HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: WhiteSwap
Date: November 5th, 2020

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 WhiteSwap (56 pages)
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Tokens Swap
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/WhiteSwap/whiteswap-contracts
Commit	E42B1ED881EEB3719029FC3C250BE783288EAB64
Timeline	30 OCT 2020 - 5 NOV 2020
Changelog	5 NOV 2020 - Initial Audit 6 NOV 2020 - Customer comments added

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by WhiteSwap (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contract and its code review conducted between October 30th, 2020 - November 5th, 2020.

Scope

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

```
Repository: https://github.com/WhiteSwap/whiteswap-contracts
Commit: E42B1ED881EEB3719029FC3C250BE783288EAB64
```

Files in scope of review
contracts/proxy/WSProxy.sol
contracts/proxy/WSProxyFactory.sol
contracts/proxy/WSProxyPair.sol
contracts/proxy/WSProxyRouter.sol
contracts/WSPair.sol
contracts/WSController.sol
contracts/WSFactory.sol
contracts/WSRouter.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:

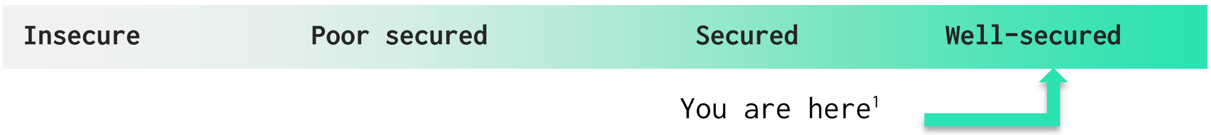
Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data Consistency

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

Executive Summary

According to the assessment, the Customer's smart contracts can be improved to follow best practices.

We described issues in the conclusion of these documents. Please read the whole document to estimate the risks well.

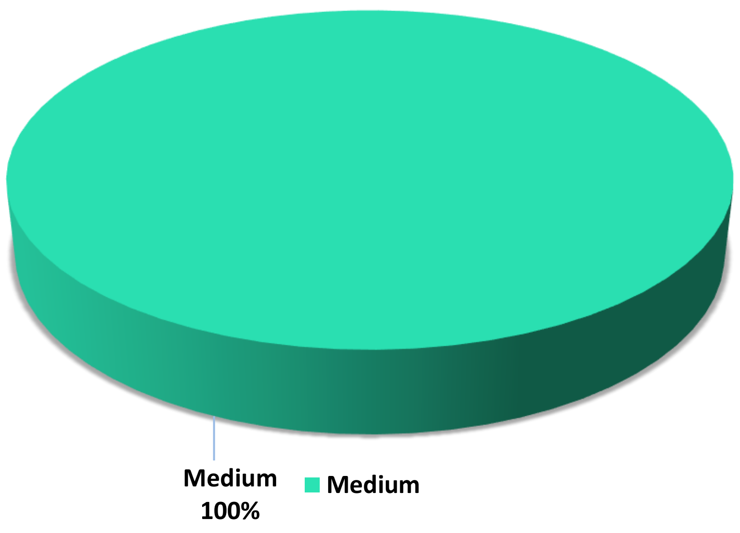


Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. A general overview is presented in AS-IS section,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

Security engineers found **1** Medium severity issue during the audit.

[1] Look for details and justification in conclusion section

Graph 1. The distribution of vulnerabilities.



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 essential 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

WSProxy.sol

Imports

WSProxy.sol file has 1 import:

- *IWSProxy.sol* – from project files;

Proxy contract

Description

Proxy is abstract contract provides a fallback function that delegates all calls to another contract using the EVM instruction *delegatecall*.

Inheritance

Proxy contract inherits nothing.

Functions

Proxy has 5 functions:

- ***_delegate***

  Description

  Delegates the current call to *implementation*.

  Visibility

  internal

  Input parameters

  - *address implementation* – an address of implementation;

  Constraints

  None

  Events emit

None

Output

None

- ***_implementation***

  Description

  This is a virtual function that should be overridden to return the address to delegate to.

  Visibility

  internal virtual view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns the address for delegation.

- ***_fallback***

  Description

  Delegates the current call.

  Visibility

  internal

  Input parameters

  None

Constraints

None

Events emit

None

Output

None

- ***fallback***

Description

A fallback function that delegates call. Will be fired if no other function in the contract matches the call data.

Visibility

payable external

Input parameters

None

Constraints

None

Events emit

None

Output

None

- ***receive***

Description

A fallback function that delegates call. Will be fired if the call data is empty.

Visibility

payable external

Input parameters

None

Constraints

None

Events emit

None

Output

None

UpgradeableProxy contract

Description

UpgradeableProxy is a contract that implements an upgradeable proxy. It is upgradeable because calls are delegated to an implementation address that can be changed.

Inheritance

UpgradeableProxy contract inherits *Proxy*.

Fields

UpgradeableProxy contract has 1 field:

- *bytes32 private constant _IMPLEMENTATION_SLOT* – storage slot with the address of the current implementation;

Functions

UpgradeableProxy has 4 functions:

- ***constructor***

  Description

Initializes contract.

Visibility

public payable

Input parameters

None

Constraints

- The *_IMPLEMENTATION_SLOT* must be the keccak-256 hash of "eip1967.proxy.implementation" subtracted by 1.

Events emit

None

Output

None

- ***_implementation***

Description

Used to get the current implementation address.

Visibility

internal view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns the current implementation address.

- *_upgradeTo*

Description

Upgrades the proxy to a new implementation.

Visibility

virtual internal

Input parameters

  - *address newImplementation* – an address of the new implementation;

Constraints

None

Events emit

  - Upgraded

Output

None

- *_setImplementation*

Description

Stores a new address in the EIP1967 implementation slot.

Visibility

private

Input parameters

  - *address newImplementation* – an address of the new implementation;

Constraints

- ○ The new implementation address must be different from the current implementation address.

Events emit

None

Output

None

TransparentUpgradeableProxy

Description

TransparentUpgradeableProxy contract implements a proxy that is upgradeable by an admin.

Inheritance

TransparentUpgradeableProxy contract
inherits *UpgradeableProxy* and *IWSProxy*.

Fields

TransparentUpgradeableProxy contract has 1 field:

- *bytes32 private constant _ADMIN_SLOT* – storage slot with the admin of the contract;

Modifiers

TransparentUpgradeableProxy contract has 1 modifier:

- *ifAdmin* – delegates the call to the implementation if the sender is not an administrator;

Functions

TransparentUpgradeableProxy has 9 functions:

- ***constructor***

Description

Initializes an upgradeable proxy managed by *_admin*

Visibility

public

Input parameters

None

Constraints

- The *_ADMIN_SLOT* must be the keccak-256 hash of "eip1967.proxy.admin" subtracted by 1.

Events emit

None

Output

None

- ***admin***

Description

Used to get the current admin.

Visibility

external

Input parameters

None

Constraints

- Only admin can call it.

Events emit

None

Output

Returns the current admin.

- ***initialize***

  Description

  Sets *_newImplementation*, *_admin* and make delegatecall with *_data*.

  Visibility

  external

  Input parameters

  - *address _newImplementation* – an address of a new implementation;
  - *address _admin* – an address of the admin;
  - *bytes calldata _data* – a call data;

  Constraints

  - Only admin can call it.

  Events emit

  None

  Output

  None

- ***implementation***

  Description

  Used to get current implementation.

  Visibility

  external

  Input parameters

  None

  Constraints

- Only admin can call it.

Events emit

None

Output

Returns the current implementation.

- *changeAdmin*

Description

Changes the admin of the proxy.

Visibility

external

Input parameters

- *address newAdmin* – an address of a new admin;

Constraints

- Only admin can call it.
- The new administrator must be different from the current administrator.

Events emit

- *AdminChanged*

Output

None

- *upgradeTo*

Description

Changes the implementation of the proxy.

Visibility

external

Input parameters

- *address newImplementation* – an address of the new implementation;

Constraints

- Only admin can call it.

Events emit

None

Output

None

- *upgradeToAndCall*

Description

Modifies the proxy implementation and then calls a function from the new implementation as specified in the *data*.

Visibility

external payable

Input parameters

- *address newImplementation* – an address of a new implementation;
- *bytes calldata data* – a call data;

Constraints

- Only admin can call it.

Events emit

None

Output

None

- *_admin*

Description

Used to get current admin address.

Visibility

internal view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns the current admin address.

- *_setAdmin*

Description

Stores a new address in the EIP1967 admin slot.

Visibility

private

Input parameters

- *address newAdmin* – an address of the new admin;

Constraints

- Admin address can't be zero.

Events emit

None

Output

None

WSProxyFactory.sol, WSProxyPair.sol, WSProxyRouter.sol

Description

WSProxyFactory, *WSProxyPair*, *WSProxyRouter* are contracts that only inherit *TransparentUpgradeableProxy*.

WSController.sol

Imports

WSController.sol file has 4 imports:

- *IWSProxy.sol* – from project files;
- *IWSController.sol* – from project files;
- *IWSImplementation.sol* – from project files;
- *Ownable.sol* – from project files;

WSController contract

Description

WSController contract used to manage the proxy.

Inheritance

WSController contract inherits *Ownable* and *IWSController*.

Fields

WSController contract contract has 3 fields:

- *pairLogic* – an address of pair logic;
- *currentAdmin* – an address of admin;
- *uint256 constant public PAIR_TYPE* – a pair type identifier;

Functions

WSController has 7 functions:

- ***constructor***

Description

Initializes the contract.

Visibility

public

Input parameters

None

Constraints

- Pair logic address can't be zero.

Events emit

None

Output

None

- *updatePairLogic*

Description

Updates pair logic.

Visibility

external

Input parameters

- *address _logic* – an addres of pair logic;

Constraints

- Only Owner can call it.

Events emit

- *NewPairLogic*

Output

None

- *updateCurrentAdmin*

Description

Updates current admin.

Visibility

external

Input parameters

  - *address _newAdmin* – an addres of a new admin;

Constraints

  - Only Owner can call it.

Events emit

  - *NewAdmin*

Output

None

- *updateProxyPair*

Description

Updates a pair of proxy.

Visibility

external

Input parameters

  - *address _proxy* – an addres of the proxy;

Constraints

  - Proxy implementation type must match *PAIR_TYPE*.

Events emit

- *UpdateProxy*

Output

None

- ***setAdminForProxy***

Description

Used to set the admin for the proxy.

Visibility

external

Input parameters

- *address _proxy* – an addres of the proxy;

Constraints

None

Events emit

- *ChangeAdmin*

Output

None

- ***getLogicForPair***

Description

Used to get the pair logic.

Visibility

external view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns an address of the pair logic.

- *getCurrentAdmin*

  Description

  Used to get the addres of the current admin.

  Visibility

  external view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns the addres of the current admin.

WSFactory.sol

Imports

WSFactory.sol file has 5 imports:

- *IWSFactory.sol* – from project files;
- *IWSController.sol* – from project files;
- *WSProxyPair.sol* – from project files;
- *IWSPair.sol* – from project files;

- *IWSImplementation.sol* – from project files;

WSFactory contract

Description

WSFactory contract used to create pairs.

Inheritance

WSFactory contract inherits *IWSFactory* and *IWSImplementation*.

Fields

WSFactory contract contract has 6 fields:

- *bool private initialized* – initialization indicator;
- *address public override feeTo* – an address to which the fee will be transferred;
- *address public override feeToSetter* – an address of fee setter;
- *address public controller* – an address of controller;
- *mapping(address => mapping(address => address)) public override getPair* – a mapping used for storing pairs;
- *address[] public override allPairs* – a list of pairs;

Functions

WSFactory has 6 functions:

- ***initialize***

  Description

  Initializes the contract.

  Visibility

  public

  Input parameters

  - *address _feeToSetter* – an address of fee setter;
  - *address _controller* – an address of controller;

  Constraints

    - The contract should not be initialized yet.
    - *_controller* should not be zero address.
    - *_feeToSetter* should not be zero address.

  Events emit

  None

  Output

  Returns true if success.

- ***allPairsLength***

  Description

  Used to get the total number of created pairs.

  Visibility

  external view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns the total number of created pairs.

- ***createPair***

  Description

  Creates a pair.

  Visibility

external

Input parameters

- *address tokenA* – an address of token;
- *address tokenB* – an address of token;

Constraints

- *tokenA* should not match *tokenB*.
- *tokenA* and *tokenB* should not be zero adresses.
- Pair should not exist yet.
- Pair should be successfully initialized.

Events emit

- *PairCreated*

Output

Returns an address of the pair.

- ***setFeeTo***

Description

Sets the address to which the fee will be transferred.

Visibility

external

Input parameters

- *address _feeTo* – an address to which the fee will be transferred;

Constraints

- Only *feeToSetter* can call it.

Events emit

None

Output

None

- *setFeeToSetter*

Description

Sets fee setter address.

Visibility

external

Input parameters

  - *address _feeToSetter* – an address of fee setter;

Constraints

  - Only *feeToSetter* can call it.

Events emit

None

Output

None

- *getImplementationType*

Description

Used to get the type of implementation.

Visibility

external pure

Input parameters

None

Constraints

None

Events emit

None

Output

Returns 1 - it is a factory type.

WSPair.sol

Imports

WSPair.sol file has 8 imports:

- *IWSPair.sol* – from project files;
- *IWSImplementation.sol* – from project files;
- *WSERC20.sol* – from project files;
- *Math.sol* – from project files;
- *UQ112x112.sol* – from project files;
- *IERC20.sol* – from project files;
- *IWSFactory.sol* – from project files;
- *IWSCallee.sol* – from project files;

WSPair contract

Description

WSPair contract used to manage token pairs.

Inheritance

WSPair contract inherits *IWSPair*, *WSERC20* and *IWSImplementation*.

Usings

WSPair contract use:

- *SafeMath* for *uint*;
- *UQ112x112* for *uint224*;

Modifiers

WSPair contract has 1 modifier:

- *lock* – blocks other calls while the current call is in progress;

Fields

WSPair contract contract has 13 fields:

- *uint public override constant MINIMUM_LIQUIDITY* – the minimum liquidity valueж
- *bytes4 private constant SELECTOR* – the selector of the transfer function;
- *address public override factory* – an address of factory;
- *address public override token0* – an address of token;
- *address public override token1* – an address of token;
- *uint112 private reserve0* – the number of tokens in *token0* reserve;
- *uint112 private reserve1* – the number of tokens in *token1* reserve;
- *uint32 private blockTimestampLast* – a *block.timestamp* of the last update;
- *uint public override price0CumulativeLast* – the price accumulator of *token0*;
- *uint public override price1CumulativeLast* – the price accumulator of *token1*;
- *uint public override kLast* – the result of multiplying *reserve0* by *reserve1*;
- *bool private initialized* – initialization indicator;
- *uint private unlocked* – lock indicator;

Functions

WSPair has 12 functions:

- ***isLocked***

  Description

  Used to get the lock indicator.

  Visibility

  external view

  Input parameters

  None

  Constraints

None

Events emit

None

Output

Returns the lock indicator.

- *getReserves*

Description

Used to get reserves and the *block.timestamp* of the last update.

Visibility

public view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns reserves and the *block.timestamp* of the last update.

- *_safeTransfer*

Description

Transfers tokens.

Visibility

private

Input parameters

- *address token* – an address of the token;
- *address to* – an address of the receiver;
- *uint value* – an amount of tokens;

Constraints

- The transfer must be successful.

Events emit

None

Output

None

- ***initialize***

Description

Initializes the contract.

Visibility

external

Input parameters

- *address _factory* – an address of factory;
- *address _token0* – an address of token0;
- *address _token1* – an address of token1;

Constraints

- The contract should not be initialized yet.

Events emit

None

Output

None

- ***_update***

  Description

  Used to update reserves, *blockTimestampLast*, and price accumulators.

  Visibility

  private

  Input parameters

  - *uint balance0* – a balance of the token0;
  - *uint balance1* – a balance of the token1;
  - *uint112 _reserve0* – a reserve of the token0;
  - *uint112 _reserve1* – a reserve of the token1;

  Constraints

  - Balances should not overflow *uint112* range.

  Events emit

  - *Sync*

  Output

  None

- ***_mintFee***

  Description

  Mints fee.

  Visibility

  private

  Input parameters

  - *uint112 _reserve0* – a reserve of the token0;
  - *uint112 _reserve1* – a reserve of the token1;

  Constraints

None

Events emit

None

Output

Returns *feeOn* bool indicator.

- ***mint***

Description

Mints tokens to the address.

Visibility

external

Input parameters

  - *address to* – an address of receiver;

Constraints

  - Liquidity must be greater than 0.

Events emit

  - *Mint*

Output

None

- ***burn***

Description

Burns all tokens from the contract and transfers them to the address.

Visibility

external

Input parameters

- *address to* – an address of receiver;

Constraints

- Liquidity must be greater than 0.

Events emit

- *Burn*

Output

None

- *swap*

Description

Used to swap tokens.

Visibility

external

Input parameters

- *uint amount0Out* – an amount outcome from the token0;
- *uint amount1Out* – an amount outcome from the token1;
- *address to* – an address of receiver;
- *bytes calldata data* – a call data;

Constraints

- Outcome amounts must be greater than 0.
- Outcome amounts must be less than reserves.
- Receiver address should not match token0 or token1 address.
- Income amounts must be greater than 0.

Events emit

- *Swap*

Output

None

- *skim*

  Description

  Makes balances match reserves.

  Visibility

  external

  Input parameters

  - *address to* – an address of receiver;

  Constraints

  None

  Events emit

  None

  Output

  None

- *sync*

  Description

  Makes reserves match balances.

  Visibility

  external

  Input parameters

  None

  Constraints

  None

Events emit

None

Output

None

- *getImplementationType*

Description

Used to get the type of implementation.

Visibility

external pure

Input parameters

None

Constraints

None

Events emit

None

Output

Returns 2 - it is a pair type.

WSRouter.sol

Imports

WSRouter.sol file has 8 imports:

- *IWSFactory.sol* – from project files;
- *TransferHelper.sol* – from project files;
- *WSLibrary.sol* – from project files;
- *IWSRouter.sol* – from project files;
- *IERC20.sol* – from project files;
- *IWSERC20.sol* – from project files;

- *IWETH.sol* – from project files;
- *IWSImplementation.sol* – from project files;

WSRouter contract

Description

WSRouter contract used to manage token pairs.

Inheritance

WSRouter contract inherits *IWSRouter* and *IWSImplementation*.

Usings

WSRouter contract use:

- *SafeMath* for *uint*;

Modifiers

WSRouter contract has 1 modifier:

- *ensure* – checks that the deadline has not expired;

Fields

WSRouter contract contract has 3 fields:

- *bool private initialized* – initialization indicator;
- *address public override factory* – an address of factory;
- *address public override WETH* – an address of WETH;

Functions

WSRouter has 27 functions:

- ***initialize***

  Description

  Initializes the contract.

  Visibility

  public

Input parameters

- *address _factory* – an address of factory;
- *address _WETH* – an address of WETH;

Constraints

- The contract should not be initialized yet.

Events emit

None

Output

Returns true if success.

- ***_addLiquidity***

Description

Calculates liquidity that should be added to pair.

Visibility

internal

Input parameters

- *address tokenA* – an address of the tokenA;
- *address tokenB* – an address of the tokenB;
- *uint amountADesired* – a desired amount of the tokenA;
- *uint amountBDesired* – a desired amount of the tokenB;
- *uint amountAMin* – a min amount of the tokenA;
- *uint amountBMin* – a min amount of the tokenB;

Constraints

- *amountA* and *amountB* should be sufficient.

Events emit

None

Output

Returns *amountA* and *amountB*.

- *addLiquidity*

Description

Adds liquidity to a pair.

Visibility

external

Input parameters

  - *address tokenA* – an address of the tokenA;
  - *address tokenB* – an address of the tokenB;
  - *uint amountADesired* – a desired amount of the tokenA;
  - *uint amountBDesired* – a desired amount of the tokenB;
  - *uint amountAMin* – a min amount of the tokenA;
  - *uint amountBMin* – a min amount of the tokenB;
  - *address to* – an address of receiver;
  - *uint deadline* – the deadline timestamp;

Constraints

  - The deadline has not expired.

Events emit

None

Output

Returns *amountA*, *amountB* and *liquidity*.

- *addLiquidityETH*

Description

Adds liquidity to a pair with ETH.

Visibility

external payable

Input parameters

  - *address token* – an address of token;

- *uint amountTokenDesired* – a desired amount of the token;
- *uint amountTokenMin* – a min amount of the token;
- *uint amountETHMin* – a min amount of the ETH;
- *address to* – an address of receiver;
- *uint deadline* – the deadline timestamp;

Constraints

- The deadline has not expired.

Events emit

None

Output

Returns *amountA*, *amountB* and *liquidity*.

- ***removeLiquidity***

Description

Removes the liquidity from a pair.

Visibility

public

Input parameters

- *address tokenA* – an address of the tokenA;
- *address tokenB* – an address of the tokenB;
- *uint liquidity* – a liquidity amount;
- *uint amountAMin* – a min amount of the tokenA;
- *uint amountBMin* – a min amount of the tokenB;
- *address to* – an address of receiver;
- *uint deadline* – the deadline timestamp;

Constraints

- *amountA* and *amountB* should be sufficient.

Events emit

None

Output

Returns *amountA* and *amountB*.

- ***removeLiquidityETH***

Description

Removes the liquidity from a pair with ETH.

Visibility

public

Input parameters

- *address token* – an address of token;
- *uint liquidity* – a liquidity amount;
- *uint amountTokenMin* – a min amount of the token;
- *uint amountETHMin* – a min amount of the ETH;
- *address to* – an address of receiver;
- *uint deadline* – the deadline timestamp;

Constraints

- The deadline has not expired.

Events emit

None

Output

Returns *amountToken* and *amountETH*.

- ***removeLiquidityWithPermit***

Description

Removes the liquidity from a pair with permit.

Visibility

external

Input parameters

- *address tokenA* – an address of the tokenA;
- *address tokenB* – an address of the tokenB;
- *uint liquidity* – a liquidity amount;
- *uint amountAMin* – a min amount of the tokenA;
- *uint amountBMin* – a min amount of the tokenB;
- *address to* – an address of receiver;
- *uint deadline* – the deadline timestamp;
- *bool approveMax* – whether or not the approval amount in the signature is for liquidity or uint(-1);
- *uint8 v* – the v component of the permit signature;
- *bytes32 r* – the r component of the permit signature;
- *bytes32 s* – the s component of the permit signature;

Constraints

- The deadline has not expired.

Events emit

None

Output

Returns *amountA* and *amountB*.

- ***removeLiquidityETHWithPermit***

Description

Removes the liquidity from a pair with ETH and with permit.

Visibility

external

Input parameters

- *address token* – an address of token;
- *uint liquidity* – a liquidity amount;
- *uint amountTokenMin* – a min amount of the token;
- *uint amountETHMin* – a min amount of the ETH;
- *address to* – an address of receiver;
- *uint deadline* – the deadline timestamp;
- *bool approveMax* – whether or not the approval amount in the signature is for liquidity or uint(-1);
- *uint8 v* – the v component of the permit signature;
- *bytes32 r* – the r component of the permit signature;

- *bytes32 s* – the s component of the permit signature;

Constraints

None

Events emit

None

Output

Returns *amountToken* and *amountETH*.

- *removeLiquidityETHSupportingFeeOnTransferTokens*

Description

Removes the liquidity from a pair with ETH, succeeds for tokens that take a fee on transfer.

Visibility

public

Input parameters

- *address token* – an address of token;
- *uint liquidity* – a liquidity amount;
- *uint amountTokenMin* – a min amount of the token;
- *uint amountETHMin* – a min amount of the ETH;
- *address to* – an address of receiver;
- *uint deadline* – the deadline timestamp;

Constraints

- The deadline has not expired.

Events emit

None

Output

Returns *amountETH*.

- *removeLiquidityETHWithPermitSupportingFeeOnTransferTokens*

  Description

  Removes the liquidity from a pair with ETH and with permit, succeeds for tokens that take a fee on transfer.

  Visibility

  external

  Input parameters

  - *address token* – an address of token;
  - *uint liquidity* – a liquidity amount;
  - *uint amountTokenMin* – a min amount of the token;
  - *uint amountETHMin* – a min amount of the ETH;
  - *address to* – an address of receiver;
  - *uint deadline* – the deadline timestamp;
  - *bool approveMax* – whether or not the approval amount in the signature is for liquidity or uint(-1);
  - *uint8 v* – the v component of the permit signature;
  - *bytes32 r* – the r component of the permit signature;
  - *bytes32 s* – the s component of the permit signature;

  Constraints

  None

  Events emit

  None

  Output

  Returns *amountETH*.

- *_swap*

  Description

  Used to swap tokens.

  Visibility

  internal

Input parameters

- *uint[] memory amounts* – amounts;
- *address[] memory path* – path;
- *address _to* – an address of receiver;

Constraints

None

Events emit

None

Output

None

- *swapExactTokensForTokens*

Description

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

Visibility

external

Input parameters

- *uint amountIn* – income amount;
- *uint amountOutMin* – min outcome amount;
- *address[] calldata path* – an array of token addresses;
- *address to* – an address of receiver;
- *uint deadline* – the deadline timestamp;

Constraints

- The deadline has not expired.
- Output amount should be greater than or equal *amountOutMin*.

Events emit

None

Output

Returns the array of amounts.

- *swapTokensForExactTokens*

Description

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

Visibility

external

Input parameters

  - *uint amountOut* – outcome amount;
  - *uint amountInMax* – max income amount;
  - *address[] calldata path* – an array of token addresses;
  - *address to* – an address of receiver;
  - *uint deadline* – the deadline timestamp;

Constraints

  - The deadline has not expired.
  - Input amount should be less than or equal *amountInMax*.

Events emit

None

Output

Returns the array of amounts.

- *swapExactETHForTokens*

Description

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

Visibility

external payable

Input parameters

- *uint amountOutMin* – min outcome amount;
- *address[] calldata path* – an array of token addresses;
- *address to* – an address of receiver;
- *uint deadline* – the deadline timestamp;

Constraints

- The deadline has not expired.
- *path[0]* should be *WETH*.
- Output amount should be greater than or equal *amountOutMin*.

Events emit

None

Output

Returns the array of amounts.

- *swapTokensForExactETH*

Description

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

Visibility

external

Input parameters

- *uint amountOut* – outcome amount;
- *uint amountInMax* – max income amount;
- *address[] calldata path* – an array of token addresses;
- *address to* – an address of receiver;
- *uint deadline* – the deadline timestamp;

Constraints

- The deadline has not expired.
- *path[path.length - 1]* should be *WETH*.
- Input amount should be less than or equal *amountInMax*.

Events emit

None

Output

Returns the array of amounts.

- *swapExactTokensForETH*

Description

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

Visibility

external

Input parameters

  - *uint amountIn* – income amount;
  - *uint amountOutMin* – min outcome amount;
  - *address[] calldata path* – an array of token addresses;
  - *address to* – an address of receiver;
  - *uint deadline* – the deadline timestamp;

Constraints

  - The deadline has not expired.
  - *path[path.length - 1]* should be *WETH*.
  - Output amount should be greater than or equal *amountOutMin*.

Events emit

None

Output

Returns the array of amounts.

- *swapETHForExactTokens*

Description

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

Visibility

external payable

Input parameters

- *uint amountOut* – outcome amount;
- *address[] calldata path* – an array of token addresses;
- *address to* – an address of receiver;
- *uint deadline* – the deadline timestamp;

Constraints

- The deadline has not expired.
- *path[0]* should be *WETH*.
- Input amount should be less than or equal *msg.value*.

Events emit

None

Output

Returns the array of amounts.

- ***_swapSupportingFeeOnTransferTokens***

Description

Swaps with supporting fee-on-transfer tokens.

Visibility

internal

Input parameters

- *address[] memory path* – an array of token addresses;
- *address to* – an address of receiver;

Constraints

None

Events emit

None

Output

None

- *swapExactTokensForTokensSupportingFeeOnTransferTokens*

Description

Swaps an exact amount of input tokens for as many output tokens as possible, succeeds for tokens that take a fee on transfer.

Visibility

external

Input parameters

- *uint amountIn* – income amount;
- *uint amountOutMin* – min outcome amount;
- *address[] calldata path* – an array of token addresses;
- *address to* – an address of receiver;
- *uint deadline* – the deadline timestamp;

Constraints

- The deadline has not expired.
- Output amount should be greater than or equal *amountOutMin*.

Events emit

None

Output

None

- *swapExactETHForTokensSupportingFeeOnTransferTokens*

Description

Swaps an exact amount of ETH for as many output tokens as possible, succeeds for tokens that take a fee on transfer.

Visibility

external payable

Input parameters

- *uint amountOutMin* – min outcome amount;
- *address[] calldata path* – an array of token addresses;
- *address to* – an address of receiver;
- *uint deadline* – the deadline timestamp;

Constraints

- The deadline has not expired.
- *path[0]* should be *WETH*.
- Output amount should be greater than or equal *amountOutMin*.

Events emit

None

Output

None

- *swapExactTokensForETHSupportingFeeOnTransferTokens*

Description

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

Visibility

external

Input parameters

- *uint amountIn* – income amount;
- *uint amountOutMin* – min outcome amount;
- *address[] calldata path* – an array of token addresses;
- *address to* – an address of receiver;

- *uint deadline* – the deadline timestamp;

Constraints

- The deadline has not expired.
- *path[path.length - 1]* should be *WETH*.
- Output amount should be greater than or equal *amountOutMin*.

Events emit

None

Output

None

- ***getImplementationType***

Description

Used to get the type of implementation.

Visibility

external pure

Input parameters

None

Constraints

None

Events emit

None

Output

Returns 3 - it is a router type.

Library functions

quote, *getAmountOut*, *getAmountIn*, *getAmountsOut*, *getAmountsIn* ar
e just wrappers for *WSLibrary* functions. Without any additional
logic.

Audit overview

■ ■ ■ ■ Critical

No critical issues were found.

■ ■ ■ High

No high issues were found.

■ ■ Medium

1. Almost all files in the audit scope import dependencies that are not included in the audit scope. As soon as those dependencies are out of the audit scope, they are relatively safe.

File	Line	Dependency
contracts\WSController.sol	8	Ownable.sol
contracts\WSPair.sol	7	WSERC20.sol
	8	Math.sol
	9	UQ112x112.sol
contracts\WSRouter.sol	6	TransferHelper.sol
	8	WSLibrary.sol

Also, among the dependencies there are files that are copied to the project from the OpenZeppelin repository. We recommend to import those files directly from the OpenZeppelin repository.

The results of manual check of this dependency show next:

File	Source	Changes(diff)
Ownable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol	solversion
WSERC20.sol	https://github.com/Uniswap/uniswap-v2-core/blob/master/contracts/UniswapV2ERC20.sol	Naming, solversion
Math.sol	https://github.com/Uniswap/uniswap-v2-core/blob/master/contracts/libraries/Math.sol	solversion
UQ112x112.sol	https://github.com/Uniswap/uniswap-v2-core/blob/master/contracts/libraries/UQ112x112.sol	solversion
TransferHelper.sol	https://github.com/Uniswap/uniswap-lib/blob/master/contracts/libraries/TransferHelper.sol	solversion

WSLibrary.sol	https://github.com/Uniswap/uniswap-v2-periphery/blob/master/contracts/libraries/UniswapV2Library.sol	Naming, solversion, init hash code
DSMath.sol	https://github.com/Uniswap/uniswap-v2-core/blob/master/contracts/libraries/SafeMath.sol	Naming, solversion

Low

No lowest severity issues were found.

Lowest / Code style / Best Practice

No lowest severity issues were found.

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.

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

Security engineers found **1** Medium severity issues during the audit.

Violations in the following categories were found and addressed to the Customer:

Category	Check Item	Comments
Code review	▪ Repository consistency	▪ The project does not follow project structure best practices.

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 the 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 the 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 own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.