RugFreeCoins Audit

Cardanomics Token

Smart Contract Security Audit

October 28, 2021

Contents

Audit details

Audited project

Cardanomics Token

Contract Address

0x9068bbcdd5a9e9f545539ce9953778967e18d5a4

Client contact

Cardanomics Team

Blockchain

Binance smart chain

Project website

https://www.cardanomics.org/

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 Cardanomics Token to perform an audit of the smart contract.

https://bscscan.com/token/0x9068bbcdd5a9e9f545539ce9953778967e18d5a4

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

Cardanomics is a token built on the Binance Smart Chain that combines the two most successful tokenomics on the Smart Chain: Rebase and Cardano Rewards. Every hour, Cardano rewards are distributed automatically to all holders, the supply diminishes, and the price per token increases. Each transaction, purchase and sale incur 15% fee.

Features

- **Rebase mechanism:** With an elastic supply, the Cardanomics token guarantees a forever growing chart, as the price per token is constantly increasing. Every hour, the supply is reduced and converges towards a reasonable, stable value.

- The **Caradano rewards** will be distributed among every holder proportional to how many tokens each individual holds in values of **5% when buying and selling**.

- The **sustainability fee of 4% for Dev and 2% for marketing when buying and selling** is what allows UpCake to hold the aforementioned promise. Tokens will be swapped into BNB and will be sent to a marketing wallet per transaction. This way, Cardanomics 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 2% from buying and selling,** which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

- Cardanomics improves on the popular buyback protocol with a new suite of innovations that will help increase returns for investors, which will be exchanging 2% of the tax for Bnb buys back from the supply every minute and burn all tokens bought automatically.

- Anti-bot and Anti-whale measures to keep the market stable.

Tokenomics

15% fee when buying and selling

- 5% of trade goes to holders' pockets in ADA tokens.
- 4% of trade goes to the marketing wallet in BNB.
- 2% of trade goes to the liquidity pool.
- 2% of trade goes to buyback and burn function.
- 2% of trade goes to the dev wallet in BNB.

Roadmap

Phase 1

- Creation of Socials and Website
- Hiring BlockChain Pros, Experts in Crypto Marketing
- Creation of Community and Organic Growth
- Contest to Participate in Exclusive Presale and Anti-Bots Measures

Phase 2

- Audit by RugFreeCoin
- Presale and Liquidity Lock (secured by PinkSale)
- PCS Launch
- Listing on CoinGecko
- 5,000 holders

Phase 3

- TrustWallet Logo
- Twitter and Telegram Marketing Campaign
- Shilling Contest
- Referral Contest
- Listing on CoinMarketCap
- 10,000 holders

Target market and the concept

Target market

- Anyone who's interested in Crypto space with long term investment plans.
- Anyone who's ready to earn a passive income in Cardano by holding tokens.
- Anyone who believes in rebase mechanisms to hold tokens for a longer duration with profits.
- Anyone who's interested in trading tokens.
- All ADA investors and fans out there.
- Anyone who's interested in taking part with the future plans of the Cardanomics token.
- Anyone who's interested in making financial transactions with any other party using ADA or Cardanomics as the currency.

Core concept

Rebase mechanism

Rebase is the newest tokenomics in BSC, which will revolutionize crypto space with an elastic supply. This guarantees a growing chart as the price per token constantly increases. There are 3 variables that influence each other to quantify the size of a token.

- Price per token
- Total supply
- Amount of BNBs in the liquidity pool

By decreasing the total supply and keeping the amount of BNBs in the LP constant, price per token mechanically increases.

The Cardanomics reward system

5% of each transaction when buying and selling gets converted to Cardano, and is split amongst all holders. Holders will be eligible to receive tokens every 3 hours and rewards are proportional to how many tokens each individual holds.

Sustainable mechanism

The **sustainability fee of 4% for marketing and 2% for development** is what allows Cardanomics to promote the token and use funds to further development of the platform. Tokens will be swapped into BNB and will be sent to a marketing wallet per transaction. This way, Cardanomics 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 2%, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

The buyback and burn mechanism collects 2% tax on each transaction, which is stored inside the contract. Whenever a buy or sell occurs, a fraction of the buyback amount is used to automatically purchase tokens from the liquidity pool. Those tokens are immediately burned after purchase, which keeps the token price stable.

Potential to grow with score points

1.	Project efficiency	9/10
2.	Project uniqueness	8/10
3	Information quality	10/10
4	Service quality	9/10
5	System quality	8/10
6	Impact on the community	8/10
7	Impact on the business	9/10
8	Preparing for the future	7/10
Total Points		**8.5/10**

Contract details

Token contract details for 28th October 2021

Contract name	Cardanomics
Contract address	0x9068bbcdd5a9e9f545539ce9953778967e18d5a4
Token supply	1,000,000,000,000,000
Token ticker	ADX
Decimals	8
Token holders	4
Transaction count	4
Auto liquidity receiver	0x59b3e0bc25d365dabd195de9d1f9cbf070863420
Contract deployer address	0x59B3e0bC25D365dabd195dE9d1f9CBF070863420
Contract's current owner address	0x59b3e0bc25d365dabd195de9d1f9cbf070863420

Tokens are distributed as follows:

- Dev wallet 3%
- Private sale 4%
- Presale & locked LP 35%
- Burn 58%

Contract code function details

No	Category	Item	Result
1	Coding conventions	BRC20 Token standards	pass
		compile errors	pass
		Compiler version security	pass
		visibility specifiers	pass
		Gas consumption	pass
		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	informational
		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		
└	Function Name	Visibility	Mutability	Modifiers
SafeMath	**Library**			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
SafeMathInt	**Library**			
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	sub	Internal 🔒		
└	add	Internal 🔒		

L	abs	Internal 🔒		
IBEP20	**Interface**			
L	totalSupply	External ❗		NO❗
L	decimals	External ❗		NO❗
L	symbol	External ❗		NO❗
L	name	External ❗		NO❗
L	getOwner	External ❗		NO❗
L	balanceOf	External ❗		NO❗
L	transfer	External ❗	🛑	NO❗
L	allowance	External ❗		NO❗
L	approve	External ❗	🛑	NO❗
L	transferFrom	External ❗	🛑	NO❗
Auth	**Implementation**			
L		Public ❗	🛑	NO❗
L	authorize	Public ❗	🛑	onlyOwner
L	unauthorize	Public ❗	🛑	onlyOwner
L	isOwner	Public ❗		NO❗

L	isAuthorized	Public ❗		NO❗
L	transferOwnership	Public ❗	🅾	onlyOwner
IDEXFactory	**Interface**			
L	createPair	External ❗	🅾	NO❗
InterfaceLP	**Interface**			
L	sync	External ❗	🅾	NO❗
IDEXRouter	**Interface**			
L	factory	External ❗		NO❗
L	WETH	External ❗		NO❗
L	addLiquidity	External ❗	🅾	NO❗
L	addLiquidityETH	External ❗	💰	NO❗
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🅾	NO❗
L	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💰	NO❗
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🅾	NO❗

IDividendDistributor	Interface			
L	setDistributionCriteria	External		NO!
L	setShare	External		NO!
L	deposit	External		NO!
L	process	External		NO!
DividendDistributor	**Implementation**	**IDividendDistributor**		
L		Public		NO!
L	setDistributionCriteria	External		onlyToken
L	setShare	External		onlyToken
L	deposit	External		onlyToken
L	process	External		onlyToken
L	shouldDistribute	Internal		
L	distributeDividend	Internal		
L	claimDividend	External		NO!
L	getUnpaidEarnings	Public		NO!
L	getCumulativeDividends	Internal		
L	addShareholder	Internal		
L	removeShareholder	Internal		

Cardanomics	**Implementation**	**IBEP20, Auth**		
L	rebase_percentage_master	Public !	◉	onlyMaster
L	rebase_percentage_owner	Public !	◉	onlyOwner
L	rebase	Public !	◉	onlyOwner
L		Public !	◉	Auth
L		External !	⚠	NO!
L	totalSupply	External !		NO!
L	decimals	External !		NO!
L	symbol	External !		NO!
L	name	External !		NO!
L	getOwner	External !		NO!
L	balanceOf	Public !		NO!
L	allowance	External !		NO!
L	approve	Public !	◉	NO!
L	approveMax	External !	◉	NO!
L	transfer	External !	◉	NO!
L	transferFrom	External !	◉	NO!

L	_transferFrom	Internal 🔒	●	
L	_basicTransfer	Internal 🔒	●	
L	checkTxLimit	Internal 🔒		
L	shouldTakeFee	Internal 🔒		
L	takeFee	Internal 🔒	●	
L	shouldSwapBack	Internal 🔒		
L	clearStuckBalance	External ❗	●	authorized
L	clearStuckBalance_sender	External ❗	●	authorized
L	set_sell_multiplier	External ❗	●	onlyOwner
L	tradingStatus	Public ❗	●	onlyOwner
L	launchStatus	Public ❗	●	onlyOwner
L	enable_blacklist	Public ❗	●	onlyOwner
L	manage_blacklist	Public ❗	●	onlyOwner
L	cooldownEnabled	Public ❗	●	onlyOwner
L	swapBack	Internal 🔒	●	swapping
L	setIsDividendExempt	External ❗	●	authorized
L	setIsFeeExempt	External ❗	●	authorized
L	setIsTxLimitExempt	External ❗	●	authorized

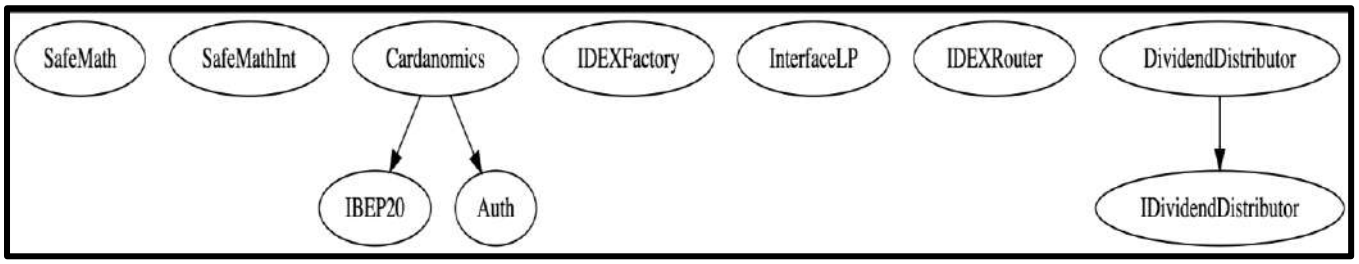
L	setIsTimelockExempt	External !	●	authorized
L	setFees	External !	●	authorized
L	setFeeReceivers	External !	●	authorized
L	setSwapBackSettings	External !	●	authorized
L	setTargetLiquidity	External !	●	authorized
L	manualSync	External !	●	NO!
L	setLP	External !	●	onlyOwner
L	setMaster	External !	●	onlyOwner
L	isNotInSwap	External !		NO!
L	checkSwapThreshold	External !		NO!
L	setDistributionCriteria	External !	●	authorized
L	setDistributorSettings	External !	●	authorized
L	rescueToken	Public !	●	onlyOwner
L	getCirculatingSupply	Public !		NO!
L	getLiquidityBacking	Public !		NO!
L	isOverLiquified	Public !		NO!
L	checkMaxWalletToken	External !		NO!
L	checkMaxTxAmount	External !		NO!

L	setMaxWalletPercent_base1000	External ❗	●	onlyOwner
L	setMaxTxPercent_base1000	External ❗	●	onlyOwner
L	multiTransfer	External ❗	●	onlyOwner
L	multiTransfer_fixed	External ❗	●	onlyOwner

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**
  - **No low severity issues found**
- **Informational**
  - **The owner can change the sell fee multiplier without any limit.**

```
ftrace | funcSig
function set_sell_multiplier(uint256 Multiplier) external onlyOwner {
    sellMultiplier = Multiplier;
}
```

  - **The owner can enable and disable trading any time**

```
ftrace | funcSig
function tradingStatus(bool _status, uint256 _deadBlocks) public onlyOwner {
    tradingOpen = _status;
    if (tradingOpen && launchedAt == 0) {
        launchedAt = block.number;
        deadBlocks = _deadBlocks;
    }
}
```

Owner privileges

- The owner can add and remove authorized wallets.

```
ftrace | funcSig
function authorize(address adr) public onlyOwner {
    authorizations[adr] = true;
}

ftrace | funcSig
function unauthorize(address adr) public onlyOwner {
    authorizations[adr] = false;
}
```

- The owner can transfer ownership.

```
ftrace | funcSig
function transferOwnership(address payable adr) public onlyOwner {
    owner = adr;
    authorizations[adr] = true;
    emit OwnershipTransferred(adr);
}
```

- The owner can add or reduce total supply maximum 20%.

```
ftrace | funcSig
function rebase_percentage_master(uint256 _percentage_base1000, bool reduce)
    public
    onlyMaster
    returns (uint256 newSupply)
{
    require(_percentage_base1000 < 201, "Cant rebase more than 20%");
    if (reduce) {
        newSupply = rebase(
            0,
            int256(_totalSupply.div(1000).mul(_percentage_base1000)).mul(-1)
        );
    } else {
        newSupply = rebase(
            0,
            int256(_totalSupply.div(1000).mul(_percentage_base1000))
        );
    }
}
```

- The owner can transfer contract bnb balance to marketing wallet.

```
ftrace | funcSig
function clearStuckBalance(uint256 amountPercentage) external authorized {
    uint256 amountBNB = address(this).balance;
    payable(marketingFeeReceiver).transfer(
        (amountBNB * amountPercentage) / 100
    );
}
```

- Thew owner can transfer contract bnb balance to his wallet.

```
ftrace | funcSig
function clearStuckBalance_sender(uint256 amountPercentage)
    external
    authorized
{
    uint256 amountBNB = address(this).balance;
    payable(msg.sender).transfer((amountBNB * amountPercentage) / 100);
}
```

- The owner can change sell fee multiplier.

```
ftrace | funcSig
function set_sell_multiplier(uint256 Multiplier) external onlyOwner {
    sellMultiplier = Multiplier;
}
```

- The owner can enable/disable trading.

```
ftrace | funcSig
function tradingStatus(bool _status, uint256 _deadBlocks) public onlyOwner {
    tradingOpen = _status;
    if (tradingOpen && launchedAt == 0) {
        launchedAt = block.number;
        deadBlocks = _deadBlocks;
    }
}
```

- The owner can change launched time.

```
ftrace | funcSig
function launchStatus(uint256 _launchblock) public onlyOwner {
    launchedAt = _launchblock;
}
```

❖ The owner can enable blacklist mode and add or remove wallets from blacklist.

```
ftrace | funcSig
function enable_blacklist(bool _status) public onlyOwner {
    blacklistMode = _status;
}

ftrace | funcSig
function manage_blacklist(address[] calldata addresses, bool status)
    public
    onlyOwner
{
    for (uint256 i; i < addresses.length; ++i) {
        isBlacklisted[addresses[i]] = status;
    }
}
```

❖ The owner can enable buy cool down.

```
ftrace | funcSig
function cooldownEnabled(bool _status, uint8 _interval) public onlyOwner {
    buyCooldownEnabled = _status;
    cooldownTimerInterval = _interval;
}

ftrace | funcSig
```

❖ The owner can change swap back and target liquidity settings.

```
ftrace | funcSig
function setSwapBackSettings(bool _enabled, uint256 _percentage_base100000)
    external
    authorized
{
    swapEnabled = _enabled;
    swapThreshold = rSupply.div(100000).mul(_percentage_base100000);
}

ftrace | funcSig
function setTargetLiquidity(uint256 _target, uint256 _denominator)
    external
    authorized
{
    targetLiquidity = _target;
    targetLiquidityDenominator = _denominator;
}
```

- The owner can add or remove wallets from dividend exempt, fee exempt, max transaction exempt and time lock exempt.

```
ftrace | funcSig
function setIsDividendExempt(address holder, bool exempt)
    external
    authorized
{
    require(holder != address(this) && holder != pair);
    isDividendExempt[holder] = exempt;
    if (exempt) {
        distributor.setShare(holder, 0);
    } else {
        distributor.setShare(holder, balanceOf(holder));
    }
}

ftrace | funcSig
function setIsFeeExempt(address holder, bool exempt) external authorized {
    isFeeExempt[holder] = exempt;
}

ftrace | funcSig
function setIsTxLimitExempt(address holder, bool exempt)
    external
    authorized
{
    isTxLimitExempt[holder] = exempt;
}

ftrace | funcSig
function setIsTimelockExempt(address holder, bool exempt)
    external
    authorized
{
    isTimelockExempt[holder] = exempt;
}
```

❖ The owner can change all fees and fee receivers.

```
ftrace | funcSig
function setFees(
    uint256 _liquidityFee,
    uint256 _reflectionFee,
    uint256 _marketingFee,
    uint256 _devFee,
    uint256 _feeDenominator
) external authorized {
    liquidityFee = _liquidityFee;
    reflectionFee = _reflectionFee;
    marketingFee = _marketingFee;
    devFee = _devFee;
    totalFee = _liquidityFee.add(_reflectionFee).add(_marketingFee).add(
        _devFee
    );
    feeDenominator = _feeDenominator;
    require(totalFee < feeDenominator / 3, "Fees cannot be more than 33%");
}

ftrace | funcSig
function setFeeReceivers(
    address _autoLiquidityReceiver,
    address _marketingFeeReceiver,
    address _devFeeReceiver
) external authorized {
    autoLiquidityReceiver = _autoLiquidityReceiver;
    marketingFeeReceiver = _marketingFeeReceiver;
    devFeeReceiver = _devFeeReceiver;
}
```

❖ The owner can change contract lp address.

```
ftrace | funcSig
function setLP(address _address) external onlyOwner {
    pairContract = InterfaceLP(_address);
    isFeeExempt[_address];
}
```

❖ The owner can change distributor gas fee.

```
ftrace | funcSig
function setDistributorSettings(uint256 gas) external authorized {
    require(gas < 900000);
    distributorGas = gas;
}
```

- The owner can get all other tokens in contract to owner wallet.

```
ftrace | funcSig
function rescueToken(address tokenAddress, uint256 tokens)
    public
    onlyOwner
    returns (bool success)
{
    return IBEP20(tokenAddress).transfer(msg.sender, tokens);
}
```

- The owner can change max wallet token and transaction amount.

```
ftrace | funcSig
function setMaxWalletPercent_base1000(uint256 maxWallPercent_base1000)
    external
    onlyOwner
{
    _maxWalletToken = rSupply.div(1000).mul(maxWallPercent_base1000);
}

ftrace | funcSig
function setMaxTxPercent_base1000(uint256 maxTXPercentage_base1000)
    external
    onlyOwner
{
    _maxTxAmount = rSupply.div(1000).mul(maxTXPercentage_base1000);
}
```

Audit conclusion

While conducting the audit of the Cardanomics smart contract, it was observed that there is nothing alarming with the code and it only contains informational concerns.