Quantstamp Security Assessment Certificate

February 18th 2022 — Quantstamp Verified

Quadrata Passport

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

Executive Summary

Type	Open Passport Network
Auditors	Souhail Mssassi, Research Engineer Poming Lee, Research Engineer Kacper Bąk, Senior Research Engineer
Timeline	2022-01-13 through 2022-02-17
EVM	London
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Quadrata Technical Document
Documentation Quality	Medium
Test Quality	Undetermined

Source Code

Repository	Commit
passport-contracts	2b0a0d0
passport-contracts	a6fc2ee

Total Issues	**9** (4 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	**2** (2 Resolved)
Low Risk Issues	**6** (2 Resolved)
Informational Risk Issues	0 (0 Resolved)
Undetermined Risk Issues	**1** (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:2 medium-severity, 6 low-severity issues, and 1 undetermined-security issues. We recommend addressing all the issues before deploying the code.
After reaudit: Quantstamp has checked the commit hash `a6fc2ee` and has determined that all the reported issues have been resolved (that is, either fixed or acknowledged) by the Quadrata team. More details regarding each of the issues are provided in the update messages below each issue recommendation.

ID	Description	Severity	Status
QSP-1	`eligibleTokenPayments` Always Set to `True`	^ Medium	Fixed
QSP-2	Usage Of `transfer` Instead Of `safeTransfer`	^ Medium	Fixed
QSP-3	Incompatibility With Deflationary Tokens	⌄ Low	Acknowledged
QSP-4	Race Condition	⌄ Low	Acknowledged
QSP-5	For Loop Over Dynamic Array	⌄ Low	Acknowledged
QSP-6	Missing input verification	⌄ Low	Fixed
QSP-7	Centralization Risk	⌄ Low	Acknowledged
QSP-8	Floating Pragma	⌄ Low	Fixed
QSP-9	Resolving the passport data without calling function `getAttribute`	? Undetermined	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 v0.8.1

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 `eligibleTokenPayments` Always Set to `True`

Severity: *Medium Risk*

Status: Fixed

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

Description: In the `eligibleTokenPayments` function, the function Authorized/Denied a payment method based on the `_isAllowed` parameter, the problem here is that the event `AllowTokenPayment` sends the correct value of the `_isAllowed` but in `L280` it's always set to `true`.

- `contracts/QuadGovernance.sol (L280);`

Recommendation: The `L280` should be changed to this code `eligibleTokenPayments[_tokenAddr] = _isAllowed`

Update: The team has solved the issue by adding `eligibleTokenPayments[_tokenAddr] = _isAllowed` in this commit `ba9012f844b90b85a48ca51f1685e8dfbb970d06`.

QSP-2 Usage Of `transfer` Instead Of `safeTransfer`

Severity: *Medium Risk*

Status: Fixed

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

Description: The ERC20 standard token implementation functions also return the transaction status as a Boolean. It is good practice to check for the return status of the function call to ensure that the transaction is successful. It is the developer's responsibility to enclose these function calls with `require()` to ensure that, when the intended ERC20 function call returns `false`, the caller transaction also fails. However, it is mostly missed by developers when they carry out checks; in effect, the transaction would always succeed, even if the token transfer didn't.

- `contracts/QuadPassport.sol (L350);`
- `contracts/QuadPassport.sol (L364);`

Recommendation: Use the `safeTransfer` function from the `SafeERC20` implementation, or put the transfer call inside an `assert` or `require` verifying that it returned `true`.

Update: The team has solved the issue by adding a require to verify the status of the transfer in this commit `23d0150d095f8390e496fc1f1edca44788d5aa1d`.

QSP-3 Incompatibility With Deflationary Tokens

Severity: *Low Risk*

Status: Acknowledged

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

Description: In the `_doTokenPayment` function, when transferring deflationary tokens, the input amount may not be equal to the received amount due to the charged (and burned) transaction fee. As a result, this may not meet the assumption behind these low-level asset-transferring routines and will bring unexpected balance inconsistencies.

Recommendation: Add necessary mitigation mechanisms to keep track of accurate balances, you can store the value of the balance of contract, after the transfer get the new balance and calculate the difference between the old and the new value of the balance.

Update: The team has acknowledged the issue knowing that the list of eligible payment tokens will not include deflationary tokens only ETH, USDC, USDT, DAI, WBTC are accepted.

QSP-4 Race Condition

Severity: *Low Risk*

Status: Acknowledged

Description: The `governance.mintPrice(),governance.mintPricePerAttribute,governance.revSplitIssuer()`, variables have a setter. If the user checks the value of this variable, then calls the `mintPassport,setAttribute,_doETHPayment` function, and the owner updates the `governance.mintPrice(),governance.mintPricePerAttribute(),governance.revSplitIssuer()`, the order of the transaction might overturn and the user's transaction in this case will be executed with the new variable without him knowing about it.

- `contracts/QuadPassport.sol (L46);`
- `contracts/QuadPassport.sol (L316);`
- `contracts/QuadPassport.sol (L335);`

Recommendation: Add the `mintPrice,mintPricePerAttribute,revSplitIssuer` in the arguments of the `mintPassport,setAttribute,_doETHPayment` functions, then add `require` statements that verify that the value provided in the arguments is the same as the one that is stored in the smart contract.

Update: The team has acknowledged the issue, knowing that all admin functions (ex: changes of price) will be behind a `TimeLock` with a time delay giving the users visibility into a future change of parameters.

QSP-5 For Loop Over Dynamic Array

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `contracts/QuadGovernance.sol`, `contracts/QuadPassport.sol`

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

- `contracts/QuadGovernance.sol (L169);`
- `contracts/QuadPassport.sol (L144);`

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

Update: The team has acknowledged this issue, knowing that a DoS from a large array isn't possible because the array is managed by `Quadrata Governance`.

QSP-6 Missing input verification

Severity: *Low Risk*

Status: Fixed

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

Description: Certain functions lack a safety check in the address. The address-type argument should include a zero-address test, otherwise, the contract's functionality may become inaccessible or tokens may be burned in perpetuity.

- `revSplitIssuer` variable should be verified it's less than 100.
- `contracts/QuadGovernance.sol (L242);`

Recommendation: It's recommended to undertake further validation prior to user-supplied data. The concerns can be resolved by utilizing a whitelist technique or a `modifier`.

Update: The team has solved the issue by adding verification to ensure that `_split` is less than 100 in this commit `44bc267158f3a67901dabe460df5920ddac7da59`.

QSP-7 Centralization Risk

Severity: *Low Risk*

Status: Acknowledged

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

Description: There are some actions that could have important consequences for end-users.

1. The `GOVERNANCE_ROLE` of the platform can modify the code logic at any moment by upgrading the implementation code for `contracts\QuadGovernance.sol` and `contracts\QuadPassport.sol`.
2. An issuer can call `contracts\QuadPassport.sol`: function `burnPassportIssuer` at any moment to remove any passport from the platform at will, even if other issuers issue the passport.

Recommendation: This centralization of power needs to be made clear to the users, especially depending on the level of privilege the contract allows to the owner.

Update: The team has acknowledged the risk, knowing that the time lock will take on the governance role, ensuring that users have enough time to leave the project should they disagree with future changes. This behavior is documented in all the governance functions.

QSP-8 Floating Pragma

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts/QuadGovernance.sol`, `contracts/QuadPassport.sol`, `contracts/QuadPassportStore.sol`

Description: The contract makes use of the floating-point pragma 0.8.0, contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensuring that contracts are not unintentionally deployed using another pragma, such as an obsolete version that may introduce issues in the contract system.

- `contracts/QuadGovernance.sol (L2);`
- `contracts/QuadPassport.sol (L2);`
- `contracts/QuadPassportStore.sol (L2);`

Recommendation: Consider locking the pragma version. It is advised that floating pragma not be used in production. Both `truffle-config.js` and `hardhat.config.js` support locking the pragma version.

Update: The team has resolved the issue by removing the floating pragma in this commit `a6fc2ee9a0e3e4bcd84b4d237839a70b5191dc5b`

QSP-9 Resolving the passport data without calling function `getAttribute`

Severity: *Undetermined*

Status: Acknowledged

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

Description: The Ethereum blockchain is a public blockchain where all the data on it is visible to anyone. Theoretically, it is possible to access and resolve those data through an Ethereum node. If a hacker does that and dumps all the data and puts it online, the financial benefits to the platform and its issuers would be jeopardized.

Recommendation: Ensure that there is no critical information/data stored on the blockchain.

Update: The team has acknowledged the issue, and they claim that no personal or private information will be stored on the public blockchain.

Automated Analyses

Slither

Slither reported the following :

- ignored return value in :(`contracts/QuadPassport.sol#364`, `contracts/test-contracts/DeFi.sol#22`)
- re-entrancy in `QuadPassport._doTokenPayment(bytes32,address,address)`, `RC1967UpgradeUpgradeable._upgradeToAndCallSecure(address,bytes,bool)`, which we classified as false positives.

Adherence to Best Practices

1. In the `allowTokenPayment()` function the contract calls `erc20.totalSupply()` in L278, but It doesn't check if the supply is non-zero, it's better to verify the total supply.

Test Results

Test Suite Results

npx hardhat test

```
QuadGovernance
  initialize
    √ success
  updateGovernanceInPassport
    √ success
    √ fail (not admin)
    √ fail (address zero)
    √ fail (passport not set)
  setTreasury
    √ succeed
    √ fail (not admin)
    √ fail (address zero)
    √ fail (treasury already set)
  setPassportContractAddress
    √ succeed
    √ fail (not admin)
    √ fail (address zero)
    √ fail (passport already set)
  setPassportVersion
    √ succeed
    √ fail (not admin)
    √ fail (version non-incremental)
  setMintPrice
    √ succeed
    √ succeed (price zero)
    √ fail (not admin)
    √ fail (minting price already set)
  setEligibleTokenId
    √ succeed
    √ fail (not admin)
    √ fail (token status already set)
  setEligibleAttribute
    √ succeed (true)
    √ succeed (turn false)
    √ succeed (turn false  - first element)
    √ succeed (getSupportedAttributesLength)
    √ fail (not admin)
    √ fail (attribute status already set)
  setEligibleAttributeByDID
    √ succeed
    √ fail (not admin)
    √ fail (attribute status already set)
  setAttributePrice
    √ succeed
    √ succeed (price 0)
    √ fail (not admin)
    √ fail (price already set)
  setAttributeMintPrice
    √ succeed
    √ succeed (price 0)
    √ fail (not admin)
    √ fail (mint attribute price already set)
  setOracle
    √ succeed
    √ fail (not a valid oracle)
    √ fail (not admin)
    √ fail (oracle already set)
    √ fail (address zero)
  addIssuer
    √ succeed
    √ fail (issuer address (0))
    √ fail (treasury address (0))
    √ fail (not admin)
  setRevSplitIssuer
    √ succeed
    √ succeed (price 0)
    √ fail (not admin)
    √ fail (rev split already set)
  allowTokenPayment
    √ succeed
    √ fail (not admin)
    √ fail (token payment status already set)
    √ fail (address zero)
    √ fail (not ERC20)
  upgrade
    √ succeed
    √ fail (not admin)

QuadPassport
  setGovernance
    √ succeed
    √ fail (not governance contract)
    √ fail (governance already set)
  upgrade
    √ fail (not admin)
    √ succeed

QuadPassport
  burnPassport
    √ success - burnPassport
    √ success - can remint after burn
    √ fail - invalid tokenId
    √ fail - passport non-existent
  burnPassportIssuer
    √ success - burnPassportIssuer
    √ success - can remint after burn
    √ fail - invalid tokenId
    √ fail - passport non-existent
    √ fail - not issuer role

QuadPassport
  calculatePaymentToken
    √ success (AML)
    √ success (COUNTRY)
    √ success (DID)
    √ fail - ineligible payment token
    √ fail - wrong erc20
    √ fail - governance incorrectly set
  calculatePaymentETH
    √ success (AML)
    √ success (COUNTRY)
    √ success (DID)
    √ fail - governance incorrectly set

QuadPassport
  getAttribute
    √ success - getAttribute(DID) - Payable
    √ success - getAttribute for free attribute
```

```
      √ fail - getAttribute(AML) - wallet not found
      √ fail - getAttribute(DID) - wallet not found
      √ fail - insufficient allowance
      √ fail - getAttribute from address(0)
      √ fail - getAttribute ineligible Token Id
      √ fail - getAttribute ineligible attribute (AML)
      √ fail - getAttribute ineligible attribute (Country)
    getAttributeFree
      √ success - getAttributeFree(AML)
      √ success - getAttributeFree(COUNTRY)
      √ fail - getAttributeFree(AML) - wallet not found
      √ fail - getAttributeFree(COUNTRY) - wallet not found
      √ fail - getAttributeFree from address(0)
      √ fail - getAttributeFree ineligible Token Id
      √ fail - getAttributeFree ineligible attribute (AML)
      √ fail - getAttributeFree ineligible attribute (Country)
      √ fail - attribute not free
    getAttributeETH
[doSomethingETH] paymentAmount 500000000000000
[doSomethingETH] msg.value 500000000000000
      √ success - getAttributeETH(DID) - Payable
      √ fail - getAttributeETH(AML) - wallet not found
      √ fail - getAttributeETH(DID) - wallet not found
      √ fail - insufficient eth amount
      √ fail - getAttributeETH from address(0)
      √ fail - getAttributeETH ineligible Token Id
      √ fail - getAttributeETH ineligible attribute (AML)
      √ fail - getAttributeETH ineligible attribute (Country)

  QuadPassport
    mintPassport
      √ success mint
      √ success - mint multiple passports with same DID
      √ success - mint multiple passports with same DID from two issuers
      √ success - mint with mint price (0)
      √ success - aml (high)
      √ success - same wallet, different tokenIds
      √ success - change of issuer treasury
      √ fail - invalid mint Price
      √ fail - invalid tokenId
      √ fail - passport already exists
      √ fail - passport already exists - two diff issuers
      √ fail - invalid hash (wrong aml)
      √ fail - invalid hash (wrong aml)
      √ fail - invalid hash (wrong country)
      √ fail - invalid hash (issuedAt)
      √ fail - invalid hash (wrong TokenId)
      √ fail - using someone else signature
      √ fail - invalid issuer

  QuadPassport
    setAttribute
      √ success - setAttribute(AML)
      √ success - setAttribute(COUNTRY)
      √ success - mintPricePerAttribute(0)
      √ succes - two issuers
      √ succes - change issuer treasury
      √ fail - setAttribute(DID)
      √ fail - passport tokenId invalid
      √ fail - no passport
      √ fail - attribute not eligible
      √ fail - invalid mint price per attribute
      √ fail - signature already used
      √ fail - using someone else signature
      √ fail - not issuer role
      √ fail - invalid sig (tokenId)
      √ fail - invalid sig (attribute type)
      √ fail - invalid sig (attribute value)
      √ fail - invalid sig (issuedAt)
    setAttributeIssuer
      √ success - setAttributeIssuer(AML)
      √ success - setAttribute(COUNTRY)
      √ succes - two issuers
      √ fail - passport tokenId invalid
      √ fail - no passport
      √ fail - attribute not eligible
      √ fail - invalid issuer role
      √ fail - setAttribute(DID)

  QuadPassport
    withdrawETH
      √ sucess - after mint
      √ sucess - after setAttribute
      √ sucess - after getAttribute(Free)
[doSomethingETH] paymentAmount 500000000000000
[doSomethingETH] msg.value 500000000000000
      √ sucess - after getAttribute(Payable)
      √ fail - withdraw to address(0)
      √ fail - withdraw to non valid issuer or treasury
      √ fail - withdraw balance 0
    withdrawToken
      √ sucess - after getAttribute(Free)
      √ sucess - after getAttribute(Payable)
      √ fail - withdraw to address(0)
      √ fail - withdraw to non valid issuer or treasury
      √ fail - withdraw balance 0

·-----------------------------------------------|--------------------------|-------------|-----------------------------·
|              Solc version: 0.8.4              ·  Optimizer enabled: true  ·  Runs: 1000  ·  Block limit: 30000000 gas  |
·-----------------------------------------------|--------------------------|-------------|-----------------------------·
|  Methods                                                                                                               |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  Contract       ·  Method                     ·  Min        ·  Max        ·  Avg        ·  # calls    ·  eur (avg)  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  DeFi           ·  doSomething                ·     117180  ·     196201  ·      191269  ·          39  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  DeFi           ·  doSomethingETH             ·     127315  ·     144415  ·      133015  ·           3  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  DeFi           ·  doSomethingFree            ·      64317  ·      70151  ·       67169  ·          90  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadGovernance ·  addIssuer                  ·      40911  ·      80263  ·       79338  ·         171  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadGovernance ·  allowTokenPayment          ·      58401  ·      58425  ·       58424  ·         165  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadGovernance ·  grantRole                  ·          -  ·          -  ·       56329  ·           2  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadGovernance ·  setAttributeMintPrice      ·      31378  ·      36250  ·       33327  ·           5  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadGovernance ·  setAttributePrice          ·      31377  ·      36249  ·       33813  ·           4  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadGovernance ·  setEligibleAttribute       ·      40140  ·      80293  ·       50558  ·          12  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadGovernance ·  setEligibleAttributeByDID  ·      31204  ·      53116  ·       36682  ·           4  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadGovernance ·  setEligibleTokenId         ·      30822  ·      52734  ·       43969  ·           5  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadGovernance ·  setMintPrice               ·      30437  ·      35309  ·       32386  ·           5  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadGovernance ·  setOracle                  ·      39757  ·      56869  ·       56661  ·         165  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadGovernance ·  setPassportContractAddress ·      35708  ·      52808  ·       52600  ·         165  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadGovernance ·  setPassportVersion         ·          -  ·          -  ·       35269  ·           2  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadGovernance ·  setRevSplitIssuer          ·      30417  ·      35229  ·       32823  ·           4  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadGovernance ·  setTreasury                ·      35709  ·      52809  ·       52602  ·         165  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadGovernance ·  updateGovernanceInPassport ·          -  ·          -  ·       46261  ·           6  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadGovernance ·  upgradeTo                  ·      52165  ·      61696  ·       56931  ·           4  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadPassport   ·  burnPassport               ·          -  ·          -  ·       70840  ·           2  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadPassport   ·  burnPassportIssuer         ·          -  ·          -  ·       74326  ·           2  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadPassport   ·  mintPassport               ·     245657  ·     441869  ·      434960  ·          97  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadPassport   ·  setAttribute               ·     106952  ·     132688  ·      117003  ·           8  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadPassport   ·  setAttributeIssuer         ·      64819  ·      70635  ·       66973  ·           4  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadPassport   ·  withdrawETH                ·          -  ·          -  ·       36620  ·          14  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadPassport   ·  withdrawToken              ·      38940  ·      60852  ·       52448  ·          11  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  USDC           ·  approve                    ·      46226  ·      46238  ·       46237  ·          19  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  USDC           ·  transfer                   ·      51493  ·      51505  ·       51499  ·         200  ·          -  |
·-----------------|-----------------------------|-------------|-------------|-------------|-------------|-------------·
|  Deployments                                  ·                                         ·  % of limit  ·             |
·-----------------------------------------------|-------------|-------------|-------------|-------------|-------------·
|  DeFi                                         ·     573704  ·     573716  ·      573715  ·      1.9 %  ·          -  |
·-----------------------------------------------|-------------|-------------|-------------|-------------|-------------·
|  QuadGovernance                               ·          -  ·          -  ·     2924755  ·      9.7 %  ·          -  |
·-----------------------------------------------|-------------|-------------|-------------|-------------|-------------·
```

```
|  QuadGovernanceV2                          ·          -  ·          -  ·     2931663  ·        9.8 %  ·          -  |
·--------------------------------------------|------------|------------|-------------|--------------|-------------·
|  QuadPassport                              ·          -  ·          -  ·     5157315  ·       17.2 %  ·          -  |
·--------------------------------------------|------------|------------|-------------|--------------|-------------·
|  QuadPassportV2                            ·          -  ·          -  ·     5164128  ·       17.2 %  ·          -  |
·--------------------------------------------|------------|------------|-------------|--------------|-------------·
|  UniswapAnchoredView                       ·          -  ·          -  ·      171043  ·        0.6 %  ·          -  |
·--------------------------------------------|------------|------------|-------------|--------------|-------------·
|  USDC                                      ·          -  ·          -  ·      678744  ·        2.3 %  ·          -  |
·--------------------------------------------|------------|------------|-------------|--------------|-------------·

  165 passing (3m)
```

Code Coverage

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	98.64	92.59	95.35	98.66	
QuadGovernance.sol	100	96.34	100	100	
QuadPassport.sol	97.44	88.75	91.67	97.48	244,245,303
QuadPassportStore.sol	100	100	100	100	
contracts/interfaces/	100	100	100	100	
IQuadPassport.sol	100	100	100	100	
IUniswapAnchoredView.sol	100	100	100	100	
contracts/test-contracts/	100	75	100	100	
DeFi.sol	100	50	100	100	
QuadGovernanceV2.sol	100	100	100	100	
QuadPassportV2.sol	100	100	100	100	
USDC.sol	100	100	100	100	
UniswapAnchoredView.sol	100	100	100	100	
All files	98.78	92.17	96.15	98.79	

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

```
ad3e474d70c8f8b7b91cb815c296aa986d0196532e0125bd994fa9fb9043603a ./contracts/QuadGovernance.sol
654701d78de446d7cc7e74f98bfc4a711d9e9e9ab00ba191781aa425208f6cef ./contracts/QuadPassport.sol
3548d89eacf3a33083e7ef3db132b14baa10d0544d31205d6c69a5ab56344634 ./contracts/QuadPassportStore.sol
47eb3fdac87f1b68c29639c15818f36794bd0a266fa57ca71e4811c3355ba4fd ./contracts/test-contracts/DeFi.sol
8697a3f018ee50e9ae005ec6d647c0a359f29ee00fae168229352a932c85562d ./contracts/test-contracts/QuadGovernanceV2.sol
000abbbfe3901c68a78f8bd98982ba899f43265bc7f242aa8d30cc3d77c6cf5e ./contracts/test-contracts/QuadPassportV2.sol
8a6ee0f4c40ede7eec4ecf050c691ca9575a65784ccabac8bd45bc93e60a2af6 ./contracts/test-contracts/UniswapAnchoredView.sol
0926cda15225f2a08eb1f40120a16f412bed55a412781275499d8acb88e662be ./contracts/test-contracts/USDC.sol
8bf89c9f3d1b3b50cc7711209bff8b788590d91e6d022cf29cf472bf6f710853 ./contracts/interfaces/IQuadPassport.sol
b2d7037277ef341c375353c5a75031782bbcc627439980eca91b7232090c6080 ./contracts/interfaces/IUniswapAnchoredView.sol
```

Tests

```
c23812435902d7cf24f6c08d8b81b37b540ff83f3b535a8209c77032b1c2a139 ./test/utils/deployment_and_init.ts
51d00e8e2c2e3f5368993671c7b5eb6b61146f33bc0e7e0d38d71f88ab2b7cd4 ./test/utils/signature.ts
7d276807431b7cab3c29ddfb442e971cd6bf162bc1a5a24ef7b5f1aaf958b60d ./test/utils/verify.ts
6aec42b38578a740713431e3fc33835181499e1a08efd813020ceda56091fa20 ./test/QuadPassport/test_admin.ts
ef7327923ac3bf13fc1f6d064a2c814db8faa9c6a3f1b303723342748d853ef6 ./test/QuadPassport/test_burn_passport.ts
801280c6a3110321dba71bcf722bc78b498d09f5030954e358716fffaa32384b ./test/QuadPassport/test_calculate_payment.ts
e0f3d5d15d742d3cd00507467cde42bc0a6435fe135608e46f8f8c94dce3c204 ./test/QuadPassport/test_get_attributes.ts
acafbe9df869e545056e28e984e716144e5a40a9496cdd21ec1f9c86846aba9b ./test/QuadPassport/test_mint.ts
ed9f03bdab5811e0475464626cb0a8d40e92e91e1d9b9f5ba0ab29409f6382ae ./test/QuadPassport/test_set_attributes.ts
82ab0eab94033bf87d67e70fc1bccfbf3779bdbcbaf754af042df09b0f1e7508 ./test/QuadPassport/test_withdraw.ts
e73fb70cc214295e31441c48809383bcda563d8dd49317ff53f1cded5aaadf2a ./test/QuadGovernance/test_admin.ts
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855 ./test/QuadGovernance/test_get_price.ts
```

Changelog

- 2022-01-17 - Initial report
- 2022-02-17 - Final report

About Quantstamp

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

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

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

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