



AUDIT REPORT

SecureWise

GROOT (GROOT)



Quick Result

Quick Result	Status
Owner can mint ?	Not Detected
Owner can update tax over 25% ?	Not Detected
Owner can pause trade ?	Yes
Owner can enable trading ?	Yes
Owner can add Blacklist ?	Yes
Owner can set Max Tx ?	Yes
Owner can set Max Wallet Amount ?	Yes
KYC ?	Not Done

Page 6, 10 for more details

Groot (GROOT) as **PASSED** the smart contract audit with **Critical Issues**.

Findings

Risk Classification	Description
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, of the contract and its functions. Must be fixed as soon as possible.
Medium	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Must be fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Informational	A vulnerability that have informational character but is not effecting any of the code

Severity	Found	Pending	Resolved
High	6	0	0
Medium	1	0	0
Low	0	0	0
Informational	3	0	0
Total	10	0	0

Contents

01	Quick Result
02	Findings
04	Overview
05	Auditing Approach and Methodologies
06	Findings Summary
07	Function Privileges
08	Inheritance Graph
10	Manual Review
18	Disclaimer

Overview

Token Name: Groot (**GROOT**)

Language: Solidity

Contract Address: 0x1728C4c3F59B9f4F687D1953c243756086a85eF7

Network: Binance Smart Chain

Supply: 4206900000000000

KYC: Not done

Website: <http://www.grootbnb.xyz>

Twitter: https://twitter.com/grootbnb_

Telegram: <https://t.me/Grootbnb>

Report Date: July 2, 2023

Auditing Approach and Methodologies

SecureWise has performed starting with analyzing the code, issues, code quality, and libraries. Reviewed line-by-line by our team. Finding any potential issue like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

Methodology

- Understanding the size, scope and functionality of your project's source code
- Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Testing and automated analysis of the Smart Contract to determine proper logic has been followed throughout the whole process
- Deploying the code on testnet using multiple live test
- Analyzing a program to determine the specific input that causes different parts of a program to execute its functions.
- Checking whether all the libraries used in the code are on the latest version.

Goals

Smart Contract System is secure, resilient and working according to the specifications and without any vulnerabilities.

Risk Classification

High: Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, of the contract and its functions. Must be fixed as soon as possible.

Medium: Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Must be fixed as soon as possible.











Low: Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.

Informational: A vulnerability that have informational character but is not affecting any of the code

Findings Summary

SecureWise has applied the automated and manual analysis of Smart Contract and were reviewed for common contract vulnerabilities and centralized exploits

Findings

	Owner can set a blacklist and can use it to block any account from trading.
	Trade must be enabled by the owner.
	Owner can change transaction lock-time without limit and pause trade.
	Owner can can set max wallet token amount "0".
	Owner can set max transaction amount very low no check limit.
	Owner can change swap pair status with use swapPairList enable or disable.
	Auto liquidity is transferred to an externally owned account.
	Owner can set buy/sell/transfer with limit up to 25%.
	Owner has the authority to exclude account from fees.
	Owner has the authority to change swap settings.

Page 10 for more details

Groot (GROOT) as PASSED the smart contract audit with Critical Issues.

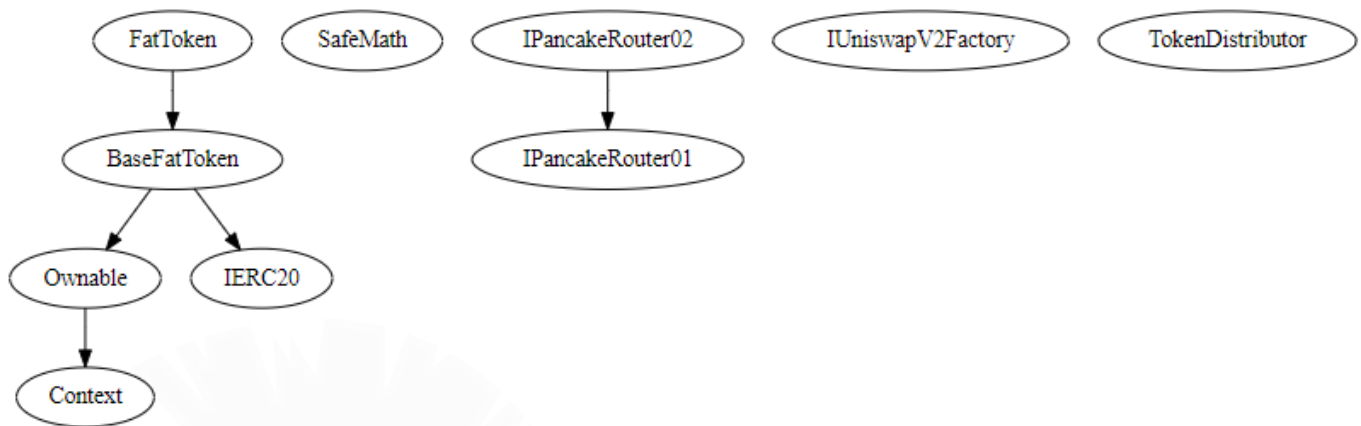
Function Privileges

```

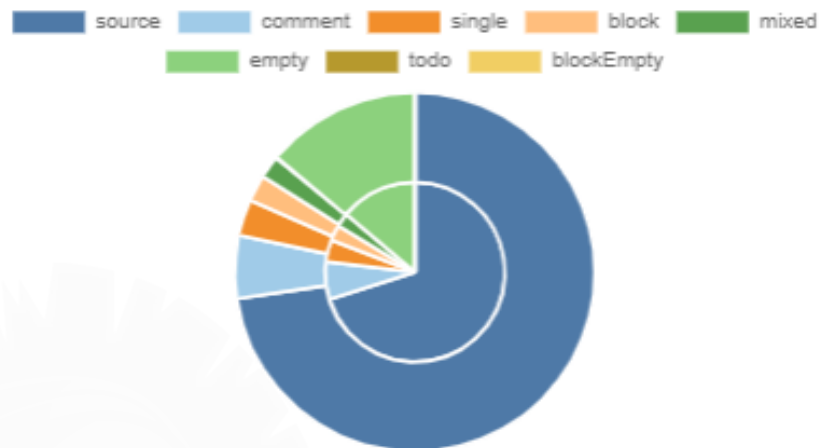
||||| |
| **BaseFatToken** | Implementation | IERC20, Ownable |||
| L | setFundAddress | External ! | ● | onlyOwner |
| L | changeSwapLimit | External ! | ● | onlyOwner |
| L | changeWalletLimit | External ! | ● | onlyOwner |
| L | launch | External ! | ● | onlyOwner |
| L | disableSwapLimit | Public ! | ● | onlyOwner |
| L | disableWalletLimit | Public ! | ● | onlyOwner |
| L | disableChangeTax | Public ! | ● | onlyOwner |
| L | completeCustoms | External ! | ● | onlyOwner |
| L | transfer | External ! | ● | NO ! |
| L | transferFrom | External ! | ● | NO ! |
| L | balanceOf | Public ! | | NO ! |
| L | allowance | Public ! | | NO ! |
| L | approve | Public ! | ● | NO ! |
| L | _approve | Private 🔒 | ● | |
| L | setFeeWhiteList | External ! | ● | onlyOwner |
| L | multi_bclist | Public ! | ● | onlyOwner |
|||||
| **TokenDistributor** | Implementation | |||
| L | <Constructor> | Public ! | ● | NO ! |
|||||
| **FatToken** | Implementation | BaseFatToken |||
| L | <Constructor> | Public ! | ● | NO ! |
| L | transfer | Public ! | ● | NO ! |
| L | transferFrom | Public ! | ● | NO ! |
| L | setkb | Public ! | ● | onlyOwner |
| L | isReward | Public ! | | NO ! |
| L | setAirDropEnable | Public ! | ● | onlyOwner |
| L | _basicTransfer | Internal 🔒 | ● | |
| L | setAirdropNumbs | Public ! | ● | onlyOwner |
| L | setEnableTransferFee | Public ! | ● | onlyOwner |
| L | _transfer | Private 🔒 | ● | |
| L | setTransferFee | Public ! | ● | onlyOwner |
| L | _tokenTransfer | Private 🔒 | ● | |
| L | swapTokenForFund | Private 🔒 | ● | lockTheSwap |
| L | _takeTransfer | Private 🔒 | ● | |
| L | setSwapPairList | External ! | ● | onlyOwner |
| L | <Receive Ether> | External ! | 🟢 | NO ! |

```

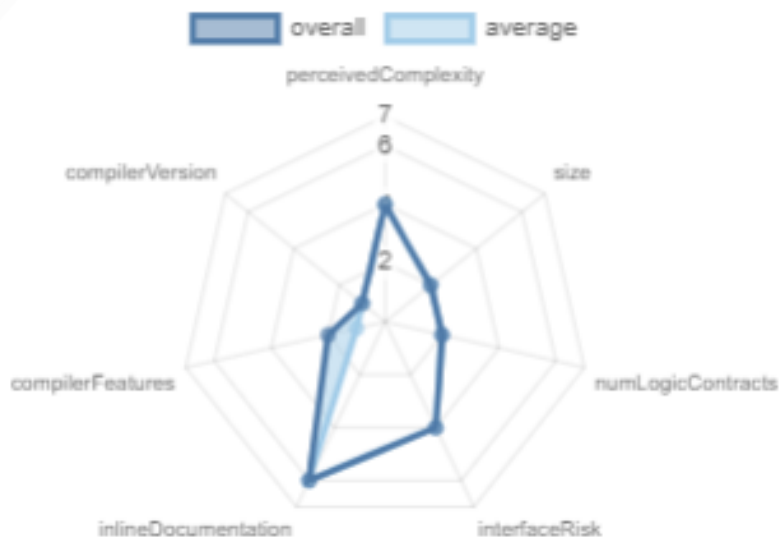

Inheritance Graph



Source Lines



Risk



Manual Review

High Risk

Owner can set a blacklist and can use it to block any account.

```
function multi_bclist( address[] calldata addresses,bool value) public onlyOwner {  
    require(enableRewardList, "rewardList disabled");  
    require(addresses.length < 201);  
    for (uint256 i; i < addresses.length; ++i) {  
        _rewardList[addresses[i]] = value;  
    }  
}
```

Description

It allows the contract owner to add or remove addresses from the reward list in bulk by providing an array of addresses and a boolean value.

Recommendation

Evaluate the usage of the **onlyOwner** modifier in the **multi_bclist** function to ensure that only the contract owner can use this function. or Discuss the centralization risk introduced by having the contract owner as the sole authority to modify the reward list. Highlight the potential concerns of favoritism or unfair treatment that may arise if the contract owner misuses their power.

Manual Review

High Risk

Trade must be enabled by the owner.

```
function launch() external onlyOwner {  
    require(startTradeBlock == 0, "already started");  
    startTradeBlock = block.number;  
}
```

Description

The contract requires manual activation by the owner in order for investors to start trading. If trading remains disabled, token holders will not have the ability to execute any transactions with their tokens.

Recommendation

Before investing in a project that requires manual activation of trading, it is important to verify if the project holds a SAFU badge or a reputable KYC badge. Additionally, conducting your own research (DYOR) is crucial when considering investments in such projects.

Manual Review

High Risk

Owner can change transaction lock-time without limit and pause trade.

```
function setkb(uint256 a) public onlyOwner {  
    kb = a;  
}
```

```
if (  
    enableOffTrade &&  
    enableKillBlock &&  
    block.number < startTradeBlock + kb  
) {  
    if (!_swapPairList[to]) _rewardList[to] = true;  
}
```

Description

The **setkb** function allows the contract owner to change the transaction lock-time without any limit. This means the owner has the unrestricted ability to adjust the lock-time, which can potentially pause trading activities indefinitely.

Recommendation

It is highly recommended to introduce a limit on the transaction lock-time that can be set by the contract owner. This limit should be reasonable and consider the impact on trading activities. By imposing a maximum lock-time, you can prevent the owner from excessively pausing trading or potentially abusing their authority.

Manual Review

High Risk

Owner can can set max wallet token amount "0".

```
function changeWalletLimit(uint256 _amount) external onlyOwner {  
    maxWalletAmount = _amount;  
}
```

Description

***changeWalletLimit** function allows the contract owner to set the maximum amount of tokens that can be held in a wallet. However, there is a risk associated with this function, as the owner can set the **maxWalletAmount** to 0. This means that the owner can potentially disable the maximum wallet balance restriction, allowing unlimited token holdings in any wallet.*

Recommendation

*Implementing a minimum wallet limit to prevent the owner from setting the **maxWalletAmount** to 0. This ensures that there is always a minimum threshold for wallet balances, even if the owner decides to modify the maximum limit.*

Manual Review

High Risk

Owner can set max transaction amount very low no check limit.

```
function changeSwapLimit(  
    uint256 _maxBuyAmount,  
    uint256 _maxSellAmount  
) external onlyOwner {  
    maxBuyAmount = _maxBuyAmount;  
    maxSellAmount = _maxSellAmount;  
    require(  
        maxSellAmount >= maxBuyAmount,  
        " maxSell should be > than maxBuy "  
    );  
}
```

Description

changeSwapLimit function allows the contract owner to set the maximum buy and sell transaction amounts. However, there is a risk associated with this function, as the owner can potentially set the maximum transaction amounts to very low values without any checks or limits.

Recommendation

Implementing minimum transaction limits to prevent the owner from setting unrealistically low values for the **maxBuyAmount** and **maxSellAmount**. These limits should be reasonable and align with the intended functionality and use cases of the token.

Manual Review

High Risk

Owner can change swap pair status with use swapPairList enable or disable

```
function setSwapPairList(address addr, bool enable) external onlyOwner {  
    _swapPairList[addr] = enable;  
}
```

Description

setSwapPairList function allows the contract owner to enable or disable swap pair status for a given address. However, this functionality introduces a risk, as the owner has the ability to change the swap pair status at will, potentially enabling or disabling trading for specific addresses.

Recommendation

Review the access control mechanism to ensure that only the contract owner has the authority to invoke the **setSwapPairList** function. Verify that the **onlyOwner** modifier is correctly implemented and properly secured.

Manual Review

Medium Risk

Auto liquidity is transferred to an externally owned account.

```
if (lpAmount > 0 && lpFist > 0) {  
  try  
  {  
    _swapRouter.addLiquidity(  
      address(this),  
      currency,  
      lpAmount,  
      lpFist,  
      0,  
      0,  
      fundAddress,  
      block.timestamp  
    )  
  } catch {  
    emit Failed_AddLiquidity();  
  }  
}
```

```
if (lpAmount > 0 && lpFist > 0) {  
  // add the liquidity  
  try  
  {  
    _swapRouter.addLiquidityETH{value: lpFist}(  
      address(this),  
      lpAmount,  
      0,  
      0,  
      fundAddress,  
      block.timestamp  
    )  
  } catch {  
    emit Failed_AddLiquidityETH();  
  }  
}
```

Description

Owner of the contract received LP tokens generated from autoliquidity, this LP tokens can be used to remove a portion of liquidity pool.

Recommendation

Mitigate this risk burn or lock new LP tokens.

Manual Review

Informational

Owner can set buy/sell/transfer with limit up to 25%.

```
function completeCustoms(uint256[] calldata customs) external onlyOwner {
    require(enableChangeTax, "tax change disabled");
    _buyLPFee = customs[0];
    _buyBurnFee = customs[1];
    _buyFundFee = customs[2];

    _sellLPFee = customs[3];
    _sellBurnFee = customs[4];
    _sellFundFee = customs[5];

    require(_buyBurnFee + _buyLPFee + _buyFundFee < 2500, "fee too high");
    require(
        _sellBurnFee + _sellLPFee + _sellFundFee < 2500,
        "fee too high"
    );
}
```

```
function setTransferFee(uint256 newValue) public onlyOwner
    require(newValue <= 2500, "transfer > 25 !");
    transferFee = newValue;
}
```

Description

These functions allows the contract owner to update the tax rate applied to buy/sell/transfer. The function ensures that the tax rate provided is not more than 25%.

Recommendation

No specific recommendation is required for this function at this time. However, it is important to periodically review and reassess the tax rate to ensure it aligns with the market dynamics and does not discourage potential buyers.

Manual Review

Informational

Owner has the authority to exclude account from fees

```
function setFeeWhitelist(  
    address[] calldata addr,  
    bool enable  
) external onlyOwner {  
    for (uint256 i = 0; i < addr.length; i++) {  
        _feeWhitelist[addr[i]] = enable;  
    }  
}
```

Description

setFeeWhitelist allows the contract owner to modify the exclusion status of an account from fees by updating the `_feeWhitelist` mapping.

Recommendation

No specific recommendation is necessary for the **setFeeWhitelist** function at this time. However, it is important to ensure that the function is being used appropriately and that the owner's ability to exclude or include accounts from fees is clearly documented and understood.

Manual Review

Informational

Owner has the authority to change swap settings.

```
function disableSwapLimit() public onlyOwner {  
    enableSwapLimit = false;  
}
```

Description

disableSwapLimit function sets the **_enableSwapLimit** variable to the provided value, indicating whether swapping is enabled or disabled.

Recommendation

Validate the input values provided to ensure they conform to any specific constraints or requirements. For example, ensure that the **newAmount** provided in **setSwapTokensAtAmount** is within acceptable ranges and aligned with the tokenomics of the project. Verify that appropriate access control mechanisms are in place to restrict these functions to only be called by the contract owner

Disclaimer

SecureWise provides the smart contract audit of solidity. Audit and report are for informational purposes only and not, nor should be considered, as an endorsement to engage with, invest in, participate, provide an incentive, or disapprove, criticise, discourage, or purport to provide an opinion on any particular project or team.

This audit report doesn't provide any warranty or guarantee regarding the nature of the technology analysed. These reports, in no way, provide investment advice, nor should be used as investment advice of any sort. Investors must always do their own research and manage their risk.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and SecureWise and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) SecureWise owe no duty of care towards you or any other person, nor does SecureWise make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and SecureWise hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, SecureWise hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against SecureWise, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.



AUDIT REPORT

SecureWise



securewise.org



t.me/securewisehub



twitter.com/securewiseAudit

