Quantstamp Security Assessment Certificate

January 26th 2023 — Quantstamp Verified

Minima (Node Finance)

This audit report was prepared by Quantstamp, the leader in blockchain security.

Executive Summary

Type	Defi
Auditors	Fatemeh Heidari, Auditing Engineer Ibrahim Abouzied, Auditing Engineer Hytham Farah, Auditing Engineer Jonathan Mevs, Auditing Engineer
Timeline	2023-01-17 through 2023-01-31
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	https://docs.router.nodefinance.org/
Documentation Quality	Low
Test Quality	High

Source Code

Repository	Commit
Node-Fi/Minima-Contracts	a3844e9 initial audit
Node-Fi/Minima-Contracts	f936c68 fixes

Total Issues	**19** (16 Resolved)
High Risk Issues	**1** (1 Resolved)
Medium Risk Issues	**4** (3 Resolved)
Low Risk Issues	**9** (8 Resolved)
Informational Risk Issues	**5** (4 Resolved)
Undetermined Risk Issues	0 (0 Resolved)

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
Undetermined	The impact of the issue is uncertain.
Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
Acknowledged	The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
Fixed	Adjusted program implementation, requirements or constraints to eliminate the risk.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

Minima is a DeFi routing protocol that supports routes through different protocols. The data structure used to enable various routing models gives the router implementation some flexible features. This increases the complexity, which increases the need for robust testing. In this regard, the number of provided tests and the test coverage are low and should be increased. During the audit, we found several issues which we recommend the Minima team address. In addition, we strongly recommend adding more test cases, increasing the overall coverage of the tests, and providing more comprehensive documentation with all details about the smart contracts.

ID	Description	Severity	Status
QSP-1	All Transactions Can Be Considered Signed by an Admin	High	Fixed
QSP-2	Weights May Not Sum to 100	Medium	Fixed
QSP-3	Intermediary Tokens May Be Locked in the Contract	Medium	Mitigated
QSP-4	Unchecked Call Return Value	Medium	Fixed
QSP-5	Missing Protection Against Signature Replay Attacks	Medium	Acknowledged
QSP-6	Divisors Can Send Funds to Completed Paths	Low	Fixed
QSP-7	Divisors Can Send Funds to Non-Revelant Paths	Low	Acknowledged
QSP-8	Possibility of Underflow	Low	Fixed
QSP-9	The Contract Owner and Null Tenant Can Drift	Low	Fixed
QSP-10	Ownership Can Be Renounced	Low	Fixed
QSP-11	Missing Input Validation	Low	Fixed
QSP-12	Missing `isContract()` Validation	Low	Fixed
QSP-13	Pairs and Paths Inconsistent Lengths	Low	Fixed
QSP-14	Greedy Contract	Low	Fixed
QSP-15	Admins Have the Power to Change Partner Admins and Partner Fees	Informational	Acknowledged
QSP-16	Outdated Compiler Version	Informational	Fixed
QSP-17	Application Monitoring Can Be Improved by Emitting More Events	Informational	Fixed
QSP-18	Potentially Misleading Event	Informational	Fixed
QSP-19	Not Using Interface	Informational	Fixed

Quantstamp Audit Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

DISCLAIMER:
This audit is restricted to the following files ONLY:

- `src/MinimaRouterV1.sol`

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:

- Slither v0.9.1

Steps taken to run the tools:

1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Findings

QSP-1 All Transactions Can Be Considered Signed by an Admin

Severity: *High Risk*

Status: Fixed

Description: `getPartnerInfo()` checks whether the signer of a transaction is an admin in `adminSigner[]`. If so, the `partnerAdmin[partnerId]` receives the partner fee. Otherwise, it defaults to the null tenant, which is the contract owner.
However, `ecrecover()` returns 0 on error. If `adminSigner[0] == true`, an invalid transaction will be considered to be signed by an admin.

Recommendation: Validate that `adminSigner[0]` cannot be set to true in either the constructor or `setAdminSigner()`. Or, if `ecrecover()` returns zero, assume that the signature was invalid, i.e. change the line to: `if (adminSigner[signer] & signer != 0) {...}`.

QSP-2 Weights May Not Sum to 100

Severity: *Medium Risk*

Status: Fixed

Description: `getDivisorTransferAmounts()` determines how many tokens should be transferred to the `toIdx` of the divisors. There is a constraint that no weight is greater than 100, but there is no confirmation that the sum of the weights is 100. If the sum of the weights is a different number, a disproportionate number of tokens will be transferred.

Recommendation: In `getDivisorTransferAmounts()`, either require the weights to sum to 100 or calculate the sum of the weights and use it as the divisor calculating `transferAmount`.

QSP-3 Intermediary Tokens May Be Locked in the Contract

Severity: *Medium Risk*

Status: Mitigated

Description: After completing a path of swaps, the final token in a path may be left untransferred if its divisor weight is zero. There is no guarantee that the token will be used in a later swap, and it will be left untracked in the `MinimaRouterV1` contract.

Recommendation: Require that all tokens be accounted for in `transferAmounts` from `getDivisorTransferAmounts()`.

QSP-4 Unchecked Call Return Value

Severity: *Medium Risk*

Status: Fixed

Description: The return value of the message call in the functions `recoverAdminFee()`, `swapExactInputForOutput()`, and `disperseWithFee()` are not checked. Execution will resume even if the called contract throws an exception. If the call fails accidentally or an attacker forces the call to fail, this may cause unexpected behaviour in the subsequent program logic. (cf. SWC-104)
As one of the possible scenarios, in the `recoverAdminFee()` function, the authors set the balance of the token to 0 before sending a transfer. Furthermore, the authors only use the `transfer()` method without checking whether the return value is successful.

Recommendation: When using `ERC20` functions which return true on success, handle the possibility that the call will fail by checking the return value or use OpenZeppelins *safe* wrappers, i.e. `safeTransfer()` which revert the execution whenever true is not returned."

QSP-5 Missing Protection Against Signature Replay Attacks

Severity: *Medium Risk*

Status: Acknowledged

Description: It is sometimes necessary to perform signature verification in smart contracts to achieve better usability or to save gas cost. A secure implementation needs to protect against Signature Replay Attacks by for example keeping track of all processed message hashes and only allowing new message hashes to be processed. A malicious user could attack a contract without such a control and get message hash that was sent by another user processed multiple times. (cf. SWC-121)

Recommendation: Include a nonce and the chain ID in the signature.

QSP-6 Divisors Can Send Funds to Completed Paths

Severity: *Low Risk*

Status: Fixed

Description: Divisors are used to allocate the output tokens of a path to the first pair of other paths, indicated by the index of the path array, `toIdx`. However, it is possible for `toIdx` to point to an already processed index. In this case, the tokens may be left unswapped in the `ISwappaPairV1` contract.

Recommendation: Validate that the `toIdx` always points to an unprocessed path.

QSP-7 Divisors Can Send Funds to Non-Revelant Paths

Severity: *Low Risk*

Status: Acknowledged

Description: Divisors allocate the output tokens of a path to the first pair of other paths. However, it is important that the output token sent to the first pair of another path is the first token of that path. otherwise, the tokens may get locked in `ISwappaPairV1` contract.

Recommendation: Validate that the output token of a path which is transferred to another path matches that path's first token.

QSP-8 Possibility of Underflow

Severity: *Low Risk*

Status: Fixed

Description: `swapExactInputForOutput()` and `getDivisorTransferAmounts()` perform unsafe subtractions. If `outputBalanceBefore[token]` is greater than the current balance, an underflow will occur.

Recommendation: Replace the minus operator with the `SafeMath.sub()` function.

QSP-9 The Contract Owner and Null Tenant Can Drift

Severity: *Low Risk*

Status: Fixed

Description: In the constructor, ownership of the contract is immediately transferred to the `admin` address. The `admin` address is also used as the address of the null tenant. If contract ownership is transferred in the future, the contract owner and the null tenant address will be different addresses.

Recommendation: Override `transferOwnership` to set `partnerAdmin[0]` to the new owner address.

QSP-10 Ownership Can Be Renounced

Severity: *Low Risk*

Status: Fixed

Description: If the owner renounces their ownership, all ownable contracts will be left without an owner. Consequently, any function guarded by the `onlyOwner` modifier will no longer be able to be executed.

Recommendation: Double check if this is the intended behavior.

QSP-11 Missing Input Validation

Severity: *Low Risk*

Status: Fixed

Related Issue(s): SWC-123

Description: It is important to validate inputs, even if they only come from trusted addresses, to avoid human error. Specifically, in the following functions arguments of type `address` may be initialized with value `0x0`:

- `recoverAdminFee()` does not check that `token` and `reciever` are different from address(0).
- `constructor()` does not check that `admin` and members of the `initialSigners` array are different from address(0).
- `setAdmin()` does not check that `addr` is different from address(0).
- `setPartnerAdmin()` does not check that `admin` is different from address(0).

In addition, there are no checks on the lengths of the `pairs` and `path` arrays being zero in the `swapExactInputForOutput()` function. The `divisor` values in `getDivisorTransferAmounts()` function are not checked for zero value or overflow situations.
In `swapExactInputForOutput()`, the `expectedOutputAmount` should be greater or equal to `minOutputAmount`.

Recommendation: We recommend adding the relevant checks.

QSP-12 Missing `isContract()` Validation

Severity: *Low Risk*

Status: Fixed

Description: The comment left above the constructor suggests that the `admin` argument should be a multi-sig wallet, and hence, a smart contract, but there is no check to ensure that it is indeed a smart contract and not an EOA.

Recommendation: Include an `isContract()` validation on the `admin` variable to decrease the likelihood of an error.

QSP-13 Pairs and Paths Inconsistent Lengths

Severity: *Low Risk*

Status: Fixed

Description: In `swapExactInputForOutput()` function, while iterating over the pairs assigned to `path[i]`, it is assumed that for any `j < details.pairs[i].length`, there exists `details.path[i][j]` and `details.path[i][j + 1]`. But it is possible that `details.path[i].length` be equal or smaller than `details.pairs[i].length`. This can lead to transaction failure due to accessing out of bound array index.
On the other hand, if the `pairs` is longer than `path` there are indices that are never accessed.

Recommendation: Add a check that `details.pairs[i].length == details.path[i].length - 1`.

QSP-14 Greedy Contract

Severity: *Low Risk*

Status: Fixed

Description: A greedy contract is a contract that can receive ether which can never be redeemed. The contract implements a payable `receive()` function despite not providing any functionality for retrieving ether, leading to potentially locked funds in the contract.

Recommendation: Remove the `receive()` function from the contract.

QSP-15 Admins Have the Power to Change Partner Admins and Partner Fees

Severity: *Informational*

Status: Acknowledged

Description: The `setPartnerAdmin()` and `setPartnerFee()` functions are guarded by the `partnerAuthorized` modifier. The functions are not exclusive to partners as the modifier authorizes any addresses approved as an `adminSigner`. Partners should be aware that admins can modify their partner fee or partner admin (the address that receives the partner fee) at any time.

Recommendation: Administrative powers to change the partner admin addresses and partner fees should either be updated to be exclusive to the partner admin OR be properly communicated to all partners in public documentation.

QSP-16 Outdated Compiler Version

Severity: *Informational*

Status: Fixed

Description: Using an outdated compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version. (cf. SWC-102)

Recommendation: It is recommended to use a recent version of the Solidity compiler, such as `0.8.16`.

QSP-17 Application Monitoring Can Be Improved by Emitting More Events

Severity: *Informational*

Status: Fixed

Description: In order to validate the proper deployment and initialization of the contracts, it is a good practice to emit events. Also, any important state transitions can be logged, which is beneficial for monitoring the contract, and also tracking eventual bugs or hacks. Below we present a non-exhaustive list of events that could be emitted to improve application management:

- `recoverAdminFee()`
- `setAdmin()`
- `setPartnerAdmin()`
- `setPartnerFee()`

Recommendation: Consider emitting the events.

QSP-18 Potentially Misleading Event

Severity: *Informational*

Status: Fixed

Description: Since there is no check to ensure that the new fee is not equal to the old fee, the event `FeeChanged()` may fire when no fee change has occurred.

Recommendation: Check that the new fee does not match the old fee.

QSP-19 Not Using Interface

Severity: *Informational*

Status: Fixed

Description: Several functions that are intended to interact with many implementations of ERC20 tokens use `ERC20` referring to OpenZeppelin's implementation, rather than the generic `IERC20`.
For example, The function `disperseWithFee()` transfers tokens to partners as well as users. Since it is intended to interface with many different ERC20 tokens, each with its own implementation, the transfers should be used with the `IERC20()` wrapper and not the `ERC20()` wrapper.

Recommendation: Change all instances of `ERC20()` to `IERC20()` in the following funcitons:

- `disperseWithFee()`
- `updateBalances()`
- `getDvisorTransferAmounts()`

Automated Analyses

Slither

Version 0.9.1 of Slither was used. There were findings related to ignoring the return value for the transfer function, the greedy contract issue, uninitialized local values, and missing input validation, which all are included in the report.

Code Documentation

- The documentation focused almost entirely on how to use APIs that interact with the protocol's smart contracts but not on the contracts themselves. This gives a feel for what the intended goals are but not any of the mechanisms for achieving this. It would be beneficial for the team to include more detailed documentation of their contracts and the mechanisms behind the functionality.

- Almost no NatSpec is present, and only some loose comments. Consider including NatSpec for each function.
- Clarify the role of the "tenant" which comes up in the comments, but is not present in the docs.

Adherence to Best Practices

1. Use the `ensure` modifier in `swapExactInputForOutput()` rather than the require statement.
2. Rename `getPartnerInfo()` to indicate that it verifies a signature transaction (Ex. `getPartnerIdFromSig()`).
3. Use `recoverSigner()` in `getPartnerInfo()`.
4. In `swapExactInputForOutput()`, check whether `transferAmount > 0` before calling `transfer()`.
5. The given implementation of signature verification using `ecrecover` directly is prone to signature malleability. (cf. SWC-117). Consider using a secure wrapper like OpenZeppelin's ECDSA utility library, which performs additional security checks on the signature parameters.
6. Some variables with identifiers `i`, `j`, and `k` are uninitialized and thus can point to unexpected storage locations in the contract, leading to intentional or unintentional vulnerabilities. (cf. SWC-109). Initialize all the variables. If a variable is meant to be initialized to zero, explicitly set it to zero to improve code readability.
7. Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: " to each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.
8. Emitting more events is recommended in order for improved logging of the state transitions in the contract.
9. It is suggested to implement whitelist checks for ERC20 tokens to prevent malicious tokens from being used in the contract. For example, it was possible to impact/block other addresses by sending them tokens from Tornado Cash.
10. The following imports are declared at the beginning of the `MinimaRouterV1.sol` file but are never used:
    - `./interfaces/IFarm.sol`
    - `./interfaces/INative.sol`
1. Some variables and functions are poorly named ad do not reflect well the functionality:
    . `partnerFees` is a percentage whereas the name suggests it is a fixed quantity
    . `ensure()` does not suggest that it is responsible for checking that the expiry deadline has not been reached.
    . The variable name `output` in the `disperseWithFee()` function is misleading as it is the address of the receiving token and not the quantity of that token.
    . `getPartnerInfo()`
2. On L#93 it is unnecessary to include that `partnerFees[0] = 0` as this is already the default value of an unassigned variable.
3. `recoverSigner()` is not used and can be removed.
4. `ensure()` modifier is not used and can be removed.
5. `MultiSwap()` event is not emitted and can be removed.
6. We generally recommend being wary of limiting the slippage when swapping tokens to ensure that contracts receive adequate funds and to defend against sandwich attacks.

Test Results

Test Suite Results

The test suite was run by calling `forge test -vvv --match-contract MinimaRouterV1Test`. 36/36 tests passed.

```
Running 36 tests for test/router/MinimaRouterV1.t.sol:MinimaRouterV1Test
[PASS] testAdminChangedEvent() (gas: 38612)
[PASS] testAdminIsContract() (gas: 2422956)
[PASS] testAllowsOutputTokenToIdx0(uint8,uint256) (runs: 256, μ: 665415, ~: 5921)
[PASS] testDisperseWFeesTransferFail() (gas: 721685)
[PASS] testGivesExpectedAmountAsMax(uint8,uint256) (runs: 256, μ: 1949815, ~: 22104)
[PASS] testPartnerAdminChanged() (gas: 38551)
[PASS] testPassesInbetweenExpectedAndMinimum(uint8,uint256) (runs: 256, μ: 2095056, ~: 21903)
[PASS] testRecoverAdminFeeEvent() (gas: 32794)
[PASS] testRecoverAdminFeeTransferFail() (gas: 659187)
[PASS] testRenounceOwnership() (gas: 15986)
[PASS] testRevertOnNotEnoughTokens(uint8,uint256) (runs: 256, μ: 2058886, ~: 21925)
[PASS] testRouteFailsWhenDivisorTooHigh(uint8,uint256) (runs: 256, μ: 1094264, ~: 16489)
[PASS] testRouteHandlesDivisor(uint8,uint256,uint8) (runs: 256, μ: 1250328, ~: 22794)
[PASS] testRouteShouldFailOnTransferToCompletedPath(uint8,uint256) (runs: 256, μ: 2384717, ~: 22315)
[PASS] testRouteShouldFailOnTransferToCurrentPath(uint8,uint256) (runs: 256, μ: 1172624, ~: 22064)
[PASS] testRouteShouldFailWithMinOutGreaterThanExpectedOut(uint8,uint256) (runs: 256, μ: 282004, ~: 22086)
[PASS] testRouteShouldFailWithPairLenEq0(uint8,uint256) (runs: 256, μ: 134207, ~: 22109)
[PASS] testRouteShouldFailWithPathPairsMismatch(uint8,uint256) (runs: 256, μ: 1323116, ~: 22128)
[PASS] testRouteSuccessful(uint8,uint256) (runs: 256, μ: 2531040, ~: 186252)
[PASS] testRouteSuccessfulSplit(uint8,uint256) (runs: 256, μ: 2538968, ~: 22064)
[PASS] testSetAdminFailsOnNonAuthorized(address) (runs: 256, μ: 14305, ~: 14305)
[PASS] testSetPartnerFee(uint256,uint256) (runs: 256, μ: 27298, ~: 17762)
[PASS] testSetPartnerFeeSameAsOld() (gas: 40019)
[PASS] testShouldFailOnRandomEthSend() (gas: 17860)
[PASS] testShouldFailWhenDivisorIsTooHigh(uint8,uint8) (runs: 256, μ: 13924, ~: 13924)
[PASS] testShouldFailWhenDivisorIsZero(uint8) (runs: 256, μ: 13675, ~: 13675)
[PASS] testShouldFailWhenDivisorsDoNotSumTo100() (gas: 25609)
[PASS] testShouldFailWhenDivisorsDoNotSumTo100Fuzz(uint8) (runs: 256, μ: 1120726, ~: 536109)
[PASS] testShouldFailWithAdminAs0Address() (gas: 65672)
[PASS] testShouldFailWithInitialSignersAs0Address() (gas: 93313)
[PASS] testShouldFailWithRecoverAdminFeeReceiver0Address() (gas: 16368)
[PASS] testShouldFailWithSetAdminAs0Address() (gas: 14204)
[PASS] testShouldFailWithSetPartnerAdminAs0Address() (gas: 14269)
[PASS] testSwapTransferFailBetweenRoutes(uint8) (runs: 256, μ: 890647, ~: 822165)
[PASS] testTransferOwnership() (gas: 2380246)
[PASS] testgetPartnerIdFromSigReturns0OnInvalidEcRecover() (gas: 25375)
Test result: ok. 36 passed; 0 failed; finished in 2.74s
```

Code Coverage

The code coverage was gathered by running `forge coverage`. The coverage for the contract within scope, `MinimaRouterV1.sol`, shows high coverage and utilizes fuzz testing.

File	% Lines	% Statements	% Branches	% Funcs
src/ MinimaRouterV1.sol	90.20% (92/102)	90.55% (115/127)	80.30% (53/66)	88.24% (15/17)
src/ MinimaRouterV1NativeHandler.sol	0.00% (0/18)	0.00% (0/22)	0.00% (0/8)	0.00% (0/3)
src/interfaces/uniswap/ BitMath.sol	0.00% (0/47)	0.00% (0/49)	0.00% (0/36)	0.00% (0/2)
src/interfaces/uniswap/ FullMath.sol	0.00% (0/31)	0.00% (0/33)	0.00% (0/10)	0.00% (0/2)
src/pairs/ DepositBalancerV2.sol	0.00% (0/32)	0.00% (0/46)	0.00% (0/4)	0.00% (0/3)
src/pairs/ DepositStableSwap.sol	0.00% (0/25)	0.00% (0/39)	0.00% (0/12)	0.00% (0/3)
src/pairs/ DepositUniswapV2.sol	0.00% (0/33)	0.00% (0/49)	0.00% (0/14)	0.00% (0/6)
src/pairs/ PairAToken.sol	0.00% (0/17)	0.00% (0/22)	0.00% (0/12)	0.00% (0/3)
src/pairs/ PairATokenV2.sol	0.00% (0/11)	0.00% (0/13)	0.00% (0/8)	0.00% (0/3)
src/pairs/ PairATokenV3.sol	0.00% (0/11)	0.00% (0/13)	0.00% (0/8)	0.00% (0/3)
src/pairs/ PairBPool.sol	0.00% (0/9)	0.00% (0/13)	0.00% (0/6)	0.00% (0/3)
src/pairs/ PairBalancerWeighted.sol	0.00% (0/42)	0.00% (0/57)	0.00% (0/10)	0.00% (0/7)
src/pairs/ PairCurve.sol	0.00% (0/23)	0.00% (0/34)	0.00% (0/12)	0.00% (0/4)
src/pairs/ PairDepositStableSwap.sol	0.00% (0/23)	0.00% (0/36)	0.00% (0/10)	0.00% (0/5)
src/pairs/ PairGaugeDeposit.sol	0.00% (0/11)	0.00% (0/13)	0.00% (0/10)	0.00% (0/3)
src/pairs/ PairMento.sol	0.00% (0/13)	0.00% (0/19)	0.00% (0/6)	0.00% (0/3)
src/pairs/ PairOpenSumSwap.sol	0.00% (0/10)	0.00% (0/14)	0.00% (0/8)	0.00% (0/3)
src/pairs/ PairRStCelo.sol	0.00% (0/14)	0.00% (0/18)	0.00% (0/10)	0.00% (0/3)
src/pairs/ PairSavingsCELO.sol	0.00% (0/9)	0.00% (0/13)	0.00% (0/6)	0.00% (0/3)
src/pairs/ PairStCelo.sol	0.00% (0/9)	0.00% (0/12)	0.00% (0/6)	0.00% (0/3)
src/pairs/ PairStableSwap.sol	0.00% (0/41)	0.00% (0/60)	0.00% (0/22)	0.00% (0/6)
src/pairs/ PairSymmetricSwap.sol	0.00% (0/9)	0.00% (0/11)	0.00% (0/8)	0.00% (0/3)
src/pairs/ PairUniswapV2.sol	0.00% (0/52)	0.00% (0/81)	0.00% (0/18)	0.00% (0/9)
src/pairs/ PairYearn.sol	0.00% (0/25)	0.00% (0/34)	0.00% (0/14)	0.00% (0/3)
src/utils/ FixedPoint.sol	0.00% (0/35)	0.00% (0/45)	0.00% (0/24)	0.00% (0/9)
src/utils/ LogExpMath.sol	0.00% (0/170)	0.00% (0/177)	0.00% (0/70)	0.00% (0/6)
src/utils/ Math.sol	0.00% (0/2)	0.00% (0/2)	100.00% (0/0)	0.00% (0/2)
src/utils/ MultiCall.sol	0.00% (0/13)	0.00% (0/16)	0.00% (0/2)	0.00% (0/8)
src/utils/ WeightedMath.sol	0.00% (0/7)	0.00% (0/11)	0.00% (0/2)	0.00% (0/2)
test/mock/ MinimaRouterV1External.sol	71.43% (5/7)	71.43% (5/7)	100.00% (0/0)	71.43% (5/7)
test/mock/ MockErc20.sol	100.00% (3/3)	100.00% (3/3)	100.00% (0/0)	100.00% (3/3)
test/mock/ MockFailingErc20.sol	100.00% (2/2)	100.00% (2/2)	100.00% (0/0)	100.00% (1/1)
test/mock/ MockMultisig.sol	100.00% (1/1)	100.00% (1/1)	100.00% (0/0)	100.00% (1/1)
test/mock/ MockPair.sol	70.00% (7/10)	71.43% (10/14)	50.00% (1/2)	66.67% (2/3)
test/utils/ ExtendedDSTest.sol	0.00% (0/47)	0.00% (0/49)	0.00% (0/18)	0.00% (0/6)
test/utils/ Utilities.sol	**0.00% (0/11)**	**0.00% (0/17)**	**100.00% (0/0)**	**0.00% (0/3)**
Total	11.89% (110/925)	11.60% (136/1172)	12.22% (54/442)	17.53% (27/154)

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

```
84fc02b3216f2243a5ca14b12f4020a77535e8c69d290ab09dc872fad4ccd263 ./src/MinimaRouterV1.sol
```

Tests

```
c2bcea2a1c019a53d6ae2041a2e1599327e994788bd120d614d77fd44c744ae8 ./router/MinimaRouterV1.t.sol
```

Changelog

- 2023-01-31 - Initial report
- 2023-02-27 - Fix-Review

About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp's team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over $200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Aave, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication or other making available of the report to you by Quantstamp.

Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites&aspo; owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on any website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any output generated by such software.

Disclaimer

The review and this report are provided on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Quantstamp disclaims all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. You agree that your access and/or use of the report and other results of the review, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. 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. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and may not be represented as such. No third party is entitled to rely on the report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or 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, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. 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.