expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

KISHU INU

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 with High Issue**
KYC Verification	**No**
Audit Date	**25 May 2023**

CONTRACT DETAILS

Token Name: Kishu Inu 2.0

Symbol: Ki2

Network: Binance Smart Chain

Language: Solidity

Contract Address:
0x5b681922c05Fe87c930180E2Cdbb6D6BB15e783e

Total Supply: 1000000000000

Owner's Wallet:
0x61dC0731569BC2Aa34B54aD3DA6c3dE3214A72DC

Deployer's Wallet:
0x61dC0731569BC2Aa34B54aD3DA6c3dE3214A72DC

OWNER PRIVILEGES

- Trading must be enabled by the owner
- Owner can include/exclude account from reward
- Owner exclude account from fees
- Owner can change fees max 10%
- Owner can change swap settings
- Owner can withdraw claimstuck token from the contract except native token

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

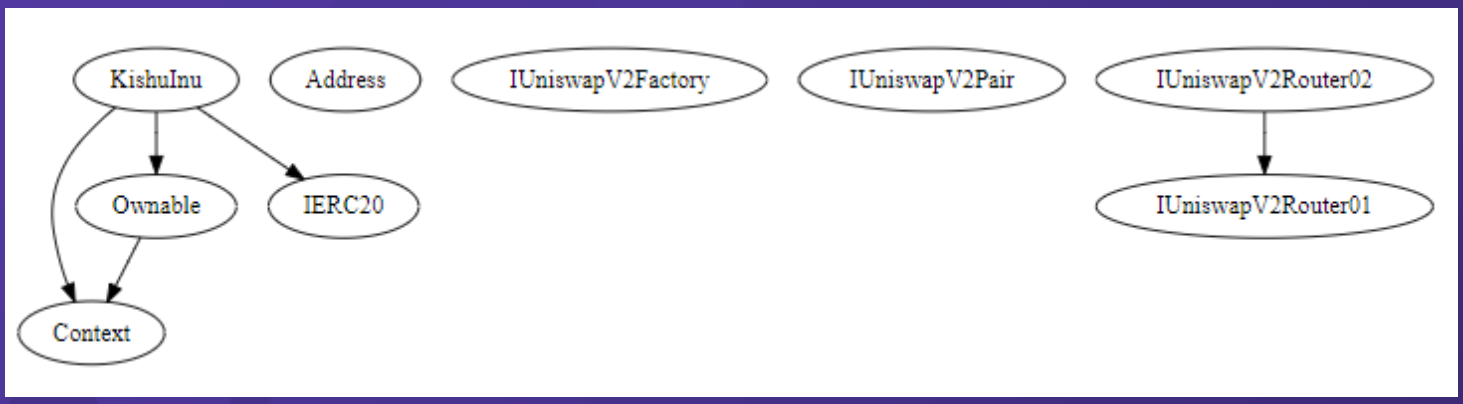
Issues on this level are minor details and warning that can remain unfixed.

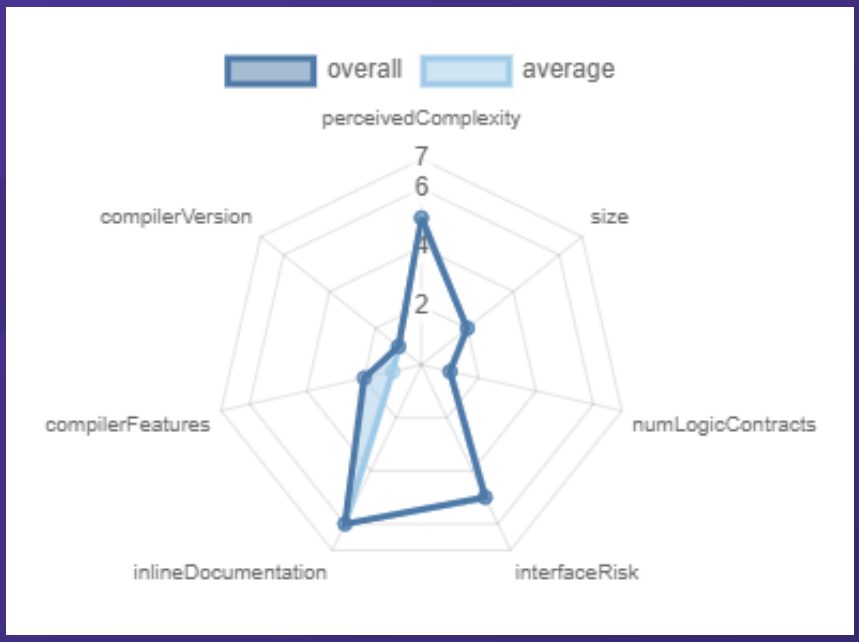
Informational

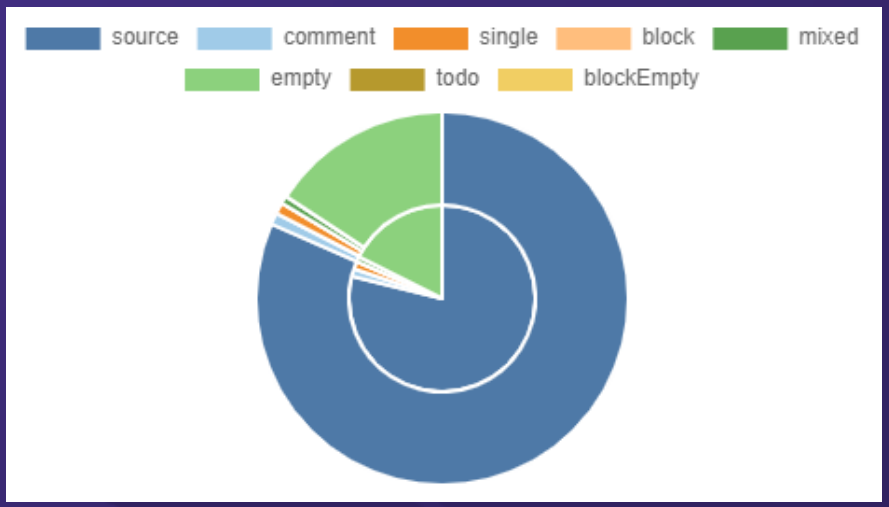
Issues on this level are minor details and warning that can remain unfixed.

expelee

INHERITANCE TREES

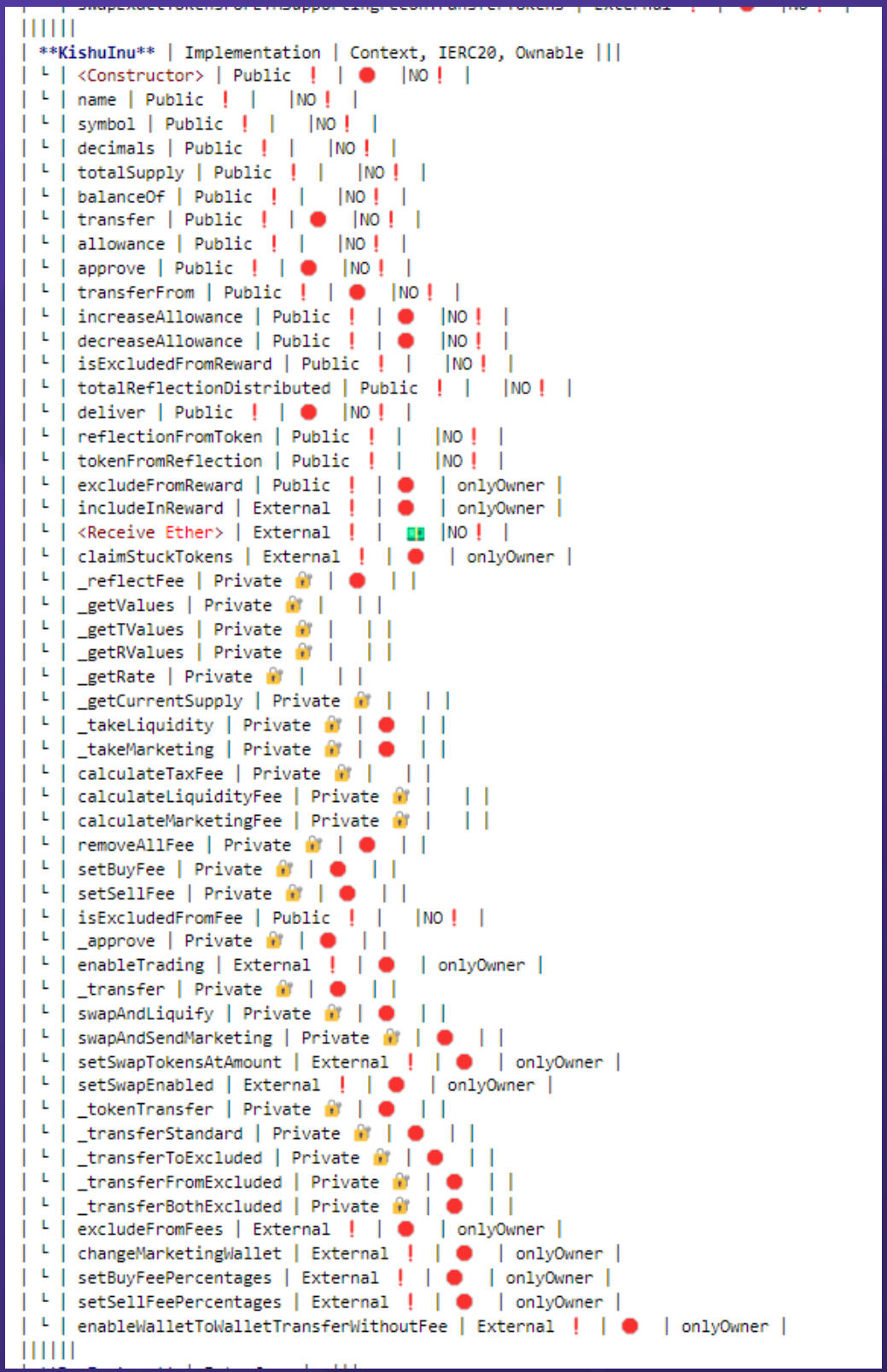






expelee

FUNCTION DETAILS



KishuInu	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

expelee

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			

expelee

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

expelee

HIGH RISK FINDING

Trade must be enable by the owner

Severity : HIGH

Overview

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

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

Recommendation

To mitigating this issue you should enable tradings before presale or transfer ownership to safu dev for initial days after presale

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.