HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: JulSwap
Date: February 25th, 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 JulSwap.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Token Swap
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	
Commit	
Deployed contract	
Timeline	22 FEB 2021 - 25 FEB 2021
Changelog	25 FEB 2021 - INITIAL 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 JulSwap (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 February 22nd, 2021 - February 25th, 2021.

Scope

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

```
Contract deployment address:
Repository
File:
      BSCswapBEP20.sol
      BSCswapFactory.sol
      BSCswapPair.sol
      BSCswapRouter.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.

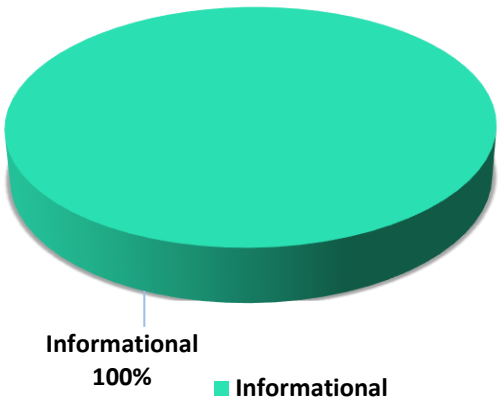
Insecure	Poor secured	Secured	Well-secured

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 **1** informational issue during the audit.

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

BSCswapFactory.sol

Description

BSCswapFactory is a contract for creating liquidity pairs.

Imports

BSCswapFactory contract has following imports:

- IBSCswapFactory.sol - an interface from the project files.
- BSCswapPair.sol - a contract from the project files.

Inheritance

BSCswapFactory is IBSCswapFactory.

Usages

BSCswapFactory contract has no custom usages.

Structs

BSCswapFactory contract has no data structures.

Enums

BSCswapFactory contract has no enums.

Events

BSCswapFactory contract has following events:

- event PairCreated(address indexed token0, address indexed token1, address pair, uint);

Modifiers

BSCswapFactory has no modifiers.

Fields

BSCswapFactory contract has following fields and constants:

- address public override feeTo

- address public override feeToSetter
- address public override migrator
- mapping(address => mapping(address => address)) public override getPair
- address[] public override allPairs

State-Changing Functions

BSCswapFactory contract has following state-changing functions:

- *constructor*

  #### Description

  Initializes the contract. Sets feeToSetter address.

  #### Visibility

  public

  #### Input parameters

  - address _feeToSetter

  #### Constraints

  None

  #### Events emit

  None

  #### Output

  None

- *createPair*

  #### Description

  Creates a pair for tokenA and tokenB if one doesn’t exist already.

  #### Visibility

  external

  #### Input parameters

- address tokenA
- address tokenB

Constraints

- tokenA and tokenB must not be equal.
- tokenA and/or tokenB must not be zero.
- The pair must not exist yet.

Events emit

Emits the PairCreated event.

Output

- address pair

- *setFeeTo*

  Description

  Used to set the feeTo address.

  Visibility

  external

  Input parameters

  - address _feeTo

  Constraints

  - The msg.sender address must be equal to the feeToSetter address.

  Events emit

  None

  Output

  None

- *setMigrator*

  Description

  Used to set the migrator contract address.

Visibility

external

Input parameters

- address _migrator

Constraints

- The msg.sender address must be equal to the feeToSetter address.

Events emit

None

Output

None

- *setFeeToSetter*

Description

Used to set the feeToSetter address.

Visibility

external

Input parameters

- address _feeToSetter

Constraints

- The msg.sender address must be equal to the feeToSetter address.

Events emit

None

Output

None

Read-Only Functions

BSCswapFactory contract has following read-only functions:

- function allPairsLength() external override view returns (uint);
- function pairCodeHash() external pure returns (bytes32);

BSCswapPair.sol

Description

BSCswapPair is a liquidity pair contract.

Imports

BSCswapPair contract has following imports:

- BSCswapBEP20.sol - from the project files.
- Math.sol - from the project files.
- UQ112x112.sol - from the project files.
- IBEP20.sol - from the project files.
- IBSCswapFactory.sol - from the project files.
- IBSCswapCallee.sol - from the project files.
- BSCswapPair defines IMigrator interface that has following functions:
  - function desiredLiquidity() external view returns (uint256);

Inheritance

BSCswapPair is BSCswapBEP20.

Usages

BSCswapPair contract has following usages:

- SafeMathBSCswap for uint;
- UQ112x112 for uint224;

Structs

BSCswapPair contract has no data structures.

Enums

BSCswapPair contract has no enums.

Events

BSCswapPair contract has following events:

- event Mint(address indexed sender, uint amount0, uint amount1);
- event Burn(address indexed sender, uint amount0, uint amount1, address indexed to);
- event Swap(address indexed sender, uint amount0In, uint amount1In, uint amount0Out, uint amount1Out, address indexed to);
- event Sync(uint112 reserve0, uint112 reserve1);

Modifiers

BSCswapPair has following modifiers:

- modifier lock();

Fields

BSCswapPair contract has following fields and constants:

- uint public constant MINIMUM_LIQUIDITY = 10**3;
- bytes4 private constant SELECTOR = bytes4(keccak256(bytes('transfer(address,uint256)')));
- address public factory;
- address public token0;
- address public token1;
- uint112 private reserve0;
- uint112 private reserve1;
- uint32 private blockTimestampLast;
- uint public price0CumulativeLast;
- uint public price1CumulativeLast;
- uint public kLast;
- uint private unlocked = 1;

State-Changing Functions

BSCswapPair contract has following state-changing functions:

- *constructor*

  #### Description

  Initializes the contract. Sets factory address.

Visibility

public

Input parameters

None

Constraints

None

Events emit

None

Output

None

- *initialize*

  Description

  Sets tokens addresses.

  Visibility

  external

  Input parameters

  None

  Constraints

  - The msg.sender must be the factory address.

  Events emit

  None

  Output

  None

- *mint*

Description

Creates pool tokens.

Visibility

external

Input parameters

- address to

Constraints

- lock modifier.
- Liquidity must be greater than 0.

Events emit

Emits the Mint event.

Output

- uint liquidity

- *burn*

Description

Destroys pool tokens.

Visibility

external

Input parameters

- address to

Constraints

- lock modifier.
- Sufficient liquidity must be burned.

Events emit

Emits the Burn event.

Output

    - uint amount0
    - uint amount1
- *swap*

  Description

  Swaps tokens.

  Visibility

  external

  Input parameters

    - uint amount0Out
    - uint amount1Out
    - address to
    - bytes calldata data

  Constraints

    - lock modifier.
    - Output amount must be sufficient.
    - Address to must be valid.
    - Input amount must be sufficient.

  Events emit

  Emits the Swap event.

  Output

  None

- *skim*

  Description

  Used to force balances to match reserves.

  Visibility

  external

  Input parameters

    - address to

Constraints

- lock modifier.

Events emit

None

Output

None

- *sync*

  Description

  Used to force reserves to match balances.

  Visibility

  external

  Input parameters

  None

  Constraints

  - lock modifier.

  Events emit

  None

  Output

  None

Read-Only and Non-Public Functions

BSCswapPair contract has following read-only and non-public functions:

- function getReserves() public view returns (uint112 _reserve0, uint112 _reserve1, uint32 _blockTimestampLast);
- function _safeTransfer(address token, address to, uint value) private;

- function _update(uint balance0, uint balance1, uint112 _reserve0, uint112 _reserve1) private;
- function _mintFee(uint112 _reserve0, uint112 _reserve1) private returns (bool feeOn);

BSCswapBEP20.sol

Description

BSCswapBEP20 is BEP-20 pool token contract.

Imports

BSCswapBEP20 contract has following imports:

- SafeMath.sol - from the project files.

Inheritance

BSCswapBEP20 inherits nothing.

Usages

BSCswapBEP20 contract has following usages:

- SafeMathBSCswap for uint;

Structs

BSCswapBEP20 contract has no data structures.

Enums

BSCswapBEP20 contract has no enums.

Events

BSCswapBEP20 contract has following events:

- event Approval(address indexed owner, address indexed spender, uint value);
- event Transfer(address indexed from, address indexed to, uint value);

Modifiers

BSCswapBEP20 has no modifiers.

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

Fields

BSCswapBEP20 contract has following fields and constants:

- string public constant name = 'SwapLiquidity LP Token';
- string public constant symbol = 'SLP';
- uint8 public constant decimals = 18;
- uint public totalSupply;
- mapping(address => uint) public balanceOf;
- mapping(address => mapping(address => uint)) public allowance;
- bytes32 public DOMAIN_SEPARATOR;
- bytes32 public constant PERMIT_TYPEHASH = 0x6e71edae12b1b97f4d1f60370fef10105fa2faae0126114a169c64845d6126c9;
- mapping(address => uint) public nonces;

State-Changing Functions

BSCswapBEP20 contract has following state-changing functions:

- *constructor*

  #### Description

  Initializes the contract.

  #### Visibility

  public

  #### Input parameters

  None

  #### Constraints

  None

  #### Events emit

  None

  #### Output

  None

- *approve*

  Description

  Lets msg.sender set their allowance for a spender.

  Visibility

  external

  Input parameters

  - address spender
  - uint value

  Constraints

  None

  Events emit

  None

  Output

  - bool

- *transfer*

  Description

  Lets msg.sender send pool tokens to an address.

  Visibility

  external

  Input parameters

  - address to
  - uint value

  Constraints

  None

  Events emit

  None

Output

- bool

- *transferFrom*

  Description

  Sends pool tokens from one address to another.

  Visibility

  external

  Input parameters

  - address from
  - address to
  - uint value

  Constraints

  None

  Events emit

  None

  Output

  - bool

- *permit*

  Description

  Sets the allowance for a spender where approval is granted via a signature.

  Visibility

  external

  Input parameters

  - address owner
  - address spender
  - uint value
  - uint deadline
  - uint8 v

- bytes32 r
- bytes32 s

Constraints

- The deadline has not expired yet.
- The signature must be valid.

Events emit

None

Output

None

Read-Only and Non-Public Functions

BSCswapBEP20 contract has following read-only and non-public functions:

- function _mint(address to, uint value) internal;
- function _burn(address from, uint value) internal;
- function _approve(address owner, address spender, uint value) private;
- function _transfer(address from, address to, uint value) private;

BSCswapRouter.sol

Description

BSCswapRouter is swap contract.

Imports

BSCswapRouter contract has following imports:

- BSCswapLibrary.sol - from the project files.
- SafeMath.sol - from the project files.
- TransferHelper.sol - from the project files.
- IBSCswapRouter02.sol - from the project files.
- IBSCswapFactory.sol - from the project files.
- IBEP20.sol - from the project files.
- IWBNB.sol - from the project files.

Inheritance

BSCswapRouter is IBSCswapRouter02.

Usages

BSCswapRouter contract has following usages:

- SafeMathBSCswap for uint;

Structs

BSCswapRouter contract has no data structures.

Enums

BSCswapRouter contract has no enums.

Events

BSCswapRouter contract has no custom events.

Modifiers

BSCswapRouter has following modifiers:

- modifier ensure(uint deadline)

Fields

BSCswapRouter contract has following fields and constants:

- address public immutable override factory;
- address public immutable override WBNB;

State-Changing Functions

BSCswapRouter contract has following state-changing functions:

- *constructor*

  #### Description

  Initializes the contract.

  #### Visibility

  public

  #### Input parameters

- address _factory
- address _WBNB

Constraints

None

Events emit

None

Output

None

- *receive*

  Description

  Used to accept BNB.

  Visibility

  external payable

  Input parameters

  None

  Constraints

  - Only accept BNB via fallback from the WBNB contract.

  Events emit

  None

  Output

  None

- *addLiquidity*

  Description

  Adds liquidity to a pool.

  Visibility

external

Input parameters

- address tokenA
- address tokenB
- uint amountADesired
- uint amountBDesired
- uint amountAMin
- uint amountBMin
- address to
- uint deadline

Constraints

- ensure(deadline) modifier.

Events emit

None

Output

- uint amountA
- uint amountB
- uint liquidity

- ***addLiquidityBNB***

Description

Adds liquidity to a pool with BNB.

Visibility

external payable

Input parameters

- address token
- uint amountTokenDesired
- uint amountTokenMin
- uint amountBNBMin
- address to
- uint deadline

Constraints

- ensure(deadline) modifier.
- BNB transferred successfully.

Events emit

None

Output

- uint amountToken
- uint amountBNB
- uint liquidity

- ***removeLiquidity***

Description

Removes liquidity from a pool.

Visibility

external

Input parameters

- address tokenA
- address tokenB
- uint liquidity
- uint amountAMin
- uint amountBMin
- address to
- uint deadline

Constraints

- ensure(deadline) modifier.
- The amount of tokenA must be sufficient.
- The amount of tokenB must be sufficient.

Events emit

None

Output

- uint amountA
- uint amountB

- ***removeLiquidityBNB***

Description

Removes liquidity from an pool and receive BNB.

Visibility

external

Input parameters

- address token
- uint liquidity
- uint amountTokenMin
- uint amountBNBMin
- address to
- uint deadline

Constraints

- ensure(deadline) modifier.

Events emit

None

Output

- uint amountToken
- uint amountBNB

- *removeLiquidityWithPermit*

Description

Removes liquidity from a pool without pre-approval.

Visibility

external

Input parameters

- address tokenA
- address tokenB
- uint liquidity
- uint amountAMin
- uint amountBMin
- address to

- uint deadline
- bool approveMax
- uint8 v
- bytes32 r
- bytes32 s

Constraints

None

Events emit

None

Output

- uint amountA
- uint amountB

- *removeLiquidityBNBWithPermit*

Description

Removes liquidity from a pool and receive BNB without pre-approval.

Visibility

external

Input parameters

- address token
- uint liquidity
- uint amountTokenMin
- uint amountBNBMin
- address to
- uint deadline
- bool approveMax
- uint8 v
- bytes32 r
- bytes32 s

Constraints

None

Events emit

None

Output

- uint amountToken
- uint amountBNB

- *removeLiquidityBNBSupportingFeeOnTransferTokens*

Description

Identical to removeLiquidityBNB, but succeeds for tokens that take a fee on transfer.

Visibility

external

Input parameters

- address token
- uint liquidity
- uint amountTokenMin
- uint amountBNBMin
- address to
- uint deadline

Constraints

- ensure(deadline) modifier.

Events emit

None

Output

- uint amountBNB

- *removeLiquidityBNBWithPermitSupportingFeeOnTransferTokens*

Description

Identical to removeLiquidityBNBWithPermit, but succeeds for tokens that take a fee on transfer.

Visibility

external

Input parameters

- address token
- uint liquidity
- uint amountTokenMin
- uint amountBNBMin
- address to
- uint deadline
- bool approveMax
- uint8 v
- bytes32 r
- bytes32 s

Constraints

None

Events emit

None

Output

- uint amountBNB

- *swapExactTokensForTokens*

Description

Swaps an exact amount of input tokens for as many output tokens as possible.

Visibility

external

Input parameters

- uint amountIn
- uint amountOutMin
- address[] calldata path
- address to
- uint deadline

Constraints

- ensure(deadline) modifier.
- Output amount must be sufficient.

Events emit

None

Output

- uint[] memory amounts

- *swapTokensForExactTokens*

Description

Receive an exact amount of output tokens for as few input tokens as possible.

Visibility

external

Input parameters

- uint amountOut
- uint amountInMax
- address[] calldata path
- address to
- uint deadline

Constraints

- ensure(deadline) modifier.
- Input amount must not be excessive.

Events emit

None

Output

- uint[] memory amounts

- *swapExactBNBForTokens*

Description

Swaps an exact amount of BNB for as many output tokens as possible.

Visibility

external payable

Input parameters

- uint amountOutMin
- address[] calldata path
- address to
- uint deadline

Constraints

- ensure(deadline) modifier.
- The path must be valid.
- Output amount must be sufficient.

Events emit

None

Output

- uint[] memory amounts

- *swapTokensForExactBNB*

Description

Receive an exact amount of BNB for as few input tokens as possible.

Visibility

external

Input parameters

- uint amountOut
- uint amountInMax
- address[] calldata path
- address to
- uint deadline

Constraints

- ensure(deadline) modifier.
- The path must be valid.
- Input amount must not be excessive.

Events emit

None

Output

- uint[] memory amounts

- *swapExactTokensForBNB*

Description

Swaps an exact amount of tokens for as much BNB as possible.

Visibility

external

Input parameters

- uint amountIn
- uint amountOutMin
- address[] calldata path
- address to
- uint deadline

Constraints

- ensure(deadline) modifier.
- The path must be valid.
- Output amount must be sufficient.

Events emit

None

Output

- uint[] memory amounts

- *swapBNBForExactTokens*

Description

Receive an exact amount of tokens for as little BNB as possible.

Visibility

external payable

Input parameters

- uint amountOut
- address[] calldata path
- address to
- uint deadline

Constraints

- ensure(deadline) modifier.
- The path must be valid.
- Input amount must not be excessive.

Events emit

None

Output

- uint[] memory amounts

- *swapExactTokensForTokensSupportingFeeOnTransferTokens*

Description

Identical to swapExactTokensForTokens, but succeeds for tokens that take a fee on transfer.

Visibility

external

Input parameters

- uint amountIn
- uint amountOutMin
- address[] calldata path
- address to
- uint deadline

Constraints

- ensure(deadline) modifier.
- Output amount must be sufficient.

Events emit

None

Output

None

- *swapExactBNBForTokensSupportingFeeOnTransferTokens*

  #### Description

  Identical to swapExactBNBForTokens, but succeeds for tokens that take a fee on transfer.

  #### Visibility

  external payable

  #### Input parameters

  - uint amountOutMin
  - address[] calldata path
  - address to
  - uint deadline

  #### Constraints

  - ensure(deadline) modifier.
  - The path must be valid.
  - Output amount must be sufficient.

  #### Events emit

  None

  #### Output

  None

- *swapExactTokensForBNBSupportingFeeOnTransferTokens*

  #### Description

  Identical to swapExactTokensForBNB, but succeeds for tokens that take a fee on transfer.

  #### Visibility

  external

  #### Input parameters

- uint amountIn
- uint amountOutMin
- address[] calldata path
- address to
- uint deadline

Constraints

- ensure(deadline) modifier.
- The path must be valid.
- Output amount must be sufficient.

Events emit

None

Output

None

Read-Only and Non-Public Functions

BSCswapRouter contract has following read-only and non-public functions:

- function _addLiquidity(address tokenA, address tokenB, uint amountADesired, uint amountBDesired, uint amountAMin, uint amountBMin) internal virtual returns (uint amountA, uint amountB)
- function _swap(uint[] memory amounts, address[] memory path, address _to) internal virtual
- function _swapSupportingFeeOnTransferTokens(address[] memory path, address _to) internal virtual
- function quote(uint amountA, uint reserveA, uint reserveB) public pure virtual override returns (uint amountB)
- function getAmountOut(uint amountIn, uint reserveIn, uint reserveOut) public pure virtual override returns (uint amountOut)
- function getAmountIn(uint amountOut, uint reserveIn, uint reserveOut) public pure virtual override returns (uint amountIn)
- function getAmountsOut(uint amountIn, address[] memory path) public view virtual override returns (uint[] memory amounts)
- function getAmountsIn(uint amountOut, address[] memory path) public view virtual override returns (uint[] memory amounts)

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

Audit overview

Critical

No critical issues were found.

High

No high severity issues were found.

Medium

No medium severity issues were found.

Low

No low severity issues were found.

Lowest / Code style / Best Practice

1. Some code style issues were found by the static code analyzers.

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** informational issue during the audit.

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.