**Quantstamp** Security Assessment Certificate

QUANTSTAMP VERIFIED SECURITY CERTIFICATE

October 19th 2021 — Quantstamp Verified

# Merit DAO Token

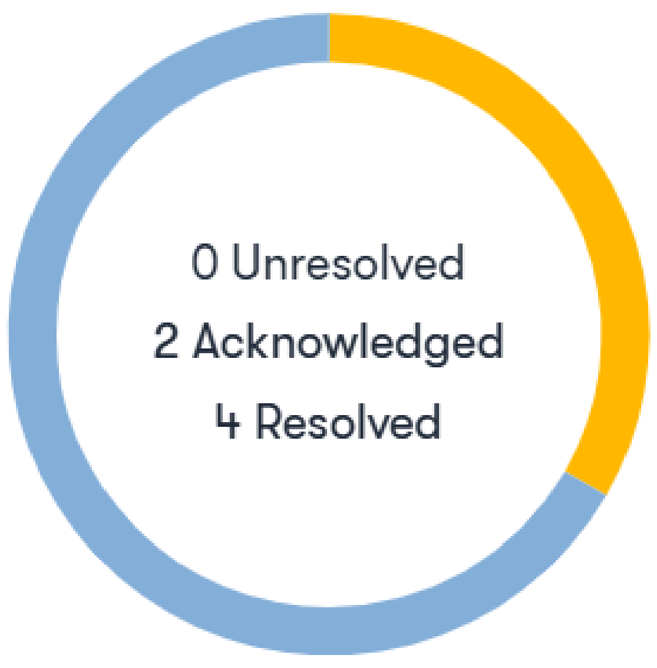
This audit report was prepared by Quantstamp, the leader in blockchain security.

## Executive Summary

| | |
|---|---|
| Type | ERC20 |
| Auditors | Jan Gorzny, Blockchain Researcher<br>Roman Rohleder, Research Engineer |
| Timeline | 2021-10-18 through 2021-10-19 |
| EVM | London |
| Languages | Solidity |
| Methods | Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review |
| Specification | Introducing Merit (Medium) |
| Documentation Quality | Undetermined |
| Test Quality | Undetermined |
| Source Code | |

| Repository | Commit |
|---|---|
| merit-dao | 8e9627a |

| | |
|---|---|
| Total Issues | **6** (4 Resolved) |
| High Risk Issues | **2** (2 Resolved) |
| Medium Risk Issues | **1** (0 Resolved) |
| Low Risk Issues | **3** (2 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 has reviewed the Merit DAO token, which is a simple implementation of an ERC20 with the ability to mint and burn tokens. The code is straightforward. Two issues, privileged roles to mint and burn and the well-known front-running issue of `approve` on ERC20 tokens, are by design, while the remaining issues (unlocked pragma, greedy token contract, incomplete role setup, and a complete lack of tests) were addressed by the team.
Note that only `MeritToken.sol` was in scope for this report.

| ID | Description | Severity | Status |
|---|---|---|---|
| QSP-1 | No Tests | High | Fixed |
| QSP-2 | Unfinished Role Setup | High | Fixed |
| QSP-3 | Privileged Roles and Ownership | Medium | Acknowledged |
| QSP-4 | Unlocked Pragma | Low | Fixed |
| QSP-5 | Greedy Contract | Low | Fixed |
| QSP-6 | Race Conditions / Front-Running | 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

Steps taken to run the tools:

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

## Findings

### QSP-1 No Tests

*Severity: High Risk*

**Status:** Fixed

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

**Description:** There are no tests. The tests included in the repository (and therefore this report) are leftover from the framework used to build the contracts (hardhat), and are not relevant to the contracts reviewed. Moreover, they do not compile or execute.

**Recommendation:** Add unit and/or functional tests for this file.

**Update:** This issue has been resolved (by adding tests).

### QSP-2 Unfinished Role Setup

*Severity: High Risk*

**Status:** Fixed

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

**Description:** Although the contract `MeritToken.sol` inherits from `AccessControlEnumerable` and two roles `MINTER_ROLE` and `BURNER_ROLE` are provided as state variables, neither they nor the `DEFAULT_ADMIN_ROLE` have been setup during the constructor call, therefore effectively preventing from granting or revoking any roles after deployment or minting and burning any tokens.

**Recommendation:** Setup an initial admin within the `constructor`, who will be able to grant and revoke further roles, i.e. by adding `_setupRole(DEFAULT_ADMIN_ROLE, _msgSender());` in line 18.

**Update:** This has been resolved (by assigning the `DEFAULT_ADMIN_ROLE` in the constructor).

### QSP-3 Privileged Roles and Ownership

*Severity: Medium Risk*

**Status:** Acknowledged

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

**Description:** There are privileged roles encoded in the ERC20, namely, `MINTER_ROLE` and `BURNER_ROLE`, who are able to mint arbitrarily many tokens (to any address), and burn any tokens (from any address).

**Recommendation:** This centralization of power needs to be made clear to the users, especially depending on the level of privilege the contract allows to those with the specified roles.

**Update:** This has been acknowledged: "[this] will be documented in the docs and only assigned to the DAO and multisig."

### QSP-4 Unlocked Pragma

*Severity: Low Risk*

**Status:** Fixed

**File(s) affected:** `MeritToken.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 issue has been resolved (by locking the pragma).

### QSP-5 Greedy Contract

*Severity: Low Risk*

**Status:** Fixed

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

**Description:** A greedy contract is a contract that can receive tokens which can never be redeemed. In this case, one can transfer tokens to this contract, which is not recommended.

**Recommendation:** Although this contract can have the balance of the tokens it owns wiped out by anyone with the "burner" role, users could transfer their tokens to this contract (which will have a non-zero address), effectively burning the tokens. This may occur accidentally (causing an annoyance) or intentionally (perhaps to simulate the "burner" role).
Prevent this action by forbidding tokens defined by this contract to be transferred to this contract (by adding a `require(recipient != address(this))` check in an over-ridden version of the `_transfer` function).

**Update:** The issue has been resolved (by adding an appropriate check).

### QSP-6 Race Conditions / Front-Running

*Severity: Low Risk*

**Status:** Acknowledged

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

**Description:** A block is an ordered collection of transactions from all around the network. It's possible for the ordering of these transactions to manipulate the end result of a block. A miner attacker can take advantage of this by generating and moving transactions in a way that benefits themselves.

**Exploit Scenario:** Imagine two friends — Alice and Bob.

1. Alice decides to allow Bob to spend some of her funds, for example, 1000 tokens. She calls the approve function with the argument equal to 1000.
2. Alice rethinks her previous decision and now she wants to allow Bob to spend only 300 tokens. She calls the approve function again with the argument value equal to 300.
3. Bob notices the second transaction before it is actually mined. He quickly sends the transaction that calls the `transferFrom` function and spends 1000 tokens.
4. Since Bob is smart, he sets very high fee for his transaction, so that miner will definitely want to include his transaction in the block. If Bob is as quick as he is generous, his transaction will be executed before the Alice's one.
5. In that case, Bob has already spent 1000 Alice's tokens. The number of Alice's tokens that Bob can transfer is equal to zero.
6. Then the Alice's second transaction is mined. That means, that the Bob's allowance is set to 300.
7. Now Bob can spend 300 more tokens by calling the `transferFrom` function. As a result, Bob has spent 1300 tokens. Alice has lost 1000 tokens and one friend.

**Recommendation:** Make this issue well known such that users who use the allowance feature would be aware of it in such transitions.

**Update:** This has been acknowledged: "[this is] default ERC20 behavior and enforcing re-setting approval to 0 first causes more issues with integrations".

## Automated Analyses

### Slither

Slither did not report any major issues.

## Code Documentation

1. In line 13 of `merit-dao/contracts/MeritToken.sol` the require error message for the `onlyHasRole` modifier states "GovToken.onlyHasRole: msg.sender does not have role", however, the contracts name is `MeritToken` not `GovToken`. The beginning of the error message should be changed accordingly to `MeritToken`. **Update:** the message has been changed.

## Test Results

### Test Suite Results

```
MeritToken
  contructor
    ✓ Constructor args should be used
    ✓ Should assign DEFAULT_ADMIN_ROLE to deployer
  mint
    ✓ Should work when calling from address which has MINTER_ROLE (76ms)
    ✓ Should revert when called from address without MINTER_ROLE
  burn
    ✓ Should work when calling from address which has BURNER_ROLE (73ms)
    ✓ Should revert when called from address without BURNER_ROLE
  transfer
    ✓ transfer to token contract should fail
    ✓ transfer should work normally (44ms)


 8 passing (2s)
```

## Code Coverage

Quantstamp was unable to compute the code coverage of the tests.

## 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

`6fbaeae882645d6d3ae9e66d5b71969f0b2cab21f1e7e01929bf55c294eb90da ./contracts/MeritToken.sol`

### Tests

`ce67e4db2dea8dccfbc829806295260a6a07fcb2eed6d179b67f3fd8f5138cae ./test/MeritToken.ts`

## Changelog

- 2021-10-18 - Initial report [`34ffd3f`]
- 2021-10-19 - Revised report [`8e9627a`]

## 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.