Quantstamp Security Assessment Certificate

April 12th 2022 — Quantstamp Verified

Omnuum

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

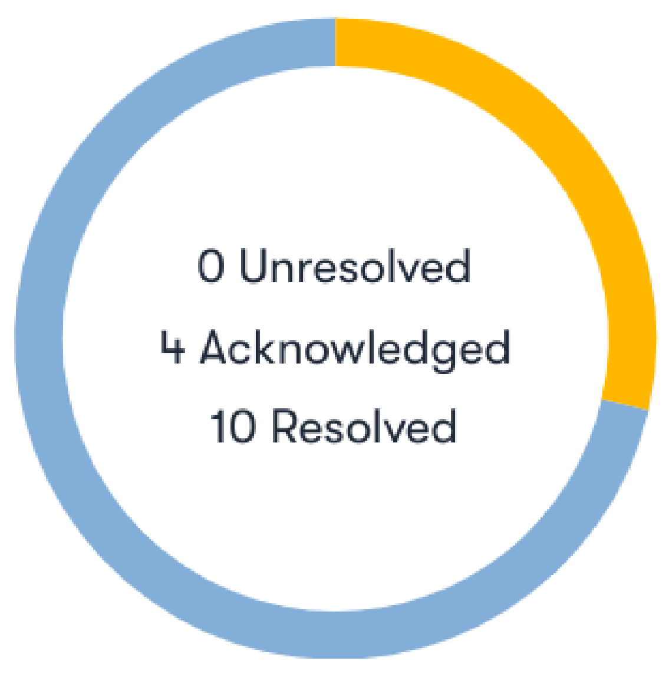
Executive Summary

Type	NFT
Auditors	Marius Guggenmos, Senior Research Engineer Rabib Islam, Research Engineer David Knott, Senior Research Engineer
Timeline	2022-03-14 through 2022-03-18
EVM	Arrow Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	GitBook
Documentation Quality	Medium
Test Quality	High

Source Code

Repository	Commit
omnuum-contracts (initial audit)	3ffb9d9
omnuum-contracts (reaudit)	9448a28
omnuum-contracts (reaudit 2)	312942c

Total Issues	**14** (10 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	**2** (2 Resolved)
Low Risk Issues	**5** (5 Resolved)
Informational Risk Issues	**7** (3 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

After initial audit:
Quantstamp has performed an audit of the Omnuum repository for commit hash `3ffb9d9`. Overall, the code base is relatively small and contains a lot of third-party code that was not part of the audit. The code is written well but lacks documentation in many places. The audit resulted in 12 findings and an additional 7 best practice violations, described below. We can confirm that all of the tests except for one are passing. The failing test is related to a revert message, and thus the failure is not critical. We recommend that all issues reported in this document be addressed.
After reaudit:
Quantstamp has checked the commit hash `9448a28` and has determined that 9 issues have been resolved (that is either fixed or mitigated) and 3 issues have been acknowledged by the Omnuum team. More details regarding each of the issues are provided in the update messages below each issue recommendation.
Additionally we found 2 more issues that were introduced in the newly added code that we recommend addressing.
After 2nd reaudit:
Quantstamp has checked the commit hash `312942c` and has determined that the 2 issues that were found in the first reaudit are now either fixed or acknowledged.

ID	Description	Severity	Status
QSP-1	Freezable Mutlisignature Wallet	^ Medium	Fixed
QSP-2	Missing Input Address Validation	⌄ Low	Fixed
QSP-3	Ignored Return Value of `IERC20.transfer`	⌄ Low	Fixed
QSP-4	Unlocked Pragma	⌄ Low	Fixed
QSP-5	Promiscuous Access Control	⌄ Low	Fixed
QSP-6	URI Reveal Frontrunnable	⌄ Low	Fixed
QSP-7	Renouncing Ownership is Possible	○ Informational	Fixed
QSP-8	Temporary Exchange Rate Arbitrage	○ Informational	Acknowledged
QSP-9	Temporary Exchange Rate Update Missing Validation	○ Informational	Acknowledged
QSP-10	Missing Contract Validation	○ Informational	Fixed
QSP-11	Block Timestamp Manipulation	○ Informational	Acknowledged
QSP-12	Missing Data Validation in `setDiscountRate`	○ Informational	Fixed
QSP-13	Multisig Consensus Ratio Not Enforced Correctly	^ Medium	Fixed
QSP-14	Gas Optimization: CAManager Should Accept Roles Pre-hashed	○ 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.

DISCLAIMER: The OmnuumExchange contract is not really functional yet and will be upgraded in the future. We therefore reported issues for this contract with low severity. Similarly, the randomness provided by OmnuumVRFManager is currently unused and the hooks that consume it will be added in a future upgrade.

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

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

Methodology

The Quantstamp auditing process follows a routine series of steps:

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

Toolset

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

Setup

Tool Setup:

- Slither v0.8.2

Steps taken to run the tools:

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

Findings

QSP-1 Freezable Mutlisignature Wallet

Severity: *Medium Risk*

Status: Fixed

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

Description: Software engineering has the concept of a "bus factor," which is derived from the phrase "if they get hit by a bus." It's a measurement of risk based on a given system's fault tolerance. The `OmnuumWallet` contract has a "bus factor" of one as it requires all `owners` to approve withdrawals and doesn't include the functionality to update owners.

Exploit Scenario:

1. An `OmnuumWallet` owner is either unable (e.g. having lost their key) or unwilling to use their key.
2. All funds are locked in the `OmnuumWallet` contract.

Recommendation: Change the required number of approvals needed to execute a withdrawal to be less than the total number of owners. Furthermore, add functionality to add and update `owners` so that they can be added or replaced for business and security reasons.

Update: Omnuum's multisignature wallet implementation has been modified to require a threshold of owners' signatures instead of requiring all owners' signatures.

QSP-2 Missing Input Address Validation

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts/OmnuumCAManager.sol`, `contracts/OmnuumExchange.sol`, `contracts/OmnuumMintManager.sol`, `contracts/OmnuumNFT1155.sol`, `contracts/OmnuumVRFManager.sol`

Description: Several functions do not check whether address arguments are equal to `address(0)`. This can lead to unexpected behavior or costly re-deployments.

- contracts/OmnuumCAManager.sol:
  - `registerContract`
- contracts/OmnuumExchange.sol:
  - `initialize`
- contracts/OmnuumMintManager.sol:
  - `setDiscountRate`
- contracts/OmnuumNFT1155.sol:
  - `initialize`
- contracts/OmnuumVRFManager.sol:
  - `constructor`
  - `requestVRFOnce`

Recommendation: Check that address parameters are not equal to `address(0)`.

Update: Input address checks have been added to the relevant functions.

QSP-3 Ignored Return Value of `IERC20.transfer`

Severity: *Low Risk*

Status: Fixed

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

Description: Depending on the implementation, a failed `IER20.transfer` might only return `false` instead of reverting. This would lead to the execution continuing under the assumption that tokens have been transferred when they were not.

Recommendation: Use OpenZeppelin's `safeTransfer` function when transferring tokens.

Update: The `transfer` call has been replaced with `safeTransfer`, which checks the return value and reverts if the value is `false`.

QSP-4 Unlocked Pragma

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts/*`

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity >=0.7.0 <0.9.0`. The greater/less than or equal (>=, <=) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version and above, hence the term "unlocked". Given that only certain versions of the Solidity compiler are recommended for production use it is dangerous to use this wide range of versions in the pragma. Moreover, the compiled contracts with one version of the compiler might behave slightly differently than when compiled with a different version of the compiler. Allowing compilation by Solidity versions prior to `0.8.0` is especially dangerous since the code does not use the `SafeMath` library for arithmetic operations and Solidity only has built-in overflow checks in versions `0.8.0` onwards.

Recommendation: For consistency and to prevent unexpected behavior in the future, it is recommended to remove greater/less than or equal signs and to lock the file onto a specific Solidity version. Some versions that fall in the existing range and are recommended for production are: 0.8.8, 0.8.9, 0.8.10.

Update: The Solidity version for all of the contracts except `Ownable` has been pinned to version `0.8.10`.

QSP-5 Promiscuous Access Control

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts/OmnuumExchange.sol`, `contracts/OmnuumVRFManager.sol`

Description: Both the `OmnuumExchange` and `OmnuumVRFManager` check whether an address is registered in the `OmnuumCAManager` to validate the usage of the following functionality:

- `contracts/OmnuumExchange::33(updateTmpExchangeRate)`
- `contracts/OmnuumExchange::39(exchangeToken)`
- `contracts/OmnuumExchange::51(withdraw)`
- `contracts/OmnuumVRFManager.sol::43(requestVRF)`
- `contracts/OmnuumVRFManager.sol::54(requestVRFOnce)`

Given that each of the above functions impacts the movement of funds, allowing any address registered in `OmnuumCAManager` to access them increases the risk of misuse.

Recommendation: Restrict access to the specified functions to be as granular as possible.

Update: A role-based access control system that allows for more fine-grained permissions has been introduced.

QSP-6 URI Reveal Frontrunnable

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts/OmnuumNFT1155.sol`

Description: When an NFT project owner calls `contracts/OmnuumNFT1155.sol::110(setUri)`, their transaction will appear in the Ethereum mempool before being mined. Anyone watching the Ethereum mempool can see the URI and buy or sell the revealing NFT project's tokens before counter-parties have access to the reveal.

Recommendation: Add end-user documentation that informs NFT project token holders that the reveal may be front-run and so they should be wary of transacting with their NFT tokens close to the reveal.

Update: The documentation now recommends not listing NFTs close to the reveal date.

QSP-7 Renouncing Ownership is Possible

Severity: *Informational*

Status: Fixed

File(s) affected: `contracts/OmnuumCAManager.sol`, `contracts/OmnuumMintManager.sol`, `contracts/OmnuumNFT1155.sol`, `contracts/OmnuumVRFManager.sol`

Description: Several contracts implement OpenZeppelin's `Ownable`, which by default provides the function `renounceOwnership` to relinquish the ownership of the contract. In case it is never planned that the contracts should be without an owner, we recommend overwriting this function to avoid accidentally leaving the contracts without an owner.

Recommendation: Consider whether renouncing the ownership is a valid use case. If it is not, disable the functionality by overwriting `renounceOwnership`.

Update: OpenZeppelin's `Ownable` and `OwnableUpgradeable` contracts have been cloned with the renounce ownership functionality removed.

QSP-8 Temporary Exchange Rate Arbitrage

Severity: *Informational*

Status: Acknowledged

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

Description: The `OmnuumExchange` exchanges `LINK` for `ETH` at a fixed rate and the rate is updated manually. If the actual `LINK` to `ETH` exchange rate becomes higher than the fixed exchange rate, NFT project owners can access Omnuum's Chainlink VRF integration for less than its actual cost at the expense of Omnuum.
Since the integration with the VRF contract is currently not finished, there is no incentive for users to use it, which is why we classified this issue as *informational*.

Recommendation: The project roadmap intends to integrate with a DEX to provide current market rates for the price of `LINK`. Meanwhile, the owners shouldn't put too much `LINK` into the `OmnuumExchange` contract to minimize mispriced VRF usage.

Update:
The Omnuum team acknowledged the issue with the following reponse:

> Until the DEX functionality is integrated into our system, we will allow users to utilize the Chainlink VRF after paying ETH equivalent to LINK through a fixed exchange rate stored in the contract. We are fully aware of the risks you mentioned, so we are trying to set the exchange rate as conservatively as possible and it is also multiplied by the safety ratio (1.5 times), further reducing the risk of higher expenses than actual. In addition, we restricted users to access to Chainlink VRF only for the NFT project owner and can call only once. This means that VRF will be used at a low frequency, so even if the actual exchange rate works against us, we think the loss will be only temporarily in a very small amount.

QSP-9 Temporary Exchange Rate Update Missing Validation

Severity: *Informational*

Status: Acknowledged

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

Description: `contracts/OmnuumExchange.sol::33(updateTmpExchangeRate)` updates the ETH to `LINK` exchange rate but doesn't validate the new exchange rate. The lack of validation allows the exchange rate to be set to 0, which would allow all NFT project owners to access Omnuum's Chainlink VRF integration for free, draining the `OmnuumExchange` of `LINK`. The rate could also be set too high, stopping NFT project owners from using Omnuum's VRF functionality to conduct their reveal.

Recommendation: Add lower and upper bound checks to the new temporary exchange rate before setting it in `contracts/OmnuumExchange.sol::33(updateTmpExchangeRate)`.

Update:
The Omnuum team acknowledged the issue with the following response:

> Since the exchange rate depends on the market, we cannot know it will change to what extent. This is why we think it is risky to set boundaries in the contract by codes. Instead, we decided to write the validation codes in the script that calls the function to set a new exchange rate that does not pass any invalid values.

QSP-10 Missing Contract Validation

Severity: *Informational*

Status: Fixed

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

Description: The `OmnuumCAManager` manages the addresses of all contracts in the Omnuum protocol, though it doesn't validate that addresses being registered are smart contracts. This can lead to non-intuitive behavior where an externally owned account (EOA) can access functionality that uses the `OmnuumCAManager` for validation.

Recommendation: Use OpenZeppelin's isContract to validate that all addresses registered are contract addresses.
If EOAs are meant to be registered, change `OmnuumCAManager` to not specifically reference "contracts".

Update: `isContract` checks have been added as recommended.

QSP-11 Block Timestamp Manipulation

Severity: *Informational*

Status: Acknowledged

File(s) affected: `contracts/OmnuumMintManager.sol`, `contracts/TicketManager.sol`

Description: Projects may rely on block timestamps for various purposes. However, it's important to realize that miners individually set the timestamp of a block, and attackers may be able to manipulate timestamps for their purposes by up to 900 seconds. In `contracts/OmnuumMintManager.sol::76` and `contracts/TicketManager.sol::62` the value of `block.timestamp` is used to check whether the deadline for a project's minting has passed. If a miner manipulates the timestamp of a block near the deadline, a user may mint past a NFT project owner's deadline.

Recommendation: Clarify to NFT project owners and end-users via publicly facing documentation that their NFT project's minting deadlines should be expected to be valid up to 900 seconds after they've been passed.

Update: The Omnuum team were already aware of this issue and intend to provide documentation to inform users about it.

QSP-12 Missing Data Validation in `setDiscountRate`

Severity: *Informational*

Status: Fixed

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

Description: `contracts/OmnuumMintManager.sol::42(setDiscountRate)` sets a `discountRate` which is subtracted from the `baseRate` in `contracts/OmnuumNFT1155.sol::57`. `setDiscountRate` contains no validation and so the `discountRate` may be set to a value greater than the `baseRate`, which would result in an underflow induced revert on `contracts/OmnuumNFT1155.sol::57`.

Recommendation: Check that the new `discountRate` is less than or equal to the `baseRate` prior to setting it in `contracts/OmnuumMintManager.sol::42(setDiscountRate)`.

Update: The fee system has been simplified and the subtraction that led to this issue has been removed.

QSP-13 Multisig Consensus Ratio Not Enforced Correctly

Severity: *Medium Risk*

Status: Fixed

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

Description: The function `requiredVotesForConsensus` computes the number of votes required for a request to pass. This is implemented by performing integer division, which means the result will be rounded down. Depending on the configuration, fewer votes than required by the consensus ratio can pass requests.

Exploit Scenario:

1. `totalVotes()` returns 3 and `consensusRatio` is set to 50.
2. `requiredVotesForConsensus()` truncates 1.5 down to 1.
3. A request is executed with one signature instead of two.

Recommendation: Compute the mathematical ceiling using OpenZeppelin's `ceilDiv` to ensure the ratio of votes is always greater than or equal to the consensus ratio.

Update: The consensus ratio computation now uses the `Math.ceilDiv` function as recommended.

QSP-14 Gas Optimization: CAManager Should Accept Roles Pre-hashed

Severity: *Informational*

Status: Acknowledged

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

Description: The interface for adding, removing and querying for roles currently accepts the role as a string in order to compute the hash for it at runtime. The hash is then used as an index into a mapping. This computation could be performed at compile time by changing the function signatures to accept hashed data directly, lowering the gas costs.

Recommendation: Change the API of the functions related to role-based access to accept the hash of a role instead of the role's name as a string.

Update: For compatibility with an off-chain indexing tool, the Omnuum team have decided to use strings for the roles instead of hashes.

Automated Analyses

Slither

Slither reported 258 results, all of which were either identified as false positives or included in the findings of this report.

Code Documentation

1. None of the contracts are consistently commented. Each contract should have a top-level comment that provides a brief overview of its functionality, as well as comments on every non-trivial function that documents what it does. The format of these functions should ideally be in the NatSpec format.
   Update: Comments have been added.

2. Instructions for collecting coverage are missing in the README file.
   Update: Instructions have been added.

3. Typo in `contracts/SenderVerifier.sol`'s `struct Payload`. Member is called `nounce` when it should be `nonce`.
   Update: Member has been renamed.

4. In `contracts/OmnuumVRFManager.sol` there is a comment that says`//RINKBY CHAINLINK`. The comment should be removed as `contracts/OmnuumVRFManager.sol` is going to be deployed to the Ethereum Mainnet as well.
   Update: Comment has been removed.

5. The Blueprint diagram at the top of page 3 in the Omnuum Project Docs says:

   . That the ticket manager will be removed but this removal isn't mentioned anywhere else. Add documentation about the removal process and how it will impact already deployed and future `OmnuumNFT1155.sol` contracts.

   . That the `OmnuumNFT1155.sol` calls `SendVerifier.sol` when Project Process on the top of page 5 says "project owner do reveal process". Update the Blueprint diagram so that it is in line with the owner of a NFT project conducting the reveal.

   Update: The Blueprint diagram has been updated to fix this.

Adherence to Best Practices

1. Many functions should be declared as external instead of public since they are not called by other functions in the contract. Refer to the output of `slither .` to get a list of candidates.
   Update: Most public functions have been changed to external where possible.
2. Tests are written in JavaScript instead of TypeScript with generated types via TypeChain.
   Update: The team acknowledged that TypeScript with TypeChain is a better way of writing tests and will keep this in mind for future development.
3. There are several instances of declaring local variables and incrementing them without an initial assignment. Even though this works since the local variables are initialized with their default value, being explicit about the initial values of variables helps when reading code.
   Update: Local variables are now initialized explicitly.
4. Visibility for contract members should always be specified explicitly.
   Update: Visibility for contract members is now specified explicitly.
5. Add event indexing to all events, particularly the `URI` reveal event declared in `contracts/OmnuumNFT1155.sol` on L19.
   Update: Events are now indexed.
6. Omnuum smart contracts' `require` statements use a mix of error codes and error messages. Choose whether to use either error codes or error messages for uniformity. If error codes are used, add end-user documentation explaining the significance of each error code.
   Update: Error codes are now used consistently throughout the code.

Test Results

Test Suite Results

Out of the 115 tests, all except for one are passing. The single failure is related to a reworded revert string and thus not critical.

Update:
The single test failure has been fixed and the new commit hash has 148 passing tests.

```
OmnuumCAManager
  Security
    ✓ [Revert] Should not initialize after deploy
  [Method] registerContract
    ✓ should register contract
    ✓ should override existing contract at indexedContracts if same topic (50ms)
    ✓ [Revert] only owner can register
    ✓ [Revert] EOA should not be registered
  [Method] removeContract, checkRegistration
    ✓ can remove contract (51ms)
    ✓ should not remove indexed contracts if indexed contract mapping overriden (57ms)
    ✓ [Revert] only owner
  [Method] addRole
    ✓ Should add role to CA
    ✓ [Revert] Cannot add EOA address
    ✓ [Revert] Only owner can add role
  [Method] hasRole
    ✓ Should check address has role
  [Method] removeRole
    ✓ Should remove role from address
    ✓ [Revert] Only owner can add role

OmnuumExchange
  Security
    ✓ [Revert] Should not initialize after deploy
  [Method] getExchangeAmount
    ✓ Get Exchange Amount for token
  [Method] exchangeToken
    ✓ Receive token from exchange
    ✓ [Revert] check sender is omnuum registered contract or address
  [Method] updateTmpExchangeRate
    ✓ should update exchangeRate
    ✓ [Revert] Only omnuum can update rate
  [Method] withdraw
    ✓ Withdraw successfully (38ms)
    ✓ [Revert] Only omnuum can withdraw

OmnuumMintManager
  Security
    ✓ [Revert] Should not initialize after deploy
  [Method] getFeeRate
    ✓ Should get fee rate - basic rate
    ✓ Should get fee rate - special rate
  [Method] changeFeeRate
    ✓ Get Fee Rate
    ✓ Change Fee Rate
    ✓ [Revert] not owner
    ✓ [Revert] Fee rate should be lower than 100%
  [Method] setSpecialFeeRate
    ✓ Set special fee rate of nft contract
    ✓ [Revert] not owner
    ✓ [Revert] Special fee rate should be lower than 100%
  [Method] setPublicMintSchedule
    ✓ Should set public mint schedule
    ✓ [Revert] only owner of collection
  [Method] publicMint
    ✓ Should public mint (46ms)
    ✓ [Revert] cannot mint after end date passed  (40ms)
    ✓ [Revert] not enough money (44ms)
    ✓ [Revert] cannot mint more than max per address (44ms)
    ✓ [Revert] cannot mint more than supply of public mint schedule (80ms)
  [method] mintMultiple
    ✓ Airdrop to multiple address (78ms)
    ✓ [Revert] not owner
    ✓ [Revert] arg length not equal
    ✓ [Revert] NFT remaining quantity is less than requested (86ms)

OmnuumNFT
  Security
    ✓ [Revert] Should not initialize after deploy
  [Method] Public Mint
    ✓ Public mint (68ms)
    ✓ Omnuum should receive fee when mint success (65ms)
    ✓ Omnuum should receive fee when mint success with special fee rate (65ms)
    ✓ [Revert] Prevent CA call to mint
    ✓ [Revert] Payload authenticate fail - (sender, signer) (62ms)
    ✓ [Revert] Cannot mint as public after public mint schedule ended (3060ms)
    ✓ [Revert] Pay less money than required
    ✓ [Revert] Cannot mint more quantity than max quantity per address (public)
    ✓ [Revert] Remaining public mint amount is not enough
  [Method] ticketMint
    ✓ Mint with ticket (51ms)
    ✓ Wallet should receive fee when mint success (49ms)
    ✓ [Revert] Prevent CA call to mint
    ✓ [Revert] Pay less money than required
    ✓ [Revert] Payload authenticate fail - (sender, signer) (42ms)
    ✓ [Revert] Invalid ticket (user, price, groupId) (86ms)
    ✓ [Revert] Time expired ticket (3056ms)
    ✓ [Revert] Minter request more quantity than ticket (149ms)
    ✓ [Revert] Minter request more quantity than total remaining quantity (158ms)
  [Method] mintDirect
    ✓ Should direct mint without payload and ether
    ✓ [Revert] only owner
    ✓ [Revert] Minter request more quantity than total remaining quantity (207ms)
  [Method] setUri
    ✓ Should set uri and reveal
    ✓ [Revert] only owner
  [Method] uri
    ✓ Should return cover uri when it is not revealed
    ✓ Should return base uri when it is revealed
  [Method] withdraw
    ✓ Should withdraw balance (126ms)
    ✓ [Revert] only owner

RevealManager
  [Method] vrfRequest
    ✓ [Revert] only project owner
    ✓ [Revert] Already revealed project

SenderVerifier
  [Method] verify
    ✓ Verify signed by omnuum
    ✓ [Revert] False topic
    ✓ [Revert] False Signer
    ✓ [Revert] False Nonce
    ✓ [Revert] False Sender

TicketManager
  [Method] setEndDate
    ✓ Should set end date
    ✓ [Revert] not owner of NFT
  [Method] verify
    ✓ Should verified as success
    ✓ [Revert] expired ticket
    ✓ [Revert] False Signer
    ✓ [Revert] False NFT
    ✓ [Revert] False Minter
  [Method] useTicket
    ✓ Can use ticket for mint (97ms)
    ✓ [Revert] Cannot mint more than remaining quantity (486ms)

OmnuumVRFManager
  [Method] requestVRF
    ✓ Should request VRF and receive response (local mock)
    ✓ Should request VRF and receive response (rinkeby)
    ✓ [Revert] When link is not enough on exchange contract (logic mock)
    ✓ [Revert] Not Omnuum contract
  [Method] requestVRFOnce
    ✓ Should request VRF and receive response (local mock) (62ms)
    ✓ Should request VRF and receive response (rinkeby)
    ✓ [Revert] only omnuum - reveal manager
    ✓ [Revert] When link is not enough on exchange contract (local mock)
    ✓ [Revert] not enough ether for LINK fee (local mock)
    ✓ [Revert] Already used address (87ms)
  [Method] updateFee
    ✓ Should update fee
    ✓ [Revert] only owner
  [Method] updateSafetyRatio
    ✓ Should update safety ratio
    ✓ [Revert] only owner

Omnuum Multi-sig Wallet
  [Constructor] works correctly
    ✓ Register owner accounts correctly
    ✓ [Revert] Registered only owners, not other account
    ✓ [Revert] Cannot register owners which cannot fulfill minLimitForConsensus
  [Method] receive
    ✓ Can receive ETH
    ✓ Receive Fee correctly
    ✓ Receive Accumulated fee from multiple accounts (135ms)
  [Method] makePayment
    ✓ Receive payment and emit event
    ✓ [Revert] Cannot send zero amount
  [Method] request
    ✓ can make a request by owner (41ms)
    ✓ [Revert] request by now owner
  [Method] approve
    ✓ can be approved by the owner
    ✓ [Revert] if approve by not onwer
    ✓ [Revert] if approve to the request which does not exist
    ✓ [Revert] if approve to the request which is already executed (81ms)
    ✓ [Revert] if approve to the request which is already canceled
    ✓ [Revert] if approve again
  [Method] revoke
    ✓ can be revoked by the owner
    ✓ [Revert] if revoke by not onwer
    ✓ [Revert] if revoke to the request which does not exist
    ✓ [Revert] if revoke to the request which is already executed (50ms)
    ✓ [Revert] if revoke to the request which is already canceled
    ✓ [Revert] if revoke the request which is not been approved
  [Method] cancel
    ✓ can cancel the request only by requester
    ✓ [Revert] if cancel by not owner
    ✓ [Revert] if cancel to the request which does not exist
    ✓ [Revert] if cancel to the request which is already executed (55ms)
    ✓ [Revert] if cancel to the request which is already canceled
  [Method] execute
    ✓ can execute withdrawal
    ✓ can execute add (59ms)
    ✓ can execute remove (62ms)
    ✓ can execute change to other owner (71ms)
    ✓ can execute change the owner level (87ms)
    ✓ [Revert] if execute the request which does not exist
    ✓ [Revert] if execute the request which is already executed
    ✓ [Revert] if execute the request which is already canceled
    ✓ [Revert] if execute by not requester
    ✓ [Revert] if execute before reaching the consensus
    ✓ [Revert] if execute the withdrawal if the contract balance is insufficient (53ms)
    ✓ [Revert] if execute the add new owner if the owner already one of members (54ms)
    ✓ [Revert] if execute the addition of new owner if it is zero address or contract address (45ms)
    ✓ [Revert] if execute the change of owner account that does not satisfy the min number of votes for consensus (379ms)
    ✓ [Revert] if execute the change of owner level that does not satisfy the min number of votes for consensus (181ms)
    ✓ [Revert] if execute the removal of owner that does not exist (50ms)
    ✓ [Revert] if execute the removal of owner that does not satisfy the min number of votes for consensus (70ms)
  [Method] getRequestIdsByExecution
    ✓ get ids (154ms)
  [Method] getRequestIdsByOwner
    ✓ get ids (239ms)
  [Method] getRequestIdsByType
    ✓ get ids (253ms)


148 passing (2m)
```

Code Coverage

Code coverage is high.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	89.38	100	100	
OmnuumCAManager.sol	100	80	100	100	
OmnuumExchange.sol	100	66.67	100	100	
OmnuumMintManager.sol	100	95	100	100	
OmnuumNFT1155.sol	100	75	100	100	
OmnuumVRFManager.sol	100	80	100	100	
OmnuumWallet.sol	100	96.67	100	100	
RevealManager.sol	100	75	100	100	
SenderVerifier.sol	100	100	100	100	
TicketManager.sol	100	100	100	100	
contracts/library/	66.67	50	100	33.33	
RevertMessage.sol	66.67	50	100	33.33	10,14
contracts/mock/	92.86	100	93.33	92.86	
MockExchange.sol	100	100	100	100	
MockLink.sol	100	100	100	100	
MockNFT.sol	83.33	100	80	83.33	25
MockVrfCoords.sol	100	100	100	100	
MockVrfRequester.sol	100	100	100	100	
contracts/utils/	88.89	62.5	90.91	90	
Ownable.sol	75	50	80	77.78	38,39
OwnableUpgradeable.sol	100	75	100	100	
All files	**98.71**	**87.65**	**98.11**	**98.53**	

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

```
ddfffc39ccac88bab8fd730b74566455e4d8e0496898dcd7f99e5cfed8e2bfdd ./contracts/OmnuumExchange.sol
88d61764432d1333e56d601f676a864c4f4cc024997868223a77831268099037 ./contracts/OmnuumMintManager.sol
596f3eb2d1d8a353e597eea28583fb8d14c13ef535e31fe9eae65a96c7b6c4bb ./contracts/OmnuumVRFManager.sol
d8492dd0720d26d53e4363b6adbcf124f7fea3b6fec5b4a6ff2b3214c0f5bcf5 ./contracts/RevealManager.sol
203fb6a462cba09b71d15b159b43ffb335a6f3118a0a686a1968ffa54cf587df ./contracts/SenderVerifier.sol
676c8a3d503585ac99bbe3ee74aaf1c5e323457b2dd4a30d30ae0f161362a5e1 ./contracts/TicketManager.sol
31cc85c298194b95034fe66227424f32d2155479feceb7b00e1f9b8c224f7074 ./contracts/OmnuumCAManager.sol
2c7bb22180d9f0fb1d5de65119569ace54b6c6a0408cd328efee1e241bf86d82 ./contracts/OmnuumNFT1155.sol
4b3195573e4611e45cf00cf83c9877bee2ffc447eb7e1c19dd44ca4b60935fe3 ./contracts/OmnuumWallet.sol
1432dfa8a3373218267ecdf06a4c7d299630c6919f90baf1877aca7d0a4c318a ./contracts/utils/Ownable.sol
0e187181d7ce944c3e1fd5a98b0bd17db64555ddfd9315cc09e4bce84e3a4652 ./contracts/utils/OwnableUpgradeable.sol
ab9f7b44256245c56a640867d14795b965f6a49e291eb76dc420ce0917e05026 ./contracts/mock/MockExchange.sol
1eb6b89e4b4fe8bcbef90b24d52ac265a8b8497268f0f4183621b7c9b52da615 ./contracts/mock/MockLink.sol
215b34b799fafdaad6fc79e1c6f6a02f04ae860d0d83a05aa4ff5257fdc661fa ./contracts/mock/MockNFT.sol
cf7d4e3fd8f3b180aa42fa5cfac2bd9bb697aa28636e15c3315125d04948c9c5 ./contracts/mock/MockVrfCoords.sol
d722a6eafc76e3235bad18f155f6d5f87027f89267dcda6a427d20b56c7d7401 ./contracts/mock/MockVrfRequester.sol
7832d5f93eef6240c8a577063eecfcbed02690677ecdb627895050d75425438e ./contracts/library/RevertMessage.sol
```

Tests

```
1e0053168234ff7b85ecc80420d45e6deba3483dc9584fdfa87dbfbdae30c726 ./test/NFT.test.js
d772f223c250f9c9d7243e8caca52fdebb03b0c7833813111dfb1ded9141cf8b ./test/exchange.test.js
08f5b26046efe6aafeb891b2bbc018d767da02ba8162775312148b913761c01f ./test/mintManager.test.js
0916ef39d1eba0a1c83acfe32468357006785e8a509da9c7c3a665f81bc1e49a ./test/revealManager.test.js
c09932e48329c65b3c08ea0849eb8337259dcf3b4c1dd24e4af6d6a9977f3937 ./test/senderVerifier.test.js
fcfbe10ed4810667bad5eff989dc3f1e0c53acda04747cf5ca6ba4e56bec9c3a ./test/ticketManager.test.js
26a6efb79d4b5c20c6ab9855830f1960cbf1a6480b29caba38487a652fc71a42 ./test/vrfManager.test.js
2ef5f6c2d4fd0407eac75066edf45ef676d1a4c557649244d239e4354fdece64 ./test/caManager.test.js
4f6276adbc0e6565f004ef9831eff6c6ccc32dbf65171e9ef35b9ba29eb1fdc0 ./test/wallet.test.js
ddde0a6a6f753b00ae34a09296831a1453d5dd1fc76b4998baf807e118cfdf65 ./test/etc/mock.js
80f3ae10b0afdf1b146883acb9869a44ffc4cc3531f2e47bfdd2558c534ade9b ./test/etc/util.js
```

Changelog

- 2022-03-21 - Initial report
- 2022-04-01 - Reaudit
- 2022-04-11 - Reaudit (2)

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.