expelee

Building the Futuristic Blockchain Ecosystem

# SECURITY AUDIT REPORT

## KERMIT

expelee

# HIGH RISK ANALYSIS

This section contains a brief Summary of all the High Risks present in this Project's Smart Contract

| Findings | Found |
| --- | --- |
| High Risk | 1 |

## High Risk Details

### Trades must be enabled by the owner

### Risk : Centralisation

### Overview

Owner must enable trading to enable public to trade their tokens, otherwise no one would be able to buy /sell their tokens except whitelisted wallets.

More Details of this High Risk are given on Page 13 of this Audit Report

# TABLE OF CONTENTS

# OVERVIEW

The 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** |
| **KYC Verification** | **No** |
| **Audit Date** | **2 June 2023** |

# CONTRACT DETAILS

Token Name: Kermit

Symbol: kermit

Network: Binance Smart Chain

Language: Solidity

Contract Address:
0x5cB1c88f98dF93B97d1577985fa2d260e3d007E8

Total Supply: 1000000000000

Owner's Wallet:
0x1a4Cb975B959AD4Ab87EB80384Cf8f9cCC31Bd1c

Deployer's Wallet:
0x1a4Cb975B959AD4Ab87EB80384Cf8f9cCC31Bd1c

Testnet Link:
https://testnet.bscscan.com/address/0x30eC5ffA36269f3A4C9ffF2CeeC387B9c5420080

# OWNER PRIVILEGES

- Trade must be enabled by the owner
- Owner can change buy fees up to 10% sell fees up to 15% at max
- Owner can exclude/include account from rewards
- Owner can exclude account from fees
- Owner can change swap setting
- Owner can withdraw claimstuck tokens except native tokens

# AUDIT METHODOLOGY

## Audit Details

Our comprehensive audit report provides a full overview of the audited system's architecture, smart contract codebase, and details on any vulnerabilities found within the system.

## Audit Goals

The audit goal is to ensure that the project is built to protect investors and users, preventing potentially catastrophic vulnerabilities after launch , that lead to scams and rugpulls.

## Code Quality

Our analysis includes both automatic tests and manual code analysis for the following aspects:

- Exploits
- Back-doors
- Vulnerability
- Accuracy
- Readability

## Tools

- DE
- Open Zeppelin
- Code Analyzer
- Solidity Code
- Compiler
- Hardhat

# VULNERABILITY CHECKS

| | |
|---|---|
| Design Logic | Passed |
| Compiler warnings | Passed |
| Private user data leaks | Passed |
| Timestamps dependence | Passed |
| Integer overflow and underflow | Passed |
| Race conditions & reentrancy. Cross-function race conditions | Passed |
| Possible delays in data delivery | Passed |
| Oracle calls | Passed |
| Front Running | Passed |
| DoS with Revert | Passed |
| DoS with block gas limit | Passed |
| Methods execution permissions | Passed |
| Economy model | Passed |
| Impact of the exchange rate on the logic | Passed |
| Malicious event log | Passed |
| Scoping and declarations | Passed |
| Uninitialized storage pointers | Passed |
| Arithmetic accuracy | Passed |
| Cross-function race conditions | Passed |
| Safe Zepplin module | Passed |

# RISK CLASSIFICATION

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and acces control issues within the code. The exploitation of these issues by malicious actors may cause serious financial damage to projects that failed to get an audit in time. We categorize these vulnerabilities by the following levels:

## High Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

## Medium Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

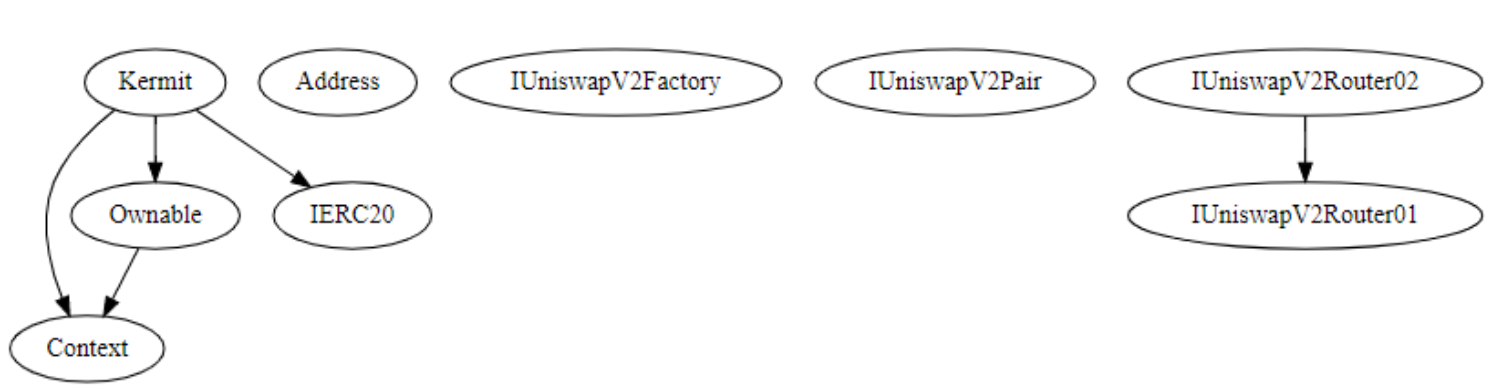
## Low Risk

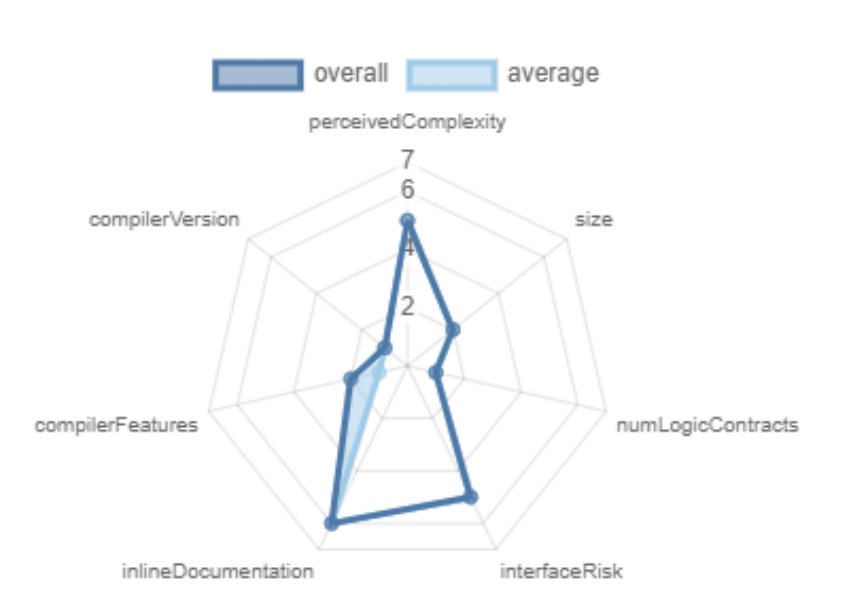
Issues on this level are minor details and warning that can remain unfixed.

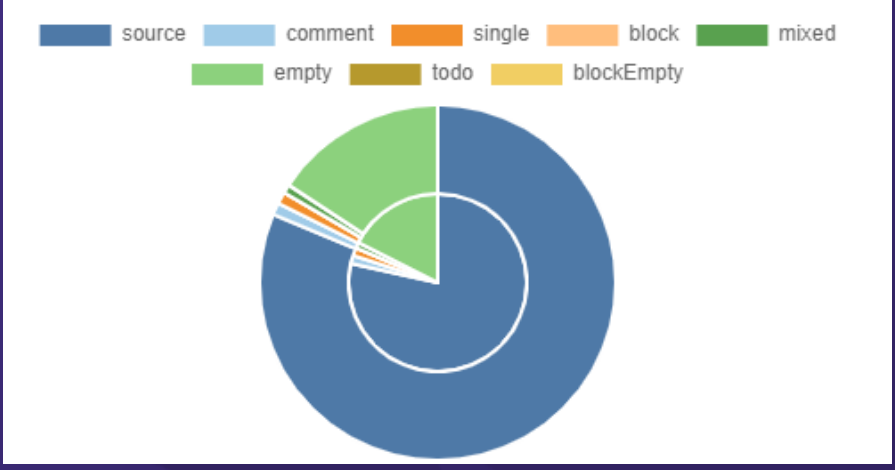
## Informational

Issues on this level are minor details and warning that can remain unfixed.

# INHERITANCE TREES



expelee

# FUNCTION DETAILS

| | | | | | |
|---|---|---|---|---|---|
| **Kermit** | 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 ! | |
| └ | isExcludedFromReward | Public ! | | NO ! | |
| └ | totalReflectionDistributed | Public ! | | NO ! | |
| └ | deliver | Public ! | 🛑 | NO ! | |
| └ | reflectionFromToken | Public ! | | NO ! | |
| └ | tokenFromReflection | Public ! | | NO ! | |
| └ | excludeFromReward | Public ! | 🛑 | onlyOwner | |
| └ | includeInReward | External ! | 🛑 | onlyOwner | |
| └ | <Receive Ether> | External ! | 💶 | NO ! | |
| └ | claimStuckTokens | External ! | 🛑 | onlyOwner | |
| └ | _reflectFee | Private 🔐 | 🛑 | | |
| └ | _getValues | Private 🔐 | | | |
| └ | _getTValues | Private 🔐 | | | |
| └ | _getRValues | Private 🔐 | | | |
| └ | _getRate | Private 🔐 | | | |
| └ | _getCurrentSupply | Private 🔐 | | | |
| └ | _takeLiquidity | Private 🔐 | 🛑 | | |
| └ | _takeMarketing | Private 🔐 | 🛑 | | |
| └ | calculateTaxFee | Private 🔐 | | | |
| └ | calculateLiquidityFee | Private 🔐 | | | |
| └ | calculateMarketingFee | Private 🔐 | | | |
| └ | removeAllFee | Private 🔐 | 🛑 | | |
| └ | setBuyFee | Private 🔐 | 🛑 | | |
| └ | setSellFee | Private 🔐 | 🛑 | | |
| └ | isExcludedFromFee | Public ! | | NO ! | |
| └ | _approve | Private 🔐 | 🛑 | | |
| └ | enableTrading | External ! | 🛑 | onlyOwner | |
| └ | _transfer | Private 🔐 | 🛑 | | |
| └ | swapAndLiquify | Private 🔐 | 🛑 | | |
| └ | swapAndSendMarketing | Private 🔐 | 🛑 | | |
| └ | setSwapTokensAtAmount | External ! | 🛑 | onlyOwner | |
| └ | setSwapEnabled | External ! | 🛑 | onlyOwner | |
| └ | _tokenTransfer | Private 🔐 | 🛑 | | |
| └ | _transferStandard | Private 🔐 | 🛑 | | |
| └ | _transferToExcluded | Private 🔐 | 🛑 | | |
| └ | _transferFromExcluded | Private 🔐 | 🛑 | | |
| └ | _transferBothExcluded | Private 🔐 | 🛑 | | |
| └ | excludeFromFees | External ! | 🛑 | onlyOwner | |
| └ | changeMarketingWallet | External ! | 🛑 | onlyOwner | |
| └ | setBuyFeePercentages | External ! | 🛑 | onlyOwner | |
| └ | setSellFeePercentages | External ! | 🛑 | onlyOwner | |
| └ | enableWalletToWalletTransferWithoutFee | External ! | 🛑 | onlyOwner | |

# MANUAL REVIEW

## Severity Criteria

Expelee assesses the severity of disclosed vulnerabilities according to methodology based on OWASP standarts.

Vulnerabilities are dividend into three primary risk categroies:
High
Medium
Low

High-level considerations for vulnerabilities span the following key areas when conducting assessments:

- Malicious input handling
- Escalation of privileges
- Arithmetic
- Gas use

| Overall Risk Severity | | | | |
|---|---|---|---|---|
| Impact | HIGH | Medium | High | Critical |
| | MEDIUM | Low | Medium | High |
| | LOW | Note | Low | Medium |
| | | LOW | MEDIUM | HIGH |
| | Likelihood | | | |

# FINDINGS

| Findings | Severity | Found |
|---|---|---|
| High Risk | High | 1 |
| Medium Risk | Medium | 0 |
| Low Risk | Low | 5 |
| Suggestion & discussion | Informational | 0 |
| Gas Optimizations | Gas Opt. | 0 |

# HIGH RISK FINDING

## Trade must be enabled by the owner

**Risk : Centralisation**
**Severity : High**

### Overview

Owner must enable trading to enable public to trade their tokens, otherwise no one would be able to buy /sell their tokens except whitelisted wallets.

```
function enableTrading() external onlyOwner{
    require(tradingEnabled == false, "Trading is already enabled");
    tradingEnabled = true;
}
```

### Recommendation

To mitigate this issue you should enable trading before presale or transfer ownership to safu dev for initial days after presale.

# LOW RISK FINDING

**Owner can change buy fees up to 10% sell fees up to 15% at max**

## Severity : Low

### Overview

Functions that allows the owner of the contract to update the buy/sell fees of the contract. These functions assumes that the input parameters are valid and do not exceed the maximum limit of 10% for buy fees and maximum limit of 15% for sell fees.

```
function setBuyFeePercentages(uint256 _taxFeeonBuy↑, uint256 _liquidityFeeonBuy↑, uint256 _marketingFeeonBuy↑) external onlyOwner {
    taxFeeonBuy = _taxFeeonBuy↑;
    liquidityFeeonBuy = _liquidityFeeonBuy↑;
    marketingFeeonBuy = _marketingFeeonBuy↑;
    totalBuyFees = _taxFeeonBuy↑ + _liquidityFeeonBuy↑ + _marketingFeeonBuy↑;
    require(totalBuyFees <= 10, "Buy fees cannot be greater than 10%");
    emit BuyFeesChanged(taxFeeonBuy, liquidityFeeonBuy, marketingFeeonBuy);
}

0 references | Control flow graph | d6a694f5 | ftrace | funcSig
function setSellFeePercentages(uint256 _taxFeeonSell↑, uint256 _liquidityFeeonSell↑, uint256 _marketingFeeonSell↑) external onlyOwner {
    taxFeeonSell = _taxFeeonSell↑;
    liquidityFeeonSell = _liquidityFeeonSell↑;
    marketingFeeonSell = _marketingFeeonSell↑;
    totalSellFees = _taxFeeonSell↑ + _liquidityFeeonSell↑ + _marketingFeeonSell↑;
    require(totalSellFees <= 15, "Sell fees cannot be greater than 15%");
    emit SellFeesChanged(taxFeeonSell, liquidityFeeonSell, marketingFeeonSell);
}
```

### Recommendation

It is recommended to add additional access control measures, such as multi-factor authentication or time-based restrictions, to limit the number of authorized users who can call these functions.

# LOW RISK FINDING

## Owner can exclude/include account from rewards

### Severity : Low

### Overview

Function that allows the owner of the contract to exclude/include an address from receiving dividends

```
function excludeFromReward(address account) public onlyOwner() {
    require(!_isExcluded[account], "Account is already excluded");
    if(_rOwned[account] > 0) {
        _tOwned[account] = tokenFromReflection(_rOwned[account]);
    }
    _isExcluded[account] = true;
    _excluded.push(account);
}

0 references | Control flow graph | 3685d419 | ftrace | funcSig
function includeInReward(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is already included");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

### Recommendation

It is recommended to add additional access control measures, such as multi-factor authentication or time-based restrictions, to limit the number of authorized users who can call these functions. The contract owner account is well secured and only accessible by authorized parties.

# LOW RISK FINDING

## Owner can exclude accounts from fees

## Severity : Low

### Overview

Excludes/Includes an address from the collection of fees

```
function excludeFromFees(address account↑, bool excluded↑) external onlyOwner {
    require(_isExcludedFromFees[account↑] != excluded↑, "Account is already the value of 'excluded'");
    _isExcludedFromFees[account↑] = excluded↑;

    emit ExcludeFromFees(account↑, excluded↑);
}
```

### Recommendation

It is recommended to add additional access control measures, such as multi-factor authentication or time-based restrictions, to limit the number of authorized users who can call these functions. The contract owner account is well secured and only accessible by authorized parties.

# LOW RISK FINDING

## Owner can change swap setting

## Severity : Low

### Overview

**setSwapTokensAtAmount** function allows the owner of the contract to update the value of **swapTokensAtAmount**. **setSwapEnabled** function allows the contract owner to enable or disable the automatic **swapping**.

```
function setSwapTokensAtAmount(uint256 newAmount!) external onlyOwner() {
    require(newAmount! > totalSupply() / 1e5, "SwapTokensAtAmount must be greater than 0.001% of total supply");
    swapTokensAtAmount = newAmount!;
    emit SwapTokensAtAmountUpdated(newAmount!);
}
```

```
function setSwapEnabled(bool _enabled!) external onlyOwner {
    swapEnabled = _enabled!;
    emit SwapEnabledUpdated(_enabled!);
}
```

### Recommendation

If the threshold is set too low, it could result in frequent and unnecessary swaps, which would increase gas fees and potentially lead to losses due to slippage. On the other hand, if the threshold is set too high, it could result in liquidity being insufficient to handle large trades, which could negatively impact the token price and liquidity pool. Be ensure that the contract owner account is well secured and only accessible by authorized parties.

# LOW RISK FINDING

## Owner can withdraw stuck BNB and stuck tokens

### Severity : Low

**Overview**

**claimStuckTokens** allow the contract owner to withdraw locked or stuck ETH and ERC20 tokens from the contract. The functions are properly restricted to only be executed by the contract owner.

```
function claimStuckTokens(address token) external onlyOwner {
    require(token != address(this), "Owner cannot claim native tokens");
    if (token == address(0x0)) {
        payable(msg.sender).sendValue(address(this).balance);    Unchecked
        return;
    }
    IERC20 ERC20token = IERC20(token);
    uint256 balance = ERC20token.balanceOf(address(this));
    ERC20token.transfer(msg.sender, balance);    Unchecked return value
}
```

**Recommendation**

While the functions are currently restricted to only be called by the contract owner, it is recommended to consider implementing a more robust access control mechanism.

# ABOUT EXPELEE

Expelee is a product-based aspirational Web3 start-up. Coping up with numerous solutions for blockchain security and constructing a Web3 ecosystem from deal making platform to developer hosting open platform, while also developing our own commercial and sustainable blockchain.

www.expelee.com

expeleeofficial

expelee

Expelee

expelee

expelee_official

expelee-co

expelee

Building the Futuristic Blockchain Ecosystem

# DISCLAIMER

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 guarantess against the sale of team tokens or the removal of liquidity by the project audited in this document.

Always do your own research and project 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. Alway 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.