
- ☒ Data truncated:
Division on int/uint values truncates the result

3- SOLIDITY UNIT TESTING

SOLIDITY UNIT TESTING

Test your smart contract in Solidity.

Select directory to load and generate test files.

Test directory:

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

✖

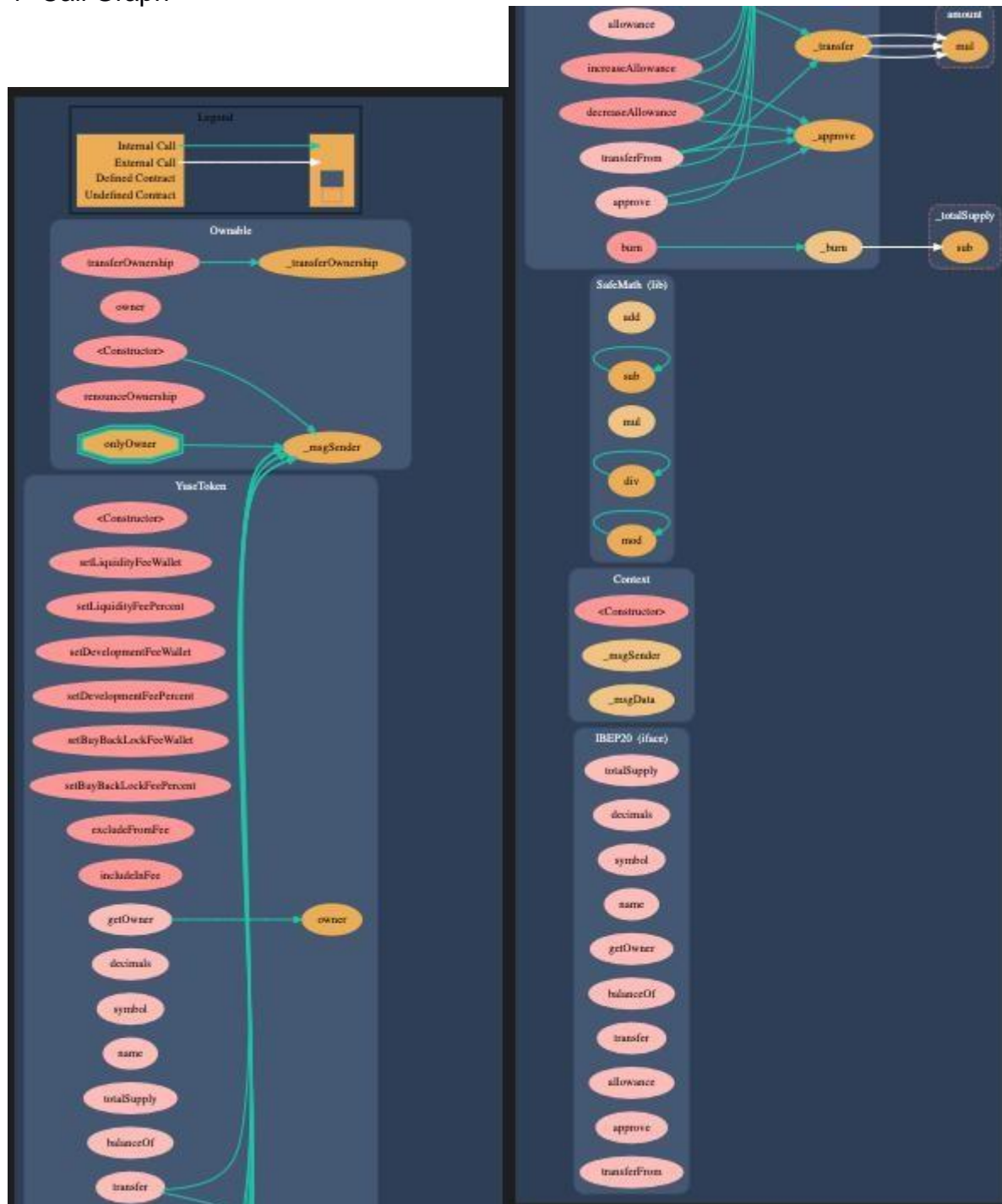
Result for tests/4_Ballot_test.sol

Passed: 2

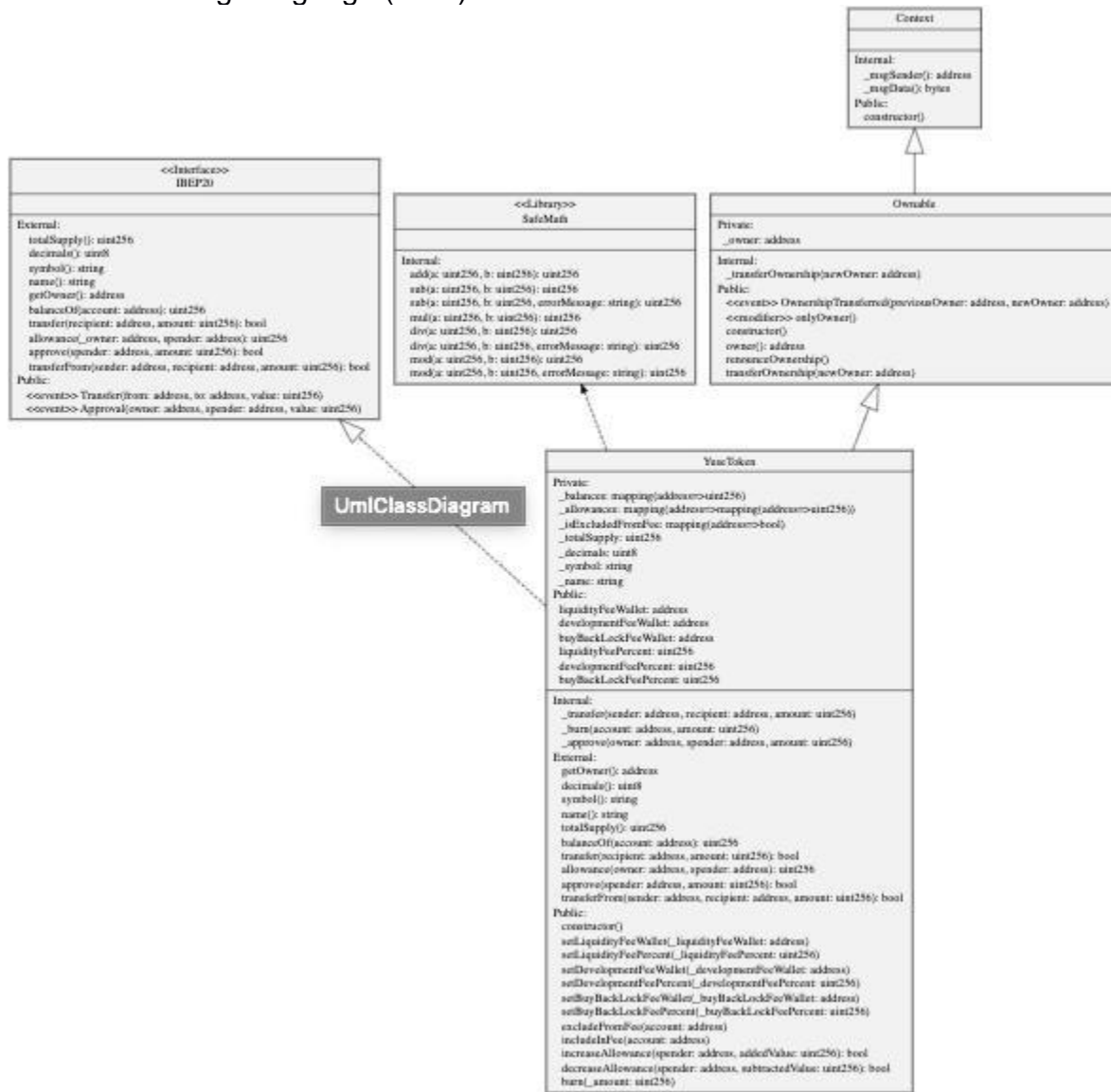
Failed: 0

Time Taken: 0.53s

4- Call Graph



Unified Modeling Language (UML)



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
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!
<Constructor>	Public !	u NO!
_msgSender	Internal Q	
_msgData	Internal Q	
SafeMath	Library	
add	Internal Q	
sub	Internal Q	
sub	Internal Q	
mul	Internal Q	
div	Internal Q	
div	Internal Q	
mod	Internal Q	
mod	Internal Q	

```

| **Ownable** | Implementation | Context |||
| L | <Constructor> | Public ! | u | NO! |
| L | owner | Public ! | | NO! |
| L | renounceOwnership | Public ! | u | onlyOwner |
| L | transferOwnership | Public ! | u | onlyOwner |
| L | _transferOwnership | Internal Q | u | |
| | | | |
| **Mr.Mint** | Implementation | IBEP20, Ownable |||
| L | <Constructor> | Public ! | u | NO! |
| L | setLiquidityFeeWallet | Public ! | u | onlyOwner |
| L | setLiquidityFeePercent | Public ! | u | onlyOwner |
| L | setDevelopmentFeeWallet | Public ! | u | onlyOwner |
| L | setDevelopmentFeePercent | Public ! | u | onlyOwner |
| L | setBuyBackLockFeeWallet | Public ! | u | onlyOwner |
| L | setBuyBackLockFeePercent | Public ! | u | onlyOwner |
| L | excludeFromFee | Public ! | u | onlyOwner |
| L | includeInFee | Public ! | u | onlyOwner |
| L | getOwner | External ! | | NO! |
| L | decimals | External ! | | NO! |
| L | symbol | External ! | | NO! |
| L | name | External ! | | NO! |
| L | totalSupply | External ! | | NO! |
| L | balanceOf | External ! | | NO! |
| L | transfer | External ! | u | NO! |
| L | allowance | External ! | | NO! |
| L | approve | External ! | u | NO! |
| L | transferFrom | External ! | u | NO! |
| L | increaseAllowance | Public ! | u | NO! |
| L | decreaseAllowance | Public ! | u | NO! |
| L | _transfer | Internal Q | u | |
| L | _burn | Internal Q | u | |
| L | _approve | Internal Q | u | |
| L | 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 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