Quantstamp

Sequence - Marketplace

Executive Summary

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

Type	NFT Marketplace
Timeline	2024-06-25 through 2024-07-02
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Source Code	• 0xsequence/marketplace-contracts #1d2625f
Auditors	• Rabib Islam Senior Auditing Engineer • Roman Rohleder Senior Auditing Engineer • Hytham Farah Auditing Engineer

Documentation quality	High
Test quality	High
Total Findings	1 Fixed: 1
High severity findings	0
Medium severity findings	0
Low severity findings	0
Undetermined severity findings	0
Informational findings	1 Fixed: 1

Summary of Findings

The current codebase functions as an NFT marketplace. Users are able to list NFTs for sale and make offers to purchase NFTs. Both native currency and ERC20s can be used for transactions, although offers cannot be created with native currency. The marketplace also includes support for ERC2981 royalties and additional fees that can be paid to integrators.

No major issues were found during the audit. However, we did find one issue concerning initializers of parent contracts.

The codebase's documentation and test suite are of good quality. However, we do recommend some additional tests in order to cover situations where native currency is used.

Update: The issue was addressed and tests were added to account for those situations where native currency is used.

ID	DESCRIPTION	SEVERITY	STATUS
SEQ-1	Missing Super Initialize	• Informational	Fixed

Assessment Breakdown

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

Disclaimer

Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

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

1. Code review that includes the following
    1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
    2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    3. 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:
    1. 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.
    2. 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, clarity, 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.

Scope

Files Included

Repo: https://github.com/0xsequence/marketplace-contracts/(1d2625f0fa3b59d00d511ecbba7f174101b8f4c0)

Files:

- `contracts/*`

Operational Considerations

1. `SequenceMarket` is an upgradeable contract. As such, the proxy's owner is capable of changing the implementation logic at any time. New logic may introduce new vulnerabilities not discussed in this audit report.
2. In case of listings, makers ultimately pay for the royalty fee.
3. For listings and offers to be seamlessy matched, corresponding ERC-20 tokens have to be pre-approved to this contract. To minimize potential abuse, these approvals should always be minimal and temporary, i.e. only targeted amounts should be approved (never the maximum) and approvals should be revoked if no longer necessary (cancellations, invalidations).
4. Integrators (i.e. UI) may demand a cut in form of additional fees, which would be deducted from listings/offerings, when calling `acceptRequest()` or `acceptRequestBatch()`.
5. Royalty info may be subject to change from the time of a listing to the time of execution. Either the token contract may implement `ERC2981` and a corresponding `royaltyInfo()` function, whose value may fluctuate, or the sequence market fallback royalty info `customRoyalties[]` may be queried, which may be changed by the contract owner through `setRoyaltyInfo()`.
6. Arbitrary contracts can be used for offers. `Request.currency` is allowed to be an arbitrary address. This allows for flexibility as users can make offers using any ERC20 token, but, it also increases the potential for users to use malicious or fraudulent contracts which may harm user's with listings.
7. Checks when making requests may no longer hold during sales. At the time when users make requests, the `SequenceMarket` contract checks that the users have the tokens they are offering and have set the approval. However, between the time of the request and the time of the sale the approval could have been revoked, or the tokens transfered out of the request creator's address. This would result in a reverted transaction if that request is used in `_acceptRequest()`.
8. The contract supports batch operations for creating, accepting, and canceling requests. While this functionality is convenient, it may lead to gas inefficiencies if not used judiciously, especially if the batch size is not optimally sized or if there are many additional fees involved.
9. There is a potential risk of front-running, where an attacker could monitor the blockchain for incoming transactions and submit a competing transaction with a higher gas fee to get it mined first. This could disrupt the intended execution of request acceptances or cancellations.

Key Actors And Their Capabilities

`SequenceMarket` inherits from OpenZeppelin's `OwnableUpgradeable`. Therefore, it has a privileged `owner` account. It has privileged access to the following functions:

1. `setRoyaltyInfo()` – modifies `customRoyalties[tokenContract]`
2. `_authorizeUpgrade()` – empty; overrides the function from parent contract `UUPSUpgradeable`
3. `renounceOwnership()`
4. `transferOwnership()`
5. `upgradeTo()`
6. `upgradeToAndCall()`

A compromised or maliciously acting contract owner could:

1. Change the royalty information for tokens that do not support `ERC2981`, by i.e. changing the receiver to oneself. This may be abused to sweep outstanding or future listing/offer currency transactions.
2. Upgrade the contract to i.e. sweep all approved tokens.

Additionally the contract has 3 other key actors, listing creators/takers, offer creators/takers and potential third party integrators (i.e. UI). While listing and offer creators/takers may create and accept each others requests or cancel their own ones, third party integrators may add an additional cut in form of additional fees when calling `acceptRequest()` or `acceptRequestBatch()`.

Findings

SEQ-1 Missing Super Initialize

Informational · Fixed

Update

Fixed by the client.

Addressed in: `10a94cc03a7ffe0b8543fe8373cb3bb4ddfff174`.

The client provided the following explanation:

> - `Calling ReentrancyGuardUpgradeable's init provides gas optimisation. Added`
> - `OwnableUpgradeable's init sets the owner to the sender, this is unintended behaviour as the owner is set to another. Ignored, added commented out for transparency`
> - `Other inherited init functions are blank. Ignored, added commented out for transparency`

File(s) affected: `SequenceMarket.sol`

Description: Failing to call super initialize functions may at times lead to uninitialized state variables and compromise the contract's integrity (although no such integrity is lost in the current case): the `SequenceMarket` contract does not invoke the following inherited initialization functions:

1. `UUPSUpgradeable.__UUPSUpgradeable_init()`
2. `ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init()`
3. `ReentrancyGuardUpgradeable.__ReentrancyGuard_init()`
4. `OwnableUpgradeable.__Ownable_init()`
5. `ContextUpgradeable.__Context_init()`

Recommendation: Ensure super initialize functions are called to prevent uninitialized state variables.

Auditor Suggestions

SEQ-S-1 Ownership Can Be Renounced

Acknowledged

Update

Marked as "Acknowledged" by the client.

The client provided the following explanation:

> `Ability to revoke is intended behaviour`

File(s) affected: `SequenceMarket.sol`

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: Confirm that this is the intended behavior. If not, override and disable the `renounceOwnership()` function in the affected contracts. For extra security, consider using a two-step process when transferring the ownership of the contract (e.g. `Ownable2Step` from OpenZeppelin).

SEQ-S-2 Application Monitoring Can Be Improved By Emitting More Events

Fixed

Update

Fixed by the client.
Addressed in: `e890c45c3b34809c4af11cb7753cc4f3ac60443e` .
The client provided the following explanation:

> - `SequenceMarket: In all cases we either emit the values suggested or intentionally omit values for efficiency.`
> - `SequenceMarketFactory: Added emit for implementation address`

File(s) affected: `SequenceMarket.sol` , `SequenceMarketFactory.sol`

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:

1. **SequenceMarket**:
    1. `_cancelRequest()` :
        - `_requests`
    2. `invalidateRequests()` :
        - `invalidBeforeId`
    3. `_createRequest()` :
        - `_nextRequestId, _requests`
    4. `_acceptRequest()` :
        - `_requests`
    5. `invalidateRequests()` :
        - `invalidTokenBeforeId`
    6. `setRoyaltyInfo()` :
        - `customRoyalties`
2. **SequenceMarketFactory**:
    1. `constructor()` :
        - `implementation`

Recommendation: Ensure that all state changes emit an event to facilitate tracking and auditing.

SEQ-S-3 Gas Optimization

Fixed

Update

Fixed by the client.
Addressed in: `3df6c742f45c5be72f059bd9f07ed16d737dd5c5` .
The client provided the following explanation:

> `Suggestions added`

File(s) affected: `SequenceMarket.sol`

Description: We identified a few areas where gas usage can be optimized.

1. When looping through a `for` loop, use `unchecked { ++i }` to increment the index. (The `unchecked` block is no longer necessary for this optimization from Solidity 0.8.22 onwards).
2. When looping through an array, cache the length of the array. Opportunities exist at lines 246 and 298.

Recommendation: Consider implementing the above optimizations.

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – 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.

- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **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).

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.

Files

- `056...b06 ./contracts/SequenceMarketFactory.sol`
- `097...90c ./contracts/SequenceMarket.sol`
- `197...96e ./contracts/interfaces/IERC2981.sol`
- `4db...c9c ./contracts/interfaces/ISequenceMarket.sol`
- `c66...6c9 ./contracts/interfaces/IERC721.sol`

Tests

- `1a9...520 ./test/SequenceMarket.t.sol`
- `d91...ffa ./test/Upgradeability.t.sol`
- `7fb...69e ./test/mocks/ERC1155RoyaltyMock.sol`
- `574...af8 ./test/mocks/ERC20TokenMock.sol`
- `bed...eb7 ./test/mocks/ERC721Mock.sol`
- `d18...e9e ./test/mocks/ERC721RoyaltyMock.sol`

Toolset

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

Setup

Tool Setup:

- Slither v0.10.0

Steps taken to run the tools:

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

Automated Analysis

Slither

Slither did not yield any serious issues for this repository.

Test Suite Results

Update: Tests were added that account for situations where native currency is used.

All the repository's tests are passing. However, we do recommend some extra tests for those situations where native currency is being used.

```
Ran 2 tests for test/Upgradeability.t.sol:UpgradeabilityTest
[PASS] test_nonAdminNoUpgrade(address,address) (runs: 256, μ: 5768390, ~: 5768390)
[PASS] test_upgrades(address,address) (runs: 256, μ: 7344533, ~: 7344533)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 620.34ms (266.64ms CPU time)

Ran 66 tests for test/SequenceMarket.t.sol:SequenceMarketTest
```

```
[PASS] test_acceptListing((bool,bool,address,uint256,uint256,uint96,address,uint256),address) (runs: 256,
μ: 293791, ~: 293539)
[PASS] test_acceptListingBatch((bool,bool,address,uint256,uint256,uint96,address,uint256),address[])
(runs: 256, μ: 478951, ~: 480580)
[PASS] test_acceptListingBatch_repeat((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs:
256, μ: 1441752, ~: 1358231)
[PASS]
test_acceptListing_additionalFees((bool,bool,address,uint256,uint256,uint96,address,uint256),uint256[])
(runs: 256, μ: 331735, ~: 330847)
[PASS]
test_acceptListing_invalidAdditionalFees((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 452053, ~: 464227)
[PASS] test_acceptListing_invalidERC721Owner((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 360020, ~: 363434)
[PASS] test_acceptListing_invalidExpiry((bool,bool,address,uint256,uint256,uint96,address,uint256),bool)
(runs: 256, μ: 246321, ~: 247455)
[PASS]
test_acceptListing_invalidQuantity_tooHigh((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 240307, ~: 239414)
[PASS]
test_acceptListing_invalidQuantity_zero((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 240252, ~: 239347)
[PASS] test_acceptListing_invalidRoyalties((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 318888, ~: 289820)
[PASS] test_acceptListing_invalidated((bool,bool,address,uint256,uint256,uint96,address,uint256),
(bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256, μ: 824732, ~: 783993)
[PASS] test_acceptListing_invalidatedToken((bool,bool,address,uint256,uint256,uint96,address,uint256),
(bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256, μ: 664464, ~: 617267)
[PASS] test_acceptListing_noFunds((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256,
μ: 284009, ~: 285396)
[PASS]
test_acceptListing_notEnoughNative((bool,bool,address,uint256,uint256,uint96,address,uint256),address)
(runs: 256, μ: 242924, ~: 242097)
[PASS] test_acceptListing_reentry((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256,
μ: 703638, ~: 705021)
[PASS] test_acceptListing_twice((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256,
μ: 315862, ~: 319151)
[PASS] test_acceptListing_twice_overQuantity((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 300375, ~: 302123)
[PASS] test_acceptOffer((bool,bool,address,uint256,uint256,uint96,address,uint256),address) (runs: 256,
μ: 293227, ~: 293288)
[PASS] test_acceptOfferBatch((bool,bool,address,uint256,uint256,uint96,address,uint256),address[]) (runs:
256, μ: 483957, ~: 485432)
[PASS] test_acceptOfferBatch_repeat((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs:
256, μ: 1329677, ~: 1243889)
[PASS]
test_acceptOffer_additionalFees((bool,bool,address,uint256,uint256,uint96,address,uint256),uint256[])
(runs: 256, μ: 332560, ~: 332083)
[PASS] test_acceptOffer_invalidAdditionalFees((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 429435, ~: 441720)
[PASS] test_acceptOffer_invalidERC721Owner((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 354515, ~: 358512)
[PASS] test_acceptOffer_invalidExpiry((bool,bool,address,uint256,uint256,uint96,address,uint256),bool)
(runs: 256, μ: 251244, ~: 251318)
[PASS]
test_acceptOffer_invalidQuantity_tooHigh((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 247181, ~: 247229)
[PASS] test_acceptOffer_invalidQuantity_zero((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 246881, ~: 246898)
[PASS] test_acceptOffer_invalidRoyalties((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 332856, ~: 303252)
[PASS] test_acceptOffer_invalidated((bool,bool,address,uint256,uint256,uint96,address,uint256),
(bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256, μ: 823145, ~: 785533)
[PASS] test_acceptOffer_invalidatedToken((bool,bool,address,uint256,uint256,uint96,address,uint256),
(bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256, μ: 669580, ~: 620150)
[PASS] test_acceptOffer_noFunds((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256,
μ: 280711, ~: 280725)
[PASS] test_acceptOffer_twice((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256, μ:
318280, ~: 319929)
[PASS] test_acceptOffer_twice_overQuantity((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 298316, ~: 299974)
[PASS] test_acceptRequestBatch_fixed() (gas: 778949)
```

```
[PASS] test_acceptRequestBatch_fuzz(uint8,(bool,bool,address,uint256,uint256,uint96,address,uint256)
[],uint8[]) (runs: 256, μ: 889280, ~: 991072)
[PASS] test_acceptRequestBatch_invalidLengths(uint8,
(bool,bool,address,uint256,uint256,uint96,address,uint256)[],uint256[],address[]) (runs: 256, μ: 830877,
~: 917819)
[PASS] test_cancelListing((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256, μ:
206881, ~: 207827)
[PASS] test_cancelListing_partialFill((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs:
256, μ: 298975, ~: 302608)
[PASS] test_cancelOffer((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256, μ:
211383, ~: 211371)
[PASS] test_cancelOffer_partialFill((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs:
256, μ: 295332, ~: 299515)
[PASS] test_cancelRequestBatch(uint8,(bool,bool,address,uint256,uint256,uint96,address,uint256)[]) (runs:
256, μ: 580880, ~: 688265)
[PASS] test_cancelRequestBatch_invalidCaller(uint8,
(bool,bool,address,uint256,uint256,uint96,address,uint256)[]) (runs: 256, μ: 1029804, ~: 1043085)
[PASS]
test_createListing_erc1155_invalidApproval((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 41288, ~: 41155)
[PASS]
test_createListing_erc1155_noToken((bool,bool,address,uint256,uint256,uint96,address,uint256),uint256)
(runs: 256, μ: 36548, ~: 36425)
[PASS]
test_createListing_erc721_invalidApproval((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 44739, ~: 44621)
[PASS]
test_createListing_erc721_noToken((bool,bool,address,uint256,uint256,uint96,address,uint256),uint256)
(runs: 256, μ: 36588, ~: 35946)
[PASS]
test_createListing_invalidExpiry((bool,bool,address,uint256,uint256,uint96,address,uint256),uint96)
(runs: 256, μ: 27586, ~: 27572)
[PASS] test_createListing_invalidPrice((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs:
256, μ: 27581, ~: 27644)
[PASS]
test_createListing_invalidQuantity_erc1155((bool,bool,address,uint256,uint256,uint96,address,uint256),uin
t256) (runs: 256, μ: 36494, ~: 36485)
[PASS]
test_createListing_invalidQuantity_erc721((bool,bool,address,uint256,uint256,uint96,address,uint256),uint
256) (runs: 256, μ: 38409, ~: 38352)
[PASS]
test_createListing_invalidToken((bool,bool,address,uint256,uint256,uint96,address,uint256),address)
(runs: 256, μ: 36271, ~: 36286)
[PASS] test_createOffer_invalidApproval((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 52292, ~: 52586)
[PASS] test_createOffer_invalidExpiry((bool,bool,address,uint256,uint256,uint96,address,uint256),uint96)
(runs: 256, μ: 28514, ~: 28492)
[PASS] test_createOffer_invalidPrice((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs:
256, μ: 27396, ~: 27442)
[PASS] test_createOffer_invalidQuantity((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 45685, ~: 45707)
[PASS] test_createRequestBatch(uint8,(bool,bool,address,uint256,uint256,uint96,address,uint256)[]) (runs:
256, μ: 717432, ~: 810987)
[PASS]
test_createRequest_interfaceInvalid((bool,bool,address,uint256,uint256,uint96,address,uint256),address)
(runs: 256, μ: 33416, ~: 33461)
[PASS] test_getRoyaltyInfo_defaultZero() (gas: 915655)
[PASS] test_getRoyaltyInfo_notOverridden(bool,uint96,address) (runs: 256, μ: 55938, ~: 56010)
[PASS] test_getRoyaltyInfo_overridden(bool,uint96,address) (runs: 256, μ: 1110806, ~: 1257604)
[PASS] test_isRequestValid_bulk(uint8,(bool,bool,address,uint256,uint256,uint96,address,uint256)
[],bool[]) (runs: 256, μ: 643887, ~: 720497)
[PASS] test_isRequestValid_expired() (gas: 768435)
[PASS] test_isRequestValid_invalidApproval() (gas: 790191)
[PASS] test_isRequestValid_invalidBalance() (gas: 867329)
[PASS] test_isRequestValid_partialValidity() (gas: 323857)
[PASS] test_isRequestValid_royaltyInvalid() (gas: 310415)
[PASS] test_setRoyaltyInfo_invalidCaller(address,address,uint96,address) (runs: 256, μ: 17848, ~: 17848)
Suite result: ok. 66 passed; 0 failed; 0 skipped; finished in 1.04s (6.76s CPU time)

Ran 2 test suites in 1.07s (1.66s CPU time): 68 tests passed, 0 failed, 0 skipped (68 total tests)
```

Code Coverage

Update: Coverage has improved further.

Coverage is high for the `SequenceMarket` contract.

File	% Lines	% Statements	% Branches	% Funcs
contracts/SequenceMarket.sol	95.93% (**165/**172)	95.17% (**197/**207)	90.28% (**65/**72)	100.00% (**23/**23)
contracts/SequenceMarketFactory.sol	55.56% (**5/**9)	52.94% (**9/**17)	100.00% (**0/**0)	66.67% (**2/**3)
test/SequenceMarket.t.sol	60.00% (**6/**10)	55.56% (**5/**9)	50.00% (**1/**2)	66.67% (**2/**3)
test/Upgradeability.t.sol	100.00% (**5/**5)	100.00% (**5/**5)	100.00% (**0/**0)	100.00% (**2/**2)
test/mocks/ERC1155RoyaltyMock.sol	58.33% (**7/**12)	58.82% (**10/**17)	50.00% (**2/**4)	66.67% (**4/**6)
test/mocks/ERC721Mock.sol	100.00% (**3/**3)	100.00% (**5/**5)	100.00% (**0/**0)	100.00% (**1/**1)
test/mocks/ERC721RoyaltyMock.sol	58.33% (**7/**12)	58.82% (**10/**17)	50.00% (**2/**4)	66.67% (**4/**6)
Total	88.79% (**198/**223)	87.00% (**241/**277)	85.37% (**70/**82)	86.36% (**38/**44)

Changelog

- 2024-07-02 - Initial report
- 2024-07-05 - Final report

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, 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

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.