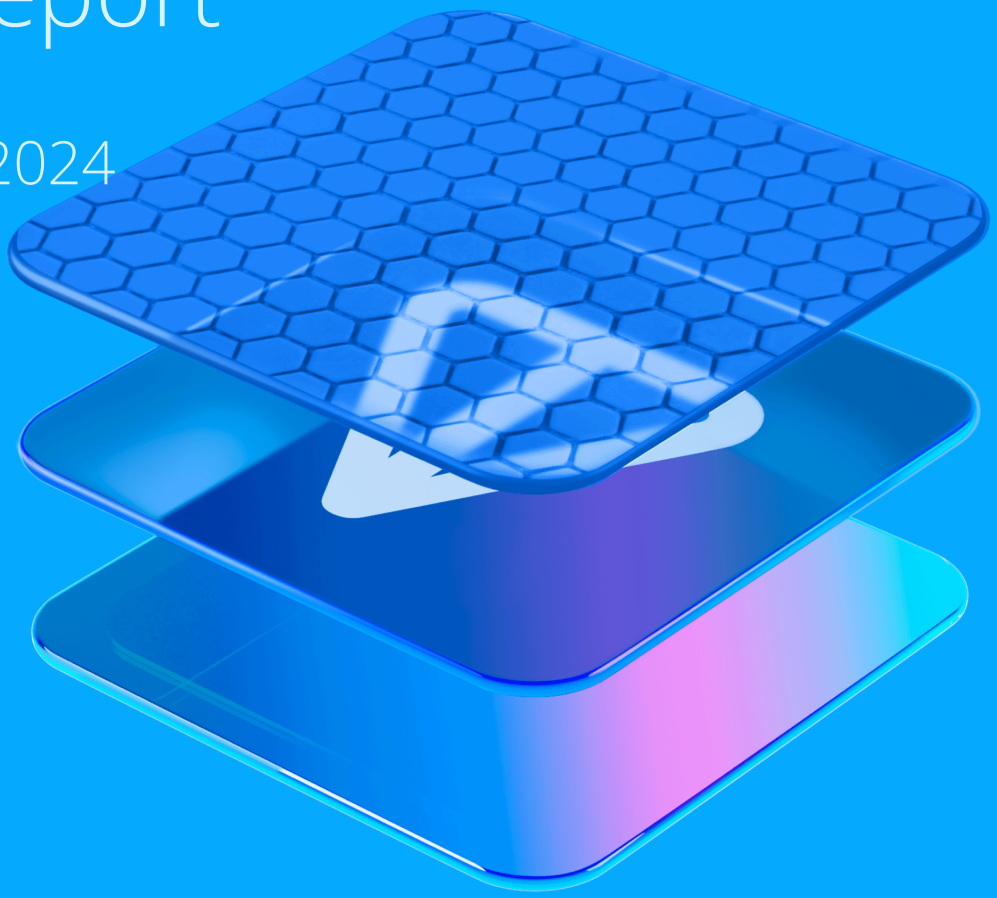
PixelSwap Audit Report

Tue Jun 04 2024



contact@bitslab.xyz https://twitter.com/tonbit_

TonBit

PixelSwap Audit Report

1 Executive Summary

1.1 Project Information

Description	PixelSwap is a decentralized trading platform that offers the ability to exchange, store, and add liquidity
Type	Dex
Auditors	TonBit
Timeline	Thu May 09 2024 - Tue Jun 04 2024
Languages	Tact
Platform	Ton
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/nx-fi/pixelswap
Commits	a66e45d9fbe069663bf55ac2f29a14f1d0405a57 5009982f22f0542cc48fd66c22627a6e60710f58

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
TIM	contracts/utils/time.tact	f5a4d3671f4c83d5c67fdd4e66e118008b39d611
NAT	contracts/utils/native.tact	0bfcc9059d04217cb994b8d8373263e307e14983
EXT	contracts/utils/extra.fc	ee4478fb3a07ad1914df8076b79ce3f57dda3a46
OPT	contracts/utils/option.tact	f475219694d5452d7379ff51c4793768d4659071
PSE	contracts/pixelswap_settlement.tact	e2407ac54af9c02d9b3243eb1639c3a3cb068d06
MSG	contracts/utils/msgtools.tact	e3696da9c393966d147d6dadacfd4622709e0552
ACSR	contracts/utils/access_control_single_role.tact	075e011129e480f538b3e2ebfe0dcd8f82be0187
MAT	contracts/utils/math.tact	8e1bd211dde901cd56e1100a248f7d09547f9ee8
PSR	contracts/utils/pausable_single_role.tact	aaa0369277b813b367e694255650287e1b8c2243
ADD	contracts/utils/address.tact	f6e157b8df5e0cf3146b94a9d4b4577d254ea7ae
AFR	contracts/utils/afr.tact	5d77afafce7d13560c5b2bb2442d3f8f9776b6f9

PFU	contracts/pixelswap_funding.tact	7d04d3b9c6bf6d67ac0532413cfa6e89d0e0c91c
PME	contracts/pixelswap_messages.tact	7d670a5d61df3181aaeb6e48168455c2eb5906a8
PSM	contracts/pixelswap_streampool_messages.tact	c9c6c7ace0b2e2c9ad1bac1fdfc051b1b0e8a871
JFA	contracts/jetton/jetton_factory.tact	87b1dad93442a061a2d52027db9e8be6c0d8f051
PST	contracts/pixelswap_streampool.tact	7100c0b5a447b64fc445424708e98635702852ee
U2M	contracts/univ2_math.tact	0e361e744d0c463a61819bb0f8e9fbc2468fbc17
PSD	contracts/pixelswap_streampool_data.tact	fb85d5743e1308f4e1eb2b58575538f140559565

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	11	8	1
Informational	2	2	0
Minor	1	1	0
Medium	6	5	0
Major	2	0	1
Critical	0	0	0

1.4 TonBit Audit Breakdown

TonBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow by bit operations
- Number of rounding errors
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting
- Unchecked CALL Return Values

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by PixelSwap to identify any potential issues and vulnerabilities in the source code of the PixelSwap smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 11 issues of varying severity, listed below.

ID	Title	Severity	Status
JFA-1	Unable to Normalise `burn` LP Tokens	Medium	Pending
JFA-2	Single-step Ownership Transfer Can be Dangerous	Medium	Fixed
MSG-1	Replace + with Bitwise OR (|)	Medium	Fixed
PFU-1	Controllable `msg.user` May Result in Loss of User Funds	Medium	Fixed
PSE-1	Users Lack the Incentive to Provide Liquidity	Major	Acknowledged
PST-1	No Direct `return` After Caller Checksums	Major	Pending
PST-2	Security Vulnerability Due to Lack of `enable_adding_liquidity` Field Validation in `PairConfig`	Medium	Fixed
PST-3	Misleading Error Message in Token Creation Validation	Minor	Fixed
PST-4	Incorrect Exception Message	Informational	Fixed

PST-5	Typo Found in `weight0` Check Error Message	Informational	Fixed
U2M-1	No Swap Fees Charged	Medium	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the PixelSwap Smart Contract :

Owner

- The owner can send an `AddFundContract` message to the `PixelswapSettlement` contract to add a new `funding` contract address
- The owner can send an `AddExecContract` message to the `PixelswapSettlement` contract to add a new `Execution` contract address
- The owner can send a `SetDefaultFundContract` message to the `PixelswapSettlement` contract to set the default `funding` contract address
- The owner can send a `SetDefaultExecContract` message to the `PixelswapSettlement` contract to set the default `Execution` contract address
- The owner can send a `SetGasConfig` message to the `PixelswapSettlement` contract to set a new `gas` configuration
- The owner can send a `SetPoolConfig` message to the `PixelswapStreamPool` contract to set the pool's parameter configuration
- The owner can send a `SetPoolGasConfig` message to the `PixelswapStreamPool` contract to set the configuration of the pool with respect to `gas`
- The owner can send a `SetTokenActiveStatus` message to the `PixelswapStreamPool` contract to set the token's active status
- The owner can send a `SetPairActiveStatus` message to the `PixelswapStreamPool` contract to set the active status of the `pair`
- The owner can send a `SetPairLockStatus` message to the `PixelswapStreamPool` contract to set the `lock` status for the `pair`
- The owner can send a `SetPairEnableAddingLiquidityFlag` message to the `PixelswapStreamPool` contract to set adding liquidity status for `pair`
- The owner can send a `SetPairProtocolFee` message to the `PixelswapStreamPool` contract to set the rate of the corresponding fee for each `pair` .

Admin

- The administrator can send a `pause` message to the `PixelswapSettlement` contract to suspend contract usage
- The administrator can resume contract usage by sending a `resume` message to the `PixelswapSettlement` contract

Validator

- The validators can send a `CollectProtocolFee` message to the `PixelswapStreamPool` contract to collect the fees generated by the token pairs
- The validators can send a `CreateToken` message to the `PixelswapStreamPool` contract to create tokens without charging any fees
- The validators can send a `CreateTradingPair` message to the `PixelswapStreamPool` contract to create trading pairs without charging any fees

User

- The users can send a `PlaceOrder` message to their own `PixelswapFundingWallet` contract to use the token balance in their `PixelswapFundingWallet` contract to send a `PlaceOrder` message to the `PixelswapSettlement` contract for exchanges and adding liquidity operations
- The users can send a `WithdrawFunds` message to their own `PixelswapFundingWallet` contract to withdraw previously deposited tokens
- The users can send an `InternalTransfer` message to their own `PixelswapFundingWallet` contract to transfer token balances to other addresses
- The users can send a `BulkInternalTransfer` message to their `PixelswapFundingWallet` contract to transfer tokens in bulk.
- The users can send a `RescueJetton` message to their own `PixelswapFundingWallet` contract to retrieve mistakenly transferred `Jetton`
- The users can send a `withdraw dust` message to their own `PixelswapFundingWallet` contract to withdraw all `TON` except for the `gas` fee from the contract
- The users can send the corresponding `Jetton` to the `JettonWallet` associated with the `PixelswapSettlement` contract and include `forward_ton_amount` and the specified `either_forward_payload` to send a `OrderJettonNotification` message to the `PixelswapSettlement` contract for deposit, exchange, and adding liquidity operations
- The users can send a `Swap` message to the `PixelswapSettlement` contract to exchange `TON` for `Jetton`

- The users can send a `Deposit` message to the `PixelswapSettlement` contract to deposit `TON` into the contract
- The users can send a `WithdrawDustFromFunding` message to the `PixelswapSettlement` contract to withdraw the excess `TON` (excluding gas fees) from the `PixelswapFunding` contract to the `PixelswapSettlement` contract
- The users can send a `WithdrawDustFromExec` message to the `PixelswapSettlement` contract to withdraw the excess `TON` (excluding gas fees) from the `PixelswapStreamPool` contract to the `PixelswapSettlement` contract
- The users can send the corresponding LP tokens to the `JettonWallet` associated with the `PixelswapStreamPool` contract and include `forward_ton_amount` and the specified `either_forward_payload` to send a `RemoveLiquidityJettonNotification` message to the `PixelswapStreamPool` contract to remove liquidity
- The users can send a `CreateToken` message to the `PixelswapStreamPool` contract, including the creation fee, to create new tokens
- The users can send a `CreateTradingPair` message to the `PixelswapStreamPool` contract, including the creation fee, to create new trading pairs

4 Findings

JFA-1 Unable to Normalise `burn` LP Tokens

Severity: Medium

Status: Pending

Code Location:

contracts/jetton/jetton_factory.tact#88-101

Descriptions:

In the `burn` function, the contract sends the `JETTONFACTORY__JettonBurn` message to the `jetton` contract, but the `jetton` contract doesn't receive this message, and the contract can't destroy its LP tokens properly.

```
fun burn(pair_id: Int, amount: Int) {
        send(SendParameters {
            to: self.jetton_address_of(pair_id),
            value: self.gas_send_burn,
            mode: SendIgnoreErrors,
            body: JETTONFACTORY__JettonBurn {
                query_id: 0,
                amount: amount,
                response_destination: myAddress(),
                custom_payload: null
            }.toCell(),
            bounce: false
        });
    }
```

Suggestion:

It is recommended that the `burn` function be changed to normal.

JFA-2 Single-step Ownership Transfer Can be Dangerous

Severity: Medium

Status: Fixed

Code Location:

contracts/jetton/jetton_factory.tact#10

Descriptions:

Single-step ownership transfer means that if a wrong address was passed when transferring ownership or admin rights it can mean that role is lost forever. If the admin permissions are given to the wrong address within this function, it will cause irreparable damage to the contract. Below is the official documentation explanation from OpenZeppelin https://docs.openzeppelin.com/contracts/4.x/api/access

`Ownable is a simpler mechanism with a single owner "role" that can be assigned to a single account. This simpler mechanism can be useful for quick tests but projects with production concerns are likely to outgrow it.`

The `jetton_factory` contract references the ownable library, which has a single-step ownership transfer process. This is quite risky.

```
@interface("org.ton.ownable.transferable.v2")
trait OwnableTransferable with Ownable {

    owner: Address;

    receive(msg: ChangeOwner) {

        // Check if the sender is the owner
        self.requireOwner();

        // Update owner
        self.owner = msg.newOwner;

        // Reply result
        self.reply(ChangeOwnerOk{ queryId: msg.queryId, newOwner:msg.newOwner
}.toCell());
    }
```

Suggestion:

It is recommended to use a two-step ownership transfer pattern

Resolution:

This issue has been fixed.

MSG-1 Replace + with Bitwise OR (|)

Severity: Medium

Status: Fixed

Code Location:

contracts/utils/msgtools.tact#112

Descriptions:

The `msgtools.send_and_deploy()` function is designed to send a message to a specified address and deploy a contract. Within this function, the protocol sets the mode parameter of the `send()` function to `SendRemainingValue + SendIgnoreErrors`.

```
extends inline fun send_and_deploy(self: Cell, toinit: StateInit) {
    send(SendParameters{
        to: contractAddress(toinit),
        value: 0,
        bounce: false,
        mode: SendRemainingValue + SendIgnoreErrors,
        body: self,
        code: toinit.code,
        data: toinit.data
    });
}
```

However, according to the official documentation, it is recommended to use the bitwise OR operator (|) instead.

```
Note, that while adding (+) base
modes together with optional flags
is possible, it is discouraged due
to the possibility of excess values.
Use the bitwise OR (|) instead, as
it's designed to work with such flag
and bit manipulations of the mode.
```

Suggestion:

It is recommended to use the bitwise OR (|) instead

Resolution:

This issue has been fixed. The client has adopted our suggestions.

PFU-1 Controllable `msg.user` May Result in Loss of User Funds

Severity: Medium

Status: Fixed

Code Location:

contracts/pixelswap_funding.tact#209

Descriptions:

In the `InternalTransfer` message in the `PixelswapFundingWallet` contract, `msg.user` is passed in by the user and sent directly to the `InternalTransfer` message in the `PixelswapFunding` contract for `require` validation. If the user mistakenly passes in the wrong `msg.user` address, an exception is thrown in the `InternalTransfer` message of the `PixelswapFunding` contract to terminate the execution, but by this time, the `PixelswapFundingWallet` contract has already deducted the user's balance, resulting in the loss of the user's balance.

```
require(sender() == self.settlement || sender() == self.get_wallet_address(msg.user), "Unauthenticated sender"); // afr::allow-trap-irrelevant-path
```

The same problem exists with the `BulkInternalTransferSameAmount` message and the `BulkInternalTransferDifferentAmounts` message.

Suggestion:

It is recommended to adjust the implementation based on the business logic.

Resolution:

This issue has been fixed.

PSE-1 Users Lack the Incentive to Provide Liquidity

Severity: Major

Status: Acknowledged

Code Location:

contracts/pixelswap_settlement.tact#576

Descriptions:

Users' swap fees are stored in `pair_status.token0_fees_payable` or `pair_status.token1_fees_payable`.

```
if (side) {
        res = calc_swap_exact_amount_in(amount_in, pair_status.reserve0,
pair_status.reserve1, pair_config.fee_bps, 0);
        if (res.amount_out < min_amount_out || pair_status.reserve1 - res.amount_out <
self.MIN_TOKENS) {
            return SwapResult{amount_in: 0, amount_out: 0, fee: 0};
        }
        // update pair status
        pair_status.reserve0 = pair_status.reserve0 + res.amount_in;
        pair_status.reserve1 = pair_status.reserve1 - res.amount_out;
        pair_status.token0_fees_payable = pair_status.token0_fees_payable + res.fee;
        self.pairs_status.set(pair_id, pair_status);
    } else {
        res = calc_swap_exact_amount_in(amount_in, pair_status.reserve1,
pair_status.reserve0, pair_config.fee_bps, 0);
        if (res.amount_out < min_amount_out || pair_status.reserve0 - res.amount_out <
self.MIN_TOKENS) {
            return SwapResult{amount_in: 0, amount_out: 0, fee: 0};
        }
        // update pair status
        pair_status.reserve1 = pair_status.reserve1 + res.amount_in;
        pair_status.reserve0 = pair_status.reserve0 - res.amount_out;
        pair_status.token1_fees_payable = pair_status.token1_fees_payable + res.fee;
        self.pairs_status.set(pair_id, pair_status);
    }
```

Authorized users can send a `CollectProtocolFee` message to `pixelswap_streampool` to collect the protocol fee.

```
receive(msg: CollectProtocolFee) {
        self.require_role(sender()); // afr::allow-trap-at-input-boundary
        let pair_config: PairConfig = self.pairs_config.get(msg.pair_id)!!; // afr::allow-trap-at-input-boundary
        let pair_status: PairStatus = self.pairs_status.get(msg.pair_id)!!; // afr::allow-trap-at-input-boundary
        let token0_config: TokenConfig = self.tokens_config.get(pair_config.token0_id)!!; // afr::allow-trap-at-input-boundary
        let token1_config: TokenConfig = self.tokens_config.get(pair_config.token1_id)!!; // afr::allow-trap-at-input-boundary

        OrderExecutionResult {
            fund_id: msg.fund_id,
            exec_id: self.exec_id,
            token0_id: pair_config.token0_id,
            token0_amt: pair_status.token0_fees_payable,
            token1_id: pair_config.token1_id,
            token1_amt: pair_status.token1_fees_payable,
            user: self.protocol_fee_recipient,
            output_to_ext: true,
            chaining_mode: 0,
            rem: self.build_rem_payload_output_to_ext(true, token0_config, token1_config)
        }.toCell().send_to(self.settlement);
        pair_status.token0_fees_payable = 0;
        pair_status.token1_fees_payable = 0;
        self.pairs_status.set(msg.pair_id, pair_status);
    }
```

This means that users do not earn any rewards when adding liquidity, resulting in a lack of incentive for users to add liquidity.

Suggestion:

It is recommended to to incentivize users to add liquidity.

PST-1 No Direct `return` After Caller Checksums

Severity: Major

Status: Pending

Code Location:

contracts/pixelswap_streampool.tact#202-204

Descriptions:

When the `RemoveLiquidityJettonNotification` message is received, if the sender is not the jetton wallet address of the specified pair, the jetton is sent back. But there is no direct `return` operation. If the `return` operation is not performed, the following removeLiquidity operation is still performed.

```
if (sender() != self.jetton_wallet_address_of(pair_id, myAddress())) { // sender not authenticated
    sender().transfer_with_text(msg.user, msg.amount, ton("0.01"), ton("0.1"), 0, "Pix: E34");
// "Invalid sender" // TODO: manage gas
}
```

Suggestion:

It is recommended that the `return` operation be added to terminate code execution.

PST-2 Security Vulnerability Due to Lack of `enable_adding_liquidity` Field Validation in `PairConfig`

Severity: Medium

Status: Fixed

Code Location:

contracts/pixelswap_streampool.tact#629

Descriptions:

There is a field called `enable_adding_liquidity` in `PairConfig`, apparently intended to track whether a `pair` allows liquidity addition. However, there is a lack of validation for this field when adding liquidity. This means that even if administrators disable liquidity addition in `pair`, users can still perform this action. This could pose significant security risks.

```
  /// Set enable_adding_liquidity for a pair.
  receive(msg: SetPairEnableAddingLiquidityFlag) {
      self.requireOwner(); // afr::allow-trap-at-input-boundary
      let pair_config: PairConfig = self.pairs_config.get(msg.pair_id)!!; // afr::allow-trap-at-input-boundary
      require(pair_config.enable_adding_liquidity != msg.enable, "No changes"); // afr::allow-trap-at-input-boundary
      pair_config.enable_adding_liquidity = msg.enable;
      self.pairs_config.set(msg.pair_id, pair_config);
  }
```

Suggestion:

It is recommended to implement proper validation for the `enable_adding_liquidity` field in `PairConfig` when adding liquidity, ensuring that user actions align with administrator settings to mitigate potential security vulnerabilities.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

PST-3 Misleading Error Message in Token Creation Validation

Severity: Minor

Status: Fixed

Code Location:

contracts/pixelswap_streampool.tact#357

Descriptions:

In the validation of `CreateToken` messages, the error message `Pair creation not possible` is displayed, which does not accurately reflect the actual error of whether token creation is allowed. This discrepancy could confuse message senders.

```
require(self.has_role(sender()) || self.config.enable_token_creation, "Pair creation not possible");
```

Suggestion:

It is recommended to update the error message in the validation of `CreateToken` messages to accurately reflect whether token creation is allowed or not, providing clear guidance to message senders and reducing confusion.

Resolution:

This issue has been fixed. The client has corrected the erroneous description.

PST-4 Incorrect Exception Message

Severity: Informational

Status: Fixed

Code Location:

contracts/pixelswap_streampool.tact#255

Descriptions:

Exception message error in `WithdrawDustFromExec` message.

```
require(msg.exec_id == self.exec_id, "Invalid fund ID"); // afr::allow-trap-impossible-path
```

Suggestion:

It is recommended that `fund` be changed to `exec`.

Resolution:

This issue has been fixed.

PST-5 Typo Found in `weight0` Check Error Message

Severity: Informational

Status: Fixed

Code Location:

contracts/pixelswap_streampool.tact#380

Descriptions:

There is a typo in the error message displayed during the `weight0` check, where the word `Weight0t0` is incorrectly spelled. This could potentially confuse readers.

```
require(msg.weight0 > 0 && msg.weight0 < 100, "Weight0t0 must be between 1% and 99%");
```

Suggestion:

It is recommended to correct the typo in the error message during the `weight0` check to avoid confusion.

Resolution:

This issue has been fixed. The client has provided the revised text with corrections.

U2M-1 No Swap Fees Charged

Severity: Medium

Status: Fixed

Code Location:

contracts/univ2_math.tact#33-40

Descriptions:

With each `swap_exact_amount_in()` and `swap_exact_amount_out()` transaction, the protocol increases the `pair_status.token0_fees_payable` or `pair_status.token1_fees_payable` value, a user with a certain role can send a `CollectProtocolFee` message to take those funds from the `settlement`, and the vault's `token 0` and `token1` balances change, while `pair_status.reserve0` and `pair_status.reserve1` do not change accordingly. `did not change accordingly. That is, the number of tokens added by the user for liquidity does not match reserve0 and reserve1 and the real balance becomes smaller. When the pool liquidity is very low, the amount calculated according to reserve0 and reserve1`when the user removes the liquidity may be larger than the current balance of the pool, resulting in the user not being able to remove the liquidity.

Suggestion:

It is recommended to adjust the implementation based on the business logic.

Resolution:

This issue has been fixed.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.