Quantstamp Security Assessment Certificate

August 3rd 2021 — Quantstamp Verified

Spring Labs Ky0x

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

Executive Summary

Type	Data Exchange
Auditors	Jose Ignacio Orlicki, Senior Engineer Kacper Bąk, Senior Research Engineer Fayçal Lalidji, Security Auditor
Timeline	2021-06-09 through 2021-07-25
EVM	Berlin
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Ky0x_Documentation.pdf Ky0x Gitbook
Documentation Quality	High
Test Quality	High

Source Code

Repository	Commit
ky0x-contracts	86bd608
ky0x-contracts	39dcbc4

Total Issues	9 (3 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	3 (2 Resolved)
Low Risk Issues	4 (0 Resolved)
Informational Risk Issues	2 (1 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

We have reviewed the code, documentation, and test suite and found several issues of various severities. Overall, we consider the code to be well-written but with insufficient documentation regarding the off-chain behavior of the system. The documentation has technical details but focused mostly on on-chain behavior. Currently, the test suite is in good shape, no tests are failing and the code coverage is very good (above 85%) with the exception of `Ky0xGovernance` that can be improved. We have outlined suggestions to better follow best practices, and recommend addressing all the findings to tighten the contracts for future deployments or contract updates. We also provide suggestions for improvements to follow the best practices. We recommend addressing all the **9** findings to harden the contracts for future deployments or contract updates. We recommend against deploying the code as-is.
Update: During reaudit we observed the test suite improved drastically up to 117 working test cases. All issues were either Fixed or Acknowledged as observed on reaudit.

ID	Description	Severity	Status
QSP-1	Floating Points and Numerical Precision	^ Medium	Fixed
QSP-2	Missing Input Validations	^ Medium	Fixed
QSP-3	Oracle Can Be Stale	^ Medium	Acknowledged
QSP-4	Fees in `queryAttributesMatch()` Do Not Depends on the Number of Nonces	⌄ Low	Acknowledged
QSP-5	KYC Status and On-chain Data Is Always Public	⌄ Low	Acknowledged
QSP-6	Execution of `queryAttributesMatch()` Off-chain Without Paying Fees	⌄ Low	Acknowledged
QSP-7	Privileged Roles and Ownership	⌄ Low	Acknowledged
QSP-8	Chainlink Oracle May Be Manipulated	○ Informational	Acknowledged
QSP-9	Unlocked Pragma	○ Informational	Fixed

Quantstamp Audit Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

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

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

Methodology

The Quantstamp auditing process follows a routine series of steps:

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

Toolset

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

Setup

Tool Setup:

- Slither v0.8.0
- Mythril v0.22.16

Steps taken to run the tools:

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

Findings

QSP-1 Floating Points and Numerical Precision

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `Ky0xMain.sol`

Description: Because Solidity is incapable of storing floating-point numbers, division and numerical precision often require special metadata and operation order to handle correctly. A loss of precision is applicable in `KyOxMain.calculateAmountPayment` , `transactionCostUSD` is first divided by `10 ** (18 - _getOracleDecimals(_tokenPayment))` to get `costByOracleDecimal`. this operation can be done later to minimize precision loss.

Recommendation: Do all the multiplication before the divisions. Instead of computing `costByOracleDecimal`, `transactionCostUSD` can directly be used in `amount` calculation and leaving the division by `10 ** (18 - _getOracleDecimals(_tokenPayment))` as last operation.

QSP-2 Missing Input Validations

Severity: *Medium Risk*

Status: Fixed

Description: Some input validations are missing in the following cases:

- `KyOxGovernance.allowTokenPayment` should validate the aggregator address.
- `KyOxGovernance.setTreasury` should validate the treasury address.
- `KyOxMain.initialize` should validate all the inputs.
- `Ky0xMain._saveWalletInfo` should allow only enabled data types.
- `Ky0xMain._saveWalletInfo`, `Ky0xMain.getNonces` should also check the input data types.

Recommendation: Include suggested validations.

QSP-3 Oracle Can Be Stale

Severity: *Medium Risk*

Status: Acknowledged

Description: `KyOxMain._getPrice` calls `AggregatorV3Interface.latestRoundData` without checking `startedAt` and `updatedAt` values, meaning that the latest round returned can be obsolete since the Chainlink aggregators rely on external nodes to be updated once a request is submitted by one of the sponsors.

Recommendation: The `updatedAt` and `answeredInRound` values returned by Chainlink aggregator must be validated and must be recent. In case the price is stale, we recommend implementing a fallback oracle (in-house or a decentralized oracle like Uniswap) or revert based on your design decisions.

Update: Acknowledge by the project. Audit response notes were *"We acknowledge the possibility of a stale oracle. Implementing a backup oracle will add complexity, while reverting in the presence of a stale oracle will render our service unavailable and impact any smart contracts integrating with Ky0x.Ky0x would rather provide a 100% uptime, and will monitor and detect any stale oracle and proceed with a contract upgrade if necessary or price adjustment. Documentation about this property has been updated in our public gitbook. See the Transaction Cost section."*.

QSP-4 Fees in `queryAttributesMatch()` Do Not Depends on the Number of Nonces

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `Ky0xMain.sol`

Description: It might not be fair that users sending a lot of nonces pay the same as user sending only few or one nonce.

Recommendation: Scale the fee paid by the number of nonces or document this behavior.

Update: Acknowledge by the project. Audit response notes were *"This is currently a business decision to charge per query instead of the number of data types being purchased. The transaction fee is associated to provide an on-chain audit trail for smart contracts calling the ky0x contract. Documentation about this property has been updated in our public gitbook. See the Transaction Cost section."*.

QSP-5 KYC Status and On-chain Data Is Always Public

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `Ky0xMain.sol`

Description: According to documentation the KYC status of accounts is only revealed to the public only after another contract (for example Uniswap) requests the KYC to the KyOx contract. See documentation `Ky0x_Documentation.pdf (shasum 3f41e17fd5ee84fecd880daa81b1a7bd4409142a)` declaring `Uniswap smart contract to retrieve her KYC status [..] Due to the nature of blockchain and its transparency attribute, once information has been retrieved once, it becomes public information moving forward`. But due to Ethereum Mainnet nature, all the information is public and accessible via API or explorers, after is recorded in KyOx and before any other contract retrieves it.

Recommendation: Clarify the documentation regarding public information, or use another method for sharing information like commit&reveal schemes with Zero-Knowledge cryptography. Another alternative is using Ethereum implementations that support private transactions such as Quorum, but this might be incompatible with existing Ethereum services.

Update: Acknowledge by the project and changed to Low priority according to business logic. Audit response notes were *"Information in the ky0x smart contract is stored in a manner where the data is obfuscated without first obtaining the signature of a nonce by the customer wallet and executing the transaction on-chain. As such, information such as the wallet address or the data (ex: KYC status) about an entity are hidden. Once the data is queried on-chain once, the information becomes public (property of the Ethereum blockchain). Information posted on-chain in the Ky0xMain smart contract is never sensitive data. Documentation about this property has been updated in our public gitbook. See the Data Ownership section."*.

QSP-6 Execution of `queryAttributesMatch()` Off-chain Without Paying Fees

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `Ky0xMain.sol`

Description: Due to the public model of Ethereum data, all contract state is available to full-nodes without any permission or private key. One could execute the function off-chain for a range of data inputs and then submit to a cloned contract that doesn't require any fees. Fees won't be collected.

Recommendation: Provide a way to collect fees on this scenario for **free off-chain queries** or document this behavior.

Update: Acknowledge by the project and changed to Low priority according to business logic. Audit response notes were *"There is no benefit to smart contracts applications who need to prove that their user-base is restricted to a KYC/AML compliance tool, to read this information off-chain without executing the verification on-chain. Executing the verification on-chain, as part of your smart contract atomic transaction, acts essentially as an verified audit trail."*.

QSP-7 Privileged Roles and Ownership

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `Ky0xGovernance.sol`

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract.

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: Added documentation to explain this behavior here.

QSP-8 Chainlink Oracle May Be Manipulated

Severity: *Informational*

Status: Acknowledged

File(s) affected: `Ky0xMain.sol`

Description: External dependencies are critical because on many occasions they can't be updated. Prices from external oracles can be a victim of traditional hacker attacks and on-chain prices can be subject to Flash-loan or evil whale attacks that make the prices just very quickly or instantly.

Recommendation: Consider adding other oracles in case Chainlink price gets manipulated or is unavailable.

Update: Acknowledge by the project. Audit response notes were *"Ky0x will be monitoring for stale price feed updates and long-term inactivity of the ChainLink oracle service. As required, Ky0x can retrieve the price feed from an alternative oracle service (with a delay of 2 days imposed by our TimelockController)Documentation about this property has been updated in our public gitbook. See the Transaction Cost section."*

QSP-9 Unlocked Pragma

Severity: *Informational*

Status: Fixed

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.8.0`. The caret (`^`) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked".

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

Automated Analyses

Slither

Slither has detected many results out of which the majority have been filtered out as false positives and the rest have been integrated into the findings from this report.

Mythril

Mythril has detected many results out of which the majority have been filtered out as false positives and the rest have been integrated into the findings from this report.

Test Results

Test Suite Results

117 test cases of 117 passing.

```
$ npx hardhat test

  Ky0xGovernance
    setDataTypeStatus
       ✓ success
       ✓ fail (not admin)
       ✓ fail (already active)
       ✓ fail (already inactive)
```

```
    setTreasury
       ✓ success
       ✓ fail (not admin)
       ✓ fail (address zero)
       ✓ fail (already set)
    setTransactionCostUSD
       ✓ success
       ✓ success (2 decimals)
       ✓ success (5 decimals)
       ✓ success ($0)
       ✓ fail (not admin)
       ✓ fail (cost too high)
       ✓ fail (negative cost)
       ✓ fail (set with same value)
    pause
       ✓ success
       ✓ fail (not admin)
       ✓ fail (already paused)
    unpause
       ✓ success
       ✓ fail (not admin)
       ✓ fail (already unpaused)
    grantRole
       ✓ success (ATTESTOR_ROLE)
       ✓ success (DEFAULT_ADMIN_ROLE)
       ✓ fail (not admin)
    revokeRole
       ✓ success (ATTESTOR_ROLE)
       ✓ success (DEFAULT_ADMIN_ROLE)
       ✓ fail (not admin)
    allowTokenPayment
       ✓ success
       ✓ success (updating Oracle Price)
       ✓ fail (already authorized)
       ✓ fail (already disabled)
       ✓ fail (not a real oracle)
       ✓ fail (not admin)
       ✓ fail (disable ERC20)
       ✓ fail (not a real ERC20)
    upgrade Ky0xMain implementation
       ✓ success
       ✓ fail (not admin)

  Ky0xMain
    signature
       ✓ verify signature
    initialize
       ✓ success (parameters correctly initialized)
       ✓ fail (already initialized)
    postAttributes
       ✓ success - (Only KYC)
       ✓ success (KYC & AML - same user)
       ✓ success (KYC & AML - batch diff user)
       ✓ fail (paused)
       ✓ fail (not attestor)
       ✓ fail (nonce bytes(0))
       ✓ fail (walletSigAndAddr bytes(0))
       ✓ fail (attestation bytes(0))
       ✓ fail (ky0xID bytes(0))
       ✓ fail (length 0)
       ✓ fail (length >=10)
       ✓ fail (bad length attestations)
       ✓ fail (bad length nonces)
       ✓ fail (bad hashWalletSigAndAddrs length)
       ✓ fail (bad kIDs length)
       ✓ fail (bad dataTypes length)
    getNonces
       ✓ success (KYC only)
       ✓ success (KYC & AML)
       ✓ success (KYC  but no AML)
       ✓ fail (wrong owner)
       ✓ fail (wrong hashWalletSig)
    getBlockNumbers
       ✓ success (KYC only)
       ✓ success (KYC & AML)
       ✓ success (KYC but no AML)
       ✓ fail (wrong owner)
       ✓ fail (wrong hashWalletSig)
    calculateAmountPayment
       ✓ calculate USDC amount
       ✓ calculate WETH amount
       ✓ calculate DAI amount
       ✓ success (6 decimals)
       ✓ success (18 decimals)
       ✓ success (1 decimals)
       ✓ success (change transactionCost)
       ✓ fail (price zero)
       ✓ fail (price negative)
       ✓ fail (token not suppported)
    queryAttributesMatch
nonce 0x58ab927ac8e40b45301364cd8b21c9b295784ff66060cc606f8ac561a354859c
       ✓ success (KYC with USDC)
       ✓ success (KYC/AML/ with WETH)
       ✓ success (KYC but no AML)
       ✓ success (wrong dataType)
       ✓ success (wrong rawValues)
       ✓ success (wrong userAddr)
       ✓ success (wrong hashWalletSig)
       ✓ success (wrong nonceSig)
       ✓ fail (bad nonceSigsKD length)
       ✓ fail (bad dataTypes length)
       ✓ fail (bad rawValues length)
       ✓ fail (not enough allowance)
       ✓ fail (not enough balance)
       ✓ fail (token not allowed)
       ✓ fail (out of balance)

  Scenarios
    Query Via MSB (MATCH)
       ✓ POST KYC (MATCH) -> QUERY KYC (PAYMENT USDC)
       ✓ POST KYC (MATCH) -> QUERY KYC (PAYMENT DAI)
       ✓ POST KYC -> QUERY KYC (PAYMENT WETH)
       ✓ POST KYC/AML (MATCH) -> Query KYC/AML
    Query Via MSB (NO_MATCH)
       ✓ Post KYC (FAIL) -> Query KYC (FAIL)
       ✓ Post KYC(PASS) & AML(LOW_RISK) -> Query KYC & AML
       ✓ KYC NO_MATCH / AML MATCH / NO_MATCH
    MSB reverts
       ✓ POST KYC (FAIL) -> Query KYC
       ✓ POST KYC (PASS) / AML (LOW_RISK) -> Query KYC / AML
       ✓ Wrong Sender (ky0xID not found)
       ✓ Post AML (LOW_RISK) -> Query KYC
    Missing DataTypes
       ✓ Post AML (LOW_RISK) -> Query KYC
       ✓ Post KYC (PASS) -> Query KYC & AML
       ✓ Post AML (LOW_RISK) -> Query KYC & AML
    Attested With Different Timeframe
       ✓ POST KYC -> POST WALLT_AML -> POST AML -> QUERY ALL
    Many Users Batching
       ✓ Post UserA & UserB & UserC-> Query User A -> Query User B -> QUery C
    Different Wallet, Same ky0xID
       ✓ Post WalletA -> Post WalletB -> Query WalletA -> Query WalletB
    Continuous Monitoring - Data Update
       ✓ [Re-using Nonce] Post KYC (PASS) -> QUERY KYC -> UPDATE KYC (PASS) -> QUERY KYC
       ✓ [New Nonce] Post KYC (PASS) -> QUERY KYC -> UPDATE KYC (PASS) -> QUERY KYC
       ✓ Post KYC (PASS) & AML (LOW_RISK) -> QUERY KYC/AML -> UPDATE AML (LOW_RISK) -> QUERY KYC/AML
    Smart Contract Update
       ✓ POST KYC/AML -> Query KYC/AML -> Update Smart Contract -> Query KYC/AML
       ✓ POST KYC/AML Update Smart Contract -> Query KYC/AML -> Post KYC/AML -> Query KYC/AML
    Change of Price Feed
       ✓ Post -> Query -> Change Price -> Query
    Query Via callKy0x
       ✓ POST KYC (MATCH) -> QUERY KYC (PAYMENT USDC)

  Sign
     ✓ Deterministic ECDSA signature

·---------------------------------------------|---------------------------|-------------|-----------------------------·
|           Solc version: 0.8.0               ·  Optimizer enabled: false  ·  Runs: 200  ·  Block limit: 12450000 gas  |
··············|··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Methods                                                                                                              |
··············|··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Contract    ·  Method      ·  Min         ·  Max  ·  Avg  ·  # calls  ·  eur (avg)  |
··············|··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  KERC20      ·  approve     ·       46845  ·  46881  ·  46877  ·       11  ·          -  |
··············|··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  KERC20      ·  transfer    ·       52475  ·  52535  ·  52518  ·      350  ·          -  |
··············|··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Ky0xGovernance  ·  allowTokenPayment  ·       41080  ·  93092  ·  92405  ·      370  ·          -  |
··············|··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Ky0xGovernance  ·  grantRole  ·       56558  ·  56930  ·  56707  ·        5  ·          -  |
··············|··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Ky0xGovernance  ·  pause      ·           -  ·      -  ·  51684  ·        5  ·          -  |
··············|··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Ky0xGovernance  ·  revokeRole  ·       22460  ·  24832  ·  23646  ·        2  ·          -  |
··············|··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
```

```
|  Ky0xGovernance  ·  setDataTypeStatus      ·          -  ·          -  ·      53349  ·            3  ·          -  |
·----------------|----------------------|------------|------------|------------|-------------|------------·
|  Ky0xGovernance  ·  setTransactionCostUSD  ·      20769  ·      35853  ·      33103  ·           11  ·          -  |
·----------------|----------------------|------------|------------|------------|-------------|------------·
|  Ky0xGovernance  ·  setTreasury            ·      36314  ·      36326  ·      36320  ·            4  ·          -  |
·----------------|----------------------|------------|------------|------------|-------------|------------·
|  Ky0xGovernance  ·  unpause                ·          -  ·          -  ·      19563  ·            2  ·          -  |
·----------------|----------------------|------------|------------|------------|-------------|------------·
|  Ky0xMain        ·  postAttributes         ·      53595  ·     311220  ·     151210  ·           51  ·          -  |
·----------------|----------------------|------------|------------|------------|-------------|------------·
|  Ky0xMain        ·  queryAttributesMatch   ·      78871  ·     123177  ·     110508  ·           10  ·          -  |
·----------------|----------------------|------------|------------|------------|-------------|------------·
|  Ky0xMain        ·  upgradeTo              ·          -  ·          -  ·      43569  ·            4  ·          -  |
·----------------|----------------------|------------|------------|------------|-------------|------------·
|  MSB             ·  callKy0x               ·          -  ·          -  ·     149730  ·            2  ·          -  |
·----------------|----------------------|------------|------------|------------|-------------|------------·
|  MSB             ·  depositWithEvents      ·      99983  ·     166286  ·     145547  ·           62  ·          -  |
·----------------|----------------------|------------|------------|------------|-------------|------------·
|  Deployments                               ·                                       ·  % of limit   ·             |
·--------------------------------------------|------------|------------|------------|-------------|------------·
|  ChainLinkPriceOracle                      ·     415586  ·     435870  ·     435441  ·       3.5 %  ·          -  |
·--------------------------------------------|------------|------------|------------|-------------|------------·
|  KERC20                                    ·    1296619  ·    1296775  ·    1296679  ·      10.4 %  ·          -  |
·--------------------------------------------|------------|------------|------------|-------------|------------·
|  Ky0xMain                                  ·          -  ·          -  ·    5288387  ·      42.5 %  ·          -  |
·--------------------------------------------|------------|------------|------------|-------------|------------·
|  Ky0xMainV2                                ·          -  ·          -  ·    4863097  ·      39.1 %  ·          -  |
·--------------------------------------------|------------|------------|------------|-------------|------------·
|  MSB                                       ·    1435427  ·    1435439  ·    1435438  ·      11.5 %  ·          -  |
·--------------------------------------------|------------|------------|------------|-------------|------------·

  117 passing (50s)
```

Code Coverage

The tests feature good coverage, excepting `Ky0xError` and `Ky0xGovernance`. We recommend improving the coverage to be at least 80% for each module. Hardhat Coverage module support was included following this tutorial.

```
$ npx hardhat coverage
```

Update: the coverage module was not working for the reaudit, but the test suit overall has improved a lot from 15 to 117 test cases.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	75	42.86	72.22	75.27	
Ky0xError.sol	*0*	100	*0*	*0*	*27, 29*
Ky0xGovernance.sol	*25*	*10*	*25*	*25*	*... 57, 58, 59, 60*
Ky0xMain.sol	87.84	53.13	92.31	88	... 149,150,152
Ky0xStore.sol	100	100	100	100	
contracts/interfaces/	100	100	100	100	
IKy0xMain.sol	100	100	100	100	
contracts/test/	100	87.5	100	100	
ERC20.sol	100	100	100	100	
Ky0xMainV2.sol	100	100	100	100	
MSB.sol	100	87.5	100	100	
PriceOracle.sol	100	100	100	100	
All files	**80.51**	**50**	**80**	**80.83**	

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

```
5e35864a994150f078966d311607b0f635041cbf51d3d4bfe048f42efa4ee013 ./contracts/Ky0xError.sol
94fb45d5384eb074bb1845e852f8bd028c0f8fa194b1ebf2a8bb1d46915d0e50 ./contracts/Ky0xGovernance.sol
527706bbfa79a73c90ef470280ab69653ffb519f76c8b5fe1f717b18d70ffc7e ./contracts/Ky0xMain.sol
b01e81a0f7a5da340f9ec1f109a84367ba5c28b3ef382b1f96b645e74d0db218 ./contracts/Ky0xStore.sol
155f394b3ab81de025ead309b057400c5adc4fcf4d20ef94d1261ab5889516cf ./contracts/interfaces/IKy0xMain.sol
ba3619cd1773a36c5cefae95a3809eca93bc48d988a3959b9d84bf11cbce4338 ./contracts/test/ERC20.sol
f7e7d94e16049c497b41211b644edb15c03069cf267293052774e73d523ddb41 ./contracts/test/PriceOracle.sol
b7245c20959cd3f7da39b987c066138ca430104d28058775512fc69a1447e52f ./contracts/test/MSB.sol
134f400caa8f6e1b42a6926f09c668a8113f35c13ac7deb22a96e75fe93f6f6c ./contracts/test/Ky0xMainV2.sol
```

Tests

```
2bfa49d9c82cee60ea0770f830b2c695e6096a36f62ea1224b60370fa94265b0 ./test/TestSign.js
4a4cf76953d0b68fec7203d6dda9923f919e98a80bcd3b781f28eb0041eadef2 ./test/TestKy0x.js
016e30827142c4a0f0eaa7b37dcaa5bc244574bd744cb68b4759d5c2922b479e ./test/utils/constant.js
61047835cfe6bfad5d345f517974561e2ed10d82964b3da5dac4eb433b884b9b ./test/utils/helpers.js
```

Changelog

- 2021-07-06 - Initial report

About Quantstamp

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

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

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

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