Quantstamp Security Assessment Certificate

October 16th 2020 — Quantstamp Verified

SkyWeaver2

This smart contract audit was prepared by Quantstamp, the protocol for securing smart contracts.

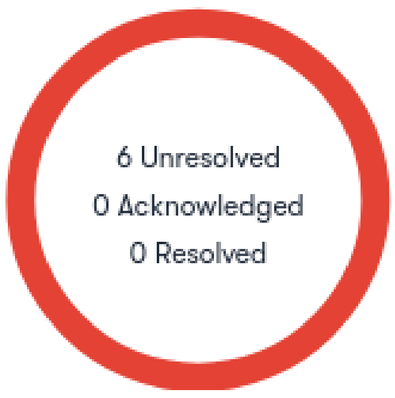
Executive Summary

Type	Audit
Auditors	Luís Fernando Schultz Xavier da Silveira, Security Consultant Kacper Bąk, Senior Research Engineer Martin Derka, Senior Research Engineer
Timeline	2020-10-05 through 2020-10-16
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	Medium
Test Quality	High

Source Code

Repository	Commit
SkyWeaver-contracts	6339e6e7...

Total Issues	**6** (0 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	0 (0 Resolved)
Low Risk Issues	**3** (0 Resolved)
Informational Risk Issues	**3** (0 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

During this audit, Quantstamp has identified 3 low-severity issues. The scope of the audit are the contracts:

- TieredOwnable;
- ConquestEntriesFactory;
- FreemintFactory;
- RewardFactory; and
- Conquest.

The code quality is good and we believe the amount of effort to ready it for production is not high.

ID	Description	Severity	Status
QSP-1	Confusing Entry Token Amount Arithmetic	Low	Unresolved
QSP-2	Incorrect Implementation of `supportsInterface` in `FreemintFactory`	Low	Unresolved
QSP-3	Privileged Roles and Ownership	Low	Unresolved
QSP-4	Unlocked Pragma	Informational	Unresolved
QSP-5	Lack of Flexibility in Conquest Rewarding	Informational	Unresolved
QSP-6	The `arcadeumCoin` Contract can mint arbitrarily many Entry Tokens	Informational	Unresolved

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 0.6.13
- Mythril v0.22.10

Steps taken to run the tools:

1. Installed the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither path/to/contract`
3. Installed the Mythril tool: `pip3 install mythril`
4. Ran the Mythril tool on each contract: `myth analyze path/to/contract`

Findings

QSP-1 Confusing Entry Token Amount Arithmetic

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `ConquestEntriesFactory.sol`

Description: As they are, the contracts don't have an actual security issue with this, but there are a few mistakes in the entry token arithmetic that cancel each other out. This could be a trap for future contract development if not properly understood.
First, according to the documentation, the constant `ARC_REQUIRED` should be `100 * 10**ARC_DECIMALS`. Also, the name `n_entries` implies it is the number of times a player can enter the conquest, but in the code it is actually around `10**ENTRY_DECIMALS` times that. This can be seen by the lack of a multiplication by `10 ** ENTRY_DECIMALS` in line 171.
This works out because line 150 doesn't divide by `10**CARD_DECIMALS` and because line 159 is affected by the wrong value of `ARC_REQUIRED`. However, this relies on the assumption that `ENTRIES_DECIMALS` and `CARD_DECIMALS` are both 2, so it is quite fragile.

Recommendation: Make `ARC_REQUIRED = 100 * 10**ARC_DECIMALS`, divide `n_entries` by `10**CARD_DECIMALS` outside the loop in line 152 and multiply `n_entries` by `10**ENTRIES_DECIMALS` in line 171.

QSP-2 Incorrect Implementation of `supportsInterface` in `FreemintFactory`

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `FreemintFactory.sol`

Description: The contract `FreemintFactory` does not implement the interface `IERC1155TokenReceiver` but its `supportsInterface` function signals it does. This can lead to issues with interacting contracts.

Recommendation: Remove line 6 and replace lines 81 and 82 by `return interfaceID == type(IERC165).interfaceId;`.

QSP-3 Privileged Roles and Ownership

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `RewardFactory.sol`

Description: We understand care was taken to limit the powers of owner users, for instance by having multiple owner tiers. However, owners of the highest tier have a great deal of control over tokens, including enabling unlimited token minting (file `RewardFactory`, function `updatePeriodMintLimit`) and token withdrawal from contract `ConquestEntriesFactory` (function `withdraw`).

Recommendation: We recommend informing contract users of the powers of high-tier owners as appropriate.

QSP-4 Unlocked Pragma

Severity: *Informational*

Status: Unresolved

File(s) affected: `TieredOwnable.sol`, `FreemintFactory.sol`, `RewardFactory.sol`, `ConquestEntriesFactory.sol`, `Conquest.sol`

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.4.*`. 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 files onto a specific Solidity version.

QSP-5 Lack of Flexibility in Conquest Rewarding

Severity: *Informational*

Status: Unresolved

File(s) affected: `Conquest.sol`

Description: The function `exitConquest` has rather stringent requirements on the form of its input. We are aware this is a feature to limit the power of low-tier owners. However, it removes future opportunities to vary the conquest rewarding system without deploying a new contract. Likewise, the conquest entry fee is fixed.

Recommendation: We recommend the evaluation of this risk. If change is desired, instead of removing the checks, add new functionality to allow high-tier owners to add new valid input forms to the contract. The `exitConquest` function would then verify its input is of a valid form.
One possible implementation is with a `mapping(uint256 => mapping(uint256 => bool)) public validInputForm`. The `exitConquest` function would then check if `validInputForm[_silverIds.length][_goldIds.length]` is true.
The same reasoning can be applied to allow high-tier owners to change the conquest entry fee.

QSP-6 The `arcadeumCoin` Contract can mint arbitrarily many Entry Tokens

Severity: *Informational*

Status: Unresolved

File(s) affected: `ConquestEntriesFactory.sol`

Description: If the `arcadeumCoin` contract is compromised or malicious, it can send arbitrarily many tokens to contract `ConquestEntriesFactory`, which will mint a proportional amount of entry tokens.

Recommendation: We recommend auditing the `arcadeumCoin` contract.

Automated Analyses

Slither

Slither pointed out the following functions could be made external in order to save gas:

- `TieredOwnable.assignOwnership`
- `TieredOwnable.getOwnerTier`
- `ConquestEntriesFactory.onERC1155Received`
- `Conquest.onERC1155Received`

Mythril

Mythril found no issues.

Code Documentation

1. It would be helpful to explain the access control mechanism that the Skyweaver assets contract uses to allow the factory contracts to mint tokens without allowing other users to do so (function `batchMint` in `FreemintFactory.sol` and `RewardFactory.sol`).

2. It would also be helpful to explain how the `Conquest` contract becomes an owner of tier at least one of contracts `silverCardFactory` and `goldCardFactory`.

3. File `Conquest.sol`, line 47: no factories are actually created.

4. File `ConquestEntriesFactory.sol`, line 178: “conquest contract” → `ConquestEntriesFactory`.

5. File `ConquestEntriesFactory.sol`, line 72: shouldn't it be `ConquestEntriesFactory`?

6. File `FreemintFactory.sol`, line 60: “factory manager” rather than “assets”?

7. File `FreemintFactory.sol`, function `supportsInterface`: trailing “s” in line 75; typo “Wheter” in line 78; the wording in line 78 could be improved. Similar comments for the documentation of the same function in files `RewardFactory.sol`, `ConquestEntriesFactory.sol` and `Conquest.sol`.

8. File `Conquest.sol`, line 200: “Number of ones”?

9. File `ConquestEntriesFactory.sol`, line 12: grammar.

10. File `RewardFactory.sol`, line 13: “assets types” → “asset types”.

11. Files `FreemintFactory.sol` (line 44) and `RewardFactory.sol` (line 108): `_tokenIds` → `_ids`.

12. File `Conquest.sol`, line 82: “unsuported” → “unsupported”.

Adherence to Best Practices

1. File `FreemintFactory.sol`: either function `batchMint` or function `skyweaverAssets.batchMint` should make sure that the IDs and amounts arrays have the same length.

2. Function `batchMint` in file `RewardFactory.sol` and function `onERC1155BatchReceived` in file `ConquestEntriesFactory.sol`: check that `_ids.length == _amounts.length`.

3. File `ConquestEntriesFactory.sol`, constructor: consider checking that `_silverRangeMin <= _silverRangeMax` and that `_arcadeumCoinId != _conquestEntryTokenId`.

4. File `RewardFactory.sol`, function `updatePeriodMintLimit`: consider checking that `_newPeriodMintLimit > 0` for consistency with line 51.

5. File `TieredOwnable.sol`, constructor: check that `_firstOwner != address(0)`.

6. File `Conquest.sol`, function `exitConquest`: consider checking that `_user != address(0)`.

7. Decimals are usually of the `uint8` type, yet the contracts use `uint256`.

8. File `RewardFactory.sol`: give a proper error message in line 129.

9. Unnecessary `pragma experimental ABIEncoderV2` in files `IRewardFactory.sol`, `RewardFactory.sol`, `ConquestEntriesFactory.sol` and `Conquest.sol`.

10. There are unused imports: file `IRewardFactory.sol`, lines 4 to 8; file `RewardFactrory.sol`, line 8; file `ISkyweaverAssets`, line 4; and file `FreemintFactory.sol`, line 6.

Test Results

Test Suite Results

All tests are passing.

```
Conquest
  Getter functions
    skyweaverAssets() function
       ✓ should return Factory manager contract address (38ms)
    supportsInterface()
       ✓ should return true for 0x01ffc9a7 (ERC165)
       ✓ should return true for 0x4e2312e0 (ERC1155Receiver)
  Conquest
    Using safeBatchTransferFrom
       ✓ should PASS if caller sends 100 ticket (187ms)
       ✓ should REVERT if token address is not ticket (370ms)
       ✓ should REVERT if ids length is more than 1 (45ms)
       ✓ should REVERT if amounts length is more than 1 (40ms)
       ✓ should REVERT if ID is not ticket (392ms)
       ✓ should REVERT if sent more than 1 ticket (57ms)
       ✓ should REVERT if caller is already in a conquest (145ms)
       ✓ should REVERT if caller tries to do conquest faster than limit (346ms)
    Using safeTransferFrom
       ✓ should PASS if caller sends 100 ticket (97ms)
       ✓ should REVERT if token address is not ticket (262ms)
       ✓ should REVERT if ID is not ticket (295ms)
       ✓ should REVERT if sent more than 1 ticket (54ms)
       ✓ should REVERT if caller is already in a conquest (122ms)
       ✓ should REVERT if caller tries to do conquest faster than limit (348ms)
    When ticket was sent to factory
       ✓ should burn entry sent balance
       ✓ should update user ticket balance
       ✓ should set user as being in conquest
       ✓ should update user conquest count
       ✓ should update next conquest time for user
       ✓ should emit ConquestEntered event (418ms)
      ConquestEntered Event
         ✓ should have user address as `tx.user` field
         ✓ should have correct time value as `tx.nConquests` field
  exitConquest()
     ✓ should REVERT if caller is not in conquest
    When user is in conquest
      With rewards
         ✓ should PASS if caller is owner (215ms)
         ✓ should PASS if rewards are duplicates (166ms)
         ✓ should PASS if rewards are unsorted (161ms)
         ✓ should REVERT if silver reward is not silver (69ms)
         ✓ should REVERT if gold reward is not gold (79ms)
         ✓ should PASS if rewards are unsorted (154ms)
         ✓ should REVERT if caller is not owner
         ✓ should REVERT if trying to mint more than possible amount (337ms)
        When exitConquest was called
           ✓ should update user conquest status
           ✓ should update user silver cards balance
           ✓ should update user gold cards balance
      No rewards
         ✓ should PASS if caller is owner (98ms)
         ✓ should PASS if rewards are duplicates (191ms)
         ✓ should PASS if rewards are unsorted (220ms)
         ✓ should REVERT if silver reward is not silver (65ms)
         ✓ should REVERT if gold reward is not gold (83ms)
         ✓ should PASS if rewards are unsorted (211ms)
         ✓ should REVERT if caller is not owner
         ✓ should REVERT if trying to mint more than possible amount (382ms)
        When exitConquest was called
           ✓ should update user conquest status
           ✓ should update user silver cards balance
           ✓ should update user gold cards balance

ConquestEntriesFactory
  Getter functions
    skyweaverAssets() function
       ✓ should return Factory manager contract address
    supportsInterface()
       ✓ should return true for 0x01ffc9a7 (ERC165)
       ✓ should return true for 0x4e2312e0 (ERC1155Receiver)
  ConquestEntriesFactory
    safeTransferFrom() with Silver Cards
       ✓ should PASS if caller sends assets (2522ms)
       ✓ should PASS if receiver is specified (2484ms)
       ✓ should REVERT if asset is not silver card or arc (157ms)
      When assets were sent to factory
        with sender being recipient
           ✓ should leave factory silver cards balance to 0 (57ms)
           ✓ should update users silver cards balance (71ms)
           ✓ should update recipient conquest entries balance
           ✓ should not increase entry balance of factory
        with sender NOT being recipient
           ✓ should leave factory silver cards balance to 0 (68ms)
           ✓ should update users silver cards balance (62ms)
           ✓ should update recipient conquest entries balance
           ✓ should not increase entry balance of factory
    safeBatchTransferFrom() with Silver Cards
       ✓ should PASS if caller sends assets (323ms)
       ✓ should PASS if receiver is specified (279ms)
       ✓ should REVERT if asset is not silver card or arc (283ms)
      When assets were sent to factory
        with sender being recipient
           ✓ should leave factory silver cards balance to 0 (56ms)
           ✓ should update users silver cards balance (73ms)
           ✓ should update recipient conquest entries balance
           ✓ should not increase entry balance of factory
        with sender NOT being recipient
           ✓ should leave factory silver cards balance to 0 (56ms)
           ✓ should update users silver cards balance (75ms)
           ✓ should update recipient conquest entries balance
           ✓ should not increase entry balance of factory
    safeTransferFrom() with ARC
       ✓ should PASS if caller sends assets (82ms)
       ✓ should PASS if receiver is specified (73ms)
       ✓ should REVERT if asset is not silver card or arc (156ms)
      When assets were sent to factory
        with sender being recipient
           ✓ should update factory arc balance
           ✓ should update user arc balance
           ✓ should update recipient conquest entries balance
           ✓ should not increase entry balance of factory
        with sender NOT being recipient
           ✓ should update factory arc balance
           ✓ should update user arc balance
           ✓ should update recipient conquest entries balance
           ✓ should not increase entry balance of factory
    safeBatchTransferFrom() with ARC
       ✓ should PASS if caller sends assets (97ms)
       ✓ should PASS if receiver is specified (83ms)
       ✓ should REVERT if asset is not silver card or arc (232ms)
      When assets were sent to factory
        with sender being recipient
           ✓ should update factory arc balance
```

```
            ✓ should update user arc balance (69ms)
            ✓ should update recipient conquest entries balance (61ms)
            ✓ should not increase entry balance of factory
          with sender NOT being recipient
            ✓ should update factory arc balance
            ✓ should update user arc balance
            ✓ should update recipient conquest entries balance
            ✓ should not increase entry balance of factory
      withdraw() function
         ✓ should PASS if caller is owner (151ms)
         ✓ should REVERT if caller is not owner
         ✓ should REVERT if recipient is 0x0
        When ARC is withdrawn
           ✓ should update factory ARC balance
           ✓ should update recipient ARC balance

TieredOwnable
  register() function
     ✓ should PASS if caller is owner (81ms)
     ✓ should REVERT if caller is not owner
    When transaction was registered
       ✓ should not store the original transaction hash
       ✓ should set triggerTime to blocktime + delay
       ✓ should prevent registering the same transaction id (45ms)
       ✓ should emit TransactionRegistered event (1178ms)
      TransactionRegistered Event
         ✓ should have Pending as `tx.status` field
         ✓ should have correct time value as `tx.triggerTime` field
         ✓ should have correct target address as `tx.target` field
         ✓ should have correct id as `tx.id` field
         ✓ should have correct data as `tx.data` field
  cancel() function
     ✓ should REVERT if transaction is not registered (56ms)
    When transaction was registered
       ✓ should PASS if caller is owner (77ms)
       ✓ should REVERT if caller is not owner
       ✓ should REVERT if tx is executed already (251ms)
      When transaction was cancelled
         ✓ should overwrite original hash
         ✓ should set status to Cancelled
         ✓ should prevent registering the same transaction id (43ms)
         ✓ should prevent cancelling the same transaction id
         ✓ should not be executable (53ms)
         ✓ should emit TransactionCancelled event (1177ms)
        TransactionCancelled Event
           ✓ should have Cancelled as `tx.status` field
           ✓ should have correct time value as `tx.triggerTime` field
           ✓ should have correct target address as `tx.target` field
           ✓ should have correct id as `tx.id` field
           ✓ should have correct data as `tx.data` field
  execute() function
     ✓ should REVERT if transaction is not registered
     ✓ should REVERT if delay is not passed (150ms)
    When transaction was registered and delay passed
       ✓ should PASS if caller is owner (179ms)
       ✓ should PASS if inner tx is successful (192ms)
       ✓ should REVERT if inner tx fails (138ms)
       ✓ should PASS if caller not owner (117ms)
       ✓ should REVERT if tx was cancelled (141ms)
      When transaction was executed
         ✓ should overwrite original hash
         ✓ should set status to Executed
         ✓ should prevent registering the same transaction id (69ms)
         ✓ should prevent cancelling the same transaction id
         ✓ should not be executable again
         ✓ should emit TransactionExecuted event (1052ms)
        TransactionExecuted Event
           ✓ should have Executed as `tx.status` field
           ✓ should have correct time value as `tx.triggerTime` field
           ✓ should have correct target address as `tx.target` field
           ✓ should have correct id as `tx.id` field
           ✓ should have correct data as `tx.data` field

FreemintFactory
  Getter functions
    getSkyweaverAssets() function
       ✓ should return Factory manager contract address
    supportsInterface()
       ✓ should return true for 0x01ffc9a7 (ERC165)
  batchMint()
     ✓ should PASS if caller is owner (236ms)
     ✓ should REVERT if caller is not owner
    When assets were minted
       ✓ should update user silver cards balance
       ✓ should update recipients assets balance (57ms)

RewardFactory
  Getter functions
    skyweaverAssets() function
       ✓ should return Factory manager contract address
    periodMintLimit() function
       ✓ should return the mint entry ratio
    getAvailableSupply() function
       ✓ should return current available supply
    supportsInterface()
       ✓ should return true for 0x01ffc9a7 (ERC165)
       ✓ should return FALSE for 0x4e2312e0 (ERC1155Receiver)
  updatePeriodMintLimit() function
     ✓ should PASS if caller is super owner (84ms)
     ✓ should REVERT if caller is sub owner
     ✓ should REVERT if caller is not owner
    When periodMintLimit was updated
       ✓ should set periodMintLimit to new periodMintLimit
       ✓ should set availableSupply to new periodMintLimit if higher
       ✓ should not set availableSupply to new periodMintLimit if lower (112ms)
       ✓ should emit PeriodMintLimitChange event (1907ms)
      PeriodMintLimitChanged Event
         ✓ should have old value as `oldMintingLimit` field
         ✓ should have new value as `newMintingLimit` field
  batchMint()
     ✓ should PASS if caller is owner (136ms)
     ✓ should PASS if caller is subowner (231ms)
     ✓ should REVERT false if caller is not owner
     ✓ should PASS if trying to mint exactly the available supply (125ms)
     ✓ should REVERT if trying to mint more than current supply (71ms)
    When cards were minted
       ✓ should update user availableSupply
       ✓ should be able to mint more assets in the same period (166ms)
       ✓ should not be able to exceed minting limit for this period with second mint tx
       ✓ should update user silver cards balance
       ✓ should refresh user availableSupply with new period (310ms)

SkyweaverAssets
  Getter functions
    getMaxIssuances() function
       ✓ should return correct value (103ms)
    getCurrentIssuances() function
       ✓ should return correct value (317ms)
  addMintPermission() function
     ✓ should REVERT if maxRange is 0
     ✓ should REVERT if minRange is lower than maxRange
     ✓ should PASS if range is valid (80ms)
     ✓ should REVERT if caller is not owner (38ms)
```

```
    ✓ should REVERT if range overlaps with locked range (167ms)
    ✓ should pass if range does not overlap with locked range (471ms)
  When mint permission was given
     ✓ should update factory's mint access range
     ✓ should emit MintPermissionAdded event (1554ms)
    MintPermissionAdded Event
       ✓ should have factory address as `factory` field
       ✓ should have range in `range` field
    When mint permission was given for a 2ND time
       ✓ should update factory's mint access range correctly
removeMintPermission() function
   ✓ should REVERT if no range exists
  When mint permission was given
     ✓ should PASS if range exists (85ms)
     ✓ should REVERT if caller is not owner
     ✓ should REVERT if range to delete does not exist (42ms)
    When mint permission was removed
       ✓ should update factory's mint access range
       ✓ should emit MintPermissionRemoved event (2027ms)
      MintPermissionRemoved Event
         ✓ should have factory address as `factory` field
         ✓ should have range in `range` field
    When multiple ranges are permitted
       ✓ should have correct range in `range` field (215ms)
       ✓ should remove the correct range (136ms)
       ✓ should reduce length of range array by 1 (158ms)
       ✓ should move last range to removed range, if different (250ms)
       ✓ should move last range to removed range, if different (#2) (246ms)
       ✓ should simply delete last element if range is last (211ms)
       ✓ should allow delete all ranges and re-order ranges (489ms)
activateFactory() function
   ✓ should PASS if caller is owner (76ms)
   ✓ should REVERT if caller is not owner
  When factory was activated
     ✓ should set factory to active
     ✓ should emit FactoryActivation event (1703ms)
    FactoryActivation Event
       ✓ should have factory address as `factory` field (210ms)
shutdownFactory() function
   ✓ should PASS if caller is owner (68ms)
   ✓ should REVERT if caller is not owner
  When factory was shutdown
     ✓ should set factory to inactive
     ✓ should emit FactoryShutdown event (1620ms)
    FactoryShutdown Event
       ✓ should have factory address as `factory` field (84ms)
lockRangeMintPermissions() function
   ✓ should PASS if caller is owner (90ms)
   ✓ should REVERT if caller is not owner
  When range was locked
     ✓ should push range in `lockedRanges` array (122ms)
     ✓ should emit RangeLocked event (1497ms)
    RangeLocked Event
       ✓ should have correct range as `range` field
setMaxIssuances() function
   ✓ should PASS if caller is owner (91ms)
   ✓ should REVERT if caller is not owner
   ✓ should REVERT if arrays are not the same length
  Wen max issuance is already set
     ✓ should PASS if new max issuance is lower (103ms)
     ✓ should REVERT if new max issuance is same
     ✓ should REVERT if new max issuance is higher
     ✓ should REVERT if new max issuance is 0
batchMint() function
   ✓ should REVERT if called by inactive factory, but authorized IDs (122ms)
   ✓ should REVERT if IDs to mint are not authorized (364ms)
   ✓ should REVERT if NO id ranges are authorized for factory (133ms)
   ✓ should REVERT if only 1 ID to mint is not authorized (84ms)
   ✓ should REVERT if exceeds max issuance (148ms)
   ✓ should PASS if reach exact max issuance (403ms)
   ✓ should update current issuance if max issuance is set (390ms)
   ✓ should NOT update current issuance if max issuance is NOT set (322ms)
   ✓ should PASS if caller is activated and authorized factory (295ms)
   ✓ should PASS if id is in a later range (733ms)
   ✓ should PASS if some ids are in different ranges (2653ms)
   ✓ should REVERT if some ids in different ranges are not sorted (755ms)
  When tokens were minted
     ✓ should mint tokens to recipient (63ms)
mint() function
   ✓ should REVERT if called by inactive factory, but authorized IDs (221ms)
   ✓ should REVERT if IDs to mint are not authorized (608ms)
   ✓ should REVERT if exceeds max issuance (196ms)
   ✓ should PASS if reach exact max issuance (272ms)
   ✓ should update current issuance if max issuance is set (307ms)
   ✓ should NOT update current issuance if max issuance is NOT set (189ms)
   ✓ should PASS if caller is activated and authorized factory (135ms)
   ✓ should PASS if id is in a later range (836ms)
  When tokens were minted
     ✓ should mint tokens to recipient

TieredOwnable
  Getter functions
    ownerTier() function
       ✓ should return expected owner tier (125ms)
  Constructor
     ✓ should not set owner to msg.sender by default (284ms)
  assignOwnership() function
     ✓ should PASS if caller is super owner (102ms)
     ✓ should REVERT if caller is sub owner (183ms)
     ✓ should REVERT if caller is not owner (54ms)
     ✓ should REVERT if address is 0x0
     ✓ should REVERT if address is sender
  When ownerTier was updated
     ✓ should set ownerTier to new ownerTier
     ✓ should emit OwnershipGranted event (2592ms)
    OwnershipGranted Event
       ✓ should have address value as `owner` field
       ✓ should have old tier as `previousTier` field
       ✓ should have new tier as `newTier` field
onlyOwnerTier
   ✓ should REVERT if caller is  has lower tier (215ms)
   ✓ should PASS if caller is  has lower tier (935ms)
```

Code Coverage

Though not complete, code coverage is quite good. A few more tests should be enough to cover everything.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
conquest/	97.56	88.46	85.71	97.37	
Conquest.sol	97.56	88.46	85.71	97.37	85
factories/	93.75	72.73	83.33	93.65	
ConquestEntriesFactory.sol	93.75	71.43	83.33	93.55	93,164
FreemintFactory.sol	85.71	50	80	85.71	70
RewardFactory.sol	96	83.33	85.71	96	96
interfaces/	100	100	100	100	
IRewardFactory.sol	100	100	100	100	
ISkyweaverAssets.sol	100	100	100	100	
mocks/	81.82	100	92.31	81.82	
ERC1155Mock.sol	100	100	100	100	
FactoryMock.sol	100	100	100	100	
OwnableMock.sol	50	100	80	50	12,20
TieredOwnableMock.sol	100	100	100	100	
tokens/	95.65	100	85	95.65	
SkyweaverAssets.sol	95.65	100	85	95.65	182,384,392
utils/	88.24	90.91	85.71	89.19	
DelayedOwner.sol	100	100	100	100	
Ownable.sol	42.86	50	50	50	36,37,38,45
TieredOwnable.sol	100	100	100	100	
All files	**93.61**	**88.04**	**86.11**	**93.58**	

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

```
f817548bcb9ff1930a902e9c33af9cf20763a12c38cb58271fd5a60aa37e9d68 ./contracts/utils/TieredOwnable.sol
f4e19d0ca1a9fa2440c880d946460a7b6b8721420bdbe49e7be41b8cc1b895e0 ./contracts/factories/ConquestEntriesFactory.sol
9c48a04678440f5b2d3d7b9ad051de3f1c33307135f05b78d5a940e9084117e5 ./contracts/factories/FreemintFactory.sol
30da8865d8f52634758bb02b3486350fb2a590fb20353401940990e671f07b9c ./contracts/factories/RewardFactory.sol
a5820756af2f852eba18be3f392b03d55a66ba32e3da8bdf72e83ebb83b677ab ./contracts/conquest/Conquest.sol
```

Tests

```
6f0c58d06d2ad3e2c181eea622b325c96c36dbfae8108c82668c0a8c08d9669c ./test/Conquest.spec.ts
a940579a7138a9f92ed1c0f217b94ec48e2bc1ad73dc041ea5cea017799b50d0 ./test/ConquestEntriesFactory.spec.ts
e6b64e58002b60330fcca980c99f8fa8fef67a3de99aa924c0f90b07a09c0880 ./test/DelayedOwner.spec.ts
6a91f2684a740a1ef02ec4355b19481f6433f99096002bdcb828393dd0db0060 ./test/FreemintFactory.spec.ts
7dd016d39015629e1ca559b6bcf7108e2095831c6956b66a00bd6c31b3396ed4 ./test/RewardFactory.spec.ts
05769598f83dc46bbb940299756448685f8ded4d0425a5301e49b0e7ed41091b ./test/SkyweavetAssets.spec.ts
10a994d0d4747cd4ba18372939b413df8fea160125ee0531d7a497b1a6f5fd1f ./test/TieredOwnable.spec.ts
970c659eb13ee8c2016fce46a156dbedc47a950dd6c97748bc73ff48f39df467 ./test/utils/contract.ts
079fc124f561d6099fb1a4a2298708963dbef05405b4b962712c3d0eb9134b3d ./test/utils/helpers.ts
27381cacec1e97994991a4f0395b68f4db8363decd378b75669132283028b409 ./test/utils/index.ts
```

Changelog

- 2020-10-16 - Initial 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.