Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

GROK THE GOAT

TOKEN OVERVIEW

Risk Findings

Severity	Found
High	0
Medium	0
Low	2
Informational	2

Centralization Risks

Owner Privileges	Description
Can Owner Set Taxes >25% ?	Not Detected
Owner Can enable trading ?	Not Detected
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

expelee

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**
KYC Verification	**-**
Audit Date	**09 Jan 2024**

CONTRACT DETAILS

Token Name: GROK THE GOAT

Symbol: GOAT

Network: BscScan

Decimals: 9

Token Type: BEP – 20

Contract Address:
0x742D784Add59b87D3EC3D7FC663C3F61fD06e5F4

Total Supply: 100,000,000

Owner's Wallet:
0xD4545DdBC582e5d80bE8d11e9e6B46871Ea91566

Deployer's Wallet:
0xD4545DdBC582e5d80bE8d11e9e6B46871Ea91566

Checksum: Ac6659e84744e0102ab19c1d1e78a87a

Testnet.
https://testnet.bscscan.com/address/0x420dbf33c6be1cb393ff032eae5b5c4af7a5f2f5#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

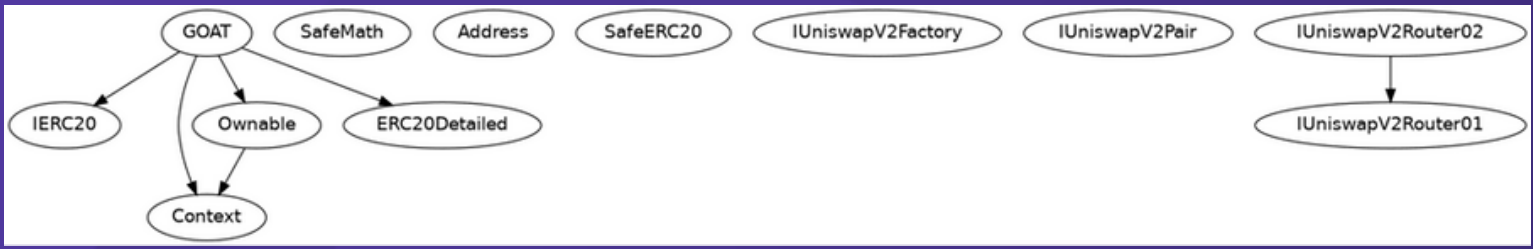
Issues on this level are minor details and warning that can remain unfixed.

Informational

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

expelee

INHERITANCE TREES



expelee

STATIC ANALYSIS

```
INFO:Detectors:
GOAT.setMarketingFeePercent(uint256,uint256) (GOAT.sol#495-499) should emit an event for:
        - buyMarketingFee = updatedBuyFee (GOAT.sol#496)
        - sellMarketingFee = updatedSellFee (GOAT.sol#497)
GOAT.changeNumTokensSellToAddToLiquidity(uint256) (GOAT.sol#512-515) should emit an event for:
        - numTokensSellToAddToLiquidity = _numTokensSellToAddToLiquidity (GOAT.sol#514)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
GOAT.setMaarketingAddress(address).wallet (GOAT.sol#502) lacks a zero-check on :
                - marketingAddress = wallet (GOAT.sol#504)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in GOAT._transfer(address,address,uint256) (GOAT.sol#541-599):
        External calls:
        - swapAndLiquify(contractTokenBalance) (GOAT.sol#575)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokensToLiquify,0,path,address(this),block.timestamp) (GOAT.sol#613-619
)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (GOAT.sol#575)
                - address(marketingAddress).transfer(bnbBalance) (GOAT.sol#622)
        State variables written after the call(s):
        - marketingFee = buyMarketingFee (GOAT.sol#582)
        - marketingFee = sellMarketingFee (GOAT.sol#583)
Reentrancy in GOAT.transferFrom(address,address,uint256) (GOAT.sol#481-485):
        External calls:
        - _transfer(sender,recipient,amount) (GOAT.sol#482)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokensToLiquify,0,path,address(this),block.timestamp) (GOAT.sol#613-619
)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (GOAT.sol#482)
                - address(marketingAddress).transfer(bnbBalance) (GOAT.sol#622)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (GOAT.sol#483)
                - _allowances[towner][spender] = amount (GOAT.sol#631)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
INFO:Detectors:
Reentrancy in GOAT._transfer(address,address,uint256) (GOAT.sol#541-599):
        External calls:
        - swapAndLiquify(contractTokenBalance) (GOAT.sol#575)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokensToLiquify,0,path,address(this),block.timestamp) (GOAT.sol#613-619
)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (GOAT.sol#575)
                - address(marketingAddress).transfer(bnbBalance) (GOAT.sol#622)
        Event emitted after the call(s):
        - Transfer(sender,recipient,TotalSent) (GOAT.sol#589)
        - Transfer(sender,address(this),taxAmount) (GOAT.sol#590)
        - Transfer(sender,recipient,amount) (GOAT.sol#596)
Reentrancy in GOAT.transferFrom(address,address,uint256) (GOAT.sol#481-485):
        External calls:
        - _transfer(sender,recipient,amount) (GOAT.sol#482)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokensToLiquify,0,path,address(this),block.timestamp) (GOAT.sol#613-619
)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (GOAT.sol#482)
                - address(marketingAddress).transfer(bnbBalance) (GOAT.sol#622)
        Event emitted after the call(s):
        - Approval(towner,spender,amount) (GOAT.sol#632)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (GOAT.sol#483)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.isContract(address) (GOAT.sol#146-152) uses assembly
        - INLINE ASM (GOAT.sol#150)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (GOAT.sol#264) is too similar to IU
niswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (GOAT.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
GOAT.constructor() (GOAT.sol#439-462) uses literals with too many digits:
        - _totalSupply = 1000000000 * (10 ** 9) (GOAT.sol#441)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
GOAT._totalSupply (GOAT.sol#407) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:GOAT.sol analyzed (12 contracts with 93 detectors), 34 result(s) found
```

TESTNET VERSION

1- Approve (passed):
https://testnet.bscscan.com/tx/0xb744dcef0bb0fee1a7a4b26d001735ac0dee5daeb5108b6f497c86ac96973b36

2- Increase Allowance (passed):
https://testnet.bscscan.com/tx/0xc845df51b85ecdb1e6177a27fc6ea8cbb51449fca442b03d9bdca154d49eb031

3- Decrease Allowance (passed):
https://testnet.bscscan.com/tx/0x90b78fdaf348c467b7552e8708e76b224072577798ccb70e8e9fdd37cf9aa132

4- Exclude From AMMS (passed):
https://testnet.bscscan.com/tx/0xee0a4b3cc6348c58e67cf5138be7be3e4720ada36be05d715a4c1f210674a219

5- Set Marketing Address (passed):
https://testnet.bscscan.com/tx/0xc803224cf9ed74f387f0a96e87d5992e2360c168cb1770a04db117764e5b337f

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			

LOW RISK FINDING

Missing Events

Centralization
Severity: Low
Subject: Missing Events
Status:Open

Overview:
They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

```
 function setMarketingFeePercent(uint256 updatedBuyFee,
uint256 updatedSellFee) external onlyOwner {
    buyMarketingFee = updatedBuyFee;
    sellMarketingFee = updatedSellFee;

  }
function setMaarketingAddress(address payable wallet)
external onlyOwner
  {
    marketingAddress = wallet;
  }
function setSwapAndLiquifyEnabled(bool _enabled) public
onlyOwner {
    swapAndLiquifyEnabled = _enabled;
emit SwapAndLiquifyEnabledUpdated(_enabled);
  }
```

LOW RISK FINDING

Missing Zero Address

Centralization

Severity: Low

Subject: Zero Check

Status:Open

Overview:

Functions can take a zero address as a parameter (0x00000...). If a function parameter of address type is not properly validated by checking for zero addresses, there could be serious consequences for the contract's functionality.

```
function setMaarketingAddress(address payable wallet)
external onlyOwner
 {
   marketingAddress = wallet;
 }
```

INFORMATIONAL RISK FINDING

Optimization
Severity: Informational
Subject: Floating Pragma.
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 ^0.8.13;

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 RISK FINDING

Optimization
Severity: Informational
Subject: Remove Safe Math
Status: Open
Line: 19-55

Overview:
Compiler version above 0.8.0 can control arithmetic overflow/underflow, I t is recommended to remove the unwanted code to avoid high gas fees.

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.