

# Smart Contract Security Audit V1

## The Game block platform

<https://gameblock.mobi/>

2/11/2021



<https://saferico.com/>

[business@saferico.com](mailto:business@saferico.com)

[https://t.me/SFI\\_ANN](https://t.me/SFI_ANN)

—

# Table of Contents

## **Table of Contents**

### **Background**

### **Project Information**

- Token Information
- Chaos Token Distribution
- Contract Interaction Details
- 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

### **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

- **Website:** <https://gameblock.mobi/>
- **Twitter:** [https://twitter.com/the\\_Game\\_Block](https://twitter.com/the_Game_Block)
- **Telegram group:** [https://t.me/gameblock\\_discus](https://t.me/gameblock_discus)
- **Medium:** <https://gameblock-chaos.medium.com/>
- **Poocoin:** <https://poocoin.app/tokens/0xd90791544905d4239f99e70f66e340c9863bc036>
- **Platform:** Binance Smart Chain
- **Contract Address:** 0xd90791544905d4239F99E70F66e340C9863BC036

**CHAOS Token** (\$CHAOS) is the Game Block platform in-game reward token. All players will be rewarded with \$CHAOS Token in the games played.

## Token Information

- Name: CHAOS
- Total Supply: 100,000,000,000
- Holders: 255 address
- Total transactions: 1522

## Contracts address deployed to test net (ETH,BSC)

Chaos token (CHAOS) contract on testnet.bsc (BSC Test Net)

<https://testnet.bscscan.com/address/0x4b747151c3438033c29e52ae260e230054b08f85>

Chaos token (CHAOS) contract on Kovan (ETH Test Net)

<https://kovan.etherscan.io/address/0xca5c651c81dee354bb254daafea912892ac9f3ff>

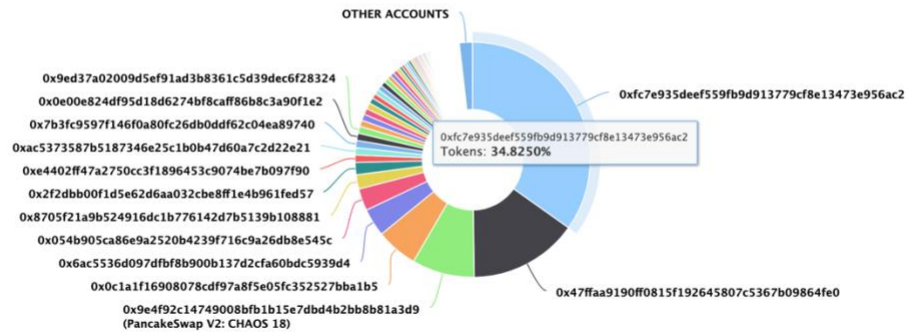
# Chaos Token Distribution

The top 100 holders collectively own 98.20% (98,197,635,117.06 Tokens) of Chaos Token

Token Total Supply: 100,000,000,000.00 Token | Total Token Holders: 256

Chaos Token Top 100 Token Holders

Source: BscScan.com

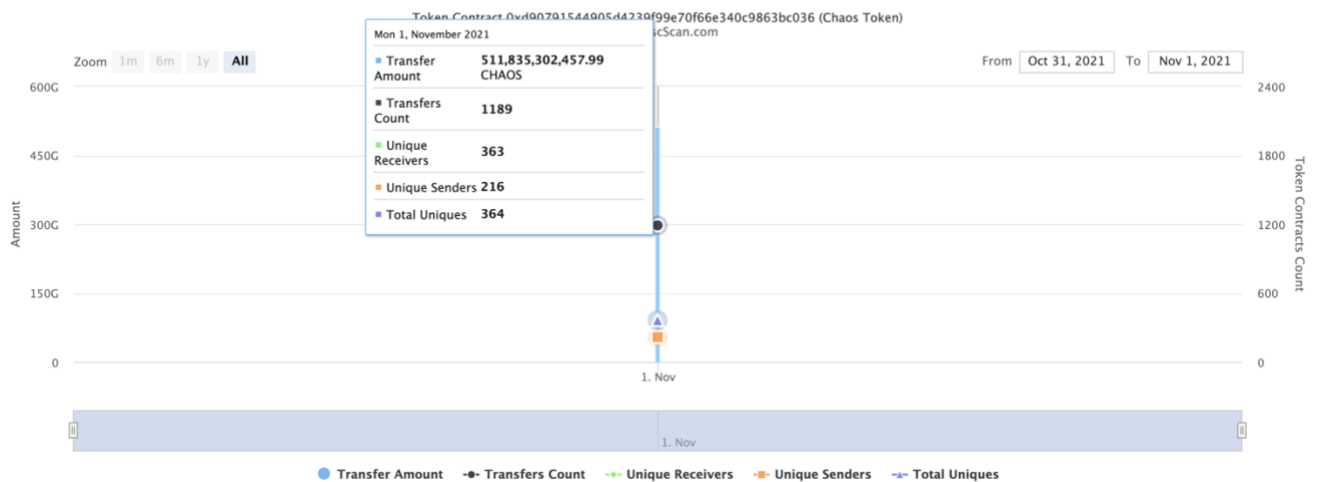


## Contract Interaction Details

### Token Contract Overview

Time Series: Token Contract Overview

Mon 1, Nov 2021 - Mon 1, Nov 2021



## Executive Summary

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

Well Secured	
<b>Secured</b>	✓
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, 1 high, 0 medium, 2 low, 1 very low-level issues and 0 notes in all solidity files of the contract

The files:

Chaos.sol

# File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
Chaos.sol	ada067995068398e05682e138b4c3635c59af43a73698987dfcf57bc49083708	0xd90791544905d4239F99E70F66e340C9863BC036

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

Function	Test Result	Type/Return Type	Score
owner	✓	Read/public	Passed
onlyOwner	✓	private	Passed
renounceOwnership	✓	Write / public	Passed
transferOwnership	✓	Write / public	Passed

- Contract: **BEP20Token**
- Inherit: Context, IBEP20, Ownable
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
name	✓	Read / private	Passed
symbol	✓	Read / private	Passed
decimals	✓	Read / private	Passed
totalSupply	✓	Read / private	Passed
balanceOf	✓	Read / private	Passed
transfer	✓	Write / public	Passed
allowance	✓	Read/public	Passed
approve	✓	Write / public	Passed
TransferFrom	✓	Write / public	Passed
increaseAllowance	✓	Write / public	Passed
decreaseAllowance	✓	Write / public	Passed
getOwer	✓	Read / public	Passed
multiTransferSingleValue	✓	Write / public	Passed
mint	✓	Write / public	Passed

_approve	✓	private	<b>Passed</b>
_transfer	✓	private	<b>Passed</b>
_mint	✓	private	<b>Passed</b>
_burn	✓	private	<b>Passed</b>
_burnFrom	✓	private	<b>Passed</b>



# 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 critical severity vulnerabilities were found.

### High:

#### Issue #1

In detail

One of the major dangers of calling external contracts is that they can take over the control flow. In the reentrancy attack (a.k.a. recursive call attack), a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways.

```
function sub(uint256 a, uint256 b) internal pure returns (uint256) {  
    return sub(a, b, "SafeMath: subtraction overflow");  
}
```

### Medium:

No Medium severity vulnerabilities were found.

### Low:

#### Issue #1

In detail

Using an outdated compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version.

```
pragma solidity ^0.5.16;
```

#### Issue #2

In detail

An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum size of a type. For instance if a number is stored in the uint8 type, it means that the number is stored in a 8 bits unsigned number ranging from 0 to  $2^8-1$ . In computer programming, an integer overflow occurs when an arithmetic operation attempts to create a numeric value that is outside of the range that can be represented with a given number of bits – either larger than the maximum or lower than the minimum representable value.

```
uint256 c = a + b;
```

### Very Low:

#### Issue #1. Out of gas:

Approve given more allowance: -

=> I have found that in approve function user can give more allowance to a user beyond their balance.

=> It is necessary to check that user can give allowance less or equal to their amount.

=> There is no validation about user balance. So, it is good to check that a user not set approval wrongly.

- Function: - `_approve`
  - Here you can check that amount is not more than balance of owner.

```
function _approve(address owner, address spender, uint256 amount)
internal {
    require(owner != address(0), "BEP20: approve from the zero address");
    require(spender != address(0), "BEP20: approve to the zero address");

    _allowances[owner][spender] = amount;
    emit Approval(owner, spender, amount);
}
```

Unchecked return value or response: -

- => I have found that you are transferring fund to address using a transfer method.
- => It is always good to check the return value or response from a function call.
- => Here are some functions where you forgot to check a response.

### Notes:

No Notes were found.

# Automatic Testing

## 1-Check for security

ada067995068398e05682e138b4c3635c59af43a73698987dfcf57bc490837...

File: Chaos.sol | Language: solidity | Size: 18579 bytes | Date: 2021-11-02T08:30:11.009Z

Critical	High	Medium	Low	Note
0	0	0	0	0



## 2- SOLIDITY STATIC ANALYSIS

### SOLIDITY STATIC ANALYSIS

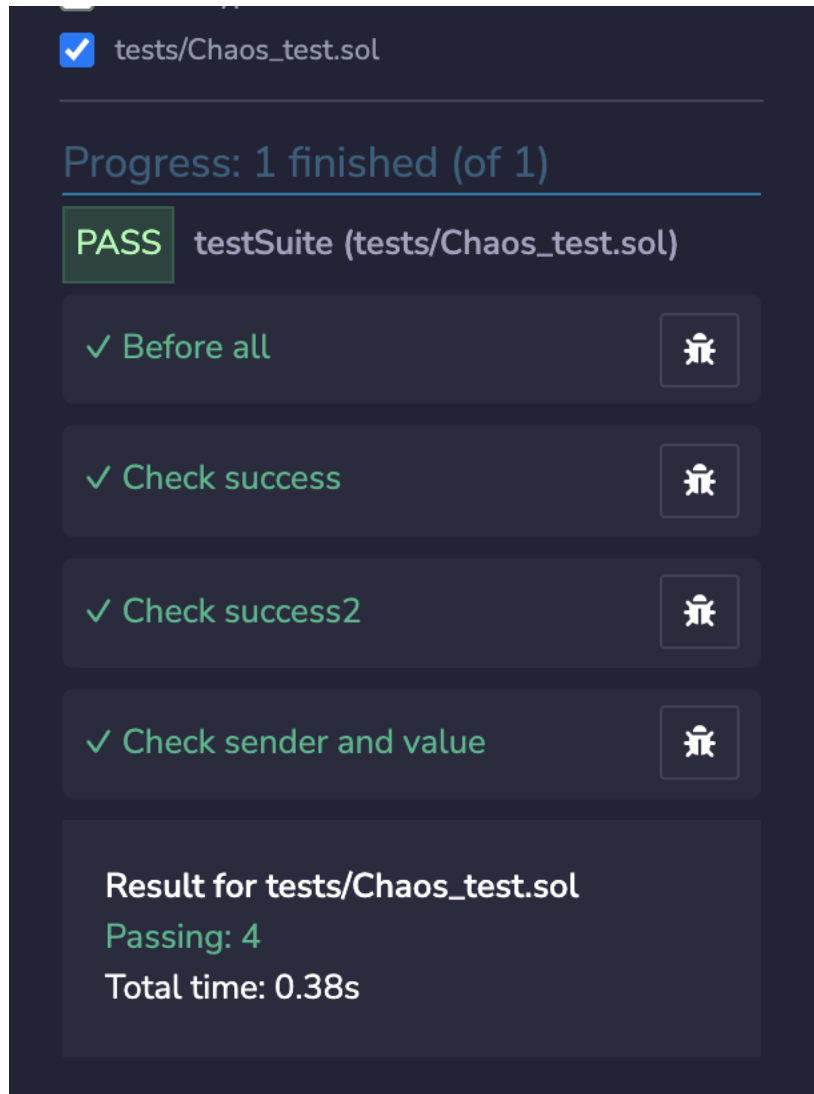
- ▼ ERC
  - ☒ Select ERC
    - ☒ ERC20:  
'decimals' should be 'uint8'
- ▼ Miscellaneous
  - ☒ Select Miscellaneous
    - ☒ Constant/View/Pure functions:  
Potentially constant/view/pure functions
    - ☒ Similar variable names:  
Variable names are too similar
    - ☒ No return:  
Function with 'returns' 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

### SOLIDITY STATIC ANALYSIS

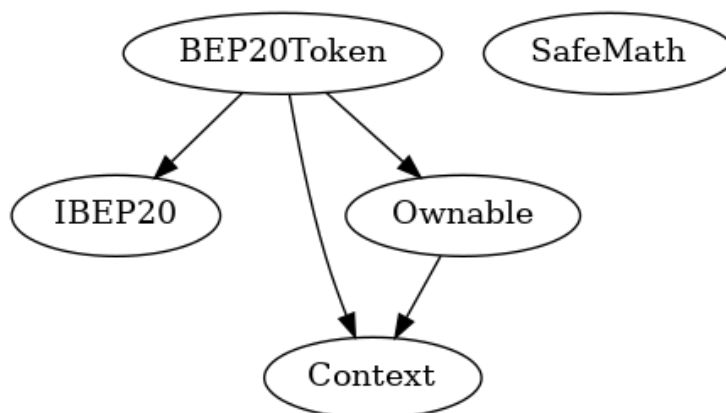
☒ Select all ☒ Autorun Run

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

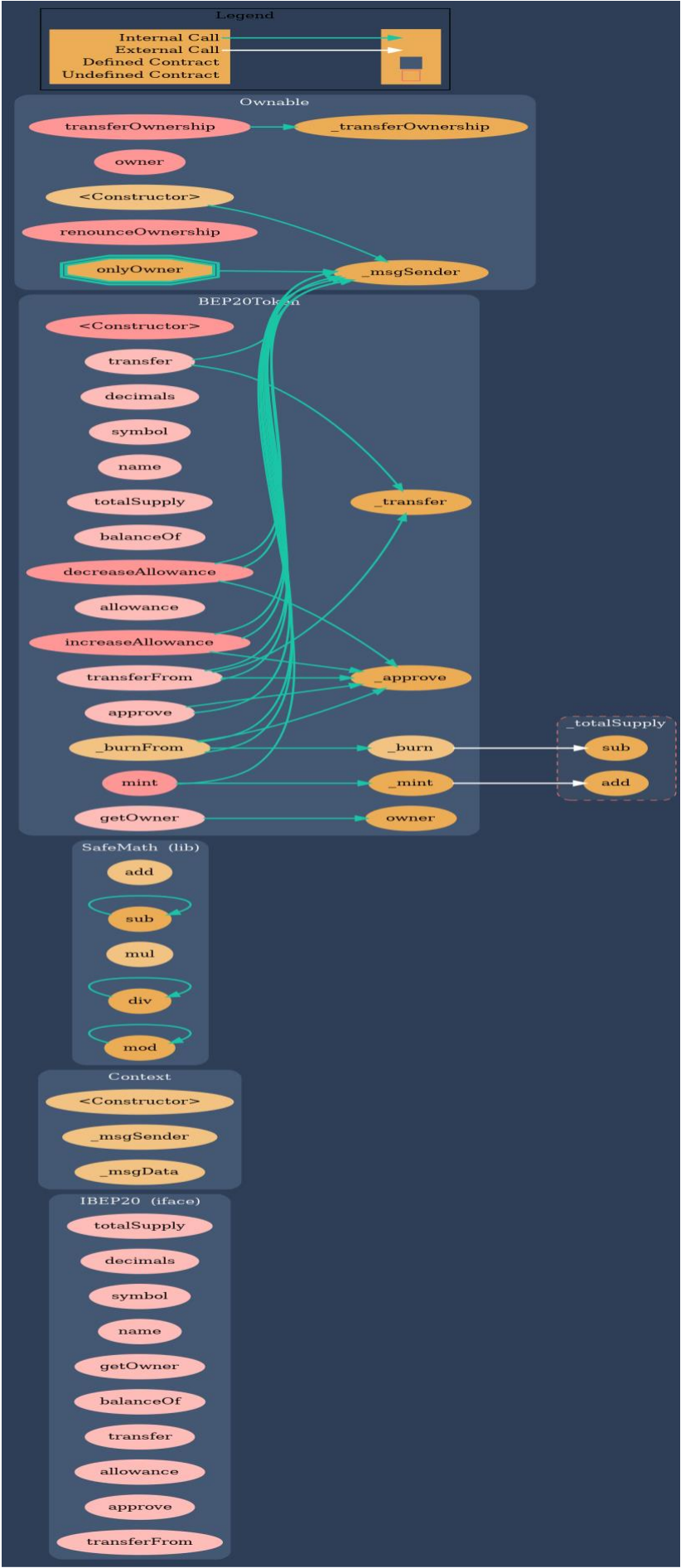
### 3- SOLIDITY UNIT TESTING



### #Inheritance graph



### #Call graph



# Automatic general report

## Files Description Table
























































File Name	SHA-1 Hash
----- -----	
/Users/macbook/Desktop/smart contracts/Chaos.sol	5dd860d6c96f8ca25f6e02200883cda9acd99d5f

## Contracts Description Table

Contract	Type	Bases		
:----- :----- :----- :----- :-----				
└	<b>Function Name</b>	<b>Visibility</b>	<b>Mutability</b>	<b>Modifiers</b>
<b>IBEP20</b>	Interface			
└	totalSupply	External	┐	NO┐
└	decimals	External	┐	NO┐
└	symbol	External	┐	NO┐
└	name	External	┐	NO┐
└	getOwner	External	┐	NO┐
└	balanceOf	External	┐	NO┐
└	transfer	External	┐	NO┐
└	allowance	External	┐	NO┐
└	approve	External	┐	NO┐
└	transferFrom	External	┐	NO┐
<b>Context</b>	Implementation			
└	<Constructor>	Internal	┐	
└	_msgSender	Internal	┐	
└	_msgData	Internal	┐	
<b>SafeMath</b>	Library			
└	add	Internal	┐	
└	sub	Internal	┐	
└	sub	Internal	┐	
└	mul	Internal	┐	
└	div	Internal	┐	
└	div	Internal	┐	
└	mod	Internal	┐	





```

|  ↳ | mod | Internal  |  || |
|||||
| **Ownable** | Implementation | Context |||
|  ↳ | <Constructor> | Internal  |  |  ||
|  ↳ | owner | Public  |  | NO  |
|  ↳ | renounceOwnership | Public  |  |  | onlyOwner |
|  ↳ | transferOwnership | Public  |  |  | onlyOwner |
|  ↳ | _transferOwnership | Internal  |  |  ||
|||||
| **BEP20Token** | Implementation | Context, IBEP20, Ownable |||
|  ↳ | <Constructor> | Public  |  |  | NO  |
|  ↳ | 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  |
|  ↳ | mint | Public  |  |  | onlyOwner |
|  ↳ | _transfer | Internal  |  |  ||
|  ↳ | _mint | Internal  |  |  ||
|  ↳ | _burn | Internal  |  |  ||
|  ↳ | _approve | Internal  |  |  ||
|  ↳ | _burnFrom | Internal  |  |  ||

```

## 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.
- ✓ Not many high severity issues were found.
- ✓ 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 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.