

expellee

Building the Futuristic **Blockchain Ecosystem**

SECURITY AUDIT REPORT



DONALD TRUMP INU

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	Inheritance Trees & Risk Overview	_____
12	Function Details	_____
13	Manual Review	_____
14	Findings	_____
26	About Expelee	_____
27	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	Not Done
Audit Date	18 April 2023

PROJECT DESCRIPTION

Donald Trump Inu a Meme Token that aims to go beyond Doge and Shiba and revolutionize DeFi while empowering democracy.



SOCIAL MEDIA PROFILES

DONALD TRUMP INU



https://t.me/Donald_Trump_inu

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

Team Expelee

CONTRACT DETAILS

Token Name: Donald Trump Inu

Symbol: -

Network: Binance Smart Chain

Language: Solidity

Contract Address:

0xa292dB8b53485E8370952dd127A88d04C10f7689

Total Supply: 10000000000

Contract SHA-256 Checksum: -

Owner's Wallet:

0x664d6aE3Ca3aF79f4042312B5EAfF94b08Bb7a9a

Deployer's Wallet:

0x664d6aE3Ca3aF79f4042312B5EAfF94b08Bb7a9a

Testnet:

<https://testnet.bscscan.com/address/0x9D7F96aEdE80d9013d09F4c81222Ed40A483390c>

OWNER PRIVILEGES

- The owner can withdraw any token and stuck ETH from the contract
- The owner can exclude accounts from fees
- The owner can set fees with limit up to 10%
- Trading must be enabled by the owner
- The owner can't set swap token amount "0" but can set very low.
- The owner can change swap settings

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 Zeppelin module	Passed

RISK CLASSIFICATION

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and access 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.

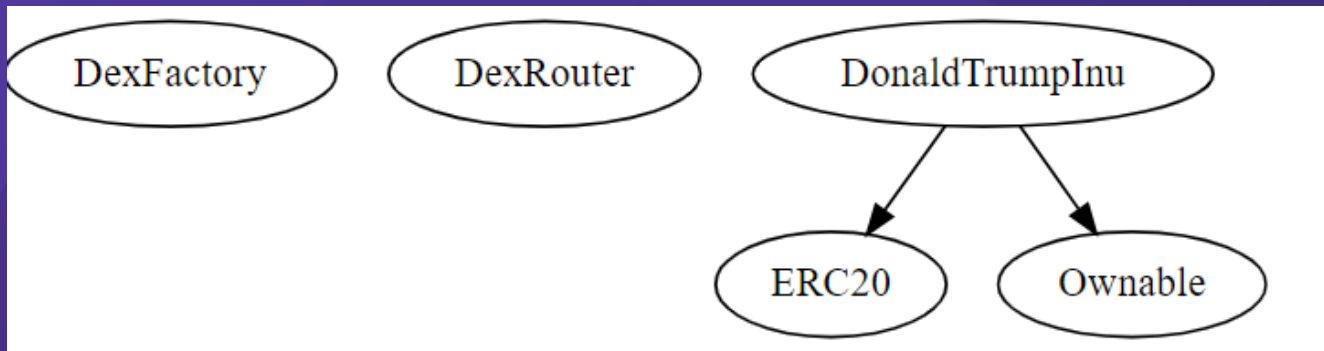
Low Risk

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

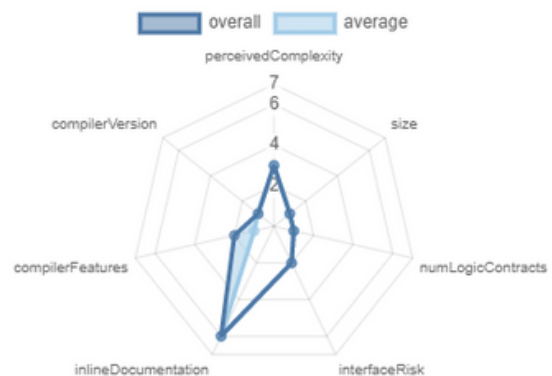
Informational

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

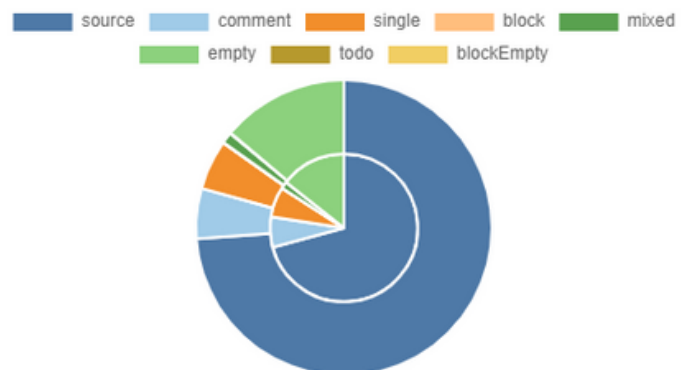
INHERITANCE TREES





Risk



Source Lines (sloc vs. nsloc)



FUNCTION DETAILS

Symbol	Meaning
	Function can modify state
	Function is payable

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
DexFactory	Interface			
L	createPair	External !	●	NO !
DexRouter	Interface			
L	factory	External !		NO !
L	WETH	External !		NO !
L	addLiquidityETH	External !	■	NO !
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External !	●	NO !
DonaldTrumpInu	Implementation	ERC20, Ownable		
L		Public !	●	ERC20
L	enableTrading	External !	●	onlyOwner
L	setMarketingWallet	External !	●	onlyOwner
L	setBuyFees	External !	●	onlyOwner
L	setSellFees	External !	●	onlyOwner
L	setSwapTokensAtAmount	External !	●	onlyOwner
L	toggleSwapping	External !	●	onlyOwner
L	setWhitelistStatus	External !	●	onlyOwner
L	checkWhitelist	External !		NO !
L	_takeTax	Internal 🔒	●	
L	_transfer	Internal 🔒	●	
L	manageTaxes	Internal 🔒	●	
L	swapToETH	Internal 🔒	●	
L	withdrawStuckETH	External !	●	onlyOwner
L	withdrawStuckTokens	External !	●	onlyOwner
L		External !	■	NO !

MANUAL REVIEW

Severity Criteria

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

Vulnerabilities are divided into three primary risk categories:

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	1
Low Risk	● Low	5
Suggestion & discussion	● Informational	5
Gas Optimizations	● Gas Opt.	0

MEDIUM RISK FINDING

The owner can withdraw any token and stuck ETH from the contract

Severity : Medium

Overview

The owner's ability to withdraw any token and any stuck ETH from the contract can be a potential security risk, especially if not implemented with appropriate safeguards. Even if there is an onlyowner modifier, it is still important to consider the potential risks associated with allowing the owner to withdraw funds, especially if the contract holds a large amount of assets.

```
function withdrawStuckETH() external onlyOwner {
    (bool success, ) = address(msg.sender).call{
        value: address(this).balance
    }("");
    require(success, "transferring ETH failed");
}
```

```
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:

Use multi-sig wallets or time-locks to ensure that the withdrawal function can only be executed after multiple owners have approved it or after a certain amount of time has elapsed. It is recommended that you still implement some safeguards to ensure that the owner cannot withdraw funds in a malicious or fraudulent manner.

LOW RISK FINDING

The owner can exclude accounts from fees

Severity : Low

Overview

Authorizing privileged roles to exclude accounts from fees. After excluding the user from accounts, the user trades without paying a any fee and the other user sees it).

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

Recommendation

You should carefully manage the private key of the owner's account. You should use powerful security mechanism that will prevent a single user from accessing the contract owner functions. That risk can be prevented by temporarily locking the contract or renouncing ownership

LOW RISK FINDING

The owner can set fees with limit up to 10%

Severity : Low

Overview

Owner can update fees up to 10% with using **setBuyFees()**, **setSellFees()** but the require message not correct and does not reflect the truth

```
#!/ owner can set fees up to 10%
ftrace | funcSig
function setBuyFees(uint256 _marketingTax!) external onlyOwner {
    require(_marketingTax! <= 10, "can not set higher than 10%");
    buyTaxes.marketingTax = _marketingTax!;
    totalBuyFees = _marketingTax!;
}

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

Recommendation

To make the require message reflect the truth and provide more clarity

LOW RISK FINDING

Trading must be enabled by the owner

Severity : Low

Overview

The **enableTrading** function can be used to enable trading on a specific pair by passing the pair's address as a parameter

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

Recommendation

Recommended to include appropriate access control mechanisms to ensure that only authorized users can modify the smart contract's critical parameters. This can help prevent unauthorized changes to the smart contract that can potentially cause issues or put the business at risk.

LOW RISK FINDING

The owner can't set swap token amount "0" but can set very low.

Severity : Low

Overview

Represents the minimum amount of tokens that need to be held by the contract before a swap can occur. The purpose of this variable is to prevent small transactions from triggering a swap, which could result in unnecessary gas fees and potentially affect the market price of the token.

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

Recommendation

Consider adding a lower bound to **swapTokensAtAmount**. **swapTokensAtAmount** is greater than 0, but you may also want to consider adding a lower bound to prevent the minimum swap amount from being set too low.

LOW RISK FINDING

The owner can change swap settings

Severity : Low

Overview

The owner can set new swap settings. In some cases, it may disrupt the functionality of this contract

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

Recommendation

Recommended to include appropriate access control mechanisms to ensure that only authorized users can modify the smart contract's critical parameters.

INFORMATIONAL RISK FINDING

Missing events arithmetic

Severity : Informational

Overview

Events are used to emit information about an action that has occurred on the blockchain, so that it can be observed by external systems or users. The contract was found to be missing these events on the function .

```
/// owner can set fees up to 10%
ftrace | funcSig
function setBuyFees(uint256 _marketingTax!) external onlyOwner {
    require(_marketingTax! <= 10, "can not set higher than 10%");
    buyTaxes.marketingTax = _marketingTax!;
    totalBuyFees = _marketingTax!;
}

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

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

Recommendation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed. Emit an event for critical parameter changes.

INFORMATIONAL RISK FINDING

Missing zero address validation

Severity : Informational

Overview

Detect missing zero address validation.

```
function enableTrading(address _pairAddress!) external onlyOwner {  
    pairAddress = _pairAddress!;  
    tradingStatus = true;  
}
```

Recommendation

Check that the address is not zero.

INFORMATIONAL RISK FINDING

Low-level calls

Severity : Informational

Overview

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

```
function manageTaxes() internal {  
    swapToETH(balanceOf(address(this)));  
    (bool success, ) = MarketingWallet.call{value: address(this).balance}(  
        ""  
    );  
}
```

Recommendation

Avoid low-level calls. Check the call success. If the call is meant for a contract, check for code existence.

INFORMATIONAL RISK FINDING

Too many digits

Severity : Informational

Overview

Literals with many digits are difficult to read and review.

```
uint256 private constant _totalSupply = 100000000000 * 1e18;
```

Recommendation

While 1_ether looks like 1 ether, it is 10 ether. As a result, it's likely to be used incorrectly.

INFORMATIONAL RISK FINDING

Outdated versions and floating pragma;

Severity : Informational

Overview

Outdated versions were detected **pragma solidity ^0.8.17;**

```
pragma solidity ^0.8.17;
```

Recommendation

Consider using the latest version of Solidity for testing. Should lock pragmas to a specific compiler version. Besides, consider the known compiler bugs in the following references and check whether the contracts include those bugs.

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.

The logo for Expelee, featuring the word "expelee" in a stylized font. The "ex" is in white, and "pelee" is in orange. The letters are bold and modern.

Building the Futuristic **Blockchain Ecosystem**