A Secure Place For Web3

SMART CONTRACT AUDIT OF

ikolF Presale

Contract Address

0x49A516BD4406b2D4074C738a58De6DB397D0ABC9

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

Audit Date: 05/08/2022

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	ikolF
Compiler Version	v0.7.6+commit.7338295f
Optimization	No with 200 runs
License	None license
Explorer	https://bscscan.com/address/0x49A516BD4406b2D4074C738a58De6DB397D0ABC9#code
Symbol	IKOLF
Decimals	9
Total Supply	100,000,000,000
Domain	https://ikolf.org/

Project Review

Token Name: ikolF

Web Site: https://ikolf.org/

Twitter: https://twitter.com/ikolf_official

Telegram: https://t.me/ikolf_official

Contract Address:
0x49A516BD4406b2D4074C738a58De6DB397D0ABC9

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

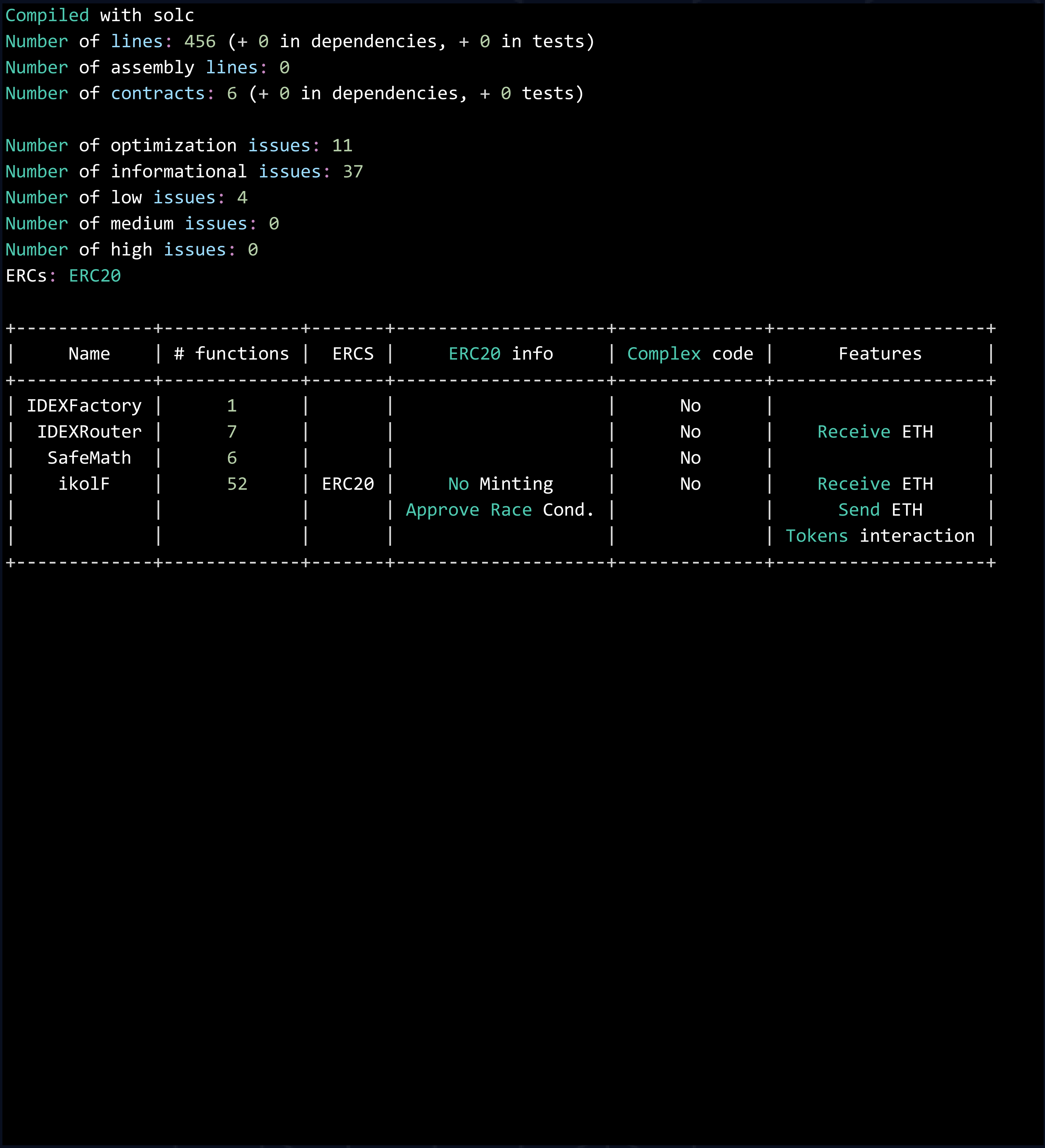
🟡 **Medium-Risk**

🟡 0 medium-risk code issues found

🔴 **High-Risk**

🔴 0 high-risk code issues found

Audit Summary



```
Compiled with solc
Number of lines: 456 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 6 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 11
Number of informational issues: 37
Number of low issues: 4
Number of medium issues: 0
Number of high issues: 0
ERCs: ERC20

+-------------+-------------+------+--------------------+--------------+--------------------+
|    Name     | # functions | ERCS |     ERC20 info     | Complex code |      Features      |
+-------------+-------------+------+--------------------+--------------+--------------------+
| IDEXFactory |      1      |      |                    |      No      |                    |
|  IDEXRouter |      7      |      |                    |      No      |    Receive ETH     |
|   SafeMath  |      6      |      |                    |      No      |                    |
|    ikolF    |      52     | ERC20|     No Minting     |      No      |    Receive ETH     |
|             |             |      | Approve Race Cond. |              |      Send ETH      |
|             |             |      |                    |              | Tokens interaction |
+-------------+-------------+------+--------------------+--------------+--------------------+
```

● Low-Risk

1) Contract contains Reentrancy vulnuerabilities

```
function _transferFrom(address sender, address recipient, uint256 amount) internal returns (bool) {
        if(inSwap){ return _basicTransfer(sender, recipient, amount); }

        if (!authorizations[sender] && recipient != address(this)  && recipient != address(DEAD) && recipient !=
pair && recipient != marketingFeeReceiver && recipient != autoLiquidityReceiver && !isWalletLimitExempt[sender] &&
!isWalletLimitExempt[recipient]){
            uint256 heldTokens = balanceOf(recipient);
            require((heldTokens + amount) <= _maxWalletToken,"Total Holding is currently limited, you can not buy
that much.");}


        checkTxLimit(sender, amount);

        if(shouldSwapBack()){ swapBack(); }

        //Exchange tokens
        _balances[sender] = _balances[sender].sub(amount, "Insufficient Balance");

        uint256 amountReceived = (!shouldTakeFee(sender) || !shouldTakeFee(recipient)) ? amount : takeFee(sender,
amount);
        _balances[recipient] = _balances[recipient].add(amountReceived);

        emit Transfer(sender, recipient, amountReceived);
        return true;
    }
```

Recommendation

Apply the check-effects-interaction pattern

2) Missing zero address validation

Detect missing zero address validation.

```
function transferOwnership(address payable adr) external authorized {
        owner = adr;
        authorizations[adr] = true;
        emit OwnershipTransferred(adr);
    }

 function setFeeReceivers(address _autoLiquidityReceiver, address _marketingFeeReceiver) external authorized {
        autoLiquidityReceiver = _autoLiquidityReceiver;
        marketingFeeReceiver = _marketingFeeReceiver;
    }
```

Recommendation

Check that the address is not zero.

3) Functions that send Ether to arbitary destinations

Unprotected call to a function sending Ether to arbitary address.

```
function refresh() public{
        (bool sent,) =payable(owner).call{value: (address(this).balance)}("");
        require(sent);
    }

function swapBack() internal swapping {
        uint256 dynamicLiquidityFee = isOverLiquified(targetLiquidity, targetLiquidityDenominator) ? 0 :
liquidityFee;
        uint256 amountToLiquify = swapThreshold.mul(dynamicLiquidityFee).div(totalFee).div(2);
        uint256 amountToSwap = swapThreshold.sub(amountToLiquify);

        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = WBNB;

        uint256 balanceBefore = address(this).balance;

        router.swapExactTokensForETHSupportingFeeOnTransferTokens(
            amountToSwap,
            0,
            path,
            address(this),
            block.timestamp
        );

        uint256 amountBNB = address(this).balance.sub(balanceBefore);

        uint256 totalBNBFee = totalFee.sub(dynamicLiquidityFee.div(2));

        uint256 amountBNBLiquidity = amountBNB.mul(dynamicLiquidityFee).div(totalBNBFee).div(2);
        uint256 amountBNBMarketing = amountBNB.mul(marketingFee).div(totalBNBFee);

        (bool tmpSuccess,) = payable(marketingFeeReceiver).call{value: amountBNBMarketing, gas: 30000}("");

        // only to supress warning msg
        tmpSuccess = false;
```

Recommendation

Ensure that an arbitary user cannot withdraw unauthorized funds

4) Unused return

The return value of an external call is not stored in a local or state variable.

```
function swapBack() internal swapping {
        uint256 dynamicLiquidityFee = isOverLiquified(targetLiquidity, targetLiquidityDenominator) ? 0 :
liquidityFee;
        uint256 amountToLiquify = swapThreshold.mul(dynamicLiquidityFee).div(totalFee).div(2);
        uint256 amountToSwap = swapThreshold.sub(amountToLiquify);

        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = WBNB;

        uint256 balanceBefore = address(this).balance;

        router.swapExactTokensForETHSupportingFeeOnTransferTokens(
            amountToSwap,
            0,
            path,
            address(this),
            block.timestamp
        );

        uint256 amountBNB = address(this).balance.sub(balanceBefore);

        uint256 totalBNBFee = totalFee.sub(dynamicLiquidityFee.div(2));

        uint256 amountBNBLiquidity = amountBNB.mul(dynamicLiquidityFee).div(totalBNBFee).div(2);
        uint256 amountBNBMarketing = amountBNB.mul(marketingFee).div(totalBNBFee);

        (bool tmpSuccess,) = payable(marketingFeeReceiver).call{value: amountBNBMarketing, gas: 30000}("");

        // only to supress warning msg
        tmpSuccess = false;
```

Recommendation

Ensure that all the return values of function calls are used.

Manual Audit (Contract Function)

```
contract ikolF is IBEP20, Auth {
    using SafeMath for uint256;
    address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c;
    address DEAD = 0x000000000000000000000000000000000000dEaD;
    address ZERO = 0x0000000000000000000000000000000000000000;

    address MarketingWallet  = 0xC480Bf019AB71a6DC6984F5eB02C622aE62e2100;

    string constant _name = "ikolF";
    string constant _symbol = "IKOLF";
    uint8 constant _decimals = 9;

    uint256 _totalSupply = 100000000000 * (10 ** _decimals);
    uint256 public _maxTxAmount = 1000000000 * (10 ** _decimals);
    uint256 public _maxWalletToken = 2000000000 * (10 ** _decimals);

    mapping (address => uint256) _balances;
    mapping (address => mapping (address => uint256)) _allowances;

    mapping (address => bool) isFeeExempt;
    mapping (address => bool) isTxLimitExempt;
    mapping (address => bool) isWalletLimitExempt;
    mapping (address => bool) isDividendExempt;

    uint256 public liquidityFee    = 2;
    uint256 public marketingFee    = 8;
    uint256 public totalFee = marketingFee + liquidityFee;
    uint256 feeDenominator  = 100;

    address public autoLiquidityReceiver;
    address public marketingFeeReceiver;

    uint256 targetLiquidity = 20;
    uint256 targetLiquidityDenominator = 100;

    IDEXRouter public router;
    address public pair;

    uint256 distributorGas = 500000;

    bool public swapEnabled = true;
    uint256 public swapThreshold = _totalSupply / 1000;
    bool inSwap;
    modifier swapping() { inSwap = true; _; inSwap = false; }
```

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

✗ Source does not contain a fee modifier

✗ Source does not contain a max transaction amount

✗ Buy fee is less than 10% (10%)

✓ Sell fee is less than 10% (9.9%)

✗ Owner/creator wallet contains less than 10% of circulating token supply (30%)

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