Quantstamp Smart Contract Audit

DigitalBitsToken

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

Quantstamp helps to secure blockchain applications such as smart contracts. We are developing a new protocol for smart contract verification, performing professional audits and consultations, and developing security tools. Quantstamp also has expertise in application security and secure software development.

Executive Summary

Type	Token Contract Audit
Auditors	Sung-Shine Lee, Research Engineer Leonardo Passos, Senior Research Engineer Jan Gorzny, Blockchain Researcher
Timeline	2019-06-27 through 2019-06-27
EVM	Byzantium
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Source Code	

Repository	Commit
digital-bits-token	f774d481882383f7fa31f1975c11f925f766e5df

Total Issues	1	(1 Resolved)
High Risk Issues	0	
Medium Risk Issues	0	
Low Risk Issues	0	
Informational Risk Issues	1	(1 Resolved)
Undetermined Risk Issues	0	

1 issue

Overall Assessment

The code is simple, well-written, and conforms to the best practices. As any ERC20 token, it is vulnerable to allowance double-spend exploit.

Severity Categories	
High	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	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.

Goals

- This report focused on evaluating security of smart contracts
- as requested by the DigitalBits team.

Changelog

- 2019-06-27 - Initial Report

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

Setup

Tool Setup:

- Truffle v4.1.12
- Ganache v1.1.0
- Oyente v1.2.5
- Mythril v0.2.7
- MAIAN commit sha: ab387e1
- Securify
- NodeJS v10.15.1

Steps taken to run the tools:

1. Installed Truffle: `npm install -g truffle`
2. Installed Ganache: `npm install -g ganache-cli`
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`
5. Flattened the source code using `truffle-flattener` to accommodate the auditing tools.
6. Installed the Mythril tool from Pypi: `pip3 install mythril`
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`
9. Installed the Oyente tool from Docker: `docker pull luongnguyen/oyente`
10. Migrated files into Oyente (root directory): `docker run -v $(pwd):/tmp - it luongnguyen/oyente`
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`
14. Installed NodeJS from here

Assessment

Findings

Allowance Double-Spend Exploit

Severity: Informational

Status: Fixed

Contract(s) affected: `DigitalBitsToken.sol`,

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

Exploit Scenario: 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)

1. 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
2. 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
3. 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
4. Before Alice notices any irregularities, Bob calls `transferFrom()` method again, this time to transfer `M` Alice's tokens.

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

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.

Test Results

Test Suite Results

```
Compiling your contracts...
===========================
> Compiling ./contracts/DigitalBitsToken.sol
> Compiling ./contracts/Migrations.sol
> Compiling openzeppelin-solidity/contracts/math/SafeMath.sol
> Compiling openzeppelin-solidity/contracts/token/ERC20/ERC20.sol
> Compiling openzeppelin-solidity/contracts/token/ERC20/ERC20Burnable.sol
> Compiling openzeppelin-solidity/contracts/token/ERC20/ERC20Detailed.sol
> Compiling openzeppelin-solidity/contracts/token/ERC20/IERC20.sol
> Artifacts written to /var/folders/_6/mttd8_xn2ws0334gr46b9f_w0000gn/T/test-119527-41872-3w4u34.k2zxn
> Compiled successfully using:
   - solc: 0.4.26+commit.4563c3fc.Emscripten.clang


  Contract: DigitalBitsToken
digitalbits
XDB
<BN: 7>
<BN: de0b6b3a7640000>
<BN: de0b6b3a7640000>
     ✓ has a name (43ms)
     ✓ has a symbol
     ✓ has an amount of decimals
     ✓ has the correct initial supply (123ms)
     ✓ has initial supply is allocated to wallet


  5 passing (553ms)
```

Code Coverage

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	100	100	100	
DigitalBitsToken.sol	**100**	**100**	**100**	**100**	
All files	100	100	100	100	

Automated Analyses

Oyente

Oyente reported integer underflow in the function `name()` and `symbol()`. Upon closer inspection, we classified it as false positives.

Mythril

Mythril reported integer overflow and underflow in the SafeMath library. We have classified it as false positives.

MAIAN

MAIAN has not detected any issues.

Securify

Securify has not detected any issues.

Adherence to Specification

The code conforms to the specification.

Code Documentation

The code is well-documented.

Adherence to Best Practices

The code adheres to best practices.

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 audit.

Contracts

```
f5c08a2e1b7808516d571f8609880dc627948a3e54db7602c79910029951800a
./contracts/DigitalBitsToken.sol
```

Tests

```
3a632462a1d2ac0d862a8b32267382335954b58e616c5023833229e504b4a269
./test/DigitalBitsToken.test.js
```

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.

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. 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, REGULATORY, OR OTHER ADVICE.