RUGFREECOINS

Squid Grok Token

RugfreeCoins Verified on November 19th, 2023

Overview

✅ No mint function found, the owner cannot mint tokens after initial deployment.

✅ The owner can't set a max transaction limit

✅ The owner can't pause trading once it's enabled

❌ The owner must enable trade for the holders, if trading remains disabled, no one would be able to buy and sell.

✅ The owner can't change fees.

✅ The owner can't blacklist wallets.

✅ The owner can't set a max wallet limit

✅ The owner can't claim the contract's balance of its own token.

❗ HIGH SEVERITY ISSUES

The owner must enable trade for the holders, if trading remains disabled, no one would be able to buy and sell.

```
function enableTrading() external onlyOwner{
    require(tradingEnabled == false, "Trading is already enabled");
    tradingEnabled = true;
}
```

The owner can change the marketing wallet, and if the owner sets it to a contract that cannot receive BNB, both swaps and sells will fail since the marketing wallet receives BNB.

```
    function changeMarketingWallet(address _marketingWallet) external onlyOwner {
        require(_marketingWallet != marketingWallet, "Marketing wallet is already
that address");
        require(_marketingWallet!=address(0), "Marketing wallet is the zero
address");
        marketingWallet = _marketingWallet;
        emit MarketingWalletChanged(marketingWallet);
    }
```

The owner can change the swap limit without specifying a maximum amount. If the owner sets the swap amount to an excessively large value, swaps will fail when the contract attempts to swap a substantial amount of tokens at once.

```
    function setSwapTokensAtAmount(uint256 newAmount) external onlyOwner() {
        require(newAmount > totalSupply() / 1e5, "SwapTokensAtAmount must be greater
than 0.001% of total supply");
        swapTokensAtAmount = newAmount;
        emit SwapTokensAtAmountUpdated(newAmount);
    }

    function setSwapEnabled(bool _enabled) external onlyOwner {
        swapEnabled = _enabled;
        emit SwapEnabledUpdated(_enabled);
    }
```

Contents

Audit details

Audited project

Squid Grok Token

Contract Address

0xC1FC3380dD5CF1E55f70C0fc97f3e17db446c0F2

Client contact

Squid Grok Token Team

Blockchain

Binance Smart chain

Project website

https://squidgrok.online/

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – **please make sure to read it in full.**

ⓘ 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 **RugfreeCoins and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (RugfreeCoins) owe no duty of care towards you or any other person**, nor does RugfreeCoins 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 RugfreeCoins 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, RugfreeCoins hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against RugfreeCoins, 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.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Background

RugfreeCoins was commissioned by the **Squid Grok Token Team** to perform an audit of the smart contract.

https://bscscan.com/address/0xC1FC3380dD5CF1E55f70C0fc97f3e17db446c0F2

This audit focuses on verifying that the smart contract is secure, resilient, and working according to the specifications.

The information in this report should be used to understand the risk exposure of the smart contract, project feasibility, and long-term sustainability, and as a guide to improving the smart contract's security posture by remediating the identified issues.

Tokenomics

▲ 5% tax when buying & selling (19/11/2023)

5% of trade goes to the marketing wallet in BNB
0% trade is distributed among holders as rewards in tokens.
0% of trade goes to the Liquidity Pool

Target market and the concept

- Anyone who's interested in the Crypto space with long-term investment plans.
- Anyone who's ready to earn a passive income by holding tokens.
- Anyone who's interested in trading tokens.
- Anyone who's interested in taking part in the Squid Grok token ecosystem.
- Anyone who's interested in taking part in the future plans of Squid Grok Token.
- Anyone who's interested in making financial transactions with any other party using Squid Grok Token as the currency.

Potential to grow with score points

⚡ Project efficiency	8 / 10
🌟 Project uniqueness	8 / 10
📊 Information quality	8 / 10
👌 Service quality	8 / 10
💻 System quality	8 / 10
🌍 Impact on the community	8 / 10
💼 Impact on the business	9 / 10
🔮 Preparing for the future	8 / 10
🔒 Smart contract security	7 / 10
🛠️ Smart contract functionality assessment	7 / 10
🏆 **Total Score**	**7.9 / 10**

Contract details

Token contract details for 19th of November 2023

Contract name	**Squid Grok**
Contract address	**0xC1FC3380dD5CF1E55f70C0fc97f3e17db446c0F2**
Token supply	**1,000,000,000**
Token ticker	**Squid Grok**
Decimals	**9**
Token holders	**2**
Transaction count	**2**
Contract deployer address	**0x0Eb2F479ab635EDbB20b6B01165F935BC3436Ec1**
Contract’s current owner address	**0x0Eb2F479ab635EDbB20b6B01165F935BC3436Ec1**
Marketing wallet	**0x3Feec370Fa390B5d6D4C6BDC21b014d73206820e**

Contract code function details

№	Category	Item	Result
1	Coding conventions	ERC20 Token standards	PASS
		Compile errors	PASS
		Compiler version security	PASS
		Visibility specifiers	PASS
		Gas consumption	MEDIUM
		SafeMath features	PASS
		Fallback usage	PASS
		tx.origin usage	PASS
		Deprecated items	PASS
		Redundant code	MEDIUM
		Overriding variables	PASS
2	Function call audit	Authorization of function call	PASS
		Low level function (call/delegate call) security	PASS
		Returned value security	PASS
		Self destruct function security	PASS
3	Business security & centralisation	Access control of owners	HIGH
		Business logics	MEDIUM
		Business implementation	PASS
4	Integer overflow/underflow		PASS
5	Reentrancy		PASS
6	Exceptional reachable state		PASS
7	Transaction ordering dependence		PASS
8	Block properties dependence		PASS
9	Pseudo random number generator (PRNG)		PASS
10	DoS (Denial of Service)		PASS
11	Token vesting implementation		PASS
12	Fake deposit		PASS
13	Event security		PASS

Contract description table

The below table represents the summary of the contracts and methods in the token contract. We scanned the whole contract and listed down all the Interfaces, functions, and implementations with their visibility and mutability.

Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
Context	**Implementation**			
L	_msgSender	Internal 🔒		
L	_msgData	Internal 🔒		
Ownable	**Implementation**	**Context**		
L		Public ❗	🛑	NO ❗
L	owner	Public ❗		NO ❗
L	renounceOwnership	Public ❗	🛑	onlyOwner
L	transferOwnership	Public ❗	🛑	onlyOwner
IERC20	**Interface**			
L	totalSupply	External ❗		NO ❗
L	balanceOf	External ❗		NO ❗
L	transfer	External ❗	🛑	NO ❗
L	allowance	External ❗		NO ❗
L	approve	External ❗	🛑	NO ❗
L	transferFrom	External ❗	🛑	NO ❗
Address	**Library**			
L	isContract	Internal 🔒		
L	sendValue	Internal 🔒	🛑	

L	functionCall	Internal 🔒	🛑	
L	functionCall	Internal 🔒	🛑	
L	functionCallWithValue	Internal 🔒	🛑	
L	functionCallWithValue	Internal 🔒	🛑	
L	_functionCallWithValue	Private 🔐	🛑	
IUniswapV2Factory	**Interface**			
L	feeTo	External ❗		NO ❗
L	feeToSetter	External ❗		NO ❗
L	getPair	External ❗		NO ❗
L	allPairs	External ❗		NO ❗
L	allPairsLength	External ❗		NO ❗
L	createPair	External ❗	🛑	NO ❗
L	setFeeTo	External ❗	🛑	NO ❗
L	setFeeToSetter	External ❗	🛑	NO ❗
IUniswapV2Pair	**Interface**			
L	name	External ❗		NO ❗
L	symbol	External ❗		NO ❗
L	decimals	External ❗		NO ❗
L	totalSupply	External ❗		NO ❗
L	balanceOf	External ❗		NO ❗
L	allowance	External ❗		NO ❗
L	approve	External ❗	🛑	NO ❗
L	transfer	External ❗	🛑	NO ❗
L	transferFrom	External ❗	🛑	NO ❗
L	DOMAIN_SEPARATOR	External ❗		NO ❗
L	PERMIT_TYPEHASH	External ❗		NO ❗

L	nonces	External ❗		NO ❗
L	permit	External ❗	🛑	NO ❗
L	MINIMUM_LIQUIDITY	External ❗		NO ❗
L	factory	External ❗		NO ❗
L	token0	External ❗		NO ❗
L	token1	External ❗		NO ❗
L	getReserves	External ❗		NO ❗
L	price0CumulativeLast	External ❗		NO ❗
L	price1CumulativeLast	External ❗		NO ❗
L	kLast	External ❗		NO ❗
L	burn	External ❗	🛑	NO ❗
L	swap	External ❗	🛑	NO ❗
L	skim	External ❗	🛑	NO ❗
L	sync	External ❗	🛑	NO ❗
L	initialize	External ❗	🛑	NO ❗
IUniswapV2Router01	**Interface**			
L	factory	External ❗		NO ❗
L	WETH	External ❗		NO ❗
L	addLiquidity	External ❗	🛑	NO ❗
L	addLiquidityETH	External ❗	💵	NO ❗
L	removeLiquidity	External ❗	🛑	NO ❗
L	removeLiquidityETH	External ❗	🛑	NO ❗
L	removeLiquidityWithPermit	External ❗	🛑	NO ❗
L	removeLiquidityETHWithPermit	External ❗	🛑	NO ❗
L	swapExactTokensForTokens	External ❗	🛑	NO ❗
L	swapTokensForExactTokens	External ❗	🛑	NO ❗
L	swapExactETHForTokens	External ❗	💵	NO ❗
L	swapTokensForExactETH	External ❗	🛑	NO ❗

L	swapExactTokensForETH	External ❗	🛑	NO ❗
L	swapETHForExactTokens	External ❗	💵	NO ❗
L	quote	External ❗		NO ❗
L	getAmountOut	External ❗		NO ❗
L	getAmountIn	External ❗		NO ❗
L	getAmountsOut	External ❗		NO ❗
L	getAmountsIn	External ❗		NO ❗
IUniswapV2 Router02	**Interface**	**IUniswapV 2Router01**		
L	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
L	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
L	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO ❗
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
TOKEN	**Implementation**	**Context, IERC20, Ownable**		
L		Public ❗	🛑	NO ❗
L	name	Public ❗		NO ❗
L	symbol	Public ❗		NO ❗
L	decimals	Public ❗		NO ❗
L	totalSupply	Public ❗		NO ❗
L	balanceOf	Public ❗		NO ❗
L	transfer	Public ❗	🛑	NO ❗
L	allowance	Public ❗		NO ❗
L	approve	Public ❗	🛑	NO ❗
L	transferFrom	Public ❗	🛑	NO ❗

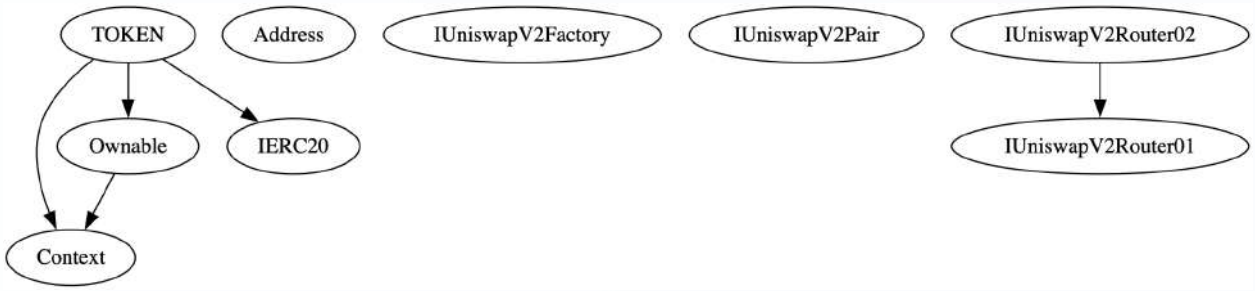
L	increaseAllowance	Public ❗	🛑	NO ❗
L	decreaseAllowance	Public ❗	🛑	NO ❗
L	isExcludedFromReward	Public ❗		NO ❗
L	totalReflectionDistributed	Public ❗		NO ❗
L	deliver	Public ❗	🛑	NO ❗
L	reflectionFromToken	Public ❗		NO ❗
L	tokenFromReflection	Public ❗		NO ❗
L	excludeFromReward	Public ❗	🛑	onlyOwner
L	includeInReward	External ❗	🛑	onlyOwner
L		External ❗	💵	NO ❗
L	claimStuckTokens	External ❗	🛑	onlyOwner
L	_reflectFee	Private 🔐	🛑	
L	_getValues	Private 🔐		
L	_getTValues	Private 🔐		
L	_getRValues	Private 🔐		
L	_getRate	Private 🔐		
L	_getCurrentSupply	Private 🔐		
L	_takeLiquidity	Private 🔐	🛑	
L	_takeMarketing	Private 🔐	🛑	
L	calculateTaxFee	Private 🔐		
L	calculateLiquidityFee	Private 🔐		
L	calculateMarketingFee	Private 🔐		
L	removeAllFee	Private 🔐	🛑	
L	setBuyFee	Private 🔐	🛑	
L	setSellFee	Private 🔐	🛑	
L	isExcludedFromFee	Public ❗		NO ❗
L	_approve	Private 🔐	🛑	
L	enableTrading	External ❗	🛑	onlyOwner
L	_transfer	Private 🔐	🛑	

L	swapAndLiquify	Private 🔐	🛑	
L	swapAndSendMarketing	Private 🔐	🛑	
L	setSwapTokensAtAmount	External ❗	🛑	onlyOwner
L	setSwapEnabled	External ❗	🛑	onlyOwner
L	_tokenTransfer	Private 🔐	🛑	
L	_transferStandard	Private 🔐	🛑	
L	_transferToExcluded	Private 🔐	🛑	
L	_transferFromExcluded	Private 🔐	🛑	
L	_transferBothExcluded	Private 🔐	🛑	
L	excludeFromFees	External ❗	🛑	onlyOwner
L	changeMarketingWallet	External ❗	🛑	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Inheritance Hierarchy



Security issue checking status

❖ High severity issues

The owner must enable trade for the holders, if trading remains disabled, no one would be able to buy and sell.

```
function enableTrading() external onlyOwner{
    require(tradingEnabled == false, "Trading is already enabled");
    tradingEnabled = true;
}
```

The owner can change the marketing wallet, and if the owner sets it to a contract that cannot receive BNB, both swaps and sells will fail since the marketing wallet receives BNB.

```
function changeMarketingWallet(address _marketingWallet) external onlyOwner {
    require(_marketingWallet != marketingWallet, "Marketing wallet is already
that address");
    require(_marketingWallet!=address(0), "Marketing wallet is the zero
address");
    marketingWallet = _marketingWallet;
    emit MarketingWalletChanged(marketingWallet);
}
```

The owner can change the swap limit without specifying a maximum amount. If the owner sets the swap amount to an excessively large value, swaps will fail when the contract attempts to swap a substantial amount of tokens at once.

```
    function setSwapTokensAtAmount(uint256 newAmount) external onlyOwner() {
        require(newAmount > totalSupply() / 1e5, "SwapTokensAtAmount must be greater
than 0.001% of total supply");
        swapTokensAtAmount = newAmount;
        emit SwapTokensAtAmountUpdated(newAmount);
    }

    function setSwapEnabled(bool _enabled) external onlyOwner {
        swapEnabled = _enabled;
        emit SwapEnabledUpdated(_enabled);
    }
```

❖ Medium severity issues

When transferring fees to the contract, it does not emit a transfer event. Consequently, the fees in the contract do not appear on BscScan.

```
    function _takeLiquidity(uint256 tLiquidity) private {
        if (tLiquidity > 0) {
            uint256 currentRate =  _getRate();
            uint256 rLiquidity = tLiquidity * currentRate;
            _rOwned[address(this)] = _rOwned[address(this)] + rLiquidity;
            if(_isExcluded[address(this)])
                _tOwned[address(this)] = _tOwned[address(this)] + tLiquidity;
        }
    }

    function _takeMarketing(uint256 tMarketing) private {
        if (tMarketing > 0) {
            uint256 currentRate =  _getRate();
            uint256 rMarketing = tMarketing * currentRate;
            _rOwned[address(this)] = _rOwned[address(this)] + rMarketing;
            if(_isExcluded[address(this)])
                _tOwned[address(this)] = _tOwned[address(this)] + tMarketing;
        }
    }
```

The reward fee and liquidity fees are default set to 0, and the owner cannot modify them subsequently. As a result, all functions associated with rewards and liquidity addition are redundant, constituting approximately 60% of the code.

```
taxFeeonBuy = 0;
taxFeeonSell = 0;

liquidityFeeonBuy = 0;
liquidityFeeonSell = 0;

marketingFeeonBuy = 5;
marketingFeeonSell = 5;
```

❖ Low severity issues

No low-severity issues found

Owner privileges

- Owner can include/exclude wallets from fees, this functions are redundant since contract never send rewards

```
function excludeFromReward(address account) public onlyOwner() {
    require(!_isExcluded[account], "Account is already excluded");
    if(_rOwned[account] > 0) {
        _tOwned[account] = tokenFromReflection(_rOwned[account]);
    }
    _isExcluded[account] = true;
    _excluded.push(account);
}

function includeInReward(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is already included");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

- Owner can get any bep20 tokens from the contract (can not withdraw native tokens)

```
function claimStuckTokens(address token) external onlyOwner {
    require(token != address(this), "Owner cannot claim native tokens");
    if (token == address(0x0)) {
        payable(msg.sender).sendValue(address(this).balance);
        return;
    }
    IERC20 ERC20token = IERC20(token);
    uint256 balance = ERC20token.balanceOf(address(this));
    ERC20token.transfer(msg.sender, balance);
}
```

❖ Owner can enable trading,once enabled can not disable again

```
function enableTrading() external onlyOwner{
    require(tradingEnabled == false, "Trading is already enabled");
    tradingEnabled = true;
}
```

❖ Owner can change the swap point and can enable/disable swapping

```
function setSwapTokensAtAmount(uint256 newAmount) external onlyOwner() {
    require(newAmount > totalSupply() / 1e5, "SwapTokensAtAmount must be greater
than 0.001% of total supply");
    swapTokensAtAmount = newAmount;
    emit SwapTokensAtAmountUpdated(newAmount);
}

function setSwapEnabled(bool _enabled) external onlyOwner {
    swapEnabled = _enabled;
    emit SwapEnabledUpdated(_enabled);
}
```

❖ Owner can include/exclude wallets from fees

```
function excludeFromFees(address account, bool excluded) external onlyOwner {
    require(_isExcludedFromFees[account] != excluded, "Account is already the
value of 'excluded'");
    _isExcludedFromFees[account] = excluded;

    emit ExcludeFromFees(account, excluded);
}
```

❖ Owner can change marketing wallet

```
    function changeMarketingWallet(address _marketingWallet) external onlyOwner {
        require(_marketingWallet != marketingWallet, "Marketing wallet is already
that address");
        require(_marketingWallet!=address(0), "Marketing wallet is the zero
address");
        marketingWallet = _marketingWallet;
        emit MarketingWalletChanged(marketingWallet);
    }
```

Audit conclusion

RugFreeCoins team has performed in-depth testing, line-by-line manual code review, and automated audit of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, manipulations, and hacks. According to the smart contract audit.

Smart contract functional Status:	PASS
Smart contract security Status:	HIGH ISSUES
Number of risk issues:	5
Solidity code functional issue level:	PASS
Number of owner privileges:	6
Centralization risk correlated to the active owner:	HIGH
Smart contract active ownership:	ACTIVE