MOVEBIT
Securing the Move Ecosystem

Cetus Concentrated Liquidity Protocol Aptos **Audit Report**

https://twitter.com/movebit_

contact@movebit.xyz

Cetus Concentrated Liquidity Protocol Aptos Audit Report

1 Executive Summary

1.1 Project Information

Type	DEX
Auditors	MoveBit
Timeline	2023-01-04 to 2023-01-30
Languages	Move
Methods	Architecture Review, Unit Testing, Formal Verification, Manual Review
Source Code	Repository: https://github.com/CetusProtocol/cetus-clmm Received Commit: e56d47667850dbc5a9553eddb0f67572e7c3c3b8 Last Reviewed Commit: 9c1e51ec72f31c6743a118c23df74e1097b4c8cc
Updates	2023-01-30, the Cetus Dev team fixed some issues, and explained the other pending issues.

1.2 Issue Statistic

Item	Count	Fixed	Pending
Total	20	17	3
Minor	8	7	1
Medium	10	8	2
Major	1	1	

Critical	1	1	

1.3 Issue Level

- **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.

1.4 Issue Status

- **Fixed:** The issue has been resolved.
- **Pending:** The issue has been acknowledged by the code owner, but has not yet been resolved. The code owner may take action to fix it in the future.

2 Summary of Findings

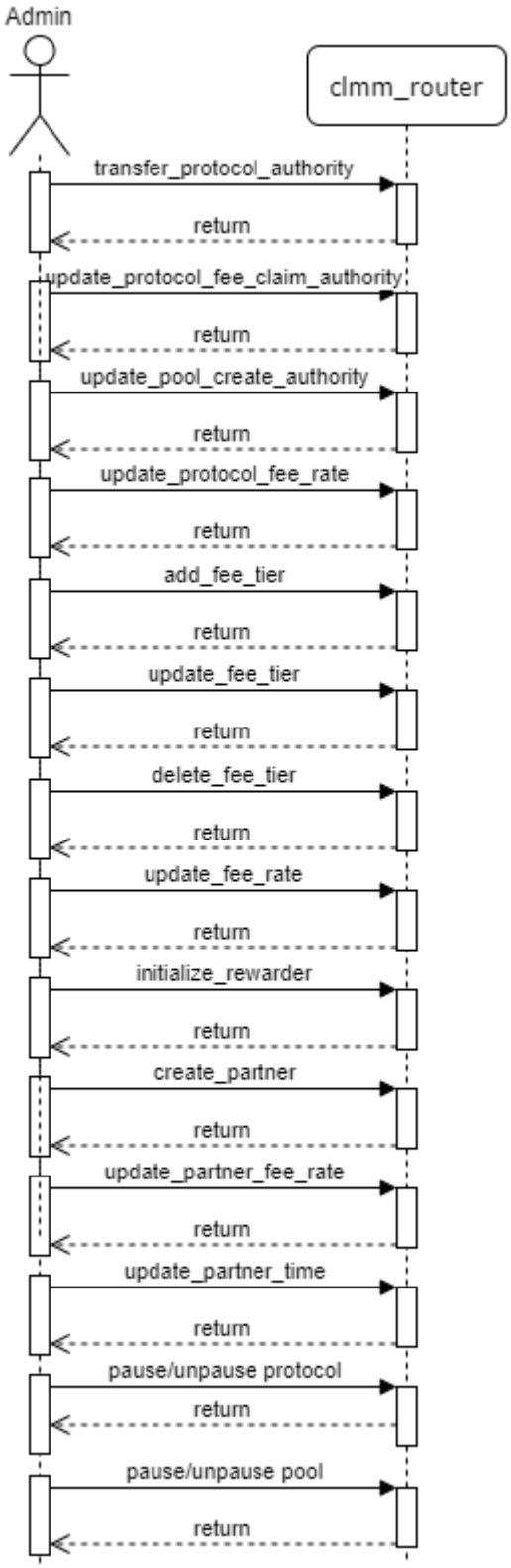
Cetus is a pioneer dex and liquidity protocol built on the Aptos blockchain. It focuses on delivering the best trading experience and superior capital efficiency to DeFi users through the process of building its concentrated liquidity protocol and a series of affiliate functional modules.

The audit team read the documents on https://cetus-1.gitbook.io/cetus-docs/ and reviewed the code of the Cetus Aptos project. The audit team mainly focused on reviewing the code security and normative, then conducted code running tests and business logic security tests on the local test net, and performed a simulation in python which took a deep look at the numeric arithmetic operation. The audit team has been in close contact with the developing team for the past two weeks. As a result, the audit team found a total of 23 issues. The audit team and development team have discussed these issues together, and the development team has addressed most of the issues.

The following are the main roles in the smart contract with their respective capabilities:

(1) Protocol Admin

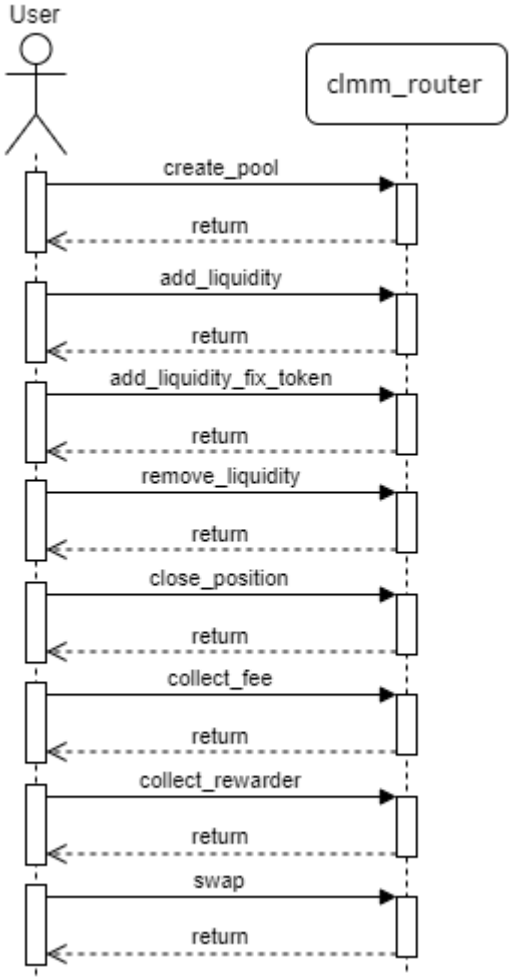
- Protocol Admin can transfer protocol authority to others.
- Protocol Admin can maintain the `fee_tier` .
- Protocol Admin can maintain the `partner` .
- Protocol Admin can maintain the `rewarder` .
- Protocol Admin can pause/unpause the `protocol` .
- Protocol Admin can pause/unpause the `pool` .



(2) User

- User can create a new `pool` .
- User can add liquidity to a `pool` .
- User can remove liquidity from a `pool` .
- User can collect fees from a `pool` .
- User can collect rewards from a `pool` .

- User can trade on a `pool` .



3 MoveBit Audit BreakDown

MoveBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues we 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
- The flow of capability
- Witness Type

4 Methodology

The security team adopted the "**Testing and Automated Analysis**" , "**Code Review**" and "**Formal Verification**" 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 in the conventions in the "Audit Objective", and that can expand to the context 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

Code scope sees **Appendix 1**.

(3) Formal Verification

Perform formal verification for key functions with the Move Prover.

(4) 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, and they should actively cooperate (which may include 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 time.

5 Findings

5.1 Invalid `end_time` argument of `partner::create_partner` may cause `partner::get_ref_fee_rate` to return incorrect fee rate

Severity: Minor
Status: Fixed
Descriptions: `partner::create_partner` doesn't check whether the argument `end_time` is greater than now. It is used to initialize the `PartnerMetadata.end_time` .If the `PartnerMetadata.end_time` is less than now, and not updated by `partner::update_time` later, the partner would always get a zero fee rate returned by `partner::get_ref_fee_rate` , and thus the partner would never receive any partner fee.

Commit: e56d47667850dbc5a9553eddb0f67572e7c3c3b8

Code Location: sources/partner.move, line 141

```
public(friend) fun create_partner(
    account: &signer,
    name: String,
```

```
    fee_rate: u64,
    receiver: address,
    start_time: u64,
    end_time: u64,
) acquires Partners {
    assert!(end_time > start_time, error::aborted(EINVALID_TIME));
    assert!(fee_rate < MAX_PARTNER_FEE_RATE, error::invalid_argument(EINVALID_PARTNER_FEE_RATE));
    ......
}
```

Suggestion: Refer the `partner::update_time`, adding an assertion statement like this `assert!(end_time > timestamp::now_seconds(), error::aborted(EINVALID_TIME));` at the beginning of this function.

5.2 The argument `current_time` of `partner::get_ref_fee_rate` may not be the current time

Severity: Medium
Status: Fixed
Descriptions: `partner::get_ref_fee_rate` is a public function, so everyone can call it. It returns the fee rate based on the input argument `current_time`. It doesn't check `current_time` to match the current time, so the caller can decide what time to pass in to get more benefits.

Commit: e56d47667850dbc5a9553eddb0f67572e7c3c3b8

Code Location: sources/partner.move, line 289

```
public fun get_ref_fee_rate(name: String, current_time: u64): u64 acquires Partners {
    let partners = &borrow_global<Partners>(@cetus_clmm).data;
    if (!table::contains(partners, name)) {
        return 0
    };
    let partner = table::borrow(partners, name);
    if (partner.metadata.start_time > current_time || partner.metadata.end_time <= current_time) {
        return 0
    };
    partner.metadata.fee_rate
}
```

Suggestion: Remove the `current_time` argument, and get the current time by calling `timestamp::now_seconds()` instead.

```
public fun get_ref_fee_rate(name: String): u64 acquires Partners {
    let current_time = timestamp::now_seconds();
    ......
}
```

5.3 Some test cases failed

Severity: Minor
Status: Fixed
Descriptions: While running the test cases, some failed in the `pool` module. For example, the `test_swap` case failed. In module `clmm_math`, the `test_get_next_price_a_down` should be renamed to `test_get_next_price_b_down` as it tests `get_next_sqrt_price_b_down`.

Commit: e56d47667850dbc5a9553eddb0f67572e7c3c3b8

Code Location: sources/pool.move, sources/math/clmm_math.move

Suggestion: Fixing the test cases.

5.4 `tick_math::get_sqrt_price_at_tick` does not check whether the `tick` is in the range

Severity: Medium
Status: Fixed

Descriptions: Some `ticks` are out of range, but still work, for example, `assert!(get_sqrt_price_at_tick(i64::neg_from(443637)) < 4295048016u128, 6)` will not abort although it's out of the tick range . These cases may cause incorrect impacts.

Commit: e56d47667850dbc5a9553eddb0f67572e7c3c3b8

Code Location: sources/math/tick_math.move, line 30

Suggestion: Check whether the tick is in range, and abort if it's out of range.

```
public fun get_sqrt_price_at_tick(tick: i64::I64): u128 {
    assert!(i64::gte(tick, min_tick()) && i64::lte(tick, max_tick()), EINVALID_TICK);
    if (i64::is_neg(tick)) {
        get_sqrt_price_at_negative_tick(tick)
    } else {
        get_sqrt_price_at_positive_tick(tick)
    }
}
```

5.5 Difference between `get_delta_a` and `get_delta_b` in module `clmm_math`

Severity: Minor
Status: Fixed

Descriptions: The `get_delta_b` function has the below check, but `get_delta_a` does not.

```
if ((sqrt_price_diff == 0) || (liquidity == 0)) {
    return 0
};
```

Commit: e56d47667850dbc5a9553eddb0f67572e7c3c3b8

Code Location: sources/math/clmm_math.move, line 55

Suggestion: Adding the checks below.

```
public fun get_delta_a(
    sqrt_price_0: u128,
    sqrt_price_1: u128,
    liquidity: u128,
    round_up: bool
): u64 {
    let sqrt_price_diff = if (sqrt_price_0 > sqrt_price_1) {
        sqrt_price_0 - sqrt_price_1
    } else {
        sqrt_price_1 - sqrt_price_0
    };
    if (sqrt_price_diff == 0 || liquidity == 0) {
        return 0
    };
    ......
}
```

5.6 `partner` and `fee_tier` modules don't have any functions to remove partner and fee

Severity: Minor
Status: Fixed

Descriptions: As time goes on, the `partner` and `fee_tier` may have a large number of partners and fee_tiers. For administration, may need a way to remove the unused `partners` and `fee_tiers`.

Commit: e56d47667850dbc5a9553eddb0f67572e7c3c3b8

Code Location: sources/partner.move, sources/fee_tier.move

Suggestion: Adding a `remove` function for `partner` and `fee_tier` module.

5.7 `router::` `create_pool` can create a pool with the same type

Severity: Medium
Status: Fixed

Descriptions: `router::create_pool<CoinA, CoinA>(...)` can succeed. The swap between `CoinA` and `CoinA` is nonsense.

Commit: e56d47667850dbc5a9553eddb0f67572e7c3c3b8

Code Location: sources/router.move, line 161

Suggestion: When calling `create_pool` with the same coin pair, abort.

5.8 Some assertions can be optimized

Severity: Medium
Status: Fixed

Descriptions: Many assertions for the input argument checks are not placed at the beginning of functions. It's suggested that we should put them at the beginning of functions, so they can fail fast, and more gas-saving. For example, `create_partner` & `update_fee_rate` in `partner.move`, and `add_fee_tier` & `update_fee_tier` in `fee_tier.move`.

Commit: e56d47667850dbc5a9553eddb0f67572e7c3c3b8

Suggestion: Put argument check assertions at the beginning of functions.

5.9 Wrong event type emitted in `factory::create_pool`

Severity: Medium
Status: Fixed

Descriptions: In `factory::create_pool`, it emits `CreatePoolEvent.coin_type_b` with `CoinTypeA` type. It's not correct, and it should be `CoinTypeB` type.

Commit: e56d47667850dbc5a9553eddb0f67572e7c3c3b8

Code Location: sources/factory.move, line 123

Suggestion: Change `CreatePoolEvent.coin_type_b` with `CoinTypeB` type.

```
public fun create_pool<CoinTypeA, CoinTypeB>(
    ......
    event::emit_event(&mut pools.create_pool_events, CreatePoolEvent {
        coin_type_a: type_of<CoinTypeA>(),
        coin_type_b: type_of<CoinTypeB>(),
        ......

}
```

5.10 Everyone can reset the initial price of a pool

Severity: Major
Status: Fixed

Descriptions: Everyone can reset the initial price of a pool by calling the public function `pool::reset_init_price`. `pool::reset_init_price` only checks whether `position_index` is equal to 1, that's not safe. Suppose in such a case, someone calls `factory::create_pool` to create a pool, and `factory::create_pool` will create a default position with index 0 for him, and he can add liquidity to that position and produce some liquidity. At this time, someone else can reset the pool's initial price to another price successfully, and even trade the creator's assets at a lower price. In that case, it causes the creator's loss of assets.

Commit: c867755da203332468a37535c45ed7a7a4bbc65a

Code Location: sources/pool.move, line 436

Suggestion: Don't let everyone call this function, just leave it to the admin of the pool.

5.11 The comments on functions are out of date

Severity: Minor
Status: Fixed

Descriptions: Many function comments are out of date. For example, there is no argument named `name` in `router::add_liquidity` , `router::add_liquidity_fix_token` , `router::remove_liquidity` , and `router::collect_rewarder` .

```
#[cmd]
/// Add liquidity into a pool. The position is identified by the name.
/// The position token is identified by (creator, collection, name), the creator is pool address.
/// Params
///     Type:
///         - CoinTypeA
///         - CoinTypeB
///     - pool
///     - delta_liquidity
///     - max_amount_a: the max number of coin_a can be consumed by the pool.
///     - max_amount_b: the max number of coin_b can be consumed by the pool.
///     - tick_lower
///     - tick_upper
///     - is_open: control whether or not to create a new position or add liquidity on existed position.
///     - name: position name. if `is_open` is true, name is no use.
/// Returns
public entry fun add_liquidity<CoinTypeA, CoinTypeB>(
    account: &signer,
    pool_address: address,
    delta_liquidity: u128,
    max_amount_a: u64,
    max_amount_b: u64,
    tick_lower: u64,
    tick_upper: u64,
    is_open: bool,
    index: u64,
) {
```

Commit: 25d115473799a9db777837553bd5e78bf88ca03a

Code Location: sources/router.move

Suggestion: Update the comments.

5.12 `pool::` `add_liquidity_fix_coin` & `pool::add_liquidity` have many duplicated codes

Severity: Minor
Status: Fixed

Descriptions: These two functions are very important to add liquidity, but they have 80% duplicated codes, which can be wrapped into a common function, and improve the code maintainability.

Commit: c867755da203332468a37535c45ed7a7a4bbc65a

Code Location: sources/pool.move

Suggestion: Refactoring these two functions, and wrapping the common codes into a new function.

5.13 `pool::remove_liquidity` does not call `pool::` `update_rewarder`

Severity: Critical
Status: Fixed

Descriptions: `pool::` `update_rewarder` is used to update the `growth_global` upon swap, add liquidity, remove liquidity, collect rewarder and update emission. But `pool::remove_liquidity` does not call this function, it would cause the reward cumulative error.

Commit: c867755da203332468a37535c45ed7a7a4bbc65a

Code Location: sources/pool.move, 747

Suggestion: Update the codes, and call `pool::update_rewarder` .

5.14 Gas cost is higher than other DEX

Severity: Minor
Status: Pending

Descriptions: We tested `create_pool` , `add_liquidity` and `swap` in module `clmm_router` , and we found the average gas consumption for these operations is `0.0n` level. This is somehow higher than other AMM DEX. As a CLMM DEX, Cetus definitely will have higher gas, and we already found some gas-optimization issues which Cetus has already taken, but Cetus still should improve to reduce the gas.

Suggestion: Keep reducing the gas for users.

5.15 `utils::str` optimization

Severity: Medium

Status: Fixed

Descriptions: The current implementation of `utils::str` is not optimized. It uses a pre-defined map to convert a `u8` to a `char` and inserts the `char` into the index 0 of the string. This is very inefficient.

Commit: e56d47667850dbc5a9553eddb0f67572e7c3c3b8

Code Location: `sources/utils.move, line 7`

```
public fun str(num: u64): String {
let ns = simple_map::create<u64, String>();
simple_map::add(&mut ns, 1, string::utf8(b”1”));
simple_map::add(&mut ns, 2, string::utf8(b”2”));
simple_map::add(&mut ns, 3, string::utf8(b”3”));
simple_map::add(&mut ns, 4, string::utf8(b”4”));
simple_map::add(&mut ns, 5, string::utf8(b”5”));
simple_map::add(&mut ns, 6, string::utf8(b”6”));
simple_map::add(&mut ns, 7, string::utf8(b”7”));
simple_map::add(&mut ns, 8, string::utf8(b”8”));
simple_map::add(&mut ns, 9, string::utf8(b”9”));
simple_map::add(&mut ns, 0, string::utf8(b”0”));
if (num == 0) {
    return string::utf8(b”0”)
};
let res = string::utf8(b””);
let remainder ;
while (num > 0) {
    remainder = num % 10;
    num = num / 10;
    string::insert(&mut res, 0, *simple_map::borrow<u64, String>(&ns, &remainder));
};
res
}
```

Suggestion: Refer the implementation below.

```
public fun str2(num: u64): String {
    if (num == 0) {
        return string::utf8(b”0”)
    };
let remainder: u8;
let digits = vector::empty<u8>();
while (num > 0) {
    remainder = (num % 10 as u8);
    num = num / 10;
    vector::push_back(&mut digits, remainder + 48);
};
vector::reverse(&mut digits);
string::utf8(digits)
}
```

Gas cost comparison between `str` and `str2` :

num	str gas cost	str2 gas cost
0	835	164
18446744073709551615	6937	339

5.16 Deploy smart contract without multi-sig

Severity: Medium
Status: Pending
Descriptions: The smart contract is not deployed under a multi-sig account. Operations performed with multiple signatures will provide greater security. Even if the loss of a single private key will not allow an attacker to gain access to the contract. Multiple trusted parties must approve the update at the same time, otherwise, it will not work.

Suggestion: Use a multi-sig account for the smart contract when deploying.

5.17 `TODO` labels still remain in the code

Severity: Minor

Status: Fixed

Descriptions: There are some `TODO` labels in `clmm_math.move` , all the left `TODO` labels are about tests. `TODO` often means work is not finished or possibility of defects. If we're not sure about the codes, we should write more tests to ensure the codes work correctly.

Commit: 25d115473799a9db777837553bd5e78bf88ca03a

Code Location: `source/math/clmm_math.move`

```
#[test]
fun test_get_next_price_a_up() {
    // TODO: Add more test for get_next_sqrt_price_a_up
    ......
}

#[test]
fun test_get_next_price_b_down() {
    // TODO: Add more test for test_get_next_price_a_down
    ......
}

#[test]
fun test_compute_swap_step() {
    // TODO: Add more test for test_compute_swap_step
    ......
}
```

Suggestion: Add more test codes to ensure the correctness of codes.

5.18 Position recalculation optimization

Severity: Medium

Status: Fixed

Descriptions: In `collect_fee` and `collect_rewarder` functions in `pool` module, there are duplicated codes to get the pool and position. The reason is `get_position_tick_range` can not borrow the `Pool` resource after the `pool` variable keeps a mutable reference to the `Pool` resource. This is a limitation of Move language to ensure security. We can solve this by introducing a helper function which uses a `&Pool` parameter to get the position tick range.

Commit: c867755da203332468a37535c45ed7a7a4bbc65a

Code Location: `sources/pool.move, line 947`

```
public fun collect_rewarder<CoinTypeA, CoinTypeB, CoinTypeC>(
    account: &signer,
    pool_address: address,
    position_index: u64,
    rewarder_index: u8,
    recalculate: bool,
): Coin<CoinTypeC> acquires Pool {
    check_position_authority<CoinTypeA, CoinTypeB>(account, pool_address, position_index);

let (pool, position) = if (recalculate) {
    let (tick_lower, tick_upper) = get_position_tick_range<CoinTypeA, CoinTypeB>(pool_address, position_index);
    let pool = borrow_global_mut<Pool<CoinTypeA, CoinTypeB>>(pool_address);
    assert_status(pool);
    update_rewarder(pool);
    let rewards_growth_inside = get_reward_in_tick_range(pool, tick_lower, tick_upper);
    let position = table::borrow_mut(&mut pool.positions, position_index);
    update_position_rewarder(position, rewards_growth_inside);
    (pool, position)
} else {
    let pool = borrow_global_mut<Pool<CoinTypeA, CoinTypeB>>(pool_address);
    assert_status(pool);
    update_rewarder(pool);
    let position = table::borrow_mut(&mut pool.positions, position_index);
    (pool, position)
};
......
}
```

Suggestion: Add a new function `get_position_tick_range_by_pool` to use a `&Pool` parameter.
Then we can rewrite the `collect_rewarder` and `collect_fee` functions to remove the duplicated code.

```
// add this new function
public fun get_position_tick_range_by_pool<CoinTypeA, CoinTypeB>(
    pool_info: &Pool<CoinTypeA, CoinTypeB>,
    position_index: u64
): (I64, I64) {
    if (!table::contains(&pool_info.positions, position_index)) {
        abort EPOSITION_NOT_EXIST
    };
    let position = table::borrow(&pool_info.positions, position_index);
    (position.tick_lower_index, position.tick_upper_index)
}

    public fun get_position_tick_range<CoinTypeA, CoinTypeB>(
        pool_address: address,
```

```
        position_index: u64
    ): (I64, I64) acquires Pool {
        let pool_info = borrow_global<Pool<CoinTypeA, CoinTypeB>>(pool_address);
        get_position_tick_range_by_pool(pool_info, position_index)
    }

    // rewrite the collect_rewarder function
    public fun collect_rewarder<CoinTypeA, CoinTypeB, CoinTypeC>(
        account: &signer,
        pool_address: address,
        position_index: u64,
        rewarder_index: u8,
        recalculate: bool,
    ): Coin<CoinTypeC> acquires Pool {
        check_position_authority<CoinTypeA, CoinTypeB>(account, pool_address, position_index);

let pool = borrow_global_mut<Pool<CoinTypeA, CoinTypeB>>(pool_address);
assert_status(pool);
update_rewarder(pool);
let position = if (recalculate) {
    let (tick_lower, tick_upper) = get_position_tick_range_by_pool<CoinTypeA, CoinTypeB>(pool, position_index);
    let rewards_growth_inside = get_reward_in_tick_range(pool, tick_lower, tick_upper);
    let position = table::borrow_mut(&mut pool.positions, position_index);
    update_position_rewarder(position, rewards_growth_inside);
    position
} else {
    table::borrow_mut(&mut pool.positions, position_index)
};
......
}
```

5.19 Dependency git rev should be a commit hash or a tag instead of a branch for reproducibility

Severity: Medium

Status: Fixed

Descriptions: The dependency git rev should be a commit hash or a tag instead of a branch for reproducibility. The branch may be updated in the future, which may cause the build to fail. An example is the frozen version(git commit hash 411cc86b1b8bd1f1ea7a8b9befd97cc3bf104efa) of cetus-clmm can not be compiled with the latest main branch of `aptos-core` (git commit hash b362344e4b74dc20caad254d356067fcf713353a). While after changing the rev to a previous version(git commit hash e5a0c085143c50dcac711c534e6b4b93d7647c29), it can be compiled successfully.

Code Location: `Move.toml`

```
[dependencies.AptosFramework]
git = 'https://github.com/aptos-labs/aptos-core.git'
rev = 'main'
subdir = 'aptos-move/framework/aptos-framework'
```

Suggestion: Use a commit hash or a tag instead of a branch for the dependency git rev.

```
[dependencies.AptosToken]
git = "https://github.com/aptos-labs/aptos-core.git"
subdir = "aptos-move/framework/aptos-token"
rev = "e5a0c085143c50dcac711c534e6b4b93d7647c29"
```

5.20 The `pool` Coin Order Handle

Severity: Medium

Status: Pending

Descriptions: In `create_pool<CoinTypeA, CoinTypeB>` , a `SimpleMap<PoolId, address>` will be kept in the `Pools` . The pool id is a struct of `{ CoinTypeA, CoinTypeB, tick_spacing }` . We can not create a new pool with the same coins and tick_spacing because the seed to generate the pool signer is derived from `hash(sorted(CoinTypeA, CoinTypeB), tick_spacing)` .
There may be `PoolId { CoinA, CoinB, TickSpacing0 }` and `PoolId { CoinB, CoinA, TickSpacing1 }` in the `Pools` at the same time. It might be confusing for the users and inconvenient for the front-end developers in the future. The `assert!( !simple_map::contains_key<PoolId, address>(&pools.data, &pool_id), EPOOL_ALREADY_INITIALIZED )` in this function will never be triggered. If `PoolId { CoinA, CoinB, TickSpacing }` is already in the `Pools` , then `PoolId { CoinA, CoinB, TickSpacing }` and `PoolId { CoinB, CoinA, TickSpacing }` will both be aborted in `account::create_resource_account(&pool_owner_signer, pool_seed);` with `ERESOURCE_ACCCOUNT_EXISTS` .

Commit: 25d115473799a9db777837553bd5e78bf88ca03a

Code Location: `sources/factory.move, line 73`

```
public fun create_pool<CoinTypeA, CoinTypeB>(
    account: &signer,
    tick_spacing: u64,
    initialize_price: u128,
    uri: String
): address acquires PoolOwner, Pools {
    ......
    // Create pool account
    let pool_id = new_pool_id<CoinTypeA, CoinTypeB>(tick_spacing);
let pool_owner = borrow_global<PoolOwner>(@cetus_clmm);
let pool_owner_signer = account::create_signer_with_capability(&pool_owner.signer_capability);

let pool_seed = new_pool_seed<CoinTypeA, CoinTypeB>(tick_spacing);
let pool_seed = bcs::to_bytes<PoolId>(&pool_seed);
let (pool_signer, signer_cap) = account::create_resource_account(&pool_owner_signer, pool_seed);
let pool_address = signer::address_of(&pool_signer);

let pools = borrow_global_mut<Pools>(@cetus_clmm);
pools.index = pools.index + 1;
assert!(
    !simple_map::contains_key<PoolId, address>(&pools.data, &pool_id),
    EPOOL_ALREADY_INITIALIZED
);
simple_map::add<PoolId, address>(&mut pools.data, pool_id, pool_address);
......
}
```

Suggestion: Force the user to create a pool with coins in order. For example, `create_pool<CoinA, CoinB>` will succeed while `create_pool<CoinB, CoinA>` will fail. Adding a coin order assert in `create_pool` will solve this. And use `pool_seed` as the key of `pools`.

Appendix 1 – Files in Scope

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

Files	SHA-1 Hash
sources/pool.move	0a6d3bae00c68b17fc3048a936c7f45fb2188f27
sources/position_nft.move	8e97c9c3926e54431793198947b7d66354e5f89d
sources/math/clmm_math.move	5cf128d78e2ff27643312a7140be481856618a08
sources/math/tick_math.move	c8f20d821db79d6a451906df23d5523ce0474790
sources/utils.move	c256376b3acf38d1199a62f474cbe2ee3473d9ce
sources/fee_tier.move	006985e4f48917f34fdb9262df6fcfb2c7328560
sources/config.move	238e01e338718e9e00725201771baa3346fa4a66
sources/router.move	09dabe2a0db5e9f6e964e0ca40d58e9eaf9ef4b6
sources/acl.move	63166b798079096eec465d06ee2f99bbe087da08
sources/partner.move	801c382bc6d12ff1aa2d5bb405d808a36ac5fa61
sources/factory.move	42a811de6a0a9585f7e64d83f9d9c3a2a1cc9131
Move.toml	d3fc1f3ca95c9a9ad24bbfb5536f8e6449b7471d

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.

MOVEBIT

Securing the Move Ecosystem

https://twitter.com/movebit_

contact@movebit.xyz