expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

Boo

TOKEN OVERVIEW

Risk Findings

Severity	Found
High	1
Medium	0
Low	0
Informational	2

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**
Audit Date	**12 February 2024**

CONTRACT DETAILS

Token Address: --

Name: Baby Boo

Symbol: BOO

Decimals: 18

Network: --

Token Type: --

Owner: --

Deployer: --

Token Supply: 500000000

Checksum: B2032c616934aeb47e6039f76b20d321

Testnet:
https://testnet.bscscan.com/token/0x2172fb514a5b41bbf4fe4373c89e55e5d2674a1b#code

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.

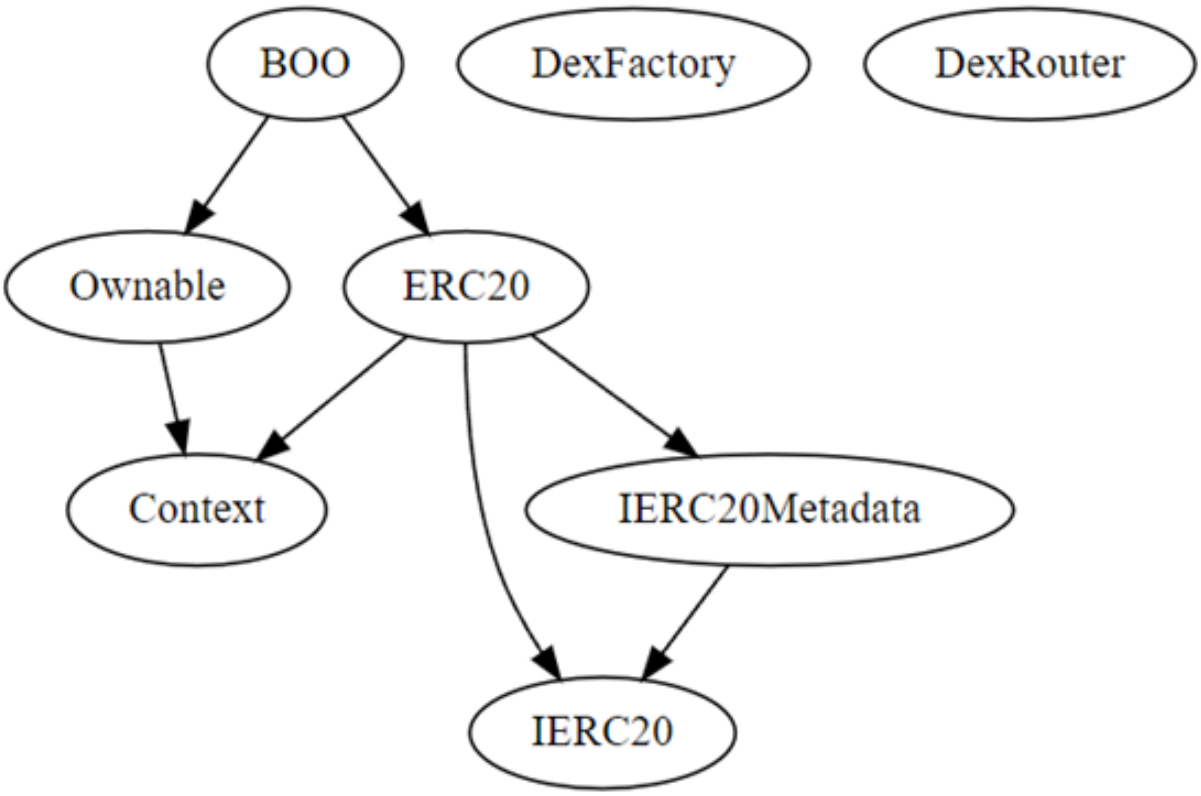
Low Risk

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



STATIC ANALYSİS

A static analysis of the code was performed using Slither. No issues were found.

```
INFO:Detectors:
Reentrancy in BOO.internalSwap() (BOO.sol#820-833):
        External calls:
        - swapToETH(balanceOf(address(this))) (BOO.sol#824)
                - uniswapRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(_amount,0,path,address(this),block.timestamp) (BOO.sol#841-847)
        - (success) = marketingWallet.call{value: address(this).balance}() (BOO.sol#825)
        External calls sending eth:
        - (success) = marketingWallet.call{value: address(this).balance}() (BOO.sol#825)
        State variables written after the call(s):
        - isSwapping = false (BOO.sol#832)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in BOO._transfer(address,address,uint256) (BOO.sol#791-818):
        External calls:
        - internalSwap() (BOO.sol#814)
                - uniswapRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(_amount,0,path,address(this),block.timestamp) (BOO.sol#841-847)
                - (success) = marketingWallet.call{value: address(this).balance}() (BOO.sol#825)
        External calls sending eth:
        - internalSwap() (BOO.sol#814)
                - (success) = marketingWallet.call{value: address(this).balance}() (BOO.sol#825)
        Event emitted after the call(s):
        - Transfer(from,to,amount) (BOO.sol#470)
                - super._transfer(_from,_to,toTransfer) (BOO.sol#817)
Reentrancy in BOO.internalSwap() (BOO.sol#820-833):
        External calls:
        - swapToETH(balanceOf(address(this))) (BOO.sol#824)
                - uniswapRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(_amount,0,path,address(this),block.timestamp) (BOO.sol#841-847)
        - (success) = marketingWallet.call{value: address(this).balance}() (BOO.sol#825)
        External calls sending eth:
        - (success) = marketingWallet.call{value: address(this).balance}() (BOO.sol#825)
        Event emitted after the call(s):
        - TransferFailed(marketingWallet,address(this).balance) (BOO.sol#828)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Different versions of Solidity are used:
        - Version used: ['0.8.19', '^0.8.0']
        - 0.8.19 (BOO.sol#621)
        - ^0.8.0 (BOO.sol#3)
        - ^0.8.0 (BOO.sol#30)
        - ^0.8.0 (BOO.sol#115)
        - ^0.8.0 (BOO.sol#200)
        - ^0.8.0 (BOO.sol#230)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

expelee

STATIC ANALYSİS

```
INFO:Detectors:
Context._msgData() (BOO.sol#20-22) is never used and should be removed
ERC20._burn(address,uint256) (BOO.sol#510-526) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (BOO.sol#3) allows old versions
Pragma version^0.8.0 (BOO.sol#30) allows old versions
Pragma version^0.8.0 (BOO.sol#115) allows old versions
Pragma version^0.8.0 (BOO.sol#200) allows old versions
Pragma version^0.8.0 (BOO.sol#230) allows old versions
Pragma version0.8.19 (BOO.sol#621) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in BOO.internalSwap() (BOO.sol#820-833):
        - (success) = marketingWallet.call{value: address(this).balance}() (BOO.sol#825)
Low level call in BOO.withdrawStuckETH() (BOO.sol#850-855):
        - (success) = address(msg.sender).call{value: address(this).balance}() (BOO.sol#851-853)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function DexRouter.WETH() (BOO.sol#633) is not in mixedCase
Event BOO.marketingWalletChanged(address) (BOO.sol#688) is not in CapWords
Parameter BOO.setmarketingWallet(address)._newmarketing (BOO.sol#713) is not in mixedCase
Parameter BOO.setBuyTaxes(uint256)._marketingTax (BOO.sol#730) is not in mixedCase
Parameter BOO.setSellTaxes(uint256)._marketingTax (BOO.sol#736) is not in mixedCase
Parameter BOO.setSwapTokensAtAmount(uint256)._newAmount (BOO.sol#741) is not in mixedCase
Parameter BOO.setWhitelistStatus(address,bool)._wallet (BOO.sol#755) is not in mixedCase
Parameter BOO.setWhitelistStatus(address,bool)._status (BOO.sol#756) is not in mixedCase
Parameter BOO.checkWhitelist(address)._wallet (BOO.sol#762) is not in mixedCase
Parameter BOO.swapToETH(uint256)._amount (BOO.sol#836) is not in mixedCase
Parameter BOO.withdrawStuckTokens(address).BEP20_token (BOO.sol#857) is not in mixedCase
Constant BOO._totalSupply (BOO.sol#661) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
BOO.slitherConstructorVariables() (BOO.sol#656-868) uses literals with too many digits:
        - swapTokensAtAmount = _totalSupply / 100000 (BOO.sol#676)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Slither:BOO.sol analyzed (8 contracts with 93 detectors), 31 result(s) found
```

TESTNET VERSION

1- Approve (passed):
https://testnet.bscscan.com/tx/0x296bc80a52886201c69a8d0cb0978d53291be1fd7e030a713ffa041e2932961a

2- Increase Allowance (passed):
https://testnet.bscscan.com/tx/0x04809f3f6a03c21ded25642025246f8713d23c8fb281946bf333b0cc7365ccdc

3- Decrease Allowance (passed):
https://testnet.bscscan.com/tx/0x3b763bad5d9e8298b0344fcb3a4150788ea465d08e79c017c13663cce070f723

4- Enable Trading (passed):
https://testnet.bscscan.com/tx/0x3613e2e9743a7521b610c70f325c0c66b34c43d4b976662e3753ad69a36f2642

5- Set Buy Taxes (passed):
https://testnet.bscscan.com/tx/0xde49ef21ac13ffb660fbdc172c7c7fed1397d82eaa924e57c131cd45d6bd5161

6- Set Sell Taxes (passed):
https://testnet.bscscan.com/tx/0xc6ccf031c3620f76edaf5b146e9dfe5848a07f31fb046f08024944edb2af1c21

7- Transfer (passed):
https://testnet.bscscan.com/tx/0x1dc0d581acca49de33e940f61d31d151630379b8af153473d0bdfe8041291dcb

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

Centralization – Enabling Trades

Severity: High

function: EnableTrading

Status: Open

Overview:

The EnableTrading function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

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

}
```

Suggestion

To reduce centralization and potential manipulation, consider one of the following approaches:

1.Automatically enable trading after a specified condition, such as the completion of a presale, is met.

2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can give investors more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

INFORMATIONAL & OPTIMIZATIONS

Optimization

Severity: Informational

Subject: Floating Pragma Solidity version

Status: Open

Overview:
It is considered best practice to pick one compiler version and stick with it. With a floating pragma, contracts may accidentally be deployed using an outdated.

pragma solidity ^0.8.19;

Suggestion:
Adding the latest constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

INFORMATIONAL & OPTIMIZATIONS

Optimization

Severity: Optimization

subject: Remove Unused Code

Status: Open

Overview:

Unused variables are allowed in Solidity, and they do. not pose a direct security issue. It is the best practice though to avoid them.

```
function _msgData() internal view virtual returns (bytes
calldata) {
return msg.data;
  }
function _burn(address account, uint256 amount) internal
virtual {
require(account != address(0), "ERC20: burn from the zero
address");

    _beforeTokenTransfer(account, address(0), amount);

uint256 accountBalance = _balances[account];
require(accountBalance >= amount, "ERC20: burn amount
exceeds balance");
    unchecked {
      _balances[account] = accountBalance - amount;
```

INFORMATİONAL & OPTİMİZATİONS

```
// Overflow not possible: amount <= accountBalance <=
totalSupply.
_totalSupply -= amount;
}

emit Transfer(account, address(0), amount);

_afterTokenTransfer(account, address(0), amount);
}
```

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.