Nemo Protocol

Audit Report

MOVEBIT

contact@bitslab.xyz

https://twitter.com/movebit_

Wed Jul 30 2025

Nemo Protocol Audit Report

1 Executive Summary

1.1 Project Information

Description	Nemo Protocol is a yield trading protocol on Sui
Type	DeFi
Auditors	MoveBit
Timeline	Thu Jul 10 2025 - Wed Jul 30 2025
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/nemo-protocol/mmt-lp-rebalancing/ https://github.com/nemo-protocol/fee-distributor/ https://github.com/nemo-protocol/mmt-router/ https://github.com/nemo-protocol/price-adapter/
Commits	https://github.com/nemo-protocol/mmt-lp-rebalancing/: 3a39167bc68e0652f1aeb64a6cb85b36cbbafe87 https://github.com/nemo-protocol/fee-distributor/: 9758aefd33cd4f24bc300de5c9b56f0fbc0ca258 2878a1036afda8cd126da410e07ff45571626e87 https://github.com/nemo-protocol/mmt-router/: 079aef375d6b384a1da0dd833790d315df9848dd

	https://github.com/nemo-protocol/price-adapter/: 791eb351e7960d92c046804c635dd2f16437b73b 47b506b48e9bebf2219a7c6890cce52f2c9ed8d1

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MOV	Move.toml	25cfe8bf5e9162d91db73004c6901fcc91cf0b88
ACL	sources/acl.move	25503f67fd460127c2f2fc36d4db294ebfd6d535
MV3A	sources/adapters/mmt_v3_adapter.move	6237de41ef1d1abeea62154f554d2f4a3dfb8f54
CAD	sources/adapters/cetus_adapter.move	bb24cd2e04ff695876019dd8616e4af6af4d7b4e
BVA	sources/bag_value.move	761151ae1a09ed0567a77ea5bbed1ab06e0728db
ACL	sources/acl.move	93a7e0d5a81763d329025b5380b22bb6f3b95391
CVE	sources/current_version.move	3acf3f998034d3ebb7117c2d6028f940e3582beb
VER	sources/version.move	ea16ca30d0a541ba49c44fd5f88b6476298e14e3
MOV1	clmm-vault/Move.toml	fce2cca2c888d4b18fc457f4da6963c68166452e
ADM	clmm-vault/sources/actions/admin.move	e1e8ee6996b4c71da5ed9b94ab15125cf9bade8b

ROU	clmm-vault/sources/actions/route.move	bcb09425ecb79fe8776c3c1d4d56bc424e3fc71b
CON2	clmm-vault/sources/storage/control.move	3510a8e5c659477e12a890223529f12a20aceaf2
MOV1	Move.toml	788c0bc019fbd096c991dc9ee99c6bf8acf40532
PAD	sources/pyth_adapter.move	648f9b2f60a49d9917b33070d3221084a3bb78b0
PSO	sources/price_source.move	6a1fca50678dbfaba1a58a363831f1702057172f
REG	sources/registry.move	aa74ffcf3a72de5c946f3b6c1042f1bb7b5009f1
MAD	sources/mmt_adapter.move	a4636cda57d25662f95d96d3e17afbacd01b43e1
MOV	Move.toml	43399dd1d715a1a557a540643cddf76a5ab30bd0
FRE	sources/fee_receipt.move	548036c9da3108d115025d8dabaf50e88c23827d
FDI	sources/fee_distributor.move	6d22753df41dddabc3c724d6ea8f6580bc6c9b4a
REG	sources/registry.move	cadf9ea086010123bd0dc5e4ad96e1268d0b96ca

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	10	10	0
Informational	5	5	0
Minor	0	0	0
Medium	4	4	0
Major	1	1	0
Critical	0	0	0

1.4 MoveBit Audit Breakdown

MoveBit 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
- The flow of capability
- Witness Type

1.5 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 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) Formal Verification(Optional)

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. 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 Nemo Protocol to identify any potential issues and vulnerabilities in the source code of the Nemo Protocol 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 10 issues of varying severity, listed below.

ID	Title	Severity	Status
ACL-1	Some functions use the deprecated `public(friend)`	Informational	Fixed
ADM-1	`set_upgeer_trigger_price_factor_drift` Incorrect Spelling Of Parameters	Informational	Fixed
ADM-2	`set_vault_parameters` slippage_up and slippage_dow Omitted When Setting Parameters	Informational	Fixed
FDI-1	implement a remove_fee_receiver() function	Medium	Fixed
FRE-1	Error Code Is Not Used	Informational	Fixed
PAD-1	Use Custom Expiry with `get_price_no_older_than()` to Avoid Stale Pyth Prices	Medium	Fixed
PSO-1	Inconsistent `dummy_field` Assignment in	Medium	Fixed

	`MmtOraclePriceSource` Regardless of `is_primary` Flag		
REG-1	Admin Cannot Revoke `FeeCollectorCap`	Medium	Fixed
VAD-1	`RATIO_SCALLING_FACTO` Misspelled Words	Informational	Fixed
VAU-1	`set_active_vault_cap` Lacks Access Control	Major	Fixed

3 Participant Process

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

Admin

- `set_k_oracle_price` : Set the oracle price data
- `upgrade_major` : Perform the major version upgrade
- `upgrade_minor` : Perform the minor version upgrade
- `set_price_pair_id` Set the price for the ID
- `add_fee_receiver` : Add a new fee receiver
- `update_receiver_percentage` : Update fee receiver percentage
- `claim_admin_fees` : Claim the accumulated fees of the administrator
- `claim_fees` : Recipient of the fee claims the accumulated fee
- `create_fee_collector_cap` : ceate a fee collector capability
- `revoke_fee_collector_cap` : Revoke a fee collector capability
- `set_position_price_scaling_for_vault` : Set the vault position price scaling factor
- `set_trigger_scalling` : set trigger scaling for uncorrelated vault
- `set_upgeer_trigger_price_factor_drift` : Set upper trigger price factor for drift vault
- `set_slippage` : Set slippage parameters
- `set_free_threshold_a` : Set free threshold for asset A
- `set_free_threshold_b` : Set free threshold for asset B
- `set_lock_threshold_a` : Set lock threshold for asset A
- `set_lock_threshold_b` : Set lock threshold for asset B

- `set_deposit_limit` : Set a deposit limit
- `set_fee` : Set vault fee
- `set_vault_parameters` : Set all vault parameters in one transaction
- `revoke_vault_config_cap` : Revoke a vault config admin capability
- `revoke_risk_admin_cap` : Revoke a risk admin capability
- `revoke_fee_admin_cap` : Revoke a fee admin capability
- `revoke_rebalance_admin_cap` : Revoke a rebalance admin capability
- `revoke_treasury_admin_cap` : Revoke a treasury admin capability
- `revoke_pause_admin_cap` : Revoke a pause admin capability
- `revoke_unpause_admin_cap` : Revoke an unpause admin capability
- `issue_vault_config_cap` : Issue a vault config admin capability
- `issue_risk_admin_cap` : Issue a risk admin capability
- `issue_fee_admin_cap` : Issue a fee admin capability
- `issue_rebalance_admin_cap` : Issue a rebalance admin capability
- `issue_treasury_admin_cap` : Issue a treasury admin capability
- `issue_pause_admin_cap` : Issue a pause admin capability
- `issue_unpause_admin_cap` : Issue an unpause admin capability
- `issue_all_capabilities` : Issue all capabilities to a user
- `issue_specific_capabilities` : Issue specific capabilities to a user

PauseAdmin

- `pause_vault` : Suspend all operations in the vault
- `add_force_rebalance_df` : Add a forced rebalancing sign

- `set_deposit_enable` : Set deposit enable/disable

UnpausePauseAdmin

- `unpause_vault` : Unpause vault

FeeAdmin

- `set_fee` : Set vault fee
- `set_withdraw_fee` : Set withdraw fee
- `withdraw_fee_a` : Withdraw fee balance for asset A
- `withdraw_fee_b` : Withdraw fee balance for asset B

RebalanceAdmin

- `set_rebalance_price_source` : Set rebalance price source
- `set_uc_target_adapter` : Set target adapter for uncorrelated vault
- `set_target_adapter` : Set target adapter for drift vault
- `set_uc_is_target_reverse` : Set target reverse flag for uncorrelated vault

TreasuryAdmin

- `issue_vault_cap` : Issue vault cap
- `set_active_vault_cap` : Set active vault cap
- `new_uncorrelated_vault` : Create a new Non-correlation strategy vault
- `new_stable_vault` : Create a new stability strategy vault
- `new_drift_vault` : Create a new trend-driven strategy vault

ValueCap

- `set_free_threshold_a` : Set the idle balance threshold of asset A
- `set_free_threshold_b` : Set the idle balance threshold of asset B

- `set_lock_threshold_a` : Set the lock-up threshold for asset A
- `set_free_threshold_b` : Set the lock-up threshold for asset B
- `set_slippage_up` : Set the upward slippage tolerance
- `set_slippage_down` : Set the tolerance for downward slippage
- `set_deposit_limit` : Set an upper limit on the deposit in the vault
- `set_fee` : Set the exchange and reward rates for the vault
- `set_withdraw_fee` : Set the withdrawal rate for the vault
- `set_target_adapter` : Set the target adapter address for the trend vault
- `set_uc_target_adapter` : Set the target adapter address for the non-correlated vault
- `set_uc_is_target_reverse` : Whether to reverse the target set for non-correlated vaults
- `issue_rebalance_cap` : Release a new RebalanceCap
- `update_rebalance_cap_ownership` : Update the whitelist address of an existing RebalanceCap

User

- `add_reward_x` : add rewards to the vault.
- `add_reward_y` : add rewards to the vault
- `swap_a2b` : Execute the exchange from CoinTypeA to CoinTypeB
- `swap_b2a` : Execute the exchange from CoinTypeB to CoinTypeA

FeeDistributor

- `consume_fee_receipt` : Use the receipt and transfer the fee to the distributor
- `issue_fee_receipt_with_coins` : Create a PermissionedReceipt with a token balance

RouterAcl

- `swap_router` : Execute cross-pool routing and switching logic

4 Findings

ACL-1 Some functions use the deprecated `public(friend)`

Severity: Informational

Status: Fixed

Code Location:

sources/acl.move#27

Descriptions:

The public(friend) modifier has been deprecated by Sui Move and should be replaced with public(package). The current grammar may cause compatibility issues in future versions

Suggestion:

Change to public(package)

Resolution:

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

ADM-1 `set_upgeer_trigger_price_factor_drift` Incorrect Spelling Of Parameters

Severity: Informational

Status: Fixed

Code Location:

clmm-vault/sources/actions/admin.move#67

Descriptions:

There is a spelling mistake in the 'set_upgeer_trigger_price_factor_drift' function. 'upgeer' should be changed to 'upper'

Suggestion:

Correct the misspelled words

Resolution:

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

ADM-2 `set_vault_parameters` slippage_up and slippage_dow Omitted When Setting Parameters

Severity: Informational

Status: Fixed

Code Location:

clmm-vault/sources/actions/admin.move#292

Descriptions:

When setting parameters in batches for a function, the initialization of key parameters was omitted, and the slippage_parameters (slippage_up/slippage_down) were not set.

```
public fun set_vault_parameters<A, B, V, T: store + copy + drop>(
        vault_config_cap: &VaultConfigAdminCap,
        risk_cap: &RiskAdminCap,
        vault: &mut Vault<A, B, V, T>,
        vault_cap: &VaultCap,
        lower_price_factor: u128,
        upper_price_factor: u128,
        slippage_up: u128,
        slippage_down: u128,
        free_threshold_a: u64,
        free_threshold_b: u64,
        lock_threshold_a: u64,
        lock_threshold_b: u64,
        deposit_limit: u64
    ) {
```

Suggestion:

Configure parameters or delete parameters that are no longer needed

Resolution:

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

FDI-1 implement a remove_fee_receiver() function

Severity: Medium

Status: Fixed

Code Location:

sources/fee_distributor.move#92-150

Descriptions:

The function `add_fee_receiver()` allows the admin to add a new fee receiver.

```
// Add a new fee receiver (requires AdminCap)
  public entry fun add_fee_receiver<V: drop>(
     _admin_cap: &registry::AdminCap,
     fee_distributor: &mut FeeDistributor,
     name: vector<u8>,
     percentage: u64,
     ctx: &mut TxContext
  ) {
     let name_string = ascii::string(name);

     // Validate percentage (0-10000 basis points, i.e., 0-100%)
     assert!(
        percentage <= FEE_PERCENTAGE_MAX,
        EInvalidFeePercentage
     );
```

However, the protocol does not provide a `remove_fee_receiver()` function. As a result, if a receiver was included in the first fee distribution but is not intended to receive fees in the second distribution, they will still receive fees. This is because the outdated or no longer eligible receiver remains in the list, leading to incorrect fee allocation.

Suggestion:

It is recommended to implement a `remove_fee_receiver()` function.

Resolution:

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

FRE-1 Error Code Is Not Used

Severity: Informational

Status: Fixed

Code Location:

sources/fee_receipt.move#20

Descriptions:

There are multiple unused error code constants in the code

```
const EInvalidReceiverName: u64 = 1;
const EReceiptAlreadyConsumed: u64 = 2;
const EInvalidFeeDistributor: u64 = 3;
```

Suggestion:

Delete the redundant code or use error codes

Resolution:

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

PAD-1 Use Custom Expiry with `get_price_no_older_than()` to Avoid Stale Pyth Prices

Severity: Medium

Status: Fixed

Code Location:

sources/pyth_adapter.move#15-24

Descriptions:

In the `get_pyth_price()` function, the protocol calls `pyth::get_price()` to fetch the price from the Pyth network.

```
    public fun get_pyth_price(s: &state::State, p: &PriceInfoObject, c: &clock::Clock) : (u64,
u8, u64) {
        let v0 = pyth::get_price(s, p, c);
        let v1 = price::get_price(&v0);
        let v2 = i64::get_magnitude_if_positive(&v1);
        let v3 = price::get_expo(&v0);
        let v4 = i64::get_magnitude_if_negative(&v3);
        assert!(v4 <= 255, 0);
        assert_price_conf_within_range(v2, price::get_conf(&v0));
        (v2, v4 as u8, price::get_timestamp(&v0))
    }
```

However, `get_price_no_older_than()` uses Pyth's default maximum age for price validity, which may not be suitable for all tokens.

```
    public fun get_price(price_identifier: PriceIdentifier): Price {
        get_price_no_older_than(price_identifier, state::get_stale_price_threshold_secs())
    }
```

Some tokens may have frequent price updates, while others update less often. As a result, relying on the default expiration threshold could lead the protocol to use stale or outdated prices, especially for tokens with slower update intervals.

Suggestion:

It is advisable to use `get_price_no_older_than()` with a custom, token-specific maximum age to ensure that the fetched price is fresh and aligned with the expected update frequency of each token.

Resolution:

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

PSO-1 Inconsistent `dummy_field` Assignment in `MmtOraclePriceSource` Regardless of `is_primary` Flag

Severity: Medium

Status: Fixed

Code Location:

sources/price_source.move#42-60

Descriptions:

In the `set_k_oracle_price()` function, the protocol decides which oracle update routine to call based on the `is_primary` flag:

```
if is_primary {
    oracle::set_primary_price(...);
} else {
    oracle::set_secondary_price(...);
}
```

However, in **both** branches the `MmtOraclePriceSource` struct is instantiated (or updated) with `dummy_field = false`. Because `dummy_field` is currently hard coded, its value never reflects the actual `is_primary` state. Please verify whether `dummy_field` is meant to distinguish primary from secondary updates; if so, it should be set to `true` when `is_primary == true` (and `false` otherwise) to keep the struct's state consistent with the branch that was executed.

Suggestion:

It is recommended to set `dummy_field` to true when `is_primary` is true, and to false when `is_primary` is false.

Resolution:

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

REG-1 Admin Cannot Revoke `FeeCollectorCap`

Severity: Medium

Status: Fixed

Code Location:

sources/registry.move#64-75

Descriptions:

The `create_fee_collector_cap` function uses `transfer::transfer` to send the newly created `FeeCollectorCap` object directly to the `collector_address`. This action transfers full ownership of the capability object away from the administrator.

The `revoke_fee_collector_cap` function requires the caller to own the `FeeCollectorCap` object, as it is passed by value. Because the administrator has already relinquished ownership during creation, they are unable to call this function to revoke the capability.

```
// Create a fee collector capability (only admin can create)
public entry fun create_fee_collector_cap(
    _admin_cap: &AdminCap,
    name: vector<u8>,
    collector_address: address,
    ctx: &mut TxContext
) {
    let collector_cap = FeeCollectorCap {
        id: object::new(ctx),
        name: ascii::string(name),
    };

    event::emit(
        FeeCollectorCapCreated {name: collector_cap.name}
    );

    transfer::transfer(collector_cap, collector_address);
}
```

```
// Revoke a fee collector capability (only admin can revoke)
public entry fun revoke_fee_collector_cap(
    _admin_cap: &AdminCap,
    collector_cap: FeeCollectorCap,
    ctx: &mut TxContext
) {
    let FeeCollectorCap {id, name,} = collector_cap;

    event::emit(FeeCollectorCapRevoked { name });

    object::delete(id);
}
```

Suggestion:

It is recommended to adjust the logic of the revoke.

Resolution:

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

VAD-1 `RATIO_SCALLING_FACTO` Misspelled Words

Severity: Informational

Status: Fixed

Code Location:

clmm-vault/sources/actions/rebalance/adapters/vsui_adapter.move#17

Descriptions:

The word "RATIO_SCALLING_FACTOR" is misspelled. "SCALLING" should be "SCALING"

```
const RATIO_SCALLING_FACTOR: u128 = 1_000_000;
const PRICE_SCALING_FACTOR: u128 = 1_000_000_000;
```

Suggestion:

Correct the misspelled words

Resolution:

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

VAU-1 `set_active_vault_cap` Lacks Access Control

Severity: Major

Status: Fixed

Code Location:

clmm-vault/sources/storage/vault.move#820-833

Descriptions:

The `set_active_vault_cap` function is declared as `public` but lacks any authorization checks. Its purpose is to set or replace the `vault_cap` dynamic field associated with a `Vault`, which is a highly sensitive operation that determines which capability object can manage the vault.

```
public fun set_active_vault_cap<A, B, V, T: store + copy + drop>(
    vault: &mut Vault<A, B, V, T>,
    vault_cap_id: ID,
    _ctx: &mut TxContext
) {
    if (df::exists_(&vault.id, b"vault_cap")) {
        df::remove<_, ID>(&mut vault.id, b"vault_cap");
    };
    df::add(
        &mut vault.id,
        b"vault_cap",
         vault_cap_id
    );
}
```

```
public fun check_vault_cap_compatibility<A, B, V, T: store + copy + drop>(
    self: &Vault<A, B, V, T>,
    vault_cap: &VaultCap
) {
    assert!(
        vault_cap.vault_id == object::id(self),
        error::invalid_vault_cap()
    );
    if (df::exists_(&self.id, b"vault_cap")) {
        assert!(
            df::borrow(&self.id, b"vault_cap") == object::id(vault_cap),
            error::invalid_vault_cap()
        );
    }
}
```

Suggestion:

It is recommended to add permission control.

Resolution:

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

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