Quantstamp Security Assessment Certificate

December 21st 2021 — Quantstamp Verified

Pawn

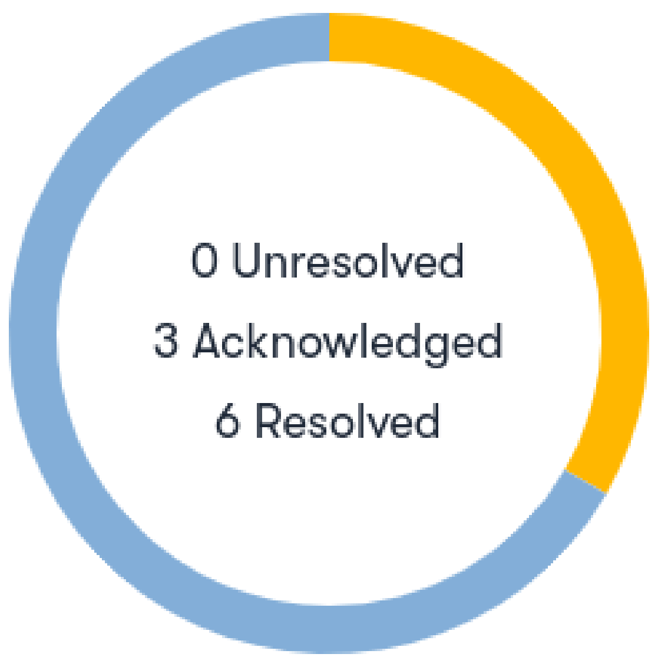
This audit report was prepared by Quantstamp, the leader in blockchain security.

Executive Summary

Type	
Auditors	Fayçal Lalidji, Security Auditor Cristiano Silva, Research Engineer Souhail Mssassi, Research Engineer
Timeline	2021-11-08 through 2021-12-20
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	LiquidMint Documentation
Documentation Quality	Medium
Test Quality	High
Source Code	

Repository	Commit
arcade	7ba9f79
arcade	97436b1

Total Issues	**9** (6 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	**2** (1 Resolved)
Low Risk Issues	**2** (1 Resolved)
Informational Risk Issues	**5** (4 Resolved)
Undetermined Risk Issues	0 (0 Resolved)



High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
Undetermined	The impact of the issue is uncertain.

Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
Acknowledged	The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
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

Initial audit:
Through reviewing the code, we found **9 potential issues** of various levels of severity. We recommend addressing the findings prior to deploying the smart contracts to the main network.
Initial audit:
All highlighted issues have been either fixed or acknowledged.

ID	Description	Severity	Status
QSP-1	Missing Validation in the `originalOwner`	^ Medium	Fixed
QSP-2	Mint Price Manipulation	^ Medium	Acknowledged
QSP-3	Gas Usage / `for` Loop Concerns	˅ Low	Acknowledged
QSP-4	Same NFT URI Can Be Uploaded Under Different Token Ids	˅ Low	Fixed
QSP-5	Owner Can Renounce Ownership	○ Informational	Fixed
QSP-6	Unlocked Pragma	○ Informational	Fixed
QSP-7	Unnecessary Use of SafeMath in Solidity 0.8.x	○ Informational	Fixed
QSP-8	Review `Require` Clauses Related to the Original Supply	○ Informational	Fixed
QSP-9	Unclear Definition of Use Case: Eoa X Smart 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 v0.8.0

Steps taken to run the tools:

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

Findings

QSP-1 Missing Validation in the `originalOwner`

Severity: Medium Risk

Status: Fixed

File(s) affected: `LiquidMint.sol`

Description: Certain functions lack a safety check in the value.

- In the `mintPrint` function, the user can mint an existing original NFT; In this process we calculate the `originalOwnerRoyalty` and send this value to `originalOwner`. The problem here is that the `originalOwner` it can be a contract and in the fallback function it will revert all the transactions sent thus the call function `mintPrint` will fail also.

Recommendation: We recommend verifying if the `originalOwner` is a contract.

Update: Fixed in https://github.com/Non-fungible-Technologies/arcade/pull/4/files.

QSP-2 Mint Price Manipulation

Severity: Medium Risk

Status: Acknowledged

File(s) affected: `LiquidMint.sol`

Description: Allowing the creator to input any polynomial coefficients can represent a risk for users, for example allowing a strict increasing monotonic function will allow the initial users to sell their copies for a higher price (please note that any other price manipulation can be used to the extent of the contract implemented limitation).

Recommendation: The usage of polynomial price representation should be clearly defined and documented, in order to clearly define the risk associated with its usage.

Update: project team reply: "In our opinion the mechanics of pricing are well-documented and the risks are clear. See the Trading Mechanics section in the liquid mint documentation for a discussion of how it works. It is intended that for some pricing curves, initial users can burn their copies for a higher-price than they paid and book a profit, a la other bonding curve models in the wild (e.g. Unisocks or Hashmasks). The risks and opportunities of the pricing model are described in the linked documentation above. Please let us know if anything is unclear or insufficient".

QSP-3 Gas Usage / `for` Loop Concerns

Severity: Low Risk

Status: Acknowledged

File(s) affected: `LiquidMint.sol`

Description: Gas usage is one of the main concerns for smart contract developers and users, since high gas costs may prevent users from wanting to use the smart contract. Even worse, some gas usage issues may prevent the contract from providing services entirely. For example, if a `for` loop requires too much gas to exit, then it may prevent the contract from functioning correctly entirely. It is best to break such loops into individual functions as possible.

Recommendation: We recommend limiting the size of the polynomial coefficients array further in `mint()` to avoid gas consumption issues when executing `getPrintPrice()`.

Update: project team reply:
"The suggested change had already been implemented - we do limit the number of coefficients allowed in the polynomial to 10. See line 246 of the current implementation."
Quantstamp reply:
A polynomial of the 10th degree is very complicated and can lead to unpredictable outcomes for price calculation.

QSP-4 Same NFT URI Can Be Uploaded Under Different Token Ids

Severity: Low Risk

Status: Fixed

File(s) affected: `LiquidMint.sol`

Description: Different creators can deploy an original NFT using the same URI, when one of the main function of the implemented contract is to only allow prints under the same original token. This behavior can hurt project and different NFTs original owners.

Recommendation: We recommend implementing a state variable that will allow verifying if a URI is already taken.

Update: Fixed in https://github.com/Non-fungible-Technologies/arcade/pull/7/files.

QSP-5 Owner Can Renounce Ownership

Severity: Informational

Status: Fixed

File(s) affected: `LiquidMint.sol`

Description: Typically, the contract's owner is the account that deploys the contract. As a result, the owner is able to perform certain privileged activities. The version used in `LiquidMint` contract implements `renounceOwnership` this can represent a certain risk if the ownership is renounced for any other reason than by design.

Recommendation: We recommend to either reimplement the function to disable it or to clearly specify if it is part of the contract design.

Update: Fixed in https://github.com/Non-fungible-Technologies/arcade/pull/6/files.

QSP-6 Unlocked Pragma

Severity: Informational

Status: Fixed

File(s) affected: `LiquidMint.sol`

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".

Recommendation: For consistency and to prevent unexpected behavior in the future, it is recommended to remove the caret to lock the file onto a specific Solidity version.

Update: Fixed in Fixed in https://github.com/Non-fungible-Technologies/arcade/pull/8/files.

QSP-7 Unnecessary Use of SafeMath in Solidity 0.8.x

Severity: Informational

Status: Fixed

File(s) affected: ` LiquidMint.sol` `

Description: Solidity 0.8.x has a built-in mechanism for dealing with overflows and underflows. There is no need to use the `SafeMath` library (it only increases gas usage).
Reference: https://docs.soliditylang.org/en/v0.8.7/080-breaking-changes.html#how-to-update-your-code.

Recommendation: We recommend avoiding using `SafeMath` for more gas optimization.

Update: Fixed in https://github.com/Non-fungible-Technologies/arcade/pull/8/files.

QSP-8 Review `Require` Clauses Related to the Original Supply

Severity: Informational

Status: Fixed

File(s) affected: `LiquidMint.sol`

Description: The `function mint()` uses `SafeMath.sol` and reverts after performing one operation. This can be optimized for saving gas in case of reverting the operation. Below we present the actual code:

```
        uint256 newOriginalsSupply = originalsMinted.add(1);
        require(newOriginalsSupply <= MAX_ORIGINAL_SUPPLY, "Max supply reached");
```

In order to save one add operation, we can reformulate the code as follows:

```
        require(originalsMinted < MAX_ORIGINAL_SUPPLY, "Max supply reached");
        uint256 newOriginalsSupply = originalsMinted + 1;
```

Recommendation: Check if the proposed change is in accordance with the functional requirements.

Update: Fixed in https://github.com/Non-fungible-Technologies/arcade/pull/10/files.

QSP-9 Unclear Definition of Use Case: Eoa X Smart Contracts

Severity: Informational

Status: Acknowledged

File(s) affected: `LiquidMint.sol`

Description: It is not clear if the system is supposed to work only with EOAs, or if it is also designed for Smart Contracts.

Recommendation: In case the system is designed only for EOA, use the Open Zeppelin `Address.sol` contract. Make this clear in the documentation.

Update: project team reply:
"The contract is agnostic to whether it is called or the parties involved are EOAs or contracts. Both should be allowed. Concerns with originalOwner contracts not being able to receive royalties have been addressed as part of QSP-2".

Automated Analyses

Slither

The reported issues by slither do not represent a risk of the users' assets.

```
RC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#385) is a local variable never initialized
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#362) is a local variable never initialized
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#358) is a local variable never initialized
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#381) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#347-368) ignores return value by
IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#358-366)
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#370-391) ignores return value by
IERC1155Receiver(to).onERC1155BatchReceived(operator,from,ids,amounts,data) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#381-389)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#358)' in
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#347-368) potentially used before declaration: response !=
IERC1155Receiver(to).onERC1155Received.selector (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#359)
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#362)' in
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#347-368) potentially used before declaration: revert(string)(reason)
(node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#363)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#381)' in
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#370-391) potentially used before declaration: response !=
IERC1155Receiver(to).onERC1155BatchReceived.selector (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#382)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#385)' in
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#370-391) potentially used before declaration: revert(string)
(reason) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#386)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in LiquidMint.burnPrint(uint256,uint256) (contracts/LiquidMint.sol#328-351):
    External calls:
    - (success) = msg.sender.call{value: burnPrice}() (contracts/LiquidMint.sol#347)
    Event emitted after the call(s):
    - PrintBurned(msg.sender,tokenId,originalId,burnPrice,newSupply,reserve) (contracts/LiquidMint.sol#350)
Reentrancy in LiquidMint.mint(string,address,uint256,int256[]) (contracts/LiquidMint.sol#237-270):
    External calls:
```

```
    - _mint(msg.sender,tokenId,1,) (contracts/LiquidMint.sol#266)
        - IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#358-366)
    Event emitted after the call(s):
    - MintOriginal(msg.sender,tokenId,newOriginalsSupply,currentCreatorFeePct) (contracts/LiquidMint.sol#268)
Reentrancy in LiquidMint.mintPrint(uint256) (contracts/LiquidMint.sol#276-320):
    External calls:
    - _mint(msg.sender,tokenId,1,) (contracts/LiquidMint.sol#298)
        - IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#358-366)
    - (success) = originalOwner.call{value: originalOwnerRoyalty}() (contracts/LiquidMint.sol#302)
    - _refundSender(printPrice) (contracts/LiquidMint.sol#307)
        - (success) = msg.sender.call{value: msg.value.sub(printPrice)}() (contracts/LiquidMint.sol#452)
    External calls sending eth:
    - (success) = originalOwner.call{value: originalOwnerRoyalty}() (contracts/LiquidMint.sol#302)
    - _refundSender(printPrice) (contracts/LiquidMint.sol#307)
        - (success) = msg.sender.call{value: msg.value.sub(printPrice)}() (contracts/LiquidMint.sol#452)
    Event emitted after the call(s):
    - PrintMinted(msg.sender,tokenId,originalId,printPrice,newSupply,originalOwnerRoyalty,reserve,originalOwner) (contracts/LiquidMint.sol#309-318)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
    - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#171-188) uses assembly
    - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#180-183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
    - Version used: ['>=0.8.0', '^0.8.0']
    - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#3)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/IERC1155.sol#3)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/IERC1155Receiver.sol#3)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/IERC1155MetadataURI.sol#3)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Counters.sol#3)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#3)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#3)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#3)
    - >=0.8.0 (contracts/LiquidMint.sol#105)
    - >=0.8.0 (contracts/interfaces/ILiquidMint.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

LiquidMint._beforeTokenTransfer(address,address,address,uint256[],uint256[],bytes) (contracts/LiquidMint.sol#517-534) is never used and should be removed
LiquidMint._isOriginal(uint256) (contracts/LiquidMint.sol#554-556) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/IERC1155.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/IERC1155Receiver.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/IERC1155MetadataURI.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Counters.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@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 version^0.8.0 (node_modules/@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 (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.8.0 (contracts/LiquidMint.sol#105) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.8.0 (contracts/interfaces/ILiquidMint.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.3 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59):
    - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-121):
    - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145):
    - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#143)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169):
    - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#167)
Low level call in LiquidMint.mintPrint(uint256) (contracts/LiquidMint.sol#276-320):
    - (success) = originalOwner.call{value: originalOwnerRoyalty}() (contracts/LiquidMint.sol#302)
Low level call in LiquidMint.burnPrint(uint256,uint256) (contracts/LiquidMint.sol#328-351):
    - (success) = msg.sender.call{value: burnPrice}() (contracts/LiquidMint.sol#347)
Low level call in LiquidMint._refundSender(uint256) (contracts/LiquidMint.sol#449-455):
    - (success) = msg.sender.call{value: msg.value.sub(printPrice)}() (contracts/LiquidMint.sol#452)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter LiquidMint.uri(uint256)._id (contracts/LiquidMint.sol#438) is not in mixedCase
Parameter LiquidMint.setPrice(uint256)._mintPrice (contracts/LiquidMint.sol#481) is not in mixedCase
Parameter LiquidMint.setCreatorFee(uint256)._creatorFeePct (contracts/LiquidMint.sol#491) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (node_modules/@openzeppelin/contracts/utils/Context.sol#21)" inContext (node_modules/@openzeppelin/contracts/utils/Context.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

LiquidMint.SIG_DIGITS (contracts/LiquidMint.sol#148) is never used in LiquidMint (contracts/LiquidMint.sol#121-557)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

renounceOwnership() should be declared external:
    - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-57)
transferOwnership(address) should be declared external:
    - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#63-67)
uri(uint256) should be declared external:
    - ERC1155.uri(uint256) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#58-60)
    - LiquidMint.uri(uint256) (contracts/LiquidMint.sol#438-440)
balanceOfBatch(address[],uint256[]) should be declared external:
    - ERC1155.balanceOfBatch(address[],uint256[]) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#81-100)
setApprovalForAll(address,bool) should be declared external:
    - ERC1155.setApprovalForAll(address,bool) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#105-110)
safeTransferFrom(address,address,uint256,uint256,bytes) should be declared external:
    - ERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#122-151)
safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) should be declared external:
    - ERC1155.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#156-191)
mint(string,address,uint256,int256[]) should be declared external:
    - LiquidMint.mint(string,address,uint256,int256[]) (contracts/LiquidMint.sol#237-270)
mintPrint(uint256) should be declared external:
    - LiquidMint.mintPrint(uint256) (contracts/LiquidMint.sol#276-320)
burnPrint(uint256,uint256) should be declared external:
    - LiquidMint.burnPrint(uint256,uint256) (contracts/LiquidMint.sol#328-351)
getOriginalDataPriceFunctionCoefficient(uint256,uint256) should be declared external:
    - LiquidMint.getOriginalDataPriceFunctionCoefficient(uint256,uint256) (contracts/LiquidMint.sol#402-414)
originalToPrintsSupply(uint256) should be declared external:
    - LiquidMint.originalToPrintsSupply(uint256) (contracts/LiquidMint.sol#423-426)
addCreator(address) should be declared external:
    - LiquidMint.addCreator(address) (contracts/LiquidMint.sol#465-467)
removeCreator(address) should be declared external:
    - LiquidMint.removeCreator(address) (contracts/LiquidMint.sol#473-475)
setPrice(uint256) should be declared external:
    - LiquidMint.setPrice(uint256) (contracts/LiquidMint.sol#481-485)
setCreatorFee(uint256) should be declared external:
    - LiquidMint.setCreatorFee(uint256) (contracts/LiquidMint.sol#491-496)
withdraw() should be declared external:
    - LiquidMint.withdraw() (contracts/LiquidMint.sol#501-504)
setEnabled(bool) should be declared external:
    - LiquidMint.setEnabled(bool) (contracts/LiquidMint.sol#510-512)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Adherence to Best Practices

- The code does not take advantage of the new features of solidity 0.8 (use of the object Error in revert operations, no necessity of using SafeMath.sol);
- The roles and privileges are not well defined.

Test Results

Test Suite Results

```
npm run test

> hardhat test

Creating Typechain artifacts in directory typechain for target ethers-v5
Successfully generated Typechain artifacts!


  LiquidMint
    Admin operations
      ✓ should fail from non-admin (497ms)
      ✓ should not allow adding a creator if the contract is disabled (272ms)
      ✓ should successfully add a creator (147ms)
      ✓ should not allow removing a creator if the contract is disabled (220ms)
      ✓ should successfully remove a creator (226ms)
      ✓ should successfully set a mint price (216ms)
      ✓ should fail to set a mint price if the contract is enabled (96ms)
      ✓ should successfully change the creator fee (200ms)
      ✓ should not allow an invalid creator fee (133ms)
      ✓ should allow ownership to be transferred (98ms)
      ✓ should not allow ownership to be renounced (76ms)
      ✓ should fail to change the creator fee is the contract is enabled (85ms)
    Mint Original
      ✓ should successfully mint an original (150ms)
      ✓ should successfully mint an original with mintPrice (188ms)
      ✓ should successfully mint an original where originalOwner is a contract (201ms)
      ✓ should fail when the contract is disabled (126ms)
      ✓ should fail when not a whitelisted creator (79ms)
      ✓ should fail if the creator cannot receive ether (133ms)
      ✓ should fail when there is a mint price and not paid (167ms)
      ✓ should fail if the token URI has already been used (158ms)
      ✓ should not allow a pricing function with more than 10 coefficients (100ms)
    Mint Print
      ✓ should successfully mint a print (212ms)
      ✓ should successfully mint multiple prints (435ms)
      ✓ should fail if beyond mint cap (191ms)
      ✓ should refund user for overpayment (175ms)
      ✓ should fail when the contract is disabled (180ms)
      ✓ should fail with unknown original (77ms)
      ✓ should fail with insufficient funds (129ms)
    Burn Print
      ✓ should successfully burn a print (302ms)
      ✓ should fail if the contract is disabled (227ms)
      ✓ should fail if less than minimumSupply (188ms)
      ✓ should fail if original does not exist (138ms)
      ✓ should fail if user doesn't own a print (204ms)
      ✓ should fail if there is no print to burn (126ms)
    Withdraw
      ✓ should successfully withdraw ether (229ms)
      ✓ should fail if not admin (182ms)
    Pricing
      f(x) = 10000000000000000 + 1000000000000000x + 1000000000000x^2 + 1000000000000x^3
        ✓ f(0) = 0.01 ETH
        ✓ f(5) = 0.01515 ETH
        ✓ f(10) = 0.0211 ETH
        ✓ f(50) = 0.1875 ETH
        ✓ f(200) = 8.25 ETH
        ✓ f(499) = 125.0095 ETH
      Failure conditions
        ✓ reverts if print price ends up negative (120ms)
        ✓ reverts if print price causes overflow (118ms)
      getOriginalDataPriceFunctionCoefficient
        ✓ returns the pricing function coefficient at a specified index for a given original (118ms)
        ✓ reverts if the requested index is out of bounds (118ms)
        ✓ reverts if the requested original does not exist (112ms)

  47 passing (8s)
```

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 an acceptable score that can be improved further.

```
npm run coverage

> hardhat coverage --solcoverjs ./.solcover.js --temp artifacts --testfiles "./test/**/*.ts"

Version
=======
> solidity-coverage: v0.7.12

Instrumenting for coverage...
=============================

> interfaces/  ILiquidMint.sol
>   LiquidMint.sol
>   MockNoReceive.sol
>   MockReceiver.sol

Compilation:
============

Compiling 14 files with 0.8.3

Compilation finished successfully
Creating Typechain artifacts in directory typechain for target ethers-v5
Successfully generated Typechain artifacts!
Creating Typechain artifacts in directory typechain for target ethers-v5
Successfully generated Typechain artifacts!

Network Info
============
> HardhatEVM: v2.0.10
> network:    hardhat

Creating Typechain artifacts in directory typechain for target ethers-v5
Successfully generated Typechain artifacts!


  LiquidMint
    Admin operations
       ✓ should fail from non-admin (315ms)
       ✓ should not allow adding a creator if the contract is disabled (253ms)
       ✓ should successfully add a creator (168ms)
       ✓ should not allow removing a creator if the contract is disabled (234ms)
       ✓ should successfully remove a creator (244ms)
       ✓ should successfully set a mint price (240ms)
       ✓ should fail to set a mint price if the contract is enabled (131ms)
       ✓ should successfully change the creator fee (236ms)
       ✓ should not allow an invalid creator fee (171ms)
       ✓ should allow ownership to be transferred (139ms)
       ✓ should not allow ownership to be renounced (111ms)
       ✓ should fail to change the creator fee is the contract is enabled (120ms)
    Mint Original
       ✓ should successfully mint an original (236ms)
       ✓ should successfully mint an original with mintPrice (253ms)
       ✓ should successfully mint an original where originalOwner is a contract (254ms)
       ✓ should fail when the contract is disabled (147ms)
       ✓ should fail when not a whitelisted creator (118ms)
       ✓ should fail if the creator cannot receive ether (150ms)
       ✓ should fail when there is a mint price and not paid (182ms)
       ✓ should fail if the token URI has already been used (219ms)
       ✓ should not allow a pricing function with more than 10 coefficients (132ms)
    Mint Print
       ✓ should successfully mint a print (314ms)
       ✓ should successfully mint multiple prints (792ms)
       ✓ should fail if beyond mint cap (324ms)
       ✓ should refund user for overpayment (261ms)
       ✓ should fail when the contract is disabled (250ms)
       ✓ should fail with unknown original (117ms)
       ✓ should fail with insufficient funds (216ms)
    Burn Print
       ✓ should successfully burn a print (529ms)
       ✓ should fail if the contract is disabled (362ms)
       ✓ should fail if less than minimumSupply (297ms)
       ✓ should fail if original does not exist (112ms)
       ✓ should fail if user doesn't own a print (333ms)
       ✓ should fail if there is no print to burn (188ms)
    Withdraw
       ✓ should successfully withdraw ether (329ms)
       ✓ should fail if not admin (296ms)
    Pricing
      f(x) = 10000000000000000 + 1000000000000000x + 1000000000000x^2 + 1000000000000x^3
         ✓ f(0) = 0.01 ETH
         ✓ f(5) = 0.01515 ETH
         ✓ f(10) = 0.0211 ETH
         ✓ f(50) = 0.1875 ETH
         ✓ f(200) = 8.25 ETH
         ✓ f(499) = 125.0095 ETH
      Failure conditions
         ✓ reverts if print price ends up negative (228ms)
         ✓ reverts if print price causes overflow (201ms)
      getOriginalDataPriceFunctionCoefficient
         ✓ returns the pricing function coefficient at a specified index for a given original (184ms)
         ✓ reverts if the requested index is out of bounds (217ms)
         ✓ reverts if the requested original does not exist (194ms)


  47 passing (10s)
```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	98.04	91.67	92.59	98.13	
LiquidMint.sol	100	91.67	100	100	
MockNoReceive.sol	0	100	0	0	9
MockReceiver.sol	0	100	0	50	10
contracts/interfaces/	100	100	100	100	
ILiquidMint.sol	100	100	100	100	
All files	**98.04**	**91.67**	**92.59**	**98.13**	

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

```
c3b5c6e78c406af3fe8aa92060a678b789746de165e444d36694b9e6eacfb68e ./contracts/LiquidMint.sol
42cfed9a16cf5209a9b72a44215c4f9a2ca5667a27844d48b0785e3fc8dea43b ./contracts/interfaces/ILiquidMint.sol
```

Tests

```
30ab3a197c7098a61bbaaae3ffd305cc62c261a6a4679d8b4a3f228145aaba6a ./test/utils/contracts.ts
2c2061e2f1af21b844004909e6c29381423cd39d5ac251336bd6e299cce5a588 ./test/utils/fixtures.ts
ada7a0548dfdfb1e31a6adafdb0ffbc81d2d123acb8374144633337b597247a9 ./test/utils/helpers.ts
766c211dc3c2d4534d4f9c58c04c310dbb6003043ed9f1cff28cf7bedf233d83 ./test/liquid-mint/LiquidMint.ts
```

Changelog

- 2021-11-11 - Initial report
- 2021-12-20 - Reaudit update (97436b1)

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.