Quantstamp

Sequence

Executive Summary

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

Type	NFT Marketplace
Timeline	2023-12-12 through 2023-12-20
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	README.md Public Documentation
Source Code	• 0xsequence/marketplace-contracts #f0d3522 • 0xsequence/contracts-library #4cbdf11 • 0xsequence/wallet-contracts #af0ad80
Auditors	• Guillermo Escobero Auditing Engineer • Jennifer Wu Auditing Engineer • Cameron Biniamow Auditing Engineer

Documentation quality	Medium
Test quality	Medium
Total Findings	23 Fixed: 15 Acknowledged: 7 Mitigated: 1
High severity findings	0
Medium severity findings	2 Fixed: 2
Low severity findings	13 Fixed: 8 Acknowledged: 4 Mitigated: 1
Undetermined severity findings	1 Acknowledged: 1
Informational findings	7 Fixed: 5 Acknowledged: 2

Summary of Findings

Quantstamp performed a security audit of the smart contracts implementing the Sequence NFT orderbook based on the code present in the listed repositories. In addition, specific contracts of two more repositories were audited: `contracts-library` and `wallet-contracts`.

Sequence offers solutions to clients building web3 gaming apps. In its marketplace, users can trade ERC-721 and ERC-1155 tokens, creating listings and offers, and using ERC-20 tokens as payment.

Users can use Sequence smart contracts to create ERC-20, ERC-721, and ERC-1155 tokens with extended functionalities (privileged roles), as well as contracts implementing public sales for them. Users of these contracts should be aware of the ability of the owners to upgrade them at any time, changing the implementation.

Sequence also offers a smart wallet (out of the scope of this audit). In this audit, a recovery mechanism for the wallet was audited (`Trust.sol`).

All issues and recommendations are discussed in the *Findings* section of this document. After that, recommendations about documentation are discussed. We strongly recommend addressing all the issues and adding tests to cover the proposed fixes before deployment.

The documentation quality is good. Public and internal documentation was provided by the Sequence team. However, it is recommended to add detailed and updated public documentation focusing on the new features of the protocol.

Regarding testing, all tests passed, and the project implements code coverage metrics except on one repository (`contracts-library`), failing due to a `StackTooDeep` error. We highly recommend improving the branch coverage to a minimum of `95%` and adding new tests to cover the proposed fixes.

Fix review: The Sequence team has either fixed, mitigated, or acknowledged all issues found within the report, and provided new commits containing fixes for the issues found.

The Sequence team added the ability to set custom royalties when trading ERC-721 and ERC-1155 tokens that do not implement ERC-2981. The `OrderBook` contract owner can set the fee percentage (up to 100%) and the recipient for each token contract.

ID	DESCRIPTION	SEVERITY	STATUS
SEQ-1	Potential Signature Replayability Across Chains	Medium	Fixed
SEQ-2	Token Sale Owners Can Front-Run Mints With Increased Cost	Medium	Fixed
SEQ-3	Order Taker Forced Into Paying Increased Royalty Fees	Low	Acknowledged
SEQ-4	Create ERC-20 Offers in an Orderbook Designed for ERC-1155 and ERC-721	Low	Fixed
SEQ-5	Use of Non-Standard Tokens	Low	Acknowledged
SEQ-6	Potential EVM Compatibility Issue in Solidity Version `>=0.8.20`	Low	Fixed
SEQ-7	Insufficient Allowance Check in NFT Order Creations	Low	Fixed
SEQ-8	Inconsistency in Order ID Retrieval for Partially Fulfilled NFT Orders	Low	Fixed
SEQ-9	Risk of NFT Loss when Accepting Order	Low	Fixed
SEQ-10	Missing Input Validation	Low	Mitigated
SEQ-11	Privileged Users Limited to One ERC-1155 Token Mint When Only the Global Sale is Enabled	Low	Fixed
SEQ-12	Ownership / Roles Can Be Rennounced	Low	Acknowledged
SEQ-13	Upgradability	Low	Acknowledged
SEQ-14	Greedy Contracts	Low	Fixed
SEQ-15	Collisions In `abi.encodePacked()`	Low	Fixed
SEQ-16	Missing Events to Signal State Changes	Informational	Acknowledged
SEQ-17	Gas Optimization: Use `calldata` Instead of `memory`	Informational	Fixed
SEQ-18	Gas Optimization: Cache Variables	Informational	Fixed
SEQ-19	Unlocked Pragma	Informational	Acknowledged
SEQ-20	Outdated Comments	Informational	Fixed
SEQ-21	Deprecated Functions	Informational	Fixed
SEQ-22	'Dead' Code	Informational	Fixed
SEQ-23	User Can Create Several Offers Without Giving Allowance	Undetermined	Acknowledged

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.
>
> 1. All wallet features (except `Trust.sol` and `TrustFactory.sol`) are out of the scope of this audit.
> 2. `0xsequence/erc-1155` repository is out of the scope of this audit.

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(f0d352260da928e54406fd41c5821ea644b22de2)
    - Files: `contracts/*`
- Repo: https://github.com/0xsequence/contracts-library(4cbdf112e0baae7f906dead63eea5e4fc19c438c)
    - Files: `src/tokens/*`
- Repo: https://github.com/0xsequence/wallet-contracts(af0ad80c291d469c9ab30216a3bd3b8fe4ccee2f)
    - Files: `Trust.sol`, `TrustFactory.sol`

Findings

SEQ-1 Potential Signature Replayability Across Chains

Medium Fixed

> **Update**
>
> The Sequence team fixed the `ERC1155BaseToken` contract by removing the `ERC1155Meta` inheritance, which removes the ability to conduct meta transactions.
>
> The Sequence team fixed the `Trust` contract by hashing the chain ID along with the signature data. Depending on the flag passed along with the signature, the expected chain ID can be zero or `block.chainid`.
>
> Addressed in: `9f97d103fc8cce74d9c6c658ff913bb5de9ff4ac`, and `b32f88b298fc73ec1c4be5e845185de91f41c9ff`.

File(s) affected: `ERC1155BaseToken.sol`, `Trust.sol`

Description: The lack of chain ID integration in contract's message hashing could pose risks in scenarios involving multiple blockchain networks.

1. The `ERC1155BaseToken` contract inherits `ERC1155Meta` (out of scope) which includes functions `metaSafeTransferFrom()` and `metaSafeBatchTransferFrom()`. The functions enable assets to be transferred from the owner to a recipient, provided there is a valid signature from the asset owner authorizing the transfer. The transfer functions use the `hashEIP712Message()` for message hashing. However, this function does not include a chain ID in its implementation. The absence of a chain ID in the EIP-712 message structure could lead to a situation where a valid signature on one blockchain might be replayed on another blockchain.
2. Although the `Trust` contract checks the validity of the signed message in `isValidSignature()`, the hashed value excludes a chain ID, and, therefore, the signed message may be reused across different Ethereum chains.

Typically, the inclusion of a chain ID in the signed message is a common security measure to prevent such replay attacks, ensuring that signatures are valid only on the intended blockchain.

Recommendation:

1. Override the `hashEIP712Message()` function in the `ERC1155Meta` contract to include the chain ID. This modification will link each signature to a specific blockchain, which helps to address the potential issue of signature replayability across different chains.
2. Include the chain ID in the hashed message to prevent replay attacks.

SEQ-2
Token Sale Owners Can Front-Run Mints With Increased Cost

Medium | Fixed

Update

Fixed by adding two new parameters to the `mint()` function: `expectedPaymentToken` and `maxTotal`. If the owner frontruns the minting transaction and sets a higher price or a different token, the transaction will revert.

Addressed in: `4096dbe47346fd17d1e658b95aa82bb6b7faa91a`.

File(s) affected: `ERC1155Sale.sol`, `ERC721Sale.sol`

Description: The `mint()` function in the contracts `ERC721Sale` and `ERC1155Sale`, is vulnerable to front running from the `MINT_ADMIN_ROLE`. If the `MINT_ADMIN_ROLE` finds a mint transaction in the mempool, and the mint caller has given sufficient approval, the `MINT_ADMIN_ROLE` can front-run the mint with a call to `setSaleDetails()` and increase the `cost`. Thus, the mint caller will pay more for the token than expected.

Exploit Scenario: Assume Bob deploys his own `ERC721Items` contract and a corresponding `ERC721Sale` contract via the respective factories. Bob initially sets the sale price at 10 USDC and sets the Merkel root to `0x0`, which allows anyone to buy NFTs.

1. Alice wants to buy one of Bob's NFTs through the `ERC721Sale` contract.
2. Alice gives maximum approval to the `ERC721Sale` contract.
3. Alice calls `ERC721Sale.mint()` expecting to pay 10 USDC.
4. Bob finds Alice's transaction in the mempool.
5. Bob notices that Alice has 1,000 USDC in her wallet.
6. Bob front runs Alice's transaction and increases the `cost` to 1,000 USDC by calling `ERC721Sale.setSaleDetails()`.
7. Alice pays 1,000 USDC for Bob's NFT rather than the expected 10 USDC.

Recommendation: Add an argument to the `mint()` function where the caller can set a maximum payment amount. The transaction should revert if the maximum payment amount is exceeded during the mint.

SEQ-3 Order Taker Forced Into Paying Increased Royalty Fees

Low | Acknowledged

Update

Acknowledged by the Sequence team:

> *"This is an acknowledged risk. When interacting with a token contract, the user has a trust assumption with the token contract owner. In this case, the token contract owner is responsible for communicating and responsibly setting royalty changes with the token holder."*

File(s) affected: `Orderbook.sol`

Description: When a user wants to purchase a given ERC-721 or ERC-1155 token, they can create an order that states the token to purchase and the price they will pay (offer). This is done by creating an order where `isListing` is set to `false`. Then, the owner of the ERC-721 or ERC-1155 token (the maker) can fill the order, if desirable, by calling `acceptOrder()`. When the order is filled, the taker (the order creator) needs to pay the price they stated in their order and the royalty fee accrued from the swap. Therefore, if the order maker also owns the ERC-721 or ERC-1155 contract, they could increase the royalty fee rate and then accept the order. Thus, the order taker will pay the cost of the purchased token and the increased royalty fee.

Exploit Scenario: Assume Bob creates an ERC-721 contract that implements ERC-2981 and has an updatable royalty fee rate. Initially, the royalty fee rate is set at 5%.

1. Bob mints token ID `1` to himself from his ERC-721 contract.
2. Assume that Alice has 1,000 USDC in her wallet and gives maximum approval to the `OrderBook` contract.

3. Alice creates a taker order for Bob's token, where the payment amount is 100 USDC.
4. Bob increases his token's royalty fee rate from 5% to 100%.
5. Bob accepts Alice's order.
6. Alice pays 100 USDC for Bob's token and another 100 USDC for the royalty fee charged during the swap.

Recommendation: Allow taker order creators to specify a maximum total amount they are willing to pay between the order cost and the royalty fee.

SEQ-4
Create ERC-20 Offers in an Orderbook Designed for ERC-1155 and ERC-721

Low Fixed

Update

The interface ID is checked for the token contracts when an order is created. The maker token must have an ERC-1155 or ERC-721 interface ID (ERC-165).

Addressed in: `d34f6c569102b9e53cd6cf06d0d67ff969604034`.

File(s) affected: `Orderbook.sol`

Description: The orderbook, designed for handling ERC-1155 and ERC-721 tokens, is vulnerable to unintended listings of ERC-20 tokens due to the identical `transferFrom()` function signature shared between ERC-20 and ERC-721 standards. In both standards, the function signature is `transferFrom(address, address, uint256)`. However, in the context of ERC-20, the third parameter (`uint256`) represents the amount of tokens to be transferred, while in ERC-721, it denotes the unique `tokenId` of the NFT. This similarity in function structure allows for the possibility of ERC-20 tokens being listed in the orderbook as if they were ERC-721 tokens, with the `tokenId` inadvertently interpreted as the quantity of ERC-20 tokens. This misalignment can lead to transactional errors and confusion, as the orderbook is not designed to accommodate the fungible nature and transactional semantics of ERC-20 tokens. Although unlikely, if a user were to unintentionally accept what they believe is an ERC-721 order, and they have a pre-existing ERC-20 token allowance set for the orderbook, this could lead to an unintentional transfer of ERC-20 tokens to the creator of the spoofed ERC-721 order.

Although there is an approval check in `_hasApprovedTokens` when creating the spoofed ERC-721 order, the ownership validation occurs only when the order is a listing.

Recommendation: This risk can be addressed by implementing one or more of the following measures based on the desired approach and system requirements:

1. **Enhanced Token Standard Checks**: Introduce stricter validation logic to accurately identify the token standard of each listed order. Utilize functionalities like `supportsInterface` to differentiate between ERC-721, ERC-1155, and ERC-20 tokens.
2. **User Interface Alerts**: If the platform includes a user interface, consider adding alerts or warnings to inform users when they are interacting with orders that might not be standard ERC-1155 or ERC-721 listings.
3. **Developer Guidance**: Update developer documentation to highlight this potential issue. Provide guidelines on implementing safeguards in the contract logic to prevent such scenarios.

SEQ-5 Use of Non-Standard Tokens

Low Acknowledged

Update

Documentation updated in: `e2cf3704e5f4bf3f6f0b894d2017192c274dda41`. Acknowledged by the Sequence team:

> *"Updated documentation"*

File(s) affected: `Orderbook.sol`

Description: `Orderbook` orders accept any address as an ERC-20 token. This token will be used in the offer/listing to pay for the ERC-721 or ERC-1155 tokens. However, special ERC-20 tokens that implement changes in the amount transferred (such as on-transfer fee tokens) can lead to potential issues when accepting the orders (not receiving the expected amount of fees or royalties, or a user not receiving the expected payment). Malicious users can also create orders with tokens implementing transfer functions with malicious code.

Recommendation: Confirm if this is expected, or if these tokens are expected to be supported in the system. A possible approach is to implement an allowlist system to accept only ERC-20 tokens approved by the Sequence team in advance.

SEQ-6 Potential EVM Compatibility Issue in Solidity Version `>=0.8.20`

Low Fixed

Update

The Sequence team locks the contract files to 0.8.19 for `OrderBook.sol`.

Addressed in: `a55ce698803e34120fd577c5f9fb34c3c4cf216a` .

Description: According to the `foundry.toml` config file, the `Orderbook` contract is compiled in Solidity version `0.8.20` with an unspecified evm version. As a result, the contract bytecode may be compiled with the default evm version Shanghai and may include `PUSH0` opcodes, which are not supported on certain EVM chains, including BSC, Arbitrum, and Optimism. The deployment or execution of the contract may fail due to the use of an invalid opcode.

Recommendation: Consider explicitly setting the EVM version as Paris in the config file to prevent the compiler from generating `PUSH0` opcodes in the bytecode. Alternatively, consider using a Solidity version of 0.8.19.

SEQ-7 Insufficient Allowance Check in NFT Order Creations

Low Fixed

Update

The royalty amount is now included in the allowance check.

Addressed in: `e27dbdbcd23d392ff6777fbd4b65b06ffc480774` and `30b0968e2106141ed80df29acdb3c98b7ea6f45b` .

File(s) affected: `Orderbook.sol`

Description: The NFT purchase process checks the buyer's token allowance based only on the token price and quantity, neglecting the additional royalty fees. This approach could result in situations where a buyer's approved allowance satisfies the token cost but falls short when royalty fees are included, leading to unsuccessful order fulfillment.

`isOrderValid()` (for an order of "offer" type) verifies that the order creator gave enough allowance to cover the pay of the offer and also the royalties linked to the token.

However, when a user creates an offer, `_createOrder()` verifies that enough allowance was provided to cover the new order, without taking royalties into account.

Recommendation: Confirm this is the intended behavior. If not, consider if it is necessary to include royalties and fees. When validating the order in `_createOrder()`, all fees should be included in the `total`. This will ensure that the buyer's allowance adequately covers both the token cost and the royalty. Frontend or other systems may show incorrect or unexpected order states.

SEQ-8 Inconsistency in Order ID Retrieval for Partially Fulfilled NFT Orders

Low Fixed

Update

Order IDs are no longer created by hashing order details. Instead, order IDs are now sequential.

Addressed in: `a91ceb32ed0270d8b8e67320944482a353839024` , `fe23bd2c0c1e5f18d5683af1aec3a6898e22beb0` .

The Sequence team provided the following explanation:

> *"Note fe23bd2c0c1e5f18d5683af1aec3a6898e22beb0 removes the hash entirely and replaces order ids with uint256 sequential ids"*

File(s) affected: `Orderbook.sol`

Description: The `hashOrder()` function generates an order ID by hashing order data, including `uint256 quantity` . However, for orders that are partially fulfilled, the `quantity` within an order changes over time with each partial fulfillment. As a result, the `hashOrder()` function cannot consistently calculate the correct `orderId` based on the current `_orders` state variable since varying quantities lead to different hash outputs and, consequently, different IDs.

In the protocol, operations such as accepting or cancelling orders use only the `orderId` to reference the order. This issue is specifically relevant for off-chain components or other protocols that use `hashOrder()` to reference or retrieve the order ID based on order data. These systems may end up with mismatched or incorrect order IDs due to the changing quantities in orders over time.

Recommendation: Document and communicate to all users about how the `hashOrder()` function operates. Specifically, explain that the order ID generated at the time of order creation reflects the initial quantity and may not accurately represent the state of partially fulfilled orders.

SEQ-9 Risk of NFT Loss when Accepting Order

Low Fixed

Update

Addressed in: `95afb46f2fb25ba7d9150526710f08e1d674fb2a` .

File(s) affected: `Orderbook.sol`

Description: When accepting orders, the transferring ERC-721 tokens uses the `transferFrom()` function, which does not include safety checks to verify if the recipient is capable of handling ERC-721 tokens. This could lead to irreversible loss of NFTs if they are sent to contracts that do not implement the ERC-721 receiver interface.

Recommendation: We recommend using `safeTransferFrom()` for ERC-721 token transfers to incorporate necessary safety checks, ensuring recipient contract compliance and preventing accidental NFT loss.

SEQ-10 Missing Input Validation

Low Mitigated

Update

Mitigated in: `f6d4dd7879068205bb1957cdd1d819beddca6f54`, `9cf014863c7391ee53ee2d2b1e9b602baa981c8b`, and `2a9894eddbd69bcbbfb72b683b62b919327bbb50`.

The Sequence team provided the following explanation:

1. *Zero cost could be intentional. No change*
2. *Time values validated*
3. *Time values validated*
4. *tokenIds and amounts length is covered in the ERC1155 implementation. No change*
5. *ERC20 transfer function in standard says "Transfers of 0 values MUST be treated as normal transfers and fire the Transfer event.". Mint 0 could be intentional. No change*
6. *^ No change*
7. *SEQ-7 covers this*
8. *Updated*
9. *address(0) is an allowed value. No change*

File(s) affected: `ERC1155Sale.sol`, `ERC20BaseToken.sol`, `ERC20Items.sol`, `ERC721Sale.sol`, `Orderbook.sol`, `Trust.sol`

Related Issue(s): SWC-123

Description: It is crucial to validate inputs, even if they come from trusted addresses, to avoid human error. A lack of robust input validation can only increase the likelihood and impact in the event of mistakes.

Following is the list of places that can potentially benefit from stricter input validation:

1. Acknowledged `ERC721Sale.sol`: the `cost` of the `setSaleDetails()` should not be zero.
2. Fixed `ERC721Sale.sol`: the `endTime` of the `setSaleDetails()` should not be zero.
3. Fixed `ERC721Sale.sol`: `setSaleDetails()` should validate that `startTime` is less than `endTime`.
4. Acknowledged `ERC1155Sale.sol`: validate that `tokenIds` and `amounts` have the same length in `mint()`.
5. Acknowledged `ERC20BaseToken.sol`: the `amount` of the `burn()` should not be zero. It is possible to burn zero amount of ERC-20 tokens with the current implementation.
6. Acknowledged `ERC20Items.sol`: the `amount` of the `mint()` should not be zero. It is possible to mint zero ERC-20 tokens with the current implementation.
7. Fixed `Orderbook._createOrder()` should validate that `request.tokenContract` and `request.currency` are not the zero address.
8. Fixed `Orderbook.isOrderValidBatch()` should validate that `orderIds` and `quantities` are of equal length.
9. Acknowledged `Trust.constructor()` should validate that `owner` and `beneficiary` are not set to the zero address.

Recommendation: Add the validations and checks listed in the description.

SEQ-11 Privileged Users Limited to One ERC-1155 Token Mint When Only the Global Sale is Enabled

Low Fixed

Update

The Merkle proof is now checked after the `for` loop. Thus, multiple tokens can be minted.

Addressed in: `46c079bdabb9969aeec0e9f5b7994a6e04e52cd4`, `850bf0b3273d5cf31b2548e7f58beaacf18f8052`.

The Sequence team provided the following explanation:

> *"Additional collision prevention added by including tokenid in merkle leaf hash"*

File(s) affected: `ERC1155Sale.sol`

Description: The function `ERC1155Sale._payForActiveMint()`, called via `mint()`, collects the mint fees and checks if the caller is a valid leaf in the Merkle tree. If the caller is a valid leaf, the Merkle root is marked as used for the caller. However, since `_payForActiveMint()` allows for batch minting, and since the Merkle proof is verified for each minted token, minting will fail after one token per caller when only the global sale is enabled. This is not an issue when individual token sales are enabled and Merkle roots are unique for each token.

Recommendation: Consider verifying the Merkle proof once before the `for` loop in `_payForActiveMint()` if it is intended for users to mint multiple tokens. Additional tests should be added to cover cases with more than one token ID.

SEQ-12 Ownership / Roles Can Be Rennounced

Low | Acknowledged

> **Update**
> Acknowledged by the Sequence team:
>
> *The ability to relinquish control is an intentional feature. Single step ownership transfer misuse is an acceptable risk.*

Description: If the owner renounces their ownership, all ownable contracts will be left without an owner. As a result, any function guarded by the `onlyOwner` modifier will no longer be able to be executed, such as upgrading the `UpgradeableBeacon` implementation contract in `SequenceProxyFactory`.

Furthermore, critical role transfers do not follow a two-step pattern. The two-step pattern ensures that there is an additional layer of verification and authorization before transferring critical roles, reducing the risk of unauthorized access or malicious actions.

This also applies to contracts inheriting from `AccessControl`. Renouncing `DEFAULT_ADMIN_ROLE` will leave the contract without an administrator.

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-13 Upgradability

Low | Acknowledged

> **Update**
> Acknowledged by the Sequence team:
>
> *"By design"*

Description: While upgradability is not a vulnerability in itself, contract users should be aware that the contracts can be upgraded at any given time by their owner. This audit does not guarantee the behavior of future contracts that the token may be upgraded to.

The Sequence team and contract owners should test the new implementation before upgrading to avoid issues such as function selector collisions or storage slot collisions.

Recommendation: The fact that the contracts can be upgraded by the Sequence team (beacon implementation), or by the contract owner (a custom implementation can bypass beacon implementation) and reasons for future upgrades should be communicated to users beforehand.

SEQ-14 Greedy Contracts

Low | Fixed

> **Update**
> Mints will fail if the payment token is not ETH and the `msg.value` is non-zero.
>
> Addressed in: `f015d8aa81bbcc9e0ae21ef924c115390e174c8c`.

File(s) affected: `ERC1155Sale.sol`, `ERC721Sale.sol`

Description: Sale contracts can be configured to accept payments in ERC-20 or native tokens. However, the functions are `payable`, so a user can send native tokens by mistake, locking them in the contract.

Recommendation: If the token address is not zero, it means that the payment is expected in ERC-20. Revert the transaction if `msg.value` is not zero.

SEQ-15 Collisions In `abi.encodePacked()`

Low | Fixed

> **Update**

Fixed by using `abi.encode()` instead of `abi.encodePacked()`.

Addressed in: `a292a88c001991db660cfe51ed83a867297f9d1b`.

File(s) affected: `ERC1155ItemsFactory.sol`, `ERC20ItemsFactory.sol`, `ERC721ItemsFactory.sol`

Description: Factory contracts call `abi.encodePacked()` to concatenate the parameters of the contract before hashing the result. This is the salt used by `CREATE2` to generate the new contract.

However, `abi.encodePacked()` does not pad the dynamic types, potentially leading to hash collisions (e.g. `[token, $token]`, will have the same hash as `[token$t, oken]`).

Recommendation: Do not use more than one dynamic type in `abi.encodePacked()`. It is recommended to use `abi.encode()` instead.

SEQ-16 Missing Events to Signal State Changes

Informational | Acknowledged

Update

Acknowledged by the Sequence team:

> *"Metadata changes may be made without a URL update. We expect project owners to call for a metadata refresh on our API endpoints when required. This includes when URIs are updated on chain."*

File(s) affected: `ERC20Items.sol`, `ERC721BaseToken.sol`, `ERC1155BaseToken.sol`

Description: Events are important for signalling state changes in a contract. This helps debug the contract in the event of any attacks or critical bugs and helps indicate to users any configuration changes in the contract.

A non-exhaustive list of functions where events can be useful includes:

1. `ERC20Items.setNameAndSymbol()`
2. `ERC721BaseToken.setNameAndSymbol()`
3. `ERC721BaseToken.setBaseMetadataURI()`
4. `ERC721BaseToken.setContractURI()`
5. `ERC1155BaseToken. setBaseMetadataURI()`
6. `ERC1155BaseToken.setContractName()`
7. `ERC1155BaseToken.setContractURI()`

Recommendation: Add events to signal the contract's state changes.

SEQ-17 Gas Optimization: Use `calldata` Instead of `memory`

Informational | Fixed

Update

Addressed in: `c9c77fa42b93b6737d564b64a037c5e5c8ffd217`.

File(s) affected: `Orderbook.sol`

Description: Solidity's `calldata` is a read-only byte-addressable space where function arguments reside. It is more cost-effective to employ `calldata` over `memory`. This is because `calldata` is not stored in memory but is directly accessed from the function call data, resulting in gas savings. It is recommended to change the `memory` to `calldata` in the mentioned functions:

1. `createOrder()`
2. `createOrderBatch()`
3. `_createOrder()`
4. `acceptOrder()`
5. `_acceptOrder()`
6. `acceptOrderBatch()`
7. `getOrderBatch()`
8. `cancelOrderBatch()`
9. `isOrderValidBatch()`

Recommendation: Consider using `calldata` instead of `memory`.

SEQ-18 Gas Optimization: Cache Variables

Informational | Fixed

Update

Addressed in: `ba69fa74c21f3e8ff6904cae57ee3c5ac36a38a2`.

File(s) affected: `Orderbook.sol`

Description: Repeatedly accessing certain variables can be gas-intensive. To optimize gas usage, values frequently accessed should be stored in memory variables. Consider storing the value in a memory variable and reference the memory variable for the following variables:

1. `requests.length` from `Orderbook.createOrderBatch()`
2. `ordersIds.length` from `Orderbook.acceptOrderBatch()`
3. `ordersIds.length` from `Orderbook.getOrderBatch()`
4. `ordersIds.length` from `Orderbook.isOrderValidBatch()`

Recommendation: Consider applying variable caching as per the recommendation provided in the issue.

SEQ-19 Unlocked Pragma

Informational | Acknowledged

Update

Acknowledged by the Sequence team:

> *"We want to allow 3rd parties to integrate with our package. These contracts should use unlocked versions to support projects that require later solidity features when building on top of these implementations."*

Related Issue(s): SWC-103

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.8.*`. The caret (`^`) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version and above, hence the term "unlocked".

The pragma versions used throughout the codebase should not be unlocked and set to the same version to ensure that they are all compiled by the same compiler version.

Recommendation: This issue should be addressed taking SEQ-6 into account. For consistency and to prevent unexpected behavior in the future, we recommend removing the caret to lock the file onto a specific Solidity version (e.g. `pragma solidity 0.8.19`).

SEQ-20 Outdated Comments

Informational | Fixed

Update

Addressed in: `7816f5c03e06c610ffdf2e1fc44f06595417f64f`.

File(s) affected: `ERC1155ItemsFactory.sol`, `ERC1155SaleFactory.sol`, `ERC20ItemsFactory.sol`, `ERC721ItemsFactory.sol`, `ERC721SaleFactory.sol`, `IERC1155ItemsFactory.sol`, `IERC1155SaleFactory.sol`, `IERC20ItemsFactory.sol`, `IERC721ItemsFactory.sol`, `IERC721SaleFactory.sol`

Description: In several factory contracts, the following comment can be found:

```
@dev As `proxyOwner` owns the proxy, it will be unable to call the [...] functions.
```

However, the Sequence factories implement a modified Transparent Proxy pattern, where the administrator can also call the implementation, apart from the administration functions.

Recommendation: Edit the comment to match the implemented behavior.

SEQ-21 Deprecated Functions

Informational | Fixed

Update

Addressed in: `c8952877d77aa0db86304fb3a2bf4ce06fb7b1e4`.

File(s) affected: `ERC1155BaseToken.sol`, `ERC1155Items.sol`, `ERC1155Sale.sol`, `ERC20BaseToken.sol`, `ERC20Items.sol`, `ERC721BaseToken.sol`, `ERC721Items.sol`, `ERC721Sale.sol`

Description: Several initializers in the codebase call `_setupRole()` to assign privileged roles to certain addresses, instead of using `_grantRole()`.

Both functions have the same effect, as `_setupRole()` just calls `_grantRole()`. `_setupRole()` will be deprecated in `v5.0.0` of OpenZeppelin contracts, and its use is not recommended in the current version used by the codebase (`v4.9.3`). See `_setupRole()` comment:

```
NOTE: This function is deprecated in favor of {_grantRole}.
```

Recommendation: Replace `_setupRole()` with `_grantRole()`.

SEQ-22 'Dead' Code

Informational | Fixed

> **Update**
> Addressed in: `119d30aad72ed1de3c9aefa0daff6eaae248a9a8`.

File(s) affected: `ERC1155Sale.sol`, `ERC721Sale.sol`

Related Issue(s): SWC-131, SWC-135

Description: "Dead" code refers to code which is never executed and hence makes no impact on the final result of running a program. Dead code raises a concern, since either the code is unnecessary or the necessary code's results were ignored.

1. `ERC1155Sale._ERC20_TRANSFERFROM_SELECTOR` is never used in `ERC1155Sale`.
2. `ERC721Sale._ERC20_TRANSFERFROM_SELECTOR` is never used in `ERC721Sale`.

Recommendation: Remove the variables if not needed.

SEQ-23
User Can Create Several Offers Without Giving Allowance

Undetermined | Acknowledged

> **Update**
> Acknowledged by the Sequence team:
>
> *"Intentional. No change."*

File(s) affected: `Orderbook.sol`

Description: When a user creates an offer, `_createOrder()` verifies that enough allowance was provided to cover the new order. However, he can create more offers as long as their total amount is equal or less than the allowance of the contract.

`isOrderValid()` can be affected as well, as it verifies each order independently.

Recommendation: Confirm this is the intended behavior. If not, consider if it is necessary to include royalties and fees in the process. Frontend or other systems may show incorrect or unexpected order states.

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).

Adherence to Specification

1. Fixed The READMEs for the contracts in the `contract-library` repository should reference `AccessControlEnumerable` instead of `AccessControl`.

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

- `f3b...3b4 ./src/tokens/ERC721/utility/sale/ERC721Sale.sol`
- `2bb...8d3 ./src/tokens/ERC20/ERC20BaseToken.sol`
- `ea8...2a4 ./src/tokens/ERC1155/utility/sale/ERC1155Sale.sol`
- `790...ac6 ./src/tokens/common/ERC2981Controlled.sol`
- `978...d38 ./src/tokens/common/WithdrawControlled.sol`
- `3bf...214 ./src/tokens/ERC1155/ERC1155BaseToken.sol`
- `cf9...fa3 ./src/tokens/ERC1155/utility/sale/ERC1155SaleFactory.sol`
- `560...06b ./src/tokens/ERC1155/utility/sale/IERC1155Sale.sol`
- `1ce...83d ./src/tokens/ERC1155/utility/sale/IERC1155SaleFactory.sol`
- `e4d...6c0 ./src/tokens/ERC1155/presets/items/ERC1155Items.sol`
- `3b1...682 ./src/tokens/ERC1155/presets/items/IERC1155Items.sol`
- `a7f...0e8 ./src/tokens/ERC1155/presets/items/IERC1155ItemsFactory.sol`
- `816...aa5 ./src/tokens/ERC1155/presets/items/ERC1155ItemsFactory.sol`
- `356...4ed ./src/tokens/ERC1155/extensions/supply/IERC1155Supply.sol`
- `bf1...562 ./src/tokens/ERC1155/extensions/supply/ERC1155Supply.sol`
- `3df...65b ./src/tokens/ERC721/ERC721BaseToken.sol`
- `f75...6c5 ./src/tokens/ERC721/utility/minter/ERC721PermissiveMinter.sol`
- `878...991 ./src/tokens/ERC721/utility/sale/IERC721Sale.sol`
- `138...d7f ./src/tokens/ERC721/utility/sale/ERC721SaleFactory.sol`
- `1c4...e3b ./src/tokens/ERC721/utility/sale/IERC721SaleFactory.sol`
- `a9f...c10 ./src/tokens/ERC721/presets/items/IERC721ItemsFactory.sol`
- `4e4...4ce ./src/tokens/ERC721/presets/items/ERC721Items.sol`
- `44f...7be ./src/tokens/ERC721/presets/items/IERC721Items.sol`
- `b77...398 ./src/tokens/ERC721/presets/items/ERC721ItemsFactory.sol`
- `ceb...7c2 ./src/tokens/common/IERC2981Controlled.sol`
- `436...a1b ./src/tokens/common/MerkleProofSingleUse.sol`
- `784...148 ./src/tokens/common/IMerkleProofSingleUse.sol`
- `b2c...e00 ./src/tokens/common/IWithdrawControlled.sol`
- `bd4...b62 ./src/tokens/ERC20/presets/items/ERC20Items.sol`
- `ce3...034 ./src/tokens/ERC20/presets/items/ERC20ItemsFactory.sol`
- `da3...c17 ./src/tokens/ERC20/presets/items/IERC20Items.sol`
- `034...444 ./src/tokens/ERC20/presets/items/IERC20ItemsFactory.sol`
- `a70...88d ./src/proxies/SequenceProxyFactory.sol`
- `f03...95d ./src/proxies/TransparentUpgradeableBeaconProxy.sol`
- `d23...255 ./src/proxies/openzeppelin/TransparentUpgradeableProxy.sol`
- `382...b02 ./src/proxies/openzeppelin/ERC1967Proxy.sol`
- `351...aad ./src/proxies/openzeppelin/BeaconProxy.sol`
- `cde...078 ./src/utils/StorageSlot.sol`
- `065...df7 ./contracts/Orderbook.sol`
- `ac7...122 ./contracts/interfaces/IERC2981.sol`
- `058...a2e ./contracts/interfaces/IERC721.sol`
- `251...1e7 ./contracts/interfaces/IOrderbook.sol`

- `cf3...cc7 ./contracts/trust/Trust.sol`
- `333...8af ./contracts/trust/TrustFactory.sol`

Tests

- `766...e6e ./foundry_test/trust/Trust.t.sol`
- `9ec...037 ./foundry_test/trust/TrustFactory.t.sol`
- `dc0...55f ./test/TestHelper.sol`
- `e3d...196 ./test/tokens/ERC1155/utility/ERC1155Sale.t.sol`
- `6e0...290 ./test/tokens/ERC1155/presets/ERC1155Items.t.sol`
- `86e...d00 ./test/tokens/ERC721/utility/ERC721Sale.t.sol`
- `8db...5b6 ./test/tokens/ERC721/presets/ERC721Items.t.sol`
- `311...c0f ./test/tokens/ERC20/ERC20Items.t.sol`
- `aca...3e4 ./test/proxies/SequenceProxyFactory.t.sol`
- `dd2...484 ./test/Orderbook.t.sol`
- `59a...021 ./test/mocks/ERC1155RoyaltyMock.sol`
- `b8e...754 ./test/mocks/ERC20TokenMock.sol`
- `3a2...13c ./test/mocks/ERC721Mock.sol`
- `2b9...626 ./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 was used to get a static analysis of the repository. All the issues and recommendations are discussed in this report or classified as false positives.

Test Suite Results

All tests passed.

Fix review update: Some tests were added to cover the fixes for the issues found. All tests passed.

```
marketplace-contracts
```

```
Running 61 tests for test/Orderbook.t.sol:OrderbookTest
[PASS] test_acceptListing((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256, μ:
280438, ~: 279277)
[PASS] test_acceptListingBatch((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256, μ:
431554, ~: 434280)
[PASS] test_acceptListingBatch_repeat((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs:
256, μ: 1393779, ~: 1300426)
[PASS]
test_acceptListing_additionalFees((bool,bool,address,uint256,uint256,uint96,address,uint256),uint256[])
(runs: 256, μ: 321988, ~: 321855)
[PASS]
test_acceptListing_invalidAdditionalFees((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 333001, ~: 342474)
```

```
[PASS] test_acceptListing_invalidERC721Owner((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 274047, ~: 278140)
[PASS] test_acceptListing_invalidExpiry((bool,bool,address,uint256,uint256,uint96,address,uint256),bool)
(runs: 256, μ: 235216, ~: 234337)
[PASS]
test_acceptListing_invalidQuantity_tooHigh((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 233841, ~: 232806)
[PASS]
test_acceptListing_invalidQuantity_zero((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 233884, ~: 232940)
[PASS] test_acceptListing_invalidRoyalties((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 302403, ~: 276042)
[PASS] test_acceptListing_noFunds((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256,
μ: 219835, ~: 220876)
[PASS] test_acceptListing_reentry((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256,
μ: 688103, ~: 690056)
[PASS] test_acceptListing_twice((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256,
μ: 305170, ~: 307716)
[PASS] test_acceptListing_twice_overQuantity((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 284725, ~: 285944)
[PASS] test_acceptOffer((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256, μ:
282676, ~: 281822)
[PASS] test_acceptOfferBatch((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256, μ:
432140, ~: 436204)
[PASS] test_acceptOfferBatch_repeat((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs:
256, μ: 1489865, ~: 1502072)
[PASS]
test_acceptOffer_additionalFees((bool,bool,address,uint256,uint256,uint96,address,uint256),uint256[])
(runs: 256, μ: 322967, ~: 323571)
[PASS] test_acceptOffer_invalidAdditionalFees((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 311727, ~: 324345)
[PASS] test_acceptOffer_invalidERC721Owner((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 272261, ~: 275632)
[PASS] test_acceptOffer_invalidExpiry((bool,bool,address,uint256,uint256,uint96,address,uint256),bool)
(runs: 256, μ: 240400, ~: 240445)
[PASS]
test_acceptOffer_invalidQuantity_tooHigh((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 240812, ~: 240874)
[PASS] test_acceptOffer_invalidQuantity_zero((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 240506, ~: 240589)
[PASS] test_acceptOffer_invalidRoyalties((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 317298, ~: 289704)
[PASS] test_acceptOffer_noFunds((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256,
μ: 217218, ~: 217194)
[PASS] test_acceptOffer_twice((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256, μ:
305870, ~: 308818)
[PASS] test_acceptOffer_twice_overQuantity((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 287226, ~: 288606)
[PASS] test_acceptOrderBatch_fixed() (gas: 755962)
[PASS] test_acceptOrderBatch_fuzz(uint8,(bool,bool,address,uint256,uint256,uint96,address,uint256)
[],uint8[]) (runs: 256, μ: 876287, ~: 963692)
[PASS] test_acceptOrderBatch_invalidLengths(uint8,
(bool,bool,address,uint256,uint256,uint96,address,uint256)[],uint256[]) (runs: 256, μ: 780828, ~: 868630)
[PASS] test_cancelListing((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256, μ:
200993, ~: 201985)
[PASS] test_cancelListing_partialFill((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs:
256, μ: 288095, ~: 291956)
[PASS] test_cancelOffer((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs: 256, μ:
205622, ~: 205602)
[PASS] test_cancelOffer_partialFill((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs:
256, μ: 285447, ~: 289203)
[PASS] test_cancelOrderBatch(uint8,(bool,bool,address,uint256,uint256,uint96,address,uint256)[]) (runs:
256, μ: 592729, ~: 683526)
[PASS] test_cancelOrderBatch_invalidCaller(uint8,
(bool,bool,address,uint256,uint256,uint96,address,uint256)[]) (runs: 256, μ: 912417, ~: 928458)
[PASS]
test_createListing_erc1155_invalidApproval((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 36147, ~: 36011)
[PASS]
test_createListing_erc1155_noToken((bool,bool,address,uint256,uint256,uint96,address,uint256),uint256)
(runs: 256, μ: 31354, ~: 31234)
[PASS]
```

```
test_createListing_erc721_invalidApproval((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 39602, ~: 39427)
[PASS]
test_createListing_erc721_noToken((bool,bool,address,uint256,uint256,uint96,address,uint256),uint256)
(runs: 256, μ: 31317, ~: 30603)
[PASS]
test_createListing_invalidExpiry((bool,bool,address,uint256,uint256,uint96,address,uint256),uint96)
(runs: 256, μ: 22377, ~: 22331)
[PASS] test_createListing_invalidPrice((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs:
256, μ: 22525, ~: 22554)
[PASS]
test_createListing_invalidQuantity_erc1155((bool,bool,address,uint256,uint256,uint96,address,uint256),uin
t256) (runs: 256, μ: 31328, ~: 31385)
[PASS]
test_createListing_invalidQuantity_erc721((bool,bool,address,uint256,uint256,uint96,address,uint256),uint
256) (runs: 256, μ: 33232, ~: 33197)
[PASS]
test_createListing_invalidToken((bool,bool,address,uint256,uint256,uint96,address,uint256),address)
(runs: 256, μ: 29294, ~: 29325)
[PASS] test_createOffer_invalidApproval((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 47263, ~: 47573)
[PASS] test_createOffer_invalidExpiry((bool,bool,address,uint256,uint256,uint96,address,uint256),uint96)
(runs: 256, μ: 23476, ~: 23412)
[PASS] test_createOffer_invalidPrice((bool,bool,address,uint256,uint256,uint96,address,uint256)) (runs:
256, μ: 22284, ~: 22283)
[PASS] test_createOffer_invalidQuantity((bool,bool,address,uint256,uint256,uint96,address,uint256))
(runs: 256, μ: 40558, ~: 40613)
[PASS] test_createOrderBatch(uint8,(bool,bool,address,uint256,uint256,uint96,address,uint256)[]) (runs:
256, μ: 743931, ~: 839374)
[PASS]
test_createOrder_interfaceInvalid((bool,bool,address,uint256,uint256,uint96,address,uint256),address)
(runs: 256, μ: 27159, ~: 27204)
[PASS] test_getRoyaltyInfo_defaultZero() (gas: 910718)
[PASS] test_getRoyaltyInfo_notOverridden(bool,uint96,address) (runs: 256, μ: 50562, ~: 50625)
[PASS] test_getRoyaltyInfo_overridden(bool,uint96,address) (runs: 256, μ: 1096992, ~: 1096951)
[PASS] test_isOrderValid_bulk(uint8,(bool,bool,address,uint256,uint256,uint96,address,uint256)[],bool[])
(runs: 256, μ: 636251, ~: 707558)
[PASS] test_isOrderValid_expired() (gas: 746536)
[PASS] test_isOrderValid_invalidApproval() (gas: 766961)
[PASS] test_isOrderValid_invalidBalance() (gas: 845201)
[PASS] test_isOrderValid_partialValidity() (gas: 301794)
[PASS] test_isOrderValid_royaltyInvalid() (gas: 290963)
[PASS] test_setRoyaltyInfo_invalidCaller(address,address,uint96,address) (runs: 256, μ: 12933, ~: 12933)
Test result: ok. 61 passed; 0 failed; 0 skipped; finished in 1.24s

Ran 1 test suites: 61 tests passed, 0 failed, 0 skipped (61 total tests)
```

```
contracts-library
```

```
Running 10 tests for test/tokens/ERC20/ERC20Items.t.sol:ERC20ItemsTest
[PASS] testBurnInsufficient(address,uint256,uint256) (runs: 1024, μ: 78497, ~: 81996)
[PASS] testBurnSuccess(address,uint256,uint256) (runs: 1024, μ: 89209, ~: 89822)
[PASS] testInitValues() (gas: 35219)
[PASS] testMintInvalidRole(address,uint256) (runs: 1024, μ: 67271, ~: 67271)
[PASS] testMintOwner(uint256) (runs: 1024, μ: 80213, ~: 80213)
[PASS] testMintWithRole(address,uint256) (runs: 1024, μ: 164033, ~: 164033)
[PASS] testOwnerHasRoles() (gas: 32554)
[PASS] testReinitializeFails() (gas: 26957)
[PASS] testSelectorCollision() (gas: 465)
[PASS] testSupportsInterface() (gas: 27843)
Test result: ok. 10 passed; 0 failed; 0 skipped; finished in 705.07ms

Running 7 tests for test/proxies/SequenceProxyFactory.t.sol:SequenceProxyFactoryTest
[PASS] testAddressCompute() (gas: 1018386)
[PASS] testDeployProxy() (gas: 1018464)
[PASS] testDeployProxyAfterUpgrade() (gas: 1030417)
[PASS] testDuplicateDeploysFail() (gas: 89373934605167613 85)
```

```
[PASS] testProxyOwnerUpgrade() (gas: 1047736)
[PASS] testProxyOwnerUpgradeUnaffectedByBeaconUpgrades() (gas: 1062503)
[PASS] testUpgradeAfterDeploy() (gas: 1032986)
Test result: ok. 7 passed; 0 failed; 0 skipped; finished in 1.04s

Running 5 tests for test/tokens/ERC1155/utility/sale/ERC1155SaleBase.t.sol:ERC1155SaleTest
[PASS] testSelectorCollision() (gas: 443)
[PASS] testSupportsInterface() (gas: 7678)
[PASS] testWithdrawERC20(bool,address,uint256) (runs: 1024, μ: 3055255, ~: 414849)
[PASS] testWithdrawETH(bool,address,uint256) (runs: 1024, μ: 2757433, ~: 46218)
[PASS] testWithdrawFail(bool,address,uint256) (runs: 1024, μ: 3119523, ~: 5798014)
Test result: ok. 5 passed; 0 failed; 0 skipped; finished in 1.14s

Running 5 tests for test/tokens/ERC721/utility/sale/ERC721SaleBase.t.sol:ERC721SaleTest
[PASS] testSelectorCollision() (gas: 443)
[PASS] testSupportsInterface() (gas: 7669)
[PASS] testWithdrawERC20(bool,address,uint256) (runs: 1024, μ: 3048817, ~: 5478856)
[PASS] testWithdrawETH(bool,address,uint256) (runs: 1024, μ: 2672256, ~: 5081824)
[PASS] testWithdrawFail(bool,address,uint256) (runs: 1024, μ: 2875038, ~: 419890)
Test result: ok. 5 passed; 0 failed; 0 skipped; finished in 1.15s

Running 14 tests for test/tokens/ERC721/utility/sale/ERC721SaleMint.t.sol:ERC721SaleTest
[PASS] testERC20Mint(bool,address,uint256) (runs: 1024, μ: 3354997, ~: 5708619)
[PASS] testERC20MintFailMaxTotal(bool,address,uint256) (runs: 1024, μ: 3226879, ~: 5604652)
[PASS] testERC20MintFailPaidETH(bool,address,uint256) (runs: 1024, μ: 3027868, ~: 526511)
[PASS] testETHMintFailMaxTotal(bool,address,uint256) (runs: 1024, μ: 2558151, ~: 86398)
[PASS] testFreeMint(bool,address,uint256) (runs: 1024, μ: 2650957, ~: 158739)
[PASS] testMerkleFailBadProof(address[],address) (runs: 1024, μ: 285561, ~: 279625)
[PASS] testMerkleFailNoProof(address[],address) (runs: 1024, μ: 278370, ~: 274580)
[PASS] testMerkleReuseFail(address[],uint256) (runs: 1024, μ: 386753, ~: 383018)
[PASS] testMerkleSuccess(address[],uint256) (runs: 1024, μ: 377848, ~: 373862)
[PASS] testMintExpiredFail(bool,address,uint256,uint64,uint64) (runs: 1024, μ: 2674973, ~: 5154664)
[PASS] testMintFailWrongPaymentToken(bool,address,uint256,address) (runs: 1024, μ: 3183884, ~: 5561657)
[PASS] testMintInactiveFail(bool,address,uint256) (runs: 1024, μ: 2531363, ~: 39587)
[PASS] testMintSuccess(bool,address,uint256) (runs: 1024, μ: 2685066, ~: 191674)
[PASS] testMintSupplyExceeded(bool,address,uint256,uint256) (runs: 1024, μ: 2622858, ~: 131387)
Test result: ok. 14 passed; 0 failed; 0 skipped; finished in 5.43s

Running 22 tests for test/tokens/ERC721/presets/ERC721Items.t.sol:ERC721ItemsTest
[PASS] testBurnBatchInvalidOwnership(address) (runs: 1024, μ: 112748, ~: 112751)
[PASS] testBurnBatchSuccess(address) (runs: 1024, μ: 174651, ~: 174651)
[PASS] testBurnInvalidOwnership(address) (runs: 1024, μ: 109883, ~: 109883)
[PASS] testBurnSuccess(address) (runs: 1024, μ: 138249, ~: 138249)
[PASS] testContractURI() (gas: 85654)
[PASS] testDefaultRoyalty(address,uint96,uint256) (runs: 1024, μ: 40646, ~: 40646)
[PASS] testMetadataInvalid(address) (runs: 1024, μ: 69182, ~: 69182)
[PASS] testMetadataOwner() (gas: 129558)
[PASS] testMetadataWithRole(address) (runs: 1024, μ: 117447, ~: 117447)
[PASS] testMintInvalidRole(address) (runs: 1024, μ: 68294, ~: 68294)
[PASS] testMintMultiple() (gas: 115998)
[PASS] testMintOwner() (gas: 102677)
[PASS] testMintWithRole(address) (runs: 1024, μ: 186766, ~: 186766)
[PASS] testNameAndSymbol() (gas: 89948)
[PASS] testOwnerHasRoles() (gas: 43752)
[PASS] testReinitializeFails() (gas: 28498)
[PASS] testRoyaltyInvalidRole(address,uint256,address,uint96,uint256) (runs: 1024, μ: 125729, ~: 125729)
[PASS] testRoyaltyWithRole(address,uint256,address,uint96,uint256) (runs: 1024, μ: 152677, ~: 152807)
[PASS] testSelectorCollision() (gas: 575)
[PASS] testSupportsInterface() (gas: 33520)
[PASS] testTokenMetadata() (gas: 174058)
[PASS] testTokenRoyalty(uint256,address,uint96,uint256) (runs: 1024, μ: 63902, ~: 63902)
Test result: ok. 22 passed; 0 failed; 0 skipped; finished in 5.43s

Running 21 tests for test/tokens/ERC1155/presets/ERC1155Items.t.sol:ERC1155ItemsTest
[PASS] testBatchMintOwner(uint256[],uint256[]) (runs: 1024, μ: 570123, ~: 584323)
[PASS] testBatchMintWithRole(address,uint256[],uint256[]) (runs: 1024, μ: 646560, ~: 668538)
[PASS] testBurnBatchInvalidOwnership(address,uint256[],uint256[]) (runs: 1024, μ: 228100, ~: 232530)
[PASS] testBurnBatchSuccess(address,uint256[],uint256[],uint256[]) (runs: 1024, μ: 197696, ~: 196772)
[PASS] testBurnInvalidOwnership(address,uint256,uint256,uint256) (runs: 1024, μ: 106208, ~: 110814)
[PASS] testBurnSuccess(address,uint256,uint256,uint256) (runs: 1024, μ: 123981, ~: 125019)
[PASS] testContractURI() (gas: 82911)
[PASS] testDefaultRoyalty(address,uint96,uint256) (runs: 1024, μ: 40669, ~: 40669)
```

```
[PASS] testMetadataInvalid(address) (runs: 1024, μ: 66739, ~: 66742)
[PASS] testMetadataOwner() (gas: 43707)
[PASS] testMetadataWithRole(address) (runs: 1024, μ: 117380, ~: 117380)
[PASS] testMintInvalidRole(address) (runs: 1024, μ: 122412, ~: 122412)
[PASS] testMintOwner(uint256,uint256) (runs: 1024, μ: 104456, ~: 104456)
[PASS] testMintWithRole(address,uint256,uint256) (runs: 1024, μ: 185877, ~: 188093)
[PASS] testOwnerHasRoles() (gas: 43385)
[PASS] testReinitializeFails() (gas: 27933)
[PASS] testRoyaltyInvalidRole(address,uint256,address,uint96,uint256) (runs: 1024, μ: 121042, ~: 121042)
[PASS] testRoyaltyWithRole(address,uint256,address,uint96,uint256) (runs: 1024, μ: 152697, ~: 152829)
[PASS] testSelectorCollision() (gas: 575)
[PASS] testSupportsInterface() (gas: 30852)
[PASS] testTokenRoyalty(uint256,address,uint96,uint256) (runs: 1024, μ: 63840, ~: 63840)
Test result: ok. 21 passed; 0 failed; 0 skipped; finished in 5.66s

Running 21 tests for test/tokens/ERC1155/utility/sale/ERC1155SaleMint.t.sol:ERC1155SaleTest
[PASS] testERC20Mint(bool,address,uint256,uint256) (runs: 1024, μ: 3307164, ~: 669348)
[PASS] testERC20MintFailPaidETH(bool,address,uint256,uint256) (runs: 1024, μ: 3279401, ~: 577355)
[PASS] testFreeGlobalMint(bool,address,uint256,uint256) (runs: 1024, μ: 2695299, ~: 158484)
[PASS] testFreeTokenMint(bool,address,uint256,uint256) (runs: 1024, μ: 3014109, ~: 5652699)
[PASS] testMerkleFailBadProof(address[],address,uint256,bool) (runs: 1024, μ: 282153, ~: 288118)
[PASS] testMerkleFailNoProof(address[],address,uint256,bool) (runs: 1024, μ: 284115, ~: 285734)
[PASS] testMerkleReuseFail(address[],uint256,uint256,bool) (runs: 1024, μ: 410694, ~: 418779)
[PASS] testMerkleSuccess(address[],uint256,uint256,bool) (runs: 1024, μ: 399057, ~: 398445)
[PASS] testMerkleSuccessGlobalMultiple(address[],uint256,uint256[]) (runs: 1024, μ: 462594, ~: 461037)
[PASS] testMintExpiredGlobalFail(bool,address,uint256,uint256,uint64,uint64) (runs: 1024, μ: 2865870, ~:
5550449)
[PASS] testMintExpiredSingleFail(bool,address,uint256,uint256,uint64,uint64) (runs: 1024, μ: 2826810, ~:
5548673)
[PASS] testMintFailMaxTotal(bool,address,uint256,uint256) (runs: 1024, μ: 2981360, ~: 5636966)
[PASS] testMintFailWrongPaymentToken(bool,address,uint256,uint256,address) (runs: 1024, μ: 3367767, ~:
6000530)
[PASS] testMintGlobalSuccess(bool,address,uint256,uint256) (runs: 1024, μ: 2756014, ~: 187290)
[PASS] testMintGlobalSupplyExceeded(bool,address,uint256,uint256,uint256) (runs: 1024, μ: 2851836, ~:
2851838)
[PASS] testMintGroupSuccess(bool,address,uint256,uint256) (runs: 1024, μ: 3036877, ~: 5739303)
[PASS] testMintInactiveFail(bool,address,uint256,uint256) (runs: 1024, μ: 2824294, ~: 5498478)
[PASS] testMintInactiveInGroupFail(bool,address,uint256,uint256) (runs: 1024, μ: 2726850, ~: 100850)
[PASS] testMintInactiveSingleFail(bool,address,uint256,uint256) (runs: 1024, μ: 2953784, ~: 5553152)
[PASS] testMintSingleSuccess(bool,address,uint256,uint256) (runs: 1024, μ: 2841116, ~: 187300)
[PASS] testMintTokenSupplyExceeded(bool,address,uint256,uint256,uint256) (runs: 1024, μ: 2797020, ~:
125343)
Test result: ok. 21 passed; 0 failed; 0 skipped; finished in 5.66s

Ran 8 test suites: 105 tests passed, 0 failed, 0 skipped (105 total tests)
```

```
wallet-contracts
```

```
Running 1 test for contracts/mocks/CallReceiverMock.sol:CallReceiverMock
[PASS] testCall(uint256,bytes) (runs: 256, μ: 77897, ~: 69361)
Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 12.21ms

Running 1 test for foundry_test/modules/commons/submodules/nonce/SubModuleNonce.t.sol:SubModuleNonceTest
[PASS] test_decodeNonce(uint160,uint96) (runs: 256, μ: 850, ~: 850)
Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 12.91ms

Running 1 test for foundry_test/modules/commons/ModuleERC5719.t.sol:ModuleERC5719Test
[PASS] test_getAlternativeSignature() (gas: 126269)
Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 632.54µs

Running 1 test for foundry_test/utils/LibAddress.t.sol:LibAddressTest
[PASS] test_isContract(address,bytes) (runs: 256, μ: 5987, ~: 5986)
Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 23.75ms

Running 2 tests for foundry_test/modules/commons/ModuleStorage.t.sol:ModuleStorageTest
[PASS] test_writeBytes32(bytes32,bytes32,bytes32,bytes32) (runs: 256, μ: 59416, ~: 59416)
[PASS] test_writeBytes32Map(bytes32,bytes32,bytes32,bytes32,bytes32,bytes32) (runs: 256, μ: 59093, ~:
```

```
59093)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 40.07ms

Running 3 tests for foundry_test/utils/LibOptim.t.sol:LibOptimTest
[PASS] test_call(address,uint256,bytes) (runs: 256, μ: 37589, ~: 39071)
[PASS] test_fkeccak256_Bytes32_Bytes32_Fuzz(bytes32,bytes32) (runs: 256, μ: 575, ~: 575)
[PASS] test_returnData_Fuzz(bytes) (runs: 256, μ: 167904, ~: 151927)
Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 78.67ms

Running 3 tests for foundry_test/modules/utils/RequireUtils.t.sol:SubModuleNonceTest
[PASS] test_requireMinNonce(uint160,uint96,uint96) (runs: 256, μ: 40952, ~: 40072)
[PASS] test_requireMinNonceWithExactNonce(uint160,uint96) (runs: 256, μ: 38889, ~: 39978)
[PASS] test_requireNonExpired(uint256) (runs: 256, μ: 5790, ~: 5511)
Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 47.38ms

Running 2 tests for foundry_test/modules/commons/Implementation.t.sol:ImplementationTest
[PASS] test_setImplementation(address,address) (runs: 256, μ: 117826, ~: 117826)
[PASS] test_setImplementation_matchWallet(bytes32,address,address) (runs: 256, μ: 368661, ~: 368671)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 88.33ms

Running 10 tests for foundry_test/utils/LibBytes.t.sol:LibBytesTest
[PASS] test_readBytes32(bytes,bytes32,bytes) (runs: 256, μ: 7167, ~: 7101)
[PASS] test_readBytes32_Fuzz_AbiDecode(bytes,uint256) (runs: 256, μ: 11308, ~: 11304)
[PASS] test_readBytes32_OutOfBounds(bytes,uint256) (runs: 256, μ: 6282, ~: 6280)
[PASS] test_readFirstUint16(uint16,bytes) (runs: 256, μ: 6821, ~: 6811)
[PASS] test_readFirstUint16_Fuzz_AbiDecode(bytes) (runs: 256, μ: 6600, ~: 6596)
[PASS] test_readFirstUint16_OutOfBounds(uint8) (runs: 256, μ: 7724, ~: 7724)
[PASS] test_readUint32(bytes,uint32,bytes) (runs: 256, μ: 7317, ~: 7278)
[PASS] test_readUint8(bytes,uint8,bytes) (runs: 256, μ: 7306, ~: 7268)
[PASS] test_readUint8_Fuzz_ReadByte(bytes,uint256) (runs: 256, μ: 13984, ~: 14140)
[PASS] test_readUint8_OutOfBounds(bytes,uint256) (runs: 256, μ: 6426, ~: 6424)
Test result: ok. 10 passed; 0 failed; 0 skipped; finished in 176.26ms

Running 4 tests for foundry_test/trust/TrustFactory.t.sol:TrustFactoryTest
[PASS] test_create_trust(address,address,uint256) (runs: 256, μ: 894334, ~: 894334)
[PASS] test_fail_deploy_low_gas(address,address,uint256,uint256) (runs: 256, μ: 768296, ~: 896965)
[PASS] test_fail_deploy_twice(address,address,uint256) (runs: 256, μ: 8937393460516756611, ~:
8937393460516756636)
[PASS] test_predict_address(address,address,uint256) (runs: 256, μ: 906032, ~: 906032)
Test result: ok. 4 passed; 0 failed; 0 skipped; finished in 182.70ms

Running 3 tests for
foundry_test/modules/commons/submodules/auth/SequenceNoChainIdSig.t.sol:SequenceNoChainIdSigTest
[PASS] test_subdigest_DiffAddress(bytes32,address,address) (runs: 256, μ: 22952, ~: 22952)
[PASS] test_subdigest_DiffChainId(bytes32,uint256,uint256) (runs: 256, μ: 16044, ~: 16114)
[PASS] test_subdigest_DiffDigest(bytes32,bytes32) (runs: 256, μ: 6837, ~: 6837)
Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 218.87ms

Running 5 tests for foundry_test/modules/commons/ModuleExtraAuth.t.sol:ModuleExtraAuthTest
[PASS] test_fail_clearExtraImageHash_notSelf(address,bytes32[]) (runs: 256, μ: 11675, ~: 11662)
[PASS] test_fail_setExtraImageHash_notSelf(address,bytes32,uint256) (runs: 256, μ: 10313, ~: 10313)
[PASS] test_shouldAcceptExtraImageHashes(bytes32,bytes32,bytes32,uint256,uint256) (runs: 256, μ: 114705,
~: 114666)
[PASS] test_shouldClearExtraImageHashes(bytes32,(bytes32,uint256)[],bytes32[]) (runs: 256, μ: 7813945, ~:
7954283)
[PASS] test_shouldRejectExpiredImageHash(bytes32,bytes32,bytes32,uint256,uint256) (runs: 256, μ: 88412,
~: 79568)
Test result: ok. 5 passed; 0 failed; 0 skipped; finished in 1.72s

Running 2 tests for foundry_test/modules/commons/ModuleIPFS.t.sol:ModuleIPFSTest
[PASS] test_exposeRoot() (gas: 87238)
[PASS] test_fail_updateIPFSRoot_notSelf(address) (runs: 256, μ: 10093, ~: 10093)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 13.20ms

Running 1 test for
foundry_test/modules/commons/submodules/auth/SequenceDynamicSig.t.sol:SequenceDynamicSigTest
[PASS] test_recover_ignoreFirstByte(uint8,bytes32,uint256,uint16,uint32,uint8) (runs: 256, μ: 26411, ~:
26337)
Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 211.80ms

Running 14 tests for foundry_test/utils/SignatureValidator.t.sol:SignatureValidatorTest
[PASS] test_isValidSignature_EIP712(uint256,bytes32) (runs: 256, μ: 18350, ~: 18252)
```

```
[PASS] test_isValidSignature_ETHSIGN(uint256,bytes32) (runs: 256, μ: 18770, ~: 18669)
[PASS] test_isValidSignature_Fail_EmptySignature(bytes32,address) (runs: 256, μ: 9651, ~: 9651)
[PASS] test_isValidSignature_Fail_UnsupportedSignatureType(bytes32,address,bytes,uint8) (runs: 256, μ:
12914, ~: 12882)
[PASS] test_isValidSignature_Fail_WALLET_BYTES32_BadBytes4(address,bytes32,bytes,bytes4) (runs: 256, μ:
15200, ~: 15146)
[PASS] test_isValidSignature_Fail_WALLET_BYTES32_BadReturn(address,bytes32,bytes,bytes) (runs: 256, μ:
13386, ~: 13300)
[PASS] test_isValidSignature_WALLET_BYTES32(address,bytes32,bytes) (runs: 256, μ: 14613, ~: 14556)
[PASS] test_recoverSigner_EIP712(uint256,bytes32) (runs: 256, μ: 18125, ~: 18038)
[PASS] test_recoverSigner_ETHSIGN(uint256,bytes32) (runs: 256, μ: 18417, ~: 18312)
[PASS] test_recoverSigner_fail_InvalidLength(bytes32,bytes) (runs: 256, μ: 10761, ~: 10711)
[PASS] test_recoverSigner_fail_InvalidSValue(bytes32,bytes32,uint256,uint8,uint8) (runs: 256, μ: 15928,
~: 16096)
[PASS] test_recoverSigner_fail_InvalidVValue(bytes32,bytes32,uint256,uint8,uint8) (runs: 256, μ: 16690,
~: 16581)
[PASS] test_recoverSigner_fail_RecoverAddressZero(bytes32,bytes32,uint256,uint8,uint8) (runs: 256, μ:
18646, ~: 18724)
[PASS] test_recoverSigner_fail_UnsupportedSignatureType(bytes32,bytes32,uint256,uint8,uint8) (runs: 256,
μ: 18838, ~: 18894)
Test result: ok. 14 passed; 0 failed; 0 skipped; finished in 3.02s

Running 8 tests for foundry_test/modules/commons/ModuleCalls.t.sol:ModuleCallsTest
[PASS] test_execute((address,uint256,bytes)[],bytes,bytes32) (runs: 256, μ: 3039114, ~: 2881228)
[PASS] test_execute_delegateCall((address,uint256,bytes)[],bytes,bytes32) (runs: 256, μ: 15323054, ~:
15249666)
[PASS] test_execute_reverts((address,uint256,bytes)[],uint256,bool,bool,bytes,bytes,bytes32) (runs: 256,
μ: 2739819, ~: 2565369)
[PASS] test_fail_noDelegatecall((address,uint256,bytes)[],bytes,uint256) (runs: 256, μ: 882474, ~:
909941)
[PASS] test_fail_selfExecute_NotSelf((address,uint256,bytes)[],address) (runs: 256, μ: 327386, ~: 310137)
[PASS] test_fail_validateNonce_Normal_Bad(uint160,uint88,uint88) (runs: 256, μ: 38311, ~: 39089)
[PASS] test_selfExecute((address,uint256,bytes)[]) (runs: 256, μ: 2691107, ~: 2465466)
[PASS] test_validateNonce_Normal(uint160,uint88) (runs: 256, μ: 45809, ~: 45906)
Test result: ok. 8 passed; 0 failed; 0 skipped; finished in 4.98s

Running 4 tests for
foundry_test/modules/commons/submodules/auth/SequenceChainedSig.t.sol:SequenceChainedSigTest
[PASS] test_chainedRecover(uint8,bytes32,(bytes32,uint256,uint256,uint56,bytes)[]) (runs: 256, μ:
21985178, ~: 21662620)
[PASS] test_chainedRecover_Fail_Checkpoint(uint8,uint256,bytes32,(bytes32,uint256,uint256,uint56,bytes)
[]) (runs: 256, μ: 21647122, ~: 20611679)
[PASS] test_chainedRecover_Fail_EmptySignature(bytes32) (runs: 256, μ: 9287, ~: 9287)
[PASS] test_chainedRecover_Fail_LowWeight(uint8,uint256,bytes32,(bytes32,uint256,uint256,uint56,bytes)[])
(runs: 256, μ: 19550806, ~: 18396952)
Test result: ok. 4 passed; 0 failed; 0 skipped; finished in 4.88s

Running 13 tests for foundry_test/utils/LibBytesPointer.t.sol:LibBytesPointerTest
[PASS] test_readBytes32(bytes,bytes32,bytes) (runs: 256, μ: 7488, ~: 7423)
[PASS] test_readBytes32_Fuzz_LibBytes(bytes,uint256) (runs: 256, μ: 6578, ~: 6576)
[PASS] test_readBytes32_OutOfBounds(bytes,uint256) (runs: 256, μ: 6537, ~: 6536)
[PASS] test_readFirstUint16_Fuzz_LibBytes(bytes) (runs: 256, μ: 6437, ~: 6435)
[PASS] test_readUint16_Fuzz_ReadFirstUint16(bytes,uint256) (runs: 256, μ: 15387, ~: 15517)
[PASS] test_readUint16_OutOfBounds(bytes,uint256) (runs: 256, μ: 6565, ~: 6563)
[PASS] test_readUint24(bytes,uint24,bytes) (runs: 256, μ: 7559, ~: 7523)
[PASS] test_readUint24_OutOfBounds(bytes,uint256) (runs: 256, μ: 6571, ~: 6569)
[PASS] test_readUint64(bytes,uint64,bytes) (runs: 256, μ: 7558, ~: 7521)
[PASS] test_readUint64_OutOfBounds(bytes,uint256) (runs: 256, μ: 6548, ~: 6546)
[PASS] test_readUint8Address(bytes,uint8,address,bytes) (runs: 256, μ: 7974, ~: 7920)
[PASS] test_readUint8Address_OutOfBounds(bytes,uint256) (runs: 256, μ: 6651, ~: 6649)
[PASS] test_readUint8_Fuzz_LibBytes(bytes,uint256) (runs: 256, μ: 6673, ~: 6671)
Test result: ok. 13 passed; 0 failed; 0 skipped; finished in 5.06s

Running 27 tests for foundry_test/trust/Trust.t.sol:TrustTest
[PASS] test_accept_contract_signature_for_beneficiary(address,uint256,uint256,uint256,bytes,bytes) (runs:
256, μ: 1350413, ~: 1350429)
[PASS] test_accept_contract_signature_for_owner(address,uint256,uint256,uint256,bytes,bytes) (runs: 256,
μ: 1349606, ~: 1349547)
[PASS] test_define_initial_parameters(uint256,uint256,uint256) (runs: 256, μ: 923193, ~: 922782)
[PASS]
test_fail_bad_contract_signature_for_beneficiary(address,uint256,uint256,uint256,bytes4,bytes,bytes)
(runs: 256, μ: 1350280, ~: 1356487)
```

```
[PASS] test_fail_bad_contract_signature_for_owner(address,uint256,uint256,uint256,bytes4,bytes,bytes)
(runs: 256, μ: 1349074, ~: 1355739)
[PASS] test_fail_bubble_up_fail(uint256,uint256,bool,uint256,uint256,uint256,uint256,bytes,bytes) (runs:
256, μ: 1090400, ~: 1080010)
[PASS]
test_fail_isValidSignature_anyNetwork_onlyEncode(uint256,uint256,uint256,uint256,uint256,uint64,bool,byte
s) (runs: 256, μ: 951386, ~: 951221)
[PASS]
test_fail_isValidSignature_anyNetwork_wrongEncode(uint256,uint256,uint256,uint256,uint256,uint64,bool,byt
es) (runs: 256, μ: 951277, ~: 951126)
[PASS] test_fail_isValidSignature_emptySignature(uint256,uint256,uint256,uint256,bytes) (runs: 256, μ:
909107, ~: 909120)
[PASS] test_fail_isValidSignature_invalid_signature(uint256,uint256,uint256,bool,uint256,uint256,bytes)
(runs: 256, μ: 947032, ~: 946834)
[PASS] test_fail_isValidSignature_pre_unlock_as_beneficiary(uint256,uint256,uint256,uint256,bytes) (runs:
256, μ: 917194, ~: 916901)
[PASS] test_fail_isValidSignature_wrongSignatureFlag(uint256,uint256,uint256,uint256,uint8,bytes,bytes)
(runs: 256, μ: 914219, ~: 914197)
[PASS] test_fail_revert_contract_signer_beneficiary(address,uint256,uint256,uint256,bytes,bytes,bytes)
(runs: 256, μ: 1403224, ~: 1394456)
[PASS] test_fail_revert_contract_signer_owner(address,uint256,uint256,uint256,bytes,bytes,bytes) (runs:
256, μ: 1401216, ~: 1393288)
[PASS] test_fail_schedule_in_the_past(uint256,uint256,uint256,uint256) (runs: 256, μ: 928208, ~: 927775)
[PASS] test_fail_schedule_non_allowed(uint256,uint256,uint256,uint256,uint256) (runs: 256, μ: 928662, ~:
928111)
[PASS] test_fail_schedule_unlock_too_early(uint256,uint256,uint256,uint256) (runs: 256, μ: 928329, ~:
928350)
[PASS]
test_fail_send_transaction_non_allowed(uint256,uint256,uint256,uint256,uint256,uint256,uint256,uint256,by
tes) (runs: 256, μ: 928233, ~: 928026)
[PASS]
test_fail_send_transaction_pre_unlock_as_beneficiary(uint256,uint256,uint256,uint256,uint256,uint256,uint
256,bytes) (runs: 256, μ: 926206, ~: 925943)
[PASS] test_isValidSignature(uint256,uint256,uint256,uint256,uint256,bool,bytes) (runs: 256, μ: 945216,
~: 945113)
[PASS] test_isValidSignature_anyNetwork(uint256,uint256,uint256,uint256,uint256,uint64,bool,bytes) (runs:
256, μ: 946047, ~: 945912)
[PASS] test_isValidSignature_pre_unlock_as_owner(uint256,uint256,uint256,uint256,bytes) (runs: 256, μ:
920037, ~: 920006)
[PASS] test_schedule_unlock(uint256,uint256,bool,uint256,uint256) (runs: 256, μ: 927393, ~: 927449)
[PASS]
test_send_transaction_after_unlock(uint256,uint256,bool,uint256,uint256,uint256,uint256,uint256,bytes)
(runs: 256, μ: 988094, ~: 989393)
[PASS] test_send_transaction_pre_unlock_as_owner(uint256,uint256,uint256,uint256,uint256,uint256,bytes)
(runs: 256, μ: 955128, ~: 956491)
[PASS] test_start_locked(uint256,uint256,uint256) (runs: 256, μ: 917428, ~: 917274)
[PASS] test_wait_for_unlock(uint256,uint256,uint256,uint256,uint256) (runs: 256, μ: 931811, ~: 931573)
Test result: ok. 27 passed; 0 failed; 0 skipped; finished in 14.36s

Running 19 tests for
foundry_test/modules/commons/submodules/auth/SequenceBaseSig.t.sol:SequenceBaseSigTest
[PASS] test_leafForAddressAndWeight(address,uint96) (runs: 256, μ: 6410, ~: 6410)
[PASS] test_leafForAddressAndWeight_fuzz(address,uint96,address,uint96) (runs: 256, μ: 7485, ~: 7485)
[PASS] test_leafForHardcodedSubdigest_fuzz(bytes32,bytes32) (runs: 256, μ: 807, ~: 807)
[PASS] test_leafForHardcodedSubdigest_fuzz_addr(address,uint96,bytes32) (runs: 256, μ: 861, ~: 861)
[PASS] test_leafForNested_fuzz(bytes32,uint256,uint256,bytes32,uint256,uint256) (runs: 256, μ: 1102, ~:
1102)
[PASS] test_leafForNested_fuzz_addr(address,uint96,bytes32,uint256,uint256) (runs: 256, μ: 969, ~: 969)
[PASS] test_leafForNested_fuzz_subdigest(bytes32,bytes32,uint256,uint256) (runs: 256, μ: 906, ~: 906)
[PASS] test_recover(bytes32,uint256,uint32,uint16,uint8) (runs: 256, μ: 20677, ~: 20591)
[PASS] test_recoverBranch_Addresses(bytes32,bytes32,address[]) (runs: 256, μ: 1444501, ~: 996185)
[PASS] test_recoverBranch_Branches(bytes32,bytes32,uint256[]) (runs: 256, μ: 256633, ~: 224009)
[PASS] test_recoverBranch_Empty(bytes32) (runs: 256, μ: 8419, ~: 8419)
[PASS] test_recoverBranch_Fail_InvalidFlag(uint8,bytes23,bytes) (runs: 256, μ: 12393, ~: 12387)
[PASS] test_recoverBranch_Nodes(bytes32,bytes32[]) (runs: 256, μ: 1328210, ~: 809917)
[PASS] test_recoverBranch_Signatures(bytes32,bytes32,uint256[]) (runs: 256, μ: 5082226, ~: 3248418)
[PASS] test_recover_Fail_EmptySignature(bytes32) (runs: 256, μ: 9020, ~: 9020)
[PASS] test_subdigest(bytes32,uint256) (runs: 256, μ: 13336, ~: 13076)
[PASS] test_subdigest_Fuzz_Address(bytes32,address,address) (runs: 256, μ: 31039, ~: 31039)
[PASS] test_subdigest_Fuzz_ChainId(bytes32,uint256,uint256) (runs: 256, μ: 16181, ~: 16237)
[PASS] test_subdigest_Fuzz_Digest(bytes32,bytes32) (runs: 256, μ: 6989, ~: 6989)
Test result: ok. 19 passed; 0 failed; 0 skipped; finished in 14.32s
```

```
Ran 20 test suites: 124 tests passed, 0 failed, 0 skipped (124 total tests)
```

Code Coverage

Code coverage for `marketplace-contracts` was not executed due to a `Stack too deep` error. We recommend the Sequence team fixes this to get updated code coverage metrics.

```
Compiler run failed:
Error: Compiler error (/Users/eop/dev/paradigm/solidity/libsolidity/codegen/LValue.cpp:56):Stack too
deep. Try compiling with `--via-ir` (cli) or the equivalent `viaIR: true` (standard JSON) while enabling
the optimizer. Otherwise, try removing local variables.
   --> contracts/Orderbook.sol:238:78:
    |
238 |     emit OrderAccepted(orderId, msg.sender, tokenContract, quantity, _orders[orderId].quantity);
    |                                                                              ^^^^^^^
```

It is recommended to reach **100% of branch coverage** in all files.

Fix review update: Code coverage for `marketplace-contracts` and `contracts-library` was not executed due to a `Stack too deep` error. We recommend the Sequence team fixes this to get updated code coverage metrics. `Trust.sol` contract shows a 95.45% of branch coverage.

Initial `contracts-library` coverage:

File	% Lines	% Statements	% Branches	% Funcs
src/proxies/SequenceProxyFactory.sol	80.00% (**8**/10)	85.71% (**12**/14)	100.00% (**0**/0)	75.00% (**3**/4)
src/proxies/TransparentUpgradeableBeaconProxy.sol	89.47% (**17**/19)	92.59% (**25**/27)	62.50% (**5**/8)	100.00% (**4**/4)
src/proxies/openzeppelin/BeaconProxy.sol	33.33% (**1**/3)	40.00% (**2**/5)	100.00% (**0**/0)	33.33% (**1**/3)
src/proxies/openzeppelin/ERC1967Proxy.sol	100.00% (**1**/1)	100.00% (**2**/2)	100.00% (**0**/0)	100.00% (**1**/1)
src/proxies/openzeppelin/TransparentUpgradeableProxy.sol	50.00% (**17**/34)	47.73% (**21**/44)	57.14% (**8**/14)	50.00% (**4**/8)
src/tokens/ERC1155/ERC1155BaseToken.sol	80.00% (**12**/15)	81.82% (**18**/22)	100.00% (**0**/0)	87.50% (**7**/8)
src/tokens/ERC1155/extensions/supply/ERC1155Supply.sol	93.55% (**29**/31)	94.74% (**36**/38)	50.00% (**2**/4)	100.00% (**5**/5)
src/tokens/ERC1155/presets/items/ERC1155Items.sol	90.00% (**9**/10)	93.33% (**14**/15)	100.00% (**2**/2)	100.00% (**4**/4)
src/tokens/ERC1155/presets/items/ERC1155ItemsFactory.sol	100.00% (**5**/5)	100.00% (**6**/6)	100.00% (**0**/0)	100.00% (**1**/1)
src/tokens/ERC1155/utility/sale/ERC1155Sale.sol	89.09% (**49**/55)	92.31% (**72**/78)	62.50% (**10**/16)	70.00% (**7**/10)
src/tokens/ERC1155/utility/s	100.00% (**5**/5)	100.00% (**6**/6)	100.00% (**0**/0)	100.00% (**1**/1)

File	% Lines	% Statements	% Branches	% Funcs
ale/ERC1155SaleFactory.sol				
src/tokens/ERC20/ERC20BaseToken.sol	92.31% (**12/**13)	90.00% (**18/**20)	50.00% (**1/**2)	100.00% (**6/**6)
src/tokens/ERC20/presets/items/ERC20Items.sol	70.00% (**7/**10)	80.00% (**12/**15)	100.00% (**2/**2)	75.00% (**3/**4)
src/tokens/ERC20/presets/items/ERC20ItemsFactory.sol	100.00% (**5/**5)	100.00% (**6/**6)	100.00% (**0/**0)	100.00% (**1/**1)
src/tokens/ERC721/ERC721BaseToken.sol	95.45% (**21/**22)	93.55% (**29/**31)	100.00% (**0/**0)	100.00% (**11/**11)
src/tokens/ERC721/presets/items/ERC721Items.sol	100.00% (**8/**8)	100.00% (**12/**12)	100.00% (**2/**2)	100.00% (**3/**3)
src/tokens/ERC721/presets/items/ERC721ItemsFactory.sol	100.00% (**6/**6)	100.00% (**6/**6)	100.00% (**0/**0)	100.00% (**1/**1)
src/tokens/ERC721/utility/minter/ERC721PermissiveMinter.sol	0.00% (**0/**1)	0.00% (**0/**1)	100.00% (**0/**0)	0.00% (**0/**1)
src/tokens/ERC721/utility/sale/ERC721Sale.sol	85.71% (**24/**28)	88.10% (**37/**42)	70.00% (**7/**10)	75.00% (**6/**8)
src/tokens/ERC721/utility/sale/ERC721SaleFactory.sol	0.00% (**0/**5)	0.00% (**0/**6)	100.00% (**0/**0)	0.00% (**0/**1)
src/tokens/common/ERC2981Controlled.sol	50.00% (**2/**4)	20.00% (**2/**10)	100.00% (**0/**0)	66.67% (**2/**3)
src/tokens/common/MerkleProofSingleUse.sol	100.00% (**5/**5)	100.00% (**9/**9)	100.00% (**4/**4)	50.00% (**1/**2)
src/tokens/common/WithdrawControlled.sol	75.00% (**3/**4)	80.00% (**4/**5)	50.00% (**1/**2)	100.00% (**2/**2)
src/utils/StorageSlot.sol	0.00% (**0/**4)	0.00% (**0/**4)	100.00% (**0/**0)	0.00% (**0/**4)
test/TestHelper.sol	0.00% (**0/**17)	0.00% (**0/**24)	100.00% (**0/**0)	0.00% (**0/**5)
test/proxies/SequenceProxyFactory.t.sol	100.00% (**5/**5)	100.00% (**7/**7)	100.00% (**0/**0)	100.00% (**5/**5)
Total	77.23% (**251/**325)	78.24% (**356/**455)	66.67% (**44/**66)	74.53% (**79/**106)

Changelog

- 2023-12-20 - Initial report
- 2024-01-12 - 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

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.

Quantstamp