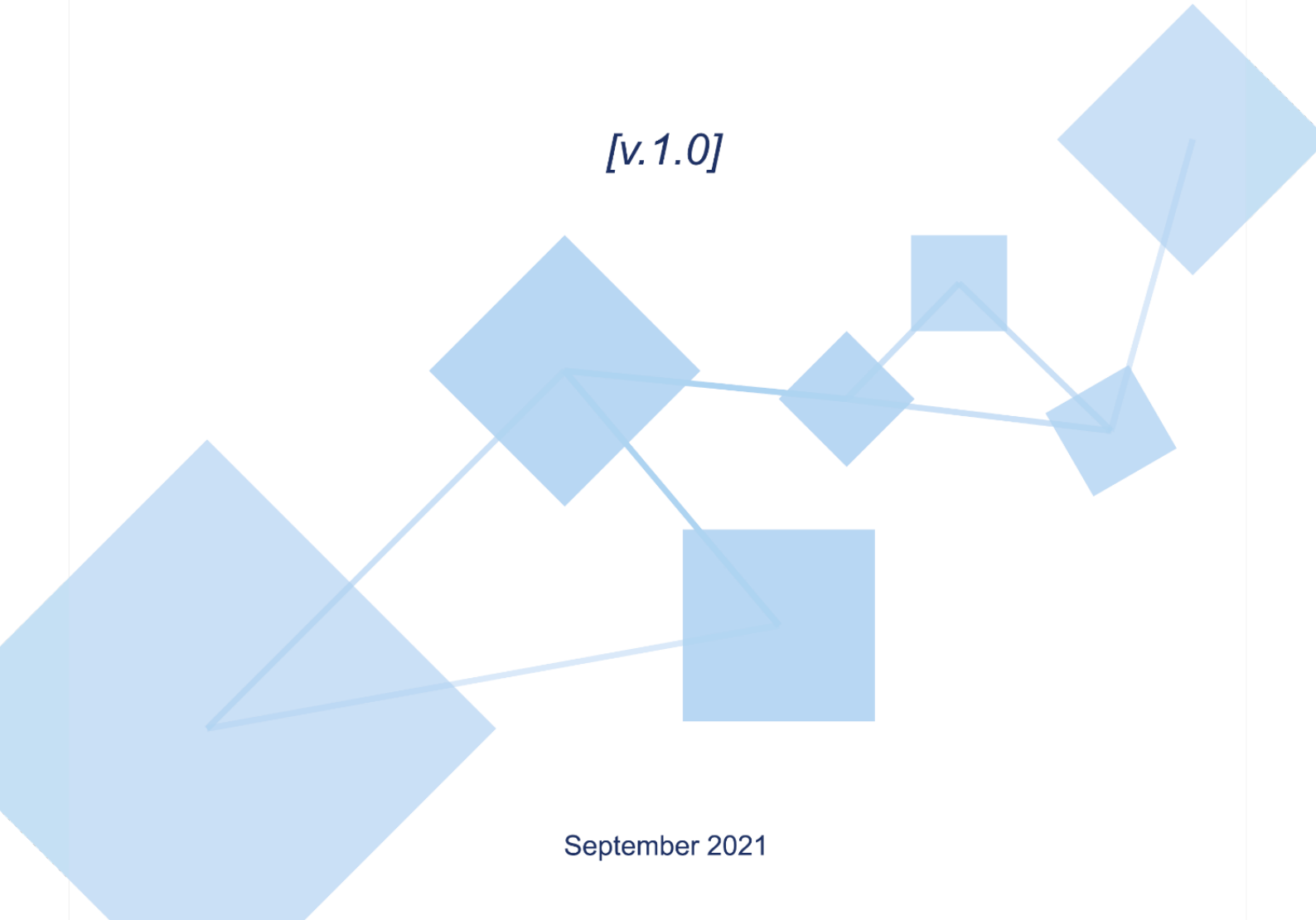
CheckPoint

Token Security Audit Report
Prepared for McCardanos

[v.1.0]



September 2021

Document Properties

Client	McCardanos
Platform	Binance Smart Chain
Language	Solidity
Codebase	0xd53E008A5e3eBd82FC65E6562F1F4c3dF56667a7

Audit Summary

Delivery Date	06.09.2021
Audit Methodology	Static Analysis, Manual Review
Auditor(s)	Erno Patiala
Classification	Publlic
Version	1.0

Contact Information

Company	CheckPoint
Name	Hanna Järvinen
Telegram	t.me/checkpointreport
E-mail	contact@checkpoint.report

Remark: For more information about this document and its contents, please contact CheckPoint team

Table Of Contents

1 Executive Summary

On 06/09/2021, CheckPoint conducted a full audit for the McCardanos to verify the overall security posture including a smart contract review to discover issues and vulnerabilities in the source code. Static Code Analysis, Dynamic Analysis, and Manual Review werdone in conjunction to identify smart contract vulnerabilities together with technical & business logic flaws that may be exposed to the potential risk of the platform and the ecosystem.

After further analysis and internal discussion, we determined a few issues of varying severities that need to be brought up and paid more attention to. More information can be found in **Section 5 'Audit Result'**. Practical recommendations are provided according to each vulnerability found and should be followed to remediate the issue.

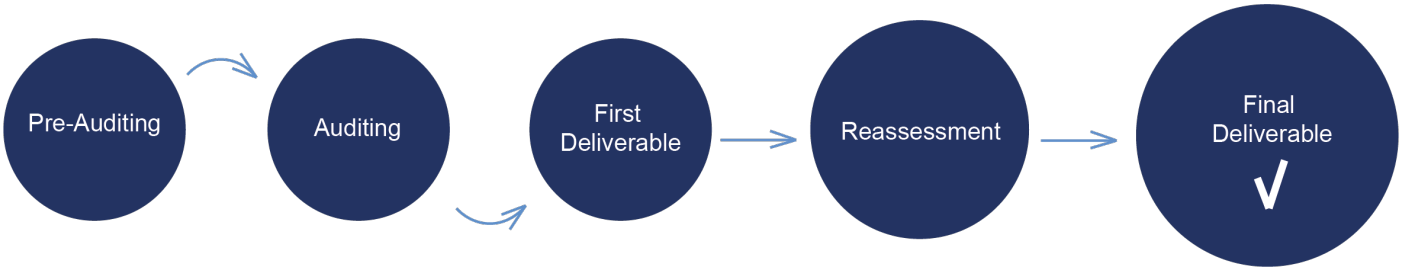
McCardanos

Low Risk Level

Communication Channels	Website Content Analysis, Social Media Listening
Smart Contract Code	Smart Contract Details, Contract Function Details, Issues Checking Status, Detailed Findings Information

THIS TOKEN PASSES CHECKPOINT'S SECURITY VERIFICATION STANDART

2 Audit Methodology



CheckPoint conducts the following procedure to enhance the security level of our clients' tokens:

- **Pre-Auditing**

  Planning a comprehensive survey of the token, its ecosystem, possible risks & prospects, getting to understand the overall operations of the related smart contracts, checking for readiness, and preparing for the auditing.

- **Auditing**

  Study of all available information about the token on the Web, inspecting the smart contracts using automated analysis tools and manual analysis by a team of professionals.

- **First Deliverable and Consulting**

  Delivering a preliminary report on the findings with suggestions on how to remediate those issues and providing consultation.

- **Reassessment**

  Verifying the status of the issues and whether there are any other complications in the fixes applied.

- **Final Deliverable**

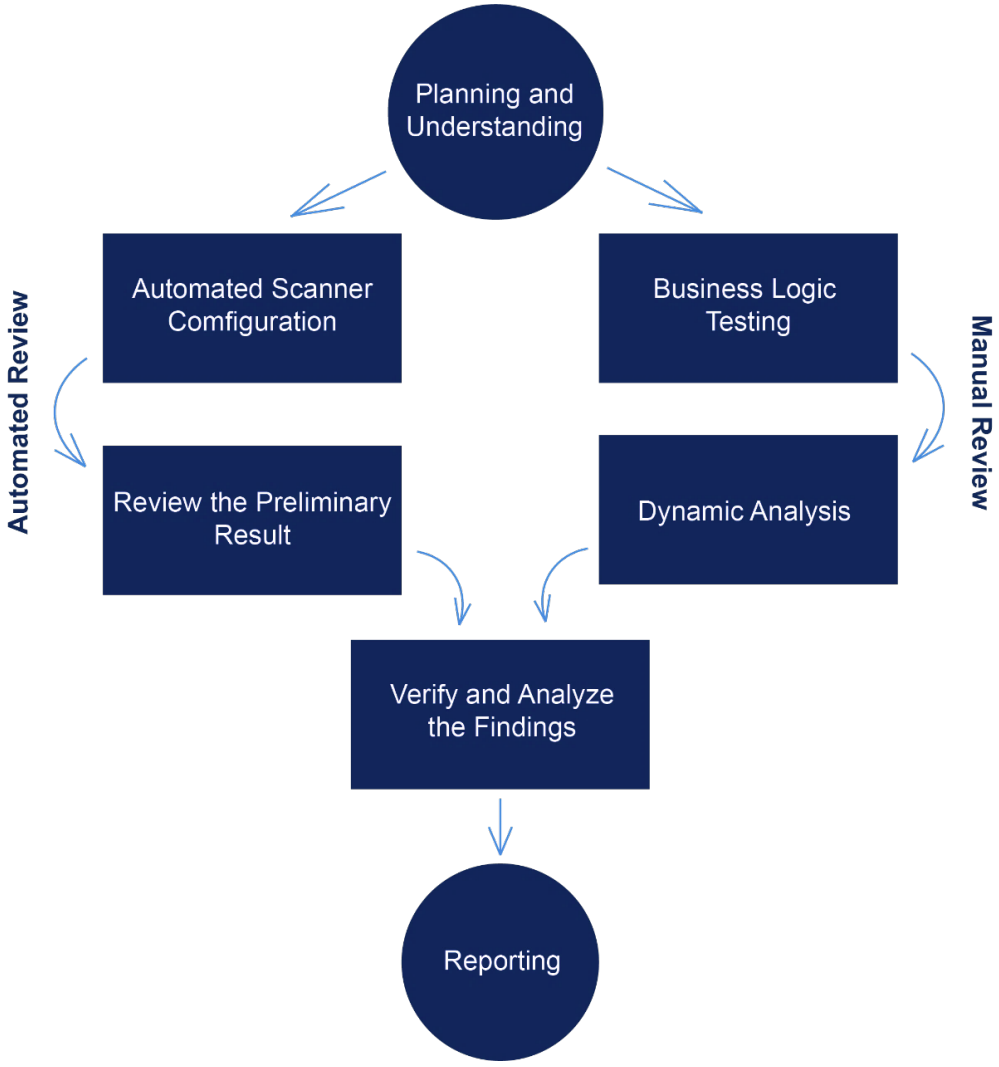
  Providing a full report with the detailed status of each issue.

The security audit process of CheckPoint includes three types testing:

1. Examining publicly available information about the token on social networks, including a detailed overview of the official website and analysis of the latest messages and opinions about the token.

2. Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.

3. Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.



Remark: Manual and Automated review approaches can be mixed and matched including business logic analysis in terms of malicious doers' perspective

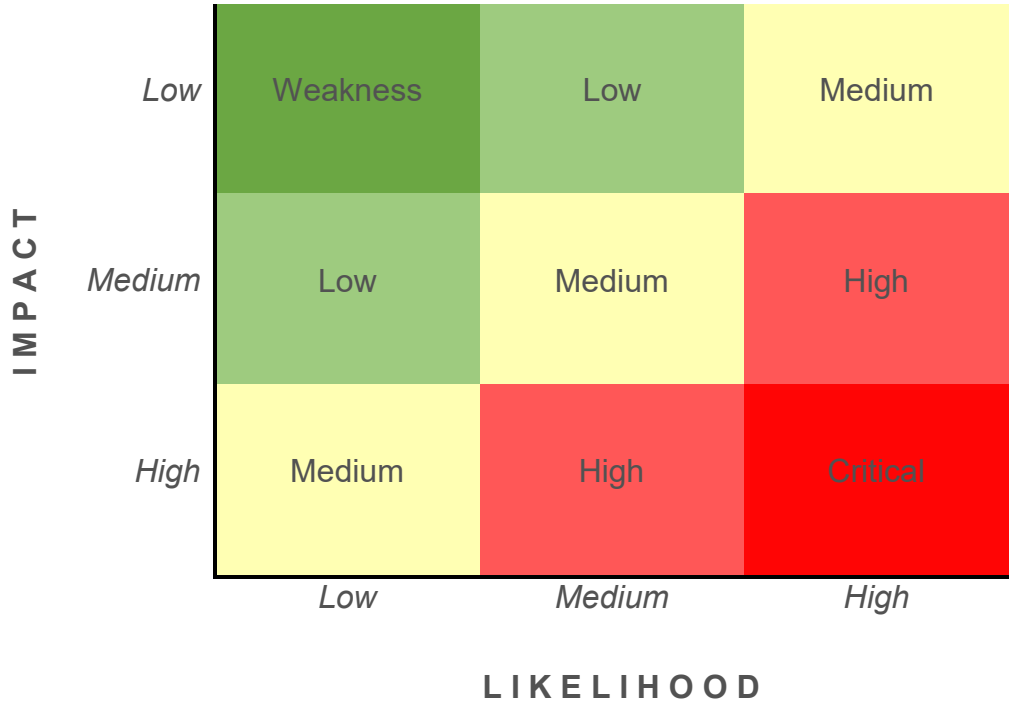
In particular, we perform the audit according to the following procedure:

- **Planning & Understanding**
  - determine scope of testing and understand application purpose and workflows;
  - identify key risk areas, including technical and business risks;
  - determine approach – which sections to review within the resource constraints and review method – automated, manual or mixed.
- **Automated Review**
  - adjust automated source code review tools to inspect the code for known unsafe coding patterns;
  - verify output of the tool in order to eliminate false positive result, and if necessary, adjust and re-run the code review tool.
- **Manual Review**
  - testing for business logic flaws requires thinking in unconventional methods;
  - identify unsafe coding behavior via static code analysis.
- **Reporting**
  - analyze the root cause of the flaws;
  - recommend coding process improvements.

3 Risk Level Classification

To standardize the evaluation, we define the following terminology based on OWASP Risk Rating Methodology:

- **Likelihood** represents how likely a particular vulnerability is to be uncovered and exploited in the wild.
- **Impact** measures the technical loss and business damage of a successful attack.
- **Severity** demonstrates the overall criticality of the risk and calculated as the product of impact and likelihood values, illustrated in a twodimensional matrix. The shading of the matrix visualizes the different risk levels.



Remark: Likelihood and Impact are categorized into three levels: H, M, and L, i.e., High, Medium and Low respectively. Severity is determined by likelihood and impact and can be classified into five categories accordingly, i.e., Critical, High, Medium, Low and Weakness

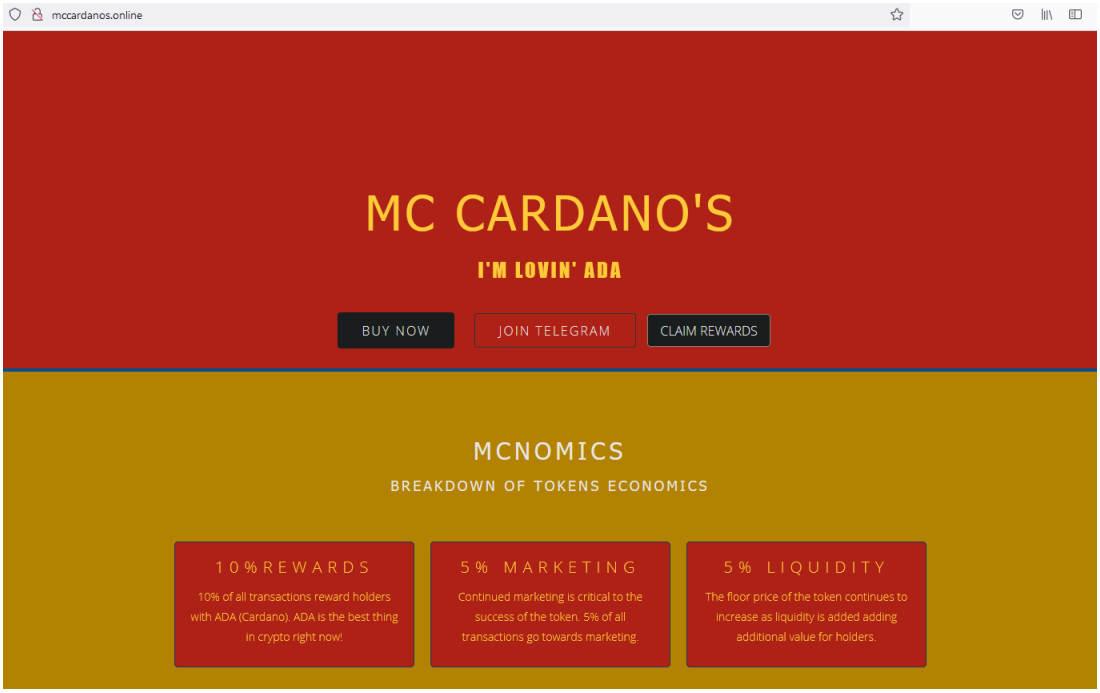
For prioritization of the vulnerabilities, we have adopted the scheme by five distinct levels for risk: Critical, High, Medium, Low, and Weakness. The risk level definitions are presented in table.

LEVEL	DESCRIPTION
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project

4 Project Overview

4.1 Communication Channels

http://mccardanos.online/



Website was registered on 05-09-2021, registration expires 05-09-2022.

Above the image is an actual snapshot of the current live website of the project.

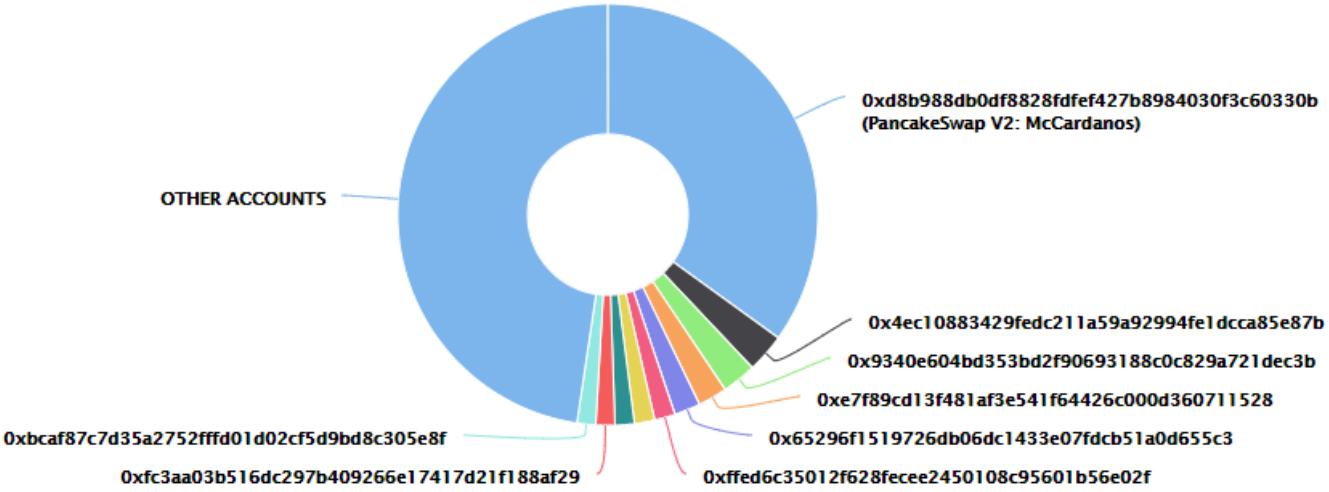
✓ Mobile Friendly	✓ 1 Social Media Networks [RISK]
✓ No JavaScript Errors	✓ < 1000 Telegram Members [RISK]
✓ No Valid SSL Certificate [RISK]	✓ No Active Voice Chats [RISK]
✓ Visionary Roadmap	✓ No Injected Spam Found
✓ No Contact Form [RISK]	✓ No Popus Found
✓ Spell Check	

Remark: This page contains active links

4.2 Smart Contract Details

Contract Name	McCardanos
Contract Address	0xd53E008A5e3eBd82FC65E6562F1F4c3dF56667a7
Total Supply	1,000,000,000
Token Ticker	McCardanos
Decimals	9
Token Holders	192
Transactions Count	920
Top 50 Holders Dominance	82,35%
Liquidity Fee	5%
Tax Fee	5%
Total Fees	14591526712856820759
Uniswap V2 Pair	0xaa7fff3209af9483267c74f25a6652fd2488bb79
Contract Deployer Address	0xd97e5c031b29a8157f2a7843eF6F26610a054624
Current Owner Address	0xd97e5c031b29a8157f2a7843eF6F26610a054624

McCardanos Top 10 Token Holders



Rank	Address	Quantity (Token)	Percentage
1	PancakeSwap V2: McCardanos	349,499,270.66743032	34.9499%
2	0x4ec10883429fedc211a59a92994fe1dcca85e87b	29,886,690.928592067	2.9887%
3	0x9340e604bd353bd2f90693188c0c829a721dec3b	26,428,995	2.6429%
4	0xe7f89cd13f481af3e541f64426c000d360711528	22,416,362.274459985	2.2416%
5	0x65296f1519726db06dc1433e07fdcb51a0d655c3	20,000,000	2.0000%
6	0xffed6c35012f628fecee2450108c95601b56e02f	15,918,757.907640123	1.5919%
7	0x78996b5a60e711904318586983f0e98fbfcf53c3	15,668,494.793163316	1.5668%
8	0x5771c595ebabaeba09dad434600435dde6f8b678	14,830,234.9	1.4830%
9	0xfc3aa03b516dc297b409266e17417d21f188af29	14,596,512.16	1.4597%
10	0xbcaf87c7d35a2752fffd01d02cf5d9bd8c305e8f	14,353,777.8	1.4354%

✓ **Pancakeswap holds ~35% of the token's supply as liquidity**

[RISK] No information about LP tokens holders

4.3 Contract Function Details

$ = payable function
# = non-constant function
[Int] = Internal
[Pub] = Public
[Prv] = Private
[Ext] = External

+ [Int] IBEP20
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] transfer #
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transferFrom #

+ [Int] IPancakeERC20
  - [Ext] name
  - [Ext] symbol
  - [Ext] decimals
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transfer #
  - [Ext] transferFrom #
  - [Ext] DOMAIN_SEPARATOR
  - [Ext] PERMIT_TYPEHASH
  - [Ext] nonces
  - [Ext] permit #

+ [Int] IPancakeFactory
  - [Ext] feeTo
  - [Ext] feeToSetter
  - [Ext] getPair
  - [Ext] allPairs
  - [Ext] allPairsLength
  - [Ext] createPair #
  - [Ext] setFeeTo #
  - [Ext] setFeeToSetter #

+ [Int] IPancakeRouter01
  - [Ext] addLiquidity #
  - [Ext] addLiquidityETH $
  - [Ext] removeLiquidity #
  - [Ext] removeLiquidityETH #
  - [Ext] removeLiquidityWithPermit #

- [Ext] removeLiquidityETHWithPermit #
- [Ext] swapExactTokensForTokens #
- [Ext] swapTokensForExactTokens #
- [Ext] swapExactETHForTokens $
- [Ext] swapTokensForExactETH #
- [Ext] swapExactTokensForETH #
- [Ext] swapETHForExactTokens $
- [Ext] factory
- [Ext] WETH
- [Ext] quote
- [Ext] getAmountOut
- [Ext] getAmountIn
- [Ext] getAmountsOut
- [Ext] getAmountsIn

+ [Int] IUniswapV2Router02 (IUniswapV2Router01)
- [Ext] removeLiquidityETHSupportingFeeOnTransferTokens #
- [Ext] removeLiquidityETHWithPermitSupportingFeeOnTransferTokens #
- [Ext] swapExactTokensForTokensSupportingFeeOnTransferTokens #
- [Ext] swapExactETHForTokensSupportingFeeOnTransferTokens $
- [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #

+ Ownable
- [Pub] owner
- [Pub] renounceOwnership #
  - modifiers: onlyOwner
- [Pub] transferOwnership #
  - modifiers: onlyOwner

+ [Lib] Address
- [Int] isContract
- [Int] sendValue #
- [Int] functionCall #
- [Int] functionCall #
- [Int] functionCallWithValue #
- [Int] functionCallWithValue #
- [Int] functionStaticCall
- [Int] functionStaticCall
- [Int] functionDelegateCall
- [Int] functionDelegateCall
- [Prv] verifyCallResult

+ [Lib] EnumerableSet
- [Prv] _add #
- [Prv] _remove #
- [Prv] _contains
- [Prv] _length
- [Prv] _at
- [Int] add #
- [Int] remove #

```
  - [Int] contains
  - [Int] length
  - [Int] at
  - [Int] add #
  - [Int] remove #
  - [Int] contains
  - [Int] length
  - [Int] at
  - [Int] add #
  - [Int] remove #
  - [Int] contains
  - [Int] length
  - [Int] at

+ McCardanos (IBEP20, Ownable)
  - [Prv] _transfer #
  - [Prv] _taxedTransfer #
  - [Prv] _feelessTransfer
  - [Prv] _calculateFee
  - [Pub] isExcludedFromStaking
  - [Pub] getTotalShares
  - [Prv] _addToken
  - [Prv] _removeToken
  - [Prv] _newDividentsOf
  - [Prv] _distributeStake
  - [Prv] claimBTC #
  - [Prv] _swapContractToken
    - modifiers: lockTheSwap
  - [Prv] _swapTokenForBnb #
  - [Prv] addLiquidity #
  - [Pub] getLiquidityReleaseTimeInSeconds
  - [Pub] getBurnedTokens
  - [Pub] getLimits
  - [Pub] getTaxes
  - [Pub] getAddressSellLockTimeInSeconds #
  - [Pub] getSellLockTimeInSeconds
  - [Pub] getAddressBuyLockTimeInSeconds
  - [Pub] AddressResetSellLock
  - [Pub] AddressResetBuyLock
  - [Pub] Rewards
  - [Pub] getDividents
  - [Pub] TeamWithdrawALLMarketingBNB $
    - modifiers: onlyOwner
  - [Pub] TeamWithdrawXMarketingBNB $
    - modifiers: onlyOwner
  - [Pub] TeamSwitchManualBNBConversion #
    - modifiers: onlyOwne
  - [Pub] TeamChangeMaxBuy
    - modifiers: onlyOwner
  - [Pub] TeamChangeTeamWallet $
```

  - modifiers: onlyOwner
- [Pub] TeamChangeWalletTwo $
  - modifiers: onlyOwner
- [Pub] TeamDisableSellLock
  - modifiers: onlyOwner
- [Pub] TeamDisableBuyLock
  - modifiers: onlyOwner
- [Pub] TeamSetSellLockTime #
  - modifiers: onlyOwner
- [Pub] TeamSetBuyLockTime #
  - modifiers: onlyOwner
- [Pub] AddWalletExclusion
  - modifiers: onlyOwner
- [Pub] TeamSetTaxes #
  - modifiers: onlyOwner
- [Pub] TeamChangeMarketingShare
  - modifiers: onlyOwner
- [Pub] TeamCreateLPandBNB
  - modifiers: onlyOwner
- [Pub] TeamUpdateLimits #
  - modifiers: onlyOwner
- [Pub] SetupEnableTrading
  - modifiers: onlyOwner
- [Pub] SetupLiquidityTokenAddress
  - modifiers: onlyOwner
- [Pub] TeamUnlockLiquidityInSeconds
  - modifiers: onlyOwner
- [Prv] _prolongLiquidityLock #
- [Pub] TeamReleaseLiquidity #
  - modifiers: onlyOwner
- [Pub] TeamRemoveLiquidity #
  - modifiers: onlyOwner
- [Pub] TeamRemoveRemainingBNB #
  - modifiers: onlyOwner
- [Ext] getOwner
- [Pub] name
- [Pub] symbol
- [Pub] decimals
- [Pub] totalSupply
- [Pub] balanceOf
- [Pub] transfer #
- [Pub] allowance
- [Pub] approve #
- [Prv] _approve #
- [Pub] transferFrom #
- [Pub] increaseAllowance #
- [Pub] decreaseAllowance #

4.4 Issues Checking Status

CHECKING ITEM	NOTES	RESULT
Arbitrary Jump with Function Type Variable	N / A	PASS
Arithmetic Accuracy Deviation	N / A	PASS
Assert Violation	N / A	PASS
Authorization through tx.origin	N / A	PASS
Business Logic	N / A	PASS
Code with No Effects	N / A	PASS
Critical Solidity Compiler	N / A	PASS
Delegatecall to Untrusted Callee	N / A	PASS
Design Logic	N / A	LOW RISK
DoS with Block Gas Limit	N / A	PASS
DoS with Failed Call	N / A	PASS
Function Default Visibility	N / A	PASS
Hash Collisions With MVLA	N / A	PASS
Incorrect Constructor Name	N / A	PASS
Incorrect Inheritance Order	N / A	PASS
Integer Overflows and Underflows	N / A	PASS
Lack of Proper Signature Verification	N / A	PASS
Message Call with Hardcoded Gas Amount	N / A	PASS
Missing Protection Against SRA	N / A	PASS
Presence of Unused Variables	N / A	PASS
Reentrancy	N / A	PASS
Requirement Violation	N / A	PASS

CHECKING ITEM	NOTES	RESULT
Right-To-Left-Override Control Character	N / A	PASS
Shadowing State Variables	N / A	PASS
Signature Malleability	N / A	PASS
State Variable Default Visibility	N / A	PASS
Timestamp Dependence	N / A	PASS
Transaction Order Dependence	N / A	PASS
Typographical Error	N / A	PASS
Unencrypted Private Data On-Chain	N / A	PASS
Unexpected Ether balance	N / A	PASS
Uninitialized Storage Pointer	N / A	PASS
Use of Deprecated Solidity Functions	N / A	PASS
Weak Sources of Randomness From CA	N / A	PASS
Write to Arbitrary Storage Location	N / A	PASS

Remark: To evaluate the risk, we go through a list of check items and each would be labeled with a severity category. For one check item, if our tool or analysis does not identify any issue, the contract is considered safe regarding the check item

4.5 Detailed Findings Information

[RISK] Logical Issue

- The if condition isLiquidityTransfer is logically wrong and will always be false.

```
bool isLiquidityTransfer = ((sender == _pancakePairAddress && recipient == pancakeRouter)
|| (recipient == _pancakePairAddress && sender == pancakeRouter));
```

https://github.com/Uniswap/uniswap-v2-periphery/blob/dda62473e2da448bc9cb8f4514dadda4aeede5f4/contracts/UniswapV2Router02.sol#L61

and addLiquifityETH

https://github.com/Uniswap/uniswap-v2-periphery/blob/dda62473e2da448bc9cb8f4514dadda4aeede5f4/contracts/UniswapV2Router02.sol#L77

both of them use transferFrom which means the sender in _transfer will be the "real" sender and not the router address. Therefore, the condition recipient == pancakeRouter will always be false and the condition sender == pancakeRouter will always be false.

There is no way to differentiate between sell transaction and addLiquidity transaction and buy transaction and removeLiquidity transaction from just looking on the sender/recipient.

[RISK] Logical Issue

- The function claimBTC calls swapExactETHForTokensSupportingFeeOnTransferTokens function, always make sure its error cases are handled gracefully!

  Recommendation: Use try-catch.

[RISK] Volatile Code

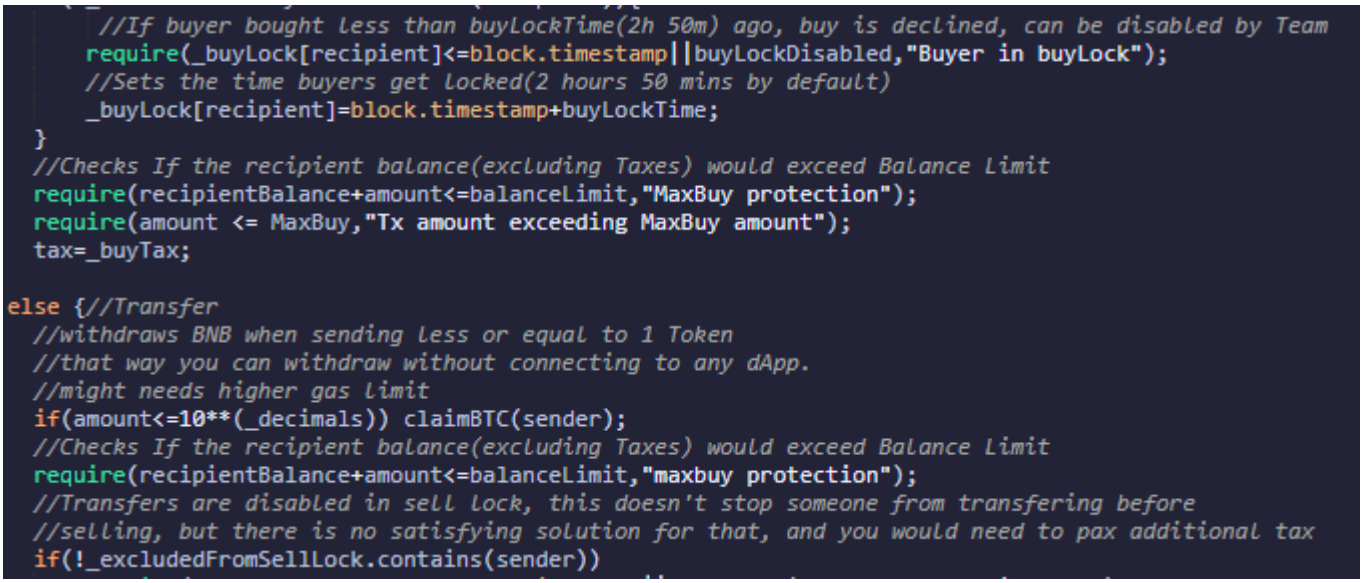
- _transfer should always work, even if there is a bug in the contract, to ensure that investors' funds are safe. If the function is critical (such as _transfer) always make sure its error cases are handled gracefully!

  _transfer calls _swapTokenForBNB, _addLiquidity, _distributeStake which could fail.

  Recommendation: Use try-catch when calling external functions.

[RISK] Logical Issue

- The owner cannot remove liquidity without setting balanceLimit to be as high as the number of tokens that are in the liquidity pool. Note that balanceLimit can never be reduced.



```
        //If buyer bought less than buyLockTime(2h 50m) ago, buy is declined, can be disabled by Team
        require(_buyLock[recipient]<=block.timestamp||buyLockDisabled,"Buyer in buyLock");
        //Sets the time buyers get locked(2 hours 50 mins by default)
        _buyLock[recipient]=block.timestamp+buyLockTime;
    }
    //Checks If the recipient balance(excluding Taxes) would exceed Balance Limit
    require(recipientBalance+amount<=balanceLimit,"MaxBuy protection");
    require(amount <= MaxBuy,"Tx amount exceeding MaxBuy amount");
    tax=_buyTax;

else {//Transfer
    //withdraws BNB when sending less or equal to 1 Token
    //that way you can withdraw without connecting to any dApp.
    //might needs higher gas limit
    if(amount<=10**(_decimals)) claimBTC(sender);
    //Checks If the recipient balance(excluding Taxes) would exceed Balance Limit
    require(recipientBalance+amount<=balanceLimit,"maxbuy protection");
    //Transfers are disabled in sell lock, this doesn't stop someone from transfering before
    //selling, but there is no satisfying solution for that, and you would need to pax additional tax
    if(!_excludedFromSellLock.contains(sender))
```

[RISK] Logical Issue

- Anyone can send BNB to the contract.

  Recommendation: Our recommendation is to slightly modify the receive() function and to reject any BNB that wasn't received from the router or other specific addresses in order to prevent a case where investors mistakenly send BNB to the contract.

[RISK] Owner Privileges (in the period when the owner is not renounced)

The contract contains the following privileged functions that are restricted by the onlyOwner.

- The owner can set the burnTaxes, liquidityTaxes, stakingTaxes, buyTax, sellTax, transferTax.
- The owner can change TeamWallet and WalletTwo.
- The owner can disable the timeLock after selling for everyone.
- The owner can change unlock time and remove liquidity.

5 Audit Result

LEVEL	ISSUES
Weakness	**Design Logic (5)**
Low	**Owner Privilegies (4)**

1. The contract don’t utilizes SafeMath libraries.

2. The team manually lock the LP tokens that were automatically added by calling swapAndLiquify.

3. The team allows unlocking LP tokens and lock the remaining LP tokens.

4. Ownership has not been renounced.

5.1 Findings Summary

McCardanos

Low Risk Level

- ✓ No external vulnerabilities were identified within the smart contract's code
- ✓ We strongly recommend that the team renounces ownership
- ✓ Please ensure trust in the team prior to investing as they have substantial control within the ecosystem
- ✓ We strongly recommend that the contract owners remove errors and re-audit

6 Disclamer

CheckPoint team issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these. For the facts that occurred or existed after the issuance, CheckPoint is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to CheckPoint by the information provider till the date of the insurance report. CheckPoint is not responsible for the background and other conditions of the project.

This security audit is not produced to supplant any other type of assessment and does not guarantee the discovery of all security vulnerabilities within the scope of the assessment. However, we warrant that this audit is conducted with goodwill, professional approach, and competence. Since an assessment from one single party cannot be confirmed to cover all possible issues within the smart contract(s), CheckPoint suggests conducting multiple independent assessments to minimize the risks. Lastly, nothing contained in this audit report should be considered as investment advice.