
# Smart Contract Security Audit

# 1. Executive Summary

On Aug. 17, 2020, the SlowMist security team received the JustSwap team's security audit application for JustSwap system, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

SlowMist Smart Contract DApp project test method:

| | |
|---|---|
| Black box testing | Conduct security tests from an attacker's perspective externally. |
| Grey box testing | Conduct security testing on code module through the scripting tool, observing the internal running status, mining weaknesses. |
| White box testing | Based on the open source code, non-open source code, to detect wether there are vulnerabilities in programs suck as nodes, SDK, etc. |

SlowMist Smart Contract DApp project risk level:

| | |
|---|---|
| Critical vulnerabilities | Critical vulnerabilities will have a significant impact on the security of the DApp project, and it is strongly recommended to fix the critical vulnerabilities. |
| High-risk vulnerabilities | High-risk vulnerabilities will affect the normal operation of DApp project. It is strongly recommended to fix high-risk vulnerabilities. |
| Medium-risk | Medium vulnerability will affect the operation of DApp project. It is recommended |

| | |
|---|---|
| vulnerablities | to fix medium-risk vulnerabilities. |
| Low-risk vulnerabilities | Low-risk vulnerabilities may affect the operation of DApp project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed. |
| Weaknesses | There are safety risks theoretically, but it is extremely difficult to reproduce in engineering. |
| Enhancement Suggestions | There are better practices for coding or architecture. |

# 2. Audit Methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and in-house automated analysis tools.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

- Reentrancy attack and other Race Conditions
- Replay attack
- Reordering attack
- Short address attack
- Denial of service attack
- Transaction Ordering Dependence attack

- Conditional Completion attack
- Authority Control attack
- Integer Overflow and Underflow attack
- TimeStamp Dependence attack
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Explicit visibility of functions state variables
- Logic Flaws
- Uninitialized Storage Pointers
- Floating Points and Numerical Precision
- tx.origin Authentication
- "False top-up" Vulnerability
- Scoping and Declarations

# 3. Project Background (Context)

## 3.1 Project Introduction

JustSwap is an exchange protocol on TRON for exchanges between TRC20 tokens.

**Project website:**

https://justswap.io/

**Audit code file:**

justswap.tar.gz:

MD5: be4b075dc236d2f614b06d6da419603b

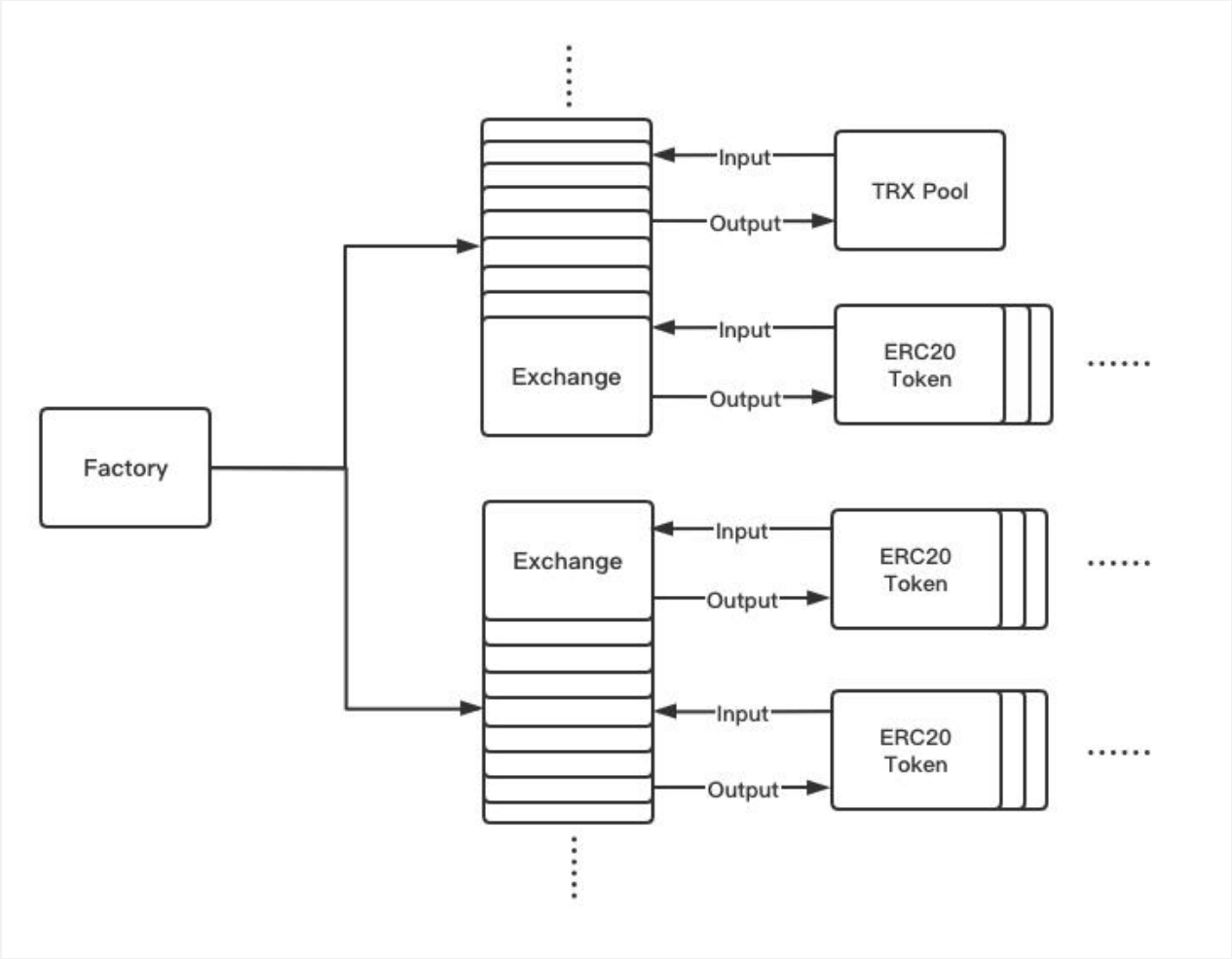
## 3.2 Project Structure

justswap:

```
.
├── README.md
├── interfaces
│   ├── IJustswapExchange.sol
│   ├── IJustswapFactory.sol
│   └── ITRC20.sol
├── justswap
│   ├── JustswapExchange.sol
│   └── JustswapFactory.sol
├── tokens
│   └── TRC20.sol
└── utils
    ├── ReentrancyGuard.sol
    ├── SafeMath.sol
    └── TransferHelper.sol
```

## 3.3 Contract Structure

JustSwap DApp is mainly divided into two parts, namely the contract factory and the token exchange part. Among them, the JustswapFactory contract is responsible for creating an independent exchange contract for each TRC20 token. The JustswapExchange contract is responsible for realizing the functions of providing a liquidity pool for token exchange, handling fee processing and custom capital pools. Each exchange contract is associated with a TRC20 token, and has a liquidity pool of TRX and this TRC20 token to realize the exchange function between TRX and token, the fee generated during the exchange process are stored in the liquidity pool. The overall structure of the contract is as follows:




# 4. Code Overview

## 4.1 Main File Hash

| No | File Name | SHA-1 Hash |
| --- | --- | --- |
| 1 | JustswapExchange.sol | 64c33c94f379890d580c5971ec44073cd42c7e7c |
| 2 | TRC20.sol | c02bb96b3a004f3b82db83a80b1c53abe3426992 |
| 3 | SafeMath.sol | 50f1e0e4fd6bc2002e4111221991382f18e0fef4 |

| | | |
|---|---|---|
| 4 | ITRC20.sol | c130889347ff735dbe86ea35af3799a67350e92c |
| 5 | IJustswapFactory.sol | fd2b8e7b53515bf242059bd1b47cdcdc9b2a0227 |
| 6 | IJustswapExchange.sol | 3b8d8cbef22e2f418f50ff1e61cf6140359ab6f8 |
| 7 | ReentrancyGuard.sol | 8170dd07eee4eb2402ad65f3323eadb0dc2d8709 |
| 8 | TransferHelper.sol | d6fe29a53d7f142e6a296e198658a28bf5ce8945 |
| 9 | JustswapFactory.sol | 84d90357c979f5f81991551986033ccf6589a729 |

## 4.2 Main function visibility analysis

| Contract Name | Function Name | Visibility |
|---|---|---|
| JustswapExchange | Implementation | TRC20, ReentrancyGuard |
| | setup | Public |
| | getInputPrice | Public |
| | getOutputPrice | Public |
| | trxToTokenInput | Private |
| | trxToTokenSwapInput | Public |
| | trxToTokenTransferInput | Public |
| | trxToTokenOutput | Private |
| | trxToTokenSwapOutput | Public |
| | trxToTokenTransferOutput | Public |

| | | |
|---|---|---|
| | tokenToTrxInput | Private |
| | tokenToTrxSwapInput | Public |
| | tokenToTrxTransferInput | Public |
| | tokenToTrxOutput | Private |
| | tokenToTrxSwapOutput | Public |
| | tokenToTrxTransferOutput | Public |
| | tokenToTokenInput | Private |
| | tokenToTokenSwapInput | Public |
| | tokenToTokenTransferInput | Public |
| | tokenToTokenOutput | Private |
| | tokenToTokenSwapOutput | Public |
| | tokenToTokenTransferOutput | Public |
| | tokenToExchangeSwapInput | Public |
| | tokenToExchangeTransferInput | Public |
| | tokenToExchangeSwapOutput | Public |
| | tokenToExchangeTransferOutput | Public |
| | getTrxToTokenInputPrice | Public |
| | getTrxToTokenOutputPrice | Public |
| | getTokenToTrxInputPrice | Public |

| | | |
|---|---|---|
| | getTokenToTrxOutputPrice | Public |
| | tokenAddress | Public |
| | factoryAddress | Public |
| | addLiquidity | Public |
| | removeLiquidity | Public |
| TRC20 | Implementation | —— |
| | totalSupply | Public |
| | balanceOf | Public |
| | allowance | Public |
| | transfer | Public |
| | approve | Public |
| | transferFrom | Public |
| | increaseAllowance | Public |
| | decreaseAllowance | Public |
| | _transfer | Internal |
| | _mint | Internal |
| | _burn | Internal |
| | _approve | Internal |
| | _burnFrom | Internal |

| | | |
|---|---|---|
| SafeMath | Library | —— |
| | mul | Internal |
| | div | Internal |
| | sub | Internal |
| | add | Internal |
| | mod | Internal |
| ITRC20 | Interface | —— |
| | transfer | External |
| | approve | External |
| | transferFrom | External |
| | totalSupply | External |
| | balanceOf | External |
| | allowance | External |
| IJustswapFactory | Interface | —— |
| | initializeFactory | External |
| | createExchange | External |
| | getExchange | External |
| | getToken | External |
| | getTokenWihId | External |

| | | |
|---|---|---|
| IJustswapExchange | Interface | —— |
| | getInputPrice | External |
| | trxToTokenTransferInput | External |
| | trxToTokenSwapOutput | External |
| | trxToTokenTransferOutput | External |
| | tokenToTrxSwapInput | External |
| | tokenToTrxTransferInput | External |
| | tokenToTrxSwapOutput | External |
| | tokenToTrxTransferOutput | External |
| | tokenToTokenSwapInput | External |
| | tokenToTokenTransferInput | External |
| | tokenToTokenSwapOutput | External |
| | tokenToTokenTransferOutput | External |
| | tokenToExchangeSwapInput | External |
| | tokenToExchangeTransferInput | External |
| | tokenToExchangeSwapOutput | External |
| | tokenToExchangeTransferOutput | External |
| | getTrxToTokenInputPrice | External |
| | getTrxToTokenOutputPrice | External |

| | | |
|---|---|---|
| | getTokenToTrxInputPrice | External |
| | getTokenToTrxOutputPrice | External |
| | tokenAddress | External |
| | factoryAddress | External |
| | addLiquidity | External |
| | removeLiquidity | External |
| ReentrancyGuard | Implementation | —— |
| TransferHelper | Library | —— |
| | safeApprove | Internal |
| | safeTransfer | Internal |
| | safeTransferFrom | Internal |
| JustswapFactory | Implementation | —— |
| | initializeFactory | Public |
| | createExchange | Public |
| | getExchange | Public |
| | getToken | Public |
| | getTokenWithId | Public |

## 4.3 Code Audit

### 4.3.1 The risk that the liquidity pool cannot be removed

`address(token).safeTransferFrom(msg.sender, address(this), token_amount)` is used in the addLiquidity function, and `address(token).safeTransfer(msg.sender, token_amount)` is used in the removeLiquidity function. The two are inconsistent it may cause the risk that liquidity cannot be removed. For example: The return value of a contract's transferFrom function conforms to the return value specification for TRC20 tokens defined in the TIP20 standard, but the return value of the transfer function does not conform to the return value specification for TRC20 tokens defined in the TIP20 standard. This may cause the addLiquidity operation to succeed, but the removeLiquidity operation cannot succeed.

**Code location:** File JustswapExchange.sol line 631, 660

```
    function addLiquidity(uint256 min_liquidity, uint256 max_tokens, uint256 deadline) public payable nonReentrant returns (uint256) {
    require(deadline > block.timestamp && max_tokens > 0 && msg.value > 0, 'JustExchange#addLiquidity: INVALID_ARGUMENT');
    uint256 total_liquidity = _totalSupply;

    if (total_liquidity > 0) {
      require(min_liquidity > 0, "min_liquidity must greater than 0");
      uint256 trx_reserve = address(this).balance.sub(msg.value);
      uint256 token_reserve = token.balanceOf(address(this));
      uint256 token_amount = (msg.value.mul(token_reserve).div(trx_reserve)).add(1);
      uint256 liquidity_minted = msg.value.mul(total_liquidity).div(trx_reserve);

      require(max_tokens >= token_amount && liquidity_minted >= min_liquidity, "max tokens not meet or liquidity_minted not meet min_liquidity");
      _balances[msg.sender] = _balances[msg.sender].add(liquidity_minted);
      _totalSupply = total_liquidity.add(liquidity_minted);
```

```
        require(address(token).safeTransferFrom(msg.sender, address(this), token_amount), "transfer failed");
        emit AddLiquidity(msg.sender, msg.value, token_amount);
        emit Snapshot(msg.sender,address(this).balance,token.balanceOf(address(this)));
        emit Transfer(address(0), msg.sender, liquidity_minted);
        return liquidity_minted;
    } else {
        require(address(factory) != address(0) && address(token) != address(0) && msg.value >= 10_000_000,
"INVALID_VALUE");
        require(factory.getExchange(address(token)) == address(this), "token address not meet exchange");
        uint256 token_amount = max_tokens;
        uint256 initial_liquidity = address(this).balance;
        _totalSupply = initial_liquidity;
        _balances[msg.sender] = initial_liquidity;

        require(address(token).safeTransferFrom(msg.sender, address(this), token_amount), "tranfer failed");
        emit AddLiquidity(msg.sender, msg.value, token_amount);
        emit Snapshot(msg.sender,address(this).balance,token.balanceOf(address(this)));
        emit Transfer(address(0), msg.sender, initial_liquidity);
        return initial_liquidity;
    }
}
```

```
  function removeLiquidity(uint256 amount, uint256 min_trx, uint256 min_tokens, uint256 deadline) public nonReentrant
returns (uint256, uint256) {
    require(amount > 0 && deadline > block.timestamp && min_trx > 0 && min_tokens > 0, "illegal input parameters");
    uint256 total_liquidity = _totalSupply;
    require(total_liquidity > 0, "total_liquidity must greater than 0");
    uint256 token_reserve = token.balanceOf(address(this));
    uint256 trx_amount = amount.mul(address(this).balance) / total_liquidity;
    uint256 token_amount = amount.mul(token_reserve) / total_liquidity;
    require(trx_amount >= min_trx && token_amount >= min_tokens, "min_token or min_trx not meet");

    _balances[msg.sender] = _balances[msg.sender].sub(amount);
    _totalSupply = total_liquidity.sub(amount);
    msg.sender.transfer(trx_amount);

    require(address(token).safeTransfer(msg.sender, token_amount), "transfer failed");
    emit RemoveLiquidity(msg.sender, trx_amount, token_amount);
    emit Snapshot(msg.sender,address(this).balance,token.balanceOf(address(this)));
    emit Transfer(msg.sender, address(0), amount);
```

```
        return (trx_amount, token_amount);
    }
}
```

Fix status: After confirming with the project party, the TRC20 specification requires that the transfer and transferFrom conform to the specified format. Justswap only supports TRC20 tokens approved by TronScan. All tokens that do not conform to the TRC20 specification will not be supported by JustSwap.

### 4.3.2 Part of the code is redundant

An initializeFactory function exists in a JustswapFactory contract to initialize the contract. During initialization, an exchangeTemplate with a non-zero address is passed in and determined before creating a transaction contract in the createExchange function. However, exchangeTemplate is not used after that, and the transaction contract is created directly using 'new JustswapExchange()'. So the initializeFactory function and the check for an exchangeTemplate in the createExchange function are redundant. The _mint function in the TRC20 contract is internal and not called by other contracts, and the _burnFrom function is internal and not called by other contracts. This code is redundant. The safeApprove function in the TransferHelper contract has internal visibility and is not invoked by any other contract. This code is redundant.

**Code location:** File JustswapFactory.sol line 25-29, 33. File TRC20.sol line 139, 183. File TransferHelper.sol line 6.

```
function initializeFactory(address template) public {
    require(exchangeTemplate == address(0), "exchangeTemplate already set");
    require(template != address(0), "illegal template");
    exchangeTemplate = template;
```

```
}
```

```
function createExchange(address token) public returns (address) {
    require(token != address(0), "illegal token");
    require(exchangeTemplate != address(0), "exchangeTemplate not set");
    require(token_to_exchange[token] == address(0), "exchange already created");
    JustswapExchange exchange = new JustswapExchange();
    exchange.setup(token);
    token_to_exchange[token] = address(exchange);
    exchange_to_token[address(exchange)] = token;
    uint256 token_id = tokenCount + 1;
    tokenCount = token_id;
    id_to_token[token_id] = token;
    emit NewExchange(token, address(exchange));
    return address(exchange);
}
```

```
function _mint(address account, uint256 value) internal {
    require(account != address(0));

    _totalSupply = _totalSupply.add(value);
    _balances[account] = _balances[account].add(value);
    emit Transfer(address(0), account, value);
}
```

```
function _burnFrom(address account, uint256 value) internal {
    _burn(account, value);
    _approve(account, msg.sender, _allowed[account][msg.sender].sub(value));
}
```

```
    function safeApprove(address token, address to, uint value) internal returns (bool){
        // bytes4(keccak256(bytes('approve(address,uint256)')));
        (bool success, bytes memory data) = token.call(abi.encodeWithSelector(0x095ea7b3, to, value));
        return (success && (data.length == 0 || abi.decode(data, (bool))));
    }
```

Fix status: After confirming with the project party, no impact on the business, and the code will not be modified.

# 5. Audit Result

## 5.1 Low-risk Vulnerability

- Remove liquidity pool design defects

## 5.2 Enhancement Suggestions

- Part of the code is redundant

## 5.3 Conclusion

Audit Result : Passed

Audit Number : 0X002008250002

Audit Date : Aug. 25, 2020

Audit Team : SlowMist Security Team

Summary conclusion: The are 2 security issues found during the audit. After communication and feedback, with the Anyswap team, confirms that the risks found in the audit process are within the tolerable range.

# 6. Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility base on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance this report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.


**Official Website**
www.slowmist.com

**E-mail**
team@slowmist.com

**Twitter**
@SlowMist_Team

**WeChat Official Account**