



RugFreeCoins Audit



Sugar Land Token

Smart Contract Security Audit

January 12, 2022

Contents

Audit details	1
Disclaimer	2
Background	3
About the project	4
Target market and the concept	10
Potential to grow with score points	13
Total Points	13
Contract details	14
Token distribution	15
Contract code function details	16
Contract description table	17
Security issue checking status	23
Owner privileges	24
Audit conclusion	27

Audit details



Audited project

Sugar Land Token



Contract Address

0xcB2aDBCa6f15E9B3F1D98FcE57aC48a093F34fA9



Client contact

Sugar Land Team



Blockchain

Binance smart chain



Project website

<http://sugarlandcoin.com/>

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 Sugar Land Token to perform an audit of the smart contract.

<https://bscscan.com/address/0xcB2aDBCa6f15E9B3F1D98FcE57aC48a093F34fA9>

The focus of this audit is to verify 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, long term sustainability and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

About the project

Sugar Land is a token built on the Binance Smart Chain that is with an innovative investment use case the main purpose of which is to seek out constant revenue sources, and heading towards building even greater Community, DAO Governed ecosystem, a Multi Chain NFT marketplace and an eventual future Game + metaverse. Each transaction, purchase incurs 12% fee, and sale incurs a 15% fee.

Vision

The team behind Sugarland believes the 'cryptoverse' is capable of creating massive opportunity and driving socioeconomic change for anyone willing to put in the effort. The core team knows first-hand the power of a strong community, and is always building from a "community-first" perspective.

Sugarland's Web 3.0 ecosystem draws from familiar 'cryptoverse' touchpoints and technologies and aims to push the limits of what is possible through community managed decentralized and/or DAO-governed revenue generating tools and dApps (decentralized applications). .

Features

- ❖ The Sugarland Token rewards will be distributed among every holder proportional to how many tokens each individual holds in values of 3% when buying and selling.
- ❖ The sustainability fee of 5% when buying and selling for marketing and dev is what allows Sugar Land to hold the aforementioned promise. Tokens will be swapped into BNB and will be sent to a marketing wallet per transaction. This way, Sugar Land will have enough funds to promote the coin and spend for future development without selling tokens as the traditional way.
- ❖ The additional component included under the sustainability section is a liquidity fee of 4% when buying and selling, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

Tokenomics

12% fee when buying and selling

- ❖ 3% of trade goes to holders' pockets in \$Sugar tokens.
- ❖ 4% of trade goes to the liquidity pool.
- ❖ 5% of trade goes to the marketing and Dev wallets.

Pre-Launch Phase In progress

- ✓ Roadmap & Contract Revision For Sugarland V2
- ✓ Private Sale Raise - 200 BNB
- ✓ Launch Marketing With Various Telegram & Twitter Influencers
- ✓ AMA with Big Crypto Communities
- ✓ Audit + KYC report From Brewlabs
- ✓ Successful Presale Round 1 For 245 BNB
- ✓ Successful Presale Round 2 For 350 BNB
- ✓ PancakeSwap Listing & Liquidity Lock for 1 Year
- ✓ Apply for Certik Audit

Phase 1 - Q1 2022

- ✓ Apply CoinGecko, CoinMarketCap and Nomics Listing
- ✓ SugarFactory - All in One dApp
- ✓ Initial Marketing With Partners
- ✓ PR Articles Publishing in Popular Media Outlets
- ✓ Banner Ads on Popular Coin Listing Sites
- ✓ Reach 10,000 Holders
- ✓ Apply for Tier 2 CEX Listings
- ✓ First NFT Lottery Round
- ✓ First NFT Series - BEP721 NFTs Carrying Most Benefits In Sugarland Ecosystem
- ✓ SugarFactory - Second Milestone For NFT Minting & Staking Going Live
- ✓ SugarFactory - Third Milestone For Governance

Phase 2

- ✔ Bring Tech & Media Partners
- ✔ NFT Marketplace MVP Development
- ✔ Marketplace Building & Testing
- ✔ Marketplace Beta Release
- ✔ Growth Acceleration For Marketplace User Base
- ✔ Features Upgrade & Scaling
- ✔ Marketplace Revenue Share Integrated For CITIZEN Series NFT Holders

Phase 3 & Onwards

- After succesful implemention of Previous Phases Short Term Roadmap, we will work towards Virtual Real Estate
- ✔ Development & Metaverse Integration using Latest Layered Blockchain Solutions
- Team Expansion Phase For Sugarland's Metaverse including
- ✔ Business and Advertising Based Revenue for Real Estate Holders
 - ✔ Virtual Real Estate NFT Sales

The Team



Mia

Founder - CEO - Lead Dev



jBob

Blockchain Advisor



Nomi

Creatives



Leben

Social Media Manager



Giuliano Maria Lodi

Front End Developer



Tyler McWilliams

Full Stack
Developer



Target market and the concept

Target market

- ❖ Anyone who's interested in the Crypto space with long-term investment plans.
- ❖ Anyone who's ready to earn a passive income in BNB by holding tokens.
- ❖ Anyone who's interested in trading tokens.
- ❖ Anyone who is ready to hold and be eligible to win in the daily lottery
- ❖ Anyone who is ready to hold a large portion of tokens and be eligible to get a high chance of winning in the weekly lottery.
- ❖ Anyone who's interested in collecting NFTs or trading NFTs.
- ❖ Anyone who's interested in taking part with the future plans of the Kira Doge token.
- ❖ Anyone who's interested in making financial transactions with any other party using BNB or Kira doge as the currency.

Core concept

\$SUGAR was initially launched, and received a positive community response, in mid-2021 as a meme-based reflection token.

The majority of V1 “Sugarlanders” / \$SUGAR holders have stuck with the core development team and will receive V2 tokens at the launch of this contract. The majority of the community approved of and endorsed the migration from V1 to V2, a contract and branding upgrade that makes the \$SUGAR token, and overall Sugarland brand a more scalable and dynamic project.

Ask anyone who was a part of V1, the Sugarland team works hard, and most importantly, cares about the community. The Sugarland V2 contract, approach, and project roadmap are all reworked and finely tuned in preparation for what's to come. SUGAR is ready to take on the next phase of DeFi and plans to be a key asset and part of the wider cryptoverse's journey to manifest the ultimate Metaverse experience.

The Sugar Land reward system

3% of each transaction when buying and selling gets converted to tokens, and is split amongst all holders. Holders will be eligible to receive tokens in each transaction and rewards are proportional to how many tokens each individual holds.

Sustainable mechanism

The sustainability fee of 5% when buying and selling for marketing and dev is what allows Sugar Land to promote the token and use funds to further the development of the platform. Tokens will be swapped into BNB and will be sent to a marketing wallet per transaction. This way, Sugar Land will have access to the funds without selling tokens as the traditional way, which will enable them to consume funds without hurting the project.

The liquidity fee of 4% when buying and selling, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

The use case

Sugarland's NFT marketplace is an origin point for an eventual 'one-stop-shop' and comprehensive, cross-chain solution & decentralized ecosystem that will scale across multiple metaverse, play to earn, and web 3.0 environments. The solution is focused on both primary and secondary markets and incorporates DAO-inspired governance and community involvement to establish a community-run and collaboratively curated NFT epicenter.

Via in-house and third-party development partners Sugarland's rich UI and DeFi navigation system will make it easy and fun to explore, discover, and collect feature-packed, dynamic NFTs. Cross-chain and partner integrations will make it easy to find, buy, trade, and sell the latest NFT collections across GameFi, DeFi, and Metaverse environments.

In addition to third-party partners and external integrations, Sugarland may develop and deploy \$SUGAR GameFi / play to earn NFTs and game environments as well as an eventual goal of a decentralized Sugarland Metaverse.

- ❖ **Rich navigation system designed to Explore and Collect feature-packed NFTs, the platform will make it accessible to easily find Latest Collections for GameFi and Defi protocol-based NFTs.**
- ❖ **With the boom in the crypto space, an NFT marketplace solution is required to showcase & prioritize NFTs with various use cases too and let the enthusiasts find them easily.**

With a business development around the above-mentioned use-cases and product releases, a dedicated team will lead the marketing and adoption of the marketplace.

Future developments at Sugarland may include the development of a wide range GameFi NFTs, with \$SUGAR being the earned currency. An eventual metaverse for Sugarland is also in the future goals.

NFT Innovation

Governance

At launch, whitelisted 'Sugarlanders' will have the opportunity to acquire non-fungible governance tokens. These select and initially rare NFTs will ultimately give the holder future opportunities to access and contribute to the management of the DAO at the core of the project.

Earn & Yield

Staking NFTs will enable a higher yield on your \$SUGAR, more details coming soon.

Grow & Access

Access-based NFTs or NFT tickets can provide holders with unique and exclusive experiences online and in the metaverse.

Dynamic & Adaptable

NFT developments are occurring daily, and Sugarland is excited to be on the cutting edge of what NFTs might be able to facilitate down the line. From real-world and digital environment access tokens to fractional ownership, and input-dependent dynamic output, Sugarland will be poised to integrate, utilize, and facilitate the exchange of NFTs of all types.

Potential to grow with score points

1.	Project efficiency	9/10
2.	Project uniqueness	9/10
3	Information quality	10/10
4	Service quality	10/10
5	System quality	10/10
6	Impact on the community	9/10
7	Impact on the business	10/10
8	Preparing for the future	9/10
Total Points		9.625/10

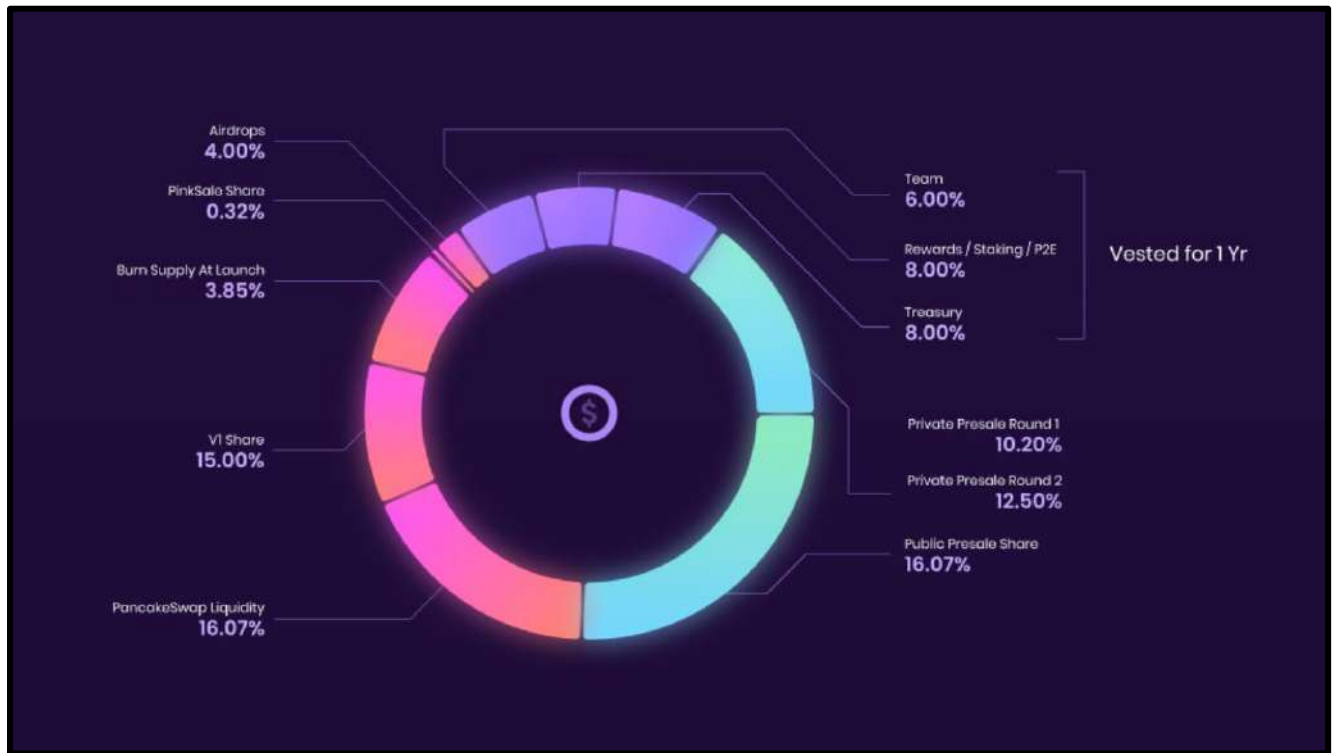
Contract details

Token contract details for 12th January 2022

Contract name	Sugarland
Contract address	0xcB2aDBCa6f15E9B3F1D98FcE57aC48a093F34fA9
Token supply	1,000,000,000
Token ticker	Sugar
Decimals	9
Token holders	2
Transaction count	2
Contract deployer address	0x3adE241eDED91fE0f10cDB93A7295c6469220b2b
Contract's current owner address	0x3ade241eded91fe0f10cdb93a7295c6469220b2b

Token distribution

Tokens are distributed as follows:











































Contract code function details








No	Category	Item	Result
1	Coding conventions	BRC20 Token standards	pass
		compile errors	low issue
		Compiler version security	pass
		visibility specifiers	pass
		Gas consumption	low issue
		SafeMath features	pass
		Fallback usage	pass
		tx.origin usage	pass
		deprecated items	pass
		Redundant code	pass
		Overriding variables	pass
2	Function call audit	Authorization of function call	pass
		Low level function (call/delegate call) security	pass
		Returned value security	pass
		Selfdestruct function security	pass
3	Business security	Access control of owners	pass
		Business logics	pass
		Business implementations	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

























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 its visibility and mutability.






Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
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	sendValue	Internal 		
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
L	_setOwner	Private 		
IFactory	Interface			
L	createPair	External 		NO 
IRouter	Interface			
L	factory	External 		NO 
L	WETH	External 		NO 
L	addLiquidityETH	External 		NO 



L	swapExactTokensForETHSupportingFeeOnTransferTokens	External ¶		NO¶
Sugarland	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	reflectionFromToken	Public ¶		NO¶

L	tokenFromReflection	Public 		NO 
L	excludeFromReward	Public 		onlyOwner
L	includeInReward	External 		onlyOwner
L	excludeFromFee	Public 		onlyOwner
L	includeInFee	Public 		onlyOwner
L	isExcludedFromFee	Public 		NO 
L	setTransferTaxes	Public 		onlyOwner
L	setBuyTaxes	Public 		onlyOwner
L	setSellTaxes	Public 		onlyOwner
L	_reflectRfi	Private 		
L	_takeLiquidity	Private 		
L	_takeMarketing	Private 		
L	_getValues	Private 		
L	_getTValues	Private 		
L	_getRValues	Private 		

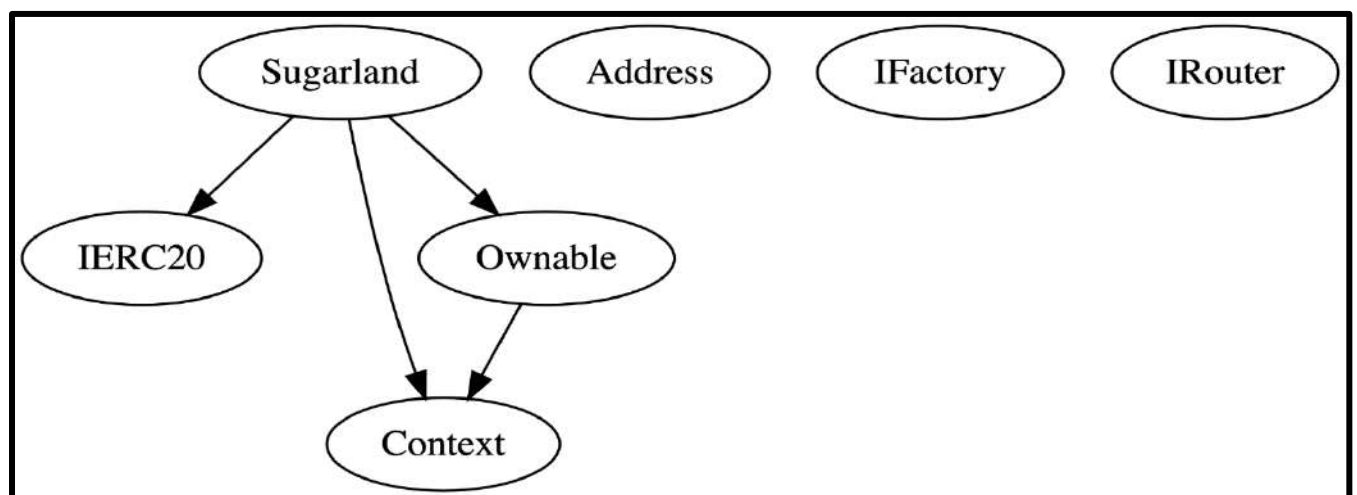
L	_getRate	Private 		
L	_getCurrentSupply	Private 		
L	_approve	Private 		
L	_transfer	Private 		
L	_tokenTransfer	Private 		
L	swapAndLiquify	Private 		lockTheSwap
L	addLiquidity	Private 		
L	swapTokensForBNB	Private 		
L	startTrading	External 		onlyOwner
L	updateMarketingWallet	External 		onlyOwner
L	updateMaxTxAmount	External 		onlyOwner
L	updateMaxWalletBalance	External 		onlyOwner
L	updateSwapTokensAtAmount	External 		onlyOwner

L	updateSwapEnabled	External !		onlyOwner
L	updateRouterAndPair	External !		onlyOwner
L	setIsBot	External !		onlyOwner
L	rescueBNB	External !		onlyOwner
L	rescueAnyBEP20Tokens	Public !		onlyOwner
L		External !		NO !

Legend

Symbol	Meaning
	Function can modify state
	Function is payable

Inheritance Hierarchy



Security issue checking status

❖ High severity issues

No high severity issues found.

❖ Medium severity issues

No medium severity issues found.

❖ Low severity issues

- The function `includeInReward()` uses the loop to find and remove addresses from the `_excluded` list. Function will be aborted with `OUT_OF_GAS` exception if there will be a long excluded addresses list.

```
ftrace | funcSig
function includeInReward(address account↑) external onlyOwner {
    require(!_isExcluded[account↑], "Account is not excluded");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account↑) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _owned[account↑] = 0;
            _isExcluded[account↑] = false;
            _excluded.pop();
            break;
        }
    }
}
```

❖ Informational

- The owner can change max transactions amount without any limit.

```
ftrace | funcSig
function updateMaxTxAmount(uint256 amount↑) external onlyOwner {
    maxTxAmount = amount↑ * 10**_decimals;
}

ftrace | funcSig
function updateMaxWalletBalance(uint256 amount↑) external onlyOwner {
    maxWalletAmount = amount↑ * 10**_decimals;
}
```

Owner privileges

- ❖ The owner can include/exclude wallets from rewards.

```
ftrace | funcSig
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↑);
}

ftrace | funcSig
function includeInReward(address account↑) external onlyOwner {
    require(!_isExcluded[account↑], "Account is not excluded");
    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;
        }
    }
}
```

- ❖ The owner can include and exclude wallets from fees.

```
ftrace | funcSig
function excludeFromFee(address account↑) public onlyOwner {
    _isExcludedFromFee[account↑] = true;
}

ftrace | funcSig
function includeInFee(address account↑) public onlyOwner {
    _isExcludedFromFee[account↑] = false;
}
```

- ❖ The owner can change all fees.

```
ftrace | funcSig
function setTransferTaxes(
    uint256 _rfi↑,
    uint256 _marketing↑,
    uint256 _liquidity↑
) public onlyOwner {
    transferTaxes = Taxes(_rfi↑, _marketing↑, _liquidity↑);
}

ftrace | funcSig
function setBuyTaxes(
    uint256 _rfi↑,
    uint256 _marketing↑,
    uint256 _liquidity↑
) public onlyOwner {
    buyTaxes = Taxes(_rfi↑, _marketing↑, _liquidity↑);
}

ftrace | funcSig
function setSellTaxes(
    uint256 _rfi↑,
    uint256 _marketing↑,
    uint256 _liquidity↑
) public onlyOwner {
    sellTaxes = Taxes(_rfi↑, _marketing↑, _liquidity↑);
}
```

- ❖ The owner can start trading.

```
ftrace | funcSig
function startTrading() external onlyOwner {
    require(!tradingActive, "Trading already active");
    tradingActive = true;
    swapEnabled = true;
}
```

- ❖ The owner can update the marketing wallet.

```
ftrace | funcSig
function updateMarketingWallet(address _marketingWallet↑)
    external
    onlyOwner
{
    marketingWallet = _marketingWallet↑;
}
```

- ❖ The owner can change max wallet balance and max transaction amount.

```
ftrace | funcSig
function updateMaxTxAmount(uint256 amount↑) external onlyOwner {
    maxTxAmount = amount↑ * 10**_decimals;
}

ftrace | funcSig
function updateMaxWalletBalance(uint256 amount↑) external onlyOwner {
    maxWalletAmount = amount↑ * 10**_decimals;
}
```

- ❖ The owner can enable/disable swapping and can change swap point.

```
ftrace | funcSig
function updateSwapTokensAtAmount(uint256 amount↑) external onlyOwner {
    swapTokensAtAmount = amount↑ * 10**_decimals;
}

ftrace | funcSig
function updateSwapEnabled(bool _enabled↑) external onlyOwner {
    swapEnabled = _enabled↑;
}
```

- ❖ The owner can update the router address.

```
ftrace | funcSig
function updateRouterAndPair(address newRouter↑, address newPair↑)
    external
    onlyOwner
{
    router = IRouter(newRouter↑);
    pair = newPair↑;
}
```


- ❖ The owner can blacklist wallets from the contract.

```
ftrace | funcSig
function setIsBot(address user↑, bool state↑) external onlyOwner {
    isBot[user↑] = state↑;
}

//Use this in case BNB are sent to the contract by mistake
```

- ❖ The owner can get BNBs and other tokens in contract to the owner wallet.

```
//Use this in case BNB are sent to the contract by mistake
ftrace | funcSig
function rescueBNB(uint256 weiAmount↑) external onlyOwner {
    require(address(this).balance >= weiAmount↑, "insufficient BNB balance");
    payable(msg.sender).transfer(weiAmount↑);
}

// Function to allow admin to claim *other* BEP20 tokens sent to this contract (by mistake)
ftrace | funcSig
function rescueAnyBEP20Tokens(
    address _tokenAddr↑,
    address _to↑,
    uint256 _amount↑
) public onlyOwner {
    IERC20(_tokenAddr↑).transfer(_to↑, _amount↑);
}
```

Audit conclusion

While conducting the audit of the Sugar Land Token smart contract, it was observed that there is nothing alarming with the code and it only contains a low severity issue.