Quantstamp Security Assessment Certificate

September 2nd 2020 — Quantstamp Verified

Serum

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

Executive Summary

Type	ERC20 Token
Auditors	Kacper Bąk, Senior Research Engineer Leonardo Passos, Senior Research Engineer Sebastian Banescu, Senior Research Engineer
Timeline	2020-08-06 through 2020-08-07
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	Undetermined
Test Quality	Undetermined

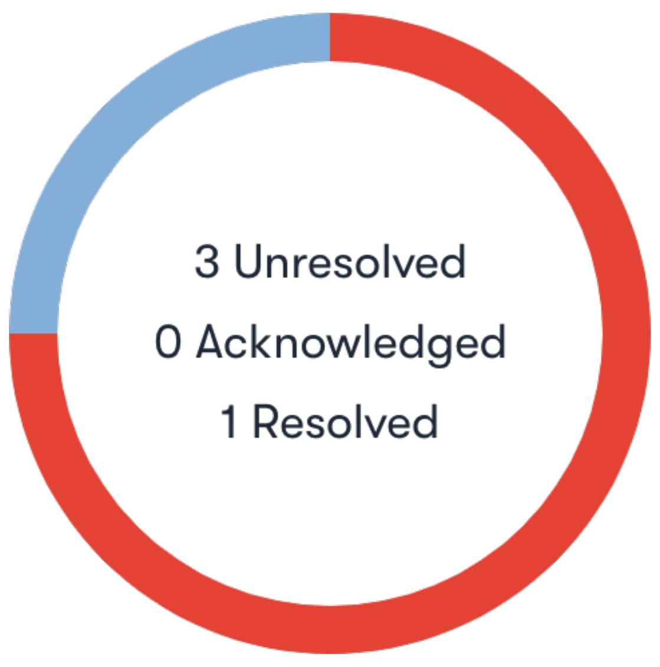
Source Code

Repository	Commit
Token 0x476c5e2	None
Token 0x1320c8c	None

Goals

- Can users' funds be locked up?
- Can an attacker steal users' funds?

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

The contract code reuses OpenZeppelin libraries for developing ERC20 tokens. The code that comes from OpenZeppelin is well-documented and tested. The remaining code, however, has no documentations and no tests. We recommend addressing these issues as well as other issues indicated in the report, notably the one that allows a contract to have no burner.

ID	Description	Severity	Status
QSP-1	Privileged Roles and Ownership	⌄ Low	Unresolved
QSP-2	Contract can be permanently left without a burner/minter/pauser	⌄ Low	Unresolved
QSP-3	Unlocked Pragma	○ Informational	Unresolved
QSP-4	Allowance Double-Spend Exploit	○ Informational	Mitigated

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 v0.6.11

Steps taken to run the tools:

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

Findings

QSP-1 Privileged Roles and Ownership

Severity: Low Risk

Status: Unresolved

File(s) affected: `0x476c5e26a75bd202a9683ffd34359c0cc15be0ff`, `0x1320c8c64b9f2eAa851F70702e6C9FC1EE4E8Ce4`

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract. The contracts also employ pauser, minter, and burner roles. Although these roles are useful, they are trusted with a reasonable use of their privileges.

Recommendation: This centralization of power needs to be made clear to the users. Properly document what operations are centralized on a single actor, and under what conditions will those operations be triggered.

QSP-2 Contract can be permanently left without a burner/minter/pauser

Severity: Low Risk

Status: Unresolved

File(s) affected: `0x476c5e26a75bd202a9683ffd34359c0cc15be0ff`, `0x1320c8c64b9f2eAa851F70702e6C9FC1EE4E8Ce4`

Description: All contracts inheriting from `BurnerRole` (directly or indirectly) may be permanently left without a burner; consequently, any operation dependent on the `onlyBurner` modifier will fail. Similar notes apply to miner and pauser roles.

Exploit Scenario:

1. Initially, the deployer is granted a burner role, as the constructor invokes `_addBurner`.
2. The deployer renounces their burner role prior to granting it to any other address.
3. From this point on, it will be impossible to add the burner role to any other address, as `addBurner` can only be called by an address that has a burner role - at this point, no such address exists.

Recommendation: Disallow removing a burner if there is only one left. Furthermore, make `BurnerRole` an Ownable contract and change `addBurner` so that it can also be called by the owner. Similar notes apply to minter and pauser roles.

QSP-3 Unlocked Pragma

Severity: Informational

Status: Unresolved

File(s) affected: `0x476c5e26a75bd202a9683ffd34359c0cc15be0ff`, `0x1320c8c64b9f2eAa851F70702e6C9FC1EE4E8Ce4`

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.4.*`. 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: In non-library code (i.e., non-OpenZeppelin code), 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 which is compatible with the versions of all imported contracts. In the current version of the contracts, it would be any version higher or equal to 0.5.5.

QSP-4 Allowance Double-Spend Exploit

Severity: Informational

Status: Mitigated

File(s) affected: `0x476c5e26a75bd202a9683ffd34359c0cc15be0ff`, `0x1320c8c64b9f2eAa851F70702e6C9FC1EE4E8Ce4`

Description: As it presently is constructed, the contract is vulnerable to the allowance double-spend exploit, as with other ERC20 tokens.

Exploit Scenario:

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.

Recommendation: 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. Furthermore, we recommend that developers of applications use the functions `increaseAllowance()` and `decreaseAllowance()` to mitigate this issue. Both functions are already present in the code. Teams who decide to wait for such a standard should make these recommendations to app developers who work with their token contract.

Automated Analyses

Slither

Slither reported that the function `CanReclaimEther.reclaimEther()` sends eth to arbitrary user. It is a false positive, however, since only owner can call this function.

Adherence to Specification

The code comes with no specification.

Code Documentation

At the very least, document all external or public functions using the natspec format.

Adherence to Best Practices

Add an error message to `require(isBurner(msg.sender));`

Test Results

Test Suite Results

The code comes with no test suite.

Changelog

- 2020-08-07 - 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.

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.

Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp, Inc. (Quantstamp). Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on the website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.