HACKEN

# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

**Customer**: Bunicorndefi
**Date**: June 11th, 2021

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io



## Document

| | |
|---|---|
| **Name** | Smart Contract Code Review and Security Analysis Report for Bunicordefi. |
| **Approved by** | Andrew Matiukhin \| CTO Hacken OU |
| **Type** | Token, Governance, TimeLock, Defi |
| **Platform** | Ethereum / Solidity |
| **Methods** | Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review |
| **Repository** | https://github.com/bunicorndefi/tokenswap_core<br><br>https://github.com/bunicorndefi/stablecoin_swap |
| **Timeline** | 25 MAY 2021 - 05 JUN 2021 |
| **Changelog** | 05 JUN 2021 - INITIAL AUDIT<br>11 JUN 2021 - SECOND REVIEW |

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Introduction

Hacken OÜ (Consultant) was contracted by Bunicorndefi (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer's smart contract and its code review conducted between May 25th, 2021 - June 05th, 2021. The second code review conducted on June 11th, 2021.

# Scope

The scope of the project is smart contracts in the repository:

```
Repositories:
      https://github.com/bunicorndefi/tokenswap_core
      https://github.com/bunicorndefi/stablecoin_swap
Files:
      BuniCornPool.sol
      BuniCornLibrary.sol
      IBuniCornLiquidityRouter.sol
      MockBuniCornDao.sol
      FeeTo.sol
      VolumeTrendRecorder.sol
      ExampleFlashSwap.sol
      IBuniCornExchangeRouter.sol
      ERC20Permit.sol
      MockBuniCornPair.sol
      IBuniCornRouter02.sol
      WBNB9.sol
      FeeFomula.sol
      IBuniCornDao.sol
      DaoRegistry.sol
      MathExt.sol
      MockVolumeTrendRecorder.sol
      MockBuniCornLibrary.sol
      IBuniCornPool.sol
      IBuniCornFactory.sol
      MockFeeOnTransferERC20.sol
      MockMathExt.sol
      IERC20Permit.sol
      MockERC20Permit.sol
      TestToken.sol
      IBuniCornCallee.sol
      IBuniCornRouter01.sol
      IERC20Metadata.sol
      MockFeeFomula.sol
      IWBNB.sol
```

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

| Category | Check Item |
| --- | --- |
| Code review | ■ Reentrancy |

| | |
|---|---|
| | ▪ Ownership Takeover<br>▪ Timestamp Dependence<br>▪ Gas Limit and Loops<br>▪ DoS with (Unexpected) Throw<br>▪ DoS with Block Gas Limit<br>▪ Transaction-Ordering Dependence<br>▪ Style guide violation<br>▪ Costly Loop<br>▪ ERC20 API violation<br>▪ Unchecked external call<br>▪ Unchecked math<br>▪ Unsafe type inference<br>▪ Implicit visibility level<br>▪ Deployment Consistency<br>▪ Repository Consistency<br>▪ Data Consistency |
| Functional review | ▪ Business Logics Review<br>▪ Functionality Checks<br>▪ Access Control & Authorization<br>▪ Escrow manipulation<br>▪ Token Supply manipulation<br>▪ Assets integrity<br>▪ User Balances manipulation<br>▪ Kill-Switch Mechanism<br>▪ Operation Trails & Event Generation |

## Executive Summary

According to the assessment, the Customer's smart contracts are secured.

| Insecure | Poor secured | Secured | Well-secured |
|---|---|---|---|

You are here

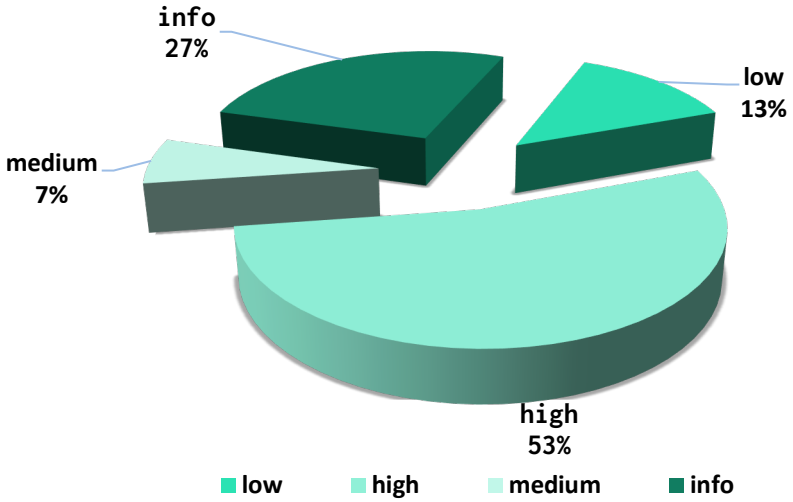
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. A general overview is presented in AS-IS section, and all found issues can be found in the Audit overview section.

Security engineers found **8** High, **1** Medium, **2** Low and **4** Info issues during the audit.

After the **second** review **4** info vulnerabilities were found.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

*Graph 1. The distribution of vulnerabilities after the first review.*



Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

## Severity Definitions

| Risk Level | Description |
|---|---|
| Critical | Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations. |
| High | High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions |
| Medium | Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations. |
| Low | Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution |
| Lowest / Code Style / Best Practice | Lowest-level vulnerabilities, code style violations, and info statements can't affect smart contract execution and can be ignored. |

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# AS-IS overview

## BConst.sol

### Description

BConst contract with constants.

### Imports

BConst has following no imports.

### Inheritance

BConst has no inheritance.

### Usages

BConst contract has no usages.

### Structs

BConst contract has no data structures.

### Enums

BConst contract has no enums.

### Events

BConst contract has no events.

### Modifiers

BConst has no modifiers.

### Fields

BuniToken contract has following fields and constants:

- uint public constant BONE = 10**18;
- uint public constant MIN_BOUND_TOKENS = 2;
- uint public constant MAX_BOUND_TOKENS = 8;
- uint public constant MIN_FEE = BONE / 10**6;
- uint public constant MAX_FEE = BONE / 10;
- uint public constant EXIT_FEE = 0;
- uint public constant MIN_WEIGHT = BONE;
- uint public constant MAX_WEIGHT = BONE * 50;

- uint public constant MAX_TOTAL_WEIGHT = BONE * 50;
- uint public constant MIN_BALANCE = BONE / 10**12;
- uint public constant INIT_POOL_SUPPLY = BONE * 100;
- uint public constant MIN_BPOW_BASE = 1 wei;
- uint public constant MAX_BPOW_BASE = (2 * BONE) - 1 wei;
- uint public constant BPOW_PRECISION = BONE / 10**10;
- uint public constant MAX_IN_RATIO = BONE / 2;
- uint public constant MAX_OUT_RATIO = (BONE / 3) + 1 wei;

**Functions**
BuniToken has no public functions.

## BuniConstants.sol

**Description**

BuniConstants contract with constants.

**Imports**

BuniConstants has following no imports.

**Inheritance**

BuniConstants has no inheritance.

**Usages**

BuniConstants contract has no usages.

**Structs**

BuniConstants contract has no data structures.

**Enums**

BuniConstants contract has no enums.

**Events**

BuniConstants contract has no events.

**Modifiers**

BuniConstants has no modifiers.

**Fields**

BuniConstants contract has following fields and constants:

- uint public constant BONE = 10**18;
- uint public constant MIN_WEIGHT = BONE;
- uint public constant MAX_WEIGHT = BONE * 50;
- uint public constant MAX_TOTAL_WEIGHT = BONE * 50;
- uint public constant MIN_BALANCE = BONE / 10**6;
- uint public constant MAX_BALANCE = BONE * 10**12;
- uint public constant MIN_POOL_SUPPLY = BONE * 100;
- uint public constant MAX_POOL_SUPPLY = BONE * 10**9;
- uint public constant MIN_FEE = BONE / 10**6;
- uint public constant MAX_FEE = BONE / 10;
- uint public constant EXIT_FEE = 0;
- uint public constant MAX_IN_RATIO = BONE / 2;
- uint public constant MAX_OUT_RATIO = (BONE / 3) + 1 wei;
- uint public constant MIN_ASSET_LIMIT = 2;
- uint public constant MAX_ASSET_LIMIT = 8;
- uint public constant MAX_UINT = uint(-1);

**Functions**
BuniConstants has no public functions.

## BPool.sol

**Description**

BPool is a liquidity pool with rewards.

**Imports**

BPool has following imports:

- BToken.sol;
- BMath.sol;

**Inheritance**

BPool is BToken and BMath.

**Usages**

BPool contract has no usages.

**Structs**

BPool contract has following data structures:

- Record

**Enums**

BPool contract has no enums.

**Events**

BPool contract has following events:

- LOG_SWAP
- LOG_JOIN
- LOG_EXIT
- LOG_CALL

**Modifiers**

BPool has following custom modifiers:

- _logs_
- _lock_
- _viewlock_

**Fields**

BPool contract has following fields and constants:

- bool private _mutex;
- address private _factory;
- address private _controller;
- bool private _publicSwap;
- uint private _swapFee;
- bool private _finalized;
- address[] private _tokens;
- mapping(address=>Record) private  _records;
- uint private _totalWeight;

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

**Functions**
BPool has following public functions:

- *constructor*
- *isPublicSwap*
- *isFinalized*
- *isBound*
- *getNumTokens*
- *getCurrentTokens*
- *getFinalTokens*
- *getDenormalizedWeight*
- *getTotalDenormalizedWeight*
- *getNormalizedWeight*
- *getBalance*
- *getSwapFee*
- *getController*
- *setSwapFee*
- *setController*
- *setPublicSwap*
- *finalize*
- *bind*
- *rebind*
- *unbind*
- *gulp*
- *getSpotPrice*
- *getSpotPriceSansFee*
- *joinPool*
- *exitPool*
- *swapExactAmountIn*
- *swapExactAmountOut*
- *joinswapExternAmountIn*

- *joinswapPoolAmountOut*
- *exitswapPoolAmountIn*
- *exitswapExternAmountOut*

## BuniCornPool.sol

### Description

BuniCornPool is a liquidity pool with rewards.

### Imports

BuniCornPool contract has following imports:

- "@openzeppelin/contracts/math/SafeMath.sol";
- "@openzeppelin/contracts/math/Math.sol";
- "@openzeppelin/contracts/utils/ReentrancyGuard.sol";
- "@openzeppelin/contracts/token/ERC20/SafeERC20.sol";
- "./libraries/MathExt.sol";
- "./libraries/FeeFomula.sol";
- "./libraries/ERC20Permit.sol";
- "./interfaces/IBuniCornFactory.sol";
- "./interfaces/IBuniCornCallee.sol";
- "./interfaces/IBuniCornPool.sol";
- "./interfaces/IERC20Metadata.sol";
- "./VolumeTrendRecorder.sol";

### Inheritance

BuniCornPool contract is:

- IBuniCornPool,
- ERC20Permit,
- ReentrancyGuard,
- VolumeTrendRecorder,

### Usages

BuniCornPool contract has following custom usages:

- using SafeMath for uint256;

- using SafeERC20 for IERC20;

**Structs**

BuniCornPool contract has following data structures:

- ReserveData.

**Enums**

BuniCornPool contract has no custom enums.

**Events**

BuniCornPool contract has following custom evets:

- Mint
- Burn
- Swap
- Sync

**Modifiers**

BuniCornPool has no custom modifiers.

**Fields**

BuniCornPool contract has following fields and constants:

- uint256 internal constant MAX_UINT112 = 2**112 - 1;
- uint256 internal constant BPS = 10000;
- uint256 public constant MINIMUM_LIQUIDITY = 10**3;
- IBuniCornFactory public override factory;
- IERC20 public override token0;
- IERC20 public override token1;
- uint112 internal reserve0;
- uint112 internal reserve1;
- uint32 public override ampBps
- uint112 internal vReserve0;
- uint112 internal vReserve1;
- uint256 public override kLast;

**Functions**

BuniCornPool has following public functions:

- constructor
- initialize
- mint
- burn
- swap
- skim
- sync
- getTradeInfo
- getReserves
- name
- symbol

## BFactory.sol

### Description

BFactory - pool factory

### Imports

BFactory contract has following imports:

- ./BPool.sol;

### Inheritance

BFactory contract has no inheritance.

### Usages

BFactory contract has no custom usages.

### Structs

BFactory contract has no custom data structures.

### Enums

BFactory contract has no custom enums.

### Events

BFactory contract has following custom evets:

- LOG_NEW_POOL
- LOG_BLABS

**Modifiers**

BFactory has no custom modifiers.

**Fields**

BFactory contract has following fields and constants:

- mapping(address=>bool) private _isBPool;
- address private _blabs.

**Functions**

BFactory has following public functions:

- constructor
- isBPool
- newBPool
- getBLabs
- setBLabs
- collect

## BuniCornRouter02.sol

**Description**

BuniCornRouter02 - funds router

**Imports**

BuniCornRouter02 contract has following imports:

- @uniswap/lib/contracts/libraries/TransferHelper.sol;
- @openzeppelin/contracts/access/Ownable.sol;
- @openzeppelin/contracts/math/SafeMath.sol;
- @openzeppelin/contracts/token/ERC20/SafeERC20.sol;
- ../interfaces/IBuniCornFactory.sol;

- ../interfaces/IBuniCornRouter02.sol;
- ../interfaces/IERC20Permit.sol;
- ../interfaces/IBuniCornPool.sol;
- ../interfaces/IWETH.sol;
- ../libraries/BuniCornLibrary.sol;

**Inheritance**

BuniCornRouter02 contract is:

- Ownable
- IBuniCornRouter02,

**Usages**

BuniCornRouter02 contract has following custom usages:

- using SafeERC20 for IERC20;
- using SafeERC20 for IWETH;
- using SafeMath for uint256;

**Structs**

BuniCornRouter02 contract has no custom data structures

**Enums**

BuniCornRouter02 contract has no custom enums.

**Events**

BuniCornRouter02 contract has no custom evets.

**Modifiers**

BuniCornRouter02 has following custom modifiers:

- ensure

**Fields**

BuniCornRouter02 contract has following fields and constants:

- uint256 internal constant BPS = 10000;
- address public immutable override factory;

- IWETH public immutable override weth;

**Functions**

BuniCornRouter02 has following public functions:

- constructor
- receive
- addLiquidity
- addLiquidityBNB
- addLiquidityNewPool
- addLiquidityNewPoolBNB
- removeLiquidity
- removeLiquidityBNB
- removeLiquidityWithPermit
- removeLiquidityBNBWithPermit
- removeLiquidityBNBSupportingFeeOnTransferTokens
- removeLiquidityBNBWithPermitSupportingFeeOnTransferTokens
- swapExactTokensForTokens
- swapTokensForExactTokens
- swapExactBNBForTokens
- swapTokensForExactBNB
- swapExactTokensForBNB
- swapBNBForExactTokens
- swapExactTokensForTokensSupportingFeeOnTransferTokens
- swapExactBNBForTokensSupportingFeeOnTransferTokens
- swapExactTokensForBNBSupportingFeeOnTransferTokens
- quote
- getAmountsOut
- getAmountsIn
- verifyPoolsPathSwap

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Audit overview

## Critical

No critical issues were found.

## High

1. BuniCornPool.mint function can be called by anyone, anytime and execute funds operations. If the attacker catches the right time, he will be able to mint someone's coins to address he wants.

   **Customer accepts this risk**

   **Customer notice:** "We use a modifier called nonReentrant. nonReentrant is understood as a lock, when an address calls to the smart contract, the lock will lock it. Thus, no one can execute any function in the pool while it is locked."

2. BuniCornPool.burn function can be called by anyone, anytime and execute funds operations. If the attacker catches the right time, he will be able to receive someone's coins to his address.

   **Customer accepts this risk**

3. BuniCornPool.swap can be called by anyone, anytime and has no user balance check. Also in this function the line *IBuniCornCallee(to).buniSwapCall();* will be executed only if calldata > 0 and this is a place where we have any interaction with a sender.

   **Customer accepts this risk**

4. BuniCornPool.skim: anyone can call and just move funds if there is any.

   **Fixed before the second audit**

5. BuniCornRouter02.addLiquidityBNB is public but it should be restricted from the strangers' calls. Unsafe finance operations.

   **Fixed before the second audit**

6. BuniCornRouter02.addLiquidity is public but should be restricted from the strangers' calls. Unsafe finance operations.

**Fixed before the second audit**

7. BuniCornRouter02.removeLiquidityBNBSupportingFeeOnTransferToken function sends all of the existed funds to the router's address to contract caller.

   **Fixed before the second audit**

8. BuniCornRouter02.removeLiquidity should be private. This is an internal function, designed to be a part of another functions. But the issue is that this function does not count how many liquidities contract receive from a caller. So, if there is some extra balance on the contract, the exchange will lose it.

   **Customer accepts this risk**

   **Customer notice:** "In case a liquidity provider approved the router address to burn their LP token, we use this function in our interface to remove liquidity from the pool. So, we think it should be a public function."

## Medium

BuniCornPool: changes all of the math operations to SafeMath lib usage.

**Fixed before the second audit**

## Low

1. Anyone can call the BFactory function newBPool(), create a new pool and start being the pool owner.

   **Fixed before the second audit**

2. BuniCornPool.swap function and others should be checked for the 0x0 address.

   **Fixed before the second audit**

## Lowest / Code style / Best Practice

1. There is a duplication of the constant contracts: BConst & BuniConstants. It is better to unify the constant usages.
2. We strongly recommend using a newer version of the solidity compiler.
3. Use modifiers to avoid duplicate validations like:

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

a. msg.sender == _controller and !_finalized,

b. "ERR_IS_FINALIZED"

in BPool contract.

4. BuniCornPool: should move initialize() function code to the constructor code-block.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

## Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, high-level description of functionality was presented in As-Is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security engineers found **8** High, **1** Medium, **2** Low and **4** Info issues during the audit.

After the **second** review **4** info vulnerabilities were found.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Disclaimers

## Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.

## Technical Disclaimer

Smart contracts are deployed and executed on blockchain platform. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.