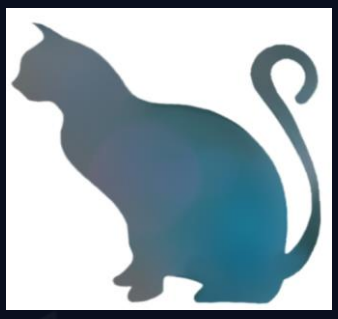
# expelee

A Secure Place For Web3

# SMART CONTRACT AUDIT OF

# 1Cat Presale



Contract Address

**0x80a314EA93e418Ea0e6CB3F4D8ce4892d7F593d2**

# 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 (Medium Risk Severity)**

Ownership: NOT **RENOUNCED**

KYC Verification: Not done till date of audit

Audit Date: 08/07/2022

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** | **OneCat** |
| **Compiler Version** | **v0.8.4+commit.c7e474f2** |
| **Optimization** | **Yes with 200 runs** |
| **License** | **MIT license** |
| **Explorer** | **https://bscscan.com/address/0x80a314EA93e418Ea0e6CB3F4D8ce4892d7F593d2#code** |
| **Symbol** | **1Cat** |
| **Decimals** | **18** |
| **Total Supply** | **100,000,000** |
| **Domain** | **https://www.1cattoken.com/** |

# Project Review

Token Name: 1Cat

Web Site: https://www.1cattoken.com/

Twitter: https://twitter.com/Monoxide_1cat

Telegram: https://t.me/token1cat

Contract Address:
0x80a314EA93e418Ea0e6CB3F4D8ce4892d7F593d2

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<br>- Transaction Order Dependency<br>- Integer Overflow<br>- Unrestricted Action<br>- Incorrect Inheritance Order<br>- Typographical Errors<br>- Requirement Violation |
| Source Code Review | - Gas Limit and Loops<br>- Deployment Consistency<br>- Repository Consistency<br>- Data Consistency<br>- Token Supply Manipulation |
| Functional Assessment | - Operations Trail & Event Generation<br>- Assets Manipulation<br>- 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**

3 low-risk code issues found

**Medium-Risk**

0 medium-risk code issues found

**High-Risk**

0 high-risk code issues found

## Low-Risk

### 1) Contract contains Reentrancy vulnuerabilities

```
function _transfer(address sender, address recipient, uint256 amount) private returns (bool) {

        require(sender != address(0), "ERC20: transfer from the zero address");
        require(recipient != address(0), "ERC20: transfer to the zero address");
        require(!isBot[recipient] && !isBot[sender], "Address is blacklisted");

        if(inSwapAndLiquify)
        {
            return _basicTransfer(sender, recipient, amount);
        }
        else
        {
            if(!isTxLimitExempt[sender] && !isTxLimitExempt[recipient]) {
                require(amount <= _maxTxAmount, "Transfer amount exceeds the maxTxAmount.");
            }

            uint256 contractTokenBalance = balanceOf(address(this));
            bool overMinimumTokenBalance = contractTokenBalance >= minimumTokensBeforeSwap;
```

#### Recommendation

Apply the check-effects-interaction pattern

## 2) Function that send Ether to arbitary destination.

Unprotected call to a function sending Ether to an arbitary address.

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
        // approve token transfer to cover all possible scenarios
        _approve(address(this), address(uniswapV2Router), tokenAmount);

        // add the liquidity
        uniswapV2Router.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount,
            0, // slippage is unavoidable
            0, // slippage is unavoidable
            owner(),
            block.timestamp
        );
    }
```

### Recommendation

Ensure that an arbitary user cannot withdraw unauthorized funds.

### 3) Unused return.

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

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
        // approve token transfer to cover all possible scenarios
        _approve(address(this), address(uniswapV2Router), tokenAmount);

        // add the liquidity
        uniswapV2Router.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount,
            0, // slippage is unavoidable
            0, // slippage is unavoidable
            owner(),
            block.timestamp
        );
    }
```

### Recommendation

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

# Audit Summary

```
Compiled with solc
Number of lines: 783 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 10 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 26
Number of informational issues: 54
Number of low issues: 3
Number of medium issues: 0
Number of high issues: 0
ERCs: ERC20, ERC2612
```

| Name | # functions | ERCS | ERC20 info | Complex code | Features |
|---|---|---|---|---|---|
| SafeMath | 8 | | | No | |
| Address | 7 | | | No | Send ETH Assembly |
| IUniswapV2Factory | 8 | | | No | |
| IUniswapV2Pair | 26 | ERC20,ERC2612 | No Minting Approve Race Cond. | No | |
| IUniswapV2Router02 | 24 | | | No | Receive ETH |
| OneCat | 54 | ERC20 | No Minting Approve Race Cond. | Yes | Receive ETH Send ETH |

# Manual Audit (Contract Function)

```
contract OneCat is Context, IERC20, Ownable {

    using SafeMath for uint256;
    using Address for address;

    string private _name = "1Cat";
    string private _symbol = "1Cat";
    uint8 private _decimals = 18;

    address payable public marketingWalletAddress = payable(0x876075fDe59810d15224506E3d903BF0BaBdb533);
    address payable public bonusWalletAddress = payable(0x6db37F98bd26a5DF65fFCA30B41be5fc7AA6dAAC);
    address public immutable deadAddress = 0x000000000000000000000000000000000000dEaD;

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

    mapping (address => bool) public isExcludedFromFee;
    mapping (address => bool) public isWalletLimitExempt;
    mapping (address => bool) public isTxLimitExempt;
    mapping (address => bool) public isMarketPair;
    mapping (address => bool) public isBot;

    uint256 public _buyLiquidityFee = 1;
    uint256 public _buyMarketingFee =3;
    uint256 public _buyBonusFee = 4;

    uint256 public _sellLiquidityFee = 1;
    uint256 public _sellMarketingFee = 3;
    uint256 public _sellBonusFee = 4;

    uint256 public _liquidityShare = 4;
    uint256 public _marketingShare = 4;
    uint256 public _bonusShare = 16;

    uint256 public _totalTaxIfBuying = 8;
    uint256 public _totalTaxIfSelling = 8;
    uint256 public _totalDistributionShares = 24;

    uint256 private _totalSupply =  100000000 * 10**_decimals;
    uint256 public _maxTxAmount =   100000000 * 10**_decimals;
    uint256 public _walletMax =     100000000 * 10**_decimals;
    uint256 private minimumTokensBeforeSwap = 100 * 10**_decimals;
```

# 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% (8%)

✗ Sell fee is less than 10% (21%)

✓ Owner/creator wallet contains less than 10% of circulating token supply (3.05%)

# 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