Haedal haeVault v2

Audit Report

MOVEBIT

contact@bitslab.xyz

https://twitter.com/movebit_

Wed Nov 12 2025

Haedal haeVault v2 Audit Report

1 Executive Summary

1.1 Project Information

Description	A decentralized structured product vault that executes complex liquidity management and market-making strategies in an external DLMM (Deep Liquidity Market Maker) protocol through an automated Keeper system, thereby generating returns for users' deposit assets
Type	DeFi
Auditors	Alex,PeiQi0
Timeline	Sun Oct 26 2025 - Wed Nov 12 2025
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/haedallsd/haevault
Commits	6d4bf5697e4b2c9e35b397710ffe6ebcc33271f8 b3f8356421ee291679e4796d96a37fb951655583e 6038f593ec934b66f96b383b02da2fdc30bd328a 50a5f9392db3767af91fc6653b681b722d409edd

1.2 Files in Scope

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

ID	File	SHA-1 Hash
ACL	packages/volatile_vault_v2/sources/acl.move	885c83a559244ec1a63a46efe939cee1460b15b1
ACA	packages/volatile_vault_v2/sources/admin_cap.move	96e84c6545f9340a29a8c4fe1265a4ad7c92a17c
BBA	packages/volatile_vault_v2/sources/balance_bag.move	b66383e87671e80b4c90563f971e00f14b50e100
VER	packages/volatile_vault_v2/sources/versioned.move	c2b8b807a1cf6e17a583db6ce380860e49089f7c
UTI	packages/volatile_vault_v2/sources/utils.move	177f2d613652e6e00b6aa40639feb4a8d9de853c
POR	packages/volatile_vault_v2/sources/pyth_oracle.move	0131ef0c8e97a326ac87e32f5604b09699952871
CON	packages/volatile_vault_v2/sources/config.move	39128891b3fb2c324eb834b9831b9b3359da6065
MOV	packages/volatile_vault_v2/Move.toml	b200e8f78c12812cdf8231b0fc2cf96fb8c07e2b
VAU	packages/volatile_vault_v2/sources/vault.move	5dddf29049df1f852419e8c28ade791edf47a0f2
MDL	packages/volatile_vault_v2/sources/market_dlmm.move	67d781069712f1883c1c91bd34bc87f1284044f6

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	4	4	0
Critical	0	0	0
Major	0	0	0
Medium	2	2	0
Minor	2	2	0
Informational	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 Haedal Protocol to identify any potential issues and vulnerabilities in the source code of the Haedal haeVault v2 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 4 issues of varying severity, listed below.

ID	Title	Severity	Status
MDL-6	`remove_full_range_liquidity_by_percent` Unusable	Minor	Fixed
VAU-3	The First Depositor can Steal Funds from the Second Depositor	Medium	Fixed
VAU-7	`create_vault` Lack Of Inspection	Minor	Fixed
VAU-11	Certified Pool Lacks A Removal Function	Medium	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the Haedal haeVault v2 Smart Contract :

Admin

- `set_roles` : Set or fully override all role permissions for the member of the specified address
- `add_role` : Add a new role for the member at the specified address
- `remove_role` : Remove the specific role of the member with the specified address
- `remove_member` : Completely remove a member and all its role permissions from the ACL
- `emergency_unpause` : Lift the emergency suspension of the agreement
- `create_vault` : Create a new instance of Vault
- `update_active_status` : Update the activity status of Vault
- `update_hard_cap` : Update the upper limit of the total asset value that Vault can manage
- `claim_protocol_fee` : Allow protocol fee collectors to withdraw the accumulated protocol fees of the specified type in Vault
- `update_protocol_fee_rate` :Update the protocol rate of Vault
- `add_dlmm_market` : Add and initialize a DLMM module for Vault
- `add_dlmm_certified_pool` :Adding an external DLMM pool as Vault's authentication pool allows Vault to conduct market-making operations within this pool
- `remove_dlmm_certified_pool` :Remove an external DLMM pool Vault's authentication pool
- `update_dlmm_certified_pool` : Update the configuration parameters of an authenticated DLMM pool

- `open_dlmm_position` : Open a new liquidity position in the specified DLMM pool using the assets in the Vault buffer pool
- `new_dlmm_add_liquidity_cert` : Create an authorization certificate to increase liquidity to the existing DLMM position
- `add_dlmm_liquidity_on_bin` : Increase liquidity within the specific price range of the DLMM position
- `repay_dlmm_add_liquidity` : Operation of increasing liquidity to the existing DLMM position
- `open_position_bid_ask` : Open a liquidity position for the buy and sell order strategy in the DLMM pool
- `open_position_spot` : Open a spot strategy position in the DLMM pool
- `open_position_curve` : Open a position for the curve strategy in the DLMM pool
- `add_dlmm_liquidity` : Increase liquidity across multiple specified price ranges
- `add_dlmm_liquidity_spot` : Increase liquidity in accordance with the spot strategy
- `add_dlmm_liquidity_curve` : Increase liquidity based on the curve strategy
- `add_liquidity_bid_ask` : Increase liquidity based on the buy and sell order strategy
- `remove_dlmm_liquidity_by_percent` : Remove liquidity by a certain percentage and return the withdrawn assets to Vault's buffer pool
- `close_dlmm_position` : Withdraw all liquidity to the buffer pool and return a closing certificate to collect the remaining rewards
- `collect_dlmm_reward_from_close_cert` : Use the vouchers obtained when closing a position to claim the remaining unclaimed rewards of that position
- `flash_loan` : Borrow a flash loan from Vault's buffer pool and return the lent token and a certificate containing repayment information
- `repay_flash_loan` : Repay a flash loan
- `upgrade` : Upgrade the version number to the current protocol version

- `add_oracle_info` : Register a new oracle message for the specified currency type
- `add_oracle_info_v2` : Like add_oracle_info, register a new oracle information for the specified currency type
- `remove_oracle_info` : Used to remove oracle information of the specified currency type
- `update_price_age` : Update the maximum valid duration of the registered currency price

Vault Manager

- `update_protocol_fee_rate` : Update the handling fee rate of the agreement
- `set_swap_slippage` : Set or update transaction slippage for specific token types

Emergency Pause Role

- `emergency_pause` : Emergency Suspension Agreement

User

- `deposit` :Users deposit two types of base tokens (CoinA and CoinB) into Vault, and based on the value of the deposited assets and the current total AUM of Vault, they mint and return the corresponding number of LP tokens
- `calculate_aum` : Calculate and update the total managed assets of Vault
- `calculate_dlmm_position_amounts` : Update the asset quantity of a specific DLMM position recorded internally by Vault
- `new_withdraw_cert` : User destroys the LP tokens they hold in exchange for a withdrawal voucher
- `withdraw_asset` :Use a finalised withdrawal voucher to withdraw the specific type of asset it is entitled to
- `destory_withdraw_cert` : Destroy a withdrawal voucher that has completed the withdrawal of all assets

- `finalize_buffer_assets` : After all position liquidity, fees and rewards have been withdrawn from the external agreement, calculate the final share of various assets of the user in the Vault buffer pool and update the withdrawal credentials
- `withdraw_dlmm` : Withdraw the corresponding proportion of liquidity from the designated DLMM pool and positions based on the user's share in the withdrawal voucher
- `collect_dlmm_fee_on_withdraw` : Collect accumulated transaction fees from the specified DLMM position
- `collect_dlmm_fee` : Collect accumulated transaction fees from the specified DLMM position
- `collect_dlmm_reward_on_withdraw` : Collect specific types of reward tokens from the designated DLMM positions
- `collect_dlmm_reward` : Collect specific types of reward tokens from the designated DLMM positions
- `deposit_fee` : Deposit the fee and use it to cover the network cost when updating the Pyth oracle price subsequently
- `update_price` : Pay fees from the contract fee pool and invoke the Pyth contract to update the on-chain price

4 Findings

MDL-6 `remove_full_range_liquidity_by_percent` Unusable

Severity: Minor

Status: Fixed

Code Location:

packages/volatile_vault_v2/sources/market_dlmm.move#414

Descriptions:

When a function is declared as public(package), it means that this function can only be called internally, but the code that calls this function is not found in the overall code

```
public(package) fun remove_full_range_liquidity_by_percent<CoinTypeA, CoinTypeB>(
    dlmm_market: &mut DLMMMarket,
    pool: &mut Pool<CoinTypeA, CoinTypeB>,
    position_id: ID,
    min_bin_id: u32,
    max_bin_id: u32,
    lp_amount: u128,
    total_lp_amount: u128,
    config: &DLMMGlobalConfig,
    versioned: &DLMMVersioned,
    clk: &Clock,
    ctx: &TxContext,
):
```

Suggestion:

Delete the code

Resolution:

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

VAU-3 The First Depositor can Steal Funds from the Second Depositor

Severity: Medium

Status: Fixed

Code Location:

packages/volatile_vault_v2/sources/vault.move

Descriptions:

In the `deposit()` function, the LP amount is calculated as:

```
lp_amount = total_amount * user_tvl / last_aum
```

```
// calculate the lp token amount need to mint to user.
let token_amount = get_lp_amount_by_tvl(total_lp_amount, user_tvl, vault.last_aum);
assert!(token_amount > 0, ELPTokenAmountIsZero);
assert!(
    token_amount <= (std::u64::max_value!() as u128) - (total_lp_amount as u128),
    ELPTokenAmountOverflow,
);
```

Here, `total_amount` represents the total value of all assets in open positions and `buffer_assets` , both valued relative to the quote token.

```
while (idx < buffer_assets.length()) {
    let (k, v) = buffer_assets.get_entry_by_idx(idx);
    if ((*v > 0) && (pyth_oracle.contain_oracle_info(*k))) {
        if (valid_assets.contains(k)) {
            let amount = valid_assets.get_mut(k);
            *amount = *amount + *v;
        } else {
            valid_assets.insert(*k, *v);
        };
    };
    idx = idx + 1;
};

let aum = calculate_tvl_base_on_quote(
    pyth_oracle,
    &valid_assets,
    vault.quote_type,
    clk,
);
```

The protocol allows keeper manager to perform flash loans. When repaying a flash loan, it enforces the condition `repay_coin.value() >= repay_amount`, after which the repaid assets are joined into `vault.buffer_assets`. This means a user can **repay more than required**.

```
assert!(type_name::with_defining_ids<CoinType>() == repay_type, EIncorrectRepay);
    assert!(repay_coin.value() >= repay_amount, EIncorrectRepay);
    assert!(object::id(vault) == vault_id, EIncorrectRepay);

    let repay_amount = repay_coin.value();
    vault.buffer_assets.join(repay_coin.into_balance());
    emit(RepayFlashLoanEvent {
        repay_type,
        vault_id: object::id(vault),
        repay_amount,
    });
```

The `finalize_buffer_assets()` function later allows users to withdraw assets from the protocol.

```
  let assets = *vault.buffer_assets.balances();
    let mut buffer_amounts = vec_map::empty<TypeName, u64>();
    let mut idx = 0;
    while (idx < vec_map::length(&assets)) {
        let (ty, asset_amount) = assets.get_entry_by_idx(idx);
        let user_share = get_user_share_by_lp_amount(
            withdraw_cert.total_lp_amount,
            withdraw_cert.user_lp_amount,
            *asset_amount as u128,
        );
        buffer_amounts.insert(*ty, user_share as u64);
        idx = idx + 1;
    };
    withdraw_cert.buffer_amounts = buffer_amounts;
    withdraw_cert.state = WithdrawStateEnum::FinishInitedBufferAssets;
```

An attack scenario arises as follows:

1. The first depositor(keeper manager) provides a very small amount and receives **1 share**, making `total_lp_amount = 1` and `last_aum = 1`.
2. The attacker observes that another user intends to deposit assets worth `299 * 1e9`.
3. The attacker takes a flash loan and **over-repays** by `200 * 1e9`, artificially inflating the vault's asset value — now `last_aum = 200 * 1e9 + 1`.
4. The second user deposits `299 * 1e9` worth of assets. According to the formula,

   ```
   lp_amount = 299 * 1e9 * 1 / (200 * 1e9 + 1) ≈ 1
   ```

   due to rounding down.
5. The attacker then withdraws. Since he holds half of the shares, he can withdraw

   ```
   ((200 + 299) * 1e9 + 1) / 2 = 249,500,000,000
   ```

   resulting in a profit of
   `249,500,000,000 - 200 * 1e9 = 49,500,000,000`

Suggestion:

Recommendation: The project team should make the first deposit with a sufficiently large amount of tokens.

Resolution:

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

VAU-7 `create_vault` Lack Of Inspection

Severity: Minor

Status: Fixed

Code Location:

packages/volatile_vault_v2/sources/vault.move#344

Descriptions:

In the `create_vault` function, the `protocol_fee_rate` parameter was not compared and verified with the global maximum handling fee rate `max_protocol_fee_rate()` defined in the `config` module when it was set. However, this verification exists in the `update_protocol_fee_rate` function. This inconsistency may lead to setting a protocol rate that far exceeds the upper limit of the protocol design when creating a Vault.

```
public fun create_vault<CoinTypeA, CoinTypeB, LPCoin>(
    registry: &mut VaultRegistry,
    lp_token_treasury: TreasuryCap<LPCoin>,
    pyth_oracle: &PythOracle,
    quote_type: u8,
    hard_cap: u128,
    protocol_fee_rate: u64,
    config: &GlobalConfig,
    versioned: &Versioned,
    ctx: &mut TxContext,
)
```

Suggestion:

Add checks

Resolution:

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

VAU-11 Certified Pool Lacks A Removal Function

Severity: Medium

Status: Fixed

Code Location:

packages/volatile_vault_v2/sources/vault.move#500

Descriptions:

The add_dlmm_certified_pool function allows the administrator (Vault Manager) to add an external DLMM pool as an "certified pool". However, a corresponding deletion and removal function is completely missing in the contract. This means that adding an authentication pool is a permanent and irreversible operation.

Suggestion:

Add and remove functions

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