## Quantstamp Contract Security Certificate

April 15th 2019 — Quantstamp Verified

Mainframe Secondary Audit

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

### **Executive Summary**

Type

**Auditors** Nadir Akhtar, Software Auditing Intern Martin Derka, Senior Research Engineer Timeline 2018-07-03 through 2018-07-04 **EVM** Byzantium Languages Solidity Methods Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review Specification Mainframe Token Contracts README Github repository instruction (ERC20 branch)

Token Contract

Source Code Repository Commit

contracts (branch: ERC20) <u>162b46a</u> 2 (O Resolved) **Total Issues** 

High Risk Issues Medium Risk Issues 2 issues Low Risk Issues 0 2 (O Resolved) Informational Risk Issues 0 **Undetermined Risk Issues** 

Goals

A High The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to ^ Medium

**Overall Assessment** 

**Severity Categories** 

catastrophic impact for client's reputation or serious financial implications for client and users. 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 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.

The modified token contracts are secure. Only one method was found to be at risk

of failing, but as it is callable by the owner only, it poses no danger to other users.

### • 2018-07-04 - Initial report

this review.

Changelog

Quantstamp Audit Breakdown

#### intended to be supplementary to the previous audit, meaning that unresolved vulnerabilities from the previous audit may not be noted in this report. The MainframeToken.sol contract transitioned from ERC827 to ERC20. Only the new token and its new structure's implications on the rest of the contracts are in scope for

Possible issues we looked for included (but are not limited to): • Transaction-ordering dependence • Timestamp dependence • Mishandled exceptions and call stack limits

This Mainframe Token Security Audit is to provide the Mainframe team with a cursory look into their ERC20 fork of their contracts GitHub repository. The report aims to

identify any issues or vulnerabilities arising from the transition from an ERC827 token standard to an ERC20 token standard. Because of limited time to perform this audit,

only the contracts MainframeToken.sol and MainframeTokenDistribution.sol were reviewed, as they are the only ones substantially affected by the transition. This is

• Integer overflow / underflow

- Number rounding errors

• Unsafe external calls

- 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:
  - Code review that includes the following Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart

contract

describe.

### Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp

Testing and automated analysis that includes the following: 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

established industry and academic practices, recommendations, and research.

The below notes outline the setup and steps performed in the process of this audit.

- those test cases.
  - Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the
- **Toolset**

Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

- Setup Tool Setup:
- Truffle v4.1.12

### Steps taken to run the tools:

• Ganache v1.1.0

• Oyente v0.2.7

• Mythril v0.18.9

• MAIAN commit sha: ab387e1

5. Flattened the source code using truffle-flattener to accommodate the auditing tools. 6. Installed the Mythril tool from Pypi: pip3 install mythril

1. Installed Truffle: npm install -g truffle

2. Installed Ganache: npm install -g ganache-cli

8. Installed the Oyente tool from Docker: docker pull luongnguyen/oyente 9. Migrated files into Oyente (root directory): docker run -v \$(pwd):/tmp - it luongnguyen/oyente

7. Ran the Mythril tool on each contract: myth -x path/to/contract

10. Ran the Oyente tool on each contract: cd /oyente/oyente && python oyente.py /tmp/path/to/contract 11. Cloned the MAIAN tool: git clone --depth 1 https://github.com/MAIAN-tool/MAIAN.git maian

3. Installed the solidity-coverage tool (within the project's root directory): npm install --save-dev solidity-coverage

4. Ran the coverage tool from the project's root directory: ./node\_modules/.bin/solidity-coverage

- 12. Ran the MAIAN tool on each contract: cd maian/tool/ && python3 maian.py -s path/to/contract contract.sol

#### Contract(s) affected: Description: In smart contracts, for loops are often prone to vulnerabilities given the nature of transactions and the concept of gas: The gas necessary for executing such a loop is proportional to the number of iterations. Using is safe when the number of iterations is predictable, but against the best practices if unknown.

Severity: Informational

Assessment

Findings

#### it would revert all progress thus far. To mitigate both these issues, it is much safer to break that functionality into smaller, digestible pieces. Recommendation: The Quantstamp team recommends replacing the loop with method with signature sendTokens(address tokenOwner, address recipient, uint256 value) that implements a single iteration of that loop, and calling this method for every recipient instead. To ensure no recipient receives tokens twice, the contract can be

Other Issues

Method with Unlimited Gas Consumption

Severity: Informational Contract(s) affected: Description: These other issues, though not immediate security vulnerabilities, were still concerns to the Quantstamp team. If possible, take some time to fix the following

potential problems.

**Test Results** 

**Test Suite Results** 

Contract: MainframeStake

✓ should assign creator as owner

✓ should set correct required stake

✓ should change the required stake correctly (44ms)

✓ should unwhitelist address and return stake (521ms)

✓ should not allow transfer of ownership by non owner

✓ should not allow transfer when to address is invalid

✓ should not allow transfer by non owner when paused (93ms)

✓ should fail decrease approval when not tradeable (220ms)

✓ should fail to distribute if allowance too low (66ms)

✓ should fail to distribute if owners balance too low (76ms)

✓ should successfully drain mistakenly sent tokens (100ms)

✓ should allow transfer of tokens by owner when paused (213ms)

✓ should allow transfer of tokens by distributor when paused (265ms)

✓ should withdraw successfully if balance is high enough (391ms) ✓ should fail to refund of balances called by non owner (144ms)

✓ should whitelist address when staking (289ms)

• Both MainframeToken#47,51 and MainframeTokenDistribution#8,19 accept uint parameters. We suggest that those be turned into uint256. • Consider making the emergencyERC20Drain() functions of MainframeToken.sol identical to that of MainframeTokenDistribution.sol to drain any and all accidental tokens, as there is likely no scenario in which it would be preferable to only drain, say, half.

• In file MainframeToken.sol in the validDestination modifier, we suggest that require(address != 0x0) be included in this modifier as well.

In MainframeTokenDistribution.sol, the distributeTokens() function loops through all recipients passed to the smart contract. Because the number of recipients is

unbounded, it is possible that the transaction will consume so much gas that it will not be able to fit within a block. In addition, if the transferFrom() function ever failed,

maintain a mapping from address to bool tracking which recipients have received tokens, or require that mainframeToken.balanceOf(recipient) == 0).

✓ should fail to deposit tokens if balance too low (97ms) ✓ should fail to withdraw if balance too low ✓ should check address has stake (106ms) ✓ should allow owner to update required stake (41ms) ✓ should successfully drain mistakenly sent tokens (85ms) ✓ should deposit successfully when staking (232ms)

 $\checkmark$  should fail to stake if sender address is different to staker param when calling contract directly (68ms)

✓ should fail to refund balances if list size exceeds limit (201ms) ✓ should refund balances if called by owner (400ms) ✓ should successfully destroy itself if balance is unchanged (0) ✓ should successfully destroy itself if balance is 0 (195ms) ✓ should fail to destroy itself if balance is higher than 0 (97ms) ✓ should fail to destroy itself if someone sends tokens to the contract address (56ms) ✓ should drain only mistakenly sent tokens and not valid deposits (343ms)

**Uncovered Lines** 

133

65

... 16,17,18,19

- ✓ should extract correct balances through event logs (624ms) Contract: MainframeToken ✓ should be named Mainframe Token ✓ should have symbol MFT ✓ should have 18 decimals
- ✓ should assign creator as owner ✓ should have correct total supply ✓ should assign initial token supply to owner ✓ should allow transfer of ownership by owner to pending owner (76ms) ✓ should not allow ownership to be claimed by non pending owner (43ms)
- ✓ should not allow transfer when to address is token contract address ✓ should allow transfer of tokens by anyone when not paused (321ms) ✓ should not allow approve by non owner when paused (91ms) ✓ should not allow transferFrom when to address is invalid (101ms) ✓ should not allow transferFrom when to address is contract address (47ms) ✓ should allow transferFrom by owner when paused (284ms) ✓ should allow transferFrom by distributor when paused (377ms)
- ✓ should allow to decrease approval when tradeable (248ms) ✓ should fail increase approval when not tradeable (219ms) ✓ should allow to increase approval when tradeable (248ms) Contract: MainframeTokenDistribution  $\checkmark$  should fail to distribute if num of recipients and values do not match (43ms)

✓ should allow transferFrom by anyone when address has approved balance and not paused (320ms)

% Stmts

98.31

97.5

100

100

90.63

• The integer overflow in the SafeMath library, which is a false positive as it is covered by the library's assertions.

% Funcs

96.3

100

90

100

100

0

0

89.66

% Lines

97.1

97.83

92.86

100

100

0

0

89.33

3fe9c7f69428b501333b76b1c7fccceca6ef26485acd8aa500491dab5f7330e5

ab31dbe9822a46b2802dbd57ce1312e9a40c6d9cd0d2271a0c11bb4776cd7f45

c2ecc7b2412a7ce753525889b863c6cd82be72403badccfe7f0588e90d36110b

b08897db1b70c3cbbfe0d82e354837d5c3ccd8f9be8410b500ecfab8d5830f8a

./test/TestMainframeTokenDistribution.js

./test/TestMainframeStake.js

./test/TestMainframeToken.js

./test/utils.js

% Branch

80.77

75

100

83.33

100

100

100

80.77

52 passing (13s) Code Coverage

✓ should distribute correct token amounts to recipients when token paused (284ms)

StakeInterface.sol 100 contracts/mocks/ 0 TestHelper.sol 0

# Mythril reported the following issues

File

contracts/

All files

Oyente

Mythril

**Automated Analyses** 

Oyente reported the following issues

MainframeStake.sol

MainframeToken.sol

MainframeTokenDistribution.sol

File Signatures

Contracts 4318e60152f648e4fef2b55813e138cccfe96ef533bbab2b12b00ef60fd4038f

./contracts/MainframeTokenDistribution.sol

240b17a721e3d9e301f90a860c49b9988f8ad27a003de2a5d129c9d6a3d0f048

83133eb4afed383dd8dcb4ed29e1b998c93a8a40fec88f8afa3e0ebc74f5bbbd

138b99e76c412e6a7e5533c968aa07a7a7a69d43b93d83bf4062ffe9f380a5a4

dffa4902acdaa5e2fd6a5da539162680b4ae320a0d364c3e5ba870dc9f36dd49 ./contracts/MainframeToken.sol 1224c3fbb283e9cf9f238f8f5f41b4a566f60a3ec68b6154b653496005bd287a ./contracts/mocks/TestHelper.sol

./contracts/MainframeStake.sol

./contracts/StakeInterface.sol

./contracts/Migrations.sol

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

 Integer Overflow • State change after external call Message call to external contract Multiple Calls Exception state Upon further inspection, we classified these as false positives. MAIAN MAIAN did not report any issues. Appendix The following are the SHA-256 hashes of the audited contracts and test files. A smart contract or file with a different SHA-256 hash has been modified, intentionally or otherwise, after the audit. 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 audit.

**Tests** 

### 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

**About Quantstamp** 

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.

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.

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

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

Links to other websites

**Timeliness of content** 

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. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The Solidity language itself and other smart contract languages remain under development and are subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond Solidity or the smart contract programming language, or other programming aspects that could present security risks. You may risk loss of tokens, Ether, and/or other loss. A report is not an endorsement (or other opinion) of any particular project or team, and the report does not guarantee the security of any particular project. A report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. To the fullest extent permitted by law, we disclaim all warranties, express 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 website, or any website or mobile application featured in any banner or other 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. You may risk loss of QSP tokens or other loss. 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,

**Quantstamp** 

REGULATORY, OR OTHER ADVICE.