Security Review Report
NM-0123 Ekubo

NETHERMIND
SECURITY

(Mar 26, 2024)

Contents

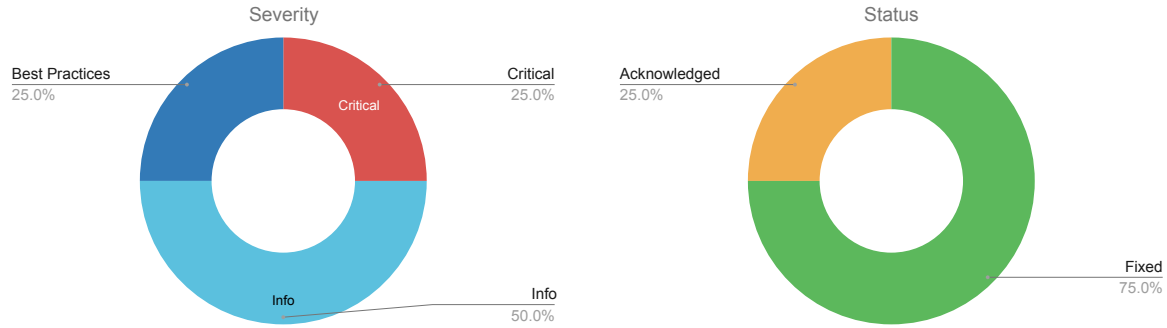
1 Executive Summary

This document outlines the security review conducted by Nethermind Security for Ekubo. `Ekubo` is a next-generation AMM (Automated Market Maker) built on Starknet. The protocol features concentrated liquidity, a singleton architecture, and extensions, all designed to leverage the Starknet architecture fully. Its aim is to deliver unparalleled execution for swappers and returns for liquidity providers. Ekubo enables users to achieve significant capital efficiency and perform swap and position management operations through a combination of architectural and implementation decisions aimed at improving gas efficiency.

The team conducted a fifteen-engineer-week review of 4044 lines of code. The `Ekubo` team has provided documentation explaining the main features of the protocol. Aside from the provided documentation, the `Ekubo` and `Nethermind Security` engaged in multiple calls to clarify any remaining questions about the expected behavior of the protocol.

The audit was performed using: (a) manual analysis of the codebase, and (b) simulation of the smart contracts. **Along this document, we report** four points of attention, where one is classified as `Critical` and three is classified as `Informational` or `Best Practice`. The issues are summarized in Fig. 1.

This document is organized as follows. Section 2 presents the files in the scope of this audit. Section 3 summarizes the issues. Section 4 presents the system overview. Section 5 discusses the risk rating methodology adopted for this audit. Section 6 details the issues. Section 7 describes complementary validations done during the assessment. Section 8 discusses the documentation provided by the client for this audit. Section 9 presents the compilation, tests, and automated tests. Section 10 concludes the document.



(a) distribution of issues according to the severity

(b) distribution of issues according to the status

Fig 1: (a) Distribution of issues: Critical (1), **High** (0), **Medium** (0), **Low** (0), **Undetermined** (0), **Informational** (2), **Best Practices** (1). **(b) Distribution of status: Fixed** (3), **Acknowledged** (1), **Mitigated** (0), **Unresolved** (0)

Summary of the Audit

Audit Type	Security Review
Initial Report	Mar 1, 2024
Final Report	Mar 22, 2024
Methods	Manual Review, Tests
Repository	EkuboProtocol
Commit Hash	e1b458753e48935e8c463578d1217830b4d74b4b
Final Commit Hash	9eda3c648fefeca28ffedd1b194267dc60c2d617
Documentation	Official Documentation
Documentation Assessment	High
Test Suite Assessment	High

2 Audited Files

	Contract	LoC	Comments	Ratio	Blank	Total
1	src/core.cairo	834	65	7.8%	170	1069
2	src/router.cairo	379	40	10.6%	69	488
3	src/lib.cairo	85	22	25.9%	8	115
4	src/positions.cairo	390	26	6.7%	64	480
5	src/owned_nft.cairo	255	29	11.4%	51	335
6	src/types/position.cairo	31	13	41.9%	7	51
7	src/types/delta.cairo	37	9	24.3%	4	50
8	src/types/fees_per_liquidity.cairo	45	10	22.2%	8	63
9	src/types/pool_price.cairo	58	8	13.8%	12	78
10	src/types/keys.cairo	35	17	48.6%	5	57
11	src/types/bounds.cairo	26	5	19.2%	3	34
12	src/types/call_points.cairo	99	6	6.1%	11	116
13	src/types/i129.cairo	194	24	12.4%	38	256
14	src/math/delta.cairo	49	7	14.3%	12	68
15	src/math/bitmap.cairo	121	19	15.7%	17	157
16	src/math/bits.cairo	37	3	8.1%	5	45
17	src/math/mask.cairo	135	1	0.7%	0	136
18	src/math/ticks.cairo	319	52	16.3%	51	422
19	src/math/swap.cairo	110	26	23.6%	20	156
20	src/math/string.cairo	38	3	7.9%	9	50
21	src/math/sqrt_ratio.cairo	72	10	13.9%	13	95
22	src/math/max_liquidity.cairo	46	8	17.4%	7	61
23	src/math/exp2.cairo	135	1	0.7%	0	136
24	src/math/muldiv.cairo	42	6	14.3%	6	54
25	src/math/liquidity.cairo	44	4	9.1%	3	51
26	src/math/fee.cairo	28	4	14.3%	5	37
27	src/components/shared_locker.cairo	37	0	0.0%	5	42
28	src/components/util.cairo	5	0	0.0%	0	5
29	src/components/upgradeable.cairo	43	8	18.6%	12	63
30	src/components/owned.cairo	53	11	20.8%	12	76
31	src/components/clear.cairo	28	8	28.6%	4	40
32	src/interfaces/core.cairo	110	67	60.9%	34	211
33	src/interfaces/upgradeable.cairo	4	2	50.0%	2	8
34	src/interfaces/erc721.cairo	24	2	8.3%	2	28
35	src/interfaces/erc20.cairo	10	10	100.0%	4	24
36	src/interfaces/positions.cairo	74	24	32.4%	19	117
37	src/interfaces/src5.cairo	12	9	75.0%	4	25
	Total	**4044**	**559**	**13.8%**	**696**	**5299**

During the engagement, the **Nethermind Security** team sought to answer the following non-exhaustive list of questions:

- Can pools become insolvent?
- Can funds get stuck in the pool?
- Are rounding operations applied correctly??
- Could the overflow of the *pool_fees* variable cause liquidity providers to claim more fees than they have earned?
- Does the following property hold *sqrt_ratio_to_tick(tick_to_sqrt_ratio(tick)) == tick*?
- Does the following property hold *sqrt_ratio_at_tick(tick_to_sqrt_ratio(tick) - 1) == tick - 1*?
- Does the following property hold *sqrt_ratio_at_tick(tick_to_sqrt_ratio(tick) + 1) == tick*?
- Is there appropriate access control for privileged operations?
- Is it possible to drain funds from a pool?
- Are there any re-entrancy vulnerabilities in the contract?
- Are inputs validated?
- Do documented interactions with the contract lead to unexpected behavior?

3 Summary of Issues

	Finding	Severity	Update
1	Re-entrancy through extension calls allows the use of stale state	Critical	Fixed
2	Edge case bypass slippage protection	Info	Fixed
3	Incomplete validation of `token` argument in swaps	Info	Acknowledged
4	Unused arguments	Best Practices	Fixed

4 System Overview

4.1 Main contracts

Ekubo is a next-generation AMM (Automated Market Maker) built on Starknet. It features concentrated liquidity, extensions, Non-fungible-tokens (NFTs) as representation for positions, and multiple fee options. All these features, combined with an architecture and implementation created with gas efficiency in mind, make Ekubo an outstanding product for users. Three contracts are integral to the protocol: **a) Core** containing the main functionalities of the protocol, **b) Router** making swaps simpler for users, and **c) Positions** to manage liquidity positions.

4.1.1 Core

At the heart of Ekubo, we find the *Core* contract. This is the main contract of the protocol, which holds all the protocol's liquidity and pools' information. The *Core* contract enables users to execute multiple swap and liquidity management operations in the same transaction and defer tokens payment to the end of the transaction, providing a cheaper execution, this behavior is referenced as the *"Till"* pattern.

Every interaction that could trigger a transfer of tokens must start with a call to the contract's function `lock(...)`. This function will start a context, which we will refer to as *locked context*, for the caller to execute one or multiple operations.

Main operations from the contract are executed through the following functions:

- **withdraw_protocol_fees(...)**: This function is only callable by the *owner* of the contract. It allows the withdrawal of any protocol fees previously collected.
- **lock(...)**: This function must be called before starting interactions with the protocol that could cause an exchange of tokens. It creates a *locked context* for the caller where multiple swaps and liquidity management operations can be done. The passed data is sent back to the caller through the *locked(...)* hook.
- **withdraw(...)**[1]: The function is used by the caller to withdraw a given token from the *Core* contract, it will generate a positive delta for the current *locked context* that must be fulfilled before finishing the operations.
- **pay(...)**[1]: The function pays a given token into *Core* and generates a negative delta for the current *locked context* that must be fulfilled before finishing the operations. The function uses a pull approach for receiving the tokens and will pull the full allowance given by the user.
- **save(...)**[1]: This function allows users to store tokens in the *Core* contract that can be used for later operations. It generates a positive delta for the current *locked context* that must be fulfilled before finishing the operations.
- **load(...)**[1]: The function recalls a balance previously saved via the *save(...)* function. It will generate a negative delta that must be fulfilled before finishing the operations.
- **initialize_pool(...)**: The function initialize a pool. The pool is specified by the tokens being swapped, the charged fee, the tick spacing and the extension contract. This function will check that the pool was not initialized before and set the initial price.
- **maybe_initialize_pool(...)**: This function executes the same operations than *initialize_pool(...)*, but it does not revert if the pool was already initialized.
- **update_position(...)**[1]: This function updates a liquidity position in a pool. It will increase or decrease the amount of liquidity assigned to a specific position. When the liquidity is increased, it will generate positive deltas to account for the tokens needed to provide that amount of liquidity; if the liquidity is being decreased it will generate negative deltas. These deltas are created for the current *locked context* and must be fulfilled before finishing the operations. The function will also update the amount of fees per liquidity earn by the position.
- **collect_fees(...)**[1]: The function collects the fees owed to a position and updates the position state. It will generate a negative delta for the current *locked context* that must be fulfilled before finishing the operations.
- **swap(...)**[1]: This function allows users to exchange tokens within a specific pool. The user will specify the amount of one of the tokens being exchanged, this amount can be positive or negative to indicate if the user is willing to give or take that amount of tokens. Users also specify what is the limit price that they accept for the swap. The function will generate deltas for the current *locked context* that must be fulfilled before finishing the operations, these deltas will be the amounts being exchanged.
- **accumulate_as_fees(...)**[1]: This function can only be called in a context created for the extension contract of the pool. It specifies a number of tokens that will be taken as fees for the currently active position. It will generate a positive delta for the current *locked context* that must be fulfilled before finishing the operations.

The contract also implements multiple functions with the main goal of fetching some relevant information about pools, positions, and balances. To add *AccessControl* and *Upgradeabilty* functionalities, the contract uses the *Owned* and *Upgradeable* components.

The *Core* contract fully implements the main functionalities of the protocol. However, it was not designed to be used directly by regular users. The *Router* and *Positions* contracts are periphery contracts that provide a simpler user experience to interact with the protocol.

[1] This function must be called within a *locked context*.

4.1.2 Router

The *Router* contract offers a straightforward interface for user interaction. It abstracts Ekubo's swap execution mechanisms and offers easy-to-use functions for complex operations.

The swaps are executed using tokens held by the router contract, this requires users to send tokens to the contract before calling the swap functions. The resulting tokens from the swap are also received by this contract and can be claimed through the functionalities provided by the *Clear* component, which also offers slippage protection mechanisms.

As mentioned, the main goal of this contract is to provide swap functionalities, and this is done through the following functions:

- **swap(...)**: This function is used to execute the most simple kind of swap. It allows the execution of one swap using only one hop.
- **multihop_swap(...)**: The function executes single swaps through one or multiple hops. Each hop's input(output) is passed as the output(input) of the next hop.
- **multi_multihop_swap(...)**: This is the most general function, the previous can be seen as simpler cases of this one. The function allows users to execute multiple multi-hop swaps.

Beyond the previously listed functions, the contract also offers functionalities for fetching information about the state of pools and simulating swaps.

The *Router* contract is the contract expected to be used by users needed to exchange tokens. For users interested in providing liquidity, the *Positions* contract offers multiple features.

4.2 Features

This subsection describes some of the main features of the Ekubo AMM. It gives an overview of Concentrated Liquidity, Extensions, the Till pattern, and the fees accrued by Ekubo.

4.2.1 Concentrated Liquidity

Concentrated liquidity allows market makers to provide liquidity within a specified price range. Each liquidity provider chooses the exact parameters of their position, though all positions in a pool are aggregated from a swapper's perspective. This is the main feature that allows Ekubo to offer better pricing than other AMMs. Unlike simple constant product AMMs, where deposited capital can be used to trade at any price ratio (P) of x to y, concentrated liquidity AMMs enable specifying the price range for trades. This significantly enhances capital efficiency for liquidity providers.

To specify price ranges, users select two ticks. Ticks are points in the price curve that can be selected as price range bounds. A tick is an integer number within the range $[-88722883, 88722883]$. The tick i represents the price 1.000001^i. Ticks can be seen as a discretization of the price curve, because of the base used for computing the price related to a tick, Ekubo allows users to specify ranges with a precision of *1/100th* of a basis point. However, this precision may vary between different pools due to the tick spacing. The set of ticks that can be selected to specify the price range of a position within a pool consists of all possible ticks that are multiples of the pool's tick spacing.

Because users can provide liquidity in specific ranges, the amount of active liquidity varies between price ranges. In order to compute the swap amounts, the following functions are used:

$$\Delta\sqrt{P} = \frac{\Delta y}{L} \tag{1}$$

$$\Delta\frac{1}{\sqrt{P}} = \Delta x * L \tag{2}$$

where P is the price, x the amount of *token0* and y the amount of *token1*, and L the amount of active liquidity.

When one of the token amounts to be exchanged is known, we can compute the new price of the pool using 1 or 2, depending on the known token amount. From the new price, we can compute what is the other token amount such that we reach that price using the equation not used before.

It is important to know that the amount of active liquidity may change between different price ranges, because of this, every time an active tick is crossed, the amount of active liquidity must be updated.

4.2.2 Extensions

Extensions enable users to insert custom logic at certain pool lifecycle events. These pools can implement new features such as oracles, limit orders, or TWAMM orders. Specified as part of the pool key, extensions form an immutable aspect of the pool's configuration. Upon pool initialization, if an extension contract exists, it is invoked prior to the initialization process. It then specifies the subsequent pool lifecycle events at which it should be called. It's important to note that the set of events triggering calls to the extension contract remains constant, this must be considered during pool initialization.

The full list of events that may trigger a call to the extension contract are:

- Before pool initialization.
- After pool initialization.
- Before a position update.
- After a position update.
- Before a swap.
- After a swap.

Users must consider extensions carefully when choosing pools to interact with, as they play a critical role in the pool lifecycle. Implementation errors in extensions could cause users to lose funds.

4.2.3 Till Pattern

Ekubo does all token balance accounting internally before transferring any token. This means users can trade with many pools, create and update as many positions as desired, and only transfer deltas at the end. Deltas are the changes in token balances that are accrued, from the pool perspective, from the operations. This makes the protocol incredibly efficient for tasks like arbitrage or creating many positions. This is possible because Ekubo is a singleton AMM that utilizes the **Till Pattern**.

In this pattern:

- The caller calls the contract and then receives a callback with a new ID.
- Within the callback, the caller can make any number of changes. In the context of Ekubo, these changes are mainly swaps and position manager actions.
- At the end of the till, the protocol will ensure that non-zero deltas do not exist. Deltas are changed through the actions in the *Core* contract.

The *Core* contract implements the *lock(...)* function that must be called for initiating the subsequent actions. It is important to note that in this pattern, re-entrancy is allowed. If *lock(...)* is called inside a *locked context*, a new ID will be created to reference the newly created context. All the generated deltas are specific to the context where they were generated.

5 Risk Rating Methodology

The risk rating methodology used by Nethermind follows the principles established by the OWASP Foundation. 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.

		Severity Risk		
Impact	**High**	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Info/Best Practices	Low	Medium
	Undetermined	Undetermined	Undetermined	Undetermined
		Low	**Medium**	**High**
		Likelihood		

To address issues that do not fit a High/Medium/Low severity, Nethermind also uses three more finding severities: **Informational**, **Best Practices**, and **Undetermined**.

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) **Undetermined** findings are used when we cannot predict the impact or likelihood of the issue.

6 Issues

6.1 [Critical] Re-entrancy through extension calls allows the use of stale state

File(s): `core.cairo`

Description: Extensions are a feature of Ekubo that enable users to insert custom logic at certain pool lifecycle events. Users can trigger the custom logic in the following events:

- Before pool initialization;
- After pool initialization;
- Before a position update;
- Before a swap;
- After a swap;

Those hooks are allowed to re-enter the `Core` contract, being able to execute swaps, manage positions, and execute any other action in the `Core` contract.

A problem arises because information fetched before calling the 'Before ...' hooks is used for operations done when the hook returns. Let's review the "Before pool initialization" case as an example.

```
fn initialize_pool(ref self: ContractState, pool_key: PoolKey, initial_tick: i129) -> u256 {
    pool_key.check_valid();

    // @audit - Pool is checked to not be initialized
    let price = self.pool_price.read(pool_key);
    assert(price.sqrt_ratio.is_zero(), 'ALREADY_INITIALIZED');

    // @audit - Custom logic executed through the extension
    let call_points = if (pool_key.extension.is_non_zero()) {
        IExtensionDispatcher { contract_address: pool_key.extension }
            .before_initialize_pool(get_caller_address(), pool_key, initial_tick)
    } else {
        Default::<CallPoints>::default()
    };

    // @audit - The pool may have been initialized and positions created
    //          through the extension call.
    //          This would set the specified price for the pool
    //          independently of the current state.
    let sqrt_ratio = tick_to_sqrt_ratio(initial_tick);
    self
        .pool_price
        .write(pool_key, PoolPrice { sqrt_ratio, tick: initial_tick, call_points });

    ...
}
```

This issue can be used for making the pool insolvent and steal tokens from other pools.

This issue is also present in the `swap(...)` and `update_position(...)` function from the `Core` contract. In those functions the current price is fetched before calling the extension. The code executed from the extension could execute swaps changing the state from the pool and after returning. The first swap will be executed using as current price the one originally fetched.

Recommendation(s): Ensure that the information used is up-to-date. This can be done by fetching the state again after the Extension logic was executed.

Status: Fixed.

Update from the client: This was fixed by reading or re-reading the pool price struct after calling the extension. Fixed in commit 9eda3c648fefeca28ffedd1b194267dc60c2d617.

6.2 [Info] Edge case bypass slippage protection

File(s): clear.cairo

Description: The `clear_minimum_to_recipient(...)` is used for pulling all the tokens from the contract. It adds a slippage protection mechanism by ensuring the existing token amount is greater than or equal to the minimum accepted amount.

```
#[inline(always)]
fn clear_minimum_to_recipient(
    self: @TContractState, token: IERC20Dispatcher, minimum: u256, recipient: ContractAddress
) -> u256 {
    let balance = token.balanceOf(get_contract_address());
    if (balance.is_non_zero()) { // @audit It has to revert if the amount is zero and the minimum is higher than zero.
        assert(balance >= minimum, 'CLEAR_AT_LEAST_MINIMUM');
        token.transfer(recipient, balance);
    }
    balance
}
```

However, when the contract balance is zero, the check of the requirement is skipped.

Recommendation(s): Consider checking the requirement independently of the contract's balance.

Status: Fixed.

Update from the client: Fixed in commit d01460c12e5737337855l6c69499187ce49cd5c5.

6.3 [Info] Incomplete validation of `token` argument in swaps

File(s): `router.cairo`

Description: When executing swaps through the `Router` contract, a `TokenAmount` argument must be provided. A `TokenAmount` value consists of the token address, indicating one of the tokens to be swapped, and an integer amount, indicating the number of the specified token to be sent or received from the swap.

To execute the swap, it is necessary to check if the `TokenAmount` corresponds to the `token1` of the pool used for the swap. To do this, the `token` from the `TokenAmount` value is compared with the `token1` field from the pool.

```
fn locked(...) -> Span<felt252> {
   ...

        loop {
            match route.pop_front() {
                Option::Some(node) => {
    // @audit - Token is only checked against token1 from the pool
                    let is_token1 = token_amount
                        .token == node
                        .pool_key
                        .token1;

                    let mut sqrt_ratio_limit = node.sqrt_ratio_limit;
                    if (sqrt_ratio_limit.is_zero()) {
                        sqrt_ratio_limit =
                            if is_price_increasing(
                                token_amount.amount.sign, is_token1
                            ) {
                                max_sqrt_ratio()
                            } else {
                                min_sqrt_ratio()
                            };
                    }

                    let delta = core
                        .swap(
                            node.pool_key,
                            SwapParameters {
                                amount: token_amount.amount,
                                is_token1: is_token1,
                                sqrt_ratio_limit,
                                skip_ahead: node.skip_ahead,
                            }
                        );
  ...
```

If the provided `token` is different from `token1`, it is not checked against `token0`. This allows the swap execution to continue even if the user provided the wrong tokens as argument. The swap will fail depending on if the `Router` contract holds the necessary tokens for the swap.

Recommendation(s): Consider fully validating the input provided by the user.

Status: Acknowledged.

Update from the client: An assert is not required because the transaction will fail if the route provided by the caller does not produce sufficient output token due to the slippage check in the separate call to *clear_minimum(...)*, or if the route is invalid, i.e. the output token from one swap is not the input token to the following swap. In cases where it does not fail due to these checks, e.g. a non-zero starting balance for the router, the user's swap is unaffected.

6.4 [Best Practices] Unused arguments

File(s): positions.cairo

Description: The functions `mint(...)` and `mint_with_referrer(...)` from the `Positions` contract require, but do not use, the `pool_key` and `bounds` arguments.

```
// @audit - `pool_key` and `bounds` arguments are not used
fn mint(ref self: ContractState, pool_key: PoolKey, bounds: Bounds) -> u64 {
    self.mint_v2(Zero::zero())
}

// @audit - `pool_key` and `bounds` arguments are not used
#[inline(always)]
fn mint_with_referrer(
    ref self: ContractState, pool_key: PoolKey, bounds: Bounds, referrer: ContractAddress
) -> u64 {
    self.mint_v2(referrer)
}
```

Recommendation(s): Consider removing unused arguments.

Status: Fixed.

Update from the client: These functions exist only for backwards compatibility. The arguments were previously only emitted in events, but those events are now removed. Documentation was added in commit 7e0fffb0633fd817c9ea30a345d298fbf96b80b5.

7 Complementary validations

7.1 Tick properties

As part of the assessment, the Nethermind Security team tried to verify the next properties for ticks:

- *sqrt_ratio_to_tick(tick_to_sqrt_ratio(tick)) == tick*
- *sqrt_ratio_at_tick(tick_to_sqrt_ratio(tick) - 1) == tick - 1*
- *sqrt_ratio_at_tick(tick_to_sqrt_ratio(tick) + 1) == tick*

These properties were checked for all the ticks in the interval $[-88722883, 88722883]$ using the starknet foundry toolkit, and no violation was found.

8 Documentation Evaluation

Software documentation refers to the written or visual information describing software's functionality, architecture, design, and implementation. It provides a comprehensive overview of the software system and helps users, developers, and stakeholders understand how the software works, how to use it, and how to maintain it. Software documentation can take different forms, such as user manuals, system manuals, technical specifications, requirements documents, design documents, and code comments. Software documentation is critical in software development, enabling effective communication between developers, testers, users, and other stakeholders. It helps to ensure that everyone involved in the development process has a shared understanding of the software system and its functionality. Moreover, software documentation can improve software maintenance by providing a clear and complete understanding of the software system, making it easier for developers to maintain, modify, and update the software over time. Smart contracts can use various types of software documentation. Some of the most common types include:

- Technical whitepaper: A technical whitepaper is a comprehensive document describing the smart contract's design and technical details. It includes information about the purpose of the contract, its architecture, its components, and how they interact with each other;
- User manual: A user manual is a document that provides information about how to use the smart contract. It includes step-by-step instructions on how to perform various tasks and explains the different features and functionalities of the contract;
- Code documentation: Code documentation is a document that provides details about the code of the smart contract. It includes information about the functions, variables, and classes used in the code, as well as explanations of how they work;
- API documentation: API documentation is a document that provides information about the API (Application Programming Interface) of the smart contract. It includes details about the methods, parameters, and responses that can be used to interact with the contract;
- Testing documentation: Testing documentation is a document that provides information about how the smart contract was tested. It includes details about the test cases that were used, the results of the tests, and any issues that were identified during testing;
- Audit documentation: Audit documentation includes reports, notes, and other materials related to the security audit of the smart contract. This type of documentation is critical in ensuring that the smart contract is secure and free from vulnerabilities.

These types of documentation are essential for smart contract development and maintenance. They help ensure that the contract is properly designed, implemented, and tested, and they provide a reference for developers who need to modify or maintain the contract in the future.

Remarks about Ekubo documentation

The Ekubo team has provided documentation about the core mechanisms of their protocol. The documentation is public as part of the developers documentation. The reviewed code also contained numerous comments giving more insights about the implementation and design choices. Moreover, the Ekubo team was available to address any inquiries or concerns raised by Nethermind Security.

9 Test Suite Evaluation

9.1 Compilation Output

```
> scarb build
Compiling ekubo v0.1.0 (.../EkuboProtocol/contracts/Scarb.toml)
    Finished release target(s) in 6 seconds
```

9.2 Tests Output

```
> scarb test
    Running cairo-test ekubo
   Compiling test(ekubo_unittest) ekubo v0.1.0 (.../EkuboProtocol/contracts/Scarb.toml)
    Finished release target(s) in 8 seconds
testing ekubo ...
running 452 tests
test ekubo::math::swap_test::test_exact_input_swap_max_fee_token0 ... ok (gas usage est.: 324350)
test ekubo::math::sqrt_ratio_test::test_next_sqrt_ratio_from_amount0_add_price_goes_down ... ok (gas usage est.: 95370)
test ekubo::math::bitmap_test::test_zeroable ... ok (gas usage est.: 3200)
test ekubo::math::muldiv_test::test_muldiv_up_div_by_zero ... ok (gas usage est.: 66960)
test ekubo::math::exp2_test::test_exp2_64 ... ok (gas usage est.: 27310)
test ekubo::math::bits_test::msb_3 ... ok (gas usage est.: 45560)
test ekubo::math::bits_test::msb_1 ... ok (gas usage est.: 45560)
test ekubo::math::bitmap_test::test_positive_cases_tick_spacing_one ... ok (gas usage est.: 3764060)
test ekubo::math::bits_test::msb_4 ... ok (gas usage est.: 45560)
test ekubo::math::max_liquidity_test::test_max_liquidity_for_token1_max_lower_half_range ... ok (gas usage est.: 515568)
test ekubo::math::bits_test::lsb_1 ... ok (gas usage est.: 53414)
test ekubo::math::max_liquidity_test::test_max_liquidity_panics_equal_ratios ... ok (gas usage est.: 169110)
test ekubo::math::max_liquidity_test::test_max_liquidity_for_token0_max_lower_half_range ... ok (gas usage est.: 638138)
test ekubo::math::exp2_test::test_exp2_126 ... ok (gas usage est.: 27310)
test ekubo::math::bitmap_test::test_positive_cases_tick_spacing_ten ... ok (gas usage est.: 3764060)
test ekubo::math::swap_test::test_exact_input_swap_max_fee_token1 ... ok (gas usage est.: 324350)
test ekubo::math::bits_test::msb_high_plus_four ... ok (gas usage est.: 45560)
test ekubo::math::muldiv_test::test_muldiv_up_div_by_zero_no_overflow ... ok (gas usage est.: 66960)
test ekubo::math::sqrt_ratio_test::test_next_sqrt_ratio_from_amount0_exact_out_overflow ... ok (gas usage est.: 92970)
test ekubo::math::max_liquidity_test::test_max_liquidity_for_token1_max_upper_half_range ... ok (gas usage est.: 516068)
test ekubo::math::bitmap_test::test_set_all_bits ... ok (gas usage est.: 21743360)
test ekubo::math::bits_test::lsb_2 ... ok (gas usage est.: 53414)
test ekubo::math::max_liquidity_test::test_max_liquidity_panics_zero_ratio_lower ... ok (gas usage est.: 169110)
test ekubo::math::max_liquidity_test::test_max_liquidity_for_token0_max_upper_half_range ... ok (gas usage est.: 638638)
test ekubo::math::ticks_test::zero_tick ... ok (gas usage est.: 501398)
test ekubo::math::exp2_test::test_exp2_127 ... ok (gas usage est.: 27310)
test ekubo::math::bits_test::msb_2_96_less_one ... ok (gas usage est.: 45560)
test ekubo::math::bitmap_test::test_positive_cases_non_one_tick_spacing ... ok (gas usage est.: 3764060)
test ekubo::math::muldiv_test::test_muldiv_overflows_exactly ... ok (gas usage est.: 66960)
test ekubo::math::sqrt_ratio_test::test_next_sqrt_ratio_from_amount0_exact_in_cant_underflow ... ok (gas usage est.:
↪  95370)
test ekubo::math::max_liquidity_test::test_max_liquidity_panics_order_ratios ... ok (gas usage est.: 169110)
test ekubo::math::bitmap_test::test_set_bit ... ok (gas usage est.: 670430)
test ekubo::math::bits_test::lsb_3 ... ok (gas usage est.: 53414)
test ekubo::math::exp2_test::test_exp2_128 ... ok (gas usage est.: 26810)
test ekubo::math::max_liquidity_test::test_max_liquidity_for_token1_max_at_full_range ... ok (gas usage est.: 17770)
test ekubo::math::max_liquidity_test::test_max_liquidity_less_than_liquidity_deltas ... ok (gas usage est.: 322410)
test ekubo::math::ticks_test::sqrt_ratio_of_max_tick_spacing ... ok (gas usage est.: 501398)
test ekubo::math::bitmap_test::test_negative_cases_tick_spacing_one ... ok (gas usage est.: 3764060)
test ekubo::math::max_liquidity_test::test_max_liquidity_concentrated_example ... ok (gas usage est.: 169610)
test ekubo::math::muldiv_test::test_muldiv_overflows_round_up ... ok (gas usage est.: 66960)
test ekubo::math::bits_test::lsb_2_127_plus_one ... ok (gas usage est.: 53414)
test ekubo::math::sqrt_ratio_test::test_next_sqrt_ratio_from_amount0_sub_price_goes_up ... ok (gas usage est.: 95370)
test ekubo::math::bitmap_test::test_set_bit_fails_max ... ok (gas usage est.: 39090)
test ekubo::math::exp2_test::test_exp2_129 ... ok (gas usage est.: 26810)
test ekubo::math::max_liquidity_test::test_liquidity_operations_rounding_increases_liquidity_in_range ... ok (gas usage
↪  est.: 1322186)
test ekubo::math::ticks_test::sqrt_ratio_of_double_max_tick_spacing ... ok (gas usage est.: 507528)
test ekubo::math::bits_test::msb_many_iterations_min_gas ... ok (gas usage est.: 53087620)
test ekubo::math::bitmap_test::test_negative_cases_tick_spacing_ten ... ok (gas usage est.: 3764060)
```

```
test ekubo::math::delta_test::test_amount1_delta_price_down ... ok (gas usage est.: 48890)
test ekubo::math::muldiv_test::test_muldiv_no_overflows_round_down ... ok (gas usage est.: 68360)
test ekubo::math::bits_test::lsb_2_127_plus_two ... ok (gas usage est.: 53414)
test ekubo::math::bitmap_test::test_unset_bit_fails_max ... ok (gas usage est.: 39090)
test ekubo::math::sqrt_ratio_test::test_next_sqrt_ratio_from_amount1_add_price_goes_up ... ok (gas usage est.: 29090)
test ekubo::math::exp2_test::test_exp2_255 ... ok (gas usage est.: 26810)
test ekubo::math::max_liquidity_test::test_liquidity_operations_rounding_increases_liquidity_price_below ... ok (gas
↪  usage est.: 1823894)
test ekubo::math::ticks_test::sqrt_ratio_of_max_tick_spacing_negative ... ok (gas usage est.: 501398)
test ekubo::math::bits_test::msb_max ... ok (gas usage est.: 45560)
test ekubo::math::bits_test::msb_0_panics ... ok (gas usage est.: 45060)
test ekubo::math::delta_test::test_amount1_delta_price_down_reverse ... ok (gas usage est.: 48890)
test ekubo::math::muldiv_test::test_muldiv_overflows_by_more ... ok (gas usage est.: 66960)
test ekubo::math::sqrt_ratio_test::test_next_sqrt_ratio_from_amount1_sub_price_goes_down ... ok (gas usage est.: 29090)
test ekubo::math::fee_test::test_compute_fee ... ok (gas usage est.: 65790)
test ekubo::math::bits_test::msb_2 ... ok (gas usage est.: 45560)
test ekubo::math::max_liquidity_test::test_liquidity_operations_rounding_increases_liquidity_price_above ... ok (gas
↪  usage est.: 1823894)
test ekubo::math::bitmap_test::test_double_set_reverts ... ok (gas usage est.: 116470)
test ekubo::math::bits_test::lsb_max ... ok (gas usage est.: 53414)
test ekubo::math::delta_test::test_amount1_delta_price_up ... ok (gas usage est.: 48890)
test ekubo::math::bits_test::lsb_0_panics ... ok (gas usage est.: 52914)
test ekubo::math::ticks_test::sqrt_ratio_of_double_max_tick_spacing_negative ... ok (gas usage est.: 507528)
test ekubo::math::muldiv_test::test_muldiv_fits ... ok (gas usage est.: 68360)
test ekubo::math::sqrt_ratio_test::test_next_sqrt_ratio_from_amount1_exact_out_overflow ... ok (gas usage est.: 26690)
test ekubo::math::delta_test::test_amount0_delta_price_down ... ok (gas usage est.: 96690)
test ekubo::math::fee_test::test_accumulate_fee_amount ... ok (gas usage est.: 40380)
test ekubo::math::bitmap_test::test_unset_not_set ... ok (gas usage est.: 39090)
test ekubo::math::ticks_test::sqrt_ratio_to_tick_negative_double ... ok (gas usage est.: 2772396)
test ekubo::math::muldiv_test::test_muldiv_div_by_zero ... ok (gas usage est.: 66960)
test ekubo::math::delta_test::test_amount1_delta_price_example_down ... ok (gas usage est.: 48890)
test ekubo::math::ticks_test::negative_zero_tick ... ok (gas usage est.: 501398)
test ekubo::math::muldiv_test::test_muldiv_up_fits_no_rounding ... ok (gas usage est.: 68360)
test ekubo::math::sqrt_ratio_test::test_next_sqrt_ratio_from_amount1_exact_in_overflow ... ok (gas usage est.: 26690)
test ekubo::math::delta_test::test_amount0_delta_price_down_reverse ... ok (gas usage est.: 96690)
test ekubo::tests::core_test::initialized_ticks::test_prev_initialized_tick_exceeds_min_tick_spacing ... ok (gas usage
↪  est.: 2360524)
test ekubo::math::ticks_test::sqrt_ratio_to_tick_max_sqrt_ratio ... ok (gas usage est.: 1419728)
test ekubo::math::liquidity_test::test_liquidity_delta_to_amount_delta_full_range_mid_price ... ok (gas usage est.:
↪  156780)
test ekubo::math::bitmap_test::test_next_set_bit_zero ... ok (gas usage est.: 496964)
test ekubo::math::ticks_test::one_tick ... ok (gas usage est.: 501398)
test ekubo::math::delta_test::test_amount1_delta_price_example_up ... ok (gas usage est.: 48890)
test ekubo::math::string_test::test_to_decimal ... ok (gas usage est.: 1629440)
test ekubo::math::bitmap_test::test_prev_set_bit_zero ... ok (gas usage est.: 576248)
test ekubo::math::muldiv_test::test_muldiv_max_inputs ... ok (gas usage est.: 68360)
test ekubo::math::ticks_test::sqrt_ratio_to_tick_min_sqrt_ratio ... ok (gas usage est.: 2274098)
test ekubo::math::liquidity_test::test_liquidity_delta_to_amount_delta_full_range_mid_price_withdraw ... ok (gas usage
↪  est.: 156780)
test ekubo::math::delta_test::test_amount0_delta_price_example_down ... ok (gas usage est.: 96690)
test ekubo::math::ticks_test::one_hundred_tick ... ok (gas usage est.: 501398)
test ekubo::math::delta_test::test_amount1_delta_price_down_round_up ... ok (gas usage est.: 48890)
test ekubo::math::ticks_test::test_min_sqrt_ratio_to_max_sqrt_ratio_size ... ok (gas usage est.: 9060)
test ekubo::math::muldiv_test::test_muldiv_up_max_inputs_no_rounding ... ok (gas usage est.: 68360)
test ekubo::math::string_test::test_large_numbers_to_decimal ... ok (gas usage est.: 2329220)
test ekubo::math::bitmap_test::test_next_set_bit_only_max_bit_set ... ok (gas usage est.: 619148)
test ekubo::math::liquidity_test::test_liquidity_delta_to_amount_delta_low_price_in_range ... ok (gas usage est.:
↪  156780)
test ekubo::math::delta_test::test_amount0_delta_price_example_up ... ok (gas usage est.: 96690)
test ekubo::math::ticks_test::negative_one_tick ... ok (gas usage est.: 501398)
test ekubo::math::delta_test::test_amount1_delta_price_up_round_up ... ok (gas usage est.: 48890)
test ekubo::math::swap_test::test_is_price_increasing_cases ... ok (gas usage est.: 3000)
test ekubo::math::bitmap_test::test_next_set_bit_only_min_bit_set ... ok (gas usage est.: 620348)
test ekubo::math::ticks_test::test_max_sqrt_ratio_size ... ok (gas usage est.: 3040)
test ekubo::math::liquidity_test::test_liquidity_delta_to_amount_delta_low_price_in_range_withdraw ... ok (gas usage
↪  est.: 156780)
test ekubo::math::muldiv_test::test_muldiv_phantom_overflow ... ok (gas usage est.: 68360)
test ekubo::math::delta_test::test_amount0_delta_price_up ... ok (gas usage est.: 96690)
test ekubo::math::ticks_test::negative_one_hundred_tick ... ok (gas usage est.: 501398)
test ekubo::math::delta_test::test_amount1_delta_overflow_entire_price_range_max_liquidity ... ok (gas usage est.:
↪  48190)
test ekubo::math::bitmap_test::test_prev_set_bit_only_min_bit_set ... ok (gas usage est.: 711146)
test ekubo::math::liquidity_test::test_liquidity_delta_to_amount_delta_high_price_in_range ... ok (gas usage est.:
↪  156780)
```

```
test ekubo::math::swap_test::test_no_op_swap_result ... ok (gas usage est.: 37440)
test ekubo::math::muldiv_test::test_muldiv_up_phantom_overflow_no_rounding ... ok (gas usage est.: 68360)
test ekubo::math::delta_test::test_amount0_delta_price_down_round_up ... ok (gas usage est.: 96690)
test ekubo::math::ticks_test::test_max_tick ... ok (gas usage est.: 501398)
test ekubo::math::delta_test::test_amount1_delta_no_overflow_half_price_range_half_liquidity ... ok (gas usage est.:
↪ 48890)
test ekubo::math::bitmap_test::test_prev_set_bit_only_max_bit_set ... ok (gas usage est.: 712346)
test ekubo::math::liquidity_test::test_liquidity_delta_to_amount_delta_concentrated_mid_price ... ok (gas usage est.:
↪ 1165336)
test ekubo::math::delta_test::test_amount0_delta_price_up_round_up ... ok (gas usage est.: 96690)
test ekubo::math::swap_test::test_swap_zero_amount_token0 ... ok (gas usage est.: 297840)
test ekubo::math::ticks_test::test_min_tick ... ok (gas usage est.: 501398)
test ekubo::math::exp2_test::test_exp2_0 ... ok (gas usage est.: 27310)
test ekubo::math::muldiv_test::test_muldiv_up_no_overflow_rounding_min ... ok (gas usage est.: 68360)
test ekubo::math::bitmap_test::test_next_set_bit_complex_scenario ... ok (gas usage est.: 3417686)
test ekubo::math::liquidity_test::test_liquidity_delta_to_amount_delta_concentrated_out_of_range_low ... ok (gas usage
↪ est.: 1162926)
test ekubo::math::mask_test::test_mask_4 ... ok (gas usage est.: 27310)
test ekubo::math::swap_test::test_swap_zero_amount_token1 ... ok (gas usage est.: 291830)
test ekubo::math::muldiv_test::test_muldiv_up_overflow_with_rounding ... ok (gas usage est.: 66960)
test ekubo::math::ticks_test::diff_between_min_tick_tick_plus_one ... ok (gas usage est.: 1012176)
test ekubo::math::exp2_test::test_exp2_1 ... ok (gas usage est.: 27310)
test ekubo::math::ticks_test::sqrt_ratio_to_tick_powers_of_tick ... ok (gas usage est.: 322436568)
test ekubo::math::mask_test::test_mask_126 ... ok (gas usage est.: 27310)
test ekubo::math::liquidity_test::test_liquidity_delta_to_amount_delta_concentrated_out_of_range_high ... ok (gas usage
↪ est.: 1163026)
test ekubo::math::swap_test::test_swap_ratio_equal_limit_token0 ... ok (gas usage est.: 291830)
test ekubo::math::muldiv_test::test_div ... ok (gas usage est.: 100520)
test ekubo::math::exp2_test::test_exp2_2 ... ok (gas usage est.: 27310)
test ekubo::math::ticks_test::tick_magnitude_exceeds_min ... ok (gas usage est.: 506458)
test ekubo::math::mask_test::test_mask_127 ... ok (gas usage est.: 27310)
test ekubo::math::ticks_test::sqrt_ratio_to_tick_max_sqrt_ratio_panics ... ok (gas usage est.: 535288)
test ekubo::math::swap_test::test_swap_ratio_equal_limit_token1 ... ok (gas usage est.: 291830)
test ekubo::math::liquidity_test::test_liquidity_delta_to_amount_delta_concentrated_in_range ... ok (gas usage est.:
↪ 1651274)
test ekubo::math::exp2_test::test_exp2_3 ... ok (gas usage est.: 27310)
test ekubo::math::mask_test::test_mask_128 ... ok (gas usage est.: 26810)
test ekubo::math::ticks_test::sqrt_ratio_to_tick_min_sqrt_ratio_less_one_panics ... ok (gas usage est.: 541308)
test ekubo::math::ticks_test::tick_magnitude_exceeds_max ... ok (gas usage est.: 505258)
test ekubo::math::mask_test::test_mask_0 ... ok (gas usage est.: 27310)
test ekubo::math::swap_test::test_swap_ratio_wrong_direction_token0_input ... ok (gas usage est.: 288330)
test ekubo::math::exp2_test::test_exp2_4 ... ok (gas usage est.: 27310)
test ekubo::math::mask_test::test_mask_129 ... ok (gas usage est.: 26810)
test ekubo::math::ticks_test::test_log2_2_128 ... ok (gas usage est.: 872120)
test ekubo::math::mask_test::test_mask_1 ... ok (gas usage est.: 27310)
test ekubo::math::swap_test::test_swap_ratio_wrong_direction_token0_input_zero_liquidity ... ok (gas usage est.: 288330)
test ekubo::math::mask_test::test_mask_255 ... ok (gas usage est.: 26810)
test ekubo::math::mask_test::test_mask_2 ... ok (gas usage est.: 27310)
test ekubo::math::ticks_test::test_internal_div_by_2_127 ... ok (gas usage est.: 24380)
test ekubo::math::swap_test::test_swap_ratio_wrong_direction_token0_zero_input_and_liquidity ... ok (gas usage est.:
↪ 297840)
test ekubo::math::max_liquidity_test::test_max_liquidity_for_token0_max_at_full_range ... ok (gas usage est.: 443730)
test ekubo::tests::core_test::owner_tests::test_replace_class_hash_cannot_be_called_by_non_owner ... ok (gas usage
↪ est.: 398940)
test ekubo::tests::core_test::initialized_ticks::test_next_prev_initialized_tick_none_initialized ... ok (gas usage
↪ est.: 6279304)
test ekubo::math::mask_test::test_mask_3 ... ok (gas usage est.: 27310)
test ekubo::math::swap_test::test_swap_ratio_wrong_direction_token0_output ... ok (gas usage est.: 288330)
test ekubo::math::ticks_test::sqrt_ratio_to_tick_zero ... ok (gas usage est.: 1403738)
test ekubo::math::swap_test::test_swap_ratio_wrong_direction_token0_output_zero_liquidity ... ok (gas usage est.:
↪ 288330)
test ekubo::math::ticks_test::sqrt_ratio_to_tick_one ... ok (gas usage est.: 1904446)
test ekubo::math::ticks_test::sqrt_ratio_to_tick_one_plus_one ... ok (gas usage est.: 1910466)
test ekubo::math::swap_test::test_swap_ratio_wrong_direction_token0_zero_output_and_liquidity ... ok (gas usage est.:
↪ 297840)
test ekubo::tests::core_test::owner_tests::test_fails_upgrading_to_other_contract ... ok (gas usage est.: 469940)
test ekubo::tests::owned_nft_test::test_nft_supports_interfaces ... ok (gas usage est.: 1583570)
test ekubo::tests::core_test::owner_tests::test_replace_class_hash_can_be_called_by_owner ... ok (gas usage est.:
↪ 533990)
test ekubo::math::ticks_test::sqrt_ratio_to_tick_one_minus_one ... ok (gas usage est.: 1907956)
test ekubo::math::swap_test::test_swap_ratio_wrong_direction_token1_input ... ok (gas usage est.: 288330)
test ekubo::math::ticks_test::sqrt_ratio_to_tick_negative_one ... ok (gas usage est.: 2772396)
test ekubo::math::swap_test::test_swap_ratio_wrong_direction_token1_input_zero_liquidity ... ok (gas usage est.: 288330)
test ekubo::math::ticks_test::sqrt_ratio_to_tick_within_ticks_example ... ok (gas usage est.: 3279624)
```

```
test ekubo::math::swap_test::test_swap_ratio_wrong_direction_token1_zero_input_and_liquidity ... ok (gas usage est.:
↪  297840)
test ekubo::math::ticks_test::sqrt_ratio_to_tick_negative_one_minus_one ... ok (gas usage est.: 2778316)
test ekubo::tests::owned_nft_test::test_replace_class_hash_can_be_called_by_owner ... ok (gas usage est.: 567490)
test ekubo::math::swap_test::test_swap_ratio_wrong_direction_token1_output ... ok (gas usage est.: 288330)
test ekubo::math::ticks_test::sqrt_ratio_to_tick_negative_one_plus_one ... ok (gas usage est.: 2778416)
test ekubo::math::swap_test::test_swap_ratio_wrong_direction_token1_output_zero_liquidity ... ok (gas usage est.:
↪  288330)
test ekubo::math::ticks_test::sqrt_ratio_to_tick_double ... ok (gas usage est.: 1904446)
test ekubo::tests::core_test::owner_tests::test_transfer_ownership ... ok (gas usage est.: 662010)
test ekubo::tests::core_test::initialize_pool_tests::test_initialize_pool_works_uninitialized ... ok (gas usage est.:
↪  2344416)
test ekubo::math::swap_test::test_swap_ratio_wrong_direction_token1_zero_output_and_liquidity ... ok (gas usage est.:
↪  297840)
test ekubo::math::swap_test::test_swap_against_liquidity_max_limit_token0_input ... ok (gas usage est.: 320650)
test ekubo::tests::owned_nft_test::test_set_metadata_callable_by_owner ... ok (gas usage est.: 949480)
test ekubo::math::swap_test::test_swap_against_liquidity_max_limit_token0_minimum_input ... ok (gas usage est.: 320650)
test ekubo::tests::core_test::initialize_pool_tests::test_initialize_pool_fails_max_tick_spacing_plus_one ... ok (gas
↪  usage est.: 1172308)
test ekubo::tests::core_test::initialize_pool_tests::test_initialize_pool_fails_token_order_same_token ... ok (gas
↪  usage est.: 1169528)
test ekubo::tests::core_test::owner_tests::test_transfer_ownership_then_replace_class_hash_fails ... ok (gas usage
↪  est.: 590700)
test ekubo::math::swap_test::test_swap_against_liquidity_min_limit_token0_output ... ok (gas usage est.: 320650)
test ekubo::tests::router_test::test_router_swap_initialized_pool_no_liquidity_token0_in ... ok (gas usage est.:
↪  105616064)
test ekubo::math::swap_test::test_swap_against_liquidity_min_limit_token0_minimum_output ... ok (gas usage est.: 320650)
test ekubo::tests::core_test::initialize_pool_tests::test_initialize_pool_fails_token_order_wrong_order ... ok (gas
↪  usage est.: 1169528)
test ekubo::tests::core_test::initialize_pool_tests::test_initialize_pool_fails_already_initialized ... ok (gas usage
↪  est.: 2157886)
test ekubo::math::swap_test::test_swap_against_liquidity_max_limit_token1_input ... ok (gas usage est.: 320650)
test ekubo::tests::core_test::owner_tests::test_transfer_ownership_then_replace_class_hash_succeeds ... ok (gas usage
↪  est.: 668500)
test ekubo::math::swap_test::test_swap_against_liquidity_max_limit_token1_minimum_input ... ok (gas usage est.: 320650)
test ekubo::tests::owned_nft_test::test_nft_custom_uri ... ok (gas usage est.: 1084210)
test ekubo::math::swap_test::test_swap_against_liquidity_min_limit_token1_output ... ok (gas usage est.: 320650)
test ekubo::tests::core_test::initialize_pool_tests::test_initialize_pool_fails_token_order_zero_token ... ok (gas
↪  usage est.: 1169528)
test ekubo::math::swap_test::test_swap_against_liquidity_min_limit_token1_minimum_output ... ok (gas usage est.: 320650)
test ekubo::tests::core_test::owner_tests::test_fails_upgrading_to_other_contract_without_interface_id ... ok (gas
↪  usage est.: 476950)
test ekubo::math::swap_test::test_swap_against_liquidity_hit_limit_token0_input ... ok (gas usage est.: 324350)
test ekubo::math::swap_test::test_swap_against_liquidity_hit_limit_token1_input ... ok (gas usage est.: 324350)
test ekubo::tests::core_test::initialize_pool_tests::test_initialize_pool_fails_zero_tick_spacing ... ok (gas usage
↪  est.: 1169528)
test ekubo::tests::core_test::initialize_pool_tests::test_maybe_initialize_pool_twice ... ok (gas usage est.: 3649864)
test ekubo::math::swap_test::test_swap_against_liquidity_hit_limit_token0_output ... ok (gas usage est.: 324350)
test ekubo::math::swap_test::test_swap_against_liquidity_hit_limit_token1_output ... ok (gas usage est.: 324350)
test ekubo::math::swap_test::test_swap_max_amount_token0 ... ok (gas usage est.: 297840)
test ekubo::tests::router_test::test_router_swap_initialized_pool_no_liquidity_token1_in ... ok (gas usage est.:
↪  102965480)
test ekubo::tests::core_test::initialize_pool_tests::test_initialize_pool_succeeds_max_tick_spacing ... ok (gas usage
↪  est.: 1180528)
test ekubo::math::swap_test::test_swap_min_amount_token0 ... ok (gas usage est.: 297840)
test ekubo::math::swap_test::test_swap_min_amount_token0_very_high_price ... ok (gas usage est.: 297840)
test ekubo::tests::positions_test::test_deposit_liquidity_concentrated_unbalanced_in_range_price_higher ... ok (gas
↪  usage est.: 12416060)
test ekubo::math::swap_test::test_swap_max_amount_token1 ... ok (gas usage est.: 297840)
test ekubo::tests::owned_nft_test::test_nft_indexing_token_ids ... ok (gas usage est.: 2052710)
test ekubo::math::swap_test::test_swap_min_amount_token1 ... ok (gas usage est.: 297840)
test ekubo::tests::core_test::initialized_ticks::test_prev_initialized_tick_min_tick_minus_one ... ok (gas usage est.:
↪  1995536)
test ekubo::math::swap_test::test_swap_min_amount_token1_very_high_price ... ok (gas usage est.: 297840)
test ekubo::math::swap_test::test_swap_max_fee ... ok (gas usage est.: 297840)
test ekubo::math::swap_test::test_swap_min_fee ... ok (gas usage est.: 297840)
test ekubo::math::swap_test::test_swap_all_max_inputs ... ok (gas usage est.: 297840)
test ekubo::math::swap_test::test_swap_all_max_inputs_no_fee ... ok (gas usage est.: 288330)
test ekubo::tests::router_test::test_router_swap_not_initialized_pool ... ok (gas usage est.: 4027796)
test ekubo::math::swap_test::test_swap_result_example_usdc_wbtc ... ok (gas usage est.: 297840)
test ekubo::tests::core_test::initialized_ticks::test_prev_initialized_tick_min_tick ... ok (gas usage est.: 1996996)
test ekubo::math::swap_test::test_exact_output_swap_max_fee_token0 ... ok (gas usage est.: 324350)
test ekubo::math::swap_test::test_exact_output_swap_max_fee_large_amount_token0 ... ok (gas usage est.: 324350)
test ekubo::math::swap_test::test_exact_output_swap_max_fee_token0_limit_reached ... ok (gas usage est.: 324350)
```

```
test ekubo::math::swap_test::test_exact_output_swap_max_fee_token1 ... ok (gas usage est.: 324350)
test ekubo::tests::owned_nft_test::test_nft_transfer_from_succeeds_from_approved ... ok (gas usage est.: 1102280)
test ekubo::math::swap_test::test_exact_output_swap_max_fee_token1_limit_reached ... ok (gas usage est.: 324350)
test ekubo::tests::owned_nft_test::test_nft_indexing_token_ids_not_sorted ... ok (gas usage est.: 2866360)
test ekubo::tests::core_test::initialized_ticks::test_next_initialized_tick_max_tick ... ok (gas usage est.: 1970552)
test ekubo::tests::owned_nft_test::test_nft_transfer_from_succeeds_from_approved_for_all ... ok (gas usage est.:
↪  1076070)
test ekubo::tests::owned_nft_test::test_our_uris_fit ... ok (gas usage est.: 579210)
test ekubo::tests::core_test::initialized_ticks::test_next_initialized_tick_max_tick_minus_one ... ok (gas usage est.:
↪  1987532)
test ekubo::tests::owned_nft_test::test_nft_token_uri ... ok (gas usage est.: 3247890)
test ekubo::tests::core_test::initialized_ticks::test_next_initialized_tick_exceeds_max_tick_spacing ... ok (gas usage
↪  est.: 1979772)
test ekubo::tests::owned_nft_test::test_nft_token_uri_reverts_too_long ... ok (gas usage est.: 380150)
test ekubo::tests::owned_nft_test::test_nft_indexing_token_ids_snake_case ... ok (gas usage est.: 2299450)
test ekubo::tests::owned_nft_test::test_nft_token_uri_reverts_token_id_too_big ... ok (gas usage est.: 380150)
test ekubo::tests::positions_test::test_withdraw_not_collected_fees_token1 ... ok (gas usage est.: 27503800)
test ekubo::tests::owned_nft_test::test_nft_approve_only_owner_can_approve ... ok (gas usage est.: 710550)
test ekubo::tests::router_test::test_router_quote_not_initialized_pool ... ok (gas usage est.: 4068886)
test ekubo::tests::positions_test::test_deposit_liquidity_concentrated_unbalanced_in_range_price_lower ... ok (gas
↪  usage est.: 12416060)
test ekubo::tests::owned_nft_test::test_burn_makes_token_non_transferrable ... ok (gas usage est.: 2305870)
test ekubo::tests::owned_nft_test::test_nft_balance_of ... ok (gas usage est.: 1144260)
test ekubo::tests::router_test::test_router_quote_initialized_pool_no_liquidity ... ok (gas usage est.: 105379644)
test ekubo::tests::router_test::test_router_quote_initialized_pool_with_liquidity ... ok (gas usage est.: 57587564)
test ekubo::tests::positions_test::test_failure_case_integration_tests_amount_cannot_be_met_due_to_overflow ... ok (gas
↪  usage est.: 28953522)
test ekubo::tests::router_test::test_router_get_market_depth_at_sqrt_ratio_starting_ratio ... ok (gas usage est.:
↪  138835224)
test ekubo::tests::positions_test::test_replace_class_hash_can_be_called_by_owner ... ok (gas usage est.: 2436038)
test ekubo::tests::positions_test::test_locked_cannot_be_called_directly ... ok (gas usage est.: 2456488)
test ekubo::tests::owned_nft_test::test_is_account_authorized ... ok (gas usage est.: 2640560)
test ekubo::tests::positions_test::test_get_pool_price_normal_pool ... ok (gas usage est.: 7059724)
test ekubo::tests::owned_nft_test::test_burn_makes_token_non_transferrable_error ... ok (gas usage est.: 1353140)
test ekubo::tests::owned_nft_test::test_nft_approve_fails_id_not_exists ... ok (gas usage est.: 437400)
test ekubo::tests::owned_nft_test::test_nft_approve_succeeds_after_mint ... ok (gas usage est.: 963730)
test ekubo::tests::positions_test::test_deposit_fails_min_liquidity ... ok (gas usage est.: 5102784)
test ekubo::tests::positions_test::test_deposit_liquidity_concentrated_out_of_range_price_upper ... ok (gas usage est.:
↪  11312940)
test ekubo::tests::router_test::test_router_quote_multihop_routes ... ok (gas usage est.: 82560294)
test ekubo::types::i129_test::test_gte ... ok (gas usage est.: 36250)
test ekubo::types::i129_test::test_eq ... ok (gas usage est.: 17150)
test ekubo::types::i129_test::test_lte ... ok (gas usage est.: 20200)
test ekubo::types::i129_test::test_mul_negative_negative ... ok (gas usage est.: 10640)
test ekubo::types::i129_test::test_mul_negative_positive ... ok (gas usage est.: 10640)
test ekubo::types::i129_test::test_mul_positive_negative ... ok (gas usage est.: 10640)
test ekubo::tests::positions_test::test_withdraw_not_collected_fees_token0 ... ok (gas usage est.: 30021582)
test ekubo::types::i129_test::test_round_trip_many_values ... ok (gas usage est.: 83540)
test ekubo::types::call_points_test::test_u8_empty_into_default_call_points ... ok (gas usage est.: 24330)
test ekubo::types::i129_test::test_store_write_read_1 ... ok (gas usage est.: 13540)
test ekubo::types::call_points_test::test_u8_max_cannot_convert ... ok (gas usage est.: 19780)
test ekubo::types::i129_test::test_store_write_read_negative_1 ... ok (gas usage est.: 13340)
test ekubo::tests::owned_nft_test::test_nft_transfer_from ... ok (gas usage est.: 2081920)
test ekubo::types::call_points_test::test_u8_into_all_call_points ... ok (gas usage est.: 24330)
test ekubo::types::i129_test::test_store_write_read_0 ... ok (gas usage est.: 13340)
test ekubo::tests::positions_test::test_deposit_liquidity_updates_tick_states_at_bounds ... ok (gas usage est.:
↪  12477320)
test ekubo::types::call_points_test::test_lower_bits_result_in_none ... ok (gas usage est.: 165480)
test ekubo::types::i129_test::test_store_write_read_negative_0 ... ok (gas usage est.: 13640)
test ekubo::types::call_points_test::test_u8_into_after_initialize_call_points ... ok (gas usage est.: 24330)
test ekubo::types::i129_test::test_store_write_read_max_value ... ok (gas usage est.: 13340)
test ekubo::tests::positions_test::test_deposit_liquidity_full_range ... ok (gas usage est.: 11800610)
test ekubo::types::call_points_test::test_u8_into_before_swap_call_points ... ok (gas usage est.: 24330)
test ekubo::types::i129_test::test_store_write_read_min_value ... ok (gas usage est.: 13340)
test ekubo::tests::core_test::initialized_ticks::test_next_prev_initialized_tick_several_initialized ... ok (gas usage
↪  est.: 33471362)
test ekubo::tests::owned_nft_test::test_nft_transfer_from_fails_not_from_owner ... ok (gas usage est.: 888350)
test ekubo::types::call_points_test::test_u8_into_after_swap_call_points ... ok (gas usage est.: 24330)
test ekubo::types::i129_test::test_store_write_min_value_minus_one ... ok (gas usage est.: 2470)
test ekubo::types::i129_test::test_store_write_max_value_plus_one ... ok (gas usage est.: 2470)
test ekubo::types::call_points_test::test_u8_into_before_update_position_call_points ... ok (gas usage est.: 24330)
test ekubo::types::i129_test::test_add_delta_no_overflow ... ok (gas usage est.: 53900)
test ekubo::types::call_points_test::test_u8_into_after_update_position_call_points ... ok (gas usage est.: 24330)
test ekubo::types::i129_test::test_add_delta_panics_underflow ... ok (gas usage est.: 10760)
```

```
test ekubo::types::call_points_test::test_conversion_all_possible_values ... ok (gas usage est.: 1739270)
test ekubo::types::i129_test::test_add_delta_panics_underflow_max ... ok (gas usage est.: 10760)
test ekubo::types::delta_test::test_delta_zeroable ... ok (gas usage est.: 10020)
test ekubo::types::i129_test::test_add_delta_max_inputs ... ok (gas usage est.: 11260)
test ekubo::tests::router_test::test_router_get_market_depth ... ok (gas usage est.: 112374668)
test ekubo::types::i129_test::test_add_delta_panics_overflow ... ok (gas usage est.: 10760)
test ekubo::types::delta_test::test_delta_add ... ok (gas usage est.: 20840)
test ekubo::tests::positions_test::test_deposit_liquidity_concentrated_out_of_range_price_lower ... ok (gas usage est.:
↪ 11332940)
test ekubo::types::i129_test::test_add_delta_panics_overflow_reverse ... ok (gas usage est.: 10760)
test ekubo::types::delta_test::test_delta_addeq ... ok (gas usage est.: 20840)
test ekubo::tests::core_test::locks::test_error_from_action_not_locked ... ok (gas usage est.: 1970918)
test ekubo::types::fees_per_liquidity_test::test_MAX_PRIME_plus_one_is_zero ... ok (gas usage est.: 1300)
test ekubo::types::keys_test::test_pool_key_hash_differs_for_any_field_or_state_change ... ok (gas usage est.: 1430900)
test ekubo::types::fees_per_liquidity_test::test_fpl_zeroable ... ok (gas usage est.: 3400)
test ekubo::types::keys_test::test_pool_key_check_valid_order_wrong_order ... ok (gas usage est.: 10140)
test ekubo::types::keys_test::test_pool_key_check_valid_order_same_token ... ok (gas usage est.: 10140)
test ekubo::types::fees_per_liquidity_test::test_fpl_add_sub_zeroable ... ok (gas usage est.: 6200)
test ekubo::types::keys_test::test_pool_key_check_non_zero ... ok (gas usage est.: 10140)
test ekubo::types::fees_per_liquidity_test::test_fpl_underflow_sub ... ok (gas usage est.: 3000)
test ekubo::types::keys_test::test_pool_key_check_tick_spacing_non_zero ... ok (gas usage est.: 10140)
test ekubo::types::fees_per_liquidity_test::test_fpl_overflow_add ... ok (gas usage est.: 3000)
test ekubo::types::keys_test::test_pool_key_check_tick_spacing_max ... ok (gas usage est.: 12720)
test ekubo::types::fees_per_liquidity_test::test_fees_per_liquidity_new ... ok (gas usage est.: 27540)
test ekubo::types::keys_test::test_pool_key_check_valid_is_valid ... ok (gas usage est.: 38460)
test ekubo::types::fees_per_liquidity_test::test_to_fees_per_liquidity_max_fees ... ok (gas usage est.: 13020)
test ekubo::types::keys_test::test_pool_key_hash ... ok (gas usage est.: 93900)
test ekubo::types::fees_per_liquidity_test::test_to_fees_per_liquidity_overflows ... ok (gas usage est.: 13020)
test ekubo::types::keys_test::test_pool_key_hash_result ... ok (gas usage est.: 23750)
test ekubo::types::fees_per_liquidity_test::test_to_fees_per_liquidity_div_by_zero ... ok (gas usage est.: 13020)
test ekubo::tests::router_test::test_router_quote_to_delta ... ok (gas usage est.: 48838724)
test ekubo::types::keys_test::test_pool_key_hash_result_reverse ... ok (gas usage est.: 23750)
test ekubo::types::fees_per_liquidity_test::test_storage_size ... ok (gas usage est.: 3670)
test ekubo::types::bounds_test::test_check_valid_fails_exceed_max_tick ... ok (gas usage est.: 30240)
test ekubo::types::keys_test::test_position_key_hash_differs_for_any_field_or_state_change ... ok (gas usage est.:
↪ 899500)
test ekubo::types::i129_test::test_legacy_hash_i129 ... ok (gas usage est.: 102640)
test ekubo::types::keys_test::test_position_key_hash ... ok (gas usage est.: 66520)
test ekubo::types::bounds_test::test_check_valid_fails_below_min_tick ... ok (gas usage est.: 31440)
test ekubo::types::keys_test::test_position_key_hash_result ... ok (gas usage est.: 22440)
test ekubo::types::i129_test::test_zero ... ok (gas usage est.: 9410)
test ekubo::types::bounds_test::test_check_valid_fails_tick_spacing_both ... ok (gas usage est.: 23880)
test ekubo::types::keys_test::test_position_key_hash_result_reverse ... ok (gas usage est.: 22440)
test ekubo::types::i129_test::test_div_i129 ... ok (gas usage est.: 28260)
test ekubo::types::bounds_test::test_check_valid_fails_tick_spacing_lower ... ok (gas usage est.: 23880)
test ekubo::types::keys_test::test_saved_balance_key_hash_differs ... ok (gas usage est.: 357260)
test ekubo::types::i129_test::test_gt ... ok (gas usage est.: 19200)
test ekubo::types::bounds_test::test_check_valid_fails_tick_spacing_upper ... ok (gas usage est.: 23880)
test ekubo::types::keys_test::test_saved_balance_key_hash ... ok (gas usage est.: 14690)
test ekubo::types::i129_test::test_lt ... ok (gas usage est.: 51870)
test ekubo::types::bounds_test::test_check_valid_tick_spacing_matches ... ok (gas usage est.: 23880)
test ekubo::tests::positions_test::test_deposit_liquidity_concentrated ... ok (gas usage est.: 12189180)
test ekubo::types::keys_test::test_saved_balance_key_hash_reverse ... ok (gas usage est.: 14690)
test ekubo::types::bounds_test::test_max_bounds_1_spacing ... ok (gas usage est.: 17400)
test ekubo::types::pool_price_test::test_packing_round_trip_many_values ... ok (gas usage est.: 634310)
test ekubo::types::bounds_test::test_max_bounds_2_spacing ... ok (gas usage est.: 17400)
test ekubo::types::pool_price_test::test_fails_if_sqrt_ratio_out_of_range_max ... ok (gas usage est.: 77110)
test ekubo::types::bounds_test::test_max_bounds_max_spacing ... ok (gas usage est.: 17400)
test ekubo::types::pool_price_test::test_fails_if_sqrt_ratio_zero ... ok (gas usage est.: 71490)
test ekubo::types::bounds_test::test_max_bounds_max_minus_one_spacing ... ok (gas usage est.: 17400)
test ekubo::types::pool_price_test::test_fails_if_sqrt_ratio_one ... ok (gas usage est.: 71090)
test ekubo::types::bounds_test::test_max_bounds_max_plus_one ... ok (gas usage est.: 11880)
test ekubo::types::pool_price_test::test_fails_if_sqrt_ratio_out_of_range_min ... ok (gas usage est.: 77110)
test ekubo::types::bounds_test::test_max_bounds_zero ... ok (gas usage est.: 11880)
test ekubo::types::pool_price_test::test_fails_if_tick_out_of_range_max ... ok (gas usage est.: 77650)
test ekubo::types::call_points_test::test_default_call_points_into_u8 ... ok (gas usage est.: 16750)
test ekubo::types::pool_price_test::test_fails_if_tick_out_of_range_min ... ok (gas usage est.: 79050)
test ekubo::types::call_points_test::test_all_call_points_into_u8 ... ok (gas usage est.: 16750)
test ekubo::types::pool_price_test::test_store_packing_pool_price ... ok (gas usage est.: 119790)
test ekubo::types::position_test::test_multiply_and_get_limb1 ... ok (gas usage est.: 481670)
test ekubo::types::position_test::test_positions_zeroable ... ok (gas usage est.: 2200)
test ekubo::types::position_test::test_is_zero_for_nonzero_fees ... ok (gas usage est.: 2200)
test ekubo::types::position_test::test_is_zero_for_nonzero_liquidity ... ok (gas usage est.: 2400)
test ekubo::types::position_test::test_fees_calculation_zero_liquidity ... ok (gas usage est.: 141640)
```

```
test ekubo::types::position_test::test_fees_calculation_one_liquidity_max_difference ... ok (gas usage est.: 141640)
test ekubo::types::position_test::test_fees_calculation_one_liquidity_max_difference_token0_only ... ok (gas usage
↪  est.: 141640)
test ekubo::tests::router_test::test_router_get_market_depth_at_sqrt_ratio_starting_ratio_diff_ratios ... ok (gas usage
↪  est.: 75952884)
test ekubo::types::position_test::test_fees_calculation_one_liquidity_max_difference_token1_only ... ok (gas usage
↪  est.: 141640)
test ekubo::tests::token_registry_test::test_ten_pow ... ok (gas usage est.: 425060)
test ekubo::types::position_test::test_fees_calculation_fees_pre_overflow ... ok (gas usage est.: 141640)
test ekubo::types::position_test::test_fees_calculation_fees_overflow ... ok (gas usage est.: 141640)
test ekubo::types::position_test::test_fees_calculation_fees_example_value ... ok (gas usage est.: 141840)
test ekubo::tests::upgradeable_test::test_replace_class_hash ... ok (gas usage est.: 481190)
test ekubo::tests::upgradeable_test::test_replace_non_zero_class_hash_not_owner ... ok (gas usage est.: 343430)
test ekubo::tests::upgradeable_test::test_replace_class_hash_not_owner_after_transfer ... ok (gas usage est.: 511500)
test ekubo::tests::core_test::locks::test_flash_borrow_balanced ... ok (gas usage est.: 9442790)
test ekubo::tests::upgradeable_test::test_replace_non_zero_class_hash_without_interface_id ... ok (gas usage est.:
↪  405230)
test ekubo::types::bounds_test::test_legacy_hash_bounds ... ok (gas usage est.: 173700)
test ekubo::types::bounds_test::test_legacy_hash_bounds_result ... ok (gas usage est.: 73720)
test ekubo::types::bounds_test::test_check_valid_succeeds_default_1 ... ok (gas usage est.: 34960)
test ekubo::types::bounds_test::test_check_valid_fails_default_123 ... ok (gas usage est.: 34960)
test ekubo::tests::positions_test::test_deposit_liquidity_concentrated_mint_and_deposit ... ok (gas usage est.:
↪  11149020)
test ekubo::tests::upgradeable_test::test_replace_class_hash_after_owner_change ... ok (gas usage est.: 589300)
test ekubo::types::bounds_test::test_check_valid_fails_zero ... ok (gas usage est.: 23880)
test ekubo::tests::upgradeable_test::test_replace_zero_class_hash ... ok (gas usage est.: 344230)
test ekubo::tests::positions_test::test_withdraw_partial_leave_fees ... ok (gas usage est.: 32963788)
test ekubo::tests::positions_test::test_deposit_swap_through_upper_tick_fees_accounting ... ok (gas usage est.:
↪  27692246)
test ekubo::tests::core_test::locks::test_flash_borrow_underpay ... ok (gas usage est.: 4781600)
test ekubo::tests::extensions_test::test_mock_extension_initialize_pool_is_called ... ok (gas usage est.: 3555088)
test ekubo::tests::core_test::locks::test_small_amount_liquidity_add_tick_spacing_not_divisible_upper ... ok (gas usage
↪  est.: 7111724)
test ekubo::tests::positions_test::test_deposit_withdraw_protocol_fee_then_deposit ... ok (gas usage est.: 26838598)
test ekubo::tests::core_test::locks::test_flash_borrow_overpay ... ok (gas usage est.: 5365830)
test ekubo::tests::positions_test::test_deposit_swap_through_lower_tick_fees_accounting ... ok (gas usage est.:
↪  30210028)
test ekubo::tests::positions_test::test_deposit_then_withdraw_with_fees ... ok (gas usage est.: 35299726)
test ekubo::tests::core_test::locks::test_small_amount_liquidity_add_no_tokens ... ok (gas usage est.: 7117244)
test ekubo::tests::router_test::test_router_get_market_depth_v2 ... ok (gas usage est.: 139842124)
test ekubo::tests::extensions_test::test_mock_extension_swap_is_called ... ok (gas usage est.: 9512474)
test ekubo::tests::core_test::locks::test_flash_borrow_more_than_core_balance ... ok (gas usage est.: 4759620)
test ekubo::tests::core_test::locks::test_swap_token0_exact_input_against_small_liquidity_no_tick_cross ... ok (gas
↪  usage est.: 23164618)
test ekubo::tests::core_test::locks::test_assert_locker_id_call ... ok (gas usage est.: 2733848)
test ekubo::tests::positions_test::test_deposit_swap_round_trip_accounting ... ok (gas usage est.: 60159608)
test ekubo::tests::extensions_test::test_mock_extension_before_update_position_only ... ok (gas usage est.: 15013910)
test ekubo::tests::positions_test::test_deposit_then_partial_withdraw_with_fees ... ok (gas usage est.: 61222980)
test ekubo::tests::core_test::locks::test_small_amount_liquidity_add ... ok (gas usage est.: 10703854)
test ekubo::tests::extensions_test::test_mock_extension_update_position_is_called ... ok (gas usage est.: 10775434)
test ekubo::tests::extensions_test::test_mock_extension_cannot_be_called_directly ... ok (gas usage est.: 1226160)
test ekubo::tests::extensions_test::test_mock_extension_can_be_called_by_core ... ok (gas usage est.: 1453650)
test ekubo::tests::core_test::locks::test_relock_call ... ok (gas usage est.: 8565338)
test ekubo::tests::positions_test::test_deposit_existing_position ... ok (gas usage est.: 45425626)
test ekubo::tests::core_test::locks::test_swap_token1_exact_input_against_small_liquidity_with_tick_cross ... ok (gas
↪  usage est.: 22791848)
test ekubo::tests::core_test::locks::test_assert_locker_id_call_wrong ... ok (gas usage est.: 2593078)
test ekubo::tests::core_test::locks::test_relock_call_fails_invalid_id ... ok (gas usage est.: 2599618)
test ekubo::tests::core_test::locks::test_accumulate_fees_per_liquidity_not_extension ... ok (gas usage est.: 11478184)
test ekubo::tests::extensions_test::test_mock_extension_not_called_back_into_own_pool ... ok (gas usage est.: 10032930)
test ekubo::tests::mock_erc20_test::test_constructor ... ok (gas usage est.: 323830)
test ekubo::tests::extensions_test::test_mock_extension_after_update_position_only ... ok (gas usage est.: 15029130)
test ekubo::tests::mock_erc20_test::test_transfer ... ok (gas usage est.: 785150)
test ekubo::tests::extensions_test::test_mock_extension_no_call_points ... ok (gas usage est.: 14712970)
test ekubo::tests::owned_nft_test::test_nft_name_symbol_token_uri ... ok (gas usage est.: 1084210)
test ekubo::tests::core_test::locks::test_swap_token1_exact_output_against_small_liquidity_with_tick_cross ... ok (gas
↪  usage est.: 25309630)
test ekubo::tests::core_test::locks::test_zero_liquidity_add ... ok (gas usage est.: 9358126)
test ekubo::tests::positions_test::test_deposit_swap_multiple_positions ... ok (gas usage est.: 71812502)
test ekubo::tests::core_test::locks::test_swap_token0_exact_input_against_small_liquidity_no_tick_cross_example ... ok
↪  (gas usage est.: 30080640)
test ekubo::tests::extensions_test::test_mock_extension_before_swap_only ... ok (gas usage est.: 15010120)
test ekubo::tests::extensions_test::test_mock_extension_is_called_back_into_other_pool ... ok (gas usage est.: 12483620)
```

```
test ekubo::tests::core_test::locks::test_small_amount_liquidity_add_tick_spacing_not_divisible_lower ... ok (gas usage
↪  est.: 7111724)
test ekubo::tests::core_test::locks::test_swap_token0_exact_input_against_small_liquidity_with_tick_cross ... ok (gas
↪  usage est.: 25309630)
test ekubo::tests::core_test::locks::test_swap_exact_input_token1_multiple_ticks_crossed_hit_limit ... ok (gas usage
↪  est.: 39777080)
test ekubo::tests::core_test::locks::test_accumulate_fees_per_liquidity_success ... ok (gas usage est.: 13989964)
test ekubo::tests::extensions_test::test_mock_extension_after_initialize_pool_only ... ok (gas usage est.: 15246310)
test ekubo::tests::core_test::save_load_tests::test_save_load_1_token ... ok (gas usage est.: 5639990)
test ekubo::tests::core_test::locks::test_larger_amount_liquidity_add ... ok (gas usage est.: 10703854)
test ekubo::tests::extensions_test::test_mock_extension_after_swap_only ... ok (gas usage est.: 15025340)
test ekubo::tests::positions_test::test_create_position_in_range_after_swap_no_fees ... ok (gas usage est.: 74902264)
test ekubo::tests::core_test::locks::test_swap_token1_exact_input_against_small_liquidity_no_tick_cross ... ok (gas
↪  usage est.: 20663960)
test ekubo::tests::core_test::locks::test_swap_token0_exact_input_no_liquidity ... ok (gas usage est.: 13018690)
test ekubo::tests::core_test::locks::test_full_range_liquidity_add ... ok (gas usage est.: 10715334)
test ekubo::tests::core_test::locks::test_swap_token0_exact_output_against_small_liquidity_with_tick_cross ... ok (gas
↪  usage est.: 22791848)
test ekubo::tests::core_test::locks::test_swap_token0_exact_output_against_small_liquidity_no_tick_cross ... ok (gas
↪  usage est.: 20663960)
test ekubo::tests::core_test::locks::test_full_range_liquidity_add_and_half_burn ... ok (gas usage est.: 19069076)
test ekubo::tests::core_test::locks::test_swap_token1_exact_output_against_small_liquidity_no_tick_cross ... ok (gas
↪  usage est.: 23164618)
test ekubo::tests::core_test::locks::test_swap_token0_exact_output_no_liquidity ... ok (gas usage est.: 10013604)
test ekubo::tests::core_test::locks::test_swap_token0_zero_amount_zero_liquidity ... ok (gas usage est.: 7547194)
test ekubo::tests::core_test::locks::test_swap_token1_exact_input_no_liquidity ... ok (gas usage est.: 10013604)
test ekubo::tests::core_test::locks::test_swap_token1_exact_output_no_liquidity ... ok (gas usage est.: 12523112)
test ekubo::tests::core_test::locks::test_full_range_liquidity_add_and_full_burn ... ok (gas usage est.: 19292344)
test ekubo::tests::core_test::locks::test_swap_exact_input_token0_multiple_ticks_crossed_hit_limit ... ok (gas usage
↪  est.: 42309206)
test result: ok. 452 passed; 0 failed; 0 ignored; 0 filtered out;
```

10 About Nethermind

Nethermind is a Blockchain Research and Software Engineering company. Our work touches every part of the web3 ecosystem - from layer 1 and layer 2 engineering, cryptography research, and security to application-layer protocol development. We offer strategic support to our institutional and enterprise partners across the blockchain, digital assets, and DeFi sectors, guiding them through all stages of the research and development process, from initial concepts to successful implementation.

We offer security audits of projects built on EVM-compatible chains and Starknet. We are active builders of the Starknet ecosystem, delivering a node implementation, a block explorer, a Solidity-to-Cairo transpiler, and formal verification tooling. Nethermind also provides strategic support to our institutional and enterprise partners in blockchain, digital assets, and decentralized finance (DeFi). In the next paragraphs, we introduce the company in more detail.

Blockchain Security: At Nethermind, we believe security is vital to the health and longevity of the entire Web3 ecosystem. We provide security services related to Smart Contract Audits, Formal Verification, and Real-Time Monitoring. Our Security Team comprises blockchain security experts in each field, often collaborating to produce comprehensive and robust security solutions. The team has a strong academic background, can apply state-of-the-art techniques, and is experienced in analyzing cutting-edge Solidity and Cairo smart contracts, such as ArgentX and StarkGate (the bridge connecting Ethereum and StarkNet). Most team members hold a Ph.D. degree and actively participate in the research community, accounting for 240+ articles published and 1,450+ citations in Google Scholar. The security team adopts customer-oriented and interactive processes where clients are involved in all stages of the work.

Blockchain Core Development: Our core engineering team, consisting of over 20 developers, maintains, improves, and upgrades our flagship product - the Nethermind Ethereum Execution Client. The client has been successfully operating for several years, supporting both the Ethereum Mainnet and its testnets, and now accounts for nearly a quarter of all synced Mainnet nodes. Our unwavering commitment to Ethereum's growth and stability extends to sidechains and layer 2 solutions. Notably, we were the sole execution layer client to facilitate Gnosis Chain's Merge, transitioning from Aura to Proof of Stake (PoS), and we are actively developing a full-node client to bolster Starknet's decentralization efforts. Our core team equips partners with tools for seamless node set-up, using generated docker-compose scripts tailored to their chosen execution client and preferred configurations for various network types.

DevOps and Infrastructure Management: Our infrastructure team ensures our partners' systems operate securely, reliably, and efficiently. We provide infrastructure design, deployment, monitoring, maintenance, and troubleshooting support, allowing you to focus on your core business operations. Boasting extensive expertise in Blockchain as a Service, private blockchain implementations, and node management, our infrastructure and DevOps engineers are proficient with major cloud solution providers and can host applications in-house or on clients' premises. Our global in-house SRE teams offer 24/7 monitoring and alerts for both infrastructure and application levels. We manage over 5,000 public and private validators and maintain nodes on major public blockchains such as Polygon, Gnosis, Solana, Cosmos, Near, Avalanche, Polkadot, Aptos, and StarkWare L2. Sedge is an open-source tool developed by our infrastructure experts, designed to simplify the complex process of setting up a proof-of-stake (PoS) network or chain validator. Sedge generates docker-compose scripts for the entire validator set-up based on the chosen client, making the process easier and quicker while following best practices to avoid downtime and being slashed.

Cryptography Research: At Nethermind, our Cryptography Research team is dedicated to continuous internal research while fostering close collaboration with external partners. The team has expertise across a wide range of domains, including cryptography protocols, consensus design, decentralized identity, verifiable credentials, Sybil resistance, oracles, and credentials, distributed validator technology (DVT), and Zero-knowledge proofs. This diverse skill set, combined with strong collaboration between our engineering teams, enables us to deliver cutting-edge solutions to our partners and clients.

Smart Contract Development & DeFi Research: Our smart contract development and DeFi research team comprises 40+ world-class engineers who collaborate closely with partners to identify needs and work on value-adding projects. The team specializes in Solidity and Cairo development, architecture design, and DeFi solutions, including DEXs, AMMs, structured products, derivatives, and money market protocols, as well as ERC20, 721, and 1155 token design. Our research and data analytics focuses on three key areas: technical due diligence, market research, and DeFi research. Utilizing a data-driven approach, we offer in-depth insights and outlooks on various industry themes.

Our suite of L2 tooling: Warp is Starknet's approach to EVM compatibility. It allows developers to take their Solidity smart contracts and transpile them to Cairo, Starknet's smart contract language. In the short time since its inception, the project has accomplished many achievements, including successfully transpiling Uniswap v3 onto Starknet using Warp.

- **Voyager** is a user-friendly Starknet block explorer that offers comprehensive insights into the Starknet network. With its intuitive interface and powerful features, Voyager allows users to easily search for and examine transactions, addresses, and contract details. As an essential tool for navigating the Starknet ecosystem, Voyager is the go-to solution for users seeking in-depth information and analysis;
- **Horus** is an open-source formal verification tool for StarkNet smart contracts. It simplifies the process of formally verifying Starknet smart contracts, allowing developers to express various assertions about the behavior of their code using a simple assertion language;
- **Juno** is a full-node client implementation for Starknet, drawing on the expertise gained from developing the Nethermind Client. Written in Golang and open-sourced from the outset, Juno verifies the validity of the data received from Starknet by comparing it to proofs retrieved from Ethereum, thus maintaining the integrity and security of the entire ecosystem.

Learn more about us at nethermind.io.

General Advisory to Clients

As auditors, we recommend that any changes or updates made to the audited codebase undergo a re-audit or security review to address potential vulnerabilities or risks introduced by the modifications. By conducting a re-audit or security review of the modified codebase, you can significantly enhance the overall security of your system and reduce the likelihood of exploitation. However, we do not possess the authority or right to impose obligations or restrictions on our clients regarding codebase updates, modifications, or subsequent audits. Accordingly, the decision to seek a re-audit or security review lies solely with you.

Disclaimer

This report is based on the scope of materials and documentation provided by you to Nethermind in order that Nethermind could conduct the security review outlined in **1. Executive Summary** and **2. Audited Files**. The results set out in this report may not be complete nor inclusive of all vulnerabilities. Nethermind has provided the review and this report on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. This report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on this report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, Nethermind disclaims any liability in connection with this report, its content, and any related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. Nethermind does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and Nethermind will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.