A Secure Place For Web3

SMART CONTRACT AUDIT OF

LaeebDoge Presale

Contract Address

0xB85095CF7c740eba210B19Dc82Ca5b03C6DbEb96

Audit Summary

Expelee team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analysed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

Audit Result: **PASSED**

Ownership: NOT **RENOUNCED**

KYC Verification: Not done till date of audit

Audit Date: 21/07/2022

Audit Team: **EXPELEE**

Be aware that smart contracts deployed on the blockchain aren’t resistant to internal exploit, external vulnerability, or hack. For a detailed understanding of risk severity, source code vulnerability, functional hack, and audit disclaimer, kindly refer to the audit.

DISCLAMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document.

Always Do your own research and protect yourselves from being scammed. The Expelee team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools.

Under no circumstances did Expelee receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams.

This document should not be presented as a reason to buy or not buy any particular token. The Expelee team disclaims any liability for the resulting losses.

Contract Review

Contract Name	LaeebDoge
Compiler Version	v0.8.0+commit.c7dfd78e
Optimization	No with 200 runs
License	None license
Explorer	https://bscscan.com/address/0xB85095CF7c740eba210B19Dc82Ca5b03C6DbEb96
Symbol	LAEEBDOGE
Decimals	18
Total Supply	1,000,000,000
Domain	https://laeebdoge.io/

Project Review

Token Name: LaeebDoge

Web Site: https://laeebdoge.io/

Twitter: https://twitter.com/laeebdogeio

Telegram: https://t.me/laeebdogeofficial

Contract Address:
0xB85095CF7c740eba210B19Dc82Ca5b03C6DbEb96

Platform: Binance Smart Chain

Token Type: BEP 20

Language: SOLIDITY

Audit Methodology

The scope of this report is to audit the smart contract source code. We have scanned the contract and reviewed the project for common vulnerabilities, exploits, hacks, and back-doors. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

Category

Smart Contract Vulnerabilities	- Unhandled Exceptions - Transaction Order Dependency - Integer Overflow - Unrestricted Action - Incorrect Inheritance Order - Typographical Errors - Requirement Violation
Source Code Review	- Gas Limit and Loops - Deployment Consistency - Repository Consistency - Data Consistency - Token Supply Manipulation
Functional Assessment	- Operations Trail & Event Generation - Assets Manipulation - Liquidity Access

Vulnerability Checklist

№	Description.	Result
1	Compiler warnings.	Passed
2	Race conditions and Re-entrancy. Cross-function raceconditions.	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.	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
19	Cross-function race conditions.	Passed
20	Safe Zeppelin module.	Passed
21	Fallback function security.	Passed

Manual Audit

🟢 **Low-Risk**

🟢 4 low-risk code issues found

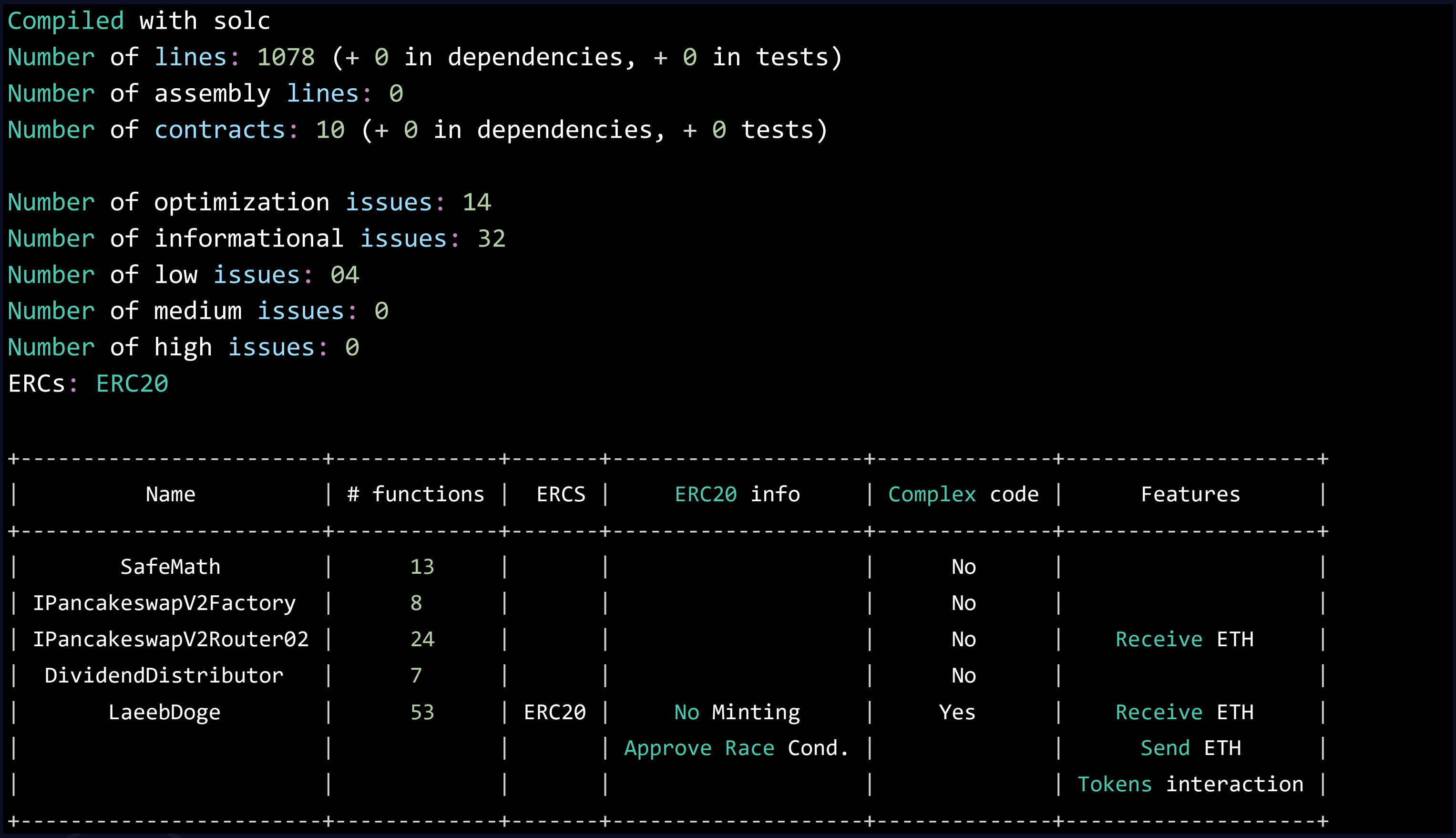
🟡 **Medium-Risk**

🟡 0 medium-risk code issues found

🔴 **High-Risk**

🔴 0 high-risk code issues found

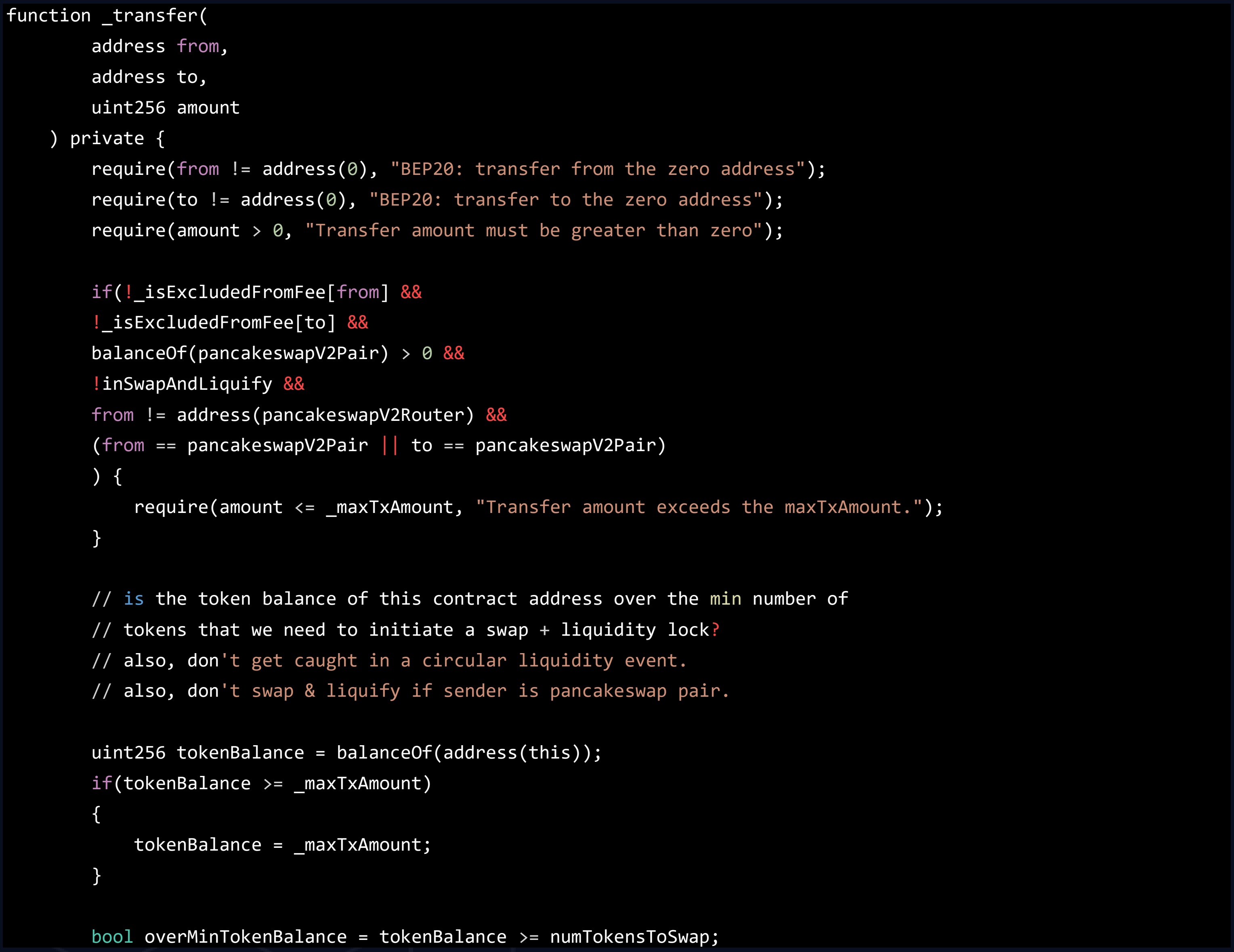
Audit Summary



```
Compiled with solc
Number of lines: 1078 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 10 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 14
Number of informational issues: 32
Number of low issues: 04
Number of medium issues: 0
Number of high issues: 0
ERCs: ERC20

+------------------------+-------------+-------+--------------------+--------------+--------------------+
|          Name          | # functions |  ERCS |     ERC20 info     | Complex code |      Features      |
+------------------------+-------------+-------+--------------------+--------------+--------------------+
|        SafeMath        |      13     |       |                    |      No      |                    |
| IPancakeswapV2Factory  |      8      |       |                    |      No      |                    |
| IPancakeswapV2Router02 |      24     |       |                    |      No      |    Receive ETH     |
|  DividendDistributor   |      7      |       |                    |      No      |                    |
|       LaeebDoge        |      53     | ERC20 |     No Minting     |     Yes      |    Receive ETH     |
|                        |             |       | Approve Race Cond. |              |      Send ETH      |
|                        |             |       |                    |              | Tokens interaction |
+------------------------+-------------+-------+--------------------+--------------+--------------------+
```

Low-Risk

1) Contract contains Reentrancy vulnuerabilities



```
function _transfer(
        address from,
        address to,
        uint256 amount
    ) private {
        require(from != address(0), "BEP20: transfer from the zero address");
        require(to != address(0), "BEP20: transfer to the zero address");
        require(amount > 0, "Transfer amount must be greater than zero");

        if(!_isExcludedFromFee[from] &&
        !_isExcludedFromFee[to] &&
        balanceOf(pancakeswapV2Pair) > 0 &&
        !inSwapAndLiquify &&
        from != address(pancakeswapV2Router) &&
        (from == pancakeswapV2Pair || to == pancakeswapV2Pair)
        ) {
            require(amount <= _maxTxAmount, "Transfer amount exceeds the maxTxAmount.");
        }

        // is the token balance of this contract address over the min number of
        // tokens that we need to initiate a swap + liquidity lock?
        // also, don't get caught in a circular liquidity event.
        // also, don't swap & liquify if sender is pancakeswap pair.

        uint256 tokenBalance = balanceOf(address(this));
        if(tokenBalance >= _maxTxAmount)
        {
            tokenBalance = _maxTxAmount;
        }

        bool overMinTokenBalance = tokenBalance >= numTokensToSwap;
```

Recommendation

Apply the check-effects-interaction pattern

2) Unchecked Transfer

The return value of an external transfer/transferFrom call is not checked .

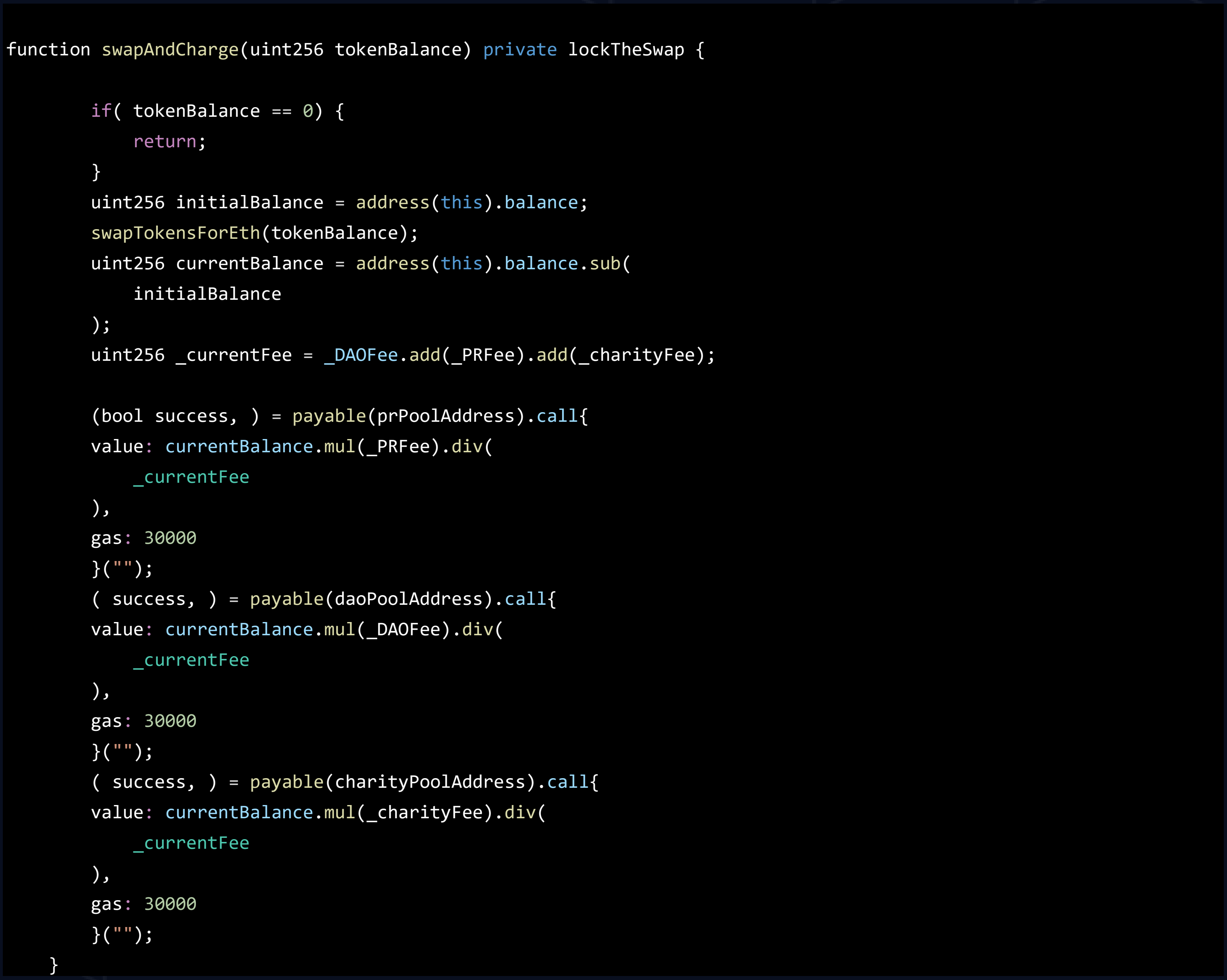
```
function recoverERC20(address tokenAddress, uint256 tokenAmount) external onlyMod {
        IBEP20(tokenAddress).transfer(modAddress, tokenAmount);
        emit Recovered(tokenAddress, tokenAmount);
    }
```

Recommendation

Use SafeERC20, or ensure that the transfer/transferFrom return value is checked.

3) Functions that send Ether to arbitary destinations

Unprotected call to a function sending Ether to arbitary address.



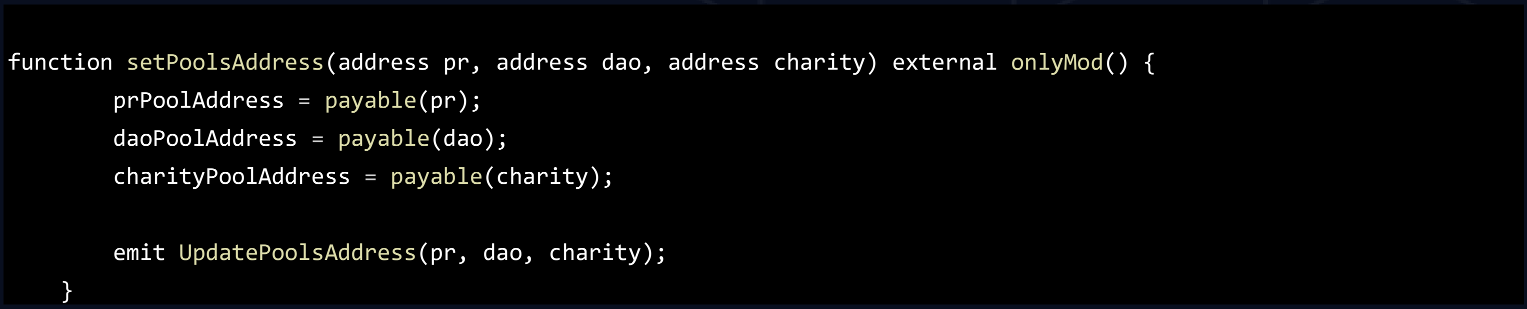
```
function swapAndCharge(uint256 tokenBalance) private lockTheSwap {

        if( tokenBalance == 0) {
            return;
        }
        uint256 initialBalance = address(this).balance;
        swapTokensForEth(tokenBalance);
        uint256 currentBalance = address(this).balance.sub(
            initialBalance
        );
        uint256 _currentFee = _DAOFee.add(_PRFee).add(_charityFee);

        (bool success, ) = payable(prPoolAddress).call{
        value: currentBalance.mul(_PRFee).div(
            _currentFee
        ),
        gas: 30000
        }("");
        ( success, ) = payable(daoPoolAddress).call{
        value: currentBalance.mul(_DAOFee).div(
            _currentFee
        ),
        gas: 30000
        }("");
        ( success, ) = payable(charityPoolAddress).call{
        value: currentBalance.mul(_charityFee).div(
            _currentFee
        ),
        gas: 30000
        }("");
    }
```

Recommendation

Ensure that an arbitary user cannot withdraw unauthorized funds

4) Missing zero address validation

Detect missing zero address validation.

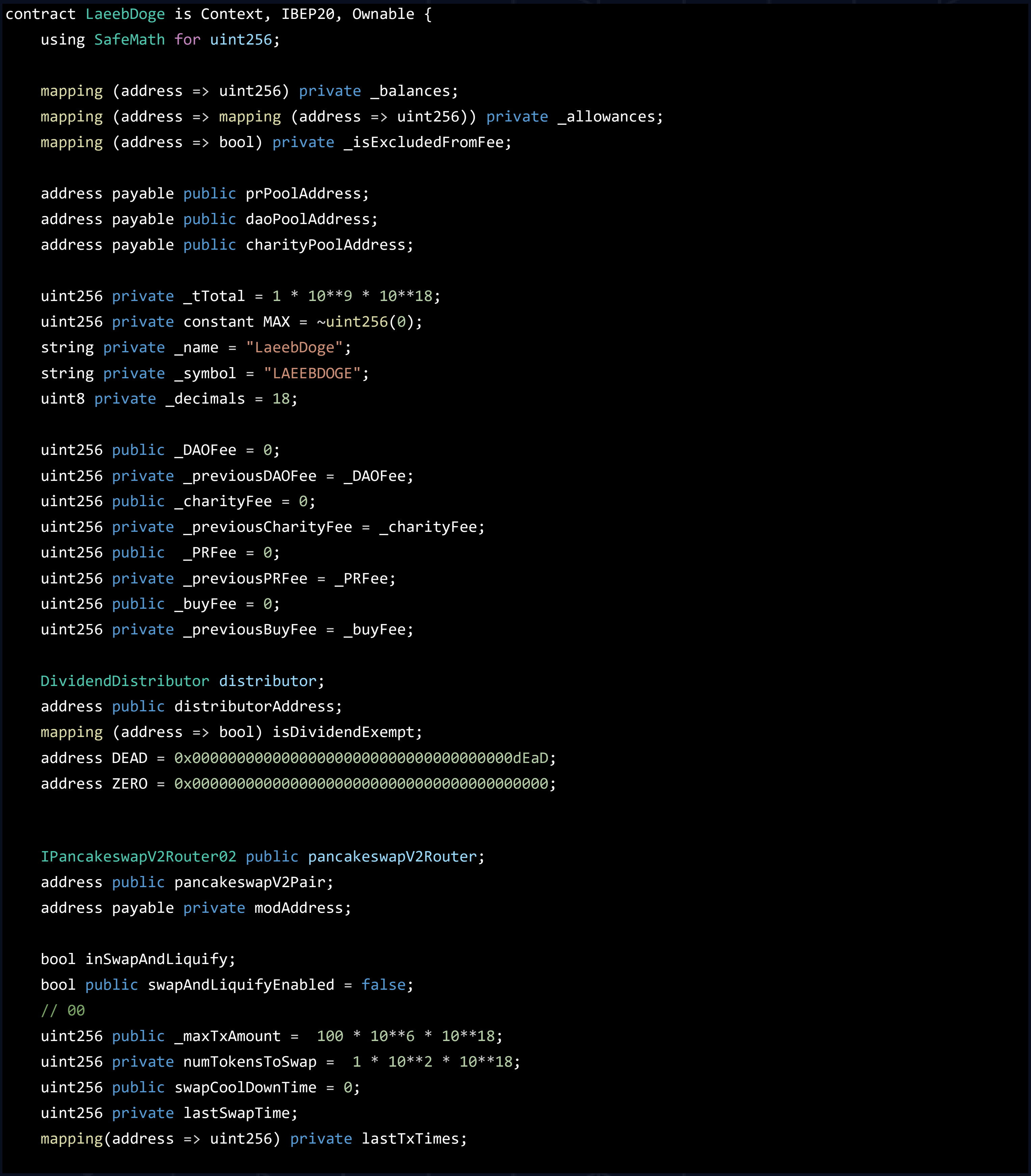


```
function setPoolsAddress(address pr, address dao, address charity) external onlyMod() {
        prPoolAddress = payable(pr);
        daoPoolAddress = payable(dao);
        charityPoolAddress = payable(charity);

        emit UpdatePoolsAddress(pr, dao, charity);
    }
```

Recommendation

Check that the address is not zero.

Manual Audit (Contract Function)



```
contract LaeebDoge is Context, IBEP20, Ownable {
    using SafeMath for uint256;

    mapping (address => uint256) private _balances;
    mapping (address => mapping (address => uint256)) private _allowances;
    mapping (address => bool) private _isExcludedFromFee;

    address payable public prPoolAddress;
    address payable public daoPoolAddress;
    address payable public charityPoolAddress;

    uint256 private _tTotal = 1 * 10**9 * 10**18;
    uint256 private constant MAX = ~uint256(0);
    string private _name = "LaeebDoge";
    string private _symbol = "LAEEBDOGE";
    uint8 private _decimals = 18;

    uint256 public _DAOFee = 0;
    uint256 private _previousDAOFee = _DAOFee;
    uint256 public _charityFee = 0;
    uint256 private _previousCharityFee = _charityFee;
    uint256 public  _PRFee = 0;
    uint256 private _previousPRFee = _PRFee;
    uint256 public _buyFee = 0;
    uint256 private _previousBuyFee = _buyFee;

    DividendDistributor distributor;
    address public distributorAddress;
    mapping (address => bool) isDividendExempt;
    address DEAD = 0x000000000000000000000000000000000000dEaD;
    address ZERO = 0x0000000000000000000000000000000000000000;


    IPancakeswapV2Router02 public pancakeswapV2Router;
    address public pancakeswapV2Pair;
    address payable private modAddress;

    bool inSwapAndLiquify;
    bool public swapAndLiquifyEnabled = false;
    // 00
    uint256 public _maxTxAmount =  100 * 10**6 * 10**18;
    uint256 private numTokensToSwap =  1 * 10**2 * 10**18;
    uint256 public swapCoolDownTime = 0;
    uint256 private lastSwapTime;
    mapping(address => uint256) private lastTxTimes;
```

Important Points To Consider

✓ Verified contract source

✓ Token is sellable (not a honeypot) at this time

✗ Ownership renounced or source does not contain an owner contract

✓ Buy fee is less than 10% (0%)

✓ Sell fee is less than 10% (0%)

✓ Owner/creator wallet contains less than 10% of circulating token supply (5.19%)

About Expelee

Expelee is a community driven organisation dedicated to fostering an anti-rug movement. We're here to keep investment safe from fraudsters. We've encountered several rug pulls and know how it feels to be duped, which is why we don't want anybody else to go through the same experience. We are here to raise awareness through our services so that the future of cryptocurrency can be rug-free.

The auditing process focuses to the following considerations with collaboration of an expert team:

- Functionality test of the Smart Contract to determine if proper logic has been followed throughout the whole process.
- Manually detailed examination of the code line by line by experts.
- Live test by multiple clients using Test net.
- Analysing failure preparations to check how the Smart
- Contract performs in case of any bugs and vulnerabilities.
- Checking whether all the libraries used in the code are on the latest version.
- Analysing the security of the on-chain data.

Social Media

/Expelee

/ExpeleeOfficial

/expelee-co