expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

Antibot

TOKEN OVERVIEW

Risk Findings

Severity	Found
High	1
Medium	1
Low	0
Informational	0

Centralization Risks

Owner Privileges	Description
Can Owner Set Taxes >25% ?	Not Detected
Owner needs to enable trading ?	Yes, owner needs to enable trades
Can Owner Disable Trades ?	Not Detected
Can Owner Mint ?	Not Detected
Can Owner Blacklist ?	Not Detected
Can Owner set Max Wallet amount ?	Not Detected
Can Owner Set Max TX amount ?	Not Detected

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 Risk**
KYC Verification	**-**
Audit Date	**13 September 2023**

CONTRACT DETAILS

Token Address: 0xcd148ce7D8Ff9EAea1a50361b17d6f5A1C6334B0

Name: Antibot

Symbol: ANTI

Decimals: 18

Network: ETH

Token Type: ERC20

Owner: 0x7cbb6d29db85a30eebe05723920d470a824d6340

Deployer: 0x7cbb6d29db85a30eebe05723920d470a824d6340

Token Supply: 15,000,000

Checksum:
8b9e85bd770dabb84288969efc38bbdc292301c7

Testnet version:
The tests conducted were performed on the contract deployed on the Binance Smart Chain (BSC) Testnet.
https://testnet.bscscan.com/token/0x4dA9D1858FD80e5Cea89aD962cA5D94d2660f5C5

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

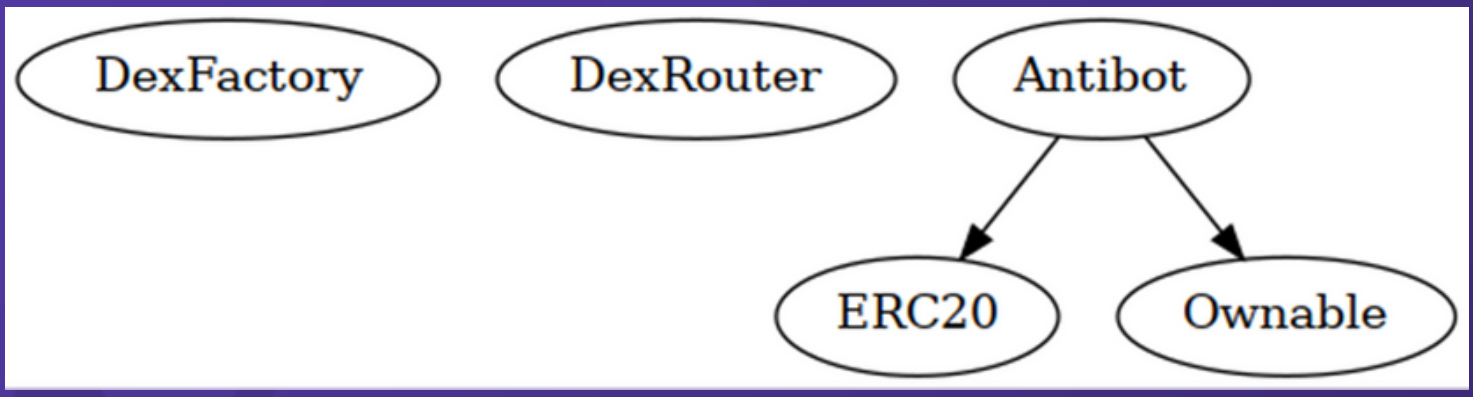
Issues on this level are minor details and warning that can remain unfixed.

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**
DexFactory	Interface			
└	createPair	External ❗	🛑	NO ❗
DexRouter	Interface			
└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
└	addLiquidityETH	External ❗	💵	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗
🛑	NO ❗
Antibot	Implementation	ERC20, Ownable		
└	<Constructor>	Public ❗	🛑	ERC20
└	setmarketingWallet	External ❗	🛑	onlyOwner
└	enableTrading	External ❗	🛑	onlyOwner
└	setBuyTaxes	External ❗	🛑	onlyOwner
└	setSellTaxes	External ❗	🛑	onlyOwner
└	setSwapTokensAtAmount	External ❗	🛑	onlyOwner
└	toggleSwapping	External ❗	🛑	onlyOwner
└	setWhitelistStatus	External ❗	🛑	onlyOwner
└	checkWhitelist	External ❗		NO ❗
└	setmaxTxAmount	External ❗	🛑	onlyOwner
└	_takeTax	Internal 🔒	🛑	
└	_transfer	Internal 🔒	🛑	
└	internalSwap	Internal 🔒	🛑	
└	swapToETH	Internal 🔒	🛑	
└	<Receive Ether>	External ❗	💵	NO ❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

expelee

TESTNET VERSION

Adding Liquidity ✅
Tx:
https://testnet.bscscan.com/tx/0x9c538d2fbed374ea6b5cc09ea3e6764e87579bb071949d479bb14fa0db149f60

==========

Buying when excluded from fees ✅
Tx (0% tax):
https://testnet.bscscan.com/tx/0x03a9ef44a246f6ff077a24441db85904ddc626192dab905dc916deecbf8e9c93

==========

Selling when excluded from fees ✅
Tx (0% tax):
https://testnet.bscscan.com/tx/0x97790e003890ac1e2ad1037d9700fc190b025e7ce163a5f07346db9042038cdb

==========

Transferring when excluded from fees ✅
Tx (0% tax):
https://testnet.bscscan.com/tx/0xf7c625d33a0c680e45e5c83dc0a6a9a2ad1cc06c04c7be2bebe12d6b11076425

==========

Buying ✅
Tx (0-8% tax):
https://testnet.bscscan.com/tx/0x44f650702f4ead3d926d22c4218c575bfecdd2ba8d922b63b370cc36eca9bd12

TESTNET VERSION

Selling ✅
Tx (0-8% tax):
https://testnet.bscscan.com/tx/0x83f2239507605a259a17c6b186f290548f64fb3b6a74fea51169169e809925b7

==========

Transferring ✅
Tx (0% tax):
https://testnet.bscscan.com/tx/0xe6a2fe8aee92170df853aa2ce27da21163c5c1508b904fc3505d7f2271aad2b4

==========

Internal swap (ETH sent to marketing wallet) ✅
Tx:
https://testnet.bscscan.com/tx/0x83f2239507605a259a17c6b186f290548f64fb3b6a74fea51169169e809925b7

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			

HIGH RISK FINDING

Category: Centralization
Subject: Trades are disabled by default
Status: Resolved (Owner is a safu developer)
Impact: High

Overview:

The contract has been structured such that all trading is disabled by default, necessitating the contract owner's manual intervention to enable trading. This can lead to a situation where, if trades remain disabled, token holders won't be able to buy, sell, or trade their tokens, causing a severe impact on the token's usability and market liquidity.

```
  function enableTrading() external onlyOwner {
require(!tradingEnabled, "Trading is already enabled");
    tradingEnabled = true;
    startTradingBlock = block.number;
  }
```

Suggestion:

To mitigate this risk, it is recommended that trading be enabled before the token presale. This can be achieved by invoking the "enableTrading" function or by transferring ownership of the contract to a third-party that has established trust with the community, such as a Certified SAFU developer. This reduces the concentration of power and the potential for malicious actions, thereby promoting a more decentralized and fair environment for all participants.

MEDİUM RISK FINDING

Category: Centralization
Subject: Buys can be disabled
Status: Resolved (Owner is a safu developer)
Impact: Medium

Overview:

The contract has a function for adjusting *_maxTxAmount*. *_maxTxAmount* is the maximum amount of tokens a wallet can buy through DEX in a single transaction.
A malicious owner may intentionally set this variable to zero in order to completely disable buys

```
function setmaxTxAmount(uint256 maxtx) external onlyOwner {
    _maxTxAmount = maxtx;
    emit maxTxAmountChanged(maxtx);
}
```

Suggestion:

To mitigate this risk, set a lower bound for _maxTxAmount to prevent this variable being set to zero.

```
function setmaxTxAmount(uint256 maxtx) external onlyOwner {
    require(maxtx >= totalSupply() / 1000, "Can't set maxtx lower than 0.1% of supply");
    _maxTxAmount = maxtx;
    emit maxTxAmountChanged(maxtx);
}
```

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.