CheckPoint

Token Security Audit Report
Prepared for Dynamix

[v.1.0]

October 2021

Document Properties

Client	Dynamix
Platform	Binance Smart Chain
Language	Solidity
Codebase	0xc41689A727469C1573009757200371edf36D540e

Audit Summary

Delivery Date	08.10.2021
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 08/10/2021, CheckPoint conducted a full audit for the Dynamix 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.

Dynamix (DYNA)
Low Risk Level

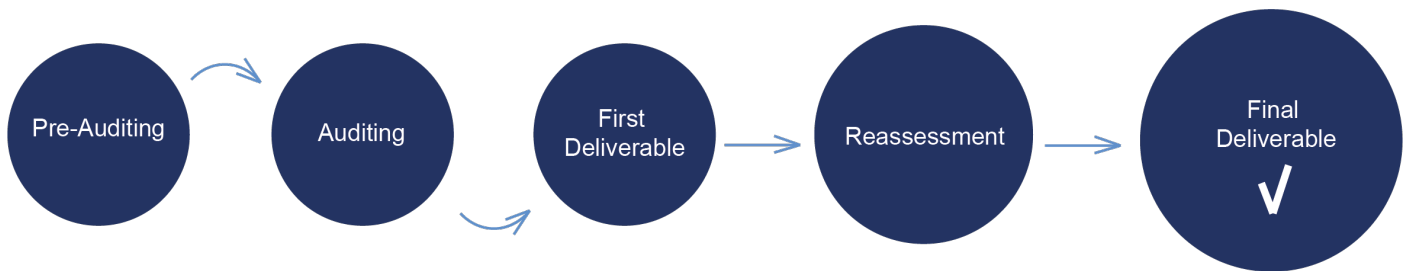|	
Communication Channels	Website Content Analysis, Social Media Listening
Smart Contract Code	Smart Contract Details, Contract Function Details, Issues Checking Status, Detailed Findings Information

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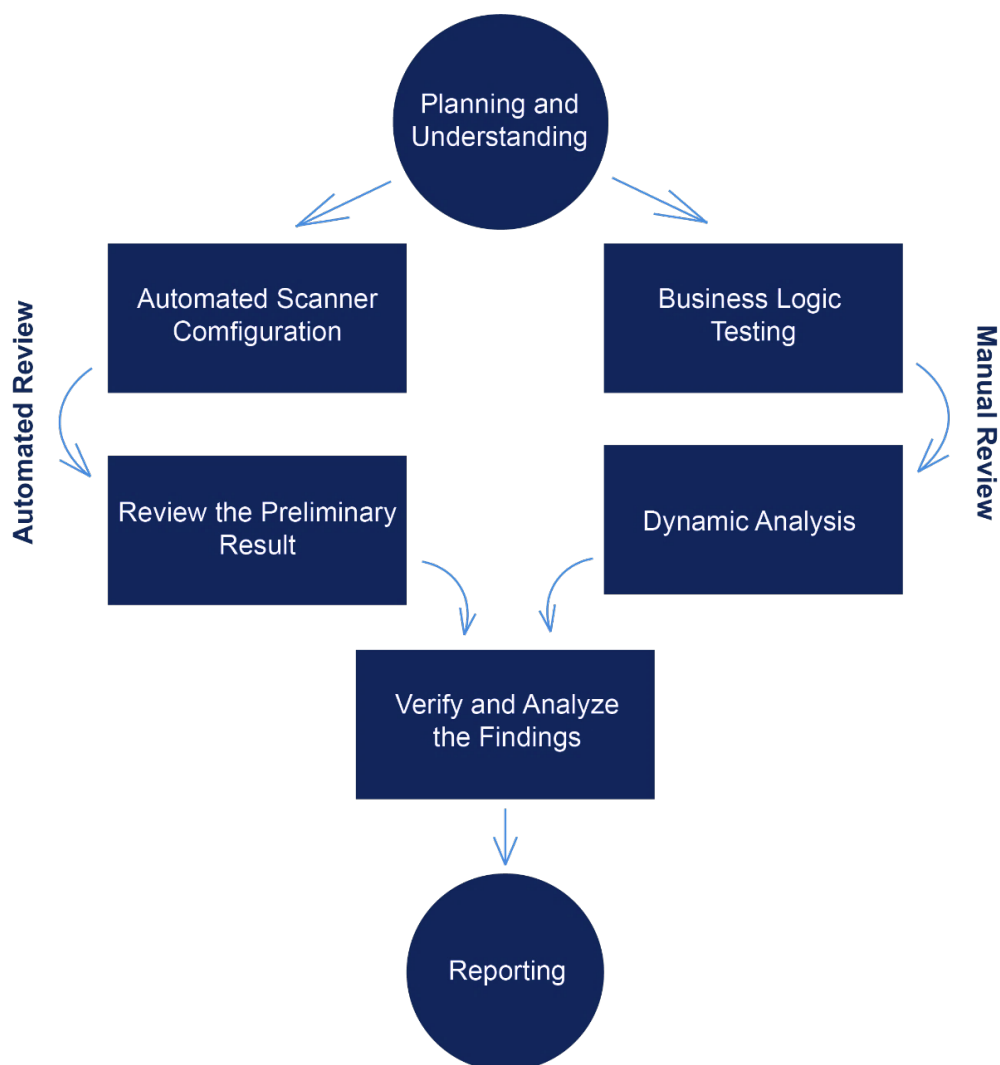
Remark: Manual and Automated review approaches can be mixed and matched including business logic analysis in terms of malicious doers' perspective

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

- **Planning & Understanding**

    o determine scope of testing and understand application purpose and workflows;

    o identify key risk areas, including technical and business risks;

    o determine approach – which sections to review within the resource constraints and review method – automated, manual or mixed.

- **Automated Review**

    o adjust automated source code review tools to inspect the code for known unsafe coding patterns;

    o 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**

    o testing for business logic flaws requires thinking in unconventional methods;

    o identify unsafe coding behavior via static code analysis.

- **Reporting**

    o analyze the root cause of the flaws;

    o 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.

IMPACT			
Low	Weakness	Low	Medium
Medium	Low	Medium	High
High	Medium	High	Critical
	Low	Medium	High

LIKELIHOOD

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

https://dynamix.finance
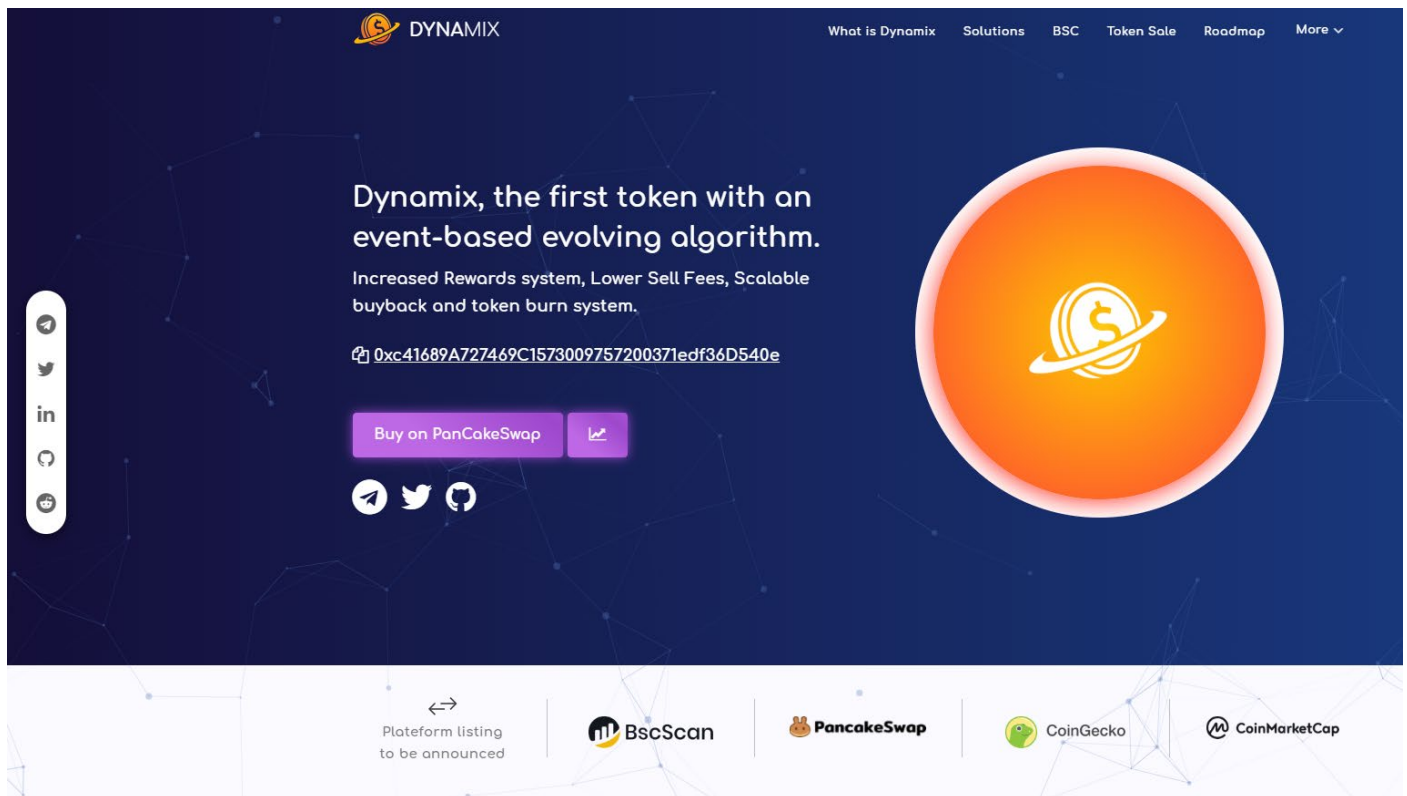
Above the image is an actual snapshot of the current live website of the project.

✓ Mobile Friendly	✓ 5 Social Media Networks
✓ No JavaScript Errors	✓ Over 10k Telegram Members
✓ Visionary Roadmap	✓ 4000+ Twitter Followers
✓ Spell Check	✓ Active Voice Chats
✓ Valid SSL Certificate	✓ No Injected Spam and Popus Found

Remark: This page contains active links

4.2 Smart Contract Details

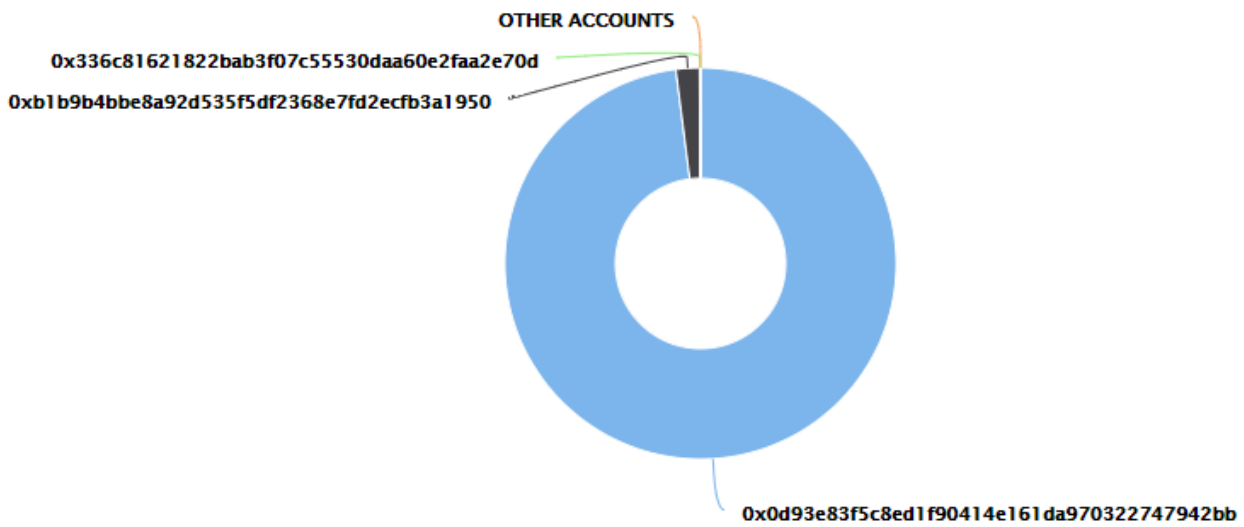|	
Contract Name	Dynamix
Contract Address	0xc41689A727469C1573009757200371edf36D540e
Total Supply	1,000,000,000,000,000
Token Ticker	DYNA
Decimals	9
Token Holders	13,693
Transactions Count	60,984
Top 100 Holders Dominance	81,84%
Buy Fee	12%
Sell Fee	17%
Minimum BNB Before Buy	1000
Minimum Tokens Before Sell	1000
Uniswap V2 Pair Contract	0xfa1163ec4f5e2cc0031b4d0eb1bab2b6d9d90674
Contract Deployer Address	0xb3fb1293533e734fd8cfb1c99ae873797401e177
Current Owner Address	0xb3fb1293533e734fd8cfb1c99ae873797401e177

Dynamix Top 10 Token Holders
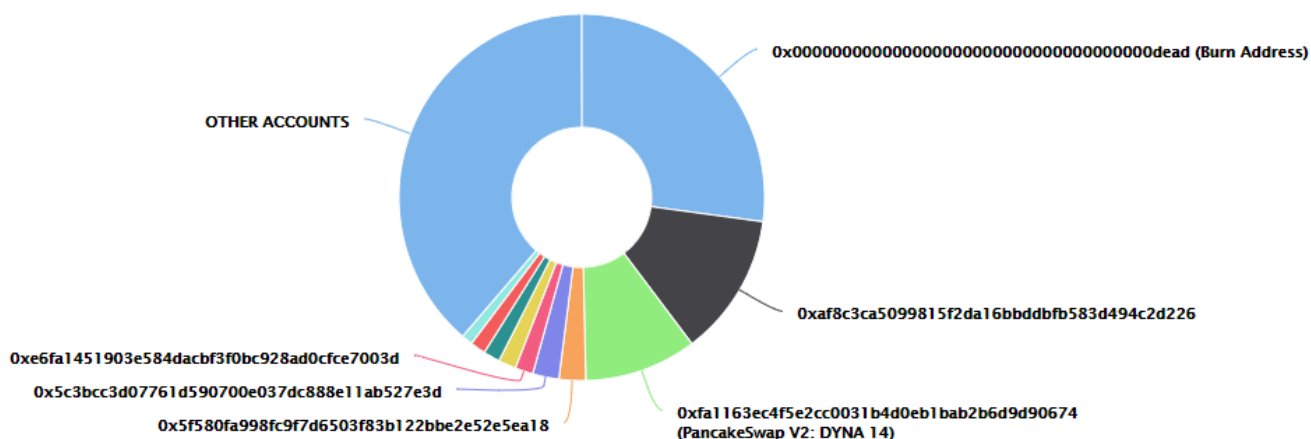
Rank	Address	Quantity (Token)	Percentage
1	Burn Address	271,370,878,602,379.418728915	27.1371%
2	0xaf8c3ca5099815f2da16bbddbfb583d494c2d226	125,779,987,033,247.15335632	12.5780%
3	PancakeSwap V2: DYNA 14	99,367,706,594,224.153244927	9.9368%
4	0x5f580fa998fc9f7d6503f83b122bbe2e52e5ea18	23,545,939,459,974.413315077	2.3546%
5	0x5c3bcc3d07761d590700e037dc888e11ab527e3d	23,293,300,036,185.317919794	2.3293%
6	0xe6fa1451903e584dacbf3f0bc928ad0cfce7003d	15,933,369,785,673.708396451	1.5933%
7	0x9f1439a98e00df8879c9564fa644f83808a2c7b0	15,563,059,639,350.285300324	1.5563%
8	0xd783263633b56156d1f8b5b16010d9be96962cfc	14,679,946,000,761.840207362	1.4680%
9	0x597fa8fc25a1d74f50d86c02e86accd225e8908a	13,625,830,965,471.830351346	1.3626%
10	0x914143c2acec9a7ef175112607ec2b785f09aeff	9,862,195,568,300.580667951	0.9862%

✓ **~27% tokens are permanently removed from circulation**

[RISK] **PancakeSwap holds only ~9,9% of the token's supply as liquidity**

Dynamix Top 3 LP Token Holders

Rank	Address	Quantity (Token)	Percentage
1	0x0d93e83f5c8ed1f90414e161da970322747942bb	5,159.87858041186615056	97.9825%
2	0xb1b9b4bbe8a92d535f5df2368e7fd2ecfb3a1950	102.906358107364994296	1.9541%
3	0x336c81621822bab3f07c55530daa60e2faa2e70d	2.86814834733474813	0.0545%

[RISK] 1 wallet have ~98,0% LP tokens

Dynamix Contract Interaction Details
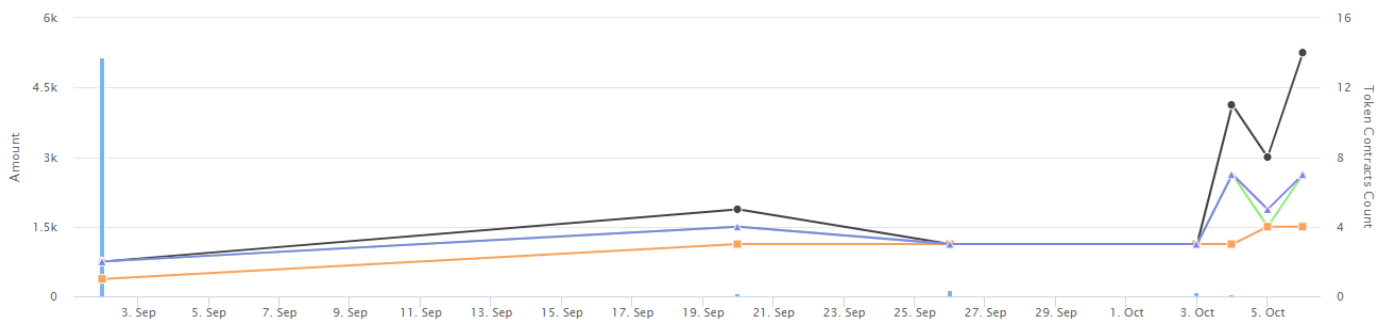
4.3 Contract Function Details

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

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

+ Fee (Ownable)
   - [Int] <Constructor> #
   - [Ext] setExcludedFromFee
      - modifiers: onlyOwner
   - [Ext] setPair
      - modifiers: onlyOwner
   - [Int] _isBuy #
   - [Int] _isSell #
   - [Int] _getBuyFee #
   - [Int] _getRewardFee #
   - [Int] _getSellFee #
   - [Int] _getHoldFee #
   - [Ext] setTeamAddress #
      - modifiers: onlyOwner
   - [Ext] disableFee #
      - modifiers: onlyOwner
   - [Ext] enableFee #
      - modifiers: onlyOwner

+ Holder
   - [Int] <Constructor> #
   - [Int] recipientTransfert #
   - [Int] senderTransfert #

+ [Int] IERC20
   - [Ext] totalSupply
   - [Ext] balanceOf
   - [Ext] transfer #
   - [Ext] allowance
   - [Ext] approve #
   - [Ext] transferFrom #

+ Context
  - [Int] _msgSender

+ Ownable (Context)
  - [Int] <Constructor> #
  - [Pub] owner
  - [Pub] renounceOwnership #
     - modifiers: onlyOwner
  - [Pub] transferOwnership #
     - modifiers: onlyOwner
  - [Pub] _setOwner

+ Reward (Ownable, Holder)
  - [Pub] <Constructor> #
  - [Int] _transfer #
  - [Int] _getTotalSupplyAndTotalReward #
  - [Int] _rewardToToken
  - [Prv] _getRate
  - [Ext] excludeAccountFromRewards #
      - modifiers: onlyOwner
  - [Ext] includeAccountInRewards #
      - modifiers: onlyOwner
  - [Ext] accountBalance #

+ [Lib] SafeMath
  - [Int] add
  - [Int] sub
  - [Int] sub
  - [Int] mul
  - [Int] div
  - [Int] div
  - [Int] mod
  - [Int] mod

+ Swap (Ownable)
  - [Int] <Constructor> #
  - [Ext] enableAutoBuy #
     - modifiers: onlyOwner
  - [Ext] disableAutoBuy #
     - modifiers: onlyOwner
  - [Int] _swapTokensForBNB
  - [Int] _buyBackAndBurnToken

+ [Int] IUniswapV2Factory
  - [Ext] feeTo
  - [Ext] feeToSetter
  - [Ext] getPair
  - [Ext] allPairs
  - [Ext] allPairsLength

   - [Ext] createPair #
   - [Ext] setFeeTo #
   - [Ext] setFeeToSetter #

+ [Int] IUniswapV2Pair
   - [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 #
   - [Ext] MINIMUM_LIQUIDITY
   - [Ext] factory
   - [Ext] token0
   - [Ext] token1
   - [Ext] getReserves
   - [Ext] price0CumulativeLast
   - [Ext] price1CumulativeLast
   - [Ext] kLast
   - [Ext] mint #
   - [Ext] burn #
   - [Ext] swap #
   - [Ext] skim #
   - [Ext] sync #
   - [Ext] initialize #

+ [Int] IUniswapV2Router01
   - [Ext] factory
   - [Ext] WETH
   - [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] quote
   - [Ext] getAmountOut

- [Ext] getAmountIn
- [Ext] getAmountsOut
- [Ext] getAmountsIn

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

+ Dynamix (Context, IERC20, Ownable)
   - [Pub] <Constructor> #
   - [Pub] totalSupply
   - [Pub] balanceOf
   - [Pub] transfer #
   - [Pub] allowance
   - [Pub] approve #
   - [Pub] transferFrom #
   - [Prv] _approve #
   - [Prv] _transfer #
   - [Ext] <Fallback> $
     - modifiers: lockTheSwap
   - [Prv] _sellAndBuy #
     - modifiers: lockTheSwap
   - [Ext] beforePreSale #
     - modifiers: onlyOwner
   - [Ext] afterPreSale #
     - modifiers: onlyOwner

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	PASS
DoS with Block Gas Limit	N / A	LOW RISK
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] DoS with Block Gas Limit

- The function _getTotalSupplyAndTotalReward uses the loop for evaluating total supply and total reward. It could be aborted with out-of-gas exception if there will be a long excluded addresses list. Including an account in the reward again may result in unexpected behavior.

```
function _getTotalSupplyAndTotalReward() private view returns(uint256, uint256) {
    uint256 rewardTotal = rewardSupply;
    uint256 tokenTotal = _tokenSupply;

    for (uint256 i = 0; i < excludedFromRewardAddr.length; i++) {
        uint256 reward = balances[ excludedFromRewardAddr[i]].reward;
        uint256 token = balances[ excludedFromRewardAddr[i]].token;

        if (reward > rewardTotal || token > tokenTotal)
            return (_rewardSupply, _tokenSupply);

        rewardTotal = rewardTotal.sub(reward);
        tokenTotal = tokenTotal.sub(token);
    }

    if (rewardTotal < rewardSupply.div( tokenSupply))
        return (_rewardSupply, _tokenSupply);

    return (rewardTotal, tokenTotal);
}
```

Recommendation: Consider removing the _getTotalSupplyAndTotalReward function. If this is not desired, consider avoiding it, especially on accounts with a significant balance.

- The function includeAccountInRewards uses the loop to find and remove addresses from the

  _excluded list. It could be aborted with out-of-gas exception if there will be a long excluded

  addresses list. Including an account in the reward again may result in unexpected behavior.

```solidity
function includeAccountInRewards(address account) external onlyOwner() {
    require(_balances[account].excludedFromReward, "Account is excluded");

    for (uint256 i = 0; i < _excludedFromRewardAddr.length; i++) {
        if (_excludedFromRewardAddr[i] == account) {
            _excludedFromRewardAddr[i] = _excludedFromRewardAddr[_excludedFromRewardAddr.length - 1];
            _excludedFromRewardAddr.pop();

            _balances[account].token = 0;
            _balances[account].excludedFromReward = false;

            break;
        }
    }
}
```

Recommendation: Consider removing the includeAccountInRewards function. If this is not desired, consider avoiding it, especially on accounts with a significant balance.

[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 of the contract can start and complete the presale. Using these functions the owner could set fees, buyback, minimum tokens before sell, and minimum BNB before buy.

```solidity
function beforePreSale() external onlyOwner() {
    sellFee = 0;
    buyFee = 0;
    autoBuyBackEnabled = false;
    minimumTokensBeforeSell = 0;
    minimumBNBBeforeBuy = 0;

    emit PreSaleStarted(sellFee, buyFee);
}

// After PreSale, initialization fees
ftrace | funcSig
function afterPreSale(address account) external onlyOwner() {
    sellFee = 17;
    buyFee = 12;
    _pair[account] = true;
    autoBuyBackEnabled = true;
    minimumTokensBeforeSell = 1000 * 10**9 * 10**9;
    minimumBNBBeforeBuy = 1 * 10**16;

    emit PreSaleCompleted(sellFee, buyFee);
}
```

- The owner of the contract has the ability to enable auto buyback.

```solidity
function enableAutoBuy(uint256 tokensBeforeSell, uint256 bnbBeforeBuy) external onlyOwner()
    autoBuyBackEnabled = true;
    minimumTokensBeforeSell = tokensBeforeSell;
    minimumBNBBeforeBuy = bnbBeforeBuy;

    emit autoBuyBackChanged(autoBuyBackEnabled, minimumTokensBeforeSell, minimumBNBBeforeBuy);
}

// Disable Auto BuyBack and Burn
ftrace | funcSig
function disableAutoBuy() external onlyOwner() {
    autoBuyBackEnabled = false;
    minimumTokensBeforeSell = 0;
    minimumBNBBeforeBuy = 0;

    emit autoBuyBackChanged(autoBuyBackEnabled, minimumTokensBeforeSell, minimumBNBBeforeBuy);
}
```

- The owner of the contract can exclude and include accounts from fees and reward distribution.

```solidity
function setExcludedFromFee(address account, bool excluded) external onlyOwner() {
    excludedFromFee[account] = excluded;
}
```

```solidity
function excludeAccountFromRewards(address account) external onlyOwner() {
    require(!_balances[account].excludedFromReward, "Account is not excluded");

    if(_balances[account].reward > 0)
        _balances[account].token = _rewardToToken(_balances[account].reward);

    _balances[account].excludedFromReward = true;

    excludedFromRewardAddr.push(account);
}

// Include an account in rewards
ftrace | funcSig
function includeAccountInRewards(address account) external onlyOwner() {
    require(_balances[account].excludedFromReward, "Account is excluded");

    for (uint256 i = 0; i < excludedFromRewardAddr.length; i++) {
        if (excludedFromRewardAddr[i] == account) {
            excludedFromRewardAddr[i] = excludedFromRewardAddr[excludedFromRewardAddr.length - 1];
            excludedFromRewardAddr.pop();

            _balances[account].token = 0;
            _balances[account].excludedFromReward = false;

            break;
        }
    }
}
```

- The owner of the contract can set to enable or disable fees.

```solidity
function disableFee() external onlyOwner() {
    sellFee = 0;
    buyFee = 0;
}

// Enable Fee
ftrace | funcSig
function enableFee() external onlyOwner() {
    sellFee = 17;
    buyFee = 12;
}
```

- The owner of the contract can set and change team address.

```
function setTeamAddress(address addr1) external onlyOwner() {
    teamAddress = addr1;
    excludedFromFee[teamAddress] = true;

    TeamAddressChanged(addr1);
}
```

5 Audit Result

LEVEL

ISSUES

Weakness

DoS with Block Gas Limit (2)

Low

Owner Privilegies (5)

5.1 Findings Summary

Dynamix
Low Risk Level

✓ **No external vulnerabilities were identified within the smart contract's code**

✓ **The code is fully customized**

✓ **Dynamix token was audited, and no issues were found**

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.