CertiK Audit Report For Vanta

Request Date: 2019-09-05
Revision Date: 2019-09-15
Platform Name: Ethereum

Contents

Disclaimer

About CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, has developed a proprietary Formal Verification technology to apply rigorous and complete mathematical reasoning against code. This process ensures algorithms, protocols, and business functionalities are secured and working as intended across all platforms.

CertiK differs from traditional testing approaches by employing Formal Verification to mathematically prove blockchain ecosystem and smart contracts are hacker-resistant and bug-free. CertiK uses this industry-leading technology together with standardized test suites, static analysis, and expert manual review to create a full-stack solution for our partners across the blockchain world to secure 6.2B in assets.

For more information: https://certik.org/

Executive Summary

This report has been prepared for Vanta to discover issues and vulnerabilities in the source code of their VantaToken smart contract. A comprehensive examination has been performed, utilizing CertiK's Formal Verification Platform, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practice and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line by line manual review of the entire codebase by industry experts.

Vulnerability Classification

CertiK categorizes issues into 3 buckets based on overall risk levels:

Critical

The code implementation does not match the specification, or it could result in the loss of funds for contract owner or users.

Medium

The code implementation does not match the specification under certain conditions, or it could affect the security standard by lost of access control.

Low

The code implementation does not follow best practices, or use suboptimal design patterns, which may lead to security vulnerabilies further down the line.

Testing Summary

PASS

CERTIK *believes this smart contract passes security qualifications to be listed on digital asset exchanges.*

Sep 14, 2019

Type of Issues

CertiK smart label engine applied 100% formal verification coverage on the source code. Our team of engineers ao scanned the source code using our proprietary static analysis tools and code-review methodologies. The following technical issues were found:

Title	Description	Issues	SWC ID
Integer Overflow and Underflow	An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum size of a type.	0	SWC-101
Function incorrectness	Function implementation does not meet the specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker is able to write to arbitrary storage locations of a contract if array of out bound happens	0	SWC-124
Reentrancy	A malicious contract can call back into the calling contract before the first invocation of the function is finished.	0	SWC-107
Transaction Order Dependence	A race condition vulnerability occurs when code depends on the order of the transactions submitted to it.	0	SWC-114
Timestamp Dependence	Timestamp can be influenced by minors to some degree.	0	SWC-116
Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0	SWC-102 SWC-103
Insecure Randomness	Block attributes are insecure to generate random numbers, as they can be influenced by minors to some degree.	0	SWC-120

"tx.origin" for authorization	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115
Delegatecall to Untrusted Callee	Calling into untrusted contracts is very dangerous, the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	Functions are public by default. A malicious user is able to make unauthorized or unintended state changes if a developer forgot to set the visibility.	0	SWC-100
Uninitialized variables	Uninitialized local storage variables can point to other unexpected storage variables in the contract.	0	SWC-109
Assertion Failure	The assert() function is meant to assert invariants. Properly functioning code should never reach a failing assert statement.	0	SWC-110
Deprecated Solidity Features	Several functions and operators in Solidity are deprecated and should not be used as best practice.	0	SWC-111
Unused variables	Unused variables reduce code quality	0	

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

No issue found.

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

- **VantaToken.sol** $0xfdf574766ba1a96a553e175aeffa85ad78063f0b$ 32e6fbd91930ff58af5e39735dc6052b4d0f696f67eb76506c0d4de0fe9319ff

Summary

CertiK was chosen by Vanta to audit the design and implementation of its new `VantaToken` smart contract. To ensure comprehensive protection,the source code has been analyzed by the proprietary CertiK formal verification engine and manually reviewed by our smart contract experts and engineers. That end-to-end process ensures proof of stability as well as a hands-on, engineering-focused process to close potential loopholes and recommend design changes in accordance with the best practices in the space.****

Overall we found the smart contracts to follow good practices. With the final update of source code and delivery of the audit report, we conclude that the contract is structurally sound and not vulnerable to any classically known anti-patterns or security issues. The audit report itself is not necessarily a guarantee of correctness or trustworthiness, and we always recommend to seek multiple opinions, keep improving the codebase, and more test coverage and sandbox deployments before the mainnet release.

Recommendations

Items in this section are low impact to the overall aspects of the smart contracts, thus will let client to decide whether to have those reflected in the final deployed version of source codes.

VantaToken.sol

- INFO `_transfer()`: Recommend checking if `to == address(0)` to prevent value loss and be consistent with `_approve()`.
- INFO `_burnFrom()`: Internal method unused.
- INFO `require()`: Recommend supplying error message.
- INFO `transferOwnership()`: Recommend using the pull model:

```
function transferOwnership(address payable _newOwner) public onlyOwner {
    newOwner = _newOwner;
}
function acceptOwnership() public {
    require(msg.sender == newOwner);
    emit OwnershipTransferred(owner, newOwner);
    owner = newOwner;
    newOwner = address(0);
}
```

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 21 in File VantaToken.sol

```
pragma solidity ^0.5.2;
```

Only these compiler versions are safe to compile your code: 0.5.10

Formal Verification Results

How to read

Detail for Request 1

transferFrom to same address

Verification date	20, Oct 2018
Verification timespan	395.38 ms
CERTIK *label location*	Line 30-34 in File howtoread.sol

CERTIK *label*

```
30  /*@CTK FAIL "transferFrom to same address"
31      @tag assume_completion
32      @pre from == to
33      @post __post.allowed[from][msg.sender] ==
34  */
```

Raw code location Line 35-41 in File howtoread.sol

Raw code

```
35  function transferFrom(address from, address to
        ) {
36      balances[from] = balances[from].sub(tokens
37      allowed[from][msg.sender] = allowed[from][
38      balances[to] = balances[to].add(tokens);
39      emit Transfer(from, to, tokens);
40      return true;
41  }
```

Counterexample This code violates the specification

Initial environment

```
 1  Counter Example:
 2  Before Execution:
 3      Input = {
 4          from = 0x0
 5          to = 0x0
 6          tokens = 0x6c
 7      }
 8      This = 0

52              }
53              balance: 0x0
54          }
55      }
56
```

Post environment

```
57  After Execution:
58      Input = {
59          from = 0x0
60          to = 0x0
61          tokens = 0x6c
```

Formal Verification Request 1

SafeMath mul zero

14, Sep 2019
14.43 ms

Line 53-58 in File VantaToken.sol

```
    /*@CTK "SafeMath mul zero"
      @tag spec
      @tag is_pure
      @pre (a == 0)
      @post __return == 0
     */
```

Line 69-81 in File VantaToken.sol

```
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b);

        return c;
    }
```

✓ The code meets the specification.

Formal Verification Request 2

SafeMath mul nonzero

14, Sep 2019
292.75 ms

Line 59-68 in File VantaToken.sol

```
    /*@CTK "SafeMath mul nonzero"
      @tag spec
      @tag is_pure
      @pre (a != 0)
      @post (a * b / a != b) == __reverted
      @post !__reverted -> __return == a * b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
     */
```

Line 69-81 in File VantaToken.sol

```
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b);

        return c;
    }
```

✓ The code meets the specification.

Formal Verification Request 3

SafeMath div

📅 14, Sep 2019
⏱ 12.07 ms

Line 86-94 in File VantaToken.sol

```
    /*@CTK "SafeMath div"
      @tag spec
      @tag is_pure
      @post (b == 0) == __reverted
      @post !__reverted -> __return == a / b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
     */
```

Line 95-102 in File VantaToken.sol

```
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        // Solidity only automatically asserts when dividing by 0
        require(b > 0);
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't hold

        return c;
    }
```

✓ The code meets the specification.

Formal Verification Request 4

SafeMath sub

📅 14, Sep 2019
⏱ 11.02 ms

Line 107-115 in File VantaToken.sol

```
    /*@CTK "SafeMath sub"
      @tag spec
```

```
     @tag is_pure
     @post (b > a) == __reverted
     @post !__reverted -> __return == a - b
     @post !__reverted -> !__has_overflow
     @post !__reverted -> !__has_assertion_failure
     @post !(__has_buf_overflow)
    */
```

Line 116-121 in File VantaToken.sol

```
   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
      require(b <= a);
      uint256 c = a - b;

      return c;
   }
```

✓ The code meets the specification.

Formal Verification Request 5

SafeMath add

14, Sep 2019
13.53 ms

Line 126-134 in File VantaToken.sol

```
   /*@CTK "SafeMath add"
     @tag spec
     @tag is_pure
     @post (a + b < a || a + b < b) == __reverted
     @post !__reverted -> __return == a + b
     @post !__reverted -> !__has_overflow
     @post !__reverted -> !__has_assertion_failure
     @post !(__has_buf_overflow)
    */
```

Line 135-140 in File VantaToken.sol

```
   function add(uint256 a, uint256 b) internal pure returns (uint256) {
      uint256 c = a + b;
      require(c >= a);

      return c;
   }
```

✓ The code meets the specification.

Formal Verification Request 6

If method completes, integer overflow would not happen.

14, Sep 2019
4.72 ms

Line 176 in File VantaToken.sol

```
//@CTK NO_OVERFLOW
```

Line 183-185 in File VantaToken.sol

```
function totalSupply() public view returns (uint256) {
    return _totalSupply;
}
```

✓ The code meets the specification.

Formal Verification Request 7

Buffer overflow / array index out of bound would never happen.

14, Sep 2019
0.33 ms

Line 177 in File VantaToken.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 183-185 in File VantaToken.sol

```
function totalSupply() public view returns (uint256) {
    return _totalSupply;
}
```

✓ The code meets the specification.

Formal Verification Request 8

Method will not encounter an assertion failure.

14, Sep 2019
0.33 ms

Line 178 in File VantaToken.sol

```
//@CTK NO_ASF
```

Line 183-185 in File VantaToken.sol

```
function totalSupply() public view returns (uint256) {
    return _totalSupply;
}
```

✓ The code meets the specification.

Formal Verification Request 9

totalSupply

14, Sep 2019
0.34 ms

Line 179-182 in File VantaToken.sol

```
  /*@CTK totalSupply
    @tag assume_completion
    @post (__return) == (_totalSupply)
   */
```

Line 183-185 in File VantaToken.sol

```
  function totalSupply() public view returns (uint256) {
      return _totalSupply;
  }
```

✓ The code meets the specification.

Formal Verification Request 10

If method completes, integer overflow would not happen.

📅 14, Sep 2019
⏱ 4.66 ms

Line 192 in File VantaToken.sol

```
  //@CTK NO_OVERFLOW
```

Line 198-200 in File VantaToken.sol

```
  function balanceOf(address owner) public view returns (uint256) {
      return _balances[owner];
  }
```

✓ The code meets the specification.

Formal Verification Request 11

Buffer overflow / array index out of bound would never happen.

📅 14, Sep 2019
⏱ 0.35 ms

Line 193 in File VantaToken.sol

```
  //@CTK NO_BUF_OVERFLOW
```

Line 198-200 in File VantaToken.sol

```
  function balanceOf(address owner) public view returns (uint256) {
      return _balances[owner];
  }
```

✓ The code meets the specification.

Formal Verification Request 12

Method will not encounter an assertion failure.

14, Sep 2019
0.3 ms

Line 194 in File VantaToken.sol

```
    //@CTK NO_ASF
```

Line 198-200 in File VantaToken.sol

```
    function balanceOf(address owner) public view returns (uint256) {
        return _balances[owner];
    }
```

✓ The code meets the specification.

Formal Verification Request 13

balanceOf

14, Sep 2019
0.33 ms

Line 195-197 in File VantaToken.sol

```
    /*@CTK balanceOf
      @post __return == __post._balances[owner]
     */
```

Line 198-200 in File VantaToken.sol

```
    function balanceOf(address owner) public view returns (uint256) {
        return _balances[owner];
    }
```

✓ The code meets the specification.

Formal Verification Request 14

If method completes, integer overflow would not happen.

14, Sep 2019
5.03 ms

Line 208 in File VantaToken.sol

```
    //@CTK NO_OVERFLOW
```

Line 214-216 in File VantaToken.sol

```
    function allowance(address owner, address spender) public view returns (uint256) {
        return _allowed[owner][spender];
    }
```

✓ The code meets the specification.

Formal Verification Request 15

Buffer overflow / array index out of bound would never happen.

14, Sep 2019
0.31 ms

Line 209 in File VantaToken.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 214-216 in File VantaToken.sol

```
    function allowance(address owner, address spender) public view returns (uint256) {
        return _allowed[owner][spender];
    }
```

✓ The code meets the specification.

Formal Verification Request 16

Method will not encounter an assertion failure.

14, Sep 2019
0.31 ms

Line 210 in File VantaToken.sol

```
    //@CTK NO_ASF
```

Line 214-216 in File VantaToken.sol

```
    function allowance(address owner, address spender) public view returns (uint256) {
        return _allowed[owner][spender];
    }
```

✓ The code meets the specification.

Formal Verification Request 17

allowance correctness

14, Sep 2019
0.47 ms

Line 211-213 in File VantaToken.sol

```
    /*@CTK "allowance correctness"
      @post __return == _allowed[owner][spender]
     */
```

Line 214-216 in File VantaToken.sol

```
    function allowance(address owner, address spender) public view returns (uint256) {
        return _allowed[owner][spender];
    }
```

✓ The code meets the specification.

Formal Verification Request 18

If method completes, integer overflow would not happen.

📅 14, Sep 2019
⏱ 60.58 ms

Line 223 in File VantaToken.sol

```
    //@CTK NO_OVERFLOW
```

Line 233-236 in File VantaToken.sol

```
    function transfer(address to, uint256 value) public returns (bool) {
        _transfer(msg.sender, to, value);
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 19

Buffer overflow / array index out of bound would never happen.

📅 14, Sep 2019
⏱ 0.69 ms

Line 224 in File VantaToken.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 233-236 in File VantaToken.sol

```
    function transfer(address to, uint256 value) public returns (bool) {
        _transfer(msg.sender, to, value);
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 20

Method will not encounter an assertion failure.

📅 14, Sep 2019
⏱ 24.28 ms

Line 225 in File VantaToken.sol

```
    //@CTK FAIL NO_ASF
```

Line 233-236 in File VantaToken.sol

```
    function transfer(address to, uint256 value) public returns (bool) {
        _transfer(msg.sender, to, value);
        return true;
    }
```

✗ This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        to = 2
        value = 241
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = false
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "ERC20",
          "balance": 0,
          "contract": {
            "_balances": [
              {
                "key": 0,
                "value": 0
              },
              {
                "key": 2,
                "value": 15
              },
              {
                "key": 8,
                "value": 0
              },
              {
                "key": 64,
                "value": 0
              },
              {
                "key": "ALL_OTHERS",
                "value": 2
              }
            ],
            "_allowed": [
              {
                "key": 0,
            "value": [
              {
                "key": 0,
                "value": 64
              },
              {
                "key": "ALL_OTHERS",
                "value": 32
              }
            ]
          },
          {
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": "ALL_OTHERS",
                "value": 2
              }
            ]
          }
        ],
        "_totalSupply": 0
      }
    }
  },
  {
    "key": "ALL_OTHERS",
    "value": "EmptyAddress"
  }
]

Function invocation is reverted.
```

Formal Verification Request 21

transfer

📅 14, Sep 2019
⏱ 22.76 ms

Line 226-232 in File VantaToken.sol

```
    /*@CTK transfer
      @tag assume_completion
      @pre msg.sender != to
      @post value <= _balances[msg.sender]
      @post __post._balances[msg.sender] == _balances[msg.sender] - value
      @post __post._balances[to] == _balances[to] + value
     */
```

Line 233-236 in File VantaToken.sol

```
    function transfer(address to, uint256 value) public returns (bool) {
        _transfer(msg.sender, to, value);
        return true;
    }
```

✅ The code meets the specification.

Formal Verification Request 22

If method completes, integer overflow would not happen.

14, Sep 2019
52.2 ms

Line 247 in File VantaToken.sol

```
247     //@CTK NO_OVERFLOW
```

Line 254-257 in File VantaToken.sol

```
254     function approve(address spender, uint256 value) public returns (bool) {
255         _approve(msg.sender, spender, value);
256         return true;
257     }
```

✓ The code meets the specification.

Formal Verification Request 23

Buffer overflow / array index out of bound would never happen.

14, Sep 2019
0.59 ms

Line 248 in File VantaToken.sol

```
248     //@CTK NO_BUF_OVERFLOW
```

Line 254-257 in File VantaToken.sol

```
254     function approve(address spender, uint256 value) public returns (bool) {
255         _approve(msg.sender, spender, value);
256         return true;
257     }
```

✓ The code meets the specification.

Formal Verification Request 24

Method will not encounter an assertion failure.

14, Sep 2019
0.8 ms

Line 249 in File VantaToken.sol

```
249     //@CTK NO_ASF
```

Line 254-257 in File VantaToken.sol

```
254     function approve(address spender, uint256 value) public returns (bool) {
255         _approve(msg.sender, spender, value);
256         return true;
257     }
```

✓ The code meets the specification.

Formal Verification Request 25

approve

14, Sep 2019
4.97 ms

Line 250-253 in File VantaToken.sol

```
    /*@CTK approve
      @tag assume_completion
      @post (__post._allowed[msg.sender][spender]) == (value)
     */
```

Line 254-257 in File VantaToken.sol

```
    function approve(address spender, uint256 value) public returns (bool) {
        _approve(msg.sender, spender, value);
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 26

If method completes, integer overflow would not happen.

14, Sep 2019
75.23 ms

Line 267 in File VantaToken.sol

```
    //@CTK NO_OVERFLOW
```

Line 278-282 in File VantaToken.sol

```
    function transferFrom(address from, address to, uint256 value) public returns (
        bool) {
        _transfer(from, to, value);
        _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 27

Buffer overflow / array index out of bound would never happen.

14, Sep 2019
9.2 ms

Line 268 in File VantaToken.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 278-282 in File VantaToken.sol

```
278    function transferFrom(address from, address to, uint256 value) public returns (
           bool) {
279        _transfer(from, to, value);
280        _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));
281        return true;
282    }
```

✓ The code meets the specification.

Formal Verification Request 28

Method will not encounter an assertion failure.

📅 14, Sep 2019
⏱ 88.7 ms

Line 269 in File VantaToken.sol

```
269    //@CTK FAIL NO_ASF
```

Line 278-282 in File VantaToken.sol

```
278    function transferFrom(address from, address to, uint256 value) public returns (
           bool) {
279        _transfer(from, to, value);
280        _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));
281        return true;
282    }
```

✗ This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        from = 0
        to = 16
        value = 63
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = false
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
```

```
  Address_Map = [
    {
      "key": 0,
      "value": {
        "contract_name": "ERC20",
        "balance": 0,
        "contract": {
          "_balances": [
            {
              "key": 128,
              "value": 0
            },
            {
              "key": 0,
              "value": 0
            },
            {
              "key": 9,
              "value": 1
            },
            {
              "key": 4,
              "value": 0
            },
            {
              "key": 64,
              "value": 0
            },
            {
              "key": 8,
              "value": 0
            },
            {
              "key": 32,
              "value": 0
            },
            {
              "key": 1,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 16
            }
          ],
          "_allowed": [
            {
              "key": 0,
              "value": [
                {
                  "key": 0,
                  "value": 40
                },
                {
                  "key": 4,
                  "value": 4
                },
                {
```

```
                "key": 1,
                "value": 64
              },
              {
                "key": "ALL_OTHERS",
                "value": 16
              }
            ]
          },
          {
            "key": 32,
            "value": [
              {
                "key": 0,
                "value": 16
              },
              {
                "key": "ALL_OTHERS",
                "value": 64
              }
            ]
          },
          {
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": "ALL_OTHERS",
                "value": 16
              }
            ]
          }
        ],
        "_totalSupply": 0
      }
    }
  },
  {
    "key": "ALL_OTHERS",
    "value": "EmptyAddress"
  }
]

Function invocation is reverted.
```

Formal Verification Request 29

transferFrom correctness

📅 14, Sep 2019
⏱ 312.51 ms

Line 270-277 in File VantaToken.sol

```
  /*@CTK "transferFrom correctness"
    @tag assume_completion
    @post value <= _balances[from] && value <= _allowed[from][msg.sender]
    @post to != from -> __post._balances[from] == _balances[from] - value
    @post to != from -> __post._balances[to] == _balances[to] + value
    @post to == from -> __post._balances[from] == _balances[from]
    @post __post._allowed[from][msg.sender] == _allowed[from][msg.sender] - value
   */
```

Line 278-282 in File VantaToken.sol

```
  function transferFrom(address from, address to, uint256 value) public returns (
      bool) {
      _transfer(from, to, value);
      _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));
      return true;
  }
```

✓ The code meets the specification.

Formal Verification Request 30

If method completes, integer overflow would not happen.

14, Sep 2019
33.63 ms

Line 294 in File VantaToken.sol

```
  //@CTK NO_OVERFLOW
```

Line 301-304 in File VantaToken.sol

```
  function increaseAllowance(address spender, uint256 addedValue) public returns (
      bool) {
      _approve(msg.sender, spender, _allowed[msg.sender][spender].add(addedValue));
      return true;
  }
```

✓ The code meets the specification.

Formal Verification Request 31

Buffer overflow / array index out of bound would never happen.

14, Sep 2019
0.6 ms

Line 295 in File VantaToken.sol

```
  //@CTK NO_BUF_OVERFLOW
```

Line 301-304 in File VantaToken.sol

```
  function increaseAllowance(address spender, uint256 addedValue) public returns (
      bool) {
      _approve(msg.sender, spender, _allowed[msg.sender][spender].add(addedValue));
      return true;
  }
```

✓ The code meets the specification.

Formal Verification Request 32

Method will not encounter an assertion failure.

14, Sep 2019
14.85 ms

Line 296 in File VantaToken.sol

```
    //@CTK FAIL NO_ASF
```

Line 301-304 in File VantaToken.sol

```
    function increaseAllowance(address spender, uint256 addedValue) public returns (
        bool) {
        _approve(msg.sender, spender, _allowed[msg.sender][spender].add(addedValue));
        return true;
    }
```

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        addedValue = 160
        spender = 8
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 128,
          "value": 0
        }
    }
    Other = {
        __return = false
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "ERC20",
          "balance": 0,
          "contract": {
            "_balances": [
              {
                "key": 8,
                "value": 16
              },
              {
                "key": "ALL_OTHERS",
                "value": 72
              }
            ],
            "_allowed": [
              {
                "key": 128,
                "value": [
                  {
                    "key": 8,
                    "value": 168
                  },
                  {
                    "key": "ALL_OTHERS",
                    "value": 160
                  }
                ]
              },
              {
                "key": "ALL_OTHERS",
                "value": [
                  {
                    "key": "ALL_OTHERS",
                    "value": 72
                  }
                ]
              }
            ],
            "_totalSupply": 0
          }
        }
      },
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

Function invocation is reverted.
```

Formal Verification Request 33

increaseApproval correctness

📅 14, Sep 2019
⏱ 4.72 ms

Line 297-300 in File VantaToken.sol

```
    /*@CTK "increaseApproval correctness"
      @tag assume_completion
      @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] +
          addedValue
     */
```

Line 301-304 in File VantaToken.sol

```
301     function increaseAllowance(address spender, uint256 addedValue) public returns (
            bool) {
302         _approve(msg.sender, spender, _allowed[msg.sender][spender].add(addedValue));
303         return true;
304     }
```

✓ The code meets the specification.

Formal Verification Request 34

If method completes, integer overflow would not happen.

14, Sep 2019
40.35 ms

Line 316 in File VantaToken.sol

```
316     //@CTK NO_OVERFLOW
```

Line 330-333 in File VantaToken.sol

```
330     function decreaseAllowance(address spender, uint256 subtractedValue) public
            returns (bool) {
331         _approve(msg.sender, spender, _allowed[msg.sender][spender].sub(subtractedValue
                ));
332         return true;
333     }
```

✓ The code meets the specification.

Formal Verification Request 35

Buffer overflow / array index out of bound would never happen.

14, Sep 2019
0.8 ms

Line 317 in File VantaToken.sol

```
317     //@CTK NO_BUF_OVERFLOW
```

Line 330-333 in File VantaToken.sol

```
330     function decreaseAllowance(address spender, uint256 subtractedValue) public
            returns (bool) {
331         _approve(msg.sender, spender, _allowed[msg.sender][spender].sub(subtractedValue
                ));
332         return true;
333     }
```

✓ The code meets the specification.

Formal Verification Request 36

Method will not encounter an assertion failure.

📅 14, Sep 2019
⏱ 14.74 ms

Line 318 in File VantaToken.sol

```
318    //@CTK FAIL NO_ASF
```

Line 330-333 in File VantaToken.sol

```
330    function decreaseAllowance(address spender, uint256 subtractedValue) public
           returns (bool) {
331        _approve(msg.sender, spender, _allowed[msg.sender][spender].sub(subtractedValue
               ));
332        return true;
333    }
```

❌ This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        spender = 8
        subtractedValue = 219
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 64,
          "value": 0
        }
    }
    Other = {
        __return = false
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "ERC20",
          "balance": 0,
          "contract": {
            "_balances": [
              {
                "key": 0,
                "value": 8
              },
            {
              "key": "ALL_OTHERS",
              "value": 219
            }
          ],
          "_allowed": [
            {
              "key": 64,
              "value": [
                {
                  "key": 0,
                  "value": 0
                },
                {
                  "key": 8,
                  "value": 0
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 219
                }
              ]
            },
            {
              "key": 0,
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 219
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 219
                }
              ]
            }
          ],
          "_totalSupply": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

Function invocation is reverted.
```

Formal Verification Request 37

decreaseApproval0

📅 14, Sep 2019
⏱ 21.82 ms

Line 319-323 in File VantaToken.sol

```
    /*@CTK decreaseApproval0
      @pre __return == true
      @pre _allowed[msg.sender][spender] <= subtractedValue
      @post __post._allowed[msg.sender][spender] == 0
     */
```

Line 330-333 in File VantaToken.sol

```
    function decreaseAllowance(address spender, uint256 subtractedValue) public
        returns (bool) {
        _approve(msg.sender, spender, _allowed[msg.sender][spender].sub(subtractedValue
            ));
        return true;
    }
```

✅ The code meets the specification.

Formal Verification Request 38

decreaseApproval

📅 14, Sep 2019
⏱ 4.88 ms

Line 324-329 in File VantaToken.sol

```
    /*@CTK decreaseApproval
      @pre __return == true
      @pre _allowed[msg.sender][spender] > subtractedValue
      @post __post._allowed[msg.sender][spender] ==
        _allowed[msg.sender][spender] - subtractedValue
     */
```

Line 330-333 in File VantaToken.sol

```
    function decreaseAllowance(address spender, uint256 subtractedValue) public
        returns (bool) {
        _approve(msg.sender, spender, _allowed[msg.sender][spender].sub(subtractedValue
            ));
        return true;
    }
```

✅ The code meets the specification.

Formal Verification Request 39

If method completes, integer overflow would not happen.

📅 14, Sep 2019
⏱ 2.24 ms

Line 341 in File VantaToken.sol

```
    //@CTK NO_OVERFLOW
```

Line 351-356 in File VantaToken.sol

```
    function _transfer(address from, address to, uint256 value) internal {

        _balances[from] = _balances[from].sub(value);
        _balances[to] = _balances[to].add(value);
        emit Transfer(from, to, value);
    }
```

✓ The code meets the specification.

Formal Verification Request 40

Buffer overflow / array index out of bound would never happen.

14, Sep 2019
0.54 ms

Line 342 in File VantaToken.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 351-356 in File VantaToken.sol

```
    function _transfer(address from, address to, uint256 value) internal {

        _balances[from] = _balances[from].sub(value);
        _balances[to] = _balances[to].add(value);
        emit Transfer(from, to, value);
    }
```

✓ The code meets the specification.

Formal Verification Request 41

Method will not encounter an assertion failure.

14, Sep 2019
27.66 ms

Line 343 in File VantaToken.sol

```
    //@CTK FAIL NO_ASF
```

Line 351-356 in File VantaToken.sol

```
    function _transfer(address from, address to, uint256 value) internal {

        _balances[from] = _balances[from].sub(value);
        _balances[to] = _balances[to].add(value);
        emit Transfer(from, to, value);
    }
```

✗ This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        from = 1
        to = 0
        value = 227
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "ERC20",
          "balance": 0,
          "contract": {
            "_balances": [
              {
                "key": 32,
                "value": 0
              },
              {
                "key": 8,
                "value": 0
              },
              {
                "key": 0,
                "value": 157
              },
              {
                "key": 129,
                "value": 0
              },
              {
                "key": 128,
                "value": 0
              },
              {
                "key": 1,
                "value": 0
              },
            {
              "key": "ALL_OTHERS",
              "value": 227
            }
          ],
          "_allowed": [
            {
              "key": 0,
              "value": [
                {
                  "key": 0,
                  "value": 0
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 64
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 227
                }
              ]
            }
          ],
          "_totalSupply": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

Function invocation is reverted.
```

Formal Verification Request 42

__transfer

14, Sep 2019
25.0 ms

Line 344-350 in File VantaToken.sol

```
    /*@CTK _transfer
      @tag assume_completion
      @pre from != to
      @post value <= _balances[from]
      @post __post._balances[from] == _balances[from] - value
      @post __post._balances[to] == _balances[to] + value
     */
```

Line 351-356 in File VantaToken.sol

```
function _transfer(address from, address to, uint256 value) internal {

    _balances[from] = _balances[from].sub(value);
    _balances[to] = _balances[to].add(value);
    emit Transfer(from, to, value);
}
```

✓ The code meets the specification.

Formal Verification Request 43

If method completes, integer overflow would not happen.

14, Sep 2019
32.92 ms

Line 365 in File VantaToken.sol

```
//@CTK NO_OVERFLOW
```

Line 374-380 in File VantaToken.sol

```
function _mint(address account, uint256 value) internal {
    require(account != address(0));

    _totalSupply = _totalSupply.add(value);
    _balances[account] = _balances[account].add(value);
    emit Transfer(address(0), account, value);
}
```

✓ The code meets the specification.

Formal Verification Request 44

Buffer overflow / array index out of bound would never happen.

14, Sep 2019
6.19 ms

Line 366 in File VantaToken.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 374-380 in File VantaToken.sol

```
function _mint(address account, uint256 value) internal {
    require(account != address(0));

    _totalSupply = _totalSupply.add(value);
    _balances[account] = _balances[account].add(value);
    emit Transfer(address(0), account, value);
}
```

✓ The code meets the specification.

Formal Verification Request 45

Method will not encounter an assertion failure.

📅 14, Sep 2019
⏱ 24.9 ms

Line 367 in File VantaToken.sol

```
367    //@CTK FAIL NO_ASF
```

Line 374-380 in File VantaToken.sol

```
374    function _mint(address account, uint256 value) internal {
375       require(account != address(0));
376
377       _totalSupply = _totalSupply.add(value);
378       _balances[account] = _balances[account].add(value);
379       emit Transfer(address(0), account, value);
380    }
```

❌ This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        account = 1
        value = 111
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "ERC20",
          "balance": 0,
          "contract": {
            "_balances": [
              {
                "key": 64,
                "value": 2
              },
              {
                "key": 2,
                "value": 0
              },
              {
                "key": 0,
                "value": 32
              },
              {
                "key": 32,
                "value": 4
              },
              {
                "key": "ALL_OTHERS",
                "value": 1
              }
            ],
            "_allowed": [
              {
                "key": 0,
                "value": [
                  {
                    "key": 0,
                    "value": 128
                  },
                  {
                    "key": "ALL_OTHERS",
                    "value": 112
                  }
                ]
              },
              {
                "key": "ALL_OTHERS",
                "value": [
                  {
                    "key": "ALL_OTHERS",
                    "value": 1
                  }
                ]
              }
            ],
            "_totalSupply": 193
          }
        }
      },
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

Function invocation is reverted.
```

Formal Verification Request 46

mint

📅 14, Sep 2019
⏱ 24.51 ms

Line 368-373 in File VantaToken.sol

```
    /*@CTK mint
      @tag assume_completion
      @post account != address(0)
      @post (__post._totalSupply) == (_totalSupply + value)
      @post (__post._balances[account]) == (_balances[account] + value)
     */
```

Line 374-380 in File VantaToken.sol

```
    function _mint(address account, uint256 value) internal {
        require(account != address(0));

        _totalSupply = _totalSupply.add(value);
        _balances[account] = _balances[account].add(value);
        emit Transfer(address(0), account, value);
    }
```

✓ The code meets the specification.

Formal Verification Request 47

If method completes, integer overflow would not happen.

📅 14, Sep 2019
⏱ 29.84 ms

Line 389 in File VantaToken.sol

```
    //@CTK NO_OVERFLOW
```

Line 399-405 in File VantaToken.sol

```
    function _burn(address account, uint256 value) internal {
        require(account != address(0));

        _totalSupply = _totalSupply.sub(value);
        _balances[account] = _balances[account].sub(value);
        emit Transfer(account, address(0), value);
    }
```

✓ The code meets the specification.

Formal Verification Request 48

Buffer overflow / array index out of bound would never happen.

📅 14, Sep 2019
⏱ 5.55 ms

Line 390 in File VantaToken.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 399-405 in File VantaToken.sol

```
    function _burn(address account, uint256 value) internal {
        require(account != address(0));

        _totalSupply = _totalSupply.sub(value);
        _balances[account] = _balances[account].sub(value);
        emit Transfer(account, address(0), value);
    }
```

✓ The code meets the specification.

Formal Verification Request 49

Method will not encounter an assertion failure.

14, Sep 2019
18.11 ms

Line 391 in File VantaToken.sol

```
    //@CTK FAIL NO_ASF
```

Line 399-405 in File VantaToken.sol

```
    function _burn(address account, uint256 value) internal {
        require(account != address(0));

        _totalSupply = _totalSupply.sub(value);
        _balances[account] = _balances[account].sub(value);
        emit Transfer(account, address(0), value);
    }
```

✗ This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        account = 8
        value = 16
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
      block = {
        "number": 0,
        "timestamp": 0
      }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "ERC20",
          "balance": 0,
          "contract": {
            "_balances": [
              {
                "key": 0,
                "value": 0
              },
              {
                "key": 8,
                "value": 0
              },
              {
                "key": "ALL_OTHERS",
                "value": 240
              }
            ],
            "_allowed": [
              {
                "key": "ALL_OTHERS",
                "value": [
                  {
                    "key": "ALL_OTHERS",
                    "value": 240
                  }
                ]
              }
            ],
            "_totalSupply": 128
          }
        }
      },
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

Function invocation is reverted.
```

Formal Verification Request 50

burn

📅 14, Sep 2019
⏱ 35.87 ms

Line 392-398 in File VantaToken.sol

```
/*@CTK burn
  @tag assume_completion
  @post account != address(0)
  @post (value <= _balances[account])
  @post (__post._totalSupply) == (_totalSupply - value)
  @post (__post._balances[account]) == (_balances[account] - value)
 */
```

Line 399-405 in File VantaToken.sol

```
function _burn(address account, uint256 value) internal {
   require(account != address(0));

   _totalSupply = _totalSupply.sub(value);
   _balances[account] = _balances[account].sub(value);
   emit Transfer(account, address(0), value);
}
```

✓ The code meets the specification.

Formal Verification Request 51

If method completes, integer overflow would not happen.

14, Sep 2019
0.52 ms

Line 414 in File VantaToken.sol

```
//@CTK NO_OVERFLOW
```

Line 423-429 in File VantaToken.sol

```
function _approve(address owner, address spender, uint256 value) internal {
   require(spender != address(0));
   require(owner != address(0));

   _allowed[owner][spender] = value;
   emit Approval(owner, spender, value);
}
```

✓ The code meets the specification.

Formal Verification Request 52

Buffer overflow / array index out of bound would never happen.

14, Sep 2019
0.45 ms

Line 415 in File VantaToken.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 423-429 in File VantaToken.sol

```
    function _approve(address owner, address spender, uint256 value) internal {
        require(spender != address(0));
        require(owner != address(0));

        _allowed[owner][spender] = value;
        emit Approval(owner, spender, value);
    }
```

✓ The code meets the specification.

Formal Verification Request 53

Method will not encounter an assertion failure.

14, Sep 2019
0.48 ms

Line 416 in File VantaToken.sol

```
    //@CTK NO_ASF
```

Line 423-429 in File VantaToken.sol

```
    function _approve(address owner, address spender, uint256 value) internal {
        require(spender != address(0));
        require(owner != address(0));

        _allowed[owner][spender] = value;
        emit Approval(owner, spender, value);
    }
```

✓ The code meets the specification.

Formal Verification Request 54

_approve

14, Sep 2019
2.62 ms

Line 417-422 in File VantaToken.sol

```
    /*@CTK _approve
      @tag assume_completion
      @post spender != address(0)
      @post owner != address(0)
      @post __post._allowed[owner][spender] == value
     */
```

Line 423-429 in File VantaToken.sol

```
    function _approve(address owner, address spender, uint256 value) internal {
        require(spender != address(0));
        require(owner != address(0));

        _allowed[owner][spender] = value;
        emit Approval(owner, spender, value);
    }
```

✓ The code meets the specification.

Formal Verification Request 55

If method completes, integer overflow would not happen.

14, Sep 2019
68.39 ms

Line 439 in File VantaToken.sol

```
    //@CTK NO_OVERFLOW
```

Line 450-453 in File VantaToken.sol

```
    function _burnFrom(address account, uint256 value) internal {
        _burn(account, value);
        _approve(account, msg.sender, _allowed[account][msg.sender].sub(value));
    }
```

✓ The code meets the specification.

Formal Verification Request 56

Buffer overflow / array index out of bound would never happen.

14, Sep 2019
9.4 ms

Line 440 in File VantaToken.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 450-453 in File VantaToken.sol

```
    function _burnFrom(address account, uint256 value) internal {
        _burn(account, value);
        _approve(account, msg.sender, _allowed[account][msg.sender].sub(value));
    }
```

✓ The code meets the specification.

Formal Verification Request 57

Method will not encounter an assertion failure.

14, Sep 2019
60.69 ms

Line 441 in File VantaToken.sol

```
    //@CTK FAIL NO_ASF
```

Line 450-453 in File VantaToken.sol

```
    function _burnFrom(address account, uint256 value) internal {
        _burn(account, value);
        _approve(account, msg.sender, _allowed[account][msg.sender].sub(value));
    }
```

✖ This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        account = 64
        value = 224
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "ERC20",
          "balance": 0,
          "contract": {
            "_balances": [
              {
                "key": 8,
                "value": 0
              },
              {
                "key": 0,
                "value": 0
              },
              {
                "key": 1,
                "value": 0
              },
              {
                "key": 64,
                "value": 0
              },
              {
                "key": "ALL_OTHERS",
                "value": 32
              }
            ],
            "_allowed": [
              {
                "key": 0,
                "value": [
                  {
                    "key": 0,
                    "value": 0
                  },
                  {
                    "key": "ALL_OTHERS",
                    "value": 32
                  }
                ]
              },
              {
                "key": "ALL_OTHERS",
                "value": [
                  {
                    "key": "ALL_OTHERS",
                    "value": 32
                  }
                ]
              }
            ],
            "_totalSupply": 224
          }
        }
      },
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

  Function invocation is reverted.
```

Formal Verification Request 58

burnFrom

📅 14, Sep 2019
⏱ 129.4 ms

Line 442-449 in File VantaToken.sol

```
    /*@CTK burnFrom
      @tag assume_completion
      @post (value <= _allowed[account][msg.sender])
      @post (value <= _balances[account])
      @post (__post._allowed[account][msg.sender]) == (_allowed[account][msg.sender] -
           value)
      @post (__post._balances[account]) == (_balances[account] - (value))
      @post __post._totalSupply == (_totalSupply - value)
     */
```

Line 450-453 in File VantaToken.sol

```
function _burnFrom(address account, uint256 value) internal {
    _burn(account, value);
    _approve(account, msg.sender, _allowed[account][msg.sender].sub(value));
}
```

✓ The code meets the specification.

Formal Verification Request 59

Ownable

📅 14, Sep 2019
⏱ 4.74 ms

Line 509-512 in File VantaToken.sol

```
/*@CTK Ownable
  @tag assume_completion
  @post __post._owner == msg.sender
 */
```

Line 513-516 in File VantaToken.sol

```
constructor () internal {
    _owner = msg.sender;
    emit OwnershipTransferred(address(0), _owner);
}
```

✓ The code meets the specification.

Formal Verification Request 60

isOwner

📅 14, Sep 2019
⏱ 4.81 ms

Line 536-539 in File VantaToken.sol

```
/*@CTK isOwner
   @tag assume_completion
   @post __return == (msg.sender == _owner)
 */
```

Line 540-542 in File VantaToken.sol

```
function isOwner() public view returns (bool) {
    return msg.sender == _owner;
}
```

✓ The code meets the specification.

Formal Verification Request 61

renounceOwnership

📅 14, Sep 2019
⏱ 20.2 ms

Line 551-555 in File VantaToken.sol

```
    /*@CTK renounceOwnership
      @tag assume_completion
      @post msg.sender == _owner
      @post __post._owner == address(0)
     */
```

Line 556-559 in File VantaToken.sol

```
    function renounceOwnership() public onlyOwner {
        emit OwnershipTransferred(_owner, address(0));
        _owner = address(0);
    }
```

✓ The code meets the specification.

Formal Verification Request 62

transferOwnership

📅 14, Sep 2019
⏱ 52.88 ms

Line 565-570 in File VantaToken.sol

```
    /*@CTK transferOwnership
      @tag assume_completion
      @pre msg.sender == _owner
      @pre newOwner != address(0)
      @post __post._owner == newOwner
     */
```

Line 571-573 in File VantaToken.sol

```
    function transferOwnership(address newOwner) public onlyOwner {
        _transferOwnership(newOwner);
    }
```

✓ The code meets the specification.

Formal Verification Request 63

_transferOwnership

📅 14, Sep 2019
⏱ 1.37 ms

Line 579-584 in File VantaToken.sol

```
/*@CTK _transferOwnership
  @tag assume_completion
  @pre msg.sender == _owner
  @pre newOwner != address(0)
  @post __post._owner == newOwner
 */
```

Line 585-589 in File VantaToken.sol

```
function _transferOwnership(address newOwner) internal {
   require(newOwner != address(0));
   emit OwnershipTransferred(_owner, newOwner);
   _owner = newOwner;
}
```

✓ The code meets the specification.

Source Code with CertiK Labels

File VantaToken.sol

```
/**
 *Submitted for verification at Etherscan.io on 2019-08-16
*/

/**
 * Copyright 2019 Vanta Network.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *    http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

pragma solidity ^0.5.2;

/**
 * @title ERC20 interface
 * @dev see https://eips.ethereum.org/EIPS/eip-20
 */
interface IERC20 {
    function transfer(address to, uint256 value) external returns (bool);

    function approve(address spender, uint256 value) external returns (bool);

    function transferFrom(address from, address to, uint256 value) external returns (
        bool);

    function totalSupply() external view returns (uint256);

    function balanceOf(address who) external view returns (uint256);

    function allowance(address owner, address spender) external view returns (uint256)
        ;

    event Transfer(address indexed from, address indexed to, uint256 value);

    event Approval(address indexed owner, address indexed spender, uint256 value);
}

/**
 * @title SafeMath
 * @dev Unsigned math operations with safety checks that revert on error
 */
library SafeMath {
    /**
     * @dev Multiplies two unsigned integers, reverts on overflow.
     */
```

```
/*@CTK "SafeMath mul zero"
  @tag spec
  @tag is_pure
  @pre (a == 0)
  @post __return == 0
 */
/*@CTK "SafeMath mul nonzero"
  @tag spec
  @tag is_pure
  @pre (a != 0)
  @post (a * b / a != b) == __reverted
  @post !__reverted -> __return == a * b
  @post !__reverted -> !__has_overflow
  @post !__reverted -> !__has_assertion_failure
  @post !(__has_buf_overflow)
 */
function mul(uint256 a, uint256 b) internal pure returns (uint256) {
    // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
    // benefit is lost if 'b' is also tested.
    // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
    if (a == 0) {
        return 0;
    }

    uint256 c = a * b;
    require(c / a == b);

    return c;
}

/**
 * @dev Integer division of two unsigned integers truncating the quotient, reverts
      on division by zero.
 */
/*@CTK "SafeMath div"
  @tag spec
  @tag is_pure
  @post (b == 0) == __reverted
  @post !__reverted -> __return == a / b
  @post !__reverted -> !__has_overflow
  @post !__reverted -> !__has_assertion_failure
  @post !(__has_buf_overflow)
 */
function div(uint256 a, uint256 b) internal pure returns (uint256) {
    // Solidity only automatically asserts when dividing by 0
    require(b > 0);
    uint256 c = a / b;
    // assert(a == b * c + a % b); // There is no case in which this doesn't hold

    return c;
}

/**
 * @dev Subtracts two unsigned integers, reverts on overflow (i.e. if subtrahend
     is greater than minuend).
 */
/*@CTK "SafeMath sub"
  @tag spec
```

```
      @tag is_pure
      @post (b > a) == __reverted
      @post !__reverted -> __return == a - b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
     */
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b <= a);
        uint256 c = a - b;

        return c;
    }

    /**
     * @dev Adds two unsigned integers, reverts on overflow.
     */
    /*@CTK "SafeMath add"
      @tag spec
      @tag is_pure
      @post (a + b < a || a + b < b) == __reverted
      @post !__reverted -> __return == a + b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
     */
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a);

        return c;
    }

    /**
     * @dev Divides two unsigned integers and returns the remainder (unsigned integer
         modulo),
     * reverts when dividing by zero.
     */
    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b != 0);
        return a % b;
    }
}

/**
 * @title Standard ERC20 token
 *
 * @dev Implementation of the basic standard token.
 * https://eips.ethereum.org/EIPS/eip-20
 * Originally based on code by FirstBlood:
 * https://github.com/Firstbloodio/token/blob/master/smart_contract/FirstBloodToken.
    sol
 *
 * This implementation emits additional Approval events, allowing applications to
    reconstruct the allowance status for
 * all accounts just by listening to said events. Note that this isn't required by the
    specification, and other
 * compliant implementations may not do it.
 */
contract ERC20 is IERC20 {
    using SafeMath for uint256;

    mapping (address => uint256) private _balances;

    mapping (address => mapping (address => uint256)) private _allowed;

    uint256 private _totalSupply;

    /**
     * @dev Total number of tokens in existence
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK totalSupply
       @tag assume_completion
       @post (__return) == (_totalSupply)
      */
    function totalSupply() public view returns (uint256) {
        return _totalSupply;
    }

    /**
     * @dev Gets the balance of the specified address.
     * @param owner The address to query the balance of.
     * @return A uint256 representing the amount owned by the passed address.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK balanceOf
      @post __return == __post._balances[owner]
     */
    function balanceOf(address owner) public view returns (uint256) {
        return _balances[owner];
    }

    /**
     * @dev Function to check the amount of tokens that an owner allowed to a spender.
     * @param owner address The address which owns the funds.
     * @param spender address The address which will spend the funds.
     * @return A uint256 specifying the amount of tokens still available for the
         spender.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "allowance correctness"
       @post __return == _allowed[owner][spender]
     */
    function allowance(address owner, address spender) public view returns (uint256) {
        return _allowed[owner][spender];
    }

    /**
     * @dev Transfer token to a specified address
     * @param to The address to transfer to.
     * @param value The amount to be transferred.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK FAIL NO_ASF
    /*@CTK transfer
      @tag assume_completion
      @pre msg.sender != to
      @post value <= _balances[msg.sender]
      @post __post._balances[msg.sender] == _balances[msg.sender] - value
      @post __post._balances[to] == _balances[to] + value
     */
    function transfer(address to, uint256 value) public returns (bool) {
        _transfer(msg.sender, to, value);
        return true;
    }

    /**
     * @dev Approve the passed address to spend the specified amount of tokens on
         behalf of msg.sender.
     * Beware that changing an allowance with this method brings the risk that someone
          may use both the old
     * and the new allowance by unfortunate transaction ordering. One possible
         solution to mitigate this
     * race condition is to first reduce the spender's allowance to 0 and set the
         desired value afterwards:
     * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
     * @param spender The address which will spend the funds.
     * @param value The amount of tokens to be spent.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK approve
      @tag assume_completion
      @post (__post._allowed[msg.sender][spender]) == (value)
     */
    function approve(address spender, uint256 value) public returns (bool) {
        _approve(msg.sender, spender, value);
        return true;
    }

    /**
     * @dev Transfer tokens from one address to another.
     * Note that while this function emits an Approval event, this is not required as
         per the specification,
     * and other compliant implementations may not emit the event.
     * @param from address The address which you want to send tokens from
     * @param to address The address which you want to transfer to
     * @param value uint256 the amount of tokens to be transferred
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK FAIL NO_ASF
    /*@CTK "transferFrom correctness"
      @tag assume_completion
      @post value <= _balances[from] && value <= _allowed[from][msg.sender]
      @post to != from -> __post._balances[from] == _balances[from] - value
      @post to != from -> __post._balances[to] == _balances[to] + value
      @post to == from -> __post._balances[from] == _balances[from]
      @post __post._allowed[from][msg.sender] == _allowed[from][msg.sender] - value
     */
    function transferFrom(address from, address to, uint256 value) public returns (
        bool) {
        _transfer(from, to, value);
        _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));
        return true;
    }

    /**
     * @dev Increase the amount of tokens that an owner allowed to a spender.
     * approve should be called when _allowed[msg.sender][spender] == 0. To increment
     * allowed value is better to use this function to avoid 2 calls (and wait until
     * the first transaction is mined)
     * From MonolithDAO Token.sol
     * Emits an Approval event.
     * @param spender The address which will spend the funds.
     * @param addedValue The amount of tokens to increase the allowance by.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK FAIL NO_ASF
    /*@CTK "increaseApproval correctness"
      @tag assume_completion
      @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] +
          addedValue
     */
    function increaseAllowance(address spender, uint256 addedValue) public returns (
        bool) {
        _approve(msg.sender, spender, _allowed[msg.sender][spender].add(addedValue));
        return true;
    }

    /**
     * @dev Decrease the amount of tokens that an owner allowed to a spender.
     * approve should be called when _allowed[msg.sender][spender] == 0. To decrement
     * allowed value is better to use this function to avoid 2 calls (and wait until
     * the first transaction is mined)
     * From MonolithDAO Token.sol
     * Emits an Approval event.
     * @param spender The address which will spend the funds.
     * @param subtractedValue The amount of tokens to decrease the allowance by.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK FAIL NO_ASF
    /*@CTK decreaseApproval0
      @pre __return == true
      @pre _allowed[msg.sender][spender] <= subtractedValue
      @post __post._allowed[msg.sender][spender] == 0
     */
    /*@CTK decreaseApproval
      @pre __return == true
      @pre _allowed[msg.sender][spender] > subtractedValue
      @post __post._allowed[msg.sender][spender] ==
        _allowed[msg.sender][spender] - subtractedValue
     */
    function decreaseAllowance(address spender, uint256 subtractedValue) public
        returns (bool) {
        _approve(msg.sender, spender, _allowed[msg.sender][spender].sub(subtractedValue
            ));
        return true;
    }

    /**
     * @dev Transfer token for a specified addresses
     * @param from The address to transfer from.
     * @param to The address to transfer to.
     * @param value The amount to be transferred.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK FAIL NO_ASF
    /*@CTK _transfer
       @tag assume_completion
       @pre from != to
       @post value <= _balances[from]
       @post __post._balances[from] == _balances[from] - value
       @post __post._balances[to] == _balances[to] + value
      */
    function _transfer(address from, address to, uint256 value) internal {

        _balances[from] = _balances[from].sub(value);
        _balances[to] = _balances[to].add(value);
        emit Transfer(from, to, value);
    }

    /**
     * @dev Internal function that mints an amount of the token and assigns it to
     * an account. This encapsulates the modification of balances such that the
     * proper events are emitted.
     * @param account The account that will receive the created tokens.
     * @param value The amount that will be created.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK FAIL NO_ASF
    /*@CTK mint
       @tag assume_completion
       @post account != address(0)
       @post (__post._totalSupply) == (_totalSupply + value)
       @post (__post._balances[account]) == (_balances[account] + value)
     */
    function _mint(address account, uint256 value) internal {
        require(account != address(0));

        _totalSupply = _totalSupply.add(value);
        _balances[account] = _balances[account].add(value);
        emit Transfer(address(0), account, value);
    }

    /**
     * @dev Internal function that burns an amount of the token of a given
     * account.
     * @param account The account whose tokens will be burnt.
     * @param value The amount that will be burnt.
     */

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK FAIL NO_ASF
    /*@CTK burn
      @tag assume_completion
      @post account != address(0)
      @post (value <= _balances[account])
      @post (__post._totalSupply) == (_totalSupply - value)
      @post (__post._balances[account]) == (_balances[account] - value)
     */
    function _burn(address account, uint256 value) internal {
        require(account != address(0));

        _totalSupply = _totalSupply.sub(value);
        _balances[account] = _balances[account].sub(value);
        emit Transfer(account, address(0), value);
    }

    /**
     * @dev Approve an address to spend another addresses' tokens.
     * @param owner The address that owns the tokens.
     * @param spender The address that will spend the tokens.
     * @param value The number of tokens that can be spent.
     */

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK _approve
      @tag assume_completion
      @post spender != address(0)
      @post owner != address(0)
      @post __post._allowed[owner][spender] == value
      */
    function _approve(address owner, address spender, uint256 value) internal {
        require(spender != address(0));
        require(owner != address(0));

        _allowed[owner][spender] = value;
        emit Approval(owner, spender, value);
    }

    /**
     * @dev Internal function that burns an amount of the token of a given
     * account, deducting from the sender's allowance for said account. Uses the
     * internal burn function.
     * Emits an Approval event (reflecting the reduced allowance).
     * @param account The account whose tokens will be burnt.
     * @param value The amount that will be burnt.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK FAIL NO_ASF
    /*@CTK burnFrom
      @tag assume_completion
      @post (value <= _allowed[account][msg.sender])
      @post (value <= _balances[account])
      @post (__post._allowed[account][msg.sender]) == (_allowed[account][msg.sender] -
           value)
      @post (__post._balances[account]) == (_balances[account] - (value))
      @post __post._totalSupply == (_totalSupply - value)
     */
    function _burnFrom(address account, uint256 value) internal {
        _burn(account, value);
        _approve(account, msg.sender, _allowed[account][msg.sender].sub(value));
    }
}

/**
 * @title ERC20Detailed token
 * @dev The decimals are only for visualization purposes.
 * All the operations are done using the smallest and indivisible token unit,
 * just as on Ethereum all the operations are done in wei.
 */
contract ERC20Detailed is IERC20 {
    string private _name;
    string private _symbol;
    uint8 private _decimals;

    constructor (string memory name, string memory symbol, uint8 decimals) public {
        _name = name;
        _symbol = symbol;
        _decimals = decimals;
    }

    /**
     * @return the name of the token.
     */
    function name() public view returns (string memory) {
        return _name;
    }

    /**
     * @return the symbol of the token.
     */
    function symbol() public view returns (string memory) {
        return _symbol;
    }

    /**
     * @return the number of decimals of the token.
     */
    function decimals() public view returns (uint8) {
        return _decimals;
    }
}

/**
 * @title Ownable
 * @dev The Ownable contract has an owner address, and provides basic authorization
     control
 * functions, this simplifies the implementation of "user permissions".
 */
contract Ownable {
    address private _owner;

    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner
        );

    /**
     * @dev The Ownable constructor sets the original `owner` of the contract to the
         sender
     * account.
     */
    /*@CTK Ownable
      @tag assume_completion
      @post __post._owner == msg.sender
     */
    constructor () internal {
        _owner = msg.sender;
        emit OwnershipTransferred(address(0), _owner);
    }

    /**
     * @return the address of the owner.
     */
    function owner() public view returns (address) {
        return _owner;
    }

    /**
     * @dev Throws if called by any account other than the owner.
     */
    modifier onlyOwner() {
        require(isOwner());
        _;
    }

    /**
     * @return true if `msg.sender` is the owner of the contract.
     */
    /*@CTK isOwner
       @tag assume_completion
       @post __return == (msg.sender == _owner)
     */
    function isOwner() public view returns (bool) {
        return msg.sender == _owner;
    }

    /**
     * @dev Allows the current owner to relinquish control of the contract.
     * It will not be possible to call the functions with the `onlyOwner`
     * modifier anymore.
     * @notice Renouncing ownership will leave the contract without an owner,
     * thereby removing any functionality that is only available to the owner.
     */
    /*@CTK renounceOwnership
       @tag assume_completion
       @post msg.sender == _owner
      @post __post._owner == address(0)
     */
    function renounceOwnership() public onlyOwner {
        emit OwnershipTransferred(_owner, address(0));
        _owner = address(0);
    }

    /**
     * @dev Allows the current owner to transfer control of the contract to a newOwner
       .
     * @param newOwner The address to transfer ownership to.
     */
    /*@CTK transferOwnership
      @tag assume_completion
      @pre msg.sender == _owner
      @pre newOwner != address(0)
      @post __post._owner == newOwner
     */
    function transferOwnership(address newOwner) public onlyOwner {
        _transferOwnership(newOwner);
    }

    /**
     * @dev Transfers control of the contract to a newOwner.
     * @param newOwner The address to transfer ownership to.
     */
    /*@CTK _transferOwnership
      @tag assume_completion
      @pre msg.sender == _owner
      @pre newOwner != address(0)
      @post __post._owner == newOwner
     */
    function _transferOwnership(address newOwner) internal {
        require(newOwner != address(0));
        emit OwnershipTransferred(_owner, newOwner);
        _owner = newOwner;
    }
}

/**
 * @title  Interface for ERC20 token transfer
 * @dev    This interface contains function for ERC20 token transfer.
 */
interface ITransferable {
    function transfer(address _to, uint _amount) external returns (bool success);
}

contract VantaToken is ERC20, ERC20Detailed, Ownable {
    string _name = "VANTA Token";
    string _symbol = "VANTA";
    uint8 _decimals = 18;
    uint256 _totalSupply = 56200000000000000000000000000;

    constructor() ERC20Detailed(_name, _symbol, _decimals) Ownable() public {
        ERC20._mint(msg.sender, _totalSupply);
    }

    /**
```

```
     * @dev Withdraw the ERC20 Token in the VANTAToken contract.
     * @param erc20 ERC20 Token address.
     * @param to To receive tokens.
     * @param amount Tokens amount.
     */
    function withdrawERC20Token(address erc20, address to, uint256 amount) external
        onlyOwner {
        require(to != address(0x0));
        require(ITransferable(erc20).transfer(to, amount));
    }
}
```