CAIRO SECURITY CLAN

VESU

SECURITY ASSESMENT REPORT

JUNE 2024

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

9 points of attention, where 0 is classified as Critical, 0 is classified as High, 2 are classified as Medium,1 is classified as Low,2 are classified as Informational and 4 are 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.

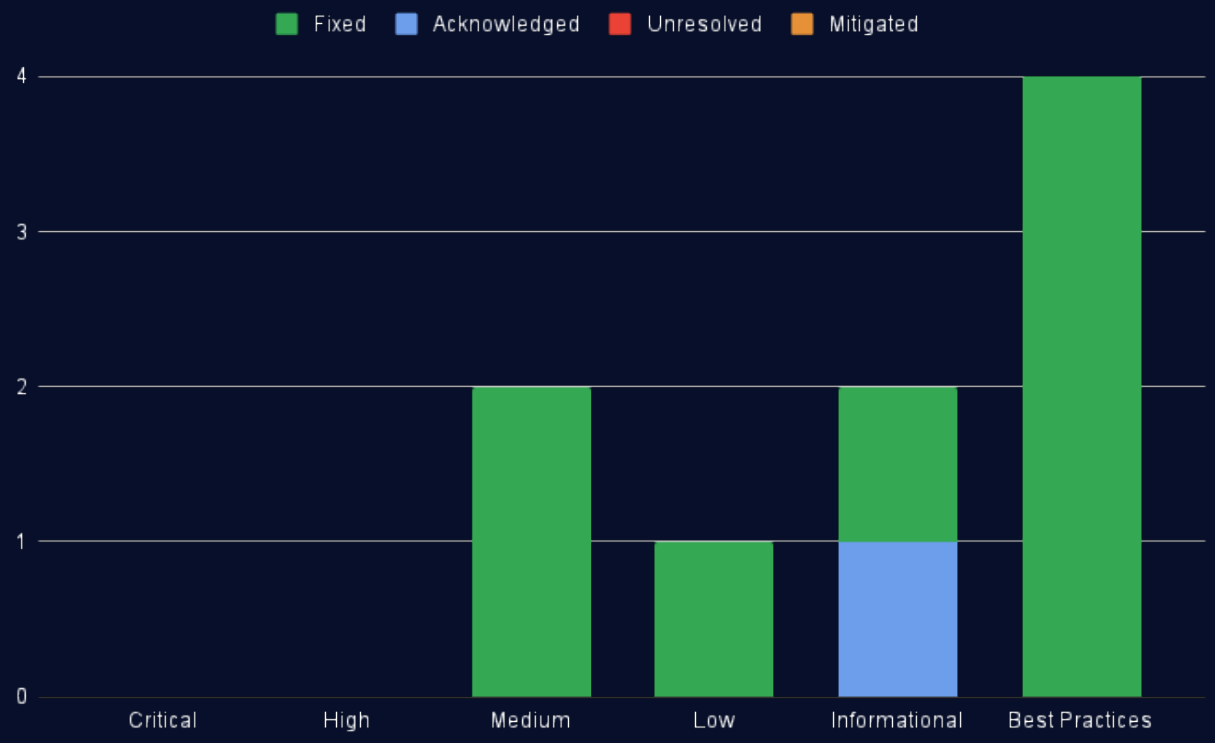

Fig 1: Distribution of issues: Critical (0), **High** (0), **Medium** (2), **Low** (1), **Informational** (2), **Best Practices** (4). **Distribution of status: Fixed** (8), **Acknowledged** (1), **Mitigated** (0), **Unresolved** (0)

4 Summary of Audit

Audit Type	Security Review
Cairo Version	2.6.5
Initial Report	29/06/2024
Response from Client	29/06/2024
Final Report	02/07/2024
Repository	vesu-v1
Initial Commit Hash	e7d122f99262eb2b94d5152ed7af4dc9d0684527
Final Commit Hash	77d3cc996273f0654b1f445c08ad2071c3e6e040
Documentation	Website documentation
Test Suite Assessment	High

4.1 Scoped Files

	Contracts
1	/src/common.cairo
2	/src/data_model.cairo
3	/src/lib.cairo
4	/src/map_list.cairo
5	/src/math.cairo
6	/src/packing.cairo
7	/src/singleton.cairo
8	/src/units.cairo
9	/src/v_token.cairo
10	/src/extension/default_extension.cairo
11	/src/extension/interface.cairo
12	/src/extension/components/fee_model.cairo
13	/src/extension/components/interest_rate_model.cairo
14	/src/extension/components/position_hooks.cairo
15	/src/extension/components/pragma_oracle.cairo
16	/src/extension/components/tokenization.cairo

4.2 Issues

	Findings	Severity	Update
1	The invalid price could be used for liquidation in normal mode.	Medium	Fixed
2	Function `attribute_fee_shares()` not called when writing new asset config results in fee being stuck in Singleton.	Medium	Fixed
3	Potential denial of service during pool creation.	Low	Fixed
4	The function `pow(...)` might result with unexpected behavior if base is zero.	Informational	Fixed
5	Users can front-run to withdraw before bad debt is accounted.	Informational	Acknowledged
6	Incorrect documentation for function `pow_scale()`	Best Practices	Fixed
7	Potentially misleading function naming for `assert_config_exists`	Best Practices	Fixed
8	Incorrect documentation for function `claim_fees`	Best Practices	Fixed
9	Inconsistent LTV config validation between `create_pool` and `set_ltv_config`.	Best Practices	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;

b) **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 Medium

6.1.1 The invalid price could be used for liquidation in normal mode

File(s):/src/extension/components/position_hooks.cairo

Description: Given that the "&&" operator has higher precedence than the "||" operator, the function `before_liquidate_position()` only checks if the price is valid for Recovery mode, not for None mode.

```
let shutdown_mode = self.update_shutdown_status(ref context);
assert!(
    shutdown_mode == ShutdownMode::None || shutdown_mode == ShutdownMode::Recovery
        && context.collateral_asset_price.is_valid
        && context.debt_asset_price.is_valid,
    "emergency-mode"
);
}
```

The price is fetched from the Pragma oracle. If Pragma returns a timeout price or with fewer sources than configured, the price will still be accepted for use in liquidation.

Recommendation(s): Consider changing the check condition to

```
(shutdown_mode == ShutdownMode::None || shutdown_mode == ShutdownMode::Recovery)
```

Status: Fixed

Update from client: Fixed in commit.

6.1.2 Function `attribute_fee_shares()` not called when writing new asset config results in fee being stuck in Singleton

File(s):/src/singleton.cairo

Description: Upon getting the asset config, `fee_shares` is already added to `total_collateral_shares`. This new `asset_config` has not yet been written to storage. When this happens, the contract should also call `attribute_fee_shares()` to account the `fee_shares` to the extension address. For instance, `donate_to_reserve()` calls this function before writing the new asset config to storage.

```
fn donate_to_reserve(ref self: ContractState, pool_id: felt252, asset: ContractAddress, amount: u256) {
    let (mut asset_config, fee_shares) = self.asset_config(pool_id, asset);
    assert!(assert_config_exists(asset_config), "asset-config-nonexistent");
    self.attribute_fee_shares(pool_id, self.extensions.read(pool_id), asset, fee_shares);
    asset_config.reserve += amount;
    self.asset_configs.write((pool_id, asset), asset_config);
    transfer_asset(asset, get_caller_address(), get_contract_address(), amount, asset_config.is_legacy);

    self.emit(Donate { pool_id, asset, amount });
}
```

However, there are instances where `attribute_fee_shares()` is not called when writing the new asset config to storage, such as in `set_asset_parameter()`. As a result, `total_collateral_shares` increases, but this additional share does not get accounted for in any address. This means these shares cannot be redeemed and remain stuck indefinitely.

```
fn set_asset_parameter(
    ref self: ContractState, pool_id: felt252, asset: ContractAddress, parameter: felt252, value: u256
) {
    assert!(!self.lock.read(), "set-asset-parameter-reentrancy");
    assert!(get_caller_address() == self.extensions.read(pool_id), "caller-not-extension");

    let (mut asset_config, fee_shares) = self.asset_config(pool_id, asset);

    if parameter == 'max_utilization' {
        asset_config.max_utilization = value;
    } else if parameter == 'floor' {
        asset_config.floor = value;
    } else if parameter == 'fee_rate' {
        asset_config.fee_rate = value;
        self.attribute_fee_shares(pool_id, self.extensions.read(pool_id), asset, fee_shares);
    } else {
        panic!("invalid-asset-parameter");
    }

    assert_asset_config(asset_config);
    assert_storable_asset_config(asset_config);
    self.asset_configs.write((pool_id, asset), asset_config);

    self.emit(SetAssetParameter { pool_id, asset, parameter, value });
}
```

Recommendation(s): Always call `attribute_fee_shares()` when writing new asset config to storage.

Status: Fixed

Update from client: Fixed in commit.

6.2 Low

6.2.1 Potential denial of service during pool creation

File(s): /src/extension/components/tokenization.cairo

Description: When creating a pool through `default_extension::create_pool`, a VToken is deployed for each asset. The deployment syscall used in `create_v_token` has the field `deploy_from_zero` set to `true`, meaning that the address from which the token is being deployed has no impact on the address of the new token contract.

```
let (v_token, _) = (deploy_syscall(
    self.v_token_class_hash.read().try_into().unwrap(),
    0,
    array![
        v_token_name.into(),
        v_token_symbol.into(),
        18,
        pool_id,
        get_contract_address().into(),
        collateral_asset.into()
    ]
        .span(),
    true
))
```

This allows a VToken to be deployed from any contract address, and if the constructor arguments are the same, the same resulting address will be calculated. A malicious actor could preemptively deploy a VToken with the same argument data using their own contract before the `create_pool` transaction executes, causing a deployment collision when the victim transaction occurs. While the previous scenario is unlikely, when Starknet sequencers are decentralized the effects of this issue will be significantly increased, as a malicious actor could monitor the mempool to reliably frontrun and revert all pool creation transactions.

Recommendation(s): Change the `deploy_syscall` argument `deploy_from_zero` to false to ensure that the deployed VToken address will be unique, preventing potential collisions.

Status: Fixed

Update from client: Fixed in commit.

6.3 Informationals

6.3.1 The function `pow(...)` might result with unexpected behavior if base is zero

File(s): /src/math.cairo

Description: The `pow(...)` function takes power of parameter x. However, the x value can be zero, and that edge case is not taken care of.

```
fn pow(x: u256, n: usize) -> u256 {
    if n == 0 {
        1
    } else if n == 1 {
        x
    } else if (n & 1) == 1 {
        x * pow(x * x, n / 2)
    } else {
        pow(x * x, n / 2)
    }
}
```

There is no case for calling this function as x is zero. However, it is a utility function, and it can be used in future features.

Recommendation(s): Consider handling edge case that if x is zero.

Status: Fixed

Update from client: Fixed in commit.

6.3.2 Users can front-run to withdraw before bad debt is accounted

File(s): /src/singleton.cairo

Description: The Vesu lending pool utilizes a bad debt socialization mechanism. If there's insufficient collateral to cover the debt, liquidation results in bad debt. This bad debt is then attributed to the pool and distributed among the lenders of the corresponding collateral asset. The liquidator receives all the collateral but is only required to repay the proportionate debt value.

In this process, all remaining lenders of the corresponding collateral asset share the bad debt (loss). However, as bad debt is only accounted for when liquidation occurs, and lenders can withdraw their funds at any time, a lender can monitor all open positions and withdraw their funds if any position is likely to accrue bad debt before being liquidated. In this way, they can consistently avoid losses due to bad debt.

Ultimately, anytime there is a bad debt position, users are going to have a gas war or bank run to withdraw funds from the protocol to avoid the bad debt.

Recommendation(s): The bad debt socialization mechanism is risky and needs careful design. Consider covering the bad debt with the protocol's revenue instead. Alternatively, think about adding a withdrawal delay for lenders, requiring them to wait a period before claiming their withdrawal.

Status: Acknowledged

Update from client:

6.4 Best Practices

6.4.1 Incorrect documentation for function `pow_scale()`

File(s): /src/math.cairo

Description: The Natspec comment above the function incorrectly states that `is_negative` is true if `x` is negative. The actual implementation, however, considers `is_negative` to be true if `n` is negative, not `x`.

```
/// # Arguments
/// * `x` - base [SCALE]
/// * `n` - exponent [decimal]
/// * `is_negative` - true if `x` is negative
/// # Returns
/// * `result` - [SCALE]
fn pow_scale(mut x: u256, mut n: u256, is_negative: bool) -> u256 {
    if is_negative {
        x = SCALE * SCALE / x;
    }

    if n == 0 {
        return SCALE;
    }
    ...
}
```

Recommendation(s): Either correct the Natspec comment or adjust the implementation to align with the comment.

Status: Fixed

Update from client: Fixed in commit.

6.4.2 Potentially misleading function naming for `assert_config_exists`

File(s): /src/data_model.cairo

Description: In data_model.cairo the function `assert_config_exists` is declared which is used to indicate if an asset config exists. The function name implies that upon a non-existent asset config it will revert, however, it returns a boolean value indicating existence instead.

```
fn assert_config_exists(asset_config: AssetConfig) -> bool {
    asset_config.last_rate_accumulator != 0
}
```

Recommendation(s): Consider adjusting either the function logic to revert on a non-existent asset config, or adjust the function name to align closer with it's actual behavior.

Status: Fixed

Update from client: Fixed in commit.

6.4.3 Incorrect documentation for function `claim_fees`

File(s): /src/extension/components/fee_model.cairo

Description: The function `claim_fees` accepts two arguments, `pool_id`, and `collateral_asset`. The inline comment documentation for this function incorrectly states that it accepts a third argument `singleton`, which is not the case.

```
/// Claims the fees accrued in the extension for a given asset in a pool and sends them to the fee recipient
/// # Arguments
/// * `singleton` - The singleton contract address
/// * `pool_id` - id of the pool
/// * `collateral_asset` - address of the collateral asset
fn claim_fees(ref self: ComponentState<TContractState>, pool_id: felt252, collateral_asset: ContractAddress) {
    let singleton = self.get_contract().singleton();
    // ...
}
```

Recommendation(s): Consider adjusting the comment to accurately represent the function arguments.

Status: Fixed

Update from client: Fixed in commit.

6.4.4 Inconsistent LTV config validation between `create_pool` and `set_ltv_config`

File(s): /src/singleton.cairo

Description: When a pool is being created, the initial LTV configs are set within the `create_pool` function. As part of this function, a check is done to ensure that the collateral and debt asset cannot be the same for a given LTV configuration, as shown below:

```
while !ltv_params
    .is_empty() {
        let params = *ltv_params.pop_front().unwrap();
        assert!(params.collateral_asset_index != params.debt_asset_index, "identical-assets");
        let collateral_asset = *asset_params.at(params.collateral_asset_index).asset;
        let debt_asset = *asset_params.at(params.debt_asset_index).asset;
        self.set_ltv_config(pool_id, collateral_asset, debt_asset, LTVConfig { max_ltv: params.max_ltv });
    };
```

However when manually setting an LTV config after the pool has been created using the function `set_ltv_config` there is no such check, allowing a config to be added where the collateral and debt asset are the same address:

```
fn set_ltv_config(...) {
    assert!(!self.lock.read(), "set-ltv-config-reentrancy");
    assert!(get_caller_address() == self.extensions.read(pool_id), "caller-not-extension");
    assert_ltv_config(ltv_config);
    self.ltv_configs.write((pool_id, collateral_asset, debt_asset), ltv_config);
    //...
}

fn assert_ltv_config(ltv_config: LTVConfig) {
    assert!(ltv_config.max_ltv.into() <= SCALE, "invalid-ltv-config");
}
```

Recommendation(s): Consider moving the validation for the collateral and debt asset being the same into the `set_ltv_config` function so the validation is consistent between pool creation and manually setting configurations after pool creation.

Status: Fixed

Update from client: Fixed in commit.

7 Test Evaluation

7.1 Compilation Output

7.1.1 Identity

```
Run scarb build
    Updating git repository (github.com/keep-starknet-strange/alexandria)
    Updating git repository (github.com/foundry-rs/starknet-foundry)
   Compiling vesu v0.1.0
    Finished release target(s) in 15 seconds
```

7.2 Tests Output

```
scarb test
     Running test vesu (snforge test)
   Compiling vesu v0.1.0 (.../012-VESU/Scarb.toml)
    Finished release target(s) in 10 seconds
[WARNING] RPC node with the url (starknet-mainnet.public.blastapi.io/rpc/v0_6) uses incompatible version 0.6.0.
     Expected version: 0.7.0


Collected 220 test(s) from vesu package
Running 220 test(s) from src/
[PASS] vesu::test::test_common::TestCommon::test_deconstruct_native_debt_target (gas: ~36)
[PASS] vesu::test::test_common::TestCommon::test_calculate_fee_shares (gas: ~48)
[PASS] vesu::test::test_common::TestCommon::test_calculate_collateral_collateral_overflow (gas: ~13)
[PASS] vesu::test::test_common::TestCommon::test_calculate_nominal_debt_nominal_debt_overflow (gas: ~7)
[PASS] vesu::test::test_common::TestCommon::test_calculate_collateral_shares_collateral_shares_overflow (gas:
    ~13)
[PASS] vesu::test::test_common::TestCommon::test_deconstruct_asset_debt_target (gas: ~90)
[PASS] vesu::test::test_common::TestCommon::test_apply_position_update_to_context_positive_delta (gas: ~61)
[PASS] vesu::test::test_common::TestCommon::test_apply_position_update_to_context_negative_delta (gas: ~61)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_extension_set_shutdown_config (gas: ~8255)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_create_pool_empty_asset_params (gas: ~2783)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_create_pool_v_token_params_mismatch (gas:
    ~2786)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_create_pool_interest_rate_params_mismatch (
    gas: ~2785)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_create_pool_pragma_oracle_params_mismatch (
    gas: ~2786)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_add_asset_not_owner (gas: ~8677)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_create_pool (gas: ~8305)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_set_pool_owner (gas: ~8253)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_add_asset (gas: ~10178)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_extension_set_asset_parameter_not_owner (
    gas: ~8252)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_add_asset_pragma_key_must_be_set (gas:
    ~8677)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_set_pool_owner_not_owner (gas: ~8252)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_extension_set_asset_parameter (gas: ~8398)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_add_asset_oracle_config_already_set (gas:
    ~8256)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_set_ltv_config_caller_not_owner (gas:
    ~8253)
[PASS] vesu::test::test_asset_retrieval::TestAssetRetrieval::test_retrieve_from_reserve_fee_shares (gas: ~12466)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::
    test_extension_set_liquidation_config_caller_not_owner (gas: ~8253)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_extension_set_ltv_config (gas: ~8258)
[PASS] vesu::test::test_reentrancy::TestReentrancy::test_asset_config_reentrancy (gas: ~3614)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::
    test_extension_set_shutdown_config_caller_not_owner (gas: ~8252)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::
    test_extension_set_shutdown_config_invalid_shutdown_config (gas: ~8252)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_extension_set_liquidation_config (gas:
    ~8256)
```

```
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_extension_set_shutdown_ltv_config (gas: ~8255)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_extension_set_shutdown_ltv_config_caller_not_owner (gas: ~8253)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_extension_set_shutdown_ltv_config_invalid_ltv_config (gas: ~8252)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_set_extension (gas: ~8256)
[PASS] vesu::test::test_asset_retrieval::TestAssetRetrieval::test_retrieve_from_reserve_total_balance (gas: ~11488)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_extension_set_oracle_parameter (gas: ~8324)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_set_extension_not_owner (gas: ~8252)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_extension_set_oracle_parameter_invalid_oracle_parameter (gas: ~8252)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_extension_set_interest_rate_parameter_caller_not_owner (gas: ~8252)
[PASS] vesu::test::test_common::TestCommon::test_calculate_nominal_debt_zero_rate_accumulator (gas: ~2)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_extension_set_oracle_parameter_caller_not_owner (gas: ~8252)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_extension_set_oracle_parameter_oracle_config_not_set (gas: ~8252)
[PASS] vesu::test::timelock_test::TestTimelock::test_queue_executed_too_late (gas: ~478)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_extension_set_interest_rate_parameter_invalid_interest_rate_parameter (gas: ~8288)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_extension_set_interest_rate_parameter (gas: ~8618)
[PASS] vesu::test::test_interest_rate_model::TestInterestRateModel::test_model_from_asset_config (gas: ~107)
[PASS] vesu::test::test_interest_rate_model::TestInterestRateModel::test_utilization (gas: ~45)
[PASS] vesu::test::test_default_extension::TestDefaultExtension::test_extension_set_interest_rate_parameter_interest_rate_config_not_set (gas: ~8258)
[PASS] vesu::test::test_common::TestCommon::test_calculate_unsafe_rate_accumulator (gas: ~82)
[PASS] vesu::test::test_common::TestCommon::test_calculate_rate_accumulator (gas: ~13)
[PASS] vesu::test::test_flash_loan::FlashLoans::test_flash_loan_entire_pool (gas: ~11670)
[PASS] vesu::test::test_flash_loan::FlashLoans::test_flash_loan_fractional_pool_amount (gas: ~11672)
[PASS] vesu::test::test_flash_loan::FlashLoans::test_flash_loan_malicious_user (gas: ~11595)
[PASS] vesu::test::test_asset_retrieval::TestAssetRetrieval::test_retrieve_from_reserve_incorrect_caller (gas: ~11489)
[PASS] vesu::test::test_transfer_position::TestTransferPosition::test_transfer_position_collateral_dusty_collateral_balance_to (gas: ~12786)
[PASS] vesu::test::test_shutdown::TestShutdown::test_recovery_mode_made_safer (gas: ~12180)
[PASS] vesu::test::test_shutdown::TestShutdown::test_recovery_mode_decreasing_collateral (gas: ~12553)
[PASS] vesu::test::test_shutdown::TestShutdown::test_recovery_mode_increasing_debt (gas: ~12553)
[PASS] vesu::test::test_shutdown::TestShutdown::test_subscription_mode_decreasing_debt (gas: ~14203)
[PASS] vesu::test::test_shutdown::TestShutdown::test_subscription_mode_decreasing_collateral (gas: ~13203)
[PASS] vesu::test::test_shutdown::TestShutdown::test_subscription_mode_increasing_collateral (gas: ~14372)
[PASS] vesu::test::test_shutdown::TestShutdown::test_redemption_mode_decreasing_collateral (gas: ~14540)
[PASS] vesu::test::test_shutdown::TestShutdown::test_subscription_mode_increasing_debt (gas: ~13203)
[PASS] vesu::test::test_shutdown::TestShutdown::test_redemption_mode_increasing_debt (gas: ~14331)
[PASS] vesu::test::test_shutdown::TestShutdown::test_redemption_mode_increasing_collateral (gas: ~16422)
[PASS] vesu::test::test_shutdown::TestShutdown::test_redemption_mode_decreasing_debt (gas: ~14332)
[PASS] vesu::test::test_common::TestCommon::test_deconstruct_collateral_target_collateral_target_negative (gas: ~23)
[PASS] vesu::test::test_shutdown::TestShutdown::test_redemption_mode_non_zero_debt (gas: ~13747)
[PASS] vesu::test::test_shutdown::TestShutdown::test_redemption_mode_max_utilization (gas: ~15261)
[PASS] vesu::test::test_asset_retrieval::TestAssetRetrieval::test_retrieve_from_reserve_partial_balance (gas: ~11577)
[PASS] vesu::test::test_shutdown::TestShutdown::test_recovery_mode_complex_oldest_timestamp (gas: ~13195)
[PASS] vesu::test::test_shutdown::TestShutdown::test_recovery_mode_complex (gas: ~14597)
[PASS] vesu::test::test_interest_rate_model::TestInterestRateModel::test_next_full_utilization_vs_last (runs: 256, gas: {max: ~42, min: ~42, mean: ~42.00, std deviation: ~0.00})
[PASS] vesu::test::test_common::TestCommon::test_calculate_collateral_and_debt_value (runs: 256, gas: {max: ~61, min: ~61, mean: ~61.00, std deviation: ~0.00})
[PASS] vesu::test::test_interest_rate_model::TestInterestRateModel::test_full_utilization_bounds (runs: 256, gas: {max: ~32, min: ~27, mean: ~30.00, std deviation: ~2.31})
[PASS] vesu::test::test_forking::TestForking::test_forking (gas: ~29311)
[PASS] vesu::test::test_common::TestCommon::test_calculate_nominal_debt_precision (gas: ~37)
[PASS] vesu::test::test_common::TestCommon::test_calculate_utilization_div_zero (gas: ~7)
[PASS] vesu::test::test_common::TestCommon::test_calculate_collateral_and_debt_value_2 (runs: 256, gas: {max: ~47, min: ~47, mean: ~47.00, std deviation: ~0.00})
[PASS] vesu::test::test_common::TestCommon::test_deconstruct_collateral_shares_asset_delta (gas: ~58)
[PASS] vesu::test::test_common::TestCommon::test_deconstruct_native_collateral_delta (gas: ~46)
```

```
[PASS] vesu::test::test_shutdown::TestShutdown::test_recovery_mode_transfer_different_pair (gas: ~12369)
[PASS] vesu::test::test_shutdown::TestShutdown::test_recovery_mode_from_none (gas: ~12308)
[PASS] vesu::test::test_shutdown::TestShutdown::test_recovery_mode_transfer_within_pair (gas: ~12681)
[PASS] vesu::test::test_shutdown::TestShutdown::test_unsafe_rate_accumulator (gas: ~13954)
[PASS] vesu::test::test_shutdown::TestShutdown::test_shutdown_collateral_accounting (gas: ~13515)
[PASS] vesu::test::test_interest_rate_model::TestInterestRateModel::test_next_full_utilization_vs_half_life (gas: ~29)
[PASS] vesu::test::test_interest_rate_model::TestInterestRateModel::test_interest_rate_config_packing (gas: ~8)
[PASS] vesu::test::test_interest_rate_model::TestInterestRateModel::test_full_utilization_extremes (runs: 256, gas: {max: ~43, min: ~43, mean: ~43.00, std deviation: ~0.00})
[PASS] vesu::test::test_liquidate_position::TestLiquidatePosition::test_before_liquidate_position_caller_not_singleton (gas: ~11118)
[PASS] vesu::test::test_interest_rate_model::TestInterestRateModel::test_set_interest_rate_model_fee_shares (gas: ~12469)
[PASS] vesu::test::test_liquidate_position::TestLiquidatePosition::test_after_liquidate_position_caller_not_singleton (gas: ~11118)
[PASS] vesu::test::test_liquidate_position::TestLiquidatePosition::test_liquidate_position_partial_no_bad_debt (gas: ~12862)
[PASS] vesu::test::test_liquidate_position::TestLiquidatePosition::test_liquidate_position_invalid_oracle_1 (gas: ~12375)
[PASS] vesu::test::test_liquidate_position::TestLiquidatePosition::test_liquidate_position_invalid_oracle_2 (gas: ~12375)
[PASS] vesu::test::test_liquidate_position::TestLiquidatePosition::test_liquidate_position_partial_bad_debt_2 (gas: ~12841)
[PASS] vesu::test::test_liquidate_position::TestLiquidatePosition::test_liquidate_position_not_undercollateralized (gas: ~12348)
[PASS] vesu::test::test_liquidate_position::TestLiquidatePosition::test_liquidate_position_partial_insufficient_collateral_released (gas: ~12420)
[PASS] vesu::test::test_liquidate_position::TestLiquidatePosition::test_liquidate_position_partial_no_bad_debt_in_shares (gas: ~12832)
[PASS] vesu::test::test_liquidate_position::TestLiquidatePosition::test_liquidate_position_invalid_liquidation_data (gas: ~12310)
[PASS] vesu::test::test_liquidate_position::TestLiquidatePosition::test_liquidate_position_partial_bad_debt (gas: ~12511)
[PASS] vesu::test::test_map_list::TestMapList::test_list_creation (gas: ~376)
[PASS] vesu::test::test_map_list::TestMapList::test_insert_zero (gas: ~370)
[PASS] vesu::test::test_map_list::TestMapList::test_remove_nonexistent (gas: ~372)
[PASS] vesu::test::test_map_list::TestMapList::test_remove (gas: ~312)
[PASS] vesu::test::test_map_list::TestMapList::test_remove_zero (gas: ~370)
[PASS] vesu::test::test_math::TestMath::test_pow_scale (gas: ~16)
[PASS] vesu::test::test_liquidate_position::TestLiquidatePosition::test_liquidate_position_scenario_1_full_liquidation (gas: ~12370)
[PASS] vesu::test::test_math::TestMath::test_pow (gas: ~5)
[PASS] vesu::test::test_liquidate_position::TestLiquidatePosition::test_liquidate_position_full_bad_debt (gas: ~12515)
[PASS] vesu::test::test_liquidate_position::TestLiquidatePosition::test_liquidate_position_scenario_2_full_liquidation (gas: ~12363)
[PASS] vesu::test::test_liquidate_position::TestLiquidatePosition::test_liquidate_position_full_no_bad_debt (gas: ~12511)
[PASS] vesu::test::test_math::TestMath::test_pow_scale_fuzz (runs: 22, gas: {max: ~9, min: ~9, mean: ~9.00, std deviation: ~0.00})
[PASS] vesu::test::test_liquidate_position::TestLiquidatePosition::test_liquidate_position_scenario_3_full_liquidation (gas: ~12365)
[PASS] vesu::test::test_common::TestCommon::test_deconstruct_native_collateral_target (gas: ~73)
[PASS] vesu::test::test_liquidate_position::TestLiquidatePosition::test_liquidate_position_scenario_5_partial_liquidation (gas: ~12697)
[PASS] vesu::test::test_liquidate_position::TestLiquidatePosition::test_liquidate_position_scenario_4_full_liquidation (gas: ~12365)
[PASS] vesu::test::test_transfer_position::TestTransferPosition::test_transfer_position_debt_asset_mismatch (gas: ~12389)
[PASS] vesu::test::test_transfer_position::TestTransferPosition::test_transfer_position_collateral_target (gas: ~11926)
[PASS] vesu::test::test_transfer_position::TestTransferPosition::test_transfer_position_debt_no_delegate (gas: ~12734)
[PASS] vesu::test::test_transfer_position::TestTransferPosition::test_transfer_position_debt (gas: ~12929)
[PASS] vesu::test::test_singleton::TestSingleton::test_create_pool_assert_ltv_config_invalid_ltv_config (gas: ~4067)
[PASS] vesu::test::test_singleton::TestSingleton::test_pool_id (gas: ~2915)
[PASS] vesu::test::test_singleton::TestSingleton::test_create_pool_assert_asset_config_scale_exceeded (gas: ~3352)
```

```
[PASS] vesu::test::test_singleton::TestSingleton::test_create_pool_no_extension (gas: ~2847)
[PASS] vesu::test::test_singleton::TestSingleton::test_set_asset_parameter_not_extension (gas: ~8252)
[PASS] vesu::test::test_singleton::TestSingleton::test_create_pool_assert_asset_config_rate_accumulator_too_low (gas: ~2924)
[PASS] vesu::test::test_singleton::TestSingleton::test_create_pool_duplicate_asset (gas: ~3089)
[PASS] vesu::test::test_singleton::TestSingleton::test_create_pool_assert_asset_config_max_utilization_exceeded (gas: ~2924)
[PASS] vesu::test::test_singleton::TestSingleton::test_create_pool_assert_asset_config_fee_rate_exceeded (gas: ~2924)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_calculate_withdrawable_assets (gas: ~37)
[PASS] vesu::test::test_singleton::TestSingleton::test_set_asset_config_not_extension (gas: ~8675)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_mint_v_token (gas: ~1590)
[PASS] vesu::test::test_singleton::TestSingleton::test_set_asset_config (gas: ~8867)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_mint_v_token_not_extension (gas: ~1453)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_burn_v_token (gas: ~1611)
[PASS] vesu::test::test_math::TestMath::test_pow_10 (gas: ~2)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_burn_v_token_not_extension (gas: ~1661)
[PASS] vesu::test::test_transfer_position::TestTransferPosition::
    test_transfer_position_debt_dusty_debt_balance_from (gas: ~12570)
[PASS] vesu::test::test_transfer_position::TestTransferPosition::test_transfer_position_debt_target (gas: ~12787)
[PASS] vesu::test::test_modify_position::TestModifyPosition::test_modify_position_zero_asset (gas: ~11361)
[PASS] vesu::test::test_transfer_position::TestTransferPosition::
    test_transfer_position_debt_from_undercollateralized (gas: ~12605)
[PASS] vesu::test::test_transfer_position::TestTransferPosition::test_transfer_position_collateral_v_token (gas: ~12421)
[PASS] vesu::test::test_transfer_position::TestTransferPosition::test_transfer_position_zero_debt_asset (gas: ~12403)
[PASS] vesu::test::test_transfer_position::TestTransferPosition::
    test_transfer_position_collateral_v_token_from_zero_debt_asset (gas: ~12096)
[PASS] vesu::test::test_transfer_position::TestTransferPosition::test_transfer_position_fee_shares (gas: ~13698)
[PASS] vesu::test::test_transfer_position::TestTransferPosition::
    test_transfer_position_debt_dusty_debt_balance_to (gas: ~12775)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_total_assets (gas: ~11606)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_convert_to_shares (gas: ~11183)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_asset (gas: ~11114)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_convert_to_assets (gas: ~11183)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_deposit (gas: ~11535)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_max_deposit (gas: ~11240)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_preview_mint (gas: ~11249)
[PASS] vesu::test::timelock_test::TestTimelock::test_deploy (gas: ~233)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_preview_deposit (gas: ~11249)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_max_withdraw (gas: ~11782)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_mint (gas: ~11574)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_max_mint (gas: ~11204)
[PASS] vesu::test::timelock_test::TestTimelock::test_queue_cancel (gas: ~479)
[PASS] vesu::test::timelock_test::TestTimelock::test_queue_execute (gas: ~552)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_preview_withdraw (gas: ~11249)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_withdraw (gas: ~11559)
[PASS] vesu::test::timelock_test::TestTimelock::test_queue_executed_too_early (gas: ~478)
[PASS] vesu::test::timelock_test::TestTimelock::test_queue_execute_twice (gas: ~550)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_withdraw_caller_not_approved (gas: ~11785)

Success data:
    0x753235365f737562204f766572666c6f77 ('u256_sub Overflow')

[PASS] vesu::test::test_v_token::TestVToken::test_v_token_preview_redeem (gas: ~11249)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_redeem_caller_not_approved (gas: ~11642)

Success data:
    0x753235365f737562204f766572666c6f77 ('u256_sub Overflow')

[PASS] vesu::test::test_v_token::TestVToken::test_v_token_max_redeem (gas: ~11782)
[PASS] vesu::test::test_v_token::TestVToken::test_v_token_redeem (gas: ~11598)
[PASS] vesu::test::test_singleton::TestSingleton::test_set_asset_parameter (gas: ~8409)
[PASS] vesu::test::test_modify_position::TestModifyPosition::test_before_modify_position_caller_not_singleton (gas: ~11118)
[PASS] vesu::test::test_modify_position::TestModifyPosition::test_modify_position_unknown_pool (gas: ~11178)
[PASS] vesu::test::test_modify_position::TestModifyPosition::test_after_modify_position_caller_not_singleton (gas: ~11118)
```

```
[PASS] vesu::test::test_modify_position::TestModifyPosition::test_modify_position_utilization_exceeded (gas: ~11769)
[PASS] vesu::test::test_modify_position::TestModifyPosition::test_modify_position_dusty_debt_balance (gas: ~11746)
[PASS] vesu::test::test_pragma_oracle::TestPragmaOracle::test_is_valid_timeout (gas: ~6650)
[PASS] vesu::test::test_singleton::TestSingleton::test_set_ltv_config (gas: ~8254)
[PASS] vesu::test::test_packing::TestPacking::test_asset_config_packing (gas: ~38)
[PASS] vesu::test::test_singleton::TestSingleton::test_set_extension_not_extension (gas: ~8252)
[PASS] vesu::test::test_modify_position::TestModifyPosition::test_modify_position_no_delegation (gas: ~11583)
[PASS] vesu::test::test_transfer_position::TestTransferPosition::test_transfer_position_same_position (gas: ~11488)
[PASS] vesu::test::test_modify_position::TestModifyPosition::test_modify_position_not_collateralized (gas: ~12061)
[PASS] vesu::test::test_modify_position::TestModifyPosition::test_modify_position_dusty_collateral_balance (gas: ~11795)
[PASS] vesu::test::test_reentrancy::TestReentrancy::test_context_reentrancy (gas: ~3614)
[PASS] vesu::test::test_pragma_oracle::TestPragmaOracle::test_is_valid_timeout_stale_price (gas: ~6644)
[PASS] vesu::test::test_singleton::TestSingleton::test_set_asset_parameter_fee_shares (gas: ~12516)
[PASS] vesu::test::test_common::TestCommon::test_deconstruct_asset_debt_delta (gas: ~35)
[PASS] vesu::test::test_pragma_oracle::TestPragmaOracle::test_is_valid_sources_not_reached (gas: ~6516)
[PASS] vesu::test::test_pragma_oracle::TestPragmaOracle::test_is_valid_sources_reached (gas: ~6523)
[PASS] vesu::test::test_modify_position::TestModifyPosition::test_modify_position_zero_asset_borrow (gas: ~11359)
[PASS] vesu::test::test_pool_donations::TestPoolDonation::test_donate_to_reserve_pool_wrong_pool_id (gas: ~11181)
[PASS] vesu::test::test_modify_position::TestModifyPosition::test_modify_position_no_pair (gas: ~11602)
[PASS] vesu::test::test_pragma_oracle::TestPragmaOracle::test_get_default_price (gas: ~11122)
[PASS] vesu::test::test_pragma_oracle::TestPragmaOracle::test_get_price_high (gas: ~11132)
[PASS] vesu::test::test_pool_donations::TestPoolDonation::test_donate_to_reserve_pool_incorrect_asset (gas: ~11638)
[PASS] vesu::test::test_pool_donations::TestPoolDonation::test_donate_to_reserve_pool (gas: ~12774)
[PASS] vesu::test::test_modify_position::TestModifyPosition::test_modify_position_debt_amounts (gas: ~14270)
[PASS] vesu::test::test_common::TestCommon::test_deconstruct_debt_target_debt_target_negative (gas: ~24)
[PASS] vesu::test::test_transfer_position::TestTransferPosition::test_transfer_position_collateral_asset_mismatch (gas: ~11592)
[PASS] vesu::test::test_shutdown::TestShutdown::test_recovery_mode_transfer_non_zero_debt (gas: ~12491)
[PASS] vesu::test::test_transfer_position::TestTransferPosition::test_transfer_position_collateral (gas: ~11926)
[PASS] vesu::test::test_transfer_position::TestTransferPosition::
    test_transfer_position_collateral_target_increase_from (gas: ~11717)
[PASS] vesu::test::test_transfer_position::TestTransferPosition::test_transfer_position_debt_target_increase_from (gas: ~12576)
[PASS] vesu::test::test_transfer_position::TestTransferPosition::
    test_transfer_position_collateral_dusty_collateral_balance_from (gas: ~12592)
[PASS] vesu::test::test_modify_position::TestModifyPosition::test_modify_position_fees (gas: ~13069)
[PASS] vesu::test::test_modify_position::TestModifyPosition::test_modify_position_complex (gas: ~13600)
[PASS] vesu::test::test_common::TestCommon::test_deconstruct_asset_collateral_target (gas: ~119)
[PASS] vesu::test::test_modify_position::TestModifyPosition::test_modify_position_accrue_interest (gas: ~13715)
[PASS] vesu::test::test_modify_position::TestModifyPosition::test_modify_position_collateral_amounts (gas: ~13509)
[PASS] vesu::test::test_common::TestCommon::test_deconstruct_native_debt_delta (gas: ~35)
[PASS] vesu::test::test_common::TestCommon::test_calculate_nominal_debt_rounding (runs: 256, gas: {max: ~15, min: ~14, mean: ~14.00, std deviation: ~0.95})
[PASS] vesu::test::test_common::TestCommon::test_calculate_utilization (runs: 256, gas: {max: ~7, min: ~7, mean: ~7.00, std deviation: ~0.00})
[PASS] vesu::test::test_common::TestCommon::test_collateral_inverse_relation (runs: 256, gas: {max: ~27, min: ~27, mean: ~27.00, std deviation: ~0.00})
[PASS] vesu::test::test_common::TestCommon::test_calculate_nominal_calculate_debt_inverse_relation (runs: 256, gas: {max: ~14, min: ~14, mean: ~14.00, std deviation: ~0.00})
[PASS] vesu::test::test_math::TestMath::test_log_10 (gas: ~12)
[PASS] vesu::test::test_common::TestCommon::test_calculate_nominial_debt_calculate_debt_inverse_relation (runs: 256, gas: {max: ~14, min: ~14, mean: ~14.00, std deviation: ~0.00})
[PASS] vesu::test::test_common::TestCommon::test_is_collateralized (runs: 256, gas: {max: ~8, min: ~8, mean: ~8.00, std deviation: ~0.00})
[PASS] vesu::test::test_modify_position::TestModifyPosition::test_fuzz_modify_position_borrow_repay_debt (runs: 256, gas: {max: ~14105, min: ~13324, mean: ~14097.00, std deviation: ~48.79})
[PASS] vesu::test::test_modify_position::TestModifyPosition::
    test_fuzz_modify_position_deposit_withdraw_collateral (runs: 256, gas: {max: ~13371, min: ~13023, mean: ~13365.00, std deviation: ~22.20})
Tests: 220 passed, 0 failed, 0 skipped, 0 ignored, 0 filtered out
Fuzzer seed: 10386720301197904590
```