expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

TEDDY INU

expelee

TABLE OF CONTENTS

02 Table of Contents
03 Overview
04 Project Description
05 Social Media Profiles
06 Contract Details
07 Owner Privileges
08 Audit Methodology
09 Vulnerabilities Checklist
10 Risk Classification
11 Inheritence Trees & Risk Overview
12 Function Details
14 Manual Review
15 Findings
23 About Expelee
24 Disclaimer

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:

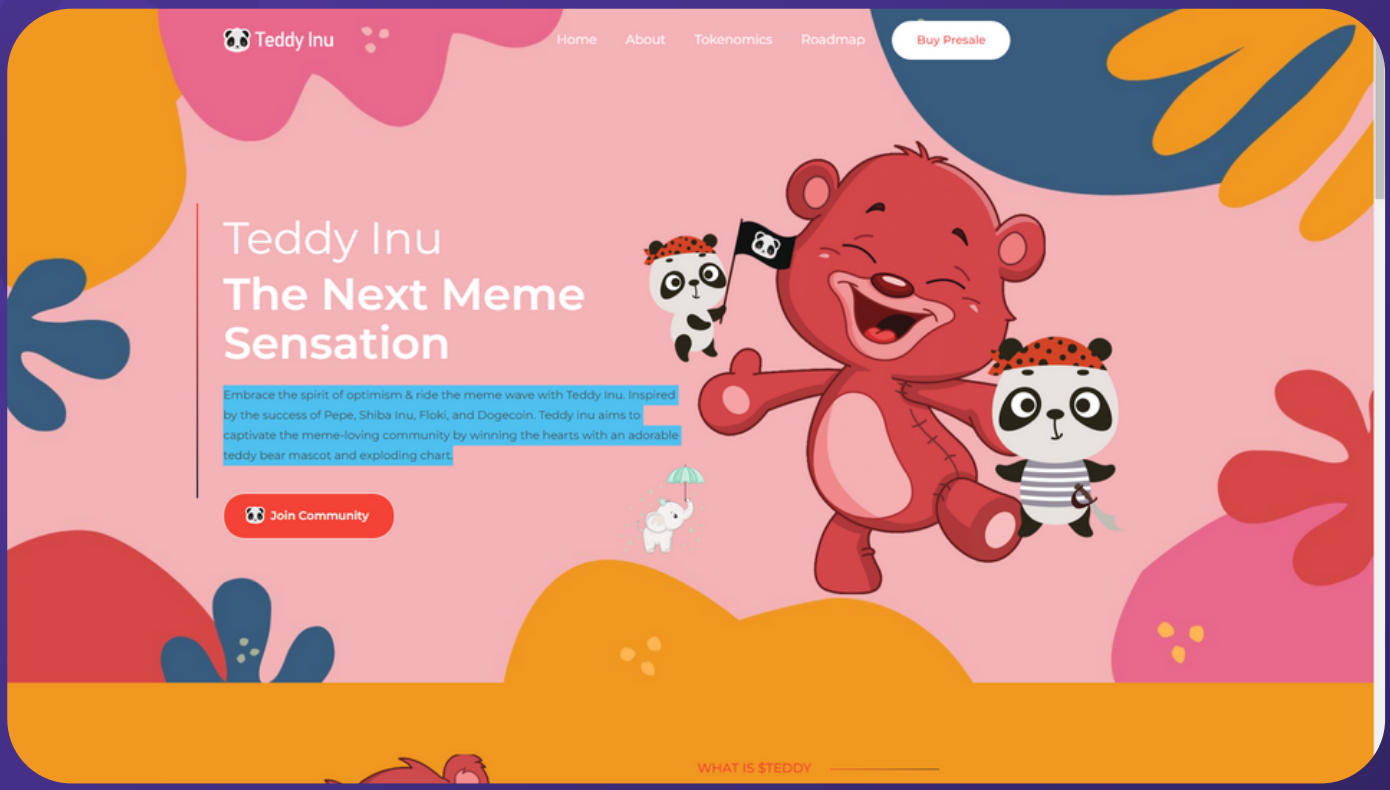
Audit Result	**Passed**
KYC Verification	**-**
Audit Date	**23 May 2023**

PROJECT DESCRIPTION

Embrace the spirit of optimism & ride the meme wave with Teddy Inu. Inspired by the success of Pepe, Shiba Inu, Floki, and Dogecoin. Teddy inu aims to captivate the meme-loving community by winning the hearts with an adorable teddy bear mascot and exploding chart.

SOCIAL MEDIA PROFILES

TEDDY INU



https://t.me/teddy_Inu

https://twitter.com/teddy_Inu

https://teddyinu.xyz/

It's always good to check the social profiles of the project, before making your investment.

Team Expelee

CONTRACT DETAILS

Token Name: Teddy Inu

Symbol: TEDDY

Network: Binance Smart Chain

Language: Solidity

Contract Address:
0x49d27fD68ADbdAd7FdcB54e966ECC224D51C5175

Total Supply: 420000000000000

Owner's Wallet:
0xc6a710726621F67827f64bBaE8cef9A3Fbff40e6

Deployer's Wallet:
0xc6a710726621F67827f64bBaE8cef9A3Fbff40e6

OWNER PRIVILEGES

- Owner can change buy/sell fees max 10%
- Owner can exclude account from fees
- Owner can enable trading
- Owner can change swap token at amount within reasonable limit
- Owner can change swap setting
- Owner can withdraw stuck BNB
- Owner can withdraw stuck tokens
- Owner can update marketing wallet

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

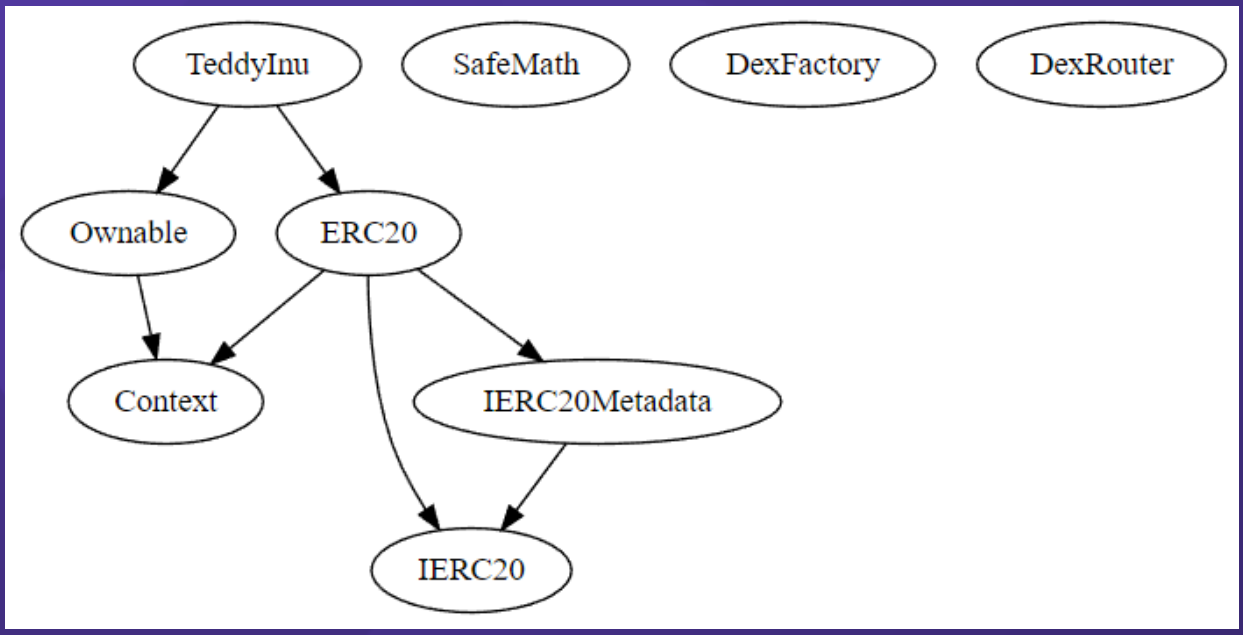
Issues on this level are minor details and warning that can remain unfixed.

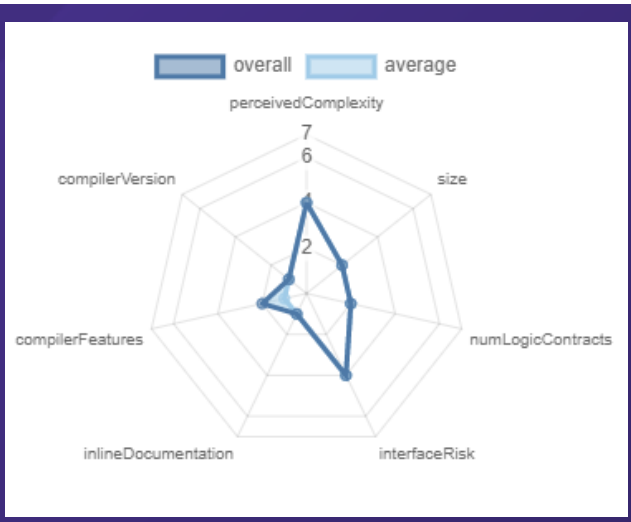
Informational

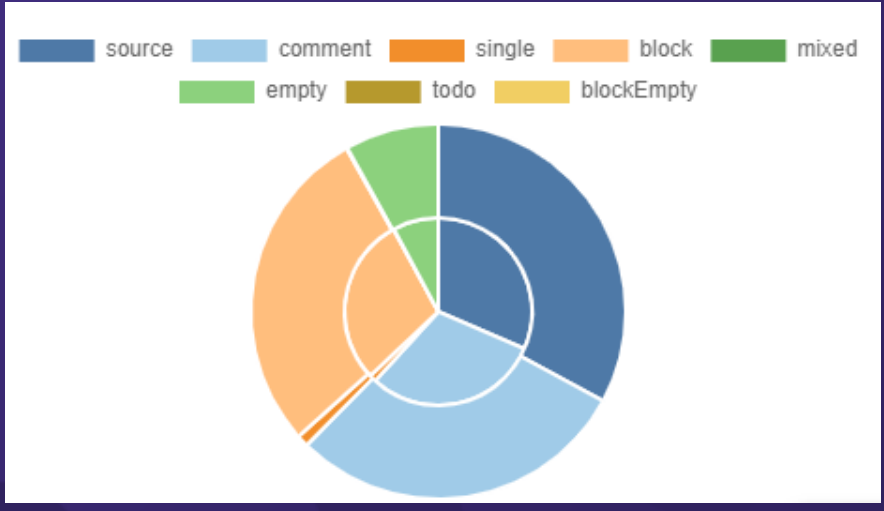
Issues on this level are minor details and warning that can remain unfixed.

expelee

INHERITANCE TREES







expelee

FUNCTION DETAILS

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	_checkOwner	Internal 🔒		
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	
ERC20	Implementation	Context, IERC20, IERC20Metadata		
└	<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 ❗
└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_spendAllowance	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
└	_afterTokenTransfer	Internal 🔒	🛑	
IERC20Metadata	Interface	IERC20		
└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	decimals	External ❗		NO ❗
IERC20	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		

expelee

FUNCTION DETAILS

SafeMath	Library				
└	tryAdd	Internal 🔒			
└	trySub	Internal 🔒			
└	tryMul	Internal 🔒			
└	tryDiv	Internal 🔒			
└	tryMod	Internal 🔒			
└	add	Internal 🔒			
└	sub	Internal 🔒			
└	mul	Internal 🔒			
└	div	Internal 🔒			
└	mod	Internal 🔒			
└	sub	Internal 🔒			
└	div	Internal 🔒			
└	mod	Internal 🔒			
DexFactory	Interface				
└	createPair	External ❗	🛑	NO ❗	
DexRouter	Interface				
└	factory	External ❗		NO ❗	
└	WETH	External ❗		NO ❗	
└	addLiquidityETH	External ❗	💵	NO ❗	
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗	
TeddyInu	Implementation	ERC20, Ownable			
└	<Constructor>	Public ❗	🛑	ERC20	
└	enableTrading	External ❗	🛑	onlyOwner	
└	setMarketingWallet	External ❗	🛑	onlyOwner	
└	setBuyFees	External ❗	🛑	onlyOwner	
└	setSellFees	External ❗	🛑	onlyOwner	
└	setSwapTokensAtAmount	External ❗	🛑	onlyOwner	
└	toggleSwapping	External ❗	🛑	onlyOwner	
└	setWhitelistStatus	External ❗	🛑	onlyOwner	
└	checkWhitelist	External ❗		NO ❗	
└	_takeTax	Internal 🔒	🛑		
└	_transfer	Internal 🔒	🛑		
└	manageTaxes	Internal 🔒	🛑		
└	swapToETH	Internal 🔒	🛑		
└	withdrawStuckETH	External ❗	🛑	onlyOwner	
└	withdrawStuckTokens	External ❗	🛑	onlyOwner	
└	<Receive Ether>	External ❗	💵	NO ❗	

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	0
Medium Risk	Medium	0
Low Risk	Low	6
Suggestion & discussion	Informational	1
Gas Optimizations	Gas Opt.	0

LOW RISK FINDING

Owner can change buy/sell fees max 10%

Severity : Low

Overview

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

```
function setBuyFees(uint256 _marketingTax1) external onlyOwner {
    require(_marketingTax1 <= 10, "can not set higher than 10%");
    buyTaxes.marketingTax = _marketingTax1;
    totalBuyFees = _marketingTax1;
}

ftrace | funcSig
function setSellFees(uint256 _marketingTax1) external onlyOwner {
    require(_marketingTax1 <= 10, "can not set higher than 10%");
    sellTaxes.marketingTax = _marketingTax1;
    totalSellFees = _marketingTax1;
}
```

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 accounts from fees

Severity : Low

Overview

Excludes/Includes an address from the collection of fees

```
function setWhitelistStatus(address _wallet,bool _status) external onlyOwner {
    whitelisted[_wallet] = _status;
}
```

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 enable trading

Severity : Low

Overview

Function enables trading by setting the **tradingStatus** true

```
function enableTrading() external onlyOwner {
    tradingStatus = true;
}
```

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 token at amount within reasonable limit

Severity : Low

Overview

setSwapTokensAtAmount function allows the owner of the contract to update the value of **swapTokensAtAmount**.

```
function setSwapTokensAtAmount(uint256 _newAmount) external onlyOwner {
    require(
        _newAmount > 0,
        "Radiate : Minimum swap amount must be greater than 0!"
    );
    swapTokensAtAmount = _newAmount;
}
```

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 change swap setting

Severity : Low

Overview

Functions allows the contract owner to enable or disable the automatic swapping.

```
function toggleSwapping() external onlyOwner {
    swapAndLiquifyEnabled = (swapAndLiquifyEnabled == true) ? false : true;
}
```

Recommendation

It is recommended to 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

withdrawStuckETH() and **withdrawStuckTokens()**, which 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 withdrawStuckETH() external onlyOwner {
    (bool success, ) = address(msg.sender).call{
        value: address(this).balance
    }("");
    require(success, "transfering ETH failed");
}

ftrace | funcSig
function withdrawStuckTokens(address erc20_token) external onlyOwner {
    bool success = IERC20(erc20_token).transfer(
        msg.sender,
        IERC20(erc20_token).balanceOf(address(this))
    );
    require(success, "trasfering tokens failed!");
}
```

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. Also Owner can withdraw native token you can add require and check should not receive native tokens.

INFORMATIONAL FINDING

Lack of Event usage

Severity : Informational

Overview

Detect missing events for critical access control parameters

Recommendation

It is recommend to use Emit an event for critical parameter changes.

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.