HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Bunicorndefi
Date: May 03th, 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 Bunicorndefi
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/farming https://github.com/bunicorndefi/stablecoin_swap
Commit	
Deployed contract	
Timeline	21 APR 2021 - 26 APR 2021
Changelog	26 APR 2021 - INITIAL AUDIT 03 MAY - SECONDARY AUDIT

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 April 21th, 2021 - April 26th, 2021.

The secondary review conducted on May 03th, 2021.

Scope

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

```
Contract deployment address:
Repository
File:
      BuniToken.sol
      MasterChef.sol
      VBuniToken.sol
      BuniCornFactory.sol
      BuniCornRouter02.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 ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data Consistency

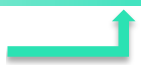
Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Functional review	■ Business Logics Review ■ Functionality Checks ■ Access Control & Authorization ■ Escrow manipulation ■ Token Supply manipulation ■ Assets integrity ■ User Balances manipulation ■ Kill-Switch Mechanism ■ Operation Trails & Event Generation

Executive Summary

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

Insecure	Poor secured	Secured	Well-secured



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.

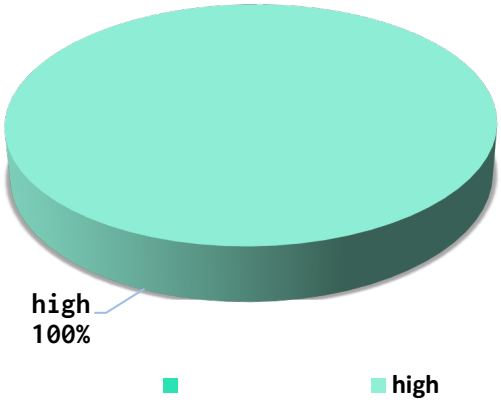
Security engineers found **1** high issue during the audit.

After the **second** review no vulnerabilities were found.

Notice: the audit scope is limited and not include all files in the repository. Though, reviewed contracts are secure, we may not guarantee secureness of contracts that are not in the scope.

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.

AS-IS overview

BuniToken.sol

Description

BuniToken token with governance.

Imports

BuniToken has following imports:

- BEP20.sol

Inheritance

BuniToken inherit:

- BEP20

Usages

BuniToken contract has no usages.

Structs

BuniToken contract has following data structures:

- Checkpoint

Enums

BuniToken contract has no enums.

Events

BuniToken contract has following events:

- DelegateChanged
- DelegateVotesChanged

Modifiers

BuniToken has no modifiers.

Fields

BuniToken contract has following fields and constants:

- mapping (address => address) internal _delegates;

- mapping (address => mapping (uint32 => Checkpoint)) public checkpoints;
- mapping (address => uint32) public numCheckpoints;
- bytes32 public constant DOMAIN_TYPEHASH = keccak256(EIP712Domain(string name,uint256 chainId,address verifyingContract));
- bytes32 public constant DELEGATION_TYPEHASH = keccak256(Delegation(address delegatee,uint256 nonce,uint256 expiry));
- mapping (address => uint) public nonces;

Functions
BuniToken has following public functions:

- mint
- delegates
- delegate
- delegateBySig
- getCurrentVotes
- getPriorVotes

MasterChef.sol

Description

MasterChef is a liquidity pool with rewards.

Imports

MasterChef has following imports:

- SafeMath.sol
- IBEP20.sol
- SafeBEP20.sol
- Ownable.sol
- interfaces/IERC721.sol
- BuniToken.sol

Inheritance

MasterChef is Ownable.

Usages

MasterChef contract has following usages:

- SafeMath for uint256
- SafeBEP20 for IBEP20

Structs

MasterChef contract has following data structures:

- UserInfo
- PoolInfo

Enums

MasterChef contract has no enums.

Events

MasterChef contract has following events:

- Deposit
- Withdraw
- Harvest
- Vesting
- EmergencyWithdraw

Modifiers

MasterChef has no custom modifiers.

Fields

MasterChef contract has following fields and constants:

- BuniToken public buni;
- IERC721 public vBuni;
- address public devaddr;
- uint256 public buniPerBlock;
- uint256 public BONUS_MULTIPLIER = 1;
- IMigratorChef public migrator;
- PoolInfo[] public poolInfo;
- mapping (uint256 => mapping (address => UserInfo)) public userInfo;
- uint256 public totalAllocPoint = 0;

- uint256 public startBlock;
- uint256 public platformFeeRate = 10;
- uint256 public withdrawDecimals = 3;
- uint256 public vestTimeLock = 30 days;
- uint256 public penaltyTime = 7 days;
- mapping (uint256 => mapping(address => uint256)) unclaimedBuni;

Functions
MasterChef has following public functions:

- constructor
- poolLength
- getMultiplier
- pendingBuni
- getWithdrawFee
- updateMultiplier
- add
- set
- setMigrator
- setTimeLock
- setPenaltyTime
- migrate
- massUpdatePools
- updatePool
- deposit
- withdraw
- harvest
- dev
- redeemBuni
- redeemBatchBuni

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

VBuniToken.sol

Description

VBuniToken is a token with ability for holders to burn (destroy) their tokens, a minter role that allows for token minting, a pauser role that allows to stop all token transfers, token ID and URI autogeneration.

Imports

VBuniToken contract has following imports:

- @openzeppelin/contracts/access/AccessControl.sol;
- @openzeppelin/contracts/utils/Context.sol;
- @openzeppelin/contracts/utils/Counters.sol;
- @openzeppelin/contracts/token/ERC721/ERC721.sol;
- @openzeppelin/contracts/token/ERC721/ERC721Burnable.sol;
- @openzeppelin/contracts/token/ERC721/ERC721Pausable.sol;

Inheritance

VBuniToken contract is:

- Context,
- AccessControl,
- ERC721Burnable,
- ERC721Pausable

Usages

VBuniToken contract has following custom usages:

- Counters for Counters.Counter;

Structs

VBuniToken contract has following data structures:

- TokenInfo.

Enums

VBuniToken contract has no custom enums.

Events

VBuniToken contract has no custom evets.

Modifiers

VBuniToken has no custom modifiers.

Fields

VBuniToken contract has following fields and constants:

- bytes32 public constant MINTER_ROLE = keccak256(MINTER_ROLE);
- bytes32 public constant PAUSER_ROLE = keccak256(PAUSER_ROLE);
- mapping(uint256 => TokenInfo) public vestedData;
- Counters.Counter private _tokenIdTracker;

Functions

VBuniToken has following public functions:

- constructor
- getTokenInfo
- mint
- pause
- unpause
- setBaseURI
- getTokenInfoOfOwnerByIndex

BuniCornFactory.sol

Description

BuniCornFactory

Imports

BuniCornFactory contract has following imports:

- @openzeppelin/contracts/access/Ownable.sol;
- @openzeppelin/contracts/utils/EnumerableSet.sol;
- ./interfaces/IBuniCornFactory.sol;

- ./BuniCornPool.sol;

Inheritance

BuniCornFactory contract is:

- Ownable,
- IBuniCornFactory

Usages

BuniCornFactory contract has following custom usages:

- EnumerableSet for EnumerableSet.AddressSet;

Structs

BuniCornFactory contract has no custom data structures

Enums

BuniCornFactory contract has no custom enums.

Events

BuniCornFactory contract has following custom evets:

- PoolCreated
- SetFeeConfiguration
- SetFeeToSetter

Modifiers

BuniCornFactory has no custom modifiers.

Fields

BuniCornFactory contract has following fields and constants:

- uint256 internal constant BPS = 10000;
- address private feeTo;
- uint16 private governmentFeeBps;
- address public override feeToSetter;
- mapping(IERC20 => mapping(IERC20 => EnumerableSet.AddressSet)) internal tokenPools;

- mapping(IERC20 => mapping(IERC20 => address)) public override getUnamplifiedPool;
- address[] public override allPools;
- address public routerAddress;

Functions

BuniCornFactory has following public functions:

- constructor
- setRouter
- createPool
- setFeeConfiguration
- setFeeToSetter
- getFeeConfiguration
- allPoolsLength
- getPools
- getPoolsLength
- getPoolAtIndex
- isPool

BuniCornRouter02.sol

Description

BuniCornRouter02

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
- addLiquidityETH
- addLiquidityNewPool
- addLiquidityNewPoolETH
- removeLiquidity
- removeLiquidityETH
- removeLiquidityWithPermit
- removeLiquidityETHWithPermit
- removeLiquidityETHSupportingFeeOnTransferTokens
- removeLiquidityETHWithPermitSupportingFeeOnTransferTokens
- swapExactTokensForTokens
- swapTokensForExactTokens
- swapExactETHForTokens
- swapTokensForExactETH
- swapExactTokensForETH
- swapETHForExactTokens
- swapExactTokensForTokensSupportingFeeOnTransferTokens
- swapExactETHForTokensSupportingFeeOnTransferTokens
- swapExactTokensForETHSupportingFeeOnTransferTokens
- quote
- getAmountsOut
- getAmountsIn

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

Audit overview

Critical

No critical issues were found.

High

BuniCornRouter02.sol has a public function `removeLiquidityETH` which has no `msg.sender` validation and can send funds to any address received in function params.

Fixed before the second audit.

Medium

No medium issues were found.

Low

No low severity issues were found.

Lowest / Code style / Best Practice

No lowest severity issues were found.

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 **1** high issue during the audit.

After the **second** review no vulnerabilities were found.

Notice: the audit scope is limited and not include all files in the repository. Though, reviewed contracts are secure, we may not guarantee secureness of contracts that are not in the scope.

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.