verichains

SECURITY AUDIT OF

SENSWAP

Public Report

Dec 21, 2023

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Solana	A decentralized blockchain built to enable scalable, user-friendly apps for the world.
SOL	A cryptocurrency whose blockchain is generated by the Solana platform.
Lamport	A fractional native token with the value of 0.000000001 SOL.
Program	An app interacts with a Solana cluster by sending it transactions with one or more instructions. The Solana runtime passes those instructions to program.
Instruction	The smallest contiguous unit of execution logic in a program.
Cross-program invocation (CPI)	A call from one smart contract program to another.
Anchor	A framework for Solana's Sealevel runtime providing several convenient developer tools for writing smart contracts.

EXECUTIVE SUMMARY

This Security Audit Report was prepared by Verichains Lab on Dec 21, 2023. We would like to thank the Sentre for trusting Verichains Lab in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the SenSwap. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team had identified some vulnerable issues in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About SenSwap

SenSwap is an open liquidity protocol built on Solana that promotes Zero Impermanent Loss and Liquidity Efficiency.

Solana is chosen to lay the foundation for the next generation of DeFi by ultimately ensuring high security, decentralization but greatly improved scalability, up to 65,000 TPS (transactions per second), and 400ms block times without applying complex solutions. Transaction fees on the Solana platform are among the cheapest, only around $0.00001.

This overcomes the obstacles of transaction fees and the increase in time per transaction, problems where ETH has yet to find an optimal solution.

Besides that, SenSwap tackles the challenge of Open Platform and Open Liquidity:

Open Platform. People can introduce their ideas to others natively on the platform.

- Sen offers developers a complete solution for DApp development.
- Developers can deliver DApps via the Sen Store.
- Users can organize favorite DApps for an optimal workflow on a single page.

Open Liquidity. Being open in terms of use and development.

- An AMM with dual and triad pools will become a liquidity accumulator.
- SEN becomes a universal interface for the entire ecosystem (i.e., price evaluation, liquidity automation)
- DAO allows people to control liquidity appropriately.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the smart contracts of SenSwap. It was conducted on commit `260147669c934d3551f41c90de981b398280ac16` from git repository *https://github.com/DescartesNetwork/balancer-amm*.

The latest version of the following file was made available in the course of the review:

SHA256 Sum	File
`e9494b51b657a1ae013f4434ca4c9df6955cd5a3a37261b7bd0e915bbfb8a36a`	`balancer-amm/src/instructions/update_actions.rs`
`aef9ace25ac7f2686a63d2cf837f307054e19530000477cea977788b0d0689d2`	`balancer-amm/src/instructions/initialize_pool.rs`

e2d286f3ca0f0e7bac1791fbf0a7d96993246cd559d25dcfc209676a8f30da8d	balancer-amm/src/instructions/swap.rs
eb7c55ec9196f8f6d0abbbf504c90e97863f6cee4d20b1e8c3bed5c41e96a291	balancer-amm/src/instructions/freeze_pool.rs
cea453c15d213707f248d5e34bdef528a34311c8b8721a709dfe037bfa8b6a1f	balancer-amm/src/instructions/safe_transfer.rs
46f71a395878005ec8ee22c9624eddf5b108274ff18445ad98cac1431d612d85	balancer-amm/src/instructions/route.rs
3599e9628859a37b0de9c32abe7ee63536d4b9513fd3a74a3f96dda6397895f6	balancer-amm/src/instructions/close_pool.rs
dcbf29a080ab47a8ecd09793db535155b1a349c8ab9f86e59bd658a2efdf9b2d	balancer-amm/src/instructions/initialize_join.rs
308af7b602a4efa7623decbeebeaf387dcf4cc014dd5785a2d014cc76199bce0	balancer-amm/src/instructions/thaw_pool.rs
51fccc53a97c419fb7d2db0d1e343608590303517486bd3ff5566aa317edae49	balancer-amm/src/instructions/mod.rs
902cfbedcbd65e0c7c39e2f5ee1bc574a89351f301b0035a78b65b388c3e3cfa	balancer-amm/src/instructions/update_weights.rs
4866fa11acc371387ff3970256b5716fb052beb966eb23116f747620a5ce24a1	balancer-amm/src/instructions/update_fee.rs
0e8f2fb2c12f4eeef91f64e014e8b97a80bf4edd870a82f27bea783f8aca4fae	balancer-amm/src/instructions/add_liquidity.rs
ca9185dbd40a190e7e5a4702f3cffa089858d330a96676fe631c86c533041de2	balancer-amm/src/instructions/transfer_ownership.rs
cde9f922f1990f8f6d87df02a7220df8dfe157686b928e229240bdfc4ba015bd	balancer-amm/src/instructions/remove_liquidity.rs
349d4750107e29952c7ec22c1e8e0881e10f1a8ef58d787d117510f447fcf350	balancer-amm/src/instructions/finalize_pool.rs

3ea13b6b89ab7ee1eb6fad2eb68cf574d517f98cf29a5707f0e9834c374dcafa	balancer-amm/src/lib.rs
0d531d57d6646605afa5ba0ce36fad6530fad87ef62109af1d64bde52f99a4ba	balancer-amm/src/math/f64_trait.rs
f1d9489fa9fc4d707b405bacb517475031c482a1f8342f5178c5c580fb16d8f7	balancer-amm/src/math/mod.rs
91bea62f965a8ba63effd9731fb8681484ecfcce42564310a282981a915ca69b	balancer-amm/src/math/oracle.rs
6b1030c388e5a7230e67e8400c9d51d7952d7e8a2c9bfa0adaca07eaf680dd91	balancer-amm/src/schema/mod.rs
031596a4225d4847b2680006c5827b2e59a0ce688a346a47b1940e266304a21a	balancer-amm/src/schema/pool_trait.rs
d77a6bbd91ea8f23477d37165dabc8929b311ed1a8bebcc9f40c9aeee5e100c1	balancer-amm/src/schema/pool.rs
43901f83b3c4971286d372c9ad3893f218d1f2be7213c3c4bf53718ef18feeae	balancer-amm/src/constant.rs
af39ad8da0fcb818de2b3c588013d81013821ca56c3e7386ba947a0a6eac1e57	balancer-amm/src/errors.rs
ded851878585c99b62dfb222550ba255fb88ad49e3ce814ea4af1644808c98e9	balancer-amm/src/utils.rs

1.3. Audit methodology

Our security audit process for Solana smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using our in-house smart contract security analysis tool.
- 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 were considered during the audit of the Solana smart contract:

- Arithmetic Overflow and Underflow
- Signer checks
- Ownership checks
- Rent exemption checks
- Account confusions

- Bump seed canonicalization
- Closing account
- Signed invocation of unverified programs
- Numerical precision errors
- Logic Flaws

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 1. Severity levels

1.4. Disclaimer

Sentre acknowledges that the security services provided by Verichains, are conducted to the best of their professional abilities but cannot guarantee 100% coverage of all security vulnerabilities. Sentre understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, Sentre agrees that Verichains shall not be held responsible or liable, and shall not be charged for any hacking incidents that occur due to security vulnerabilities not identified during the audit process.

1.5. Acceptance Minute

This final report served by Verichains to the Sentre will be considered an Acceptance Minute. Within 7 days, if no any further responses or reports is received from the Sentre, the final report will be considered fully accepted by the Sentre without the signature.

2. AUDIT RESULT

2.1. Overview

The SenSwap (`balancer_amm` program) was written in `Rust` programming language and `Anchor` framework.

SenSwap is a Balancer-like AMM on Solana. The AMM implementation is heavily relying on Balancer's Whitepaper.

Besides that, the Sentre team also add some extra features to help leverage others on-top application in the future. Below is the overview of the SenSwap flow:

2.1.1. Initialize Pool

To initialize a pool, the pool owner needs to call `initialize_pool` first. The function will ask for basic info like mints, treasuries, weights, etc., to create a pool account and store these info.

Note that `initialize_pool` won't create any accounts (except the pool account) and keep the status of `PoolState::Uninitialized` until the pool when owner calls `initialize_join`.

The `taxman` account can only be set by the pool owner. The `taxman` account is the account that will receive the tax fee from the pool.

Based on the current pool state, we can have the corresponding action table as below (some actions can be invoked by the pool owner only):

Pool state	Next possible actions	Description
`Uninitialized`	`initialize_join`, `close_pool`	Set by `initialize_pool`
`Initializing`	`add_liquidity`, `remove_liquidity`, `swap`, `finalize_pool`, `freeze_pool`, `update_weights`	Set by `initialize_join`, `thaw_pool`
`Initialized`	`add_liquidity`, `remove_liquidity`, `swap`	Set by `finalize_pool`
`Frozen`	`thaw_pool`	Set by `freeze_pool`
`Deleted`	None	Set by `close_pool`

2.1.2. Initialize Join

To create the treasuries (aka token accounts) corresponding to the mints, and also deposit the initial amount of tokens into the pool, the pool owner must call `initialize_join` for all mints

one by one. After all, the pool will transmit the status from `PoolState::Uninitialized` to `PoolState::Initializing` and there are a number LP initialized for the pool owner aka the first liquidity provider.

2.1.3. Close Pool

In case of incorrect configs, the pool owner can cancel the pool by `close_pool`. This function is only possible when the pool is `PoolState::Uninitialized`.

The current pool limit is 8 mints.

2.1.4. Finalize Pool

Once the pool state is set to `PoolState::Initializing`, only the pool owner is able to update weights, add/remove liquidity. This limit is to avoid rug pull and build a foundation for liquidity bootstrapping (aka. launchpad).

To open the pool to the public, the pool owner must call `finalize_pool` to set the pool state from `PoolState::Initializing` to `PoolState::Initialized`. At that time, the public can join and add/remove liquidity to the pool.

2.1.5. Add Liquidity

To add liquidity to the pool, the user must call `add_liquidity` with the amount of tokens to deposit. By adding liquidity, the user will become a liquidity provider. However, `add_liquidity` differs from `initialize_pool` that it requires all token deposited at once. In exchange, the liquidity provider will receive a corresponding number of LP tokens.

2.1.6. Remove Liquidity

The liquidity providers can return LP tokens via `remove_liquidity` to get back their deposited tokens. The `remove_sided_liquidity` instruction has been removed for the sake of simplicity, this instruction is supported by the original version of Balancer.

2.1.7. Swap

Users can run a swap by calling swap. There exists fees for each transaction. You can find more in Fee & Tax.

2.1.8. Route

The `route` instruction is a high-level abstract function of `swap`. Users can call multiple `swap`-s in a monolithic transaction of `route`. The basic idea is that `route` will verify params for each `swap` then self-invoke the program by calling `swap`. This function is really helpful for AMM aggregators.

2.1.9. Fee & Tax

On each swap, there are fees that the trader must pay:

- Liquidity Provision Fee (aka fee):
  - IT'S COLLECTED OVER BIDDING TOKENS.
  - This fee is to incentivize people to provide liquidity into the pool and avoid permanent loss.
- Platform Fee (aka tax):
  - IT'S COLLECTED OVER ASKING TOKENS.
  - This tax is to help SenSwap maintain the system and develop more features to the platform.
  - However, the tax is not only for the SenSwap foundation, but also being structured for the referral system. When referrer addresses are injected in a transaction, the tax will be split equally for the platform fee and referral fees. For example, there are 3 referrer addresses in a swap transaction and 100 tokens for the tax, then the platform fee will be 25 tokens, and 25 tokens for each referrer.

2.2. Findings

During the audit process, the audit team had identified some vulnerable issues in the given version of SenSwap.

Sentre team fixed these issues, according to Verichains's draft report, in commit `c946b7a4e4134f862ec6ccbf95c55adc5a8a205c`.

#	Severity	Name	Status
1	HIGH	Missing constraint for `taxman` in `remove_sided_liquidity` instruction	FIXED
2	HIGH	Incorrect `GROUP_ACCOUNTS` for `close_pool` instruction	FIXED
3	HIGH	Pool owner can update weights without restrictions	FIXED
4	LOW	Number of referrers is not limited	FIXED
5	LOW	Taxman address can be set by the pool creator	ACKNOWLEDGED
6	INFORMATIVE	Integer underflow in `route` instruction	FIXED

#	Severity	Name	Status
7	INFORMATIVE	Inconsistent fee calculation in multiple instructions	FIXED

2.2.1. Missing constraint for `taxman` in `remove_sided_liquidity` instruction HIGH

Affected files:

- instructions/remove_sided_liquidity.rs

In the `remove_sided_liquidity` instruction, the `taxman` account is not required to be the same as the `pool.taxman` account. As a result, the tax fees can be withdrawn to any account.

```
#[derive(Accounts)]
pub struct RemoveSidedLiquidity<'info> {
  #[account(mut)]
  pub authority: Signer<'info>,
  // Pool info
  #[account(mut, has_one = mint_lpt)]
  pub pool: Account<'info, Pool>,
  #[account(mut)]
  /// CHECK: Just a pure account
  pub taxman: AccountInfo<'info>,
  #[account(
    seeds = [b"treasurer", &pool.key().to_bytes()],
    bump
  )]
  /// CHECK: Just a pure account
  pub treasurer: AccountInfo<'info>,
  #[account(mut)]
  pub mint_lpt: Account<'info, token::Mint>,
  #[account(
    init_if_needed,
    payer = authority,
    associated_token::mint = mint_lpt,
    associated_token::authority = authority
  )]
  pub associated_token_account_lpt: Box<Account<'info, token::TokenAccount>>,

  pub mint: Box<Account<'info, token::Mint>>,
  #[account(
    mut,
    associated_token::mint = mint,
    associated_token::authority = treasurer
  )]
  pub treasury: Box<Account<'info, token::TokenAccount>>,
  #[account(
    init_if_needed,
```

```
    payer = authority,
    associated_token::mint = mint,
    associated_token::authority = authority
  )]
  pub dst_associated_token_account: Box<Account<'info, token::TokenAccount>>,
  #[account(
    init_if_needed,
    payer = authority,
    associated_token::mint = mint,
    associated_token::authority = taxman
  )]
  pub associated_token_account_taxman: Box<Account<'info, token::TokenAccount>>,

  // Programs
  pub system_program: Program<'info, System>,
  pub token_program: Program<'info, token::Token>,
  pub associated_token_program: Program<'info, associated_token::AssociatedToken>,
  pub rent: Sysvar<'info, Rent>,
}
```

UPDATES

- *Dec 21, 2023*: This issue has been acknowledged and fixed by Sentre team.

2.2.2. Incorrect `GROUP_ACCOUNTS` for `close_pool` instruction HIGH

Affected files:

- instructions/close_pool.rs

The value of `GROUP_ACCOUNTS` is incorrect. It should be 3 instead of 2. As a result, incorrect account indices are used to access the `remaining_accounts` array, which may cause reverted when closing the pool.

```
const TREASURY_IDX: usize = 0;
const TOKEN_ACC_IDX: usize = 1;
const MINT_IDX: usize = 2;
const GROUP_ACCOUNTS: usize = 2; // INCORRECT: should be 3

// ...
pub fn exec<'a, 'b, 'c, 'info>(ctx: Context<'a, 'b, 'c, 'info, ClosePool<'info>>) ->
Result<()> {
    // ...

    // Drain all the tokens
    for idx in 0..pool.reserves.len() {
        let amount_out = pool.reserves[idx];
        if amount_out > 0 {
            let transfer_ctx = CpiContext::new_with_signer(
                ctx.accounts.this.to_account_info(),
```

```
                safe_transfer::__cpi_client_accounts_safe_transfer::SafeTransfer {
                    payer: ctx.accounts.authority.to_account_info(),
                    mint: ctx.remaining_accounts[GROUP_ACCOUNTS * idx +
MINT_IDX].to_account_info(),
                    src: ctx.accounts.treasurer.to_account_info(),
                    src_token_account: ctx.remaining_accounts[GROUP_ACCOUNTS * idx +
TREASURY_IDX]
                        .to_account_info(),
                    dst: ctx.accounts.authority.to_account_info(),
                    dst_token_account: ctx.remaining_accounts[GROUP_ACCOUNTS * idx +
TOKEN_ACC_IDX]
                        .to_account_info(),
                    system_program: ctx.accounts.system_program.to_account_info(),
                    token_program: ctx.accounts.token_program.to_account_info(),
                    associated_token_program:
ctx.accounts.associated_token_program.to_account_info(),
                    rent: ctx.accounts.rent.to_account_info(),
                },
                seeds,
            );
            // ...
        }
    }
    // ...
  });
  // ...
}
```

UPDATES

- *Dec 21, 2023*: This issue has been acknowledged and fixed by Sentre team.

2.2.3. Pool owner can update weights without restrictions HIGH

Affected files:

- instructions/update_weights.rs

According to the current implementation, since the pool owner can set new weights for the pool at any time, the pool cannot be finalized to become a "finalized pool" as described in the Balancer white paper.

```
#[derive(Accounts)]
pub struct UpdateWeights<'info> {
  pub authority: Signer<'info>,
  // pool info
  #[account(mut, has_one = authority @ErrorCode::InvalidPermission)]
  pub pool: Account<'info, Pool>,
}
```

```
pub fn exec(ctx: Context<UpdateWeights>, weights: Vec<u64>) -> Result<()> {
  let pool = &mut ctx.accounts.pool;

  if weights.len() != pool.mints.len() {
    return err!(ErrorCode::ParamsLength);
  }
  for weight in weights.iter() {
    if *weight == 0 {
      return err!(ErrorCode::ParamsZero);
    }
  }

  // update new weights
  pool.weights = weights;

  emit!(UpdateWeightsEvent {
    authority: ctx.accounts.authority.key(),
    pool: pool.key(),
    weights: pool.weights.clone()
  });

  Ok(())
}
```

UPDATES

- *Dec 21, 2023*: This issue has been acknowledged and fixed by Sentre team.

2.2.4. Number of referrers is not limited LOW

Affected files:

- instructions/swap.rs

The referrer accounts are specified in the `remaining_accounts` when executing the swap instruction. Since the number of `remaining_accounts` is not limited, users can include more referrer accounts that belong to them to decrease the tax amount.

```
pub fn exec<'a, 'b, 'c, 'info>(
    ctx: Context<'a, 'b, 'c, 'info, Swap<'info>>,
    bid_amount: u64,
    limit: u64,
) -> Result<u64> {
    // ...
    // Transfer fee share
    for i in 0..ctx.remaining_accounts.len() {
      let referral_ctx = CpiContext::new_with_signer(
        ctx.accounts.token_program.to_account_info(),
        token::Transfer {
          from: ctx.accounts.dst_treasury.to_account_info(),
```

```
          to: ctx.remaining_accounts[i].to_account_info(),
          authority: ctx.accounts.treasurer.to_account_info(),
        },
        seeds,
      );
      pool.unsafe_transfer_out(
        unit_tax_fee_amount,
        referral_ctx,
        PoolValidate {
          mint_states: vec![MintActionState::Active, MintActionState::AskOnly],
          pool_states: vec![PoolState::Initialized],
        },
      )?;
    }
    // ...
}
```

UPDATES

- *Dec 21, 2023*: This issue has been acknowledged and fixed by limiting the number of referrers to 2.

2.2.5. Taxman address can be set by the pool creator LOW

Affected files:

- instructions/initialize_pool.rs

According to the provided document in the GitHub repository, `taxman` should be the address to collect platform fee. However, pool can be created by anyone and the taxman address is set by the pool owner, pointing to any account. As a result, the platform owner cannot collect the tax fees.

```
#[derive(Accounts)]
pub struct InitializePool<'info> {
  #[account(mut)]
  pub authority: Signer<'info>,
  // pool info
  #[account(init, payer = authority, space = Pool::LEN)]
  pub pool: Account<'info, Pool>,
  #[account(mut)]
  /// CHECK: Just a pure account
  pub taxman: AccountInfo<'info>,
  #[account(seeds = [b"treasurer", &pool.key().to_bytes()], bump)]
  /// CHECK: Just a pure account
  pub treasurer: AccountInfo<'info>,
  // mint_lpt
  #[account(
    init,
    payer = authority,
```

```
    mint::decimals = MINT_LPT_DECIMALS,
    mint::authority = treasurer,
    mint::freeze_authority = treasurer
  )]
  pub mint_lpt: Account<'info, token::Mint>,
  // programs
  pub system_program: Program<'info, System>,
  pub token_program: Program<'info, token::Token>,
  pub associated_token_program: Program<'info, associated_token::AssociatedToken>,
  pub rent: Sysvar<'info, Rent>,
}
```

RECOMMENDATION

The taxman account can be a PDA generated from this program with a corresponding instruction to withdraw the collected tax fees.

UPDATES

- *Dec 21, 2023*: This issue has been acknowledged but not fixed.

2.2.6. Integer underflow in `route` instruction INFORMATIVE

Affected files:

- instructions/route.rs

When the length of `remaining_accounts` is less than `GROUP_ACCOUNTS`, the value of `steps` will be 0, which causes an integer underflow for the `last_step` variable. However, the code will be reverted since the `overflow-checks` in the `Cargo.toml` is enabled. Even if the `overflow-checks` is disabled, the code will still be reverted since `GROUP_ACCOUNTS * last_step + ASK_MINT_IDX` index will be out of the range of `remaining_accounts`. So, we will mark this issue as `INFO`.

```
pub fn exec<'a, 'b, 'c, 'info>(
  ctx: Context<'a, 'b, 'c, 'info, Route<'info>>,
  bid_amount: u64,
  limit: u64,
) -> Result<()> {
    let steps = ctx.remaining_accounts.len() / GROUP_ACCOUNTS;
    let last_step = steps - 1; // ERROR: INTEGER UNDERFLOW
    let mut route_bid_amount = bid_amount;

    for i in 0..steps {
        let route_limit = if i == last_step { limit } else { 0 };
        // ...
    }
    // ...
    emit!(RouteEvent {
        authority: ctx.accounts.authority.key(),
```

```
        bid_mint: ctx.remaining_accounts[BID_MINT_IDX].key(),
        ask_mint: ctx.remaining_accounts[GROUP_ACCOUNTS * last_step + ASK_MINT_IDX].key(),
        bid_amount,
        limit,
        ask_amount: route_bid_amount,
    });

    Ok(())
}
```

UPDATES

- *Dec 21, 2023*: This issue has been acknowledged and fixed by Sentre team.

2.2.7. Inconsistent fee calculation in multiple instructions INFORMATIVE

Affected files:

- math/oracle.rs

The fee for swap sometimes is calculated from the output amount and sometimes is calculated from the input amount. This inconsistency should be clarified.

For example, in the `calc_ask_amount_swap` function, fee is calculated from the output amount.

```
// File: math/oracle.rs
pub fn calc_ask_amount_swap(
    bid_amount: u64,
    bid_reserve: u64,
    bid_weight: f64,
    ask_reserve: u64,
    ask_weight: f64,
    fee: u64,
) -> Option<u64> {
    let _bi_bi_ai = bid_reserve
        .to_f64()?
        .checked_div(bid_reserve.to_f64()?.checked_add(bid_amount.to_f64()?)?)?;
    let _wi_wo = bid_weight.checked_div(ask_weight)?;
    let ask_amount = ask_reserve
        .to_f64()?
        .checked_mul(1_f64.checked_sub(_bi_bi_ai.checked_pow(_wi_wo)?)?)?;
    // fee
    let total_fee = fee.to_f64()?.checked_div(PRECISION_F64)?;
    // FEE IS CALCULATED FROM OUTPUT AMOUNT
    return Some((ask_amount.checked_mul(1_f64.checked_sub(total_fee)?)?).to_u64()?);
}
```

However, in the `calc_received_lpt_amount_after_add_full_side` function, fee is calculated from the input amount.

```
// File: math/oracle.rs
pub fn calc_received_lpt_amount_after_add_full_side(
    supply: u64,
    amounts_in: Vec<u64>,
    weights: Vec<u64>,
    reserves: Vec<u64>,
    total_fee: u64,
) -> Option<u64> {
    // ...
    for idx in 0..amounts_in.len() {
        let balance = reserves[idx].to_f64()?;
        let amount_in = amounts_in[idx].to_f64()?;
        let normalize_weight = normalize_weight(idx, weights.clone())?;

        let mut amount_in_without_fee = amount_in;

        if balance_ratios_with_fee[idx] > invariant_ratio_with_fees {
            let non_taxable_amount =
                balance.checked_mul(invariant_ratio_with_fees.checked_sub(1_f64)?)?;
            let taxable_amount = amount_in.checked_sub(non_taxable_amount)?;
            let fee_ratio = total_fee.to_f64()?.checked_div(PRECISION_F64)?;
            // FEE IS CALCULATED FROM INPUT AMOUNT
            amount_in_without_fee = non_taxable_amount
                .checked_add(taxable_amount.checked_mul(1_f64.checked_sub(fee_ratio)?)?)?;
        }

        let balance_ratio =
((balance.checked_add(amount_in_without_fee))?).checked_div(balance)?;
        invariant_ratio =
invariant_ratio.checked_mul(balance_ratio.checked_pow(normalize_weight)?)?;
    }
    // ...
}
```

UPDATES

- *Dec 21, 2023*: This issue has been acknowledged and fixed by Sentre team.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Dec 21, 2023*	Public Report	Verichains Lab

Table 2. Report versions history