Quantstamp Security Assessment Certificate

September 22nd 2020 — Quantstamp Verified

Swarm Token

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

Executive Summary

Type	Token Contract
Auditors	Shunsuke Tokoshima, Software Engineer Ed Zulkoski, Senior Security Engineer
Timeline	2020-08-31 through 2020-09-22
EVM	Muir Glacier
Languages	Solidity, Javascript
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	High
Test Quality	Undetermined
Source Code	see table below

Repository	Commit
swarm-token	b341c20

Goals

- Is there any centralization of power?
- Does the code conform to ERC20?
- Are general issues such as re-entrancy or underflow possible?

Total Issues	**2** (1 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	0 (0 Resolved)
Low Risk Issues	0 (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 code is generally well-written and well-documented. No critical security issues were detected during this audit.

ID	Description	Severity	Status
QSP-1	Unlocked Pragma	Informational	Fixed
QSP-2	Clone-and-Own	Informational	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 v0.6.12

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 Unlocked Pragma

Severity: Informational

Status: Fixed

File(s) affected: `Token.sol`

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.5.0`. 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." 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.

Recommendation: Locking pragma versions, e.g., pragma solidity 0.5.0.
Update: This issue has been fixed in `7bec1cf1`

QSP-2 Clone-and-Own

Severity: *Informational*

Status: Acknowledged

File(s) affected: `Token.sol`

Description: The clone-and-own approach involves copying and adjusting open source code at one's own discretion. Considering the code before L714 is identical to the codebase in an npm package @openzeppelin/contracts v2.5.0 , the situation is not problematic for the time being. However, in case that development continues based on this file, it will possibly be accompanied by some risks as the code may not follow the best practices, may contain a security vulnerability, or may include intentionally or unintentionally modified upstream libraries.

Automated Analyses

Slither

No security threats reported; only issues related to naming, and making `public` functions not called by the contract as `external` (saves gas).

Code Documentation

The code is well-documented.

Test Results

Test Suite Results

```
✓ Token Tests
   Parent input validation checks
      ✓ (detailed) Correct deployment (325ms, 1949574 gas)
    ✓  ✓ (cap) Can't deploy a 0 cap (65ms)

  Total Gas Used: 1949574


 2 passing (432ms)
```

Code Coverage

The code features low code coverage due to the cloned code.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	61.36	40	66.67	61.11	
Mock_dont_audit_curve.sol	100	75	100	100	
Token.sol	59.52	36.11	64.44	59.3	… 604,605,648
All files	**61.36**	**40**	**66.67**	**61.11**	

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

`856d342ed6d05ce7592129f46ec1e00fc446c4ea92d6329f7fe2f3c57313ea4a ./contracts/Token.sol`

Tests

`23ad2f674ad1d3c7a1e3c3c605a17e91808dbe52c730a79a1b288a74861cb6bd ./test/Token.test.js`

`e92c1c8420d30240d4895eaa9de6a7c1eb607a91d5e2a83dc38313c078cf2d41 ./test/Settings.js`

Changelog

- 2020-09-02 - Initial report
- 2020-09-22 - Report revised based on commit `7bec1cf1`

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.