expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

SPACE GROK

TOKEN OVERVIEW

Risk Findings

Severity	Found
High	1
Medium	0
Low	1
Informational	2

Centralization Risks

Owner Privileges	Description
Can Owner Set Taxes >25% ?	Not Detected
Owner Can enable trading ?	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 with high risk**
KYC Verification	**-**
Audit Date	**09 Jan, 2024**

CONTRACT DETAILS

Token Name: SPACE GROK

Symbol: SpaceGrok

Network: BscScan

Decimal: 9

Token Type: BEP - 20

Token Address:
0xcd79b119DD4727fFe50A10CFbcA60a31c37Fd91e

Total Supply: 420,690,000,000

Owner's Wallet:
0x1942952fF5acaaA7CEa3906F60a5CBa28CAfaCD9

Deployer's Wallet:
0x1942952fF5acaaA7CEa3906F60a5CBa28CAfaCD9

CheckSum:
Ae1c3a4fbb6e83e8393a57617b5a5b17

Testnet.
https://testnet.bscscan.com/address/0x528e8976ae7c27e7
7426234c9739e5721e0a054d#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

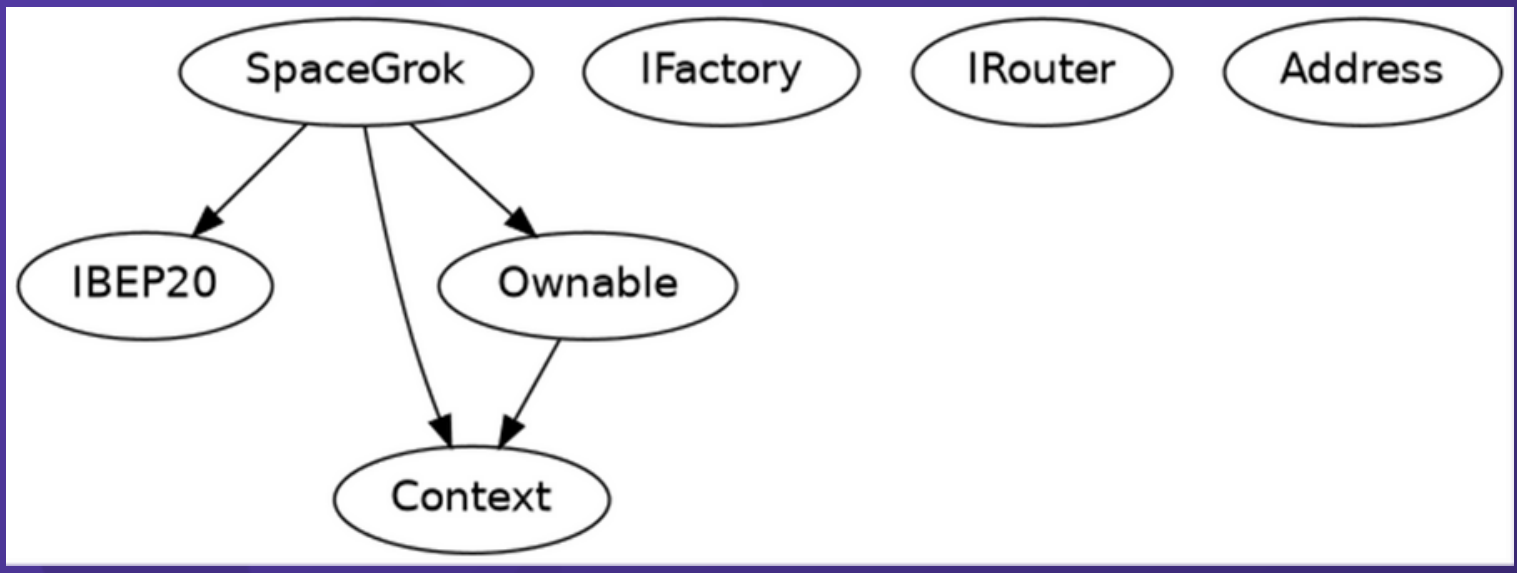
Issues on this level are minor details and warning that can remain unfixed.

Informational

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

expelee

INHERITANCE TREES



expelee

STATIC ANALYSIS

```
INFO:Detectors:
SpaceGrok.swapAndLiquify(uint256,SpaceGrok.Taxes) (SpaceGrok.sol#627-666) performs a multiplication on the result of a division:
        - unitBalance = deltaBalance / (denominator - temp.liquidity) (SpaceGrok.sol#644)
        - bnbToAddLiquidityWith = unitBalance * temp.liquidity (SpaceGrok.sol#645)
SpaceGrok.swapAndLiquify(uint256,SpaceGrok.Taxes) (SpaceGrok.sol#627-666) performs a multiplication on the result of a division:
        - unitBalance = deltaBalance / (denominator - temp.liquidity) (SpaceGrok.sol#644)
        - marketingAmt = unitBalance * 2 * temp.marketing (SpaceGrok.sol#652)
SpaceGrok.swapAndLiquify(uint256,SpaceGrok.Taxes) (SpaceGrok.sol#627-666) performs a multiplication on the result of a division:
        - unitBalance = deltaBalance / (denominator - temp.liquidity) (SpaceGrok.sol#644)
        - devAmt = unitBalance * 2 * temp.dev (SpaceGrok.sol#657)
SpaceGrok.swapAndLiquify(uint256,SpaceGrok.Taxes) (SpaceGrok.sol#627-666) performs a multiplication on the result of a division:
        - unitBalance = deltaBalance / (denominator - temp.liquidity) (SpaceGrok.sol#644)
        - opsAmt = unitBalance * 2 * temp.ops (SpaceGrok.sol#662)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
SpaceGrok.addLiquidity(uint256,uint256) (SpaceGrok.sol#668-681) ignores return value by router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0
,0,deadWallet,block.timestamp) (SpaceGrok.sol#673-680)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
SpaceGrok.allowance(address,address).owner (SpaceGrok.sol#252) shadows:
        - Ownable.owner() (SpaceGrok.sol#51-53) (function)
SpaceGrok._approve(address,address,uint256).owner (SpaceGrok.sol#539) shadows:
        - Ownable.owner() (SpaceGrok.sol#51-53) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
SpaceGrok.updatedeadline(uint256) (SpaceGrok.sol#322-326) should emit an event for:
        - deadline = _deadline (SpaceGrok.sol#325)
SpaceGrok.updateSwapTokensAtAmount(uint256) (SpaceGrok.sol#722-725) should emit an event for:
        - swapTokensAtAmount = amount * 10 ** _decimals (SpaceGrok.sol#724)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
SpaceGrok.constructor(address)._pair (SpaceGrok.sol#208) lacks a zero-check on :
                - pair = _pair (SpaceGrok.sol#211)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in SpaceGrok._transfer(address,address,uint256) (SpaceGrok.sol#549-584):
        External calls:
        - swapAndLiquify(swapTokensAtAmount,sellTaxes) (SpaceGrok.sol#575)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (SpaceGrok.sol#673-680)
                - (success) = recipient.call{value: amount}() (SpaceGrok.sol#114)
```

```
INFO:Detectors:
Context._msgData() (SpaceGrok.sol#36-39) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
SpaceGrok._rTotal (SpaceGrok.sol#144) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX % _tTotal))
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state
INFO:Detectors:
Pragma version^0.8.19 (SpaceGrok.sol#7) 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 Address.sendValue(address,uint256) (SpaceGrok.sol#111-116):
        - (success) = recipient.call{value: amount}() (SpaceGrok.sol#114)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IRouter.WETH() (SpaceGrok.sol#83) is not in mixedCase
Struct SpaceGrok.valuesFromGetValues (SpaceGrok.sol#181-195) is not in CapWords
Function SpaceGrok.EnableTrading() (SpaceGrok.sol#315-320) is not in mixedCase
Parameter SpaceGrok.updatedeadline(uint256)._deadline (SpaceGrok.sol#322) is not in mixedCase
Parameter SpaceGrok.updateSwapEnabled(bool)._enabled (SpaceGrok.sol#727) is not in mixedCase
Parameter SpaceGrok.rescueAnyBEP20Tokens(address,address,uint256)._tokenAddr (SpaceGrok.sol#736) is not in mixedCase
Parameter SpaceGrok.rescueAnyBEP20Tokens(address,address,uint256)._to (SpaceGrok.sol#736) is not in mixedCase
Parameter SpaceGrok.rescueAnyBEP20Tokens(address,address,uint256)._amount (SpaceGrok.sol#736) is not in mixedCase
Constant SpaceGrok._decimals (SpaceGrok.sol#140) is not in UPPER_CASE_WITH_UNDERSCORES
Variable SpaceGrok.genesis_block (SpaceGrok.sol#148) is not in mixedCase
Constant SpaceGrok._name (SpaceGrok.sol#156) is not in UPPER_CASE_WITH_UNDERSCORES
Constant SpaceGrok._symbol (SpaceGrok.sol#157) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (SpaceGrok.sol#37)" inContext (SpaceGrok.sol#31-40)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
SpaceGrok.slitherConstructorVariables() (SpaceGrok.sol#119-743) uses literals with too many digits:
        - _tTotal = 420690000000 * 10 ** _decimals (SpaceGrok.sol#143)
SpaceGrok.slitherConstructorVariables() (SpaceGrok.sol#119-743) uses literals with too many digits:
        - swapTokensAtAmount = 250000000 * 10 ** 9 (SpaceGrok.sol#146)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```

expelee

STATIC ANALYSIS

```
INFO:Detectors:
SpaceGrok._lastSell (SpaceGrok.sol#135) is never used in SpaceGrok (SpaceGrok.sol#119-743)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
Loop condition i < _excluded.length (SpaceGrok.sol#528) should use cached array length instead of referencing `length` member of the storage array.
 Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cache-array-length
INFO:Detectors:
SpaceGrok._tTotal (SpaceGrok.sol#143) should be constant
SpaceGrok.deadWallet (SpaceGrok.sol#151) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
SpaceGrok.pair (SpaceGrok.sol#138) should be immutable
SpaceGrok.router (SpaceGrok.sol#137) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:SpaceGrok.sol analyzed (7 contracts with 93 detectors), 46 result(s) found
```

TESTNET VERSION

1- Approve (passed):
https://testnet.bscscan.com/tx/0xc67a9cfb3b93e05cd6146bbcb044dc82c870e1e86521148a40a00d0c287cd891

2- Increase Allowance (passed):
https://testnet.bscscan.com/tx/0x29643ac6fd7524634fd5a49dba27084f9209c8fd59cc3f620ca1196994a606a9

3- Decrease Allowance (passed):
https://testnet.bscscan.com/tx/0x227feaf26eaea5fd15cf0f773dbf5cf98748c680141848d058ac7f6c68f23462

4- Enable Trading (passed):
https://testnet.bscscan.com/tx/0x87b897fe9f6444b4c0661010e63dcf0ef7c8dc1794797f172d5dc71ed1329705

5- Exclude From Fee (passed):
https://testnet.bscscan.com/tx/0x266f0b4350686fd6293488907a1dc739647edad742f122b56f39884973fe8f03

6- Exclude From Rewards (passed):
https://testnet.bscscan.com/tx/0xe6e7d644ed3c4e5dd71bd6f721c311e9235407ef4c33f2bd44d735b9a70d4a97

7- Transfer (passed):
https://testnet.bscscan.com/tx/0x627904cbdd081f07cca29be753b16df3e66b2c1c8da7c322692cf31ceacd9c7c

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

Enabling Trades

Category: Centralization
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, "Cannot re-enable trading.");
   tradingEnabled = true;
   swapEnabled = true;
   genesis_block = 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.

HIGH RISK FINDING

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.

LOW RISK FINDING

Missing Events

Category: 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 updateMarketingWallet(address newWallet) external
onlyOwner {
require(newWallet != address(0),"Fee Address cannot be
dead address.");
    marketingWallet = newWallet;
  }
```

LOW RISK FINDING

```
function updateDevWallet(address newWallet) external
onlyOwner {
require(newWallet != address(0),"Fee Address cannot be
dead address.");
 devWallet = newWallet;
 }
 function updateOpsWallet(address newWallet) external
onlyOwner {
require(newWallet != address(0),"Fee Address cannot be
dead address.");
 opsWallet = newWallet;
 }
 function updatedeadline(uint256 _deadline) external
onlyOwner {
require(!tradingEnabled, "Can't change when trading has
started.");
require(_deadline < 5,"Deadline should be less than 5 Blocks.");
 deadline = _deadline;
 }
```

INFORMATIONAL RISK FINDING

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) {
    this; // silence state mutability warning without
generating bytecode - see
https://github.com/ethereum/solidity/issues/2691
return msg.data;
  }
}
event FeesChanged();
event UpdatedRouter(address oldRouter, address
newRouter);
```

INFORMATIONAL RISK FINDING

Category: 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.

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.