**Quantstamp** Contract Security Certificate


June 18th 2020 — Quantstamp Verified

# TokenSoft Token

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

## Executive Summary

| | |
|---|---|
| Type | Audit |
| Auditors | Kevin Feng, Software Engineer<br>Kacper Bąk, Senior Research Engineer<br>Ed Zulkoski, Senior Security Engineer |
| Timeline | 2020-06-02 through 2020-06-16 |
| EVM | Muir Glacier |
| Languages | Solidity, Javascript |
| Methods | Architecture Review, Unit Testing, Computer-Aided Verification, Manual Review |
| Specification | Github README.md specification |
| Documentation Quality | Medium |
| Test Quality | High |

Source Code

| Repository | Commit |
|---|---|
| tokensoft_token | 8cdbf3e |
| tokensoft_token | c09eeb3 |

Changelog

- 2020-06-03 - Initial report [commit: `8cdbf3e`]
- 2020-06-16 - Reaudit report [commit: `c09eeb3`]

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

The code is overall, well-written, easy to understand and is accompanied with thorough testing. The code behaviour also matches the specifications of the token. No critical security issues were detected during this audit. However, the project would benefit greatly from additional documentation, especially in addressing the centralization of power within the token's privileged roles. We recommend these issues be reviewed and resolved prior to the code being used in production.
**Update:** As of commit `c09eeb3`, all of the issues are acknowledged or fixed. Documentation is updated to address our highlighted risks and input validations are added in accordingly. However, some best practise issues/typos still remain.

| ID | Description | Severity | Status |
|---|---|---|---|
| QSP-1 | Privileged Roles and Ownership | Medium | Acknowledged |
| QSP-2 | Possible Transfer to 0x0 Contract Address | Low | Fixed |
| QSP-3 | Event Emitted for Removing a Non-whitelisted Address | Low | Fixed |
| QSP-4 | No Proxy Verification for a Contract Account Address | Low | Acknowledged |
| QSP-5 | Allowance Double-Spend Exploit | 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:

- Truffle
- SolidityCoverage
- Mythril
- Truffle-Flattener
- Securify
- Slither

Steps taken to run the tools:

1. Installed Truffle: `npm install -g truffle`
2. Installed the solidity-coverage tool (within the project's root directory): `npm install --save-dev solidity-coverage`
3. Ran the coverage tool from the project's root directory: `./node_modules/.bin/solidity-coverage`
4. Flattened the source code using `truffle-flattener` to accommodate the auditing tools.
5. Installed the Mythril tool from Pypi: `pip3 install mythril`
6. Ran the Mythril tool on each contract: `myth -x path/to/contract`
7. Ran the Securify tool: `java -Xmx6048m -jar securify-0.1.jar -fs contract.sol`
8. Installed the Slither tool: `pip install slither-analyzer`
9. Run Slither from the project directory `slither .`

# Assessment

## Findings

### QSP-1 Privileged Roles and Ownership

Severity: *Medium Risk*

**Status:** Acknowledged

**File(s) affected:** `Revocable.sol`, `OwnerRole.sol`

**Description:** Smart contracts will often have `Owner` variables to designate the person with special privileges to make modifications to the smart contract. Currently, the `Owner` role can add/remove other roles as well as other owners. Moreover, the `Revoker` role can transfer funds between users readily, which may be a dangerous privilege. If these privileged members lose their private keys, regular users may suffer from lost funds and unexpected behaviour.

**Recommendation:** Centralization of power needs to be clearly documented to users. Users should be made clear about any risks associated with these privileged roles.
**Update:** Addressed in README documentation.

### QSP-2 Possible Transfer to 0x0 Contract Address

Severity: *Low Risk*

**Status:** Fixed

**File(s) affected:** `Proxy.sol`, `TokenSoftToken.sol`, `Whitelistable.sol`

**Description:** It is rarely desirable for tokens to be sent to the `0x0` address (intentional token burning is a notable exception) nor to the contract itself. However, these mistakes are often made due to human errors. Hence, it's often a good idea to prevent these mistakes from happening within the smart contract itself. Currently, we've observed that most of the methods that have an address parameter do not check for `0x0` (e.g. `_removeFromWhitelist(...)` in the file `Whitelistable.sol` and `constructor(...)` in `Proxy.sol`).

**Recommendation:** Each of the methods described should add a `require` statement to check the input for the possibility of address `0x0`.
**Update:** Validation has been added to address this.

### QSP-3 Event Emitted for Removing a Non-whitelisted Address

Severity: *Low Risk*

**Status:** Fixed

**File(s) affected:** `Whitelistable.sol`

**Description:** Inside the function `_removeFromWhitelist()`, an event is emitted regardless of whether the address `addressToRemove` is present inside the whitelist. As a result, the event information emitted may be misleading.

**Recommendation:** Check for the existence of `addressToRemove` first before emitting the event.
**Update:** Validation has been added to address this.

### QSP-4 No Proxy Verification for a Contract Account Address

Severity: *Low Risk*

**Status:** Acknowledged

**File(s) affected:** `Proxy.sol`

**Description:** In the constructor of `Proxy.sol`there is no verification as to whether or not `contractLogic` contains code. This creates a risk for human errors were during the `Proxy` contract deployment where an arbitrary externally owned account may be stored instead of a contract account inside the proxy. Furthermore, executing transactions on a non-contract address will not throw any errors, which can be problematic.

**Recommendation:** There should be a check for the parameter `contractLogic` so see if the address contains a contract.
**Update:** Addressed in README documentation.

### QSP-5 Allowance Double-Spend Exploit

Severity: *Informational*

**Status:** Acknowledged

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

**Exploit Scenario:** 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 the use of functions that increase/decrease the allowance relative to its current value, such as `increaseAllowance` and `decreaseAllowance`.

**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. Teams who decide to wait for such a standard should make these recommendations to app developers who work with their token contract. The possibility of this exploit and its associated risks should be clearly documented to the users.
**Update:** Addressed in README documentation.

### Automated Analyses

#### Mythril

The Mythril analysis was completed successfully. No issues were detected.

#### Securify

We have analyzed each vulnerability reported by Securify, filtering out false positives. The issues after the filtering have already been mentioned in the rest of this report.

#### Slither

Slither suggests that the following methods should be declared `external`:

- `RevokerRole.addRevoker` (roles/RevokerRole.sol#31-33),
- `RevokerRole.removeRevoker` (roles/RevokerRole.sol#35-37),
- `WhitelisterRole.addWhitelister` (roles/WhitelisterRole.sol#31-33),
- `WhitelisterRole.removeWhitelister` (roles/WhitelisterRole.sol#35-37),
- `PauserRole.addPauser` (roles/PauserRole.sol#31-33),
- `PauserRole.removePauser` (roles/PauserRole.sol#35-37),
- `OwnerRole.addOwner` (roles/OwnerRole.sol#22-24),
- `OwnerRole.removeOwner` (roles/OwnerRole.sol#26-29),
- `Burnable.burn` (capabilities/Burnable.sol#18-20),
- `Pausable.pause` (capabilities/Pausable.sol#69-71),
- `Pausable.unpause` (capabilities/Pausable.sol#76-78),
- `TokenSoftToken` (TokenSoftToken.sol#29-37),
- `TokenSoftToken.updateCodeAddress` (TokenSoftToken.sol#42-44),
- `Whitelistable.setWhitelistEnabled` (capabilities/Whitelistable.sol#116-118),
- `Whitelistable.addToWhitelist` (capabilities/Whitelistable.sol#123-125),
- `Whitelistable.removeFromWhitelist` (capabilities/Whitelistable.sol#130-132),
- `Whitelistable.updateOutboundWhitelistEnabled` (capabilities/Whitelistable.sol#137-139),
- `Mintable.mint` (capabilities/Mintable.sol#18-20),
- `Revocable.revoke` (capabilities/Revocable.sol#25-27),
- `Proxiable.getLogicAddress` (capabilities/Proxiable.sol#20-24),
- `Proxiable.proxiableUUID` (capabilities/Proxiable.sol#26-28),
- `MinterRole.addMinter` (roles/MinterRole.sol#31-33),
- `MinterRole.removeMinter` (roles/MinterRole.sol#35-37),
- `BurnerRole.addBurner` (roles/BurnerRole.sol#31-33),
- `BurnerRole.removeBurner` (roles/BurnerRole.sol#35-37)

## Adherence to Specification

The code adheres to the specification very well. We have found no deviations.

## Code Documentation

Most of the core methods are accompanied by good documentation. However, for all public and external methods, we recommend adding developer documentation stating the semantics of the method, its parameters, and return values. There seems to also exist typos in the documents `TokenSoftToken.sol`(@ line 40: "retricted") and `Whitelistable.sol`(@ line 6: "reciever").

## Adherence to Best Practices

- The value `0xc5f16f0fcc639fa48a6947836d9850f504798523bf8c9a3a87d5876cf622bcf7` should be constrained inside a `byte32` constant in `Proxy.sol` and `Proxiable.sol`

### Test Results

**Test Suite Results**

All tests are currently passing.

```
Compiling your contracts...
===========================
> Compiling ./contracts/ERC1404.sol
> Compiling ./contracts/Migrations.sol
> Compiling ./contracts/Proxy.sol
> Compiling ./contracts/TokenSoftToken.sol
> Compiling ./contracts/capabilities/Burnable.sol
> Compiling ./contracts/capabilities/Mintable.sol
> Compiling ./contracts/capabilities/Pausable.sol
> Compiling ./contracts/capabilities/Proxiable.sol
> Compiling ./contracts/capabilities/Revocable.sol
> Compiling ./contracts/capabilities/Whitelistable.sol
> Compiling ./contracts/roles/BurnerRole.sol
> Compiling ./contracts/roles/MinterRole.sol
> Compiling ./contracts/roles/OwnerRole.sol
> Compiling ./contracts/roles/PauserRole.sol
> Compiling ./contracts/roles/RevokerRole.sol
> Compiling ./contracts/roles/WhitelisterRole.sol
> Compiling ./contracts/testing/Escrowable.sol
> Compiling ./contracts/testing/EscrowerRole.sol
> Compiling ./contracts/testing/TokenSoftTokenEscrow.sol
> Compiling ./contracts/testing/TokenSoftTokenEscrowNotProxiable.sol
> Compiling @openzeppelin/contracts-ethereum-package/contracts/GSN/Context.sol
> Compiling @openzeppelin/contracts-ethereum-package/contracts/access/Roles.sol
> Compiling @openzeppelin/contracts-ethereum-package/contracts/math/SafeMath.sol
> Compiling @openzeppelin/contracts-ethereum-package/contracts/token/ERC20/ERC20.sol
> Compiling @openzeppelin/contracts-ethereum-package/contracts/token/ERC20/ERC20Detailed.sol
> Compiling @openzeppelin/contracts-ethereum-package/contracts/token/ERC20/IERC20.sol
> Compiling @openzeppelin/upgrades/contracts/Initializable.sol
> Artifacts written to /home/wlfeng/Quantstamp/smart_contracts/tokensoft_token-master/tokensoft_token-
master/build/contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang

Starting our own ganache instance
Truffle v5.1.28 (core: 5.1.28)
Solidity - 0.5.16 (solc-js)
Node v10.19.0
Web3.js v1.2.1
Using network 'test'.


Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.



  Contract: 1404 Restrictions
     ✓ should fail with non whitelisted accounts (740ms)
     ✓ Should fail when paused (99ms)
     ✓ should allow whitelists to be removed (72ms)
     ✓ should handle unknown error codes

  Contract: Burnable
  Contract: Burnable
     ✓ Admin should be able to burn tokens from any account (182ms)
     ✓ Non admins should not be able to burn tokens (148ms)
     ✓ should emit event when tokens are burnd (98ms)

  Contract: Mintable
     ✓ Owner should be able to mint tokenst (122ms)
     ✓ Non-MinterRole accounts should not be able to mint tokens to any account (41ms)
     ✓ should emit event when tokens are minted (70ms)

  Contract: Pauseable
     ✓ Owner should be able to pause contract (91ms)
     ✓ Only owner should be able to pause contract (105ms)
     ✓ Paused contract should prevent all transfers (391ms)
     ✓ should emit event when contract is unpaused (62ms)
     ✓ should emit event when contract is unpaused (93ms)
     ✓ should not allow calls when already in the state (155ms)

  Contract: Upgradeable
     ✓ Can upgrade to proxiable contract (55ms)
     ✓ Cannot upgrade to non proxiable contract (67ms)
     ✓ Upgrading contract fires event (40ms)
     ✓ Contract cannot be upgraded by non owner (40ms)
     ✓ Transfer rules can be upgraded (271ms)
     ✓ Balance are maintained after upgrade (160ms)

  Contract: Revocable
     ✓ Admin should be able to revoke tokens from any account (162ms)
     ✓ Non admins should not be able to revoke tokens (188ms)
```

```
    ✓ should emit event when tokens are revoked (137ms)

  Contract: Whitelistable
    ✓ should allow adding and removing an address to a whitelist (361ms)
    ✓ should only allow admins adding or removing on whitelists (315ms)
    ✓ should validate if addresses are not on a whitelist (523ms)
    ✓ should trigger events (190ms)
    ✓ should validate outbound whitelist enabled flag (345ms)
    ✓ should trigger events for whitelist enable/disable (139ms)
    ✓ should not allow adding an address to invalid whitelist ID (0) (103ms)
    ✓ should allow disabling and re-enabling the whitelist logic (231ms)
    ✓ should not allow non-owner disabling the whitelist logic (47ms)

  Contract: BurnerRole
    ✓ should allow an owner to add/remove burners (104ms)
    ✓ should not allow a non owner to add/remove burners (145ms)
    ✓ should emit events for adding burners
    ✓ should emit events for removing burners (52ms)
    ✓ owner can add and remove themselves (73ms)

  Contract: MinterRole
    ✓ should allow an owner to add/remove minters (183ms)
    ✓ should not allow a non owner to add/remove minters (117ms)
    ✓ should emit events for adding minters
    ✓ should emit events for removing minters (58ms)
    ✓ owner can add and remove themselves (97ms)

  Contract: OwnerRole
    ✓ should allow an owner to add/remove owners (118ms)
    ✓ should not allow an owner to remove itself
    ✓ should not allow a non owner to add/remove owners (86ms)
    ✓ should emit events for adding owners
    ✓ should emit events for removing owners (62ms)

  Contract: PauserRole
    ✓ should allow an owner to add/remove pausers (94ms)
    ✓ should not allow a non owner to add/remove pausers (135ms)
    ✓ should emit events for adding pausers
    ✓ should emit events for removing pausers (57ms)
    ✓ owner can add and remove themselves (74ms)

  Contract: RevokerRole
    ✓ should allow an owner to add/remove revokers (97ms)
    ✓ should not allow a non owner to add/remove revokers (98ms)
    ✓ should emit events for adding revokers
    ✓ should emit events for removing revokers (60ms)
    ✓ owner can add and remove themselves (76ms)

  Contract: WhitelisterRole
    ✓ should allow an owner to add/remove whitelisters (90ms)
    ✓ should not allow a non owner to add/remove whitelisters (92ms)
    ✓ should emit events for adding whitelisters (41ms)
    ✓ should emit events for removing whitelisters (56ms)
    ✓ owner can add and remove themselves (76ms)

  Contract: TokenSoftToken
    ✓ should deploy
    ✓ should have correct details set (50ms)
    ✓ should mint tokens to owner (59ms)
    ✓ should mint tokens to different owner (182ms)

  Contract: Transfers
    ✓ All users should be blocked from sending to non whitelisted non role-assigned accounts (202ms)
    ✓ Initial transfers should fail but succeed after white listing (646ms)


  70 passing (28s)
```

## Code Coverage

Test coverage is evaluated as excellent. The test coverage metric is above 100% for statements and 100% for branches. However, we have noticed that there is only 1 assertion in `Transfer.js`. As a followup, adding additional assertions `Transfer.js`could provide more robust coverage for `TokenSoftToken.sol`.

| File | % Stmts | % Branch | % Funcs | % Lines | Uncovered Lines |
|---|---|---|---|---|---|
| **contracts/** | 100 | 100 | 100 | 100 | |
| ERC1404.sol | 100 | 100 | 100 | 100 | |
| Proxy.sol | 100 | 100 | 100 | 100 | |
| TokenSoftToken.sol | 100 | 100 | 100 | 100 | |
| **contracts/capabilities/** | 100 | 100 | 100 | 100 | |
| Burnable.sol | 100 | 100 | 100 | 100 | |
| Mintable.sol | 100 | 100 | 100 | 100 | |
| Pausable.sol | 100 | 100 | 100 | 100 | |
| Proxiable.sol | 100 | 100 | 100 | 100 | |
| Revocable.sol | 100 | 100 | 100 | 100 | |
| Whitelistable.sol | 100 | 100 | 100 | 100 | |
| **contracts/roles/** | 100 | 100 | 100 | 100 | |
| BurnerRole.sol | 100 | 100 | 100 | 100 | |
| MinterRole.sol | 100 | 100 | 100 | 100 | |
| OwnerRole.sol | 100 | 100 | 100 | 100 | |
| PauserRole.sol | 100 | 100 | 100 | 100 | |
| RevokerRole.sol | 100 | 100 | 100 | 100 | |
| WhitelisterRole.sol | 100 | 100 | 100 | 100 | |
| **contracts/testing/** | 31.15 | 15 | 37.5 | 32.26 | |
| Escrowable.sol | 34.29 | 12.5 | 42.86 | 34.29 | … 162,163,164 |
| EscrowerRole.sol | 55.56 | 25 | 66.67 | 60 | 27,28,36,37 |
| TokenSoftTokenEscrow.sol | 22.22 | 100 | 33.33 | 22.22 | … 35,42,49,78 |
| TokenSoftTokenEscrowNotProxiable.sol | 0 | 100 | 0 | 0 | … 41,48,58,70 |
| **All files** | **76.92** | **72.58** | **84.04** | **78.57** | |

# 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

```
dcb72fc96ef74fbf965afa39939eb821c453cc7538e7bcae6eaec05e7fd8b3a2 ./contracts/ERC1404.sol
728612063ac7e53093d7b3f00b88bc1627e3f1e6b7abe6988f6cb08a1978a15e ./contracts/Migrations.sol
a3a90c12fb3ee950a95c41ee4dc83e353fd9fdaaf5f071e2f5c1b9c6a36ef028 ./contracts/Proxy.sol
acf7ca4c70a2a3cc1e00edb7f10618953f0f4bfc4b51fb2db25ea4a066cd348d ./contracts/TokenSoftToken.sol
ad0bd96145826be8a5cfaf0fbc22c96dd0c808a3525b764099067d8dc5391fd6 ./contracts/testing/Escrowable.sol
e23d37022bb225a7d45bbd1764c93c780376b02a1d44df0a6c9ac5d1dcbf5493 ./contracts/testing/EscrowerRole.sol
d81ac5abfc25f77cd5bcbf6dab91671c742ec9fad8a36a126dc96258093a1fa9 ./contracts/testing/TokenSoftTokenEscrow.sol
37758701150c5f7449690df579049147b7df054627ffff32ead659a6c40ac3d2 ./contracts/testing/TokenSoftTokenEscrowNotProxiable.sol
bdd2c0884a33694929d0ef5247afce55bba998899835200d9cdaf96e02782a93 ./contracts/roles/BurnerRole.sol
156addaf2baec8ce3c285c6931a8482b865767fd2bbaf0a15b6317b88c0c1267 ./contracts/roles/MinterRole.sol
0054ce118e91ae893d7a5bb0a0936aa007b39531bfb5fd9ce688d977ff486cae ./contracts/roles/OwnerRole.sol
57abe71521e6c8302271d6781ab8c8be945cd7e93cbffabfe9da5ec057a853d2 ./contracts/roles/PauserRole.sol
d9d91c9e4d715460c894fba1ff30a20890fb5c452c7a298d754a5a30d9cc6211 ./contracts/roles/RevokerRole.sol
5917eccd8511d66c93c6729596cc663b033c0659737ce23716f0373412e55fb8 ./contracts/roles/WhitelisterRole.sol
941513fde0b712d9e74a4ead2e13c9d2050bb01c74019485b12da914e928d91c ./contracts/capabilities/Burnable.sol
cf344a98e79efe40d05f63aa34ff14ed1340bd505c680f18a418854d54aa4910 ./contracts/capabilities/Mintable.sol
522d18eabbae3b55d53e9c6453a44cb49fa2c7993724e366bcbd79cceb3c01f4 ./contracts/capabilities/Pausable.sol
98cf1535c87dfa4439b6942de48ed4fdc7bf35d1492b1c181e690030334ee4c7 ./contracts/capabilities/Proxiable.sol
9c6f515bcc9b299ad5d7b1b69e38d829272898d2d16eeac75560b47824f72e2e ./contracts/capabilities/Revocable.sol
3642720c12efada92795dfad2b30ffd6fe7533fa3669ff9ddb4edbf647cb3e6f ./contracts/capabilities/Whitelistable.sol
```

## About Quantstamp

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

Quantstamp's team boasts decades of combined experience in formal verification, static analysis, and software verification. Collectively, our individuals have over 500 Google scholar citations and numerous published papers. In its mission to proliferate development and adoption of blockchain applications, Quantstamp is also developing a new protocol for smart contract verification to help smart contract developers and projects worldwide to perform cost-effective smart contract security audits.

To date, Quantstamp has helped to secure hundreds of millions of dollars of transaction value in smart contracts and has assisted dozens of blockchain projects globally with its white glove security auditing 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.

Finally, Quantstamp's dedication to research and development in the form of collaborations with leading academic institutions such as National University of Singapore and MIT (Massachusetts Institute of Technology) reflects Quantstamp's commitment to enable world-class smart contract innovation.