

Hub Token

April 15th 2019 — Quantstamp Verified

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

Executive Summary

Type

Auditors Martin Derka, Senior Research Engineer Kacper Bąk, Senior Research Engineer Sebastian Banescu, Senior Research Engineer Timeline 2019-01-22 through 2019-02-04

EVM Byzantium Solidity, Javascript Languages

Token Audit

Methods Architecture Review, Unit Testing, Functional Testing,

Computer-Aided Verification, Manual Review Specification None

Source Code Repository

14268d2 hub-ethereum-token **Total Issues 3** (2 Resolved)

0 High Risk Issues Medium Risk Issues 0

3 issues 0 3 (2 Resolved) 0

Low Risk Issues

Informational Risk Issues **Undetermined Risk Issues** Goals

Commit

the project. This makes the implementation simple. However, cloning the libraries contradicts best practices for the smart contract development. As a result of this,

A High

Overall Assessment

the test coverage is inadequate. The tests suite also contains several constants that need to be manually initialized which makes execution of test coverage tools out of the box impossible. Overall, the implementation of the token can be deemed secure, but the project code quality and setup can be improved. **Severity Categories**

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

The implementation of the Hub Token relies on external libraries that are cloned to

	financial implications for client and users.
^ Medium	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.

• 2019-01-23 - Initial report • 2019-02-04 - Revised report based on commit 1ebc22a

Changelog

Quantstamp Audit Breakdown

were audited. Possible issues we looked for included (but are not limited to):

• Timestamp dependence • Mishandled exceptions and call stack limits

This report focused on evaluating security of smart contracts, as requested by the Hub Token ERC20 team. Only the token smart contracts (along with their dependencies)

- Unsafe external calls
 - Integer overflow / underflow
 - Number rounding errors
 - Reentrancy and cross-function vulnerabilities

• Denial of service / logical oversights

- Centralization of power
- Code clones, functionality duplication • Gas usage

Code review that includes the following

The Quantstamp auditing process follows a routine series of steps:

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 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 describe.

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

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

10. Migrated files into Oyente (root directory): docker run -v \$(pwd):/tmp - it luongnguyen/oyente

- <u>Truffle v4.1.12</u>

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

• Ganache v1.1.0

• Mythril v0.2.7 • MAIAN commit sha: ab387e1

• Securify

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

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

- 7. Ran the Mythril tool on each contract: myth -x path/to/contract 8. Ran the Securify tool: java -Xmx6048m -jar securify-0.1.jar -fs contract.sol
- 11. Ran the Oyente tool on each contract: cd /oyente/oyente && python oyente.py /tmp/path/to/contract 12. Cloned the MAIAN tool: git clone --depth 1 https://github.com/MAIAN-tool/MAIAN.git maian 13. Ran the MAIAN tool on each contract: cd maian/tool/ && python3 maian.py -s path/to/contract contract.sol
- Allowance Double-Spend Exploit

Exploit Scenario:

Findings

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

somewhere

vulnerable to the double-spend exploit if these methods are not used.

Recommendation: The exploit (as described above) is mitigated through use of functions that increase/decrease the allowance relative to its current value, such as increaseApproval and decreaseApproval.

Contract(s) affected: HubToken.sol, BurnableToken.sol, StandardToken.sol, ERC20.sol, Pausable.sol

Unlocked Pragma Severity: Informational

The Quantstamp team recommends that the existence of increaseApproval and decreaseApproval be clearly communicated to the users. The Hub Token contract is still

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

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

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.

Description: The clone-and-own approach involves copying and adjusting open source code at one's own discretion. From the development perspective, it is initially beneficial as it reduces the amount of effort. However, from the security perspective, it involves 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. Recommendation: Rather than the clone-and-own approach, a good industry practice is to use the Truffle framework for managing library dependencies. This eliminates the clone-and-own risks yet allows for following best practices, such as, using libraries. The Quantstamp team recommends that the OpenZeppelin libraries be used as

Test Results Test Suite Results

async () => {

Contract: HubToken

√ should transfer (131ms)

} failed as expected

✓ should burnFrom (196ms)

√ should increase allowance (110ms)

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

managed dependencies instead of cloning contracts.

Clone-and-Own

Severity: Informational

async ()=> { await instance.transferFrom(HUB_TOKEN_OWNER_WALLET_ADDRESS, TEST_TARGET_WALLET_ADDRESS, 10, { from: PURSER_WALLET_ADDRESS }) } failed as expected async ()=> { await instance.burn(10);

Code Coverage

✓ should decrease allowance (119ms) 7 passing (1s)

await instance.transfer(TEST_TARGET_WALLET_ADDRESS, 10);

contracts/lifecycle/ 100 75 100 | 100 | 75 Pausable.sol 100 | 100 | 16.67 | 37.5 contracts/math/ 37.5 SafeMath.sol 37.5 | . . . 32,35,63,64 37.5 I 16.67 25 I 25 | contracts/ownership/ 33.33 33.33 | . . . 52,60,61,62 | Ownable.sol 25 | 25 40 contracts/token/ERC20/ | 78.95 | 91.3 50 BurnableToken.sol 19,28 33.33 50 33.33 100 ERC20.sol 100 100 | 100 100 | 27 HubToken.sol 80 80 100 | 100 StandardToken.sol 95.12 50 90.91 95.12 27,153 41.67 I 38.51 21.43 **Automated Analyses** Oyente Oyente did not detect any issues. Mythril

Securify reported missing input validations in contracts Ownable.sol and Pausable.sol. The Quantstamp team verified that all the reports are benign or false positives.

No specification was provided for this audit. The implemented token is a standard ERC20 token with pausability feature.

Maian did not detect any issues.

Adherence to Specification

MAIAN

Securify

Contracts 55a5b0e785e49777aac9361ab8fb22050b957fd816749519c5c1d7c3ec7dc98a ./contracts/token/ERC20/StandardToken.sol

./contracts/ownership/Ownable.sol

./contracts/Migrations.sol ./contracts/math/SafeMath.sol

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

Tests

./test/test.js

d39f16c48dde8e24cfa4741a5f2f6b9e258988abddeecc2f5f2e25f95e211c37

ed979bdd492b210048e0539357e413c0eaff9df06e588658aeab0266eaa538b1 7a088b95d0b62f96edbb1d9a4a9161a507a1a543510bb2bb043dbc002002970f

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.

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. **Timeliness of content**

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

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

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,

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. **Quantstamp*** Hub Token Audit

• Transaction-ordering dependence

 Access control • Business logic contradicting the specification

• Arbitrary token minting Methodology

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

Setup Tool Setup:

established industry and academic practices, recommendations, and research.

Steps taken to run the tools: 1. Installed Truffle: npm install -g truffle

• Oyente v1.2.5

9. Installed the Oyente tool from Docker: docker pull luongnguyen/oyente

- Assessment
- Severity: Informational
- Status: Fixed Contract(s) affected: HubToken.sol, BurnableToken.sol, StandardToken.sol Description: As it presently is constructed, the contract is vulnerable to the allowance double-spend exploit, as with other ERC20 tokens.
 - 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.

Status: Fixed Contract(s) affected: HubToken.sol, BurnableToken.sol, StandardToken.sol, ERC20.sol, Pausable.sol

} failed as expected async ()=> { await instance.burnFrom(HUB_TOKEN_OWNER_WALLET_ADDRESS, 10, { from: PURSER_WALLET_ADDRESS }); } failed as expected √ should pause transfers and burns (368ms) ✓ should approve allowance (113ms) √ should transferFrom (256ms)

4.55 I contracts/ 7.06 7.41 I 6.93 MultiSigWallet.sol 7.06 4.55 7.41 6.93 | . . . 389,390,391 |

Mythril detected potential integer overflow in method add() in the SafeMath library. This is a false positive.

% Funcs 1

Code Documentation The code is appropriately documented.

audit.

./contracts/token/ERC20/ERC20.sol

./contracts/MultiSigWallet.sol

./contracts/lifecycle/Pausable.sol

./contracts/token/ERC20/HubToken.sol

./contracts/token/ERC20/BurnableToken.sol

About Quantstamp

adoption of this exponentially growing technology.

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

Disclaimer

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,

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;

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

completeness of any outcome generated by such software.

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. 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

Adherence to Best Practices With the exception of the aforementioned issues, the code adheres to best practices for Solidity. Appendix File Signatures 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

18dde0e106decb6f4f11452a6de60cba1806e8b486c47b6ff1ebe5ee02d4df5c

ea492bb295be274e4e1c86f4538b3efcde6aa0e2993c61ecf308e9fd34504f24

b9cf9686555738975f7ed256c8617a3b8535b64dcb4c7b50c1779bb88fdbc15e

3279980b962a1374a710b936a067a5262ad2c3ea3778e2380989f0e39928baf4

6037cc1d2a8adff9501b652df2689b95f4435d9cd745a030f8a93527872ac2a9

45c062a2e7039e75c47b128c592a587e554b85aed97d52ffd9e5382219c3fe1b