CALO

Smart Contract Audit Report

SYSFIXED

19 Dec 2021

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
CALO	CALO	Binance Smart Chain

Addresses

Contract address	0xb6b91269413b6b99242b1c0bc611031529999999
Contract deployer address	0xad80314c566Be4Bacbb3992F844faE914D9Fd31d

Project Website

https://calo.run/

Codebase

https://bscscan.com/address/0xb6b91269413b6b99242b1c0bc611031529999999#code

SUMMARY

Calo is a healthy application based on blockchain technology. Work out every day, burn your calories, participate in challenges, and earn money. CALO Token is a blockchain-based platform crypto currency with limited supply. CALO Token is the main currency of the Calo ecosystem.

Contract Summary

Documentation Quality

CALO provides a very good documentation with standard of solidity base code.

- The technical description is provided clearly and structured and also dont have any high risk issue.

Code Quality

The Overall quality of the basecode is standard.

- Standard solidity basecode and rules are already followed by CALO with the discovery of several low issues.

Test Coverage

Test coverage of the project is 100% (Through Codebase)

Audit Findings Summary

- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 34, 50, 67, 68, 83, 97, 111, 128, 143, 144, 162, 179, 201, 225, 249, 452, 543, 543, 927, 930, 943, 946, 959, 962, 998, 1001, 1022, 1025, 1400, 1400, 1400 and 1400.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 470.
- SWC-110 SWC-123 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 928, 931, 944, 947, 960, 963, 999, 1002, 1023 and 1026.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 537, 538, 539, 540 and 541.

CONCLUSION

We have audited the CALO project released on December 2021 to discover issues and identify potential security vulnerabilities in CALO Project. This process is used to find technical issues and security loopholes which might be found in the smart contract.

The security audit report provides satisfactory results with low-risk issues.

The issues found in the CALO smart contract code do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some arithmetic operation issues, a floating pragma is set, x.origin as a part of authorization control and out-of-bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value. Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you know what you are doing. tx.origin is a global variable in Solidity that returns the account's address that sent the transaction. Using the variable for authorization could make a contract vulnerable if an authorized account calls into a malicious contract. A call could be made to the vulnerable contract that passes the authorization check since tx.origin returns the original sender of the transaction, which in this case is the authorized account. tx.origin should not be used for authorization. Use msg.sender instead.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	PASS
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
Unprotected Ether Withdrawal	SWC-105	Due to missing or insufficient access controls, malicious parties can withdraw from the contract.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Uninitialized Storage Pointer	SWC-109	Uninitialized local storage variables can point to unexpected storage locations in the contract.	PASS
Assert Violation	SWC-110 SWC-123	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS

DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	ISSUE FOUND
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique ID	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	PASS
Write to Arbitrary Storage Location	SWC-124	The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations.	PASS
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS
Insufficient Gas Griefing	SWC-126	Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract.	PASS
Arbitrary Jump Function	SWC-127	As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value.	PASS

Typographical Error	SWC-129	A typographical error can occur for example when the intent of a defined operation is to sum a number to a variable.	PASS
Override control character	SWC-130	Malicious actors can use the Right-To-Left-Override unicode character to force RTL text rendering and confuse users as to the real intent of a contract.	PASS
Unused variables	SWC-131 SWC-135	Unused variables are allowed in Solidity and they do not pose a direct security issue.	PASS
Unexpected Ether balance	SWC-132	Contracts can behave erroneously when they strictly assume a specific Ether balance.	PASS
Hash Collisions Variable	SWC-133	Using abi.encodePacked() with multiple variable length arguments can, in certain situations, lead to a hash collision.	PASS
Hardcoded gas amount	SWC-134	The transfer() and send() functions forward a fixed amount of 2300 gas.	PASS
Unencrypted Private Data	SWC-136	It is a common misconception that private type variables cannot be read.	PASS

SMART CONTRACT ANALYSIS

Started	Saturday Dec 18 2021 21:44:22 GMT+0000 (Coordinated Universal Time)
Finished	Sunday Dec 19 2021 00:24:14 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Token.sol

Detected Issues

ID	Title	Severity	Status
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "%" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "%" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "%" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 34

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
33   {
34   uint256 c = a + b;
35   if (c < a) return (false, 0);
36   return (true, c);
37   }
38
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 50

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
49   if (b > a) return (false, 0);
50   return (true, a - b);
51   }
52
53   /**
54
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 67

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
66   if (a == 0) return (true, 0);
67   uint256 c = a * b;
68   if (c / a != b) return (false, 0);
69   return (true, c);
70   }
71
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 68

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
67   uint256 c = a * b;
68   if (c / a != b) return (false, 0);
69   return (true, c);
70   }
71
72
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 83

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
82   if (b == 0) return (false, 0);
83   return (true, a / b);
84   }
85
86   /**
87
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 97

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
96   if (b == 0) return (false, 0);
97   return (true, a % b);
98   }
99
100   /**
101
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 111

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
110   function add(uint256 a, uint256 b) internal pure returns (uint256) {
111   uint256 c = a + b;
112   require(c >= a, "SafeMath: addition overflow");
113   return c;
114   }
115
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 128

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
127   require(b <= a, "SafeMath: subtraction overflow");
128   return a - b;
129   }
130
131   /**
132
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 143

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
142   if (a == 0) return 0;
143   uint256 c = a * b;
144   require(c / a == b, "SafeMath: multiplication overflow");
145   return c;
146   }
147
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 144

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
143   uint256 c = a * b;
144   require(c / a == b, "SafeMath: multiplication overflow");
145   return c;
146   }
147
148
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 162

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
161   require(b > 0, "SafeMath: division by zero");
162   return a / b;
163   }
164
165   /**
166
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 179

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
178   require(b > 0, "SafeMath: modulo by zero");
179   return a % b;
180   }
181
182   /**
183
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 201

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
200   require(b <= a, errorMessage);
201   return a - b;
202   }
203
204   /**
205
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 225

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
224   require(b > 0, errorMessage);
225   return a / b;
226   }
227
228   /**
229
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 249

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
248   require(b > 0, errorMessage);
249   return a % b;
250   }
251   }
252
253
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 452

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
451   _owner = address(0);
452   _lockTime = block.timestamp + time;
453   emit OwnershipTransferred(_owner, address(0));
454   }
455
456
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 543

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
542   _mintable = true;
543   _numTokensSellToAddToLiquidity= (_pamount*1) / 10000; /** 0,01 % total supply */
544   }
545
546   /**
547
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 543

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
542   _mintable = true;
543   _numTokensSellToAddToLiquidity= (_pamount*1) / 10000; /** 0,01 % total supply */
544   }
545
546   /**
547
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 927

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
926   ) public onlyOwner {
927   for (uint256 index; index < newWhiteList.length; index++) {
928   whiteListSender[newWhiteList[index]] = true;
929   }
930   for (uint256 index; index < removedWhiteList.length; index++) {
931
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 930

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
929   }
930   for (uint256 index; index < removedWhiteList.length; index++) {
931   whiteListSender[removedWhiteList[index]] = false;
932   }
933   }
934
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 943

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
942   ) public onlyOwner {
943   for (uint256 index; index < newWhiteList.length; index++) {
944   whiteListReceiver[newWhiteList[index]] = true;
945   }
946   for (uint256 index; index < removedWhiteList.length; index++) {
947
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 946

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
945   }
946   for (uint256 index; index < removedWhiteList.length; index++) {
947   whiteListReceiver[removedWhiteList[index]] = false;
948   }
949   }
950
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 959

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
958   ) public onlyOwner {
959   for (uint256 index; index < newWhiteList.length; index++) {
960   blackList[newWhiteList[index]] = true;
961   }
962   for (uint256 index; index < removedWhiteList.length; index++) {
963
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 962

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
961   }
962   for (uint256 index; index < removedWhiteList.length; index++) {
963   blackList[removedWhiteList[index]] = false;
964   }
965   }
966
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 998

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
997   ) public onlyOwner {
998   for (uint256 index; index < newWhiteList.length; index++) {
999   whiteListBot[newWhiteList[index]] = true;
1000   }
1001   for (uint256 index; index < removedWhiteList.length; index++) {
1002
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1001

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1000   }
1001   for (uint256 index; index < removedWhiteList.length; index++) {
1002   whiteListBot[removedWhiteList[index]] = false;
1003   }
1004   }
1005
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1022

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1021   ) public onlyOwner {
1022   for (uint256 index; index < newWhiteList.length; index++) {
1023   whiteListPool[newWhiteList[index]] = true;
1024   }
1025   for (uint256 index; index < removedWhiteList.length; index++) {
1026
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1025

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1024   }
1025   for (uint256 index; index < removedWhiteList.length; index++) {
1026   whiteListPool[removedWhiteList[index]] = false;
1027   }
1028   }
1029
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1400

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1399
1400   uint256 public maxSupply = 500 * 10**6 * 10**18;
1401
1402   IUniswapV2Router02 public immutable uniswapV2Router;
1403   address public uniswapV2Pair;
1404
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1400

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1399
1400   uint256 public maxSupply = 500 * 10**6 * 10**18;
1401
1402   IUniswapV2Router02 public immutable uniswapV2Router;
1403   address public uniswapV2Pair;
1404
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1400

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1399
1400   uint256 public maxSupply = 500 * 10**6 * 10**18;
1401
1402   IUniswapV2Router02 public immutable uniswapV2Router;
1403   address public uniswapV2Pair;
1404
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1400

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1399
1400   uint256 public maxSupply = 500 * 10**6 * 10**18;
1401
1402   IUniswapV2Router02 public immutable uniswapV2Router;
1403   address public uniswapV2Pair;
1404
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 470

low SEVERITY

The current pragma Solidity directive is ""^0.7.4"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- Token.sol

Locations

```
469
470   pragma solidity ^0.7.4;
471
472   /**
473   * @dev Implementation of the {IERC20} interface.
474
```

SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 537

low SEVERITY

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source File

- Token.sol

Locations

```
536   _decimals = _pdecimals;
537   _feeWallet = tx.origin;
538   _mint(tx.origin, _pamount);
539   whiteListSender[tx.origin] = true;
540   whiteListReceiver[tx.origin] = true;
541
```

SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 538

low SEVERITY

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source File

- Token.sol

Locations

```
537   _feeWallet = tx.origin;
538   _mint(tx.origin, _pamount);
539   whiteListSender[tx.origin] = true;
540   whiteListReceiver[tx.origin] = true;
541   whiteListBot[tx.origin] = true;
542
```

SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 539

low SEVERITY

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source File

- Token.sol

Locations

```
538   _mint(tx.origin, _pamount);
539   whiteListSender[tx.origin] = true;
540   whiteListReceiver[tx.origin] = true;
541   whiteListBot[tx.origin] = true;
542   _mintable = true;
543
```

SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 540

low SEVERITY

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source File

- Token.sol

Locations

```
539   whiteListSender[tx.origin] = true;
540   whiteListReceiver[tx.origin] = true;
541   whiteListBot[tx.origin] = true;
542   _mintable = true;
543   _numTokensSellToAddToLiquidity= (_pamount*1) / 10000; /** 0,01 % total supply */
544
```

SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 541

low SEVERITY

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source File

- Token.sol

Locations

```
540   whiteListReceiver[tx.origin] = true;
541   whiteListBot[tx.origin] = true;
542   _mintable = true;
543   _numTokensSellToAddToLiquidity= (_pamount*1) / 10000; /** 0,01 % total supply */
544   }
545
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 928

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
927   for (uint256 index; index < newWhiteList.length; index++) {
928   whiteListSender[newWhiteList[index]] = true;
929   }
930   for (uint256 index; index < removedWhiteList.length; index++) {
931   whiteListSender[removedWhiteList[index]] = false;
932
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 931

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
930   for (uint256 index; index < removedWhiteList.length; index++) {
931   whiteListSender[removedWhiteList[index]] = false;
932   }
933   }
934
935
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 944

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
943   for (uint256 index; index < newWhiteList.length; index++) {
944   whiteListReceiver[newWhiteList[index]] = true;
945   }
946   for (uint256 index; index < removedWhiteList.length; index++) {
947   whiteListReceiver[removedWhiteList[index]] = false;
948
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 947

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
946   for (uint256 index; index < removedWhiteList.length; index++) {
947   whiteListReceiver[removedWhiteList[index]] = false;
948   }
949   }
950
951
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 960

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
959   for (uint256 index; index < newWhiteList.length; index++) {
960   blackList[newWhiteList[index]] = true;
961   }
962   for (uint256 index; index < removedWhiteList.length; index++) {
963   blackList[removedWhiteList[index]] = false;
964
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 963

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
962   for (uint256 index; index < removedWhiteList.length; index++) {
963   blackList[removedWhiteList[index]] = false;
964   }
965   }
966
967
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 999

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
998   for (uint256 index; index < newWhiteList.length; index++) {
999   whiteListBot[newWhiteList[index]] = true;
1000   }
1001   for (uint256 index; index < removedWhiteList.length; index++) {
1002   whiteListBot[removedWhiteList[index]] = false;
1003
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1002

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
1001   for (uint256 index; index < removedWhiteList.length; index++) {
1002   whiteListBot[removedWhiteList[index]] = false;
1003   }
1004   }
1005
1006
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1023

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
1022   for (uint256 index; index < newWhiteList.length; index++) {
1023   whiteListPool[newWhiteList[index]] = true;
1024   }
1025   for (uint256 index; index < removedWhiteList.length; index++) {
1026   whiteListPool[removedWhiteList[index]] = false;
1027
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1026

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
1025   for (uint256 index; index < removedWhiteList.length; index++) {
1026   whiteListPool[removedWhiteList[index]] = false;
1027   }
1028   }
1029
1030
```

DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to, or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without Sysfixed's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Sysfixed to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model, or legal compliance.

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 us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

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 Sysfixed and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Sysfixed) owe no duty of care.

ABOUT US

Sysfixed is a blockchain security certification organization established in 2021 with the objective to provide smart contract security services and verify their correctness in blockchain-based protocols. Sysfixed automatically scans for security vulnerabilities in Ethereum and other EVM-based blockchain smart contracts. Sysfixed a comprehensive range of analysis techniques—including static analysis, dynamic analysis, and symbolic execution—can accurately detect security vulnerabilities to provide an in-depth analysis report. With a vibrant ecosystem of world-class integration partners that amplify developer productivity, Sysfixed can be utilized in all phases of your project's lifecycle. Our team of security experts is dedicated to the research and improvement of our tools and techniques used to fortify your code.