Quantstamp Security Assessment Certificate

October 30th 2020 — Quantstamp Verified

KIRA Token

This security assessment was prepared by Quantstamp, the leader in blockchain security

Executive Summary

Type	ERC20 Token
Auditors	Martin Derka, Senior Research Engineer
Timeline	2020-10-30 through 2020-10-30
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	High
Test Quality	High

Source Code

Repository	Commit
liquidity-program	a571f1a

Goals

- The ERC20 security
- Whitelist and blacklist correctness

Total Issues	**3** (1 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	0 (0 Resolved)
Low Risk Issues	0 (0 Resolved)
Informational Risk Issues	**3** (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

The codebase implements an ERC20 with granular permissions on transfer: a blacklist and whitelist where the blacklist supersedes the whitelist, and the whitelist stipulates whether an account can send and/or receive tokens. The implementation is very clean, rather minimalistic, and relies on well-known dependencies. Quantstamp did not discover any defects beyond a redundant `Transfer` emitted during deployment (also disclosed by the client), and possible transaction ordering dependence of whitelisting and blacklisting transactions, and transfers. The audit is conducted post-deployment.

ID	Description	Severity	Status
QSP-1	Privileged Roles and Ownership	Informational	Unresolved
QSP-2	Allowance Double-Spend Exploit	Informational	Mitigated
QSP-3	Transaction Ordering Dependence	Informational	Unresolved

Quantstamp Audit Breakdown

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

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

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

Methodology

The Quantstamp auditing process follows a routine series of steps:

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

Toolset

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

Setup

Tool Setup:

- Slither v0.6.6
- Mythril v0.2.7

Steps taken to run the tools:

1. Installed the Slither tool: `pip install slither-analyzer`
2. Run Slither from the project directory: `slither .s`
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 Privileged Roles and Ownership

Severity: Informational

Status: Unresolved

File(s) affected: `KiraToken.sol`

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract. However, 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.
In KIRA's case, the owner has the privilege of whitelisting and blacklisting accounts, effectively enabling or disabling them from sending and receiving tokens. The operator also has the option of freezing token transfers for all accounts that are not explicitly whitelisted.

Recommendation: Quantstamp recommends informing the users of the owner's privileges.

QSP-2 Allowance Double-Spend Exploit

Severity: Informational

Status: Mitigated

File(s) affected: `KiraToken.sol`

Description: As it presently is constructed, the contract is vulnerable to the allowance double-spend exploit, as with other ERC20 tokens. An example of an exploit goes as follows:

1. Alice allows Bob to transfer `N` amount of Alice's tokens (`N>0`) by calling the `approve()` method on `Token` smart contract (passing Bob's address and `N` as method arguments)
2. After some time, Alice decides to change from `N` to `M` (`M>0`) the number of Alice's tokens Bob is allowed to transfer, so she calls the `approve()` method again, this time passing Bob's address and `M` as method arguments
3. Bob notices Alice's second transaction before it was mined and quickly sends another transaction that calls the `transferFrom()` method to transfer `N` Alice's tokens somewhere
4. If Bob's transaction will be executed before Alice's transaction, then Bob will successfully transfer `N` Alice's tokens and will gain an ability to transfer another `M` tokens
5. Before Alice notices any irregularities, Bob calls `transferFrom()` method again, this time to transfer `M` Alice's tokens. The exploit (as described above) is mitigated through use of functions that increase/decrease the allowance relative to its current value, such as `increaseAllowance` and `decreaseAllowance`.

Pending community agreement on an ERC standard that would protect against this exploit, we recommend that developers of applications dependent on `approve()` / `transferFrom()` should keep in mind that they have to set allowance to 0 first and verify if it was used before setting the new value. Teams who decide to wait for such a standard should make these recommendations to app developers who work with their token contract.

Recommendation: The code contains mitigation in the form of methods for increasing and decreasing allowance. Quantstamp recommends instructing the users to use these methods.

QSP-3 Transaction Ordering Dependence

Severity: Informational

Status: Unresolved

File(s) affected: `KiraToken.sol`

Description: The owner has the privilege of blacklisting and whitelisting accounts. As several transactions can be submitted to the network and wait for inclusion in blocks, they can be raced by the accounts being blacklisted (or whitelisted with decreased permissions). The ordering of transaction is up to miners' discretion, thus the accounts can be capable of migrating their tokens into other accounts prior to getting blacklisted.

Recommendation: This appears to be an inherent issue of on-chain whitelist and blacklist. There does not seem to be an obvious way of mitigating the problem using the current design of the token, so the operator should simply be aware of this property.

Automated Analyses

Slither

Slither reported no relevant issues.

Mythril

Slither reported no relevant issues.

Adherence to Specification

The code adheres to provided scpecification.

Code Documentation

The specification for the token was provided via email. The code adheres to it. The in-code documentation provides the right level of detail and appears very clean. The supplementary documentation inside the repository also appears very accurate.

Adherence to Best Practices

The constructor of the tokens contains a `Transfer()` event that is emitted during the token deployment in addition to the event emitted by the `_mint()` method. This event could be removed, however, its existence has no impact on the security of the token.

Test Results

Test Suite Results

The tests have the right focus, contain the expected assertions, and generally meet all the expectations.

```
Contract: KiraToken Test
  totalSupply
     ✓ all tokens should be in the deployer account (68ms)
  freeze
     ✓ should be freezed at first and the transfer should be rejected (135ms)
     ✓ should NOT be able to freeze when it was already freezed (38ms)
     ✓ should reject freeze call from non owner
     ✓ should make the token as freeze when it was unfreezed (270ms)
  unfreeze
     ✓ should NOT be able to unfreeze when it was already unfreezed (98ms)
     ✓ should reject unfreeze call from non owner
     ✓ should make the token as unfreeze when it was freezed (259ms)
     ✓ should be able to transfer freely once unfreezed (263ms)
  whitelist
     ✓ onwer should have full whitelist (43ms)
     ✓ should NOT be able to configure whitelist of 0 address
     ✓ should NOT be able to configure whitelist without owner permission (77ms)
     ✓ should whitelist any options of multiple addresses (118ms)
  blacklist
     ✓ should NOT be able to add 0x0 to the blacklist
     ✓ should NOT be able to remove 0x0 from the blacklist
     ✓ should NOT be able to add/remove blacklist without owner permission (140ms)
     ✓ should add to blacklist (103ms)
     ✓ should remove from blacklist (83ms)
  transfer when unfreezed
     ✓ should NOT be able to transfer (from: blacklisted) (186ms)
     ✓ should NOT be able to transfer (to: blacklisted) (126ms)
     ✓ should transfer (from: blacklisted [no], to: blacklisted [no]) (138ms)
  transfer when freezed
     ✓ should NOT be able to transfer (from: blacklisted) even if its allow_transfer is true (271ms)
     ✓ should NOT be able to transfer (to: blacklisted) even if its allow_deposit is true (185ms)
     ✓ should NOT be able to transfer (from: allow_transfer [no], to: allow_deposit [yes]) (135ms)
     ✓ should NOT be able to transfer (from: allow_transfer [yes], to: allow_deposit [no]) (188ms)
     ✓ should transfer (from: allow_transfer [yes], to: allow_deposit [yes]) (149ms)
     ✓ should transfer (from: allow_unconditional_transfer [yes], to: allow_deposit [no]) (165ms)
     ✓ should transfer (from: allow_transfer [no], to: allow_unconditional_deposit [yes]) (325ms)
  multi transfer
     ✓ should be able to transfer to multiple accounts (222ms)


29 passing (7s)
```

Code Coverage

While the coverage tool reports low coverage of certain whitelist branches, upon manual inspection of the test suite, Quantstamp considers the coverage very good.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	70.83	100	100	
KiraToken.sol	100	70.83	100	100	
All files	**100**	**70.83**	**100**	**100**	

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

```
b5184c06c1537b07e334afb7544d0c1dea83e0d838ce346895262883e47e7e8c ./contracts/KiraToken.sol

b129965d95472eca4378e5533e6e783e66fb84df6c8e07a87d8fb06f5b481afb ./contracts/Migrations.sol
```

Tests

```
11117cdb20c1d007f57b456f2afc506fd2e5225f97f410f178afe7d58cae19ef ./test/KiraToken.test.js
```

Changelog

- 2020-10-30 - 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.