Security Assessment

Quadrata Audit (Passport)

Jul 15th, 2022

Table of Contents

Summary

This report has been prepared for Quadrata to discover issues and vulnerabilities in the source code of the Quadrata Audit (Passport) project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Quadrata Audit (Passport)
Platform	EVM Compatible
Language	Solidity
Codebase	https://github.com/QuadrataNetwork/passport-contracts
Commit	ad7c3d96dc04dd7907be4062f23a08a201676a6f

Audit Summary

Delivery Date	Jul 15, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

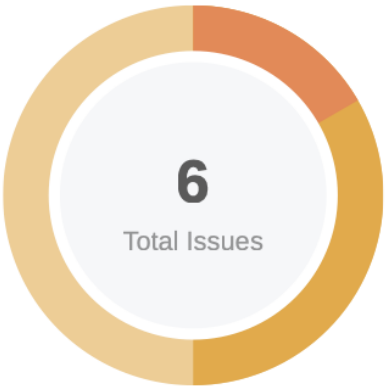
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
Critical	0	0	0	0	0	0	0
Major	1	0	0	0	1	0	0
Medium	2	0	0	2	0	0	0
Minor	3	0	0	1	0	0	2
Optimization	3	0	0	1	0	0	2
Informational	0	0	0	0	0	0	0
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
QGS	contracts/storage/QuadGovernanceStore.sol	2e8cfbebed3797b36822967fdb32dce605151d729bdb2f8b81b5898f8d9c2833
QPS	contracts/storage/QuadPassportStore.sol	db3de5deb7ef4e300cb4f3fedb70aadf17e3b2757a0448d71e1e07625f49ad27
QRS	contracts/storage/QuadReaderStore.sol	93359dde94b7f1fd0ac5e33bb50b99d6ae945758bd15c7cf9d99844e195cd407
QGQ	contracts/QuadGovernance.sol	a2658628ded7456c38dc4b31078385ee8c5791d6e499aa76de56494aa391e32f
QPQ	contracts/QuadPassport.sol	30d9a156aec902c5e062302b0ff6c389207e50ce1d138d1d50a40b2ac16b45e6
QRQ	contracts/QuadReader.sol	52fd9377dd2fb630d7c61b84cf80f30caea75a6b6f33e205c0dd6d2bc7c192a1

Findings



Critical	**0** (0.00%)
Major	**1** (16.67%)
Medium	**2** (33.33%)
Minor	**3** (50.00%)
Informational	**0** (0.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
QGQ-01	No Empty Initializer Constructor For UUPS Upgradable Contracts	Language Specific	Minor	Resolved
QNB-01	Centralization Risks In Passport Contract	**Centralization / Privilege**	**Major**	Mitigated
QNB-02	Lack Of Storage Gap	Language Specific	Medium	Acknowledged
QPQ-01	No Empty Initializer Constructor For UUPS Upgradable Contracts	Language Specific	Minor	Resolved
QRQ-01	Locked Ether And Token	Logical Issue	Medium	Acknowledged
QRQ-02	Incompatibility With Deflationary Tokens	Logical Issue	Minor	Acknowledged

QGQ-01 | No Empty Initializer Constructor For UUPS Upgradable Contracts

Category	Severity	Location	Status
Language Specific	Minor	contracts/QuadGovernance.sol (v1): 32	Resolved

Description

It's recommended the team verify the UUPS implementation contract is initialized correctly. Besides initializing the contract, another way to prevent the implementation contract from being used is to invoke the "_disableInitializers" function in the constructor to automatically lock it when it is deployed. For more information, please refer to: https://docs.openzeppelin.com/upgrades-plugins/1.x/writing-upgradeable#initializing_the_implementation_contract

Recommendation

We advise the client to add an empty initializer constructor to the linked contract.

```
1   constructor() initializer {}
```

Alleviation

The client revised the code and resolved the issue in this commit.

QNB-01 | Centralization Risks In Passport Contract

Category	Severity	Location	Status
Centralization / Privilege	**Major**	contracts/QuadGovernance.sol (v1): 76, 88, 101, 112, 124, 137, 149, 173, 185, 198, 212, 224, 239, 254, 273, 293, 315, 340; contracts/QuadPassport.sol (v1): 62, 91, 177, 304, 319, 335, 349, 369; contracts/QuadReader.sol (v1): 34	Mitigated

Description

In the contract `QuadGovernance` the role `GOVERNANCE_ROLE` has authority over the functions. Any compromise to the `GOVERNANCE_ROLE` account may allow the hacker to take advantage of this authority and

- set QuadPassport treasury wallet to withdraw the protocol fees through `setTreasury()`
- set QuadPassport contract address through `setPassportContractAddress()`
- set the QuadGovernance address in the QuadPassport contract through `updateGovernanceInPassport()`
- set the QuadPassport deployed version through `setPassportVersion()`
- set the price for minting the QuadPassport through `setMintPrice()`
- Set the eligibility status for a tokenId passport through `setEligibleTokenId()`
- set the eligibility status for an attribute type through `setEligibleAttribute()`
- set the eligibility status for an attribute type grouped by DID through `setEligibleAttributeByDID()`
- set the price to update/set a single attribute after owning a passport through `setAttributePrice()`
- set the price to update/set a single business attribute after owning a passport through `setBusinessAttributePrice()`
- set the min price to update/set a single attribute after owning a passport through `setAttributeMintPrice()`
- set the `UniswapAnchorView` oracle through `setOracle()`
- set the revenue split percentage between Issuers and Quadrata Protocol through `setRevSplitIssuer()`
- add a new issuer or update treasury through `setIssuer()`
- delete issuer through `deleteIssuer()`
- deactivate issuer through `setIssuerStatus()`
- authorize or denied a payment to be received in Toke through `allowTokenPayment()`
- authorize to upgrade through `_authorizeUpgrade()`

In the contract `QuadPassport` the role `GOVERNANCE_ROLE` has authority over the functions. Any compromise to the `GOVERNANCE_ROLE` account may allow the hacker to take advantage of this authority and

- authorize to upgrade through `_authorizeUpgrade()`

In the contract `QuadPassport` the role `READER_ROLE` has authority over the functions. Any compromise to the `READER_ROLE` account may allow the hacker to take advantage of this authority and

- increase ETH balance of account through `increaseAccountBalanceETH()`
- increase balance of account through `increaseAccountBalance()`
- allow an authorized readers to get attribute information about a passport holder for a specific issuer through `attributes()`

In the contract `QuadPassport` the role `ISSUER_ROLE` has authority over the functions. Any compromise to the `ISSUER_ROLE` account may allow the hacker to take advantage of this authority and

- issuer can burn an account's Quadrata passport when requested through `burnPassportIssuer()`
- update or set a new attribute for your existing passport through `setAttribute()`
- claim and mint a wallet account Quadrata Passport through `mintPassport()`

In the contract `QuadReader` the role `GOVERNANCE_ROLE` has authority over the functions. Any compromise to the `GOVERNANCE_ROLE` account may allow the hacker to take advantage of this authority and

- authorize to upgrade through `_authorizeUpgrade()`

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

[Quadrata Team]:

The team has switched to a 4/6 Multi-Sign wallet for timelock operation.

The team has revoked the roles of the deployer.

Transaction proof:
https://etherscan.io/tx/0x528687cb1db9d8eca2c38675537e6429ac263a271944e85d52a6d70b2748857f

QNB-02 | Lack Of Storage Gap

Category	Severity	Location	Status
Language Specific	Medium	contracts/QuadGovernance.sol (v1): 15; contracts/QuadPassport.sol (v1): 24; contracts/QuadReader.sol (v1): 23	Acknowledged

Description

The code base contains a transparent_proxy folder, implying the potential to upgrade implementation contracts in the future.

For upgradeable contracts, there must be storage gap to "allow developers to freely add new state variables in the future without compromising the storage compatibility with existing deployments". Otherwise it may be very difficult to write new implementation code. Without storage gap, the variable in child contract might be overwritten by the upgraded base contract if new variables are added to the base contract.

Refer to https://docs.openzeppelin.com/upgrades-plugins/1.x/writing-upgradeable

Recommendation

We advise the client to add appropriate storage gap at the end of upgradeable contracts.

Alleviation

[Quadrata Team]:

The team will not be leaving gaps in our Storage contracts. These storage contracts will always be declared as the rightmost base contract, thereby making them the least base contract. That means so as long as no variables are declared in the child contract, there should be no problem updating the storage contract. The child contracts will not have any state variables and will follow a storage variable scheme similar to Compound.

QPQ-01 | No Empty Initializer Constructor For UUPS Upgradable Contracts

Category	Severity	Location	Status
Language Specific	Minor	contracts/QuadPassport.sol (v1): 20	Resolved

Description

It's recommended the team verify the UUPS implementation contract is initialized correctly. Besides initializing the contract, another way to prevent the implementation contract from being used is to invoke the "_disableInitializers" function in the constructor to automatically lock it when it is deployed. For more information, please refer to: https://docs.openzeppelin.com/upgrades-plugins/1.x/writing-upgradeable#initializing_the_implementation_contract

Recommendation

We advise the client to add an empty initializer constructor to the linked contract.

```
1   constructor() initializer {}
```

Alleviation

The client revised the code and resolved the issue in this commit.

QRQ-01 | Locked Ether And Token

Category	Severity	Location	Status
Logical Issue	Medium	contracts/QuadReader.sol (v1): 405, 433	ⓘ Acknowledged

Description

Due to the existence of division when adding balance to the issuers will result in a small amount of ETHs or Tokens stranded in the contract `QuadPassport`, but the contract `QuadPassport` does not provide any extraction function.

Recommendation

We advise the client to add a function to extract ETHs and tokens from this contract in the contract `QuadPassport`.

Alleviation

[Quadrata Team]:

Issue acknowledged. I won't make any changes for the current version.

The team chose not to handle dust in this current version to optimize gas cost, with any dust amount being negligible. The team will decide at a later stage if it'll be worth adding a dust-sweeping functionality.

QRQ-02 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Logical Issue	Minor	contracts/QuadReader.sol (v1): 417	Acknowledged

Description

When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee. As a result, an inconsistency in the amount will occur and the transaction may fail due to the validation checks.

Recommendation

We advise the client to regulate tokens supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

Alleviation

[Quadrata Team]:

Issue acknowledged. I won't make any changes for the current version.

The limitation regarding deflationary tokens as payment methods is understood. Quadrata Governance will not allow deflationary tokens to be allowed as eligible payment tokens.

Optimizations

ID	Title	Category	Severity	Status
QGQ-02	Costly Operation Inside Loop	Gas Optimization	Optimization	Acknowledged
QGQ-03	Not All Parent Contract Initializing Functions Are Called	Logical Issue	Optimization	Resolved
QPQ-02	Gas Optimization	Gas Optimization	Optimization	Resolved

QGQ-02 | Costly Operation Inside Loop

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/QuadGovernance.sol (v1): 160	Acknowledged

Description

Accessing storage variables in a loop can be costly in terms of gas consumption.

File: contracts/QuadGovernance.sol (Line 160, Function `QuadGovernance.setEligibleAttribute`)

```
                    eligibleAttributesArray.pop();
```

Recommendation

We recommend using a local variable to hold the intermediate result.

Alleviation

[Quadrata Team]:

Issue acknowledged. I won't make any changes for the current version.

This function is only executed as an admin operation. Quadrata is aware of the linear gas cost of looping through a list. however, the gas/cost risk is mitigated due to our use case only requiring a small array of 7 or so attributes. This set of attributes will only be expanded should the Quadrata Governance agree.

QGQ-03 | Not All Parent Contract Initializing Functions Are Called

Category	Severity	Location	Status
Logical Issue	Optimization	contracts/QuadGovernance.sol (v1): 35	Resolved

Description

Contract QuadGoverance extends AccessControlUpgradable, while `__AccessControl_init_unchained()` is not called in the initialize function. Generally the initializer function of an upgradeable contract should always call all the initializer functions of the contracts that it extends.

Recommendation

We advise the client to call `__AccessControl_init_unchained()` in the linked contract.

Alleviation

The client revised the code and resolved the issue in this commit.

QPQ-02 | Gas Optimization

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/QuadPassport.sol (v1): 167	Resolved

Description

The fetching of `attributeType` can be placed in the first loop to save gas.

Recommendation

We advise the client to move the `attributeType` to the first loop.

Alleviation

The client revised the code and resolved the issue in this commit.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND

“AS AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER’S OR ANY OTHER PERSON’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK’S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER’S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED “AS IS” AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK’S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.