CAIRO SECURITY CLAN

VESU - PERIPHERY UPDATE

SECURITY ASSESMENT REPORT

MARCH 2025

Prepared for

VESU

Contents

1 About Cairo Security Clan

Cairo Security Clan is a leading force in the realm of blockchain security, dedicated to fortifying the foundations of the digital age. As pioneers in the field, we specialize in conducting meticulous smart contract security audits, ensuring the integrity and reliability of decentralized applications built on blockchain technology.

At Cairo Security Clan, we boast a multidisciplinary team of seasoned professionals proficient in blockchain security, cryptography, and software engineering. With a firm commitment to excellence, our experts delve into every aspect of the Web3 ecosystem, from foundational layer protocols to application-layer development. Our comprehensive suite of services encompasses smart contract audits, formal verification, and real-time monitoring, offering unparalleled protection against potential vulnerabilities.

Our team comprises industry veterans and scholars with extensive academic backgrounds and practical experience. Armed with advanced methodologies and cutting-edge tools, we scrutinize and analyze complex smart contracts with precision and rigor. Our track record speaks volumes, with a plethora of published research papers and citations, demonstrating our unwavering dedication to advancing the field of blockchain security.

At Cairo Security Clan, we prioritize collaboration and transparency, fostering meaningful partnerships with our clients. We believe in a customer-oriented approach, engaging stakeholders at every stage of the auditing process. By maintaining open lines of communication and soliciting client feedback, we ensure that our solutions are tailored to meet the unique needs and objectives of each project.

Beyond our core services, Cairo Security Clan is committed to driving innovation and shaping the future of blockchain technology. As active contributors to the ecosystem, we participate in the development of emerging technologies such as Starknet, leveraging our expertise to build robust infrastructure and tools. Through strategic guidance and support, we empower our partners to navigate the complexities of the blockchain landscape with confidence and clarity.

In summary, Cairo Security Clan stands at the forefront of blockchain security, blending technical prowess with a client-centric ethos to deliver unparalleled protection and peace of mind in an ever-evolving digital landscape. Join us in safeguarding the future of decentralized finance and digital assets with confidence and conviction.

2 Disclaimer

Disclaimer Limitations of this Audit:

This report is based solely on the materials and documentation provided by you to Cairo Security Clan for the specific purpose of conducting the security review outlined in the Summary of Audit and Scoped Files. The findings presented here may not be exhaustive and may not identify all potential vulnerabilities. Cairo Security Clan provides this review and report on an "as-is" and "as-available" basis. You acknowledge that your use of this report, including any associated services, products, protocols, platforms, content, and materials, occurs entirely at your own risk.

Inherent Risks of Blockchain Technology:

Blockchain technology remains in its developmental stage and is inherently susceptible to unknown risks and vulnerabilities. This review is specifically focused on the smart contract code and does not extend to the compiler layer, programming language elements beyond the reviewed code, or other potential security risks outside the code itself.

Report Purpose and Reliance:

This report should not be construed as an endorsement of any specific project or team, nor does it guarantee the absolute security of the audited smart contracts. No third party should rely on this report for any purpose, including making investment or purchasing decisions.

Liability Disclaimer:

To the fullest extent permitted by law, Cairo Security Clan disclaims all liability associated with this report, its contents, and any related services and products arising from your use. This includes, but is not limited to, implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Third-Party Products and Services:

Cairo Security Clan does not warrant, endorse, guarantee, or assume responsibility for any products or services advertised by third parties within this report, nor for any open-source or third-party software, code, libraries, materials, or information linked to, referenced by, or accessible through this report, its content, and related services and products. This includes any hyperlinked websites, websites or applications appearing on advertisements, and Cairo Security Clan will not be responsible for monitoring any transactions between you and third-party providers. It is recommended that you exercise due diligence and caution when considering any third-party products or services, just as you would with any purchase or service through any medium.

Disclaimer of Advice:

FOR THE AVOIDANCE OF DOUBT, THIS REPORT, ITS CONTENT, ACCESS, AND/OR USE, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHOULD NOT BE CONSIDERED OR RELIED UPON AS FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER PROFESSIONAL ADVICE.

3 Executive Summary

This document presents the security review performed by Cairo Security Clan on the Vesu protocol.

Vesu, DeFi's latest progression in the on-chain lending space, is a pioneering platform designed to facilitate fully permissionless, over-collateralized lending agreements. With its ambitious design, Vesu looks to combine the best aspects of both worlds: a liquidity monolith with permissionless, multi-asset lending compartments aka lending pools. Learn more from docs.

The audit was performed using

- manual analysis of the codebase,
- automated analysis tools,
- simulation of the smart contract,
- analysis of edge test cases

2 points of attention, where 0 is classified as Critical, 0 is classified as High, 0 is classified as Medium,0 is classified as Low,2 are classified as Informational and 0 is classified as Best Practices. The issues are summarized in Fig. 1.

This document is organized as follows. Section 1 About Cairo Security Clan. Section 2 Disclaimer. Section 3 Executive Summary. Section 4 Summary of Audit. Section 5 Risk Classification. Section 6 Issues by Severity Levels. Section 7 Test Evaluation.

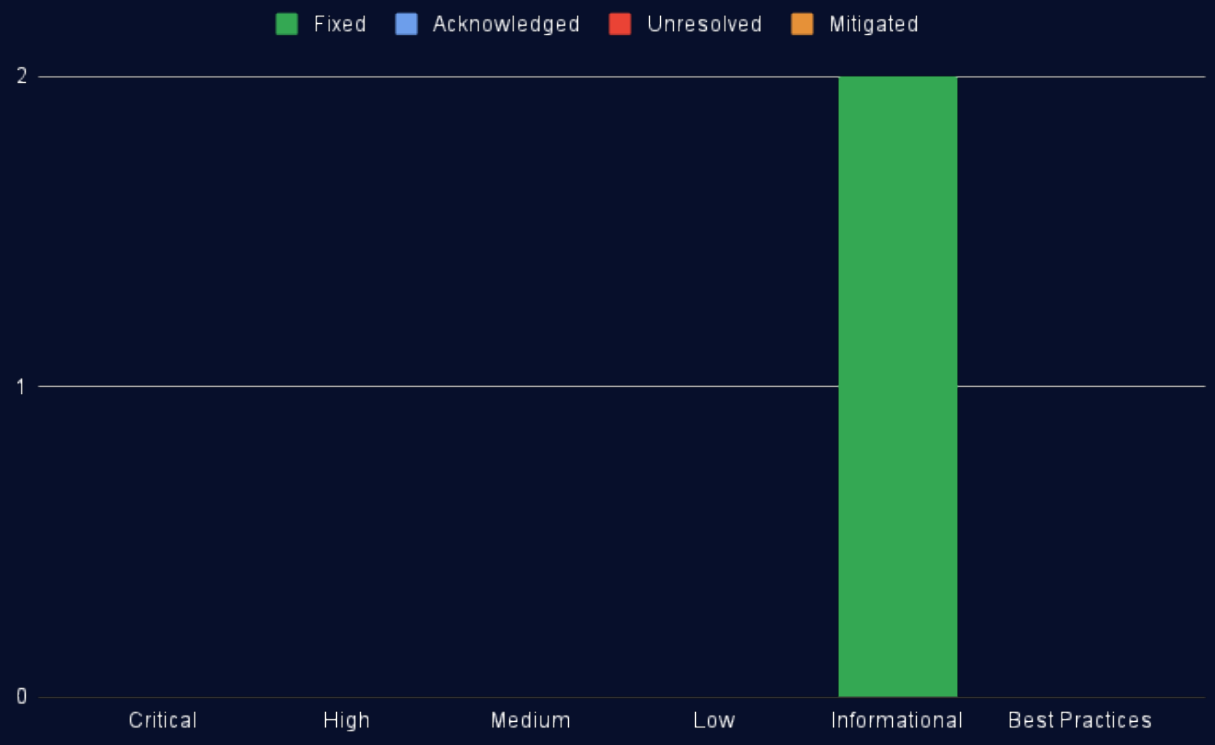

Fig 1: Distribution of issues: Critical (0), **High** (0), **Medium** (0), **Low** (0), **Informational** (2), **Best Practices** (0). **Distribution of status: Fixed** (2), **Acknowledged** (0), **Mitigated** (0), **Unresolved** (0).

4 Summary of Audit

Audit Type	Security Review
Cairo Version	2.6.3
Final Report	12/03/2025
Repository	vesuxyz/vesu-periphery
Initial Commit Hash	ddc01ea38054ad68e1cb194f11c4cc7732959352
Final Commit Hash	5f1274f90ea49dda43fab3f1e69b87b70ed482a0
Documentation	Website documentation
Test Suite Assessment	High

4.1 Scoped Files

	Contracts
1	src/rebalance.cairo
2	src/multiply4626.cairo

4.2 Issues

	Findings	Severity	Update
1	Approval amount is more than needed in case `add_margin_is_wrapped = true`	Informational	Fixed
2	Missing check for `add_margin_is_wrapped` when swapping margin asset to debt asset	Informational	Fixed

5 Risk Classification

The risk rating methodology used by Cairo Security Clan follows the principles established by the CVSS risk rating methodology. The severity of each finding is determined by two factors: **Likelihood** and **Impact**.

Likelihood measures how likely an attacker will uncover and exploit the finding. This factor will be one of the following values:

a) **High**: The issue is trivial to exploit and has no specific conditions that need to be met;

b) **Medium**: The issue is moderately complex and may have some conditions that need to be met;

c) **Low**: The issue is very complex and requires very specific conditions to be met.

When defining the likelihood of a finding, other factors are also considered. These can include but are not limited to Motive, opportunity, exploit accessibility, ease of discovery, and ease of exploit.

Impact is a measure of the damage that may be caused if an attacker exploits the finding. This factor will be one of the following values:

a) **High**: The issue can cause significant damage such as loss of funds or the protocol entering an unrecoverable state;

b) **Medium**: The issue can cause moderate damage such as impacts that only affect a small group of users or only a particular part of the protocol;

c) **Low**: The issue can cause little to no damage such as bugs that are easily recoverable or cause unexpected interactions that cause minor inconveniences.

When defining the impact of a finding other factors are also considered. These can include but are not limited to Data/state integrity, loss of availability, financial loss, and reputation damage. After defining the likelihood and impact of an issue, the severity can be determined according to the table below.

		Likelihood		
		High	**Medium**	**Low**
Impact	**High**	Critical	High	Medium
	Medium	High	Medium	Low
	Low	Medium	Low	Info/Best Practices

To address issues that do not fit a High/Medium/Low severity, Cairo Security Clan also uses three more finding severities: **Informational**, **Best Practices** and **Gas**

a) **Informational** findings do not pose any risk to the application, but they carry some information that the audit team intends to formally pass to the client;

b) **Best Practice** findings are used when some piece of code does not conform with smart contract development best practices;

c) **Gas** findings are used when some piece of code uses more gas than it should be or have some functions that can be removed to save gas.

6 Issues by Severity Levels

6.1 Informational

6.1.1 Approval amount is more than needed in case `add_margin_is_wrapped = true`

File(s): multiply4626.cairo

Description: When `add_margin_is_wrapped = true`, the margin asset deposited is already in the form of the wrapped asset (Vault token). This means it does not need to be deposited into the ERC4626 Vault again.

In the current implementation, the transfer logic correctly excludes the `margin_amount` if the margin token is already wrapped. However, the approval logic does not account for this scenario and unnecessarily approves the sum of `margin_amount` and `lever_amount` for the debt asset. This can lead to over-approval, which is inefficient and could pose security risks.

```
// @audit only need to approve lever_amount in case add_margin_is_wrapped == true
IERC20Dispatcher { contract_address: debt_asset }
    .approve(collateral_asset, (margin_amount + lever_amount).into());

// exclude margin_amount if margin token is already the wrapped token
let mut wrapped_amount = I4626Dispatcher { contract_address: collateral_asset }
    .deposit(
        if add_margin_is_wrapped {
            lever_amount.into()
        } else {
            (margin_amount + lever_amount).into()
        },
        get_contract_address()
    );
```

Recommendation(s): Consider updating the approval logic to only approve the `lever_amount` when `add_margin_is_wrapped = true`.

Status: Fixed

Update from the client: Fixed in 5f1274f9

6.1.2 Missing check for `add_margin_is_wrapped` when swapping margin asset to debt asset

File(s): multiply4626.cairo

Description: The Multiply4626 contract is designed to work with tokens that have a 4626 Vault. It allows users to leverage the asset token of the Vault by following these steps:

1. Transfer the margin amount if the margin token is already the desired token, or swap the margin token to the debt token.
2. Flashloan the debt token from **Ekubo**.
3. Deposit the debt token into the Vault to receive Vault tokens.
4. Use the Vault tokens as collateral to borrow the exact flashloaned debt token amount from **Vesu**.
5. Repay the flashloan using the borrowed debt token.

Users are required to provide certain configurations as input parameters to handle the first step. One of these parameters is `add_margin_is_wrapped`, which indicates whether the margin asset is already in the form of the Vault token (wrapped).

However, when the margin asset is swapped, it is always swapped to the debt asset. There is no configuration to swap to the wrapped asset (Vault token), and the code is missing a check to ensure that `add_margin_is_wrapped` is set to `false` in the case of a swap.

```
fn increase_lever(
    ref self: ContractState, increase_lever_params: IncreaseLeverParams
) -> ModifyLeverResponse {
    let core = self.core.read();

    let IncreaseLeverParams { pool_id,
    collateral_asset,
    user,
    add_margin,
    add_margin_is_wrapped,
    margin_swap,
    margin_swap_limit_amount,
    lever_amount } =
        increase_lever_params;

    let debt_asset = I4626Dispatcher { contract_address: collateral_asset }.asset();

    // - swap margin asset to debt asset (1.)
    let margin_amount = if margin_swap.len() != 0 {
        let (margin_amount_, debt_amount_) = swap(
            core, margin_swap.clone(), margin_swap_limit_amount
        );
        assert!(
            add_margin == 0 && debt_amount_.token == debt_asset,
            "invalid-margin-swap-assets"
        ); // @audit consider adding check `add_margin_is_wrapped = false`
```

Recommendation(s): Consider adding a check to ensure that `add_margin_is_wrapped` is `false` when swapping the margin asset to the debt asset.

Status: Fixed

Update from the client: Fixed in f7ee0956

7 Test Evaluation

7.1 Compilation Output

```
scarb build
   Compiling vesu_periphery v0.1.0 (/vesu/Scarb.toml)
    Finished release target(s) in 32 seconds
```

7.2 Tests Output

```
./scripts/test.sh
    Updating git repository github.com/ekuboprotocol/abis
    Updating git repository github.com/vesuxyz/vesu-v1
    Updating git repository github.com/foundry-rs/starknet-foundry
    Updating git repository github.com/keep-starknet-strange/alexandria
   Compiling vesu_periphery v0.1.0 (/vesu-periphery/Scarb.toml)
    Finished release target(s) in 14 seconds


Collected 29 test(s) from vesu_periphery package
Running 0 test(s) from src/
Running 29 test(s) from tests/
[PASS] tests::test_multiply::Test_896150_Multiply::test_modify_lever_no_lever_swap (gas: ~3833)
[PASS] tests::test_liquidate::Test_896150_Liquidate::test_liquidate_position_full_liquidation_multi_swap (gas: ~5642)
[PASS] tests::test_liquidate::Test_896150_Liquidate::test_liquidate_position_partial_liquidation_multi_swap_no_bad_debt (gas: ~6396)
[PASS] tests::test_multiply::Test_896150_Multiply::test_modify_lever_exact_collateral_deposit (gas: ~4555)
[PASS] tests::test_multiply::Test_896150_Multiply::test_modify_lever_margin_asset_swap_exact_in (gas: ~6379)
[PASS] tests::test_multiply::Test_896150_Multiply::test_modify_lever_exact_debt_borrow (gas: ~4555)
[PASS] tests::test_multiply::Test_896150_Multiply::test_modify_lever_exact_collateral_withdrawal (gas: ~7265)
[PASS] tests::test_multiply::Test_896150_Multiply::test_modify_lever_margin_asset_swap_exact_out (gas: ~6517)
[PASS] tests::test_multiply::Test_896150_Multiply::test_modify_lever_exact_collateral_withdrawal_no_lever_swap (gas: ~7113)
[PASS] tests::test_multiply::Test_896150_Multiply::test_modify_lever_close_multi_swap (gas: ~9352)
[PASS] tests::test_multiply::Test_896150_Multiply::test_modify_lever_exact_debt_repay (gas: ~7257)
[PASS] tests::test_multiply::Test_896150_Multiply::test_modify_lever_close_weight_sum_not_1 (gas: ~4587)
[PASS] tests::test_multiply::Test_896150_Multiply::test_modify_lever_withdraw_swap_exact_in (gas: ~7290)
[PASS] tests::test_multiply::Test_896150_Multiply::test_modify_lever_close (gas: ~7909)
[PASS] tests::test_multiply::Test_896150_Multiply::test_modify_lever_multi_swap_limit_amount_exceeded (gas: ~1774)
[PASS] tests::test_multiply::Test_896150_Multiply::test_modify_lever_multi_swap (gas: ~4949)
[PASS] tests::test_rebalance::Test_896150_Rebalance::test_rebalance_set_owner (gas: ~833)
[PASS] tests::test_rebalance::Test_896150_Rebalance::test_rebalance_set_rebalancer (gas: ~833)
[PASS] tests::test_rebalance::Test_896150_Rebalance::test_rebalance_increase_only_rebalancer (gas: ~834)
[PASS] tests::test_rebalance::Test_896150_Rebalance::test_rebalance_increase_target_ltv_min_delta (gas: ~5735)
[PASS] tests::test_multiply::Test_896150_Multiply::test_modify_lever_split_multi_swap (gas: ~6036)
[PASS] tests::test_rebalance::Test_896150_Rebalance::test_rebalance_increase_target_ltv_tolerance (gas: ~8747)
[PASS] tests::test_rebalance::Test_896150_Rebalance::test_rebalance_increase_with_fee (gas: ~11904)
[PASS] tests::test_liquidate::Test_896150_Liquidate::test_liquidate_position_full_liquidation_multi_split_swap_no_bad_debt_weight_sum_not_1 (gas: ~4750)
[PASS] tests::test_rebalance::Test_896150_Rebalance::test_rebalance_decrease_with_fee (gas: ~11551)
[PASS] tests::test_rebalance::Test_896150_Rebalance::test_rebalance_increase_with_fee_split_swap (gas: ~15236)
[PASS] tests::test_liquidate::Test_896150_Liquidate::test_liquidate_position_full_liquidation_multi_swap_no_bad_debt (gas: ~5482)
[PASS] tests::test_liquidate::Test_896150_Liquidate::test_liquidate_position_full_liquidation_multi_split_swap_no_bad_debt (gas: ~26231)
[PASS] tests::test_rebalance::Test_896150_Rebalance::test_rebalance_decrease_split_swap (gas: ~17011)
Tests: 29 passed, 0 failed, 0 skipped, 0 ignored, 4 filtered out
   Compiling vesu_periphery v0.1.0 (/home/runner/work/vesu-periphery/vesu-periphery/Scarb.toml)
    Finished release target(s) in 14 seconds
```

```
Collected 4 test(s) from vesu_periphery package
Running 0 test(s) from src/
Running 4 test(s) from tests/
[PASS] tests::test_multiply4626::Test_974640_Multiply4626::test_modify_lever_4626_xstrk_no_flash_loan (gas: ~5066)
[PASS] tests::test_multiply4626::Test_974640_Multiply4626::test_modify_lever_4626_xstrk (gas: ~5291)
[PASS] tests::test_multiply4626::Test_974640_Multiply4626::test_modify_lever_4626_xstrk_wrapped_margin (gas: ~5514)
[PASS] tests::test_multiply4626::Test_974640_Multiply4626::test_modify_lever_4626_xstrk_margin_swap (gas: ~5911)
Tests: 4 passed, 0 failed, 0 skipped, 0 ignored, 29 filtered out
```