SHERLOCK

Security Review For Vesu

Collaborative Audit Prepared For: **Vesu**
Lead Security Expert(s): **CODESPECT**
Date Audited: **November 10 - November 13, 2025**

Introduction

Vesu is a fully open and permissionless lending protocol built on Starknet. Users can supply crypto assets (earn), borrow crypto assets and build new lending experiences on Vesu without relying on intermediaries. The Vesu lending protocol is not controlled by a governance body and there exists no governance token. Instead, Vesu is built as a public infrastructure giving everyone equal access to all functions and is free for everyone to use.

Scope

Repository: vesuxyz/vesu-v2-periphery

Audited Commit: c07c27212e5d1da8ee2f03d0c6dced8f3f458c2a

Final Commit: ccea4e3761da0bb35e701bfe65760e312178eeb6

Files:

- src/lib.cairo
- src/liquidate.cairo
- src/migrate.cairo
- src/multiply.cairo
- src/swap.cairo

Final Commit Hash

ccea4e3761da0bb35e701bfe65760e312178eeb6

Findings

Each issue has an assigned severity:

- High issues are directly exploitable security vulnerabilities that need to be fixed.
- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- Low/Info issues are non-exploitable, informational findings that do not pose a security risk or impact the system's integrity. These issues are typically cosmetic or related to compliance requirements, and are not considered a priority for remediation.

Issues Found

High	Medium	Low/Info
1	1	0

Issues Not Fixed and Not Acknowledged

High	Medium	Low/Info
0	0	0

Issue H-1: Missing ownership validation of positions allows unauthorized migration [RESOLVED]

Source: https://github.com/sherlock-audit/2025-11-vesu-nov-10th/issues/6

Summary

The migrator contract does not verify whether the caller is the actual owner of the position being migrated. If a user previously delegated the migrator contract to manage their V1/V2 position, an attacker could exploit this oversight to migrate positions without authorization and effectively steal user assets.

Vulnerability Detail

The function responsible for migrating a V2 position (with a similar implementation for V1) lacks a proper ownership validation. As seen in the code below, there is no check ensuring that `get_contract_address() == from_user`, which allows arbitrary migration of positions and potential theft of user funds.

```
fn migrate_position_from_v2(ref self: ContractState, params:
↪  MigratePositionFromV2Params) {
    let MigratePositionFromV2Params {
        from_pool, to_pool, collateral_asset, debt_asset, from_user,
        ↪  debt_to_migrate, ..,
    } = params.clone();

    let from_pool = IPoolDispatcher { contract_address: from_pool };
    let to_pool = IPoolDispatcher { contract_address: to_pool };
    let (_, _, debt) = from_pool.position(collateral_asset, debt_asset, from_user);
    let debt = if (debt_to_migrate == 0 || debt_to_migrate > debt) {
        debt
    } else {
        debt_to_migrate
    };

    let migrate_action = MigrateAction::MigratePositionFromV2(params);
    let mut data: Array<felt252> = array![];
    Serde::serialize(@migrate_action, ref data);
    // @audit Missing ownership validation before calling flashloan
    self.call_flash_loan(pool: to_pool, asset: debt_asset, amount: debt, data:
    ↪  data.span());
}
```

Impact

An attacker can steal previously delegated user positions by triggering unauthorized migrations. Users who granted delegation to the migrator and did not revoke it remain vulnerable to having their positions migrated and assets drained.

Code Snippet

vesu-v2-periphery/src/migrate.cairo#L386

Tool Used

Manual Review

Recommendation

Add a strict ownership validation to ensure the migration can only be initiated by the actual position owner.

```
+ assert(get_contract_address() == from_user, "unauthorized: caller is not the
↪  position owner");
```

Additionally, consider revoking migrator permissions automatically after migration or enforcing explicit re-authorization to minimize attack surface.

Discussion

jo-es

The idea is to wrap the migrate call within delegate, undelegate calls. If we perform explicit ownership checks then the migrator contract can't be used for users that deploy proxies to manage their positions. e.g. for a virtual multi account support.

Kalogerone

> The idea is to wrap the migrate call within delegate, undelegate calls. If we perform explicit ownership checks then the migrator contract can't be used for users that deploy proxies to manage their positions. e.g. for a virtual multi account support.

It is one of the options we think about but that does not fix the issue, it is still there if the user did not remove delegation.

You can implement a check if it's your contract calling the migration to skip the `from_user == get_caller_address()` check.

jo-es

> It is one of the options we think about but that does not fix the issue, it is still there if the user did not remove delegation.

Don't understand the response. I wasn't proposing a fix.

> You can implement a check if it's your contract calling the migration to skip the from_-user == get_caller_address() check.

If you pass it in as a flag then everyone can set it to false to skip that check

talfao

We understand that you were not proposing a fix; we only pointed out that your mitigation works, although it does not eliminate the issue itself.

We believe that if you do not implement an ownership check for the position, the issue should be marked as Acknowledged with a note explaining your mitigation.

It is important to communicate this clearly, because if users interact with the migrator without your frontend, they could lose funds due to this implementation.

jo-es

We've added a check for delegation: https://github.com/vesuxyz/vesu-v2-periphery/pull/3/commits/74e963c2d3cd02d559c11607c548a40c177fa8a0

talfao

Fix looks good!

Issue M-1: DOS for certain scenarios depending on the LTVs of the 2 positions [RESOLVED]

Source: https://github.com/sherlock-audit/2025-11-vesu-nov-10th/issues/7

Summary

The `Migrate.cairo` contract allows existing users to migrate their positions. Users are required to choose a `max_ltv_delta` value, which ensures that their new position won't be liquidatable in case of differences in oracle price or interest rates between pools and pairs. However, depending on the old and the new position LTVs, it is possible that there is no `max_ltv_delta` value that will allow the transaction to complete.

Vulnerability Detail

The issue lies in the final check of the LTVs:

```
fn create_v2_position(
    ref self: ContractState,
    to_pool: IPoolDispatcher,
    to_user: ContractAddress,
    mut collateral_asset: ContractAddress,
    mut debt_asset: ContractAddress,
    collateral_delta: u256,
    debt_delta: u256,
    from_ltv: u256,
    max_ltv_delta: u256,
) {
    // ...

    let (_, collateral_value, debt_value) = to_pool
        .check_collateralization(collateral_asset, debt_asset, to_user);
    let to_ltv = debt_value * SCALE / collateral_value;

    assert!(from_ltv - max_ltv_delta <= to_ltv && to_ltv <= from_ltv +
    ↪  max_ltv_delta, "ltv-out-of-range");

    // ...
}
```

Here is a scenario that no `max_ltv_delta` satisfies the assertion:

A user is migrating from a position with 0 debt and only collateral, so `from_ltv` is `0`. This means that `max_ltv_delta` can only be `0`, else the `from_ltv - max_ltv_delta` calculation will revert. If the user is migrating to an existing position with `ltv > 0`, then after the migration, the `to_ltv` will be greater than `0`. Since `max_ltv_delta` is `0` for the previous

reason, the calculation `to_ltv <= from_ltv + max_ltv_delta` can never be true, since `from_ltv + max_ltv_delta = 0`.

The issue is caused because the `max_ltv_delta` is bounded by the `from_ltv` value, so it doesn't underflow and revert. If the initial position has less LTV than the position that it gets migrated to, this DOS issue can happen.

Impact

Migrations can never work for certain positions, depending on their LTVs before the migration starts.

Code Snippet

vesu-v2-periphery/src/migrate.cairo#L319

Tool Used

Manual Review

Recommendation

If the `max_ltv_delta` is bigger than the `from_delta` consider skipping the first check, to allow for bigger `max_ltv_delta` values for the next check.

Discussion

talfao

Looks fixed

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users' sole risk and is the users' responsibility.