Quantstamp Security Assessment Certificate

January 18th 2021 — Quantstamp Verified

Portto Cross-Chain Token Lock

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

Executive Summary

Type	Cross-chain Token Lock
Auditors	Jan Gorzny, Blockchain Researcher Ed Zulkoski, Senior Security Engineer Leonardo Passos, Senior Research Engineer
Timeline	2021-01-11 through 2021-01-18
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	Undetermined
Test Quality	Undetermined

Source Code

Repository	Commit
bloctoswap-contracts	aa2dd0f

Goals

• Review the code for common pitfalls and errors, including potential issues which may result in locked tokens.

Total Issues	**5** (3 Resolved)
High Risk Issues	**2** (2 Resolved)
Medium Risk Issues	0 (0 Resolved)
Low Risk Issues	**3** (1 Resolved)
Informational Risk Issues	0 (0 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

Quantstamp audited the the cross-chain token lock code provided, and despite the small code base, found a number of issues, which have been addressed.

ID	Description	Severity	Status
QSP-1	Possible Movement of Locked Funds	High	Fixed
QSP-2	Contract Can Be Left Without An Owner	High	Fixed
QSP-3	Privileged Roles and Ownership	Low	Acknowledged
QSP-4	Unlocked Pragma	Low	Fixed
QSP-5	Unlock may Transfer Tokens to an Unexpected Ethereum Address	Low	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 v0.6.6
- 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 .`
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 Possible Movement of Locked Funds

Severity: *High Risk*

Status: Fixed

File(s) affected: `TeleportCustody.sol`

Description: The functions `unlockByOwner` and `lock` do not have the `notFrozen` qualifier on it, which may allow funds to be moved even if the account is frozen.

Recommendation: Add the `notFrozen` qualifier to the function definitions.
Update: the qualifier has been added.

QSP-2 Contract Can Be Left Without An Owner

Severity: *High Risk*

Status: Fixed

File(s) affected: `TeleportAdmin.sol`, `TeleportCustody.sol`

Description: Since an owner can renounce its ownership, it is possible that all contracts inheriting from `Ownable` to be left without an owner. If that happens, one will lose the ability to freeze the contract in case something bad happens.

Recommendation: Override `renounceOwnership` function so that ownership cannot be renounced.
Update: the function has been overridden.

QSP-3 Privileged Roles and Ownership

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `TeleportCustody.sol`, `TeleportAdmin.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: The developers will make sure these roles are known to end-users; comments have been added to the code.

QSP-4 Unlocked Pragma

Severity: *Low Risk*

Status: Fixed

File(s) affected: `TeleportCustody.sol`, `TeleportAdmin.sol`

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: 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.
Update: The pragma has been locked at 0.6.12.

QSP-5 Unlock may Transfer Tokens to an Unexpected Ethereum Address

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `TeleportCustody.so`

Description: `unlock` does not guarantee that an unlock occurs to an specific Ethereum address. An admin, given an allowed amount X, can unlock X tokens from any account holding at least X tokens, which may not be the expected behavior.

Recommendation: Make sure that any admin is granted an allowed amount for an specific ethereum address.
Update: The developers have stated that "the user who locked the tokens and the user who will receive the unlocked tokens may very likely be different users. So we cannot know the unlock target beforehand and we cannot check if unlock target is really an address user has access to."

Adherence to Specification

No specification was provided.

Code Documentation

The code has minimal documentation.

Adherence to Best Practices

1. Functions are not ordered by visibility, which may make the code hard to maintain or read. **Update**: the functions have been sorted.

2. `TeleportAdmin.sol` line 190: the "-=" operation cannot underflow, as L183 requires that the allowed amount is always greater than the amount to subtract. Nonetheless, we advise adding a comment on L190 stating that this is a safe operation to bring confidence to those who read the code. **Update**: a comment has been added.

3. `TetherToken.sol` should be an interface. **Update**: it is now an interface.

Test Results

Test Suite Results

```
Contract: TeleportCustody (USDT) Tests
   TeleportAdmin
     updateAdmin()
        ✓ should set allowedAmount of an admin (86ms)
        ✓ should update allowedAmount of an admin (122ms)
        ✓ can only be called by owner (83ms)
        ✓ can have multiple admins (134ms)
     freeze() & unfreeze()
        ✓ should be able to freeze contract (103ms)
        ✓ should be able to unfreeze contract (115ms)
        ✓ can only be called by owner (95ms)
     renounceOwnership()
        ✓ should be rejected (45ms)
        ✓ can only be called by owner (50ms)
   TeleportCustody
     lock()
        ✓ can lock tokens from users (58ms)
        ✓ should block when teleport service is frozen (98ms)
     unlock()
        ✓ requires authorization from owner (56ms)
        ✓ requires sufficient authorization from owner (81ms)
        ✓ should unlock if admin has enough authorization (96ms)
        ✓ should unlock multiple times if admin has enough authorization (141ms)
        ✓ should block duplicated Flow hash (164ms)
        ✓ should block when target address is 0 (92ms)
        ✓ should block when admin has depleted authorization (159ms)
        ✓ should block when teleport service is frozen (142ms)
        ✓ should unlock when teleport service is unfrozen (250ms)
     unlockByOwner()
        ✓ should unlock (51ms)
        ✓ should unlock multiple times (93ms)
        ✓ should block duplicated Flow hash (80ms)
        ✓ should block duplicated Flow hash by admin (197ms)
        ✓ should block when teleport service is frozen (115ms)
        ✓ should unlock when teleport service is unfrozen (232ms)
        ✓ can only be called by the owner (93ms)


  27 passing (5s)
```

Code Coverage

Although tests were provided, we were unable to compute the test coverage.

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

```
696fead6ed2630d6744742172c2ddf90c0a98d26cda375dc0f8b97482acf70d8 ./contracts/teleport/ethereum/TeleportAdmin.sol

909e0dbb3017cdc6b87ee916c793c88cf9063cd61594424c7e3ae50ef2875354 ./contracts/teleport/ethereum/TeleportCustody.sol

8cf3d7bba7027c05a28943840d31555fa64aa353d669ba48f074bd920c4e5ebf ./contracts/teleport/ethereum/TeleportCustodyTest.sol

5149c3a61c2d31b063655eef20371c51cbb0dc47b73c4ab9e842879c77d76117 ./contracts/teleport/ethereum/TetherToken.sol

99b540a847fd20f514d4632d8295af96e7fd8df08a26c17f05a0d927df0e7877 ./contracts/teleport/ethereum/TetherTokenTest.sol
```

Tests

```
3b191a6b4d879a7904ea22e06b84bae3f3c2f55f124ec6248fb086179f5bf016 ./test/TeleportCustody.js
```

Changelog

- 2021-01-11 - Initial report [`3cd8c18` - gist.github.com]
- 2021-01-18 - Revised report [`aa2dd0f`]

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.