Quantstamp Security Assessment Certificate

QUANTSTAMP VERIFIED SECURITY CERTIFICATE

December 16th 2021 — Quantstamp Verified

MakersPlace CollectionCore Audit

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

Executive Summary

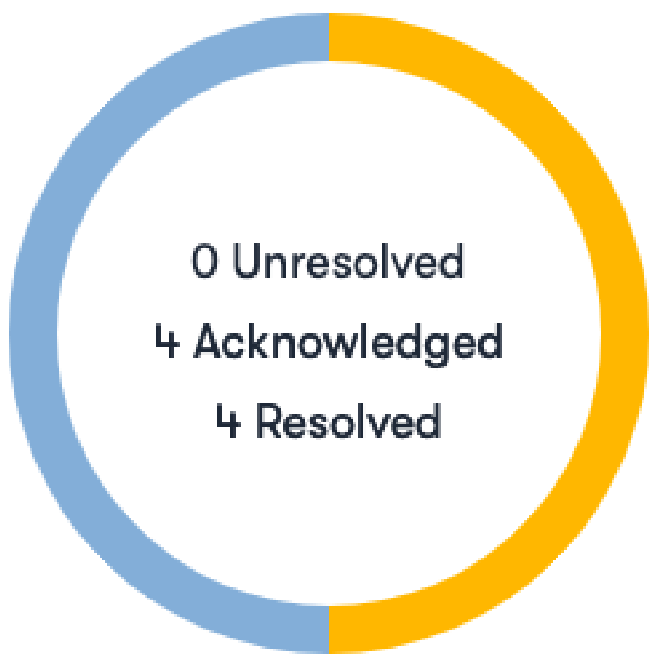
Type	NFT
Auditors	Cristiano Silva, Research Engineer Jake Bunce, Research Engineer Souhail Mssassi, Research Engineer
Timeline	2021-12-02 through 2021-12-15
EVM	London
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	[Collection Pools Documentation]()
Documentation Quality	Medium
Test Quality	Low
Source Code	

Repository	Commit
[collection]()	22adc58
None	cb123b6

Goals

- Match code against specification
- Find logical bugs
- Find potential exploits

Total Issues	**8** (4 Resolved)
High Risk Issues	**1** (1 Resolved)
Medium Risk Issues	0 (0 Resolved)
Low Risk Issues	0 (0 Resolved)
Informational Risk Issues	**7** (3 Resolved)
Undetermined Risk Issues	0 (0 Resolved)



FAST RESPONSE TIMES

⌃ 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).
○ Resolved	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

The code is well written, the solution relies on best practices.

ID	Description	Severity	Status
QSP-1	Sanitize input in `CollectionCore.constructor(...)`	High	Fixed
QSP-2	For Loop Over Dynamic Array	Informational	Acknowledged
QSP-3	Implementation of OBO Operator List	Informational	Acknowledged
QSP-4	Missing validation in function `setBaseURI(...)`	Informational	Acknowledged
QSP-5	Unimplemented code in `CollectionCore.constructor(...)`	Informational	Fixed
QSP-6	Unimplemented code in `CollectionCore.mintTokens(...)`	Informational	Fixed
QSP-7	Ownership can be renounced	Informational	Fixed
QSP-8	No event emitted on deployment	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 v0.6.6
- Truffle v5.3.6

Steps taken to run the tools:

Installed the Slither tool: `pip install slither-analyzer` Run Slither from the project directory: `slither .` Installed Truffle: `npm install -g truffle` truffle test ./test/CollectionCore.js

Findings

QSP-1 Sanitize input in `CollectionCore.constructor(...)`

Severity: *High Risk*

Status: Fixed

File(s) affected: `contracts/CollectionCore.sol`

Description: The `constructor(...)` does not check if`_crsAddress` and `_creator` are different from address 0x0. Similarly, the `constructor()` does not validate if the `_royaltyPercentage` is within a valid range. Thus, any number (even exceeding 100%) can be passed to `_royaltyPercentage` with serious consequences. For instance, the computation of royalties in function `royaltyInfo(...)` will be compromised. The `_initialBaseURI` is not validated as well. There is no inline comment indicating the reason for the command `_pause()`. The code is reproduced below.

```
    constructor(string memory _tokenName, string memory _tokenSymbol,
            address _crsAddress, uint256 _totalSupply,
            address _creator, uint16 _royaltyPercentage, string memory _initialBaseURI) ERC721(_tokenName, _tokenSymbol) {
        setCreatorRegistryStore(_crsAddress);
        require(_totalSupply > 0, "supply > 0");
        // TODO: Should we have a higher limit for totalSupply
        totalMediaSupply = _totalSupply;
        creator = _creator;
        royaltyPercentage = _royaltyPercentage;
        baseURI = _initialBaseURI;
        _pause();
    }
```

Exploit Scenario: A malicious user can use this vulnerability to impose royalties above 100% of the value of the NFT, draining funds from the contract.

Recommendation: Properly validate the input according to the best programming practices. The validation of the `_crsAddress` can take place inside the function `setCreatorRegistryStore(...)`, reverting the operation in case of failure.

Update: Addressed in a pull request. https://github.com/yashh/collection/pull/1

QSP-2 For Loop Over Dynamic Array

Severity: *Informational*

Status: Acknowledged

File(s) affected: `contracts/CollectionCore.sol (L103, L118)`

Description: When smart contracts are deployed or their associated functions are invoked, the execution of these operations always consumes a certain quantity of gas, according to the amount of computation required to accomplish them. Modifying an unknown-size array that grows in size over time can result in a Denial-of-Service attack. Simply by having an excessively huge array, users can exceed the gas limit, therefore preventing the transaction from ever succeeding.

Recommendation: Avoid actions that involve looping across the entire data structure. If you really must loop over an array of unknown size, arrange for it to consume many blocs and thus multiple transactions.

Update: The client argues that dynamic array is only used as an ABI v2 encoder to mint multiple tokens in one call and save transaction fee. I think this pattern is not changing the length of the array that's passed in. It simply allows us to mint multiple tokens in one call. However I do understand the concern of passing a super long list and blowing out the amount of gas. We will not make that mistake, even if we do, our accounts will get clogged since that transaction won't succeed on the chain.

QSP-3 Implementation of OBO Operator List

Severity: *Informational*

Status: Acknowledged

File(s) affected: `contracts/CollectionCore.sol`

Description: `isApprovedForAll` is a check for an oboApproval list for the event of an OBO operator being compromised. A better implementation of this is to use a multisig account where there are defined patterns for user management for this specific event.

Recommendation: Use a multisig wallet where users can be added and removed and avoid defining this yourselves.

Update: OBOOperator List is a list that stores operators that act On Behalf Of (OBO) the custodial accounts of the client. Since custodial account wallets are owned by the client, they have full autonomy over what they approve and disapprove. Therefore, the client does not consider multisig a good solution.

QSP-4 Missing validation in function `setBaseURI(...)`

Severity: *Informational*

Status: Acknowledged

File(s) affected: `contracts/CollectionCore.sol`

Description: The function `setBaseURI(...)` does not validate the input parameter. Thus, any string can be inserted as the `baseURI`. However, there are basic checks that can be implemented to assure that the string is properly formatted (`ipfs://<hash>`). The code is reproduced below:

```
    function setBaseURI(string memory _newBaseURI) external onlyOwner whenNotPaused {
        require(isImmutable == false, "cant change");
        baseURI = _newBaseURI;
    }
```

Recommendation: Validate if `_newBaseURI` is a valid address. Also, consider validating is the `BaseURI` has already been used, avoiding different users from inserting the same value for `BaseURI` in case this is an important feature for the contract. It is not clear in the documentation if users are allowed to reassign the same `BaseURI`.

Update: The client explained that `setBaseURI()` does not validate input parameters in order to save gas fees. Moreover, since this is a deployment parameter, it's the client responsability to not make a mistake.

QSP-5 Unimplemented code in `CollectionCore.constructor(...)`

Severity: *Informational*

Status: Fixed

File(s) affected: `contracts/CollectionCore.sol`

Description: The `constructor(...)` has a comment stating that there is code to be implemented related to validating the upper limit for `total Supply`.

Recommendation: The team must agree on this, reflect this in the code and in the documentation, and send the code for re-audit.

Update: Update from the client: QSP addressed in a github pull request. https://github.com/yashh/collection/pull/2

QSP-6 Unimplemented code in `CollectionCore.mintTokens(...)`

Severity: Informational

Status: Fixed

File(s) affected: `contracts/CollectionCore.sol`

Description: The function `mintTokens(...)` has code still to be implemented. The same piece of code is probably missing in the function `oboMintTokens(...)`.

Recommendation: Implement the missing code before returning to the re-audit. Also double-check if the missing code must appear in the function `oboMintTokens(...)`.

Update: Update from the client: QSP addressed in a github pull request. https://github.com/yashh/collection/pull/2

QSP-7 Ownership can be renounced

Severity: Informational

Status: Fixed

File(s) affected: `all ownable contracts`

Description: The contract is Ownable. `Ownable.sol` has the `function renounceOwnership()`. Although it can only be called by the `owner`, such a function leaves the contract without an owner, which surely compromises any ability to manage the contract. Below we present the code of this dangerous function.

```
    function renounceOwnership() public virtual onlyOwner {
        _transferOwnership(address(0));
    }
```

Recommendation: Overwrite this function to make it revert in order to eliminate the risk of leaving the contract without an owner and thus making unclaimed funds unrecoverable.

Update: Update from the client: QSP addressed in a github pull request. https://github.com/yashh/collection/pull/2

QSP-8 No event emitted on deployment

Severity: Informational

Status: Acknowledged

File(s) affected: `contracts/CollectionCore.sol`

Description: The constructor does not emit an event when it has been called. This is useful for monitoring post-deployment which arguments were passed and validation of correct parameters.

Recommendation: Emit an event with the constructor arguments at deploy time.

Update: Update from the cliente: We do not emit any event to reduce the price to deploy this contract. Moreover, It's a manual deployment.

Automated Analyses

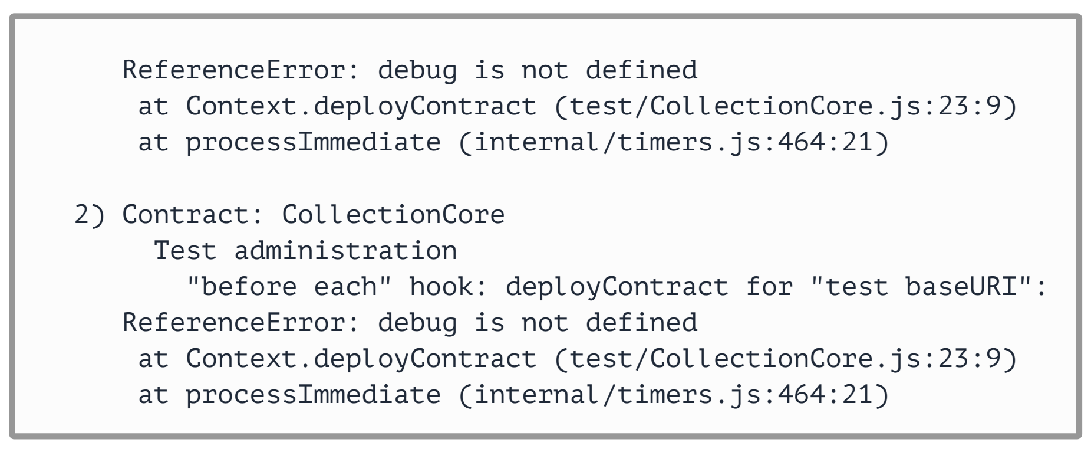
Slither

```
slither .
'npx truffle@5.3.6 compile --all' running (use --truffle-version truffle@x.x.x to use specific version)

Compiling your contracts...
> Compiling ./contracts/ApprovedCreatorRegistry.sol
> Compiling ./contracts/ApprovedCreatorRegistryInterface.sol
> Compiling ./contracts/ApprovedCreatorRegistryReadOnly.sol
> Compiling ./contracts/CollectionCore.sol
> Compiling ./contracts/DigitalMediaBurnInterfaceV2.sol
> Compiling ./contracts/DigitalMediaBurnInterfaceV3.sol
> Compiling ./contracts/DigitalMediaSaleBase.sol
> Compiling ./contracts/OBOControl.sol
> Compiling ./contracts/RoyaltyRegistry.sol
> Compiling ./contracts/RoyaltyRegistryInterface.sol
> Compiling ./contracts/VaultCore.sol
> Compiling ./contracts/VaultCoreInterface.sol
> Compiling ./contracts/utils/HelperUtils.sol
> Compiling @openzeppelin/contracts/access/Ownable.sol
> Compiling @openzeppelin/contracts/security/Pausable.sol
> Compiling @openzeppelin/contracts/token/ERC721/ERC721.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721Receiver.sol
> Compiling @openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol
> Compiling @openzeppelin/contracts/token/ERC721/extensions/IERC721Enumerable.sol
> Compiling @openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol
> Compiling @openzeppelin/contracts/token/ERC721/utils/ERC721Holder.sol
> Compiling @openzeppelin/contracts/utils/Address.sol
> Compiling @openzeppelin/contracts/utils/Context.sol
> Compiling @openzeppelin/contracts/utils/Strings.sol
> Compiling @openzeppelin/contracts/utils/cryptography/ECDSA.sol
> Compiling @openzeppelin/contracts/utils/introspection/ERC165.sol
> Compiling @openzeppelin/contracts/utils/introspection/IERC165.sol
> Artifacts written to /home/cris/auditorias/MAKERS-PLACE-AT922/collection-build/build/contracts
> Compiled successfully using:
   - solc: 0.8.9+commit.e5eed63a.Emscripten.clang

CollectionCore.isApprovedForAll(address,address)._owner (CollectionCore.sol#131) shadows:
    - Ownable._owner (@openzeppelin/contracts/access/Ownable.sol#20) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

OBOControl.setOBOAdmin(address)._oboAdmin (OBOControl.sol#28) lacks a zero-check on :
        - oboAdmin = _oboAdmin (OBOControl.sol#29)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

OBOControl.addApprovedOBO(address) (OBOControl.sol#36-41) uses timestamp for comparisons
    Dangerous comparisons:
    - require(bool,string)(approvedOBOs[_oboAddress] == 0,already added) (OBOControl.sol#38)
OBOControl.isValidApprovedOBO(address) (OBOControl.sol#74-80) uses timestamp for comparisons
    Dangerous comparisons:
    - createdAt == 0 (OBOControl.sol#76)
    - block.timestamp - createdAt > newAddressWaitPeriod (OBOControl.sol#79)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Different versions of Solidity is used:
    - Version used: ['0.8.9', '^0.8.0']
    - ^0.8.0 (@openzeppelin/contracts/utils/Strings.sol#3)
    - ^0.8.0 (@openzeppelin/contracts/token/ERC721/IERC721.sol#3)
    - ^0.8.0 (@openzeppelin/contracts/token/ERC721/utils/ERC721Holder.sol#3)
    - ^0.8.0 (@openzeppelin/contracts/access/Ownable.sol#3)
    - 0.8.9 (DigitalMediaSaleBase.sol#3)
    - 0.8.9 (VaultCore.sol#3)
    - ^0.8.0 (@openzeppelin/contracts/utils/introspection/IERC165.sol#3)
    - ^0.8.0 (@openzeppelin/contracts/token/ERC721/ERC721.sol#3)
    - ^0.8.0 (@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#3)
    - 0.8.9 (ApprovedCreatorRegistryInterface.sol#3)
    - ^0.8.0 (@openzeppelin/contracts/utils/cryptography/ECDSA.sol#3)
    - ^0.8.0 (@openzeppelin/contracts/utils/Address.sol#3)
    - 0.8.9 (utils/HelperUtils.sol#3)
    - ^0.8.0 (@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol#3)
    - ^0.8.0 (@openzeppelin/contracts/token/ERC721/extensions/IERC721Enumerable.sol#3)
    - ^0.8.0 (@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#3)
    - 0.8.9 (DigitalMediaBurnInterfaceV3.sol#3)
    - 0.8.9 (DigitalMediaBurnInterfaceV2.sol#3)
    - 0.8.9 (RoyaltyRegistryInterface.sol#3)
    - ^0.8.0 (@openzeppelin/contracts/security/Pausable.sol#3)
    - 0.8.9 (RoyaltyRegistry.sol#3)
    - 0.8.9 (OBOControl.sol#3)
    - ^0.8.0 (@openzeppelin/contracts/utils/Context.sol#3)
    - 0.8.9 (VaultCoreInterface.sol#3)
    - ^0.8.0 (@openzeppelin/contracts/utils/introspection/ERC165.sol#3)
    - 0.8.9 (ApprovedCreatorRegistry.sol#3)
    - 0.8.9 (CollectionCore.sol#3)
    - 0.8.9 (ApprovedCreatorRegistryReadOnly.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Pragma version^0.8.0 (@openzeppelin/contracts/utils/Strings.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC721/IERC721.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC721/utils/ERC721Holder.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/access/Ownable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (DigitalMediaSaleBase.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (VaultCore.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/introspection/IERC165.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC721/ERC721.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (ApprovedCreatorRegistryInterface.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/cryptography/ECDSA.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/Address.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (utils/HelperUtils.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC721/extensions/IERC721Enumerable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (DigitalMediaBurnInterfaceV3.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (DigitalMediaBurnInterfaceV2.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (RoyaltyRegistryInterface.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/security/Pausable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (RoyaltyRegistry.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (OBOControl.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (VaultCoreInterface.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/introspection/ERC165.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (ApprovedCreatorRegistry.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (CollectionCore.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (ApprovedCreatorRegistryReadOnly.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

ApprovedCreatorRegistry (ApprovedCreatorRegistry.sol#16-95) should inherit from ApprovedCreatorRegistryInterface (ApprovedCreatorRegistryInterface.sol#12-21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance

Parameter RoyaltyRegistry.setCustomCommissionRoyaltyForContract(address,uint16,uint16,bool)._contract (RoyaltyRegistry.sol#118) is not in mixedCase
Parameter RoyaltyRegistry.setCustomCommissionRoyaltyForContract(address,uint16,uint16,bool)._commission (RoyaltyRegistry.sol#119) is not in mixedCase
Parameter RoyaltyRegistry.setCustomCommissionRoyaltyForContract(address,uint16,uint16,bool)._royalty (RoyaltyRegistry.sol#119) is not in mixedCase
Parameter RoyaltyRegistry.setCustomCommissionRoyaltyForContract(address,uint16,uint16,bool)._delete (RoyaltyRegistry.sol#119) is not in mixedCase
Parameter RoyaltyRegistry.setCreatorRegistryStore(address)._crsAddress (RoyaltyRegistry.sol#141) is not in mixedCase
Parameter RoyaltyRegistry.setVaultStore(address)._vaultStore (RoyaltyRegistry.sol#156) is not in mixedCase
Parameter OBOControl.setOBOAdmin(address)._oboAdmin (OBOControl.sol#28) is not in mixedCase
Parameter OBOControl.addApprovedOBO(address)._oboAddress (OBOControl.sol#36) is not in mixedCase
Parameter OBOControl.removeApprovedOBO(address)._oboAddress (OBOControl.sol#46) is not in mixedCase
Parameter OBOControl.addApprovedOBOImmediately(address)._oboAddress (OBOControl.sol#55) is not in mixedCase
Parameter OBOControl.addApprovedOBOAfterDeploy(address)._oboAddress (OBOControl.sol#62) is not in mixedCase
Parameter OBOControl.isValidApprovedOBO(address)._oboAddress (OBOControl.sol#74) is not in mixedCase
Constant OBOControl.newAddressWaitPeriod (OBOControl.sol#10) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter ApprovedCreatorRegistry.setOboApprovalForAll(address,bool)._operator (ApprovedCreatorRegistry.sol#55) is not in mixedCase
Parameter ApprovedCreatorRegistry.setOboApprovalForAll(address,bool)._approved (ApprovedCreatorRegistry.sol#56) is not in mixedCase
Parameter ApprovedCreatorRegistry.approveTokenContract(address)._contract (ApprovedCreatorRegistry.sol#67) is not in mixedCase
Parameter ApprovedCreatorRegistry.disapproveTokenContract(address)._contract (ApprovedCreatorRegistry.sol#74) is not in mixedCase
Parameter ApprovedCreatorRegistry.isOperatorApprovedForCustodialAccount(address,address)._operator (ApprovedCreatorRegistry.sol#87) is not in mixedCase
Parameter ApprovedCreatorRegistry.isOperatorApprovedForCustodialAccount(address,address)._custodialAddress (ApprovedCreatorRegistry.sol#88) is not in mixedCase
Parameter CollectionCore.setSignerAddress(address,bool)._signerAddress (CollectionCore.sol#64) is not in mixedCase
Parameter CollectionCore.setSignerAddress(address,bool)._enableExternalMinting (CollectionCore.sol#64) is not in mixedCase
Parameter CollectionCore.setCreatorRegistryStore(address)._crsAddress (CollectionCore.sol#73) is not in mixedCase
Parameter CollectionCore.setBaseURI(string)._newBaseURI (CollectionCore.sol#90) is not in mixedCase
Parameter CollectionCore.isApprovedForAll(address,address)._owner (CollectionCore.sol#131) is not in mixedCase
Parameter CollectionCore.isApprovedForAll(address,address)._operator (CollectionCore.sol#131) is not in mixedCase
Parameter CollectionCore.royaltyInfo(uint256,uint256)._salePrice (CollectionCore.sol#147) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

RoyaltyRegistry.setVaultStore(address) (RoyaltyRegistry.sol#156-161) uses literals with too many digits:
    - require(bool,string)(contractType == 0x6d707661756c7400000000000000000000000000000000000000000000000000,invalid mpvault) (RoyaltyRegistry.sol#159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

renounceOwnership() should be declared external:
    - CollectionCore.renounceOwnership() (CollectionCore.sol#160-162)
    - Ownable.renounceOwnership() (@openzeppelin/contracts/access/Ownable.sol#53-55)
transferOwnership(address) should be declared external:
    - Ownable.transferOwnership(address) (@openzeppelin/contracts/access/Ownable.sol#61-64)
typeOfContract() should be declared external:
    - VaultCore.typeOfContract() (VaultCore.sol#31-34)
storeTokens(VaultCore.TokenRequest[]) should be declared external:
    - VaultCore.storeTokens(VaultCore.TokenRequest[]) (VaultCore.sol#48-58)
name() should be declared external:
    - ApprovedCreatorRegistryInterface.getVersion() (ApprovedCreatorRegistryInterface.sol#14)
typeOfContract() should be declared external:
    - ApprovedCreatorRegistryInterface.typeOfContract() (ApprovedCreatorRegistryInterface.sol#15)
isOperatorApprovedForCustodialAccount(address,address) should be declared external:
    - ApprovedCreatorRegistryInterface.isOperatorApprovedForCustodialAccount(address,address) (ApprovedCreatorRegistryInterface.sol#16-18)
tokenOfOwnerByIndex(address,uint256) should be declared external:
    - RoyaltyRegistryInterface.typeOfContract() (RoyaltyRegistryInterface.sol#13)
VERSION() should be declared external:
    - RoyaltyRegistryInterface.VERSION() (RoyaltyRegistryInterface.sol#14)
typeOfContract() should be declared external:
    - RoyaltyRegistry.typeOfContract() (RoyaltyRegistry.sol#68-70)
getVersion() should be declared external:
    - ApprovedCreatorRegistry.getVersion() (ApprovedCreatorRegistry.sol#41-43)
typeOfContract() should be declared external:
    - ApprovedCreatorRegistry.typeOfContract() (ApprovedCreatorRegistry.sol#46-48)
approveTokenContract(address) should be declared external:
    - ApprovedCreatorRegistry.approveTokenContract(address) (ApprovedCreatorRegistry.sol#67-71)
disapproveTokenContract(address) should be declared external:
    - ApprovedCreatorRegistry.disapproveTokenContract(address) (ApprovedCreatorRegistry.sol#74-78)
isOperatorApprovedForCustodialAccount(address,address) should be declared external:
    - ApprovedCreatorRegistry.isOperatorApprovedForCustodialAccount(address,address) (ApprovedCreatorRegistry.sol#86-94)
typeOfContract() should be declared external:
    - ApprovedCreatorRegistryReadOnly.typeOfContract() (ApprovedCreatorRegistryReadOnly.sol#26-28)
isOperatorApprovedForCustodialAccount(address,address) should be declared external:
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
. analyzed (28 contracts with 75 detectors), 146 result(s) found
```

Truffle

```
truffle tests
Using network 'development'.


Compiling your contracts...

> Everything is up to date, there is nothing to compile.



  Contract: CollectionCore
    Test contract setup
      1) "before each" hook: deployContract for "test version"
    Test administration
      2) "before each" hook: deployContract for "test baseURI"


  0 passing (428ms)
  2 failing

  1) Contract: CollectionCore
       Test contract setup
         "before each" hook: deployContract for "test version":
```



```
    ReferenceError: debug is not defined
     at Context.deployContract (test/CollectionCore.js:23:9)
     at processImmediate (internal/timers.js:464:21)

 2) Contract: CollectionCore
      Test administration
        "before each" hook: deployContract for "test baseURI":
    ReferenceError: debug is not defined
     at Context.deployContract (test/CollectionCore.js:23:9)
     at processImmediate (internal/timers.js:464:21)
```

Adherence to Specification

The code adheres to the specification provided. At some points, the specification is superficial, not presenting all the details involved in the construction of the contracts nor the functional requirements that must be fulfilled by the contract. Some business rules still lack definition (marked in the code with the tag `TODO`). There is not a formal white paper describing the project so far.

Code Documentation

The code features comments explaining the input parameters in most functions and a generic (and short) explanation about each function. This explanation could be more detailed, including listing which functional requirement of the specification is being handled in each function, allowing the reader of the specification to see the requirement being implemented in code.

Adherence to Best Practices

Good coding style. Missing validation of some input parameters, and the code could present more inline comments and docstrings. Medium documentation of the contracts. Medium level of documentation along the code. Missing the functional requirements of each contract.

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

```
2f929819ed9429a0dbc85f2103ba314f7f98f5e4a33b1f23e8b98c892fe7e5c2 ./contracts/OBOControl.sol
68a5e6af18b74bd84e045b38dd350bd2d3076e89440a4d932fbda947924c3f64 ./contracts/ApprovedCreatorRegistry.sol
4e9968a770c0b6159788724e0e20d00f3b18c97e4a9efaf534a3a0cb69e72477 ./contracts/RoyaltyRegistryInterface.sol
927961b9264902855cedcac709d89fa8e2e30a5c1d2a2b8cf5cdd5a3ce383800 ./contracts/CollectionCore.sol
```

Tests

```
3d926c442345cdbfd21f98b1dc342ae1b5b300cd911337abe6225665f2251020 ./test/CollectionCore.js
6d8ae2be6a5b586c8061ae09226416f777255bf4787ab4ac82596cb6c82c6bd2 ./test/test-utils.js
```

Changelog

- 2021-12-06 - Initial report
- 2021-12-15 - Final report

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.

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.

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, Inc. (Quantstamp). Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' 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 the website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim 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. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.