Quantstamp Security Assessment Certificate

May 6th 2022 — Quantstamp Verified

Rara

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

Executive Summary

Type	
Auditors	Fayçal Lalidji, Senior Security Engineer Marius Guggenmos, Senior Research Engineer Roman Rohleder, Research Engineer
Timeline	2022-04-11 through 2022-05-05
EVM	
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	Medium
Test Quality	Medium

Source Code

Repository	Commit
rara-protocol	a4a2474

Total Issues	**13** (9 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	**1** (1 Resolved)
Low Risk Issues	**7** (4 Resolved)
Informational Risk Issues	**5** (4 Resolved)
Undetermined Risk Issues	0 (0 Resolved)

ALL ISSUES ADDRESSED

Documentation Issues Addressed

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

Summary of Findings

Initial audit:
Through reviewing the code, we found **12 potential issues.** We recommend carefully re-considering the logic to ensure the safety of users. In addition, the contracts grant many privileges to trusted roles, which opens room for manipulating users' assets.
Reaudit:
All highlighted issues have been either fixed, mitigated or acknowledged.

ID	Description	Severity	Status
QSP-1	Use of Unsafe Cast Operations	^ Medium	Fixed
QSP-2	Violating Checks Effects Interactions Pattern	⌄ Low	Fixed
QSP-3	Privileged Roles and Ownership	⌄ Low	Mitigated
QSP-4	Missing Input Validation	⌄ Low	Fixed
QSP-5	Re-registering NFTs Transfers Taker Rewards	⌄ Low	Acknowledged
QSP-6	Re-registering NFTs Without De-registering Is Possible	⌄ Low	Acknowledged
QSP-7	Front Running `fxRootTunnel` Setting	⌄ Low	Fixed
QSP-8	Possible Wrong Taker Amount Value	⌄ Low	Acknowledged
QSP-9	Events Not Emitted on State Change	○ Informational	Fixed
QSP-10	Using Low Level Calls	○ Informational	Fixed
QSP-11	Duplicate Overflow Checks	○ Informational	Fixed
QSP-12	Bonding Curve Parameters Modification	○ Informational	Fixed
QSP-13	Upgradeable Contracts	○ Informational	Acknowledged

Quantstamp Audit Breakdown

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

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

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

Methodology

The Quantstamp auditing process follows a routine series of steps:

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

Toolset

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

Setup

Tool Setup:

- Slither v0.8.2

Steps taken to run the tools:

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

Findings

QSP-1 Use of Unsafe Cast Operations

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `SigmoidCuratorVault.sol`, `Sigmoid.sol`

Description: In L111 and L140 of `Sigmoid.sol` and in L123-L128 and L189-L194 of `SigmoidCuratorVault.sol` primitive cast operations (`uint256(...)` and `int256(...)`), which are prone for over-/underflows, are used.

Recommendation: Replace the unsafe cast operations with ones that check whether the target type can represent the value being cast. Refer to or use OpenZeppelin's SafeCast library.

Update: Fixed in `https://github.com/rara-social/rara-protocol/pull/54`.

QSP-2 Violating Checks Effects Interactions Pattern

Severity: *Low Risk*

Status: Fixed

File(s) affected: `SigmoidCuratorVault.sol`, `ReactionVault.sol`

Description: The main purpose of the Checks Effects Interactions pattern is to reduce the attack surface for malicious contracts trying to hijack control flow after an external call. Even if it seems that implementing a non-reentrant modifier will actually prevent reentrancy at a contract level, other Dapps or contracts of the same application might rely on a sensitive state that can be manipulated by an attacker.
`ReactionVault._buyReaction()` calls `paymentToken.safeTransferFrom` before updating the contract state. Similarly, `SigmoidCuratorVault.buyCuratorTokens()` calls `paymentToken.safeTransferFrom()`.

Recommendation: All external calls should be moved to the end of the functions to avoid tampering with the contract state.

Update: Fixed in `https://github.com/rara-social/rara-protocol/pull/55`.

QSP-3 Privileged Roles and Ownership

Severity: *Low Risk*

Status: Mitigated

File(s) affected: `CuratorToken1155.sol`, `AddressManager.sol`, `RoleManager.sol`, `ParameterManager.sol`, `SigmoidCuratorVault.sol`, `ReactionNft1155.sol`

Description: Certain contracts have state variables, e.g. `owner`, which provide certain addresses with privileged roles. Such roles may pose a risk to end-users.
The `AddressManager.sol` contract contains the following privileged roles:

- `ADDRESS_MANAGER_ADMIN`, as set by the `protocolAdmin` of `RoleManager.sol`:
  . Renounce his role and thereby disable all subsequently listed actions, by calling `RoleManager.renounceRole()`.
  . Change the role manager address (`roleManager`) to a new non-zero address that implements a `isAdmin()` function, by calling `setRoleManager()`.
  . Change the parameter manager address (`parameterManager`) to an arbitrary non-zero address, by calling `setParameterManager()`.
  . Change the maker registrar address (`makerRegistrar`) to an arbitrary non-zero address, by calling `setMakerRegistrar()`.
  . Change the reaction NFT contract address (`reactionNftContract`) to an arbitrary non-zero address, by calling `setReactionNftContract()`.
  . Change the default curator vault address (`defaultCuratorVault`) to an arbitrary non-zero address, by calling `setDefaultCuratorVault()`.
  . Change the L2 bridge registrar address (`childRegistrar`) to an arbitrary non-zero address, by calling `setChildRegistrar()`.

The `RoleManager.sol` contract contains the following privileged roles:

- `DEFAULT_ADMIN_ROLE`, as initialized to the parameter `protocolAdmin` during `initialize()`:
  . Renounce his role and thereby disable all subsequently listed actions, by calling `renounceRole()`.
  . Grant and revoke arbitrary roles (such as `ADDRESS_MANAGER_ADMIN`, `PARAMETER_MANAGER_ADMIN`, `REACTION_NFT_ADMIN`, `CURATOR_VAULT_PURCHASER` and `CURATOR_TOKEN_ADMIN`) to/of arbitrary addresses, by calling `grantRole()` and `revokeRole()`.

The `ParameterManager.sol` contract contains the following privileged roles:

- `PARAMETER_MANAGER_ADMIN`, as set by the `protocolAdmin` of `RoleManager.sol`:
  . Renounce his role and thereby disable all subsequently listed actions, by calling `RoleManager.renounceRole()`.
  . Change the accepted payment token to an arbitrary non-zero address, by calling `setPaymentToken()`.
  . Change the reaction price to an arbitrary non-zero value, by calling `setReactionPrice()`.
  . Change the cut of purchase price going to the curator liability (even above 100% - see input validation finding), by calling `setSaleCuratorLiabilityBasisPoints()`.
  . Change the cut of purchase price going to the referrer (even above 100% - see input validation finding), by calling `setSaleReferrerBasisPoints()`.
  . Change the cut of spend curator liability going to the taker (even above 100% - see input validation finding), by calling `setSpendTakerBasisPoints()`.
  . Change the cut of spend curator liability going to the referrer (even above 100% - see input validation finding), by calling `setSpendReferrerBasisPoints()`.

-
  . Add/Remove arbitrary non-zero addresses from the approved curator vault mapping, by calling `setApprovedCuratorVaults()`.

  . Change the sigmoid bonding curve parameters to arbitrary non-zero values, by calling `setBondingCurveParams()`.

The `SigmoidCuratorVault.sol` contract contains the following privileged roles:

- `CURATOR_VAULT_PURCHASER`, as set by the `protocolAdmin` of `RoleManager.sol`:
  . Renounce his role and thereby disable all subsequently listed actions, by calling `RoleManager.renounceRole()`.

  . Buy curator tokens when reactions are spent, by calling `buyCuratorTokens()`.

The `CuratorToken115.sol` contract contains the following privileged roles:

- `CURATOR_TOKEN_ADMIN`, as set by the `protocolAdmin` of `RoleManager.sol`:
  . Renounce his role and thereby disable all subsequently listed actions, by calling `RoleManager.renounceRole()`.

  . Mint tokens to an arbitrary address, by calling `mint()`.

  . Burn tokens of an arbitrary address, by calling `burn()`.

The `ReactionNft1155.sol` contract contains the following privileged roles:

- `REACTION_NFT_ADMIN`, as set by the `protocolAdmin` of `RoleManager.sol`:
  . Renounce his role and thereby disable all subsequently listed actions, by calling `RoleManager.renounceRole()`.

  . Mint tokens to an arbitrary address, by calling `mint()`.

  . Burn tokens of an arbitrary address, by calling `burn()`.

Recommendation: Clarify the impact of these privileged actions to the end-users via publicly facing documentation.

Update: Fixed in `https://github.com/rara-social/rara-protocol/pull/56`.

QSP-4 Missing Input Validation

Severity: *Low Risk*

Status: Fixed

File(s) affected: `ParameterManager.sol`, `RoleManager.sol`, `SigmoidCuratorVault.sol`, `ReactionVault.sol`

Description: It is important to validate inputs, even if they only come from trusted addresses, to avoid human error. The following functions do not have a proper validation of input parameters:

1. `ParameterManager.setSaleCuratorLiabilityBasisPoints()` does not check that parameter `_saleCuratorLiabilityBasisPoints` is less than 10000 basis points (100%).
2. `ParameterManager.setSaleReferrerBasisPoints()` does not check that parameter `_saleReferrerBasisPoints` is less than 10000 basis points (100%).
3. `ParameterManager.setSpendTakerBasisPoints()` does not check that parameter `_spendTakerBasisPoints` is less than 10000 basis points (100%).
4. `ParameterManager.setSpendReferrerBasisPoints()` does not check that parameter `_spendReferrerBasisPoints` is less than 10000 basis points (100%).
5. `ParameterManager.setBondingCurveParams()` does not check that parameters `a`, `b` and `c` are less than `type(int256).max`.
6. `ParameterManager.intialize()` does not check the manager address is different than `0x0`.
7. `RoleManager.initialize()` does not check the protocol admin address is different than `0x0`.
8. `SigmoidCuratorVault.initialize()` does not validate the inputs.
9. `ReactionVault.initialize()` does not validate the inputs.

Recommendation: Consider adding the relevant input checks.

Update: Fixed in `https://github.com/rara-social/rara-protocol/pull/57`.

QSP-5 Re-registering NFTs Transfers Taker Rewards

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `ReactionVault.sol`

Description: The rewards for `takers` whenever reactions are spent are stored in the `nftOwnerRewards` mapping. This is a mapping from `rewardsIndex` to the rewards. Since the `rewardsIndex` does not include any information about the address owner, the address that is currently registered as the owner of the NFT will be able to claim the rewards.

Recommendation: Make sure users of the platform are aware that they need to claim any taker rewards before they end up selling their NFT or track the rewards by the owner.

Update: "This was a known issue that we have discussed internally when building the protocol. We will add a section to the public documentation before launch explaining the edge cases around rewards and transferring NFT ownership."

QSP-6 Re-registering NFTs Without De-registering Is Possible

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `MakerRegistrar.sol`

Description: The design plan document states:

If an NFT is already registered, trying to re-register will fail. The owner will need to de-register before the register call can be made again.

However, the `_registerForOwner` function in `MakerRegistrar` does not check if the NFT is already registered. It simply overwrites the `sourceToDetailsLookup` mapping with the new information.

Recommendation: Add a require check in `_registerForOwner` that enforces that `sourceToDetailsLookup[sourceId].registered` is `false`.

Update: "Updated design docs with comments about why we originally wanted to prevent re-registration. Code behaves as expected."

QSP-7 Front Running `fxRootTunnel` Setting

Severity: *Low Risk*

Status: Fixed

File(s) affected: `ChildRegistrar.sol`, `RootRegistrar.sol`

Description: `fxRootTunnel` is not initialized in the constructor of `ChildRegistrar` and `ChildRegistrar.setFxRootTunnel()` can be front-run since it is not restricted and can be called by anyone.
Similarly, `fxChildTunnel` can be set by anyone. Even if there is no direct consequence to this action, it is recommended not to leave any function unrestricted.

Recommendation: We recommend setting `fxRootTunnel` and `fxChildTunnel` in every respective constructor.

Update: "Since both contracts need to be deployed before the addresses can be updated, this could not be done in the constructor without pre-determining deployment addresses. Instead, we have submitted a PR to limit the address update to only the account that deploys the contract."
Fixed in `https://github.com/rara-social/rara-protocol/pull/58`.

QSP-8 Possible Wrong Taker Amount Value

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `ReactionVault.sol`

Description: Executing `info.totalCuratorLiability -= info.referrerCut` in `ReactionVault._spendReaction()` before computing `takerAmount` reduces the taker value.

Recommendation: It is unclear if the described behavior is part of the application design, if so it should be clearly documented.

Update: "This was expected behavior that referral fees both on buying reactions and spending reactions subtract the value available to use further for incentives."

QSP-9 Events Not Emitted on State Change

Severity: *Informational*

Status: Fixed

File(s) affected: `AddressManager.sol`, `ParameterManager.sol`

Description: An event should always be emitted when a state change is performed in order to facilitate smart contract monitoring by other systems which want to integrate with the smart contract.
This is not the case for the functions and the correspondingly modified state variables:

1. `AddressManager.setRoleManager()`, after changing `roleManager`.
2. `AddressManager.setParameterManager()`, after changing `parameterManager`.
3. `AddressManager.setMakerRegistrar()`, after changing `makerRegistrar`.
4. `AddressManager.setReactionNftContract()`, after changing `reactionNftContract`.
5. `AddressManager.setDefaultCuratorVault()`, after changing `defaultCuratorVault`.
6. `AddressManager.setChildRegistrar()`, after changing `childRegistrar`.
7. `ParameterManager.setPaymentToken()`, after changing `paymentToken`.
8. `ParameterManager.setReactionPrice()`, after changing `reactionPrice`.
9. `ParameterManager.setSaleCuratorLiabilityBasisPoints()`, after changing `saleCuratorLiabilityBasisPoints`.
10. `ParameterManager.setSaleReferrerBasisPoints()`, after changing `saleReferrerBasisPoints`.
11. `ParameterManager.setSpendTakerBasisPoints()`, after changing `spendTakerBasisPoints`.
12. `ParameterManager.setSpendReferrerBasisPoints()`, after changing `spendReferrerBasisPoints`.
13. `ParameterManager.setApprovedCuratorVaults()`, after changing `approvedCuratorVaults[]`.
14. `ParameterManager.setBondingCurveParams()`, after changing `bondingCurveParams`.

Recommendation: Emit an event in the aforementioned functions.

Update: Fixed in `https://github.com/rara-social/rara-protocol/pull/59`.

QSP-10 Using Low Level Calls

Severity: *Informational*

Status: Fixed

File(s) affected: `NftOwnership.sol`

Description: The `_verifyOwnership` function uses `staticcall` directly by encoding the parameters with `abi.encodeWithSignature` and decoding the results with `abi.decode`. Since this is not type-checked by the compiler, it is easy to introduce errors this way.

Recommendation: Use solidity's try/catch statement instead of low level calls.

Update: Fixed in `https://github.com/rara-social/rara-protocol/pull/60`.

QSP-11 Duplicate Overflow Checks

Severity: Informational

Status: Fixed

File(s) affected: `ExtendedMath.sol`

Description: The `pow2` and `pow3` functions from the `ExtendedMath` library implement explicit overflow checks. Since the contract is compiled with solidity version `0.8.9`, these computations are already checked. That means, the require statements will never be reached in case of an overflow but are still executed when there is no overflow, which wastes gas.

Recommendation: Either remove the explicit overflow checks or add `unchecked` blocks around the computations in case the revert strings are required.

Update: Fixed in `https://github.com/rara-social/rara-protocol/pull/61`.

QSP-12 Bonding Curve Parameters Modification

Severity: Informational

Status: Fixed

File(s) affected: `ParameterManager.sol`

Description: Modifying the bonding curve parameters using `ParameterManager.setBondingCurveParams(...)` can be problematic, if a curator id is already set, meaning that users might not get the expected token amount when selling back to the contract if `a`, `b`, and `c` parameters are updated after its creation.

Recommendation: This behavior must be clearly documented.

Update: "PR submitted to remove updates to bonding curve parameters in the `ParameterManager`. Instead, the parameters are set at the contract initializer and will not be changeable. If a new bonding curve is needed, we will deploy a new one and update the default bonding curve in the `ParameterManager`."
Fixed in `https://github.com/rara-social/rara-protocol/pull/62`.

QSP-13 Upgradeable Contracts

Severity: Informational

Status: Acknowledged

Description: Many contracts are implemented using the proxy pattern to be upgradeable. As a result, some address that owns the proxy is able to upgrade the contract and completely change its code. Since the new code could be anything, it is important to implement good security practices around the address that is capable of upgrading the contracts.

Recommendation: Ensure that no single person can upgrade the contracts. Use either a multi-signature wallet or a DAO to control access to the upgrade functionality. Document your approach transparently for end-users.

Update: "The system will be managed via a Gnosis multisig for the top-level administrative account. We will add notes to the public documentation page before launch explaining the risks."

Automated Analyses

Slither

Slither did not return any significant results.

Adherence to Specification

1. The specification states that `1155's with multiple supply of a single NFT with multiple copies (e.g. supply > 1) will not be supported`. This is however not reflected in the corresponding code, i.e. `NftOwnership._verifyOwnership()` does allow ERC1155 tokens with balances greater than one.
2. Registration of Reaction NFTs should not be possible, however corresponding checks are missing (see TODO item on L145 of `MakerRegistrar.sol`: `TODO: ? Block registration of a RaRa reaction NFT once Reaction Vault is built out`).
3. According to the specification all curator shares of a taker are sold into the bonding curve, when withdrawing (`The curator shares specified are all sold into the bonding curve`). However, the corresponding withdrawal function (`withdrawTakerRewards()`) allows for specifying an amount (`tokensToBurn`) to be exchange with the bonding curve, instead of all.
4. The "UC4: Reactor Spends Reaction" steps list "Quantity to buy" instead of "Quantity to burn/spend".
5. The "UC4: Reactor Spends Reaction" outcomes state that "If the Curator Vault does not yet exist, it will be initialized". However, `ReactionVault L441` only retrieves the default curator vault and attempts to call functions on it without any checks for whether it exists.

Code Documentation

1. Typos and copy and paste errors:
    1. L18 of `IAddressManager.sol`: `role` -> `parameter`.
    2. L21 of `IAddressManager.sol`: `role` -> `parameter`.
    3. L22 of `AddressManagerStorage.sol`: `payment` -> `parameter`.
    4. L35 of `AddressManager.sol`: `role` -> `parameter`.
    5. L53 of `AddressManager.sol`: `maker` -> `reaction NFT contract`.
    6. L9 of `IRoleManager.sol`: `udpate` -> `update`.

7. L23 of `IRoleManager.sol`: `to to` -> `to`.
8. L27 of `RoleManager.sol`: `udpate` -> `update`.
9. L47 of `RoleManager.sol`: `to to` -> `to`.
10. L57 of `RoleManager.sol`: `token` -> `tokens`.
11. L67 of `RoleManager.sol`: `token` -> `tokens`.
12. L47 of `IParameterManager.sol`: `taker` -> `referrer`.
13. L39 of `ParameterManager.sol`: `Setter for the reaction price` -> `Setter for the cut of purchase price going to the curator liability`.
14. L47 of `ParameterManager.sol`: `Setter for the reaction price` -> `Setter for the cut of purchase price going to the referrer`.
15. L90 of `Sigmoid.sol`: `the of` -> `of the`.
16. L89 of `Sigmoid.sol`: `mount` -> `amount`.
17. L119 of `Sigmoid.sol`: `mount` -> `amount`.
18. L187 of `SigmoidCuratorVault.sol`: `minted based on the price` -> `returned based on the tokens`.
19. L42 of `RootRegister.sol`: Missing `creatorSaleBasisPoints` as 2nd last item.
20. L18 of `ReactionVault.sol`: `contract` -> `contract is for`.
21. L111 of `ReactionVault.sol`: `wants` -> `wants to`.
22. L201 of `ReactionVault.sol`: `the this` -> `this`.

2. Unresolved `TODO` items in code:
    1. L20 of `Standard1155.sol`: `Should the URI be updateable?`.
    2. L145 of `MakerRegistrar.sol`: `? Block registration of a RaRa reaction NFT once Reaction Vault is built out`.

3. Incorrect/Incomplete NatSpec code comments:
    1. `IRoleManager.isAdmin()` missing comment for the return value.
    2. `IRoleManager.isAddressManagerAdmin()` missing comment for the return value.
    3. `IRoleManager.isParameterManagerAdmin()` missing comment for the return value.
    4. `IRoleManager.isReactionNftAdmin()` missing comment for the return value.
    5. `IRoleManager.isCuratorVaultPurchaser()` missing comment for the return value.
    6. `IRoleManager.isCuratorTokenAdmin()` missing comment for the return value.
    7. `RoleManager.isAdmin()` missing comment for the return value.
    8. `RoleManager.isAddressManagerAdmin()` missing comment for the return value.
    9. `RoleManager.isParameterManagerAdmin()` missing comment for the return value.
    10. `RoleManager.isReactionNftAdmin()` missing comment for the return value.
    11. `RoleManager.isCuratorVaultPurchaser()` missing comment for the return value.
    12. `RoleManager.isCuratorTokenAdmin()` missing comment for the return value.
    13. `ExtendedMath.pow2()` missing comment for parameter `a`.
    14. `ExtendedMath.sqrt()` missing comment for parameter `y`.
    15. `Sigmoid.calculateTokensBoughtFromPayment()` missing comment for the return value.
    16. `Sigmoid.calculatePaymentReturnedFromTokens()` missing comment for the return value.
    17. `SigmoidCuratorVault.buyCuratorTokens()` missing comments for all parameters.
    18. `SigmoidCuratorVault.sellCuratorTokens()` missing comments for all parameters.
    19. `ReactionVault.withdrawErc20Rewards()` missing comment for the return value.

4. In L19 and L29 of `CuratorToken1155.sol` the comments mention two different roles (`reaction minter role` and `reaction burner role`), however the corresponding functions (`mint()` and `burn()`) are both access controlled by only one role, `onlyCuratorTokenAdmin` (`CURATOR_TOKEN_ADMIN`, as per `RoleManager.sol`). Same applies to L8-9, L18 and L28 of `ReactionNft1155.sol`.

Adherence to Best Practices

1. To facilitate logging it is recommended to index address parameters within events. Therefore the `indexed` keyword should be added to the address parameters in
    1. `SigmoidCuratorVault.CuratorTokensBought()`,
    2. `MakerRegistrar.Registered()`,
    3. `ReactionVault.ReactionsPurchased()`,
    4. `ReactionVault.ReactionsSpent()`,
    5. `ReactionVault.CreatorRewardsGranted()`,
    6. `ReactionVault.ReferrerRewardsGranted()`,
    7. `ReactionVault.MakerRewardsGranted()`,
    8. `ReactionVault.ERC20RewardsClaimed()`.

2. To improve readability and lower the risk of introducing errors when making code changes, it is advised to not use magic constants throughout code, but instead declare them once (as constant and commented) and use these constant variables instead. Following instances should therefore be changed accordingly:

    . L230, L248, L262, L401, L409, L431 and L682 of `ReactionVault.sol`: `10_000`.

3. `CuratorToken1155.onlyCuratorTokenAdmin()` returns an unclear message in case of a restricted address call, a better message can be `Not curator token admin`.

Test Results

Test Suite Results

```
Generating typings for: 65 artifacts in dir: typechain for target: ethers-v5
Successfully generated 111 typings!
Compiled 61 Solidity files successfully

Network Info
============
> HardhatEVM: v2.9.3
> network:    hardhat

No need to generate any newer typings.


  Child Registrar
     ✓ Should validate only deployer can update (1687ms)

  Bridge Registrar
     ✓ Should get initialized with address manager (1044ms)
     ✓ Should prevent non child registrar from registering or de-registering (1024ms)
     ✓ Should allow register and de-register via bridge (1047ms)

  Root Registrar
     ✓ Should validate only deployer can update (76ms)

  AddressManager
     ✓ Should get initialized with role manager (974ms)
     ✓ Should allow owner to set role manager address (1072ms)
     ✓ Should allow owner to set parameter manager address (984ms)
     ✓ Should allow owner to set maker registrar address (996ms)
     ✓ Should allow owner to set reaction NFT address (986ms)
     ✓ Should allow owner to set default curator vault address (985ms)
     ✓ Should allow owner to set child registrar address (1026ms)

  CuratorToken
     ✓ Should get initialized with address manager (1075ms)
     ✓ Should only allow curator vault admin to mint or burn (997ms)

  CuratorVault
     ✓ Should get initialized with address manager (962ms)
     ✓ Should not allow address other than reaction vault purchase (996ms)
     ✓ Should verify payment token (1165ms)
     ✓ Should allow purchase and sale (1095ms)
     ✓ Should allow purchase and sale with increasing price (1125ms)

  Sigmoid Curator Vault
     ✓ Should check address on init (68ms)
     ✓ Should buy and sell small amounts (155ms)
     ✓ Should be able to sell the same amount it bought (802ms)
     ✓ Should buy the max curve with 10^6 decimal token (1397ms)
     ✓ Should validate full curve price (126ms)
     ✓ Should buy small amount at start and end of curve (501ms)

  MakerRegistrar
     ✓ Should get initialized with address manager (895ms)
     ✓ Should verify NFT ownership on register (923ms)
     ✓ Should allow 721 NFT registration  (966ms)
     ✓ Should allow NFT registration again w/ different parameters (1154ms)
     ✓ Should check creator BP out of bounds (944ms)
     ✓ Should emit registration event and verify mappings (999ms)
     ✓ Should verify NFT ownership on deregister (959ms)
     ✓ Should register and deregister and check event (1002ms)

  ParameterManager
     ✓ Should get initialized with address manager (939ms)
     ✓ Should check address on init (40ms)
     ✓ Should allow owner to set payment token address (966ms)
     ✓ Should allow owner to set reaction price (951ms)
     ✓ Should allow owner to set curator liability (987ms)
     ✓ Should allow owner to set sale referrer bp (994ms)
     ✓ Should allow owner to set spend Taker bp (979ms)
     ✓ Should allow owner to set spend Referrer bp (951ms)
     ✓ Should allow owner to set allowed curator vault (998ms)

  RoleManager
     ✓ Should set deploying address as owner by default (77ms)
     ✓ Should check address on init (42ms)
     ✓ Should allow owner to set address manager (131ms)
     ✓ Should allow owner to set parameter manager (127ms)
     ✓ Should allow owner to set reaction nft admin (134ms)
     ✓ Should allow owner to set curator vault purchaser (125ms)
     ✓ Should allow owner to set curator tokens admin (134ms)

  ReactionVault Buy
     ✓ Should buy and spend a single reaction (1432ms)
     ✓ Should buy and spend multiple reactions (1155ms)

  Reaction1155 Token
     ✓ Should get initialized with address manager (938ms)
     ✓ Should mint tokens if authorized (965ms)

  ReactionVault Taker Rewards
     ✓ Should fail without rewards allocated (920ms)
     ✓ Should spend reaction and let taker withdraw - all tokens (1420ms)
     ✓ Should spend reaction and let taker withdraw - some tokens (1413ms)
     ✓ Should spend reaction and let creator get all rewards withdraw (1314ms)

  ReactionVault Buy
     ✓ Should get initialized with address manager (944ms)
     ✓ Should check address on init
     ✓ Should verify NFT is registered (1030ms)
     ✓ Should validate payment succeeds (1061ms)
     ✓ Should buy a single reaction (1003ms)
     ✓ Should buy a multiple reactions (1048ms)

  ReactionVault Withdraw ERC20
     ✓ Should buy a single reaction (1096ms)

  ReactionVault Sell
     ✓ Should verify spender has reaction NFT (963ms)
     ✓ Should verify reaction quantity > 0 (928ms)
     ✓ Should allow spending (1198ms)
     ✓ Should verify referrer cut (1247ms)
     ✓ Should reject invalid curator vault (1086ms)
     ✓ Should allow spending with custom curator vault (1592ms)

  Standard1155 Token
     ✓ Should get initialized with address manager (1001ms)
     ✓ Should mint tokens if authorized (983ms)


  72 passing (1m)
```

Code Coverage

Quantstamp usually recommends developers to increase the branch coverage to 90% and above before a project goes live, in order to avoid hidden functional bugs that might not be easy to spot during the development phase. For branch code coverage, the current targeted files by the audit achieve a lower score that should be improved further.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
Bridge/	20.69	17.65	28.57	20.69	
ChildRegistrar.sol	33.33	37.5	40	35.71	… ,90,105,112
FxBaseChildTunnel.sol	**14.29**	**0**	**20**	**12.5**	**… 50,58,59,72**
FxBaseRootTunnel.sol	7.69	0	16.67	7.69	… 176,196,197

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
RootRegistrar.sol	40	37.5	40	40	... 71,79,89,93
Config/	100	94.44	100	100	
AddressManager.sol	100	94.44	100	100	
AddressManagerStorage.sol	100	100	100	100	
IAddressManager.sol	100	100	100	100	
CuratorVault/	100	100	100	100	
ICuratorVault.sol	100	100	100	100	
CuratorVault/Token/	100	100	100	100	
CuratorToken1155.sol	100	100	100	100	
Maker/	96.67	100	100	96.67	
IMakerRegistrar.sol	100	100	100	100	
MakerRegistrar.sol	100	100	100	100	
MakerRegistrarStorage.sol	100	100	100	100	
NftOwnership.sol	85.71	100	100	85.71	46
Parameters/	100	100	100	100	
IParameterManager.sol	100	100	100	100	
ParameterManager.sol	100	100	100	100	
ParameterManagerStorage.sol	100	100	100	100	
Permissions/	100	100	100	100	
IRoleManager.sol	100	100	100	100	
RoleManager.sol	100	100	100	100	
RoleManagerStorage.sol	100	100	100	100	
Reactions/	100	96.88	100	100	
IReactionVault.sol	100	100	100	100	
ReactionVault.sol	100	96.88	100	100	
ReactionVaultStorage.sol	100	100	100	100	
Reactions/NFT/	100	100	100	100	
ReactionNft1155.sol	100	100	100	100	
SigmoidCuratorVault/	100	75	100	100	
SigmoidCuratorVault.sol	100	75	100	100	
SigmoidCuratorVaultStorage.sol	100	100	100	100	
SigmoidCuratorVault/Curve/	100	66.67	100	100	
ExtendedMath.sol	100	66.67	100	100	
Sigmoid.sol	100	100	100	100	
Testing/	62.5	100	62.5	62.5	
TestErc1155.sol	50	100	50	50	26
TestErc20.sol	66.67	100	66.67	66.67	24
TestErc721.sol	66.67	100	66.67	66.67	23
Token/	100	100	100	100	
IStandard1155.sol	100	100	100	100	
Standard1155.sol	100	100	100	100	
Standard1155Storage.sol	100	100	100	100	
All files	**84.03**	**76.39**	**82**	**84.23**	

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

```
682c12187b5671ab2bc673db5e46e8472d9b5cd816c13f114cb8e93a333e2f80 ./contracts/Parameters/IParameterManager.sol
585fcbc065491f94dc36162cf31d194a593ee85443f50f20faf037ea283caef9 ./contracts/Parameters/ParameterManagerStorage.sol
ee45e3d1b9dfd6d9cdbf1ecb6feaf4928b2c30fc75a4854bb42c8fb2acfe5c62 ./contracts/Parameters/ParameterManager.sol
a54cbdb2b3d9161d6e43ddf282ec65e78e1731b0b59c75d0b3c58763f0340f04 ./contracts/Permissions/RoleManagerStorage.sol
2472e9016a794b5f05b06f78afc7cd14bf005a77628dbce720e4e90aa7b20ec4 ./contracts/Permissions/RoleManager.sol
3f6b9b2123a215aed5a3f3d53162dffb26467b8e244e551352c87957e2ef3a78 ./contracts/Permissions/IRoleManager.sol
0e07556a03d0f25425584f633116be4b924aaa65ca2f503b795e5b6f06e98f02 ./contracts/Token/IStandard1155.sol
1e7f07d84f73b85679145bca36d48b420f5187a46c13fdd92d695cd602ad2fa5 ./contracts/Token/Standard1155Storage.sol
37ae17be7b8fbde9c96c654003adf0a8c61c705868a470ebad519745445cbe7c ./contracts/Token/Standard1155.sol
63e009ebf3ff65c2d2d6061f235550259bbe820c14c4669660cbb65efe0ca38e ./contracts/Reactions/ReactionVault.sol
f4f4643bc495b74133f78db476df5f73a1b47655cf27d7ee70bdaab902c7c606 ./contracts/Reactions/IReactionVault.sol
72c85e0ce0f8544c6ad7f584f561099c869162f0c2db00180fa0b4bee4597271 ./contracts/Reactions/ReactionVaultStorage.sol
53d30cbac4ded8c42ad833b57cbae32943d4e4f6671dbc33006a14353d097ec2 ./contracts/Reactions/NFT/ReactionNft1155.sol
19b433335b4bf7a611f3294347ae6358086e2b73e77d8494a19da26b07ad9637 ./contracts/SigmoidCuratorVault/SigmoidCuratorVault.sol
6c671cc547acd676fc0facd4fb0e0d1bdf66ed1db4f0a03dc80b98e637c27348 ./contracts/SigmoidCuratorVault/SigmoidCuratorVaultStorage.sol
ea66dd1cf45f0a219e6550fa28ad06fce0622d691f1a9ef000c27028d48946e5 ./contracts/SigmoidCuratorVault/Curve/Sigmoid.sol
c3dccf3e087d441c6ba908af740f0ff7115fe90c8cd544f26a852ae103a8230d ./contracts/SigmoidCuratorVault/Curve/ExtendedMath.sol
30655e1312612c7d98751bdfc4af1c32ab9e88c39d51b643736e9358abdcbb9f ./contracts/Maker/NftOwnership.sol
5307711fbed17751eb6560f91813cc63e3ce731e926209b5194eadf6847972cd ./contracts/Maker/MakerRegistrarStorage.sol
af379b1fbd7868c2c07eaaae1118a8c21e07e459229757316dd4ca9f3c5611b2 ./contracts/Maker/MakerRegistrar.sol
a64f41b5a76c28500dbfa8b85ce089d9838c81137a129b91f8468ceeb19e4f50 ./contracts/Maker/IMakerRegistrar.sol
d27ad4bf46977b52328939126c6e18ecb9037f48d4ebe884c51b6810dc3057b8 ./contracts/CuratorVault/ICuratorVault.sol
d113d16bab86d10a506458a7934648222e4b3c6a1f37248725e14ba616dd4e5a ./contracts/CuratorVault/Token/CuratorToken1155.sol
4a0598db44494f85f51256ac67fb8c030148c401ec1d848bb03afa3e28bbae36 ./contracts/Bridge/RootRegistrar.sol
b1c237c4a341148d5f62778664585e1e33f94ba9fd8b444a09a36c607f3f2136 ./contracts/Bridge/ChildRegistrar.sol
a828bdf6743554a7f8b33330436a62b2307c629afef60b3887d33e213e3b0453 ./contracts/Config/AddressManagerStorage.sol
beaccf7de210f4037b24538476b426303470d4dc5d9cc44798b2a89a507d8967 ./contracts/Config/IAddressManager.sol
04e7dc16a9da29b2493bd4276821f25bea1469b1609ad77e6e2a2703e1053812 ./contracts/Config/AddressManager.sol
```

Tests

```
6f075a0ba54b20a8b9fe2640b928ddc4c9937184eac3c6f383290376d0a4113d ./test/Parameters/ParameterManager.ts
b3a806f997e0000201b227ed94ff5fd7f229cdca0ce2acc6eda35611656a2bf3 ./test/Permissions/RoleManager.ts
0ba63bb3f0061334dea6033b2b7f2c2c6c91d6f95bc3b1d4e3cff3f22b150d63 ./test/Token/Standard1155.ts
eec1f262a81dd8ba9d4dfb991ee977f2ac973349b2721e981d9595b469d0c4b6 ./test/Maker/MakerRegistrar.ts
4ee3a66109beb82b14f8479390eba5b203b7b70407d69cb00e1a528945dcbed4 ./test/Scripts/errors.ts
9185d857502d7c192db4290b59b1eb59085faf2869d269c7f8fbaaff16c110a3 ./test/Scripts/derivedParams.ts
f5a41bf543765887b6c9c73f2461be27daf736d674211f60cba3e7c363c8ee0a ./test/Scripts/setup.ts
9b369e629222026988e657da6c72495794ad222eb181257874e48d6ef05125dc ./test/Scripts/constants.ts
9f37945c401c5ded10e1eac8e3e165d09e660be293bd5bfc9aa0363d62929f8d ./test/Reaction/ReactionBuyAndSpend.ts
9c44fe07911a3cc27bf08c310e5436ffa637d8a7129db288da4d966d3402ffb8 ./test/Reaction/ReactionTakerRewards.ts
01e203d31693b49fb70cdac49bf5482296998fc8f0caf449dfde91dc9a7f055e ./test/Reaction/ReactionVaultRewards.ts
6a3b9a173de23fd932e35d313499eba22ff1ec745923fe9da2453d55b1cc10b0 ./test/Reaction/ReactionVaultBuy.ts
fe0d52ed794607e5c3d0d3c7a1b73d1e7abd1335405d2cc065d958ca02551f12 ./test/Reaction/ReactionNftTokens.ts
56ccaf925b73d2b8bd7b1465b2c8100bdb5fb87ae78fbb6bd8a1f831def9c268 ./test/Reaction/ReactionVaultSell.ts
6eabebf56d1114a6ca9cc7e1cbfdae8abded6459a4fc6f44cc910f4d90f9d98b ./test/CuratorVault/CuratorVault.ts
5b4e7313050e1d06bda3a3bc70746968061f09166b99e7706faeb51811f8b44e ./test/CuratorVault/CuratorShares.ts
0830e0ed52155335b774a652f848c90b09ce987c1cc9ceb30c22956c8d59611c ./test/CuratorVault/SigmoidCuratorVault.ts
5dff46a5fdcb467211624c22ec49ce69cc3da832da176ecb57e7024c9981236e ./test/Bridge/MakerRegistrar.ts
34292bc88dfeae6c10c38e07758b97eabf3e83e39ed0f44182e828357f8f9a08 ./test/Config/AddressManager.ts
```

Changelog

- 2022-04-21 - Initial report
- 2022-05-05 - Re-audit report (be43caf)

About Quantstamp

Quantstamp is a Y Combinator-backed company that helps to secure blockchain platforms at scale using computer-aided reasoning tools, with a mission to help boost the adoption of this exponentially growing technology.

With over 1000 Google scholar citations and numerous published papers, Quantstamp's team has decades of combined experience in formal verification, static analysis, and software verification. Quantstamp has also developed a protocol to help smart contract developers and projects worldwide to perform cost-effective smart contract security scans.

To date, Quantstamp has protected $5B in digital asset risk from hackers and assisted dozens of blockchain projects globally through its white glove security assessment services. As an evangelist of the blockchain ecosystem, Quantstamp assists core infrastructure projects and leading community initiatives such as the Ethereum Community Fund to expedite the adoption of blockchain technology.

Quantstamp's collaborations with leading academic institutions such as the National University of Singapore and MIT (Massachusetts Institute of Technology) reflect our commitment to research, development, and enabling world-class blockchain security.