Mar. 15th, 2023

ZAN

Security Assessment

LabsGroup

—

Express Service

Table of Contents

1. Overview

1.1. Project Summary

Project Name	LabsGroup
Platform	BSC Network
Language	Solidity
Code Repository	audited codebase : https://testnet.bscscan.com/address/0x013650Adeb00583c84FF955E5F17f5C6616D5fEA#code updated codebase : https://bscscan.com/address/0x510Ca7D22A84599e7d0f15F09E674056a6255389#code

1.2. Assessment Summary

Delivery Date	Mar. 15th, 2023
Audit Methodology	Static Analysis, Formal Verification

1.3. Assessment Scope

ID	File
01	LabsGroup.sol
02	interface/IBEP20.sol

2. Checklist

2.1. Code Security

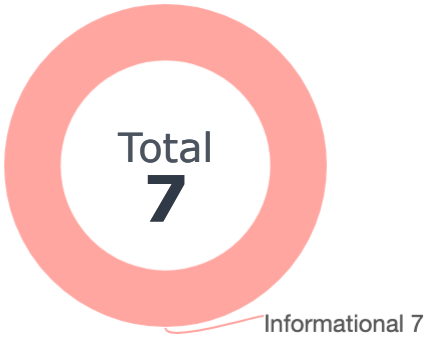
Reentrancy	Integer Overflow
DelegateCall	Frozen Money
Broken Functionalities	Arbitrary External Call
Unauthenticated Storage Access	Right-To-Left-Override Character
Misuse of tx.origin	Weak Sources of Randomness
Diamond Inheritance	Unchecked Call Return Value
VarType Deduction	Strict Balance Checking
Uninitialized Variables	Externally Controlled Array Length
Division Before Multiplication	Shadowing State Variables
Affected by Compiler Bug	MsgValue In Loop
DelegateCall In Loop	Incorrect EIP712 Signature Encode
Incorrect Shift In Assembly	

2.2. Optimization Suggestion

2.2.1 Code Convention

Shadowing Local Variables	Risk of Low-Level Calls
ERC20/ERC721/ERC777 Token Standard	Compiler Version Security
Code Style	Risk of External Calls
Return Value Specifications	Revert Specifications
Error Message Specifications	Reference Variable Specifications
Import Specifications	Function Visibility Specifications
Constant Specifications	Global Variable Dependency
Constructor Validation	Array Length Manipulation
State Write Specifications	Event Specifications
Usage of Incorrect Operator	Modifier Specifications
Risk of Signature Replay	Usage of SafeMath Library
Risks of Using Assembly	Loop Specifications
Inheritance Specifications	Input Validation

3. Findings

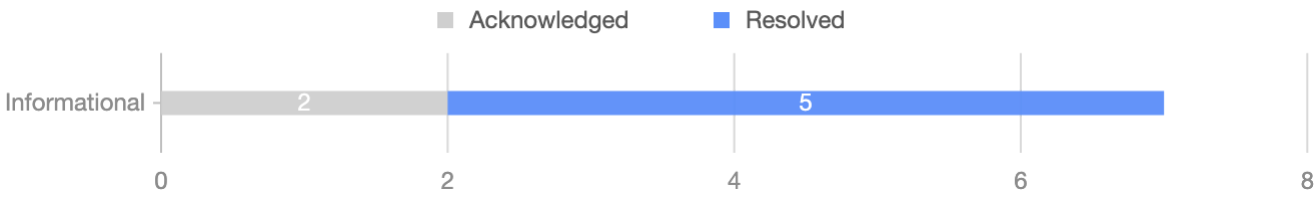


Code Security

Total: 0, no vulnerabilities.

Optimization Suggestion

Total: 7, Informational: 7



ID	Title	Category	Severity	Status
I-01	Incompatible with BEP20 Standard	Optimization Suggestion	Informational	Resolved
I-02	Floating Pragma	Optimization Suggestion	Informational	Resolved
I-03	Long String in require	Optimization Suggestion	Informational	Acknowledged
I-04	Code layout Conventions	Optimization Suggestion	Informational	Acknowledged
I-05	Unused Internal Function	Optimization Suggestion	Informational	Resolved
I-06	Variables Can Be Constants	Optimization Suggestion	Informational	Resolved
I-07	Function Visibility Can Be External	Optimization Suggestion	Informational	Resolved

I-01 | Incompatible with BEP20 Standard

Category : Optimization Suggestion

Severity : Informational

File Location : LabsGroup.sol

Description

In BEP20 standard, there is a function getOwner() which is an extended method of EIP20. Tokens which don’t implement this method will never flow across the BNB Beacon Chain and BNB Smart Chain. But contract LabsGroup doesn't implement the function getOwner().

Recommendation

It is recommended to implement function getOwner() as follows:

```
function getOwner() external view returns (address) {
        return owner();
}
```

Alleviation

The project team added the getOwner() function. The issue was resolved in https://bscscan.com/address/0x510Ca7D22A84599e7d0f15F09E674056a6255389#code.

I-02 | Floating Pragma

Category : Optimization Suggestion

Severity : Informational

File Location : LabsGroup.sol:2, interface/IBEP20.sol:2

Description

An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version.
This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers.
This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

```
2 pragma solidity ^0.8.0;
```

Recommendation

Lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

Alleviation

The project team followed our advice and updated the code in https://bscscan.com/address/0x510Ca7D22A84599e7d0f15F09E674056a6255389#code.

I-03 | Long String in require

Category : Optimization Suggestion

Severity : Informational

File Location : LabsGroup.sol: 109, 150, 172, 173, 176, 196, 245, 246

Description

Compared to using function require, using CustomError is more gas efficient, especially when the string parameter in the require function exceeds 32 bytes. Because the error message described using CustomError is only compiled into four bytes.

```
1  // loc:109
2  require(currentAllowance >= amount, "BEP20: transfer amount exceeds allowance"
   );
3  // loc:150
4  require(currentAllowance >= subtractedValue,
   "BEP20: decreased allowance below zero");
5  // loc:172
6  require(sender != address(0), "BEP20: transfer from the zero address");
7  // loc:173
8  require(recipient != address(0), "BEP20: transfer to the zero address");
9  // loc:176
10 require(senderBalance >= amount, "BEP20: transfer amount exceeds balance");
11 // loc:196
12 require(account != address(0), "BEP20: mint to the zero address");
13 // loc:245
14 require(owner != address(0), "BEP20: approve from the zero address");
15 // loc:246
16 require(spender != address(0), "BEP20: approve to the zero address");
```

Recommendation

When reverting, it is recommended to use CustomError instead of ordinary strings to describe the error message. Examples are as follows:

```
1  error ZeroAddress(address addr);
2
3  function func(address sender) public {
4    if (sender == address(0))
5        revert ZeroAddress(sender);
6    ......
7  }
```

Alleviation

The project team acknowledged the issue.

I-04 | Code layout Conventions

Category : Optimization Suggestion

Severity : Informational

File Location : LabsGroup.sol, interface/IBEP20.sol

Description

In the solidity document(https://docs.soliditylang.org/en/v0.8.17/style-guide.html), there are the following conventions for code layout:

Layout contract elements in the following order: 1. Pragma statements, 2. Import statements, 3. Interfaces, 4. Libraries, 5. Contracts.

Inside each contract, library or interface, use the following order: 1. Type declarations, 2. State variables, 3. Events, 4. Modifiers, 5. Functions.

Functions should be grouped according to their visibility and ordered: 1. constructor, 2. receive function (if exists), 3. fallback function (if exists), 4. external, 5. public, 6. internal, 7. private.

Recommendation

Recommended to Follow Code layout Conventions.

Alleviation

The project team acknowledged the issue.

I-05 | Unused Internal Function

Category : Optimization Suggestion

Severity : Informational

File Location : LabsGroup.sol: 217-229

Description

Internal function _burn is defined but not called by the contract.

```
217  function _burn(address account, uint256 amount) internal {
218      require(account != address(0), "BEP20: burn from the zero address");
219
220
221      uint256 accountBalance = balanceOf(account);
222      require(accountBalance >= amount, "BEP20: burn amount exceeds balance");
223      unchecked {
224          _balances[account] = accountBalance - amount;
225          // Overflow not possible: amount <= accountBalance <= totalSupply.
226          _totalSupply -= amount;
227      }
228      emit Transfer(account, address(0), amount);
229  }
```

Recommendation

It is recommended to remove function _burn in the contract.

Alleviation

The project team followed our advice and updated the code in https://bscscan.com/address/0x510Ca7D22A84599e7d0f15F09E674056a6255389#code.

I-06 | Variables Can Be Constants

Category : Optimization Suggestion

Severity : Informational

File Location : LabsGroup.sol: 14, 15, 16

Description

There are unchanging state variables _name, _symbol and _decimals can be declared as constant to save gas.

```
14  string private _name;
15  string private _symbol;
16  uint8 private _decimals;
```

Recommendation

Change variables _name, _symbol and _decimals to constant.

Alleviation

The project team followed our advice and updated the code in https://bscscan.com/address/0x510Ca7D22A84599e7d0f15F09E674056a6255389#code.

I-07 | Function Visibility Can Be External

Category : Optimization Suggestion

Severity : Informational

File Location : LabsGroup.sol: 128-131, 147-155

Description

Following functions in contract LabsGroup will never be called by the contract and should be modified to external functions. External functions are more efficient than public functions.

```
128  function increaseAllowance(address spender, uint256 addedValue) public returns
     (bool)
     {
129      _approve(_msgSender(), spender, allowance(_msgSender(), spender) +
     addedValue);
130      return true;
131  }


147  function decreaseAllowance(address spender, uint256 subtractedValue) public
     returns (bool)
     {
148      address owner = _msgSender();
149      uint256 currentAllowance = allowance(owner, spender);
150      require(currentAllowance >= subtractedValue,
     "BEP20: decreased allowance below zero");
151      unchecked {
152          _approve(owner, spender, currentAllowance - subtractedValue);
153      }
154      return true;
155  }
```

Recommendation

Functions that are not called in the contract should be declared as external.

Alleviation

The project team followed our advice and updated the code in https://bscscan.com/address/0x510Ca7D22A84599e7d0f15F09E674056a6255389#code.

4. Disclaimer

No description, statement, recommendation or conclusion in this report shall be construed as endorsement, affirmation or confirmation of the project. The security assessment is limited to the scope of work as stipulated in the Statement of Work.

This report is prepared in response to source code, and based on the attacks and vulnerabilities in the source code that already existed or occurred before the date of this report, excluding any new attacks or vulnerabilities that exist or occur after the date of this report. The security assessment are solely based on the documents and materials provided by the customer, and the customer represents and warrants documents and materials are true, accurate and complete.

CONSULTANT DOES NOT MAKE AND HEREBY DISCLAIMS ANY REPRESENTATIONS OR WARRANTIES OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, REGARDING THE SERVICES, DELIVERABLES, OR ANY OTHER MATTER PERTAINING TO THIS REPORT.

CONSULTANT SHALL NOT BE RESPONSIBLE FOR AND HEREBY DISCLAIMS MERCHANTABILITY, FITNESS FOR PURPOSE, TITLE, NON-INFRINGEMENT OR NON-APPROPRIATION OF INTELLECTUAL PROPERTY RIGHTS OF A THIRD PARTY, SATISFACTORY QUALITY, ACCURACY, QUALITY, COMPLETENESS, TIMELINESS, RESPONSIVENESS, OR PRODUCTIVITY OF THE SERVICES OR DELIVERABLES.

CONSULTANT EXCLUDES ANY WARRANTY THAT THE SERVICES AND DELIVERABLES WILL BE UNINTERRUPTED, ERROR FREE, FREE OF SECURITY DEFECTS OR HARMFUL COMPONENTS, REVEAL ALL SECURITY VULNERABILITIES, OR THAT ANY DATA WILL NOT BE LOST OR CORRUPTED.

CONSULTANT SHALL NOT BE RESPONSIBLE FOR (A) ANY REPRESENTATIONS MADE BY ANY PERSON REGARDING THE SUFFICIENCY OR SUITABILITY OF SERVICES AND DELIVERABLES IN ANY ACTUAL APPLICATION, OR (B) WHETHER ANY SUCH USE WOULD VIOLATE OR INFRINGE THE APPLICABLE LAWS, OR (C) REVIEWING THE CUSTOMER MATERIALS FOR ACCURACY.

5. Appendix

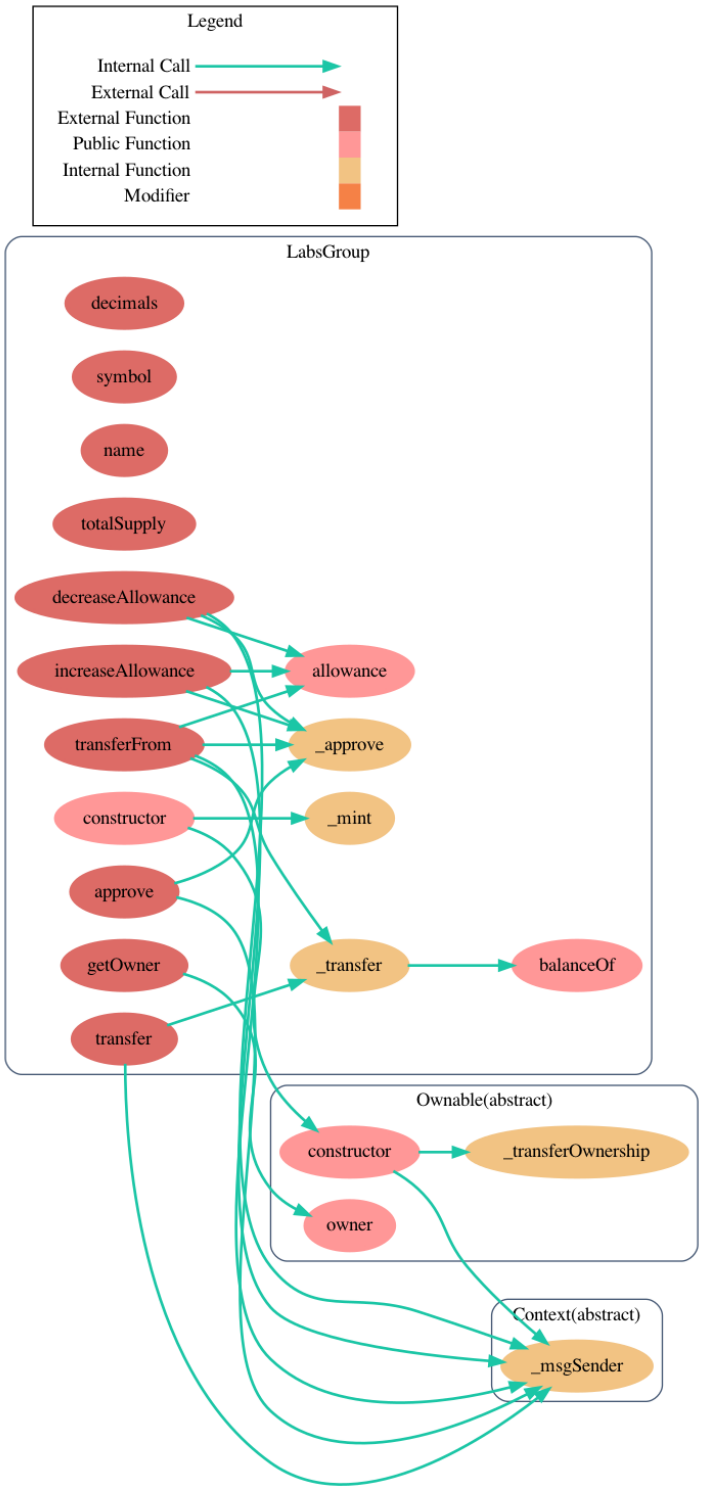
5.1 Visibility

Contract	FuncName	Visibility	Mutability	Modifiers
LabsGroup	_CTOR_	public	Y	
LabsGroup	decimals	external	N	
LabsGroup	symbol	external	N	
LabsGroup	name	external	N	
LabsGroup	totalSupply	external	N	
LabsGroup	balanceOf	public	N	
LabsGroup	getOwner	external	N	
LabsGroup	transfer	external	Y	
LabsGroup	allowance	public	N	
LabsGroup	approve	external	Y	
LabsGroup	transferFrom	external	Y	
LabsGroup	increaseAllowance	external	Y	
LabsGroup	decreaseAllowance	external	Y	
LabsGroup	_transfer	internal	Y	
LabsGroup	_mint	internal	Y	
LabsGroup	_approve	internal	Y	

5. Appendix

5.2 Call Graph

LabsGroup.sol



5. Appendix

5.3 Inheritance Graph

LabsGroup.sol