Quantstamp Security Assessment Certificate

QUANTSTAMP VERIFIED · SECURITY CERTIFICATE

July 28th 2020 — Quantstamp Verified

DECA DEcentralized CArbon tokens - ITDE (initial token distribution event)

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

Executive Summary

Type	Token sale
Auditors	Kacper Bąk, Senior Research Engineer Ed Zulkoski, Senior Security Engineer Sung-Shine Lee, Research Engineer
Timeline	2019-09-30 through 2020-06-30
EVM	Byzantium
Languages	Solidity
Methods	Architecture Review, Computer-Aided Verification, Manual Review
Specification	
Source Code	

Repository	Commit
DCC	fc1c6377

Goals

- Can users purchase tokens?
- Does the sale conform to the provided specification?
- Can owner take advantage of the sale?

Total Issues	9	(9 Resolved)
High Risk Issues	4	(4 Resolved)
Medium Risk Issues	1	(1 Resolved)
Low Risk Issues	2	(2 Resolved)
Informational Risk Issues	2	(2 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

We have found a number of issues with the code. Importantly, the owner can perform arbitrary token minting and will receive a much higher number of tokens than the comment in the code would suggest. The code adheres to some best practices, but lacks a test suite.
Update (1): although the previously found issues have been mostly addressed, the team introduced two new high-severity issues. One of the issues could've been avoided, had the team not relied on clone-and-own code reuse and used OpenZeppelin ERC20 implementation instead. Furthermore, as is, the code is not fully ERC20-compatible. We recommend against deploying the current code.
Update (2): the two new high-severity issues have been addressed. We limited re-audit to `DECA_ERC20_0.5.3.sol`. `DECA_ERC20_0.4.26.sol` remains vulnerable.
Update (3): the team has addressed all our findings.
Update (4): we have reviewed the changed up to commit efc8046. One medium-severity issue was found.
Update (5): All issues have been addressed as of commit bcf24df.

ID	Description	Severity	Status
QSP-1	The owner would receive significantly more tokens than they should	High	Resolved
QSP-2	Arbitrary token minting by the owner	High	Resolved
QSP-3	Testing code manages bonus sales	High	Resolved
QSP-4	The function `approve()` does not set an allowance	High	Resolved
QSP-5	Integer Overflow / Underflow	Medium	Fixed
QSP-6	Allowance Double-Spend Exploit	Low	Resolved
QSP-7	Outdated Syntax	Low	Resolved
QSP-8	Unlocked Pragma	Informational	Resolved
QSP-9	Clone-and-Own	Informational	Resolved

Quantstamp Audit Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

Toolset

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

Setup

Tool Setup:

- Maian commit sha: ab387e1
- Mythril v0.2.7
- Securify None
- Slither v0.6.6

Steps taken to run the tools:

1. Cloned the MAIAN tool: `git clone --depth 1 https://github.com/MAIAN-tool/MAIAN.git maian`
2. Ran the MAIAN tool on each contract: `cd maian/tool/ && python3 maian.py -s path/to/contract contract.sol`
3. Installed the Mythril tool from Pypi: `pip3 install mythril`
4. Ran the Mythril tool on each contract: `myth -x path/to/contract`
5. Ran the Securify tool: `java -Xmx6048m -jar securify-0.1.jar -fs contract.sol`
6. Installed the Slither tool: `pip install slither-analyzer`
7. Run Slither from the project directory: `slither .s`

Findings

QSP-1 The owner would receive significantly more tokens than they should

Severity: *High Risk*

Status: Resolved

File(s) affected: `DECA_ERC20_0.4.18.sol`

Description: According to the comment in line 232, the owner is entitled to 2% of the tokens. In the implementation, however, (in lines 243 and 247: `safeDiv(tokens, percentage)`), the owner will actually get 50% of the tokens.

Recommendation: We recommend updating the code so that the owner is entitled to the 2% supply instead of 50%.
Update: the code has been updated, however, the line 126 refers to `README.md`, but the file contains no relevant information.

QSP-2 Arbitrary token minting by the owner

Severity: *High Risk*

Status: Resolved

File(s) affected: `DECA_ERC20_0.4.18.sol`

Description: The owner can mint an arbitrary amount of tokens for themselves by simply calling the payable function repeatedly (which sends the ETH right back to the owner). Since this is an uncapped token, users cannot know in advance what percentage of the total supply they will have at the end of the token sale.

Recommendation: We recommend updating the code in a way so that the fallback function does not send the funds to the owner but keeps them in the contract for as long as the token sale is pending.

QSP-3 Testing code manages bonus sales

Severity: High Risk

Status: Resolved

File(s) affected: `DECA_ERC20_0.5.3.sol`

Description: The lines 112-116 contain testing code as opposed to production code. Specifically, time-dependent bonuses are given according to intervals determined by hours instead of weeks.

Recommendation: We recommend updating the code so that it uses weeks instead of hours to determine bonuses.

QSP-4 The function `approve()` does not set an allowance

Severity: High Risk

Status: Resolved

File(s) affected: `DECA_ERC20_0.5.3.sol`

Description: The ERC20 function `approve()` is supposed to set an allowance. In the current implementation it only emits an event and returns a boolean value. Furthermore, it cannot be called if there is any existing allowance, which does not conform to the ERC20 standard.

Recommendation: We recommend: 1) adding the statement `allowed[msg.sender][spender] = tokens;` before line 156; and 2) removing the condition in line 155.
This issue could've been avoided, had the team followed our initial advice on using OpenZeppelin ERC20 implementation instead of clone-and-own code reuse.

QSP-5 Integer Overflow / Underflow

Severity: Medium Risk

Status: Fixed

File(s) affected: `DECA.sol`

Description: Integer overflow/underflow occur when an integer hits its bit-size limit. Every integer has a set range; when that range is passed, the value loops back around. A clock is a good analogy: at 11:59, the minute hand goes to 0, not 60, because 59 is the largest possible minute. Integer overflow and underflow may cause many unexpected kinds of behavior and was the core reason for the `batchOverflow` attack. Here's an example with `uint8` variables, meaning unsigned integers with a range of `0..255`. `function under_over_flow() public { uint8 num_players = 0; num_players = num_players - 1; // 0 - 1 now equals 255! if (num_players == 255) { emit LogUnderflow(); // underflow occurred } uint8 jackpot = 255; jackpot = jackpot + 1; // 255 + 1 now equals 0! if (jackpot == 0) { emit LogOverflow(); // overflow occurred } }`
In the function `appendWeeks()`, if the argument `appendWeeks` is too large, the `endDate` computation on L160 may overflow and cause the sale to immediately end.

Recommendation: Use the OpenZeppelin SafeMath library, or add a require-statement ensuring that the updated `endDate` will be greater than its previous value.

QSP-6 Allowance Double-Spend Exploit

Severity: Low Risk

Status: Resolved

File(s) affected: `DECA_ERC20_0.4.18.sol`

Related Issue(s): SWC-114

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)
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.

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.

QSP-7 Outdated Syntax

Severity: Low Risk

Status: Resolved

File(s) affected: `DECA_ERC20_0.4.18.sol`

Related Issue(s): SWC-102

Description: As security standards develop, so does the Solidity language. In order to stay up to date with current practices, it's important to use a recent version of Solidity and recent conventions. For example, the attribute `constant` on functions is deprecated.

Recommendation: We recommend updating the required compiler version to at least 0.5.3.

QSP-8 Unlocked Pragma

Severity: Informational

Status: Resolved

File(s) affected: `DECA_ERC20_0.4.18.sol`

Related Issue(s): SWC-103

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.

QSP-9 Clone-and-Own

Severity: Informational

Status: Resolved

File(s) affected: `DECA_ERC20_0.4.18.sol`

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. Specifically, we noticed code cloned from OpenZeppelin 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.
Update: as of commit `f609015` the contract cloned the code of OpenZeppelin `Ownable` and `Pausable` contracts because the owner is not `payable` in OpenZeppelin contracts. Another possibility would be to reuse the OpenZeppelin contracts and then convert the owner address to `payable` as needed.

Adherence to Specification

There is no specification besides some inline comments in the code.
Although it is not clear whether the token should support burning or not, tokens can be "effectively burned" by sending them to address `0x0`. It is possible since the function `transfer()`does not enforce that the value of `to` is different from `0x0`. Furthermore, the function `totalSupply()`, in line 144 excludes the balance of the address `0x0` from the total supply. **Update:** fixed.
The meaning of `_CCDBAddress` is unclear. If it is meant to be a URL, the `string` type is fine, but if it's an Ethereum address, it should not be of type `string`. **Update:** fixed.

Code Documentation

The code is well documented.

Adherence to Best Practices

The code adheres to some best practices, however:

- **Update:** fixed. it relies on clone-and-own for library reuse,
- **Update:** fixed. it has no test suite,
- **Update:** fixed. the field `startDate` appears to be unused,
- **Update:** fixed. lines 247 and 243 repeat computations,
- **Update:** fixed. the fields in lines 109-117 should be declared constants (with the exception of `_totalSupply` if there is no cap on the supply).
- `divBy` can be declared as a contract's constant.
- Undocumented constants in lines 128-134.

Test Results

Test Suite Results

```
  Contract: DECA
    check pause
'0xDf08F82De32B8d460adbE8D72043E3a7e25A3B39'
'999999999999999999999999999999999999999999999999999999999999999999999999993344259999999999'
      ✓ should get/set pause (129ms)
'0x6704Fbfcd5Ef766B287262fA2281C105d57246a6'
'999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999'
      ✓ should fail on pay (194ms)
'0x9E1Ef1eC212F5DFfB41d35d9E5c14054F26c6560'
'999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999'
'0xce42bdB34189a93c55De250E011c68FaeE374Dd3'
'999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999'
      ✓ check intruder pause (195ms)
    check crowdsale dates
'0x97A3FC5Ee46852C1Cf92A97B7BaD42F2622267cC'
'999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999'
'0xB9dcBf8A52Edc0C8DD9983fCc1d97b1F5d975Ed7'
'999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999'
      ✓ check preICOEnds (112ms)
'0x26064a2E2b568D9A6D01B93D039D1da9Cf2A58CD'
'999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999'
'0xe84Da28128a48Dd5585d1aBB1ba67276FdD70776'
'999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999'
      ✓ check bonus1Ends (106ms)
'0xCc036143C68A7A9a41558Eae739B428eCDe5EF66'
'999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999'
'0xE2b3204F29Ab45d5fd074Ff02aDE098FbC381D42'
```

```
'999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999'
       ✓ check bonus2Ends (107ms)
'0x040b798028e9abded00Bfc65e7CF01484013db17'
'999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999'
       ✓ check endDate (43ms)
'0x396e6fdcCcff2903f1ec095662FE496492E473a8'
'999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999'
       ✓ check appendWeeks (107ms)
    transferAnyERC20Token
'0x8B18A408FfEB025Bf89bA4A4cEB509997B179C16'
'999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999'
'0x2Ae0b32Dd2a99d3799F93b6040CE0AFB45BD3745'
'999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999'
AssertionError: transferAnyERC20Token should return positive result: expected true to equal { Object (tx, receipt, ...) }
    at Context.<anonymous> (/Users/ezulkosk/audits/dcc/test/DECA.js:240:24)
    at runMicrotasks (<anonymous>)
    at processTicksAndRejections (internal/process/task_queues.js:93:5) {
  message: 'transferAnyERC20Token should return positive result: expected true to equal { Object (tx, receipt, ...) }',
  showDiff: true,
  actual: true,
  expected: {
    tx: '0x97b41190e7478930f60c16dba23d0a895a9a18f265c588549b16c111034e4f57',
    receipt: {
      transactionHash: '0x97b41190e7478930f60c16dba23d0a895a9a18f265c588549b16c111034e4f57',
      transactionIndex: 0,
      blockHash: '0x08f963e07a4c4c2d3355f287b368f2dc26594b171a9883dac833cf871b3eaa8e',
      blockNumber: 33,
      from: '0x8b18a408ffeb025bf89ba4a4ceb509997b179c16',
      to: '0xff819ac59fae1647df9a74af7d6ed3216aa295ca',
      gasUsed: 24069,
      cumulativeGasUsed: 24069,
      contractAddress: null,
      logs: [Array],
      status: true,
      logsBloom:
'0x00000000408000000000000000000000000000000000000000000000000000000000000000000000000040000000000000000000024000000000000000000000000400000000000000000000008000000000000000000000000000000000000000000000000000000000000000000000000800000800000000000000000
      v: '0x1b',
      r: '0x8ac5a301b9fb66024392e89172df7d2d407d1093336ced643cd1ccabad3d4c84',
      s: '0x56dafb9de55b64f160d6829321befcb0878384b195a9c04aafc6a96bc6389b6',
      rawLogs: [Array]
    },
    logs: [ [Object] ]
  }
}
       ✓ check transfer from external (259ms)


  9 passing (2s)
```

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

`2c6bcaf67cd1eb6e293356b69d98acdbb216cc8c34b6022fbe5e2494d87a75a3 ./contracts/Migrations.sol`

`1cdf1689c814e9370b11f09e2803e3a31932cfab88c4a8ddb7f31bc676b669a7 ./contracts/DECA.sol`

Tests

`6a00df1d07fe06ba2f3b4f452c8989707f2e0a6503851b6eb8d119b7e47182cb ./test/DECA.js`

Changelog

- 2019-10-02 - Initial report
- 2019-11-08 - Revised report based on commit `c1b7d8f`
- 2019-12-02 - Revised report based on commit `1af0f37`
- 2020-03-06 - Revised report based on commit `f609015`
- 2020-06-18 - Revised report based on commit `efc8046`
- 2020-06-30 - Revised report based on commit `bcf24df`

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.