SMART CONTRACT AUDIT REPORT

For

berylbit (BRB)

Prepared By: SFI Team

Prepared on: 21/12/2021

Prepared for: berylbit Team

Contract address:
0xcA0823d3D04b9FAeA7803cCb87fa8596775190DD

Table of Content

- ## 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) 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.

• Scope of the audit

The scope of this audit was to analyze and document the berylbit smart contract codebase for quality, security, and correctness (automatically)

• Introduction

During the period of **December 19, 2021, to December 21, 2021** - SaferICO

Team performed a security audit for **berylbit** smart contracts.

The project has 1 file. It contains approx 1003 lines of Solidity code. Most of the functions and state variables are well commented on using the Nat spec documentation, but that does not create any vulnerability.

Source Code: https://bscscan.com/address/0xca0823d3d04b9faea7803ccb87fa8596775190dd#code

• Check Vulnerabilities

In order to check for the security of the contract, we tested several attacks in order to make sure that the contract is secure and follows best practices automatically.

1. Unit tests passing.

2. Compilator warnings;

3. Race Conditions. Reentrancy. Cross-function Race Conditions. Pitfalls in Race Condition solutions;

4. Possible delays in data delivery;

5. Transaction-Ordering Dependence (front running);

6. Timestamp Dependence;

7. Integer Overflow and Underflow;

8. DoS with (unexpected) Revert;

9. DoS with Block Gas Limit

10. Call Depth Attack. Not relevant in modern ethereum network

11. Methods execution permissions;

12. Oracles calls;

13. Economy model. It's important to forecast scenarios when a user is provided with additional economic motivation or faced with limitations. If application logic is based on incorrect economy model, the application will not function correctly and participants will incur financial losses. This type of issue is most often found in bonus rewards systems.

14. The impact of the exchange rate on the logic;

15. Private user data leaks.

- ## Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
High	A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.
Medium	The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.
Low	Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.
Informational	These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

• Issues Found – Code Review Automatically

High severity issues

There are no High severity vulnerabilities found

Medium severity issues

There are no Medium severity vulnerabilities found

Low severity issues

#Pragam version not fixed
Description

It is a good practice to lock the solidity version for a live deployment (use 0.8.7 instead of ^0.8.7). contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors.

Remediation
Remove the ^ sign to lock the pragma version

Status: Closed. It already deployed at the same compiler

#use of block.timestamp

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. that means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

```
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
        tokenAmount,
        0,
        path,
```

```
        address(this),
        block.timestamp
    );
```

Remediation

Avoid use of block.timestamp

Status: Acknowledged

Informational issues

#Gas Cost:

Gas requirement of function BaryIBit.functions is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed Please avoid loops in your functions or actions that modify large areas of storage
(This includes clearing or copying arrays in storage)

```solidity
function name() public view returns (string memory) {
    return _name;
}

function symbol() public view returns (string memory) {
    return _symbol;
}

function decimals() public view returns (uint8) {
    return _decimals;
}

function allowance(address owner, address spender) public view override returns (uint256) {
    return _allowances[owner][spender];
}
function transferFrom(address sender, address recipient, uint256 amount) public override returns (bool) {
    _transfer(sender, recipient, amount);
    _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance"));
    return true;
}
```

```solidity
    function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
        _approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
        return true;
    }

    function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
        _approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20: decreased allowance
below zero"));
        return true;
    }
function process_Tokens_Now (uint256 percent_Of_Tokens_To_Process) public onlyOwner {
        // Do not trigger if already in swap
        require(!inSwapAndLiquify, "Currently processing, try later.");
        if (percent_Of_Tokens_To_Process > 100){percent_Of_Tokens_To_Process == 100;}
        uint256 tokensOnContract = balanceOf(address(this));
        uint256 sendTokens = tokensOnContract*percent_Of_Tokens_To_Process/100;
        swapAndLiquify(sendTokens);
    }
```

Remediation

Make these methods external instead public.

Status: Acknowledged

Constant calculations in the contract

Description

recalculated initialization will save 2847 units of gas in deployment

```solidity
uint256 private _tTotal = 300000000 * 10**18;
```

Recommendation

Replace the initialization as

```solidity
uint256 private _tTotal = 300000000000000000000000000000;
```

Status: Acknowledged

- **Source Lines**
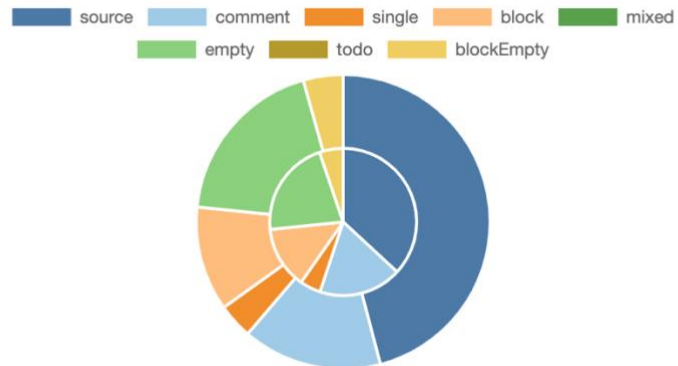
- **Risk Level**
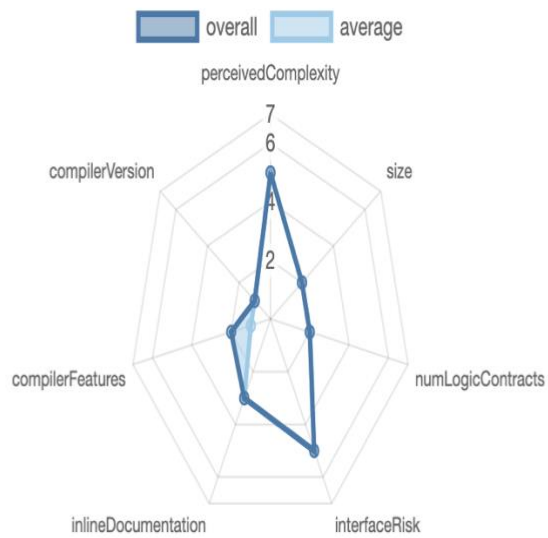
- **Testing proves**

- **Check for security**

e1298c813e57e43438801858cfae53cdb56d469dfeb762db50e34f8fe5361d62

File: BerylBit.sol | Language: solidity | Size: 32461 bytes | Date: 2021-12-21T04:51:10.641Z

Critical	High	Medium	Low	Note
0	0	0	2	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

- **Inheritance graph**
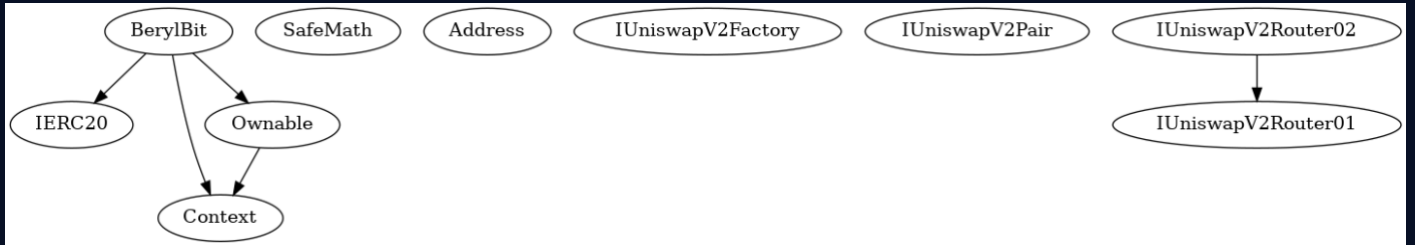
- **Solidity Unit Testing Code & Results**

```solidity
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.4.22 <0.9.0;

// This import is automatically injected by Remix
import "remix_tests.sol";

// This import is required to use custom transaction context
// Although it may fail compilation in 'Solidity Compiler' plugin
// But it will work fine in 'Solidity Unit Testing' plugin
import "remix_accounts.sol";
import "../BerylBit.sol";

// File name has to end with '_test.sol', this file can contain more than one testSuite contracts
contract testSuite {

    /// 'beforeAll' runs before all other tests
    /// More special functions are: 'beforeEach', 'beforeAll', 'afterEach' & 'afterAll'
    function beforeAll() public {
        // <instantiate contract>
        Assert.equal(uint(1), uint(1), "1 should be equal to 1");
    }

    function checkSuccess() public {
        // Use 'Assert' methods: https://remix-ide.readthedocs.io/en/latest/assert_library.html
```

Test your smart contract in Solidity.

Select directory to load and generate test files.

Test directory:

tests | Create

Generate | How to use...

▶ Run | ■ Stop

☑ Select all

☑ tests/BerylBit_test.sol

Progress: 1 finished (of 1)

PASS testSuite (tests/BerylBit_test.sol)

✓ Before all

✓ Check success

✓ Check success2

✓ Check failure

✓ Check sender and value

Result for tests/BerylBit_test.sol
Passing: 5
Total time: 6.13s

```solidity
        Assert.ok(2 == 2, 'should be true');

        Assert.greaterThan(uint(2), uint(1), "2 should be greater than to 1");

        Assert.lesserThan(uint(2), uint(3), "2 should be lesser than to 3");
    }


    function checkSuccess2() public pure returns (bool) {

        // Use the return value (true or false) to test the contract

        return true;
    }


    function checkFailure() public {

        Assert.notEqual(uint(1), uint(2), "1 should not be equal to 1");
    }


    /// Custom Transaction Context: https://remix-
ide.readthedocs.io/en/latest/unittesting.html#customization

    /// #sender: account-1

    /// #value: 100

    function checkSenderAndValue() public payable {

        // account index varies 0-9, value is in wei

        Assert.equal(msg.sender, TestsAccounts.getAccount(1), "Invalid sender");

        Assert.equal(msg.value, 100, "Invalid value");
    }

}
```

Call Graph
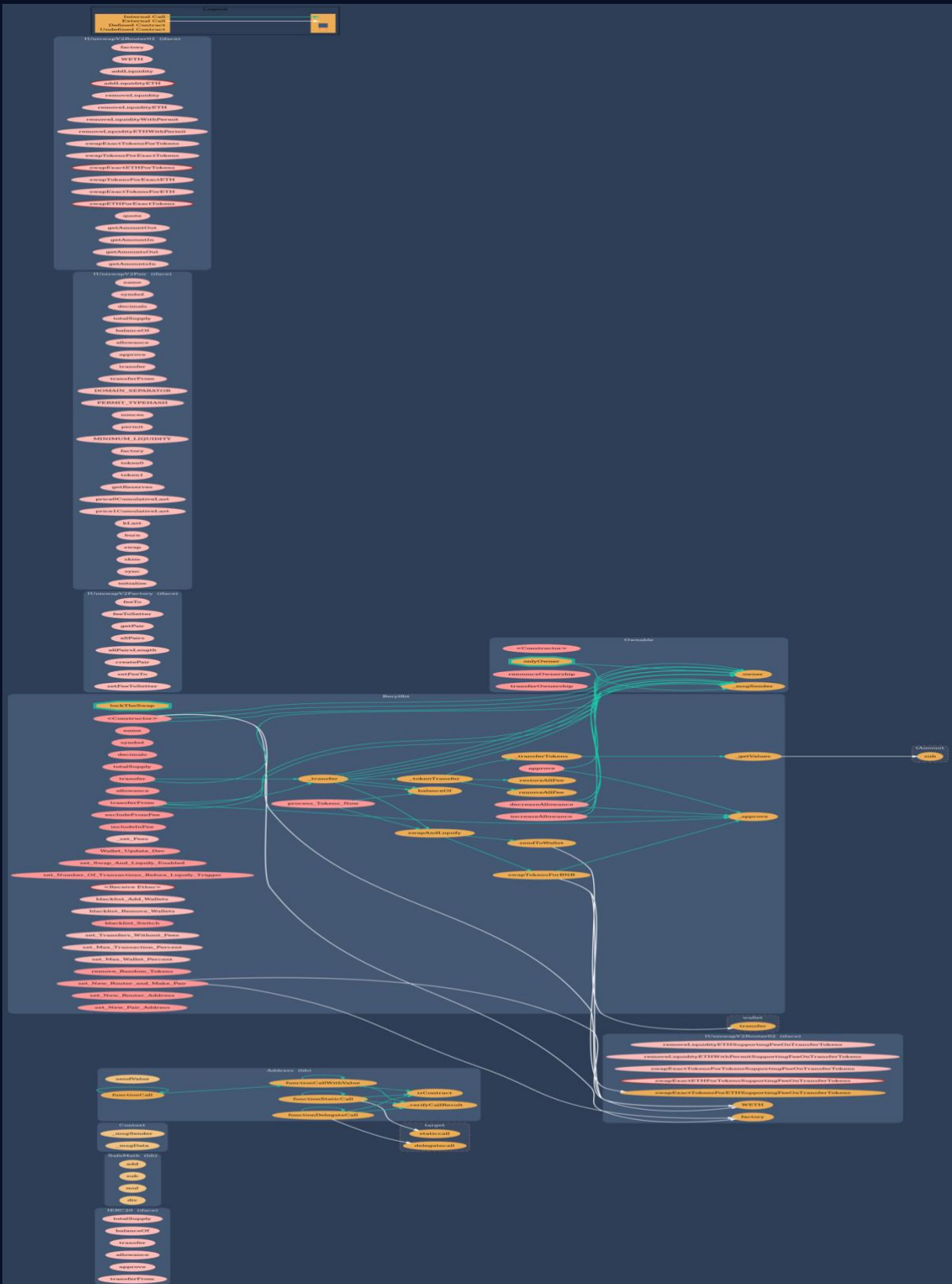
Unified Modeling Language (UML)
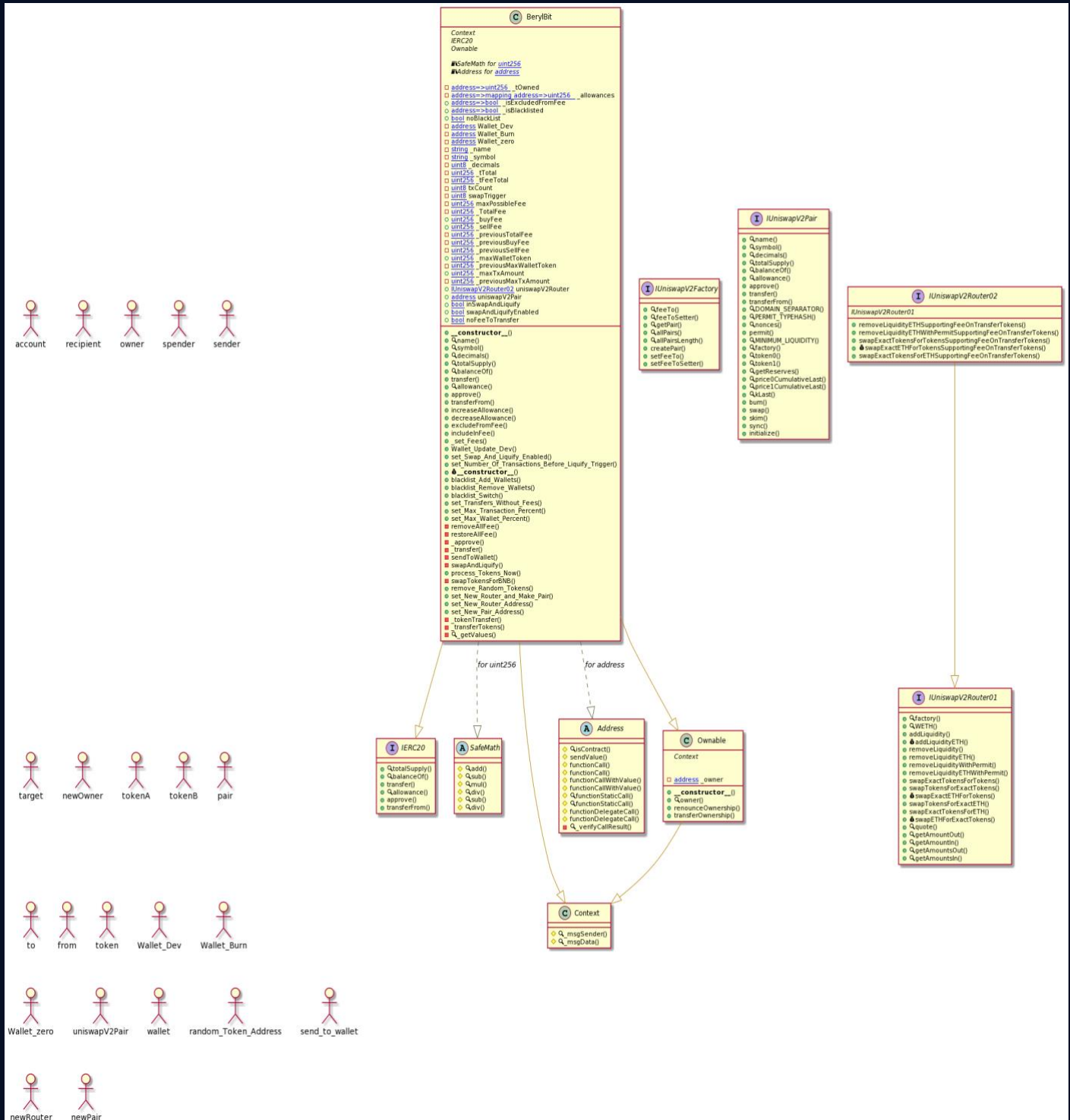
• Capabilities

Components

📝 Contracts	📚 Libraries	🔍 Interfaces	🍪 Abstract
1	2	5	2

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐 Public	💰 Payable
96	5

External	Internal	Private	Pure	View
71	88	11	17	35

StateVariables

Total	🌐 Public
32	12

Capabilities

Solidity Versions observed	🖊 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
^0.8.7		yes	yes (2 asm blocks)	

🛬 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🎴 Uses Hash Functions	🖋 ECRecover	🌀 New/Create/Create2
yes		yes			

♻ TryCatch	Σ Unchecked
	yes

• Source Units In Scope

Source Units in Scope

Source Units Analyzed: **1**
Source Units in Scope: **1** (100%)

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝📚🔍🍪	BerylBit.sol	5	5	1003	793	410	203	489	🖥💰🛬👥☀Σ
📝📚🔍🍪	Totals	5	5	1003	793	410	203	489	🖥💰🛬👥☀Σ

Legend: [—]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

- ## **Function Signature**

```
 16279055  =>  isContract(address)
39509351  =>  increaseAllowance(address,uint256)
94220184  =>  set_Max_Wallet_Percent(uint256)
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
b745d336  =>  div(uint256,uint256,string)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
24a084df  =>  sendValue(address,uint256)
a0b5ffb0  =>  functionCall(address,bytes)
241b5886  =>  functionCall(address,bytes,string)
2a011594  =>  functionCallWithValue(address,bytes,uint256)
d525ab8a  =>  functionCallWithValue(address,bytes,uint256,string)
c21d36f3  =>  functionStaticCall(address,bytes)
dbc40fb9  =>  functionStaticCall(address,bytes,string)
ee33b7e2  =>  functionDelegateCall(address,bytes)
57387df0  =>  functionDelegateCall(address,bytes,string)
18c2c6a2  =>  _verifyCallResult(bool,bytes,string)
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
017e7e58  =>  feeTo()
094b7415  =>  feeToSetter()
e6a43905  =>  getPair(address,address)
1e3dd18b  =>  allPairs(uint256)
574f2ba3  =>  allPairsLength()
```

```
c9c65396  =>  createPair(address,address)
f46901ed  =>  setFeeTo(address)
a2e74af6  =>  setFeeToSetter(address)
06fdde03  =>  name()
95d89b41  =>  symbol()
313ce567  =>  decimals()
3644e515  =>  DOMAIN_SEPARATOR()
30adf81f  =>  PERMIT_TYPEHASH()
7ecebe00  =>  nonces(address)
d505accf  =>  permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
ba9a7a56  =>  MINIMUM_LIQUIDITY()
c45a0155  =>  factory()
0dfe1681  =>  token0()
d21220a7  =>  token1()
0902f1ac  =>  getReserves()
5909c0d5  =>  price0CumulativeLast()
5a3d5493  =>  price1CumulativeLast()
7464fc3d  =>  kLast()
89afcb44  =>  burn(address)
022c0d9f  =>  swap(uint256,uint256,address,bytes)
bc25cf77  =>  skim(address)
fff6cae9  =>  sync()
485cc955  =>  initialize(address,address)
ad5c4648  =>  WETH()
e8e33700  =>  addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
f305d719  =>  addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
baa2abde  =>  removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
02751cec  =>  removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
2195995c  =>  removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
ded9382a  =>  removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
38ed1739  =>  swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
8803dbee  =>  swapTokensForExactTokens(uint256,uint256,address[],address,uint256)
7ff36ab5  =>  swapExactETHForTokens(uint256,address[],address,uint256)
4a25d94a  =>  swapTokensForExactETH(uint256,uint256,address[],address,uint256)
18cbafe5  =>  swapExactTokensForETH(uint256,uint256,address[],address,uint256)
fb3bdb41  =>  swapETHForExactTokens(uint256,address[],address,uint256)
ad615dec  =>  quote(uint256,uint256,uint256)
054d50d4  =>  getAmountOut(uint256,uint256,uint256)
```

```
85f8c259 => getAmountIn(uint256,uint256,uint256)

d06ca61f => getAmountsOut(uint256,address[])

1f00ca74 => getAmountsIn(uint256,address[])

af2979eb => removeLiquidityETHSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256)

5b0d5984 =>
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes
32,bytes32)

5c11d795 => swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)

b6f9de95 => swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256)

791ac947 => swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)

a457c2d7 => decreaseAllowance(address,uint256)

437823ec => excludeFromFee(address)

ea2f0b37 => includeInFee(address)

67cbd84c => _set_Fees(uint256,uint256)

8824e16e => Wallet_Update_Dev(address)

f7739b5f => set_Swap_And_Liquify_Enabled(bool)

7caefa89 => set_Number_Of_Transactions_Before_Liquify_Trigger(uint8)

a9de975d => blacklist_Add_Wallets(address[])

8ec0e9a1 => blacklist_Remove_Wallets(address[])

c1f61908 => blacklist_Switch(bool)

a514a07d => set_Transfers_Without_Fees(bool)

2e39c6c6 => set_Max_Transaction_Percent(uint256)

301370af => removeAllFee()

e7e3e3a7 => restoreAllFee()

104e81ff => _approve(address,address,uint256)

30e0789e => _transfer(address,address,uint256)

6c390b8d => sendToWallet(address,uint256)

173865ad => swapAndLiquify(uint256)

ddbf5266 => process_Tokens_Now(uint256)

56c3726b => swapTokensForBNB(uint256)

3343ab83 => remove_Random_Tokens(address,address,uint256)

1282a0a0 => set_New_Router_and_Make_Pair(address)

36b1a1bc => set_New_Router_Address(address)

d785d5be => set_New_Pair_Address(address)

b09bbc79 => _tokenTransfer(address,address,uint256,bool)

fc1efa66 => _transferTokens(address,address,uint256)

d4780e36 => _getValues(uint256)
```

- ## **Automatic General Report**

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/BerylBit.sol	dee8ff66c03a67a9db745c54ffeaf9b3c70ab76e

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface				
└	totalSupply	External ▯		|NO▯	
└	balanceOf	External ▯		|NO▯	
└	transfer	External ▯	◉	|NO▯	
└	allowance	External ▯		|NO▯	
└	approve	External ▯	◉	|NO▯	
└	transferFrom	External ▯	◉	|NO▯	
SafeMath	Library				
└	add	Internal 🔒			
└	sub	Internal 🔒			
└	mul	Internal 🔒			
└	div	Internal 🔒			
└	sub	Internal 🔒			
└	div	Internal 🔒			
Context	Implementation				
└	_msgSender	Internal 🔒			
└	_msgData	Internal 🔒			
Address	Library				

└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	⬤	
└	functionCall	Internal 🔒	⬤	
└	functionCall	Internal 🔒	⬤	
└	functionCallWithValue	Internal 🔒	⬤	
└	functionCallWithValue	Internal 🔒	⬤	
└	functionStaticCall	Internal 🔒		
└	functionStaticCall	Internal 🔒		
└	functionDelegateCall	Internal 🔒	⬤	
└	functionDelegateCall	Internal 🔒	⬤	
└	_verifyCallResult	Private 🔐		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	⬤	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	⬤	onlyOwner
└	transferOwnership	Public ❗	⬤	onlyOwner
IUniswapV2Factory	Interface			
└	feeTo	External ❗		NO❗
└	feeToSetter	External ❗		NO❗
└	getPair	External ❗		NO❗
└	allPairs	External ❗		NO❗
└	allPairsLength	External ❗		NO❗
└	createPair	External ❗	⬤	NO❗
└	setFeeTo	External ❗	⬤	NO❗
└	setFeeToSetter	External ❗	⬤	NO❗
IUniswapV2Pair	Interface			
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	⬤	NO❗
└	transfer	External ❗	⬤	NO❗
└	transferFrom	External ❗	⬤	NO❗

└	DOMAIN_SEPARATOR	External			NO	
└	PERMIT_TYPEHASH	External			NO	
└	nonces	External			NO	
└	permit	External		◉	NO	
└	MINIMUM_LIQUIDITY	External			NO	
└	factory	External			NO	
└	token0	External			NO	
└	token1	External			NO	
└	getReserves	External			NO	
└	price0CumulativeLast	External			NO	
└	price1CumulativeLast	External			NO	
└	kLast	External			NO	
└	burn	External		◉	NO	
└	swap	External		◉	NO	
└	skim	External		◉	NO	
└	sync	External		◉	NO	
└	initialize	External		◉	NO	
IUniswapV2Router01	Interface			
└	factory	External			NO	
└	WETH	External			NO	
└	addLiquidity	External		◉	NO	
└	addLiquidityETH	External		💳	NO	
└	removeLiquidity	External		◉	NO	
└	removeLiquidityETH	External		◉	NO	
└	removeLiquidityWithPermit	External		◉	NO	
└	removeLiquidityETHWithPermit	External		◉	NO	
└	swapExactTokensForTokens	External		◉	NO	
└	swapTokensForExactTokens	External		◉	NO	
└	swapExactETHForTokens	External		💳	NO	
└	swapTokensForExactETH	External		◉	NO	
└	swapExactTokensForETH	External		◉	NO	
└	swapETHForExactTokens	External		💳	NO	
└	quote	External			NO	
└	getAmountOut	External			NO	
└	getAmountIn	External			NO	
└	getAmountsOut	External			NO	
└	getAmountsIn	External			NO	

||||||

| **IUniswapV2Router02** | Interface | IUniswapV2Router01 |||

| └ | removeLiquidityETHSupportingFeeOnTransferTokens | External ▮ | ◉ |NO▮ |

| └ | removeLiquidityETHWithPermitSupportingFeeOnTransferTokens | External ▮ | ◉ |NO▮ |

| └ | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ▮ | ◉ |NO▮ |

| └ | swapExactETHForTokensSupportingFeeOnTransferTokens | External ▮ | 💵 |NO▮ |

| └ | swapExactTokensForETHSupportingFeeOnTransferTokens | External ▮ | ◉ |NO▮ |

||||||

| **BerylBit** | Implementation | Context, IERC20, Ownable |||

| └ | <Constructor> | Public ▮ | ◉ |NO▮ |

| └ | name | Public ▮ | |NO▮ |

| └ | symbol | Public ▮ | |NO▮ |

| └ | decimals | Public ▮ | |NO▮ |

| └ | totalSupply | Public ▮ | |NO▮ |

| └ | balanceOf | Public ▮ | |NO▮ |

| └ | transfer | Public ▮ | ◉ |NO▮ |

| └ | allowance | Public ▮ | |NO▮ |

| └ | approve | Public ▮ | ◉ |NO▮ |

| └ | transferFrom | Public ▮ | ◉ |NO▮ |

| └ | increaseAllowance | Public ▮ | ◉ |NO▮ |

| └ | decreaseAllowance | Public ▮ | ◉ |NO▮ |

| └ | excludeFromFee | Public ▮ | ◉ | onlyOwner |

| └ | includeInFee | Public ▮ | ◉ | onlyOwner |

| └ | _set_Fees | External ▮ | ◉ | onlyOwner |

| └ | Wallet_Update_Dev | Public ▮ | ◉ | onlyOwner |

| └ | set_Swap_And_Liquify_Enabled | Public ▮ | ◉ | onlyOwner |

| └ | set_Number_Of_Transactions_Before_Liquify_Trigger | Public ▮ | ◉ | onlyOwner |

| └ | <Receive Ether> | External ▮ | 💵 |NO▮ |

| └ | blacklist_Add_Wallets | External ▮ | ◉ | onlyOwner |

| └ | blacklist_Remove_Wallets | External ▮ | ◉ | onlyOwner |

| └ | blacklist_Switch | Public ▮ | ◉ | onlyOwner |

| └ | set_Transfers_Without_Fees | External ▮ | ◉ | onlyOwner |

| └ | set_Max_Transaction_Percent | External ▮ | ◉ | onlyOwner |

| └ | set_Max_Wallet_Percent | External ▮ | ◉ | onlyOwner |

| └ | removeAllFee | Private 🔐 | ◉ ||

| └ | restoreAllFee | Private 🔐 | ◉ ||

| └ | _approve | Private 🔐 | ◉ ||

| └ | _transfer | Private 🔐 | ◉ ||

└	sendToWallet	Private 🔒	◉	
└	swapAndLiquify	Private 🔒	◉	lockTheSwap
└	process_Tokens_Now	Public ▯	◉	onlyOwner
└	swapTokensForBNB	Private 🔒	◉	
└	remove_Random_Tokens	Public ▯	◉	onlyOwner
└	set_New_Router_and_Make_Pair	Public ▯	◉	onlyOwner
└	set_New_Router_Address	Public ▯	◉	onlyOwner
└	set_New_Pair_Address	Public ▯	◉	onlyOwner
└	_tokenTransfer	Private 🔒	◉	
└	_transferTokens	Private 🔒	◉	
└	_getValues	Private 🔒		

Legend

Symbol	Meaning
◉	Function can modify state
◰	Function is payable

- ## Summary of the Audit

According to automatically test, the customer`s solidity smart contract is **Secured**.

The general overview is presented in the Project Information section and all issues found are located in the audit overview section.

The test found 0 critical, 0 high, 0 medium, 2 low issues, and 2 note.