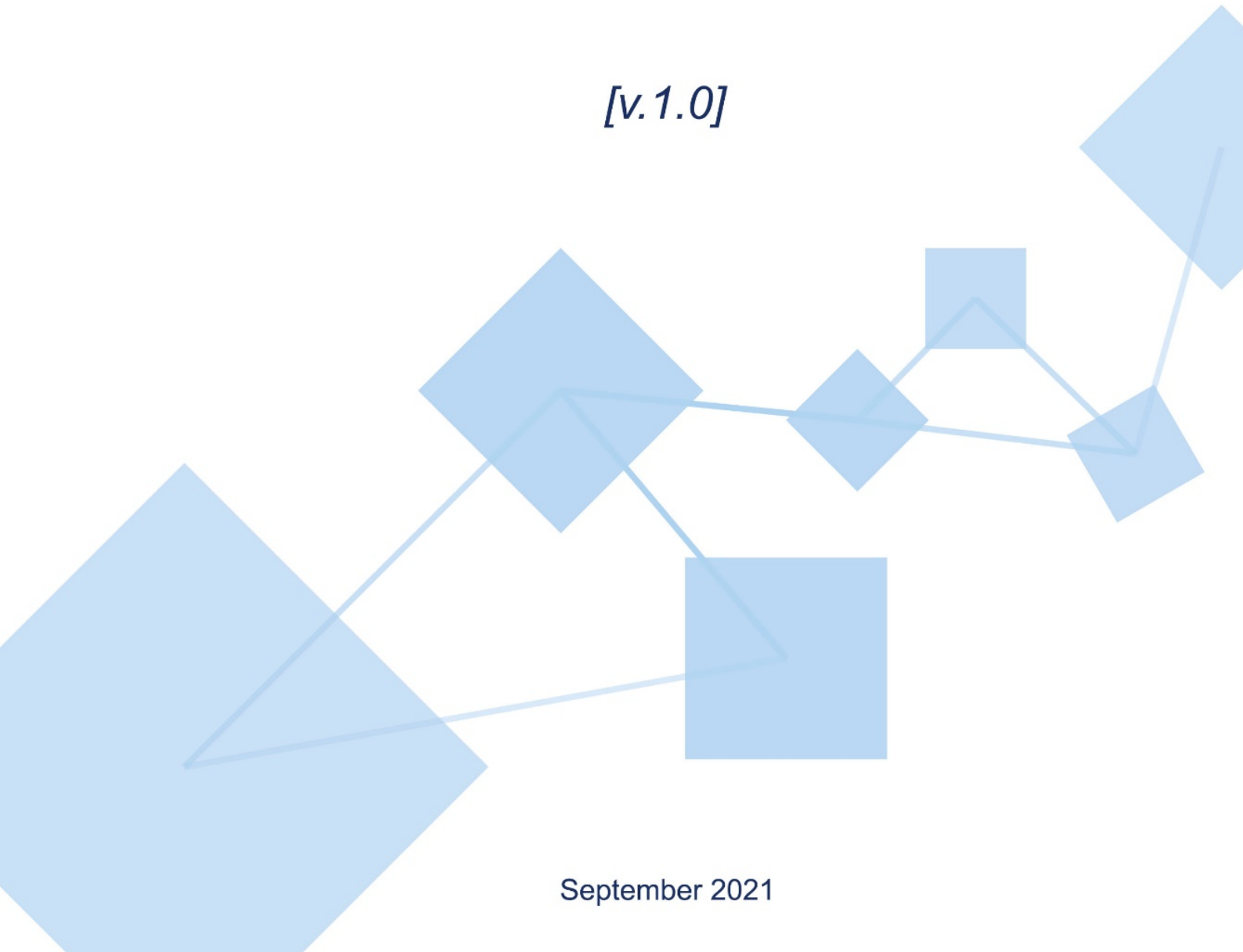
CheckPoint

Token Security Audit Report
Prepared for MiniSports

[v.1.0]



September 2021

Document Properties

Client	MiniSports
Platform	Binance Smart Chain
Language	Solidity
Codebase	0x75856ea207CE7194E4E65c334BEc143D77701E4a

Audit Summary

Delivery Date	10.09.2021
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 10/09/2021, CheckPoint conducted a full audit for the MiniSports 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.

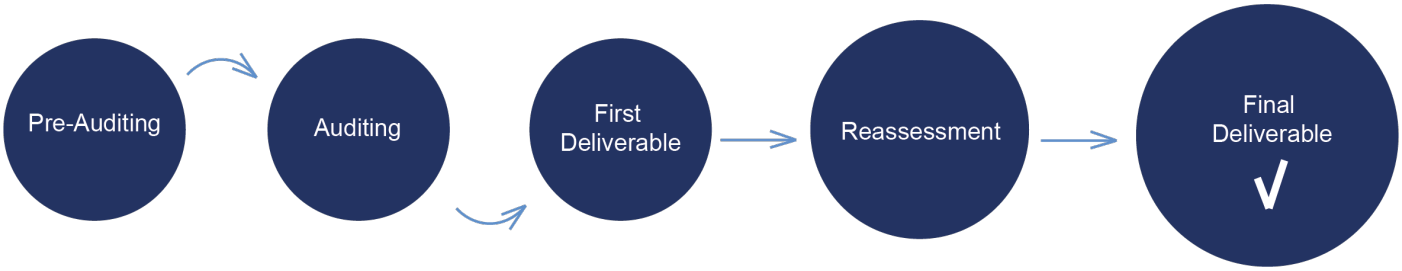
MiniSports

Medium 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**

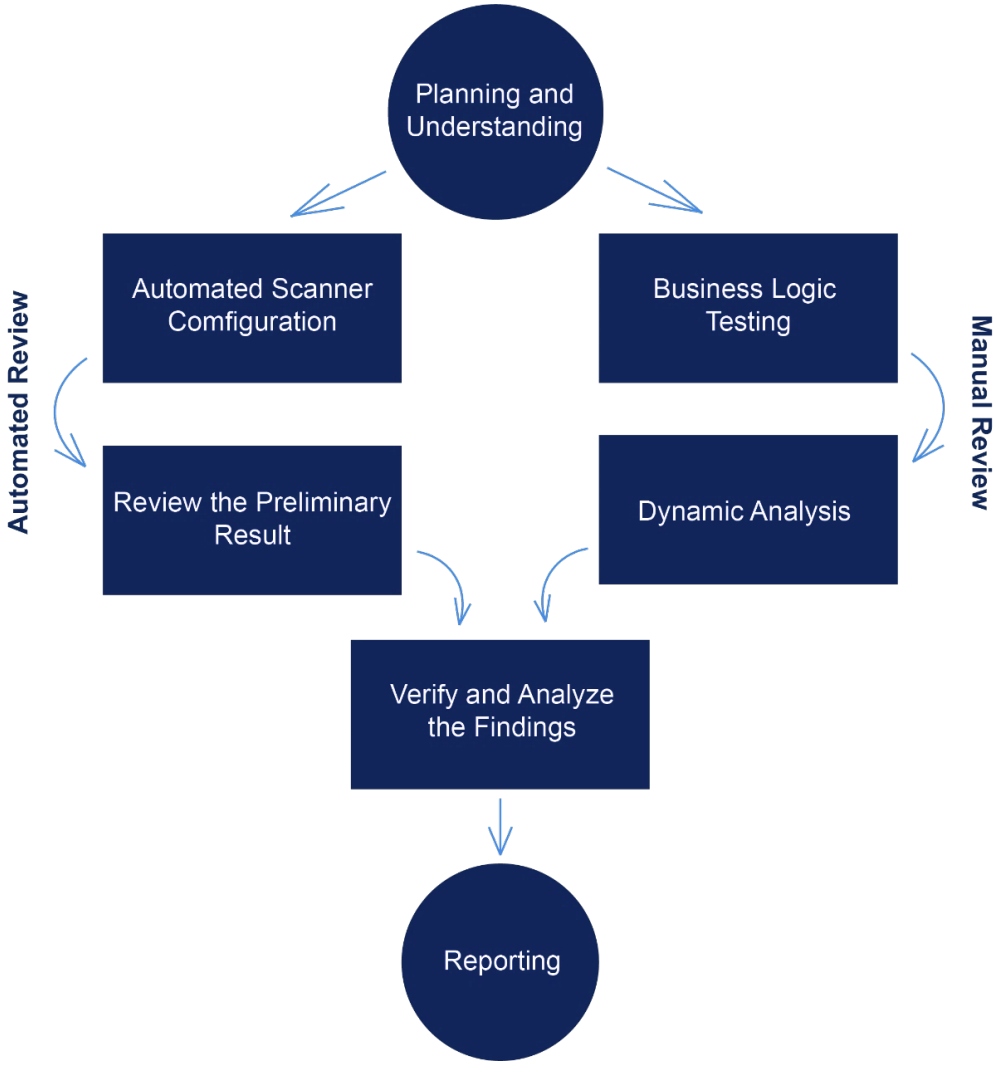
  Providing a full report with the detailed status of each issue.

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

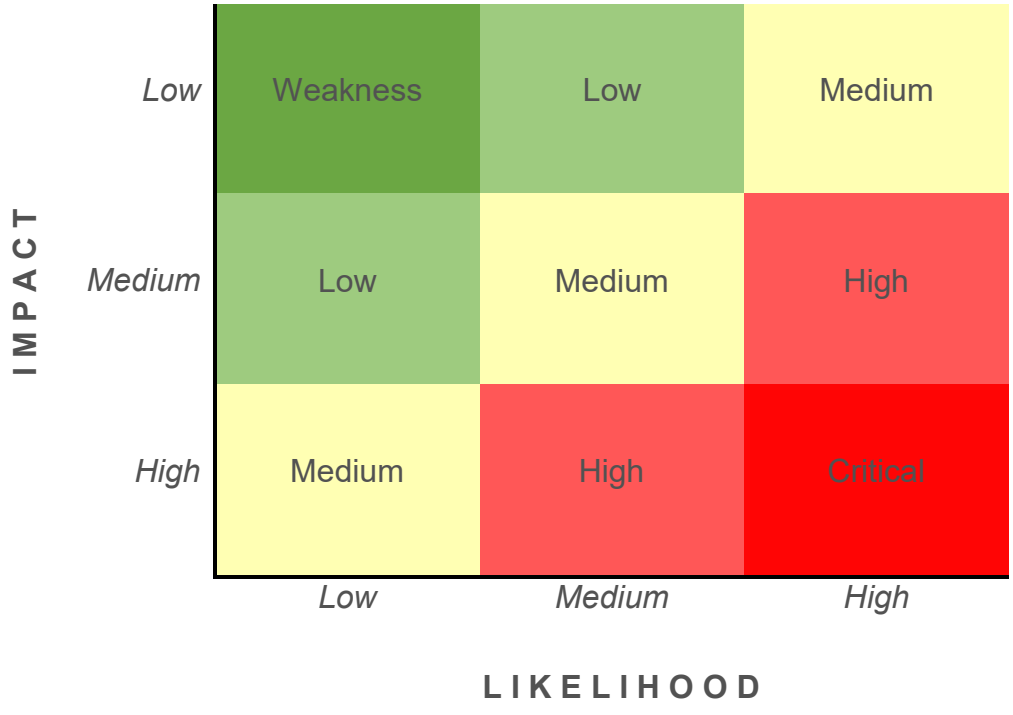
In particular, we perform the audit according to the following procedure:

- **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

https://www.minisportstoken.com/

Website was registered on 22-08-2021, registration expires 22-08-2022.

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

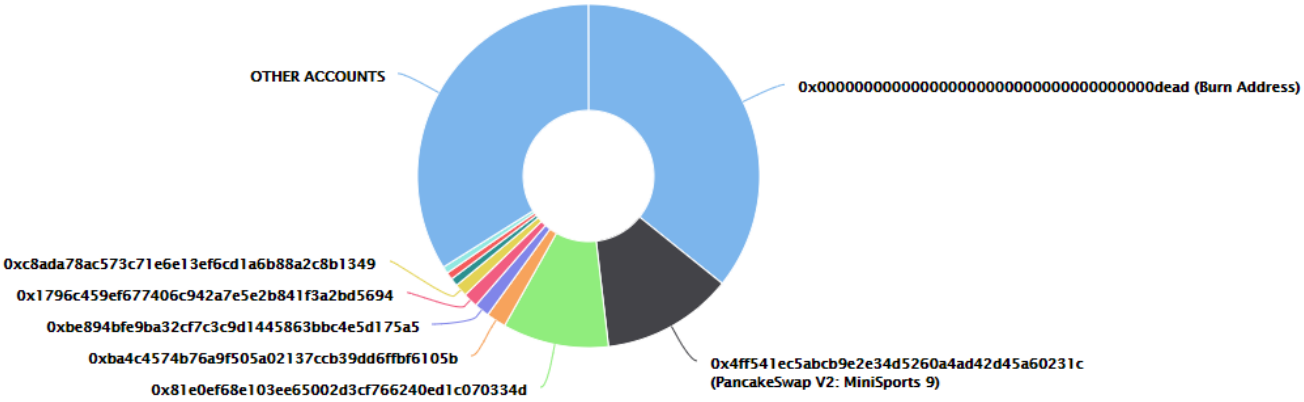
✓ Mobile Friendly	✓ 2 Social Media Networks
✓ No JavaScript Errors	✓ 2000+ Telegram Members
✓ Valid SSL Certificate	✓ 3000+ Twitter Followers
✓ Visionary Roadmap	✓ Active Voice Chats
✓ No Contact Form [RISK]	✓ No Injected Spam Found
✓ Spell Check	✓ No Popus Found

Remark: This page contains active links

4.2 Smart Contract Details

Contract Name	MiniSports
Contract Address	0x75856ea207CE7194E4E65c334BEc143D77701E4a
Total Supply	1,000,000,000,000,000
Token Ticker	MiniSports
Decimals	9
Token Holders	3,502
Transactions Count	9,857
Top 100 Holders Dominance	84,04%
Liquidity Fee	20%
Tax Fee	2%
Total Fees	18333157798308722900651
Uniswap V2 Pair	0x4ff541ec5abcb9e2e34d5260a4ad42d45a60231c
Contract Deployer Address	0x1796c459ef677406c942a7e5e2b841f3a2bd5694
Current Owner Address	0x1796c459ef677406c942a7e5e2b841f3a2bd5694

MiniSports Top 10 Token Holders

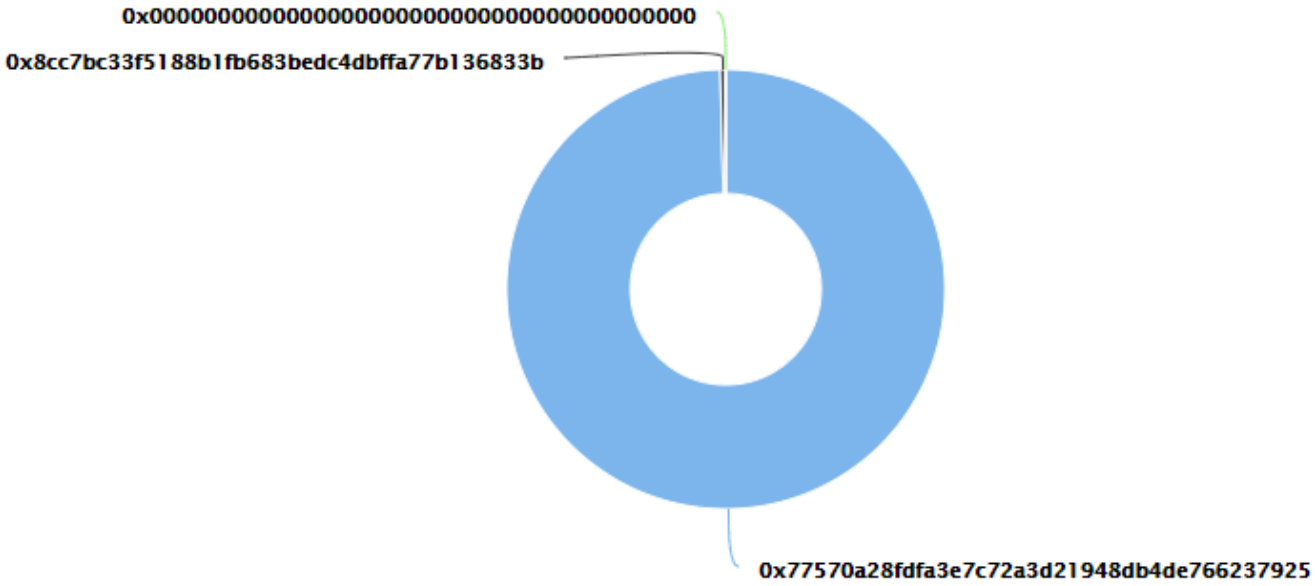


Rank	Address	Quantity (Token)	Percentage
1	Burn Address	357,637,681,648,819.189345195	35.7638%
2	PancakeSwap V2: MiniSports 9	123,827,681,395,311.578143363	12.3828%
3	0x81e0ef68e103ee65002d3cf766240ed1c070334d	100,263,160,674,591.08	10.0263%
4	0xba4c4574b76a9f505a02137ccb39dd6ffbf6105b	18,575,143,201,979.735025225	1.8575%
5	0xbe894bfe9ba32cf7c3c9d1445863bbc4e5d175a5	14,152,606,027,596.515011483	1.4153%
6	0x1796c459ef677406c942a7e5e2b841f3a2bd5694	13,890,903,152,708.997511896	1.3891%
7	0xc8ada78ac573c71e6e13ef6cd1a6b88a2c8b1349	11,868,291,731,726.05702052	1.1868%
8	0xdf2db035dad45354ab217799cefe136ac280296d	7,544,162,629,380.217262966	0.7544%
9	0xff06b16d39f612f57f65902d93bcc47fb58c42f6	6,794,950,574,310.033704467	0.6795%
10	0x8943ee1d6f6ff89245059a38be892cc5df6371a0	6,589,872,025,671.012245091	0.6590%

✓ **Pancakeswap holds ~12,4% of the token's supply as liquidity**

✓ **35% tokens are permanently removed from circulation**

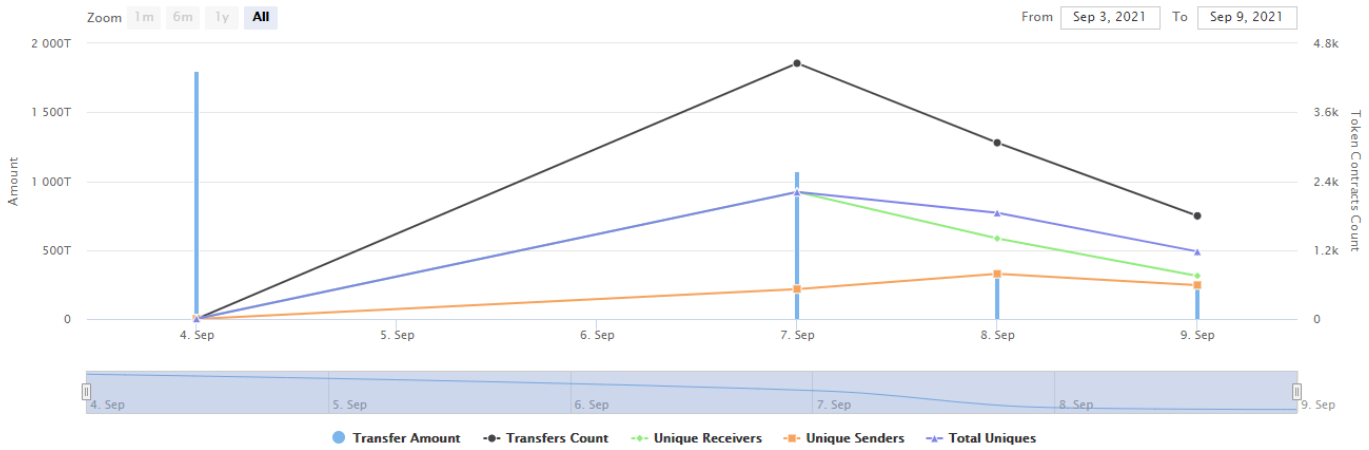
PhoenixChain Top 3 LP Token Holders



Rank	Address	Quantity (Token)	Percentage
1	0x77570a28fdfa3e7c72a3d21948db4de766237925	9,942.199516090607314881	99.5956%
2	0x8cc7bc33f5188b1fb683bedc4dbffa77b136833b	40.36655527455249462	0.4044%
3	0x0000000000000000000000000000000000000000	0.000000000000001	0.0000%

[RISK] 1 wallet have ~99,6% LP tokens

PhoenixChain Contract Interaction Details



4.3 Contract Function Details

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

+ Context
   - [Int] _msgSender
   - [Int] _msgData


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

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

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

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

- [Pub] geUnlockTime
- [Pub] getTime

+ [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] 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 #

+ MiniSports (Context, IERC20, Ownable)
  - [Pub] #
  - [Pub] name
  - [Pub] symbol
  - [Pub] decimals
  - [Pub] totalSupply
  - [Pub] balanceOf
  - [Pub] transfer #
  - [Pub] allowance
  - [Pub] approve #
  - [Pub] transferFrom #
  - [Pub] increaseAllowance #
  - [Pub] decreaseAllowance #
  - [Pub] isExcludedFromReward
  - [Pub] totalFees
  - [Pub] minimumTokensBeforeSwapAmount
  - [Pub] buyBackUpperLimitAmount
  - [Pub] deliver #
  - [Pub] reflectionFromToken
  - [Pub] tokenFromReflection
  - [Pub] excludeFromReward #
    - modifiers: onlyOwner
  - [Ext] includeInReward #
    - modifiers: onlyOwner
  - [Prv] _approve #
  - [Prv] _transfer #
  - [Prv] swapTokens #
    - modifiers: lockTheSwap
  - [Prv] buyBackTokens #
    - modifiers: lockTheSwap
  - [Prv] _swapTokensForEth #
```

- [Prv] _swapETHForTokens #
- [Prv] _addLiquidity #
- [Prv] _tokenTransfer #
- [Prv] _transferStandard #
- [Prv] _transferToExcluded #
- [Prv] _transferFromExcluded #
- [Prv] _transferBothExcluded #
- [Prv] _reflectFee #
- [Prv] _getValues
- [Prv] _getTValues
- [Prv] _getRValues
- [Prv] _getRate
- [Prv] _getCurrentSupply
- [Prv] _takeLiquidity #
- [Prv] calculateTaxFee
- [Prv] calculateLiquidityFee
- [Prv] removeAllFee #
- [Prv] restoreAllFee #
- [Pub] isExcludedFromFee
- [Pub] excludeFromFee #
  - modifiers: onlyOwner
- [Pub] includeInFee #
  - modifiers: onlyOwner
- [Pub] excludeFromSwapAndLiquify #
  - modifiers: onlyOwner
- [Pub] includeFromSwapAndLiquify #
  - modifiers: onlyOwner
- [Prv] _getSellBnBAmount
- [Prv] _removeOldSellHistories #
- [Ext] SetBuyBackMaxTimeForHistories #
  - modifiers: onlyOwner
- [Ext] SetBuyBackDivisor #
  - modifiers: onlyOwner
- [Pub] GetBuyBackTimeInterval
- [Ext] SetBuyBackTimeInterval #
  - modifiers: onlyOwner
- [Ext] SetBuyBackRangeRate #
  - modifiers: onlyOwner
- [Pub] GetSwapMinutes
- [Ext] SetSwapMinutes #
- [Ext] setTaxFeePercent #
  - modifiers: onlyOwner
- [Ext] setBuyFee #
  - modifiers: onlyOwner
- [Ext] setSellFee #
  - modifiers: onlyOwner
- [Ext] setLiquidityFeePercent #
  - modifiers: onlyOwner
- [Ext] setBuyBackSellLimit #
  - modifiers: onlyOwner

- [Ext] setMaxTxAmount #
  - modifiers: onlyOwner
- [Ext] setMarketingDivisor #
  - modifiers: onlyOwner
- [Ext] setNumTokensSellToAddToBuyBack #
  - modifiers: onlyOwner
- [Ext] setMarketingAddress #
  - modifiers: onlyOwner
- [Pub] setSwapAndLiquifyEnabled #
  - modifiers: onlyOwner
- [Pub] setBuyBackEnabled #
  - modifiers: onlyOwner
- [Pub] setAutoBuyBackEnabled #
  - modifiers: onlyOwner
- [Ext] prepareForPreSale #
  - modifiers: onlyOwner
- [Ext] afterPreSale #
  - modifiers: onlyOwner
- [Prv] transferToAddressETH #
- [Pub] getPairAddress
  - modifiers: onlyOwner
- [Pub] changeRouterVersion #
  - modifiers: onlyOwner
- [Ext] <Fallback> $
- [Ext] setAdressFee #
  - modifiers: onlyOwner
- [Ext] setBuyAdressFee #
  - modifiers: onlyOwner
- [Ext] setSellAdressFee #
  - 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 includeInReward uses the loop to find and remove addresses from the _excluded list. It also 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 includeInReward(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is already 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 function _removeOldSellHistories() uses the loop for removing old sell histories. It also could be aborted with out-of-gas exception if there will be a long excluded addresses list.

```
function _removeOldSellHistories() private {
    uint256 i = 0;
    uint256 maxStartTimeForHistories = block.timestamp - _buyBackMaxTimeForHistories;

    for (uint256 j = 0; j < _sellHistories.length; j ++) {

        if (_sellHistories[j].time >= maxStartTimeForHistories) {

            _sellHistories[i].time = _sellHistories[j].time;
            _sellHistories[i].bnbAmount = _sellHistories[j].bnbAmount;

            i = i + 1;
        }
    }

    uint256 removedCnt = _sellHistories.length - i;

    for (uint256 j = 0; j < removedCnt; j ++) {

        _sellHistories.pop();
    }

}
```

- The function _getCurrentSupply() uses the loop for evaluating total supply. It also could be aborted with out-of-gas exception if there will be a long excluded addresses list.

```
function _getCurrentSupply() private view returns(uint256, uint256) {
    uint256 rSupply = _rTotal;
    uint256 tSupply = _tTotal;
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal, _tTotal);
        rSupply = rSupply.sub(_rOwned[_excluded[i]]);
        tSupply = tSupply.sub(_tOwned[_excluded[i]]);
    }
    if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
    return (rSupply, tSupply);
}
```

Recommendation: Consider removing the functions. If this is notdesired, 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 exclude/include accounts from/to transfer fees and reward distribution.

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

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

```
function excludeFromReward(address account) public onlyOwner() {
    // require(account != 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D, 'We can not exclude Pancake router.');
    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 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 has the ability to enable and disable the Buyback functionality.

```
function SetBuyBackMaxTimeForHistories(uint256 newMinutes) external onlyOwner {
    _buyBackMaxTimeForHistories = newMinutes * 1 minutes;
}

function SetBuyBackDivisor(uint256 newDivisor) external onlyOwner {
    _buyBackDivisor = newDivisor;
}
```

```
function SetBuyBackTimeInterval(uint256 newMinutes) external onlyOwner {
    _buyBackTimeInterval = newMinutes * 1 minutes;
}

function SetBuyBackRangeRate(uint256 newPercent) external onlyOwner {
    require(newPercent <= 100, "The value must not be larger than 100.");
    _buyBackRangeRate = newPercent;
}
```

```
function setBuyBackSellLimit(uint256 buyBackSellSetLimit) external onlyOwner {
    buyBackSellLimit = buyBackSellSetLimit;
}
```

```
function setBuyBackEnabled(bool _enabled) public onlyOwner {
    buyBackEnabled = _enabled;
    emit BuyBackEnabledUpdated(_enabled);
}

function setAutoBuyBackEnabled(bool _enabled) public onlyOwner {
    _isAutoBuyBack = _enabled;
    emit AutoBuyBackEnabledUpdated(_enabled);
}
```

- The owner can change the maximal amount per transaction, marketing divisor and marketing address.

```
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner {
    _maxTxAmount = maxTxAmount;
}

function setMarketingDivisor(uint256 divisor) external onlyOwner {
    marketingDivisor = divisor;
}

function setNumTokensSellToAddToBuyBack(uint256 _minimumTokensBeforeSwap) external onlyOwner {
    minimumTokensBeforeSwap = _minimumTokensBeforeSwap;
}

function setMarketingAddress(address _marketingAddress) external onlyOwner {
    marketingAddress = payable(_marketingAddress);
}

function setSwapAndLiquifyEnabled(bool _enabled) public onlyOwner {
    swapAndLiquifyEnabled = _enabled;
    emit SwapAndLiquifyEnabledUpdated(_enabled);
}
```

- The owner can set a ‘Tax fee’ and a ‘Liquidity fee’.

```
function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
    _taxFee = taxFee;
}

function setBuyFee(uint256 buyTaxFee, uint256 buyLiquidityFee) external onlyOwner {
    _buyTaxFee = buyTaxFee;
    _buyLiquidityFee = buyLiquidityFee;
}

function setSellFee(uint256 sellTaxFee, uint256 sellLiquidityFee) external onlyOwner {
    _sellTaxFee = sellTaxFee;
    _sellLiquidityFee = sellLiquidityFee;
}

function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner {
    _liquidityFee = liquidityFee;
}
```

5 Audit Result

LEVEL	ISSUES
Low	**DoS with Block Gas Limit**
Medium	**Owner Privilegies (20)**

1. The contract utilizes SafeMath libraries along with following the ERC20 standard. As the project is deployed with Solidity v0.8.4, it is protected from overflows.

2. There is a 'tax fee' and a 'liquidity fee' on all transactions for any "non-excluded" address that participates in a transfer. The owner has the ability to modify the these fees to any percentage at any time.

3. Users who hold tokens will automatically benefit from the frictionless fee redistribution at the time of each transaction as the tokens collected through the tax fee are removed from the circulating supply.

4. The liquidity fee that is charged on transactions is used to buy BNB via the ‘swaptokens’ function which will be stored in the contract address. Upon each BNB purchase made by the contract address, a percentage (determined by the owner) will be sent to the 'marketing address'. This percentage of the BNB to be transferred can be modified by the owner to any amount at any time.

5. A logical issue exists within the swapping functionality. If the liquidity fee is set (by the owner) to a number that is less than or equal to the 'Marketing Divisor', the resulting BNB to be transferred to the marketing address might be more than the contract balance can support which would result in the transaction reverting. If this functionality is used in such a way, it will allow the project team to receive all of the BNB in the contract address.

6. Each time that a $MiniSports Holder sells tokens to Pancakeswap, the transaction details are stored in the ‘SellHistory’ array which is used to aggregate sell information over a certain period of time which is later used to determine the average sell amount. A portion (determined by the owner) of this average BNB returned per token sale for the buyback period is used to determine the maximum buyback limit.

7. On each transfer that occurs while the minimum threshold (determined by the owner) is met, the protocol will determine an amount of BNB to apply toward buying $MiniSports tokens that will subsequently be burned. The owner has some control over the amount of BNB that is transferred during buybacks as they can update the variables that are used when determining the minimum and maximum buyback ranges. The owner has the ability to enable and disable the Buyback functionality at any time.

8. After the maximum and minimum range has been determined, the contract will pseudo-randomly select a value within the range to apply to the buyback transfer.

9. The contract has built-in checks in place to ensure that the buyback functionality will only run if the contract's BNB balance can support the buyback transfer.

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

11. The owner has the ability to update the address associated with the Pancakeswap router to a new address at any time. The owner can also update the ‘Marketing’ wallet at any time.

12. Ownership has not been renounced.

5.1 Findings Summary

MiniSports

Medium 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.