CertiK Verification Report For Blockcloud

Request Date: 2019-04-04
Revision Date: 2019-04-16

Disclaimer

PASS

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

Apr 16, 2019

Summary

This audit report summarises the smart contract verification service requested by Blockcloud. The goal of this security audit is to guarantee that the audited smart contracts are robust enough to avoid any potential security loopholes.
The result of this report is only a reflection of the source code that was determined in this scope, and of the source code at the time of the audit.

Type of Issues

CertiK smart label engine applied 100% coveraged formal verification labels on the source code, and scanned the code using our proprietary static analysis and formal verification engine to detect the follow type of issues.

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Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0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

`transferFrom` succeeds even when the source address is the same as the destination address. There is a potential side effect that reduce the allowance even though balances remain unchanged.

For every issues found, CertiK categorizes them into 3 buckets based on its risk level:

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

- Medium: The code implementation does not match the specification at certain condition, or it could affect the security standard by lost of access control.
- Low: The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerability, but no concern found yet.

Source Code with CertiK Labels

File BlockcloudToken.sol

```
pragma solidity ^0.4.13;

library SafeMath {

  /**
  * @dev Multiplies two numbers, throws on overflow.
  */

  /*@CTK SafeMath_mul
    @tag spec
    @post __reverted == __has_assertion_failure
    @post __has_assertion_failure == __has_overflow
    @post __reverted == false -> __return == a * b
    @post a == 0 -> __return == 0
    @post msg == msg__post
    @post (a > 0 && (a * b / a != b)) == __has_assertion_failure
    @post __addr_map == __addr_map__post
  */
  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
    if (a == 0) {
      return 0;
    }
    uint256 c = a * b;
    assert(c / a == b);
    return c;
  }

  /**
  * @dev Integer division of two numbers, truncating the quotient.
  */

  /*@CTK SafeMath_div
    @tag spec
    @post __reverted == __has_assertion_failure
    @post b == 0 -> __reverted == true // solidity throws on 0.
    @post __has_overflow == true -> __has_assertion_failure == true
    @post __reverted == false -> __return == a / b
    @post msg == msg__post
    @post __addr_map == __addr_map__post
  */
  function div(uint256 a, uint256 b) internal pure returns (uint256) {
    // assert(b > 0); // Solidity automatically throws when dividing by 0
    uint256 c = a / b;
    // assert(a == b * c + a % b); // There is no case in which this doesn't hold
    return c;
  }


  /**
  * @dev Substracts two numbers, throws on overflow (i.e. if subtrahend is greater
      than minuend).
  */

  /*@CTK SafeMath_sub
```

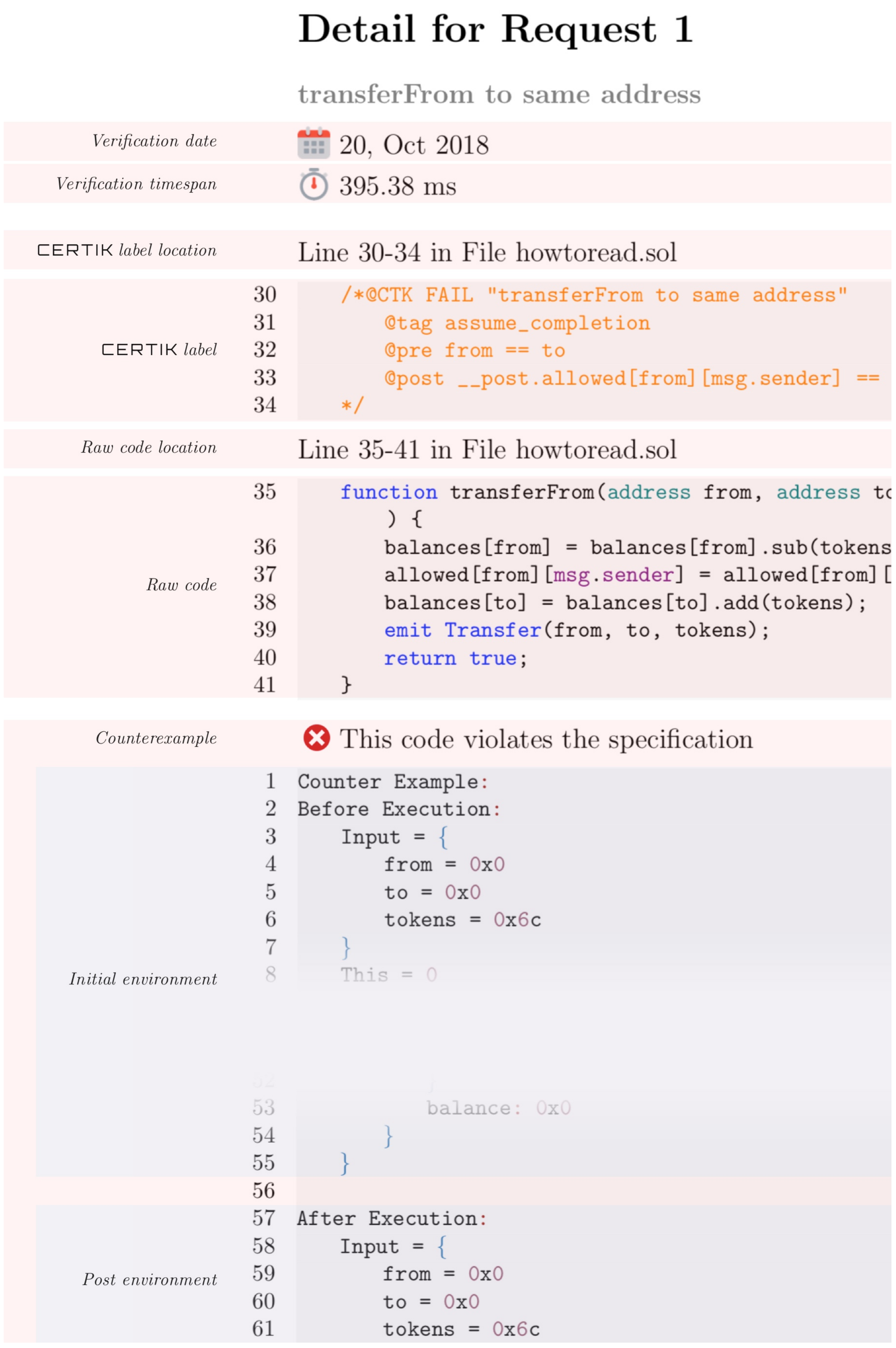
```
    @tag spec
    @post __reverted == __has_assertion_failure
    @post __has_overflow == true -> __has_assertion_failure == true
    @post __reverted == false -> __return == a - b
    @post msg == msg__post
    @post (a < b) == __has_assertion_failure
    @post __addr_map == __addr_map__post
  */
  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    assert(b <= a);
    return a - b;
  }

  /**
  * @dev Adds two numbers, throws on overflow.
  */

  /*@CTK SafeMath_add
    @tag spec
    @post __reverted == __has_assertion_failure
    @post __has_assertion_failure == __has_overflow
    @post __reverted == false -> __return == a + b
    @post msg == msg__post
    @post (a + b < a) == __has_assertion_failure
    @post __addr_map == __addr_map__post
  */
  function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    assert(c >= a);
    return c;
  }
}

contract ERC20Basic {
  function totalSupply() public view returns (uint256);
  function balanceOf(address who) public view returns (uint256);
  function transfer(address to, uint256 value) public returns (bool);
  event Transfer(address indexed from, address indexed to, uint256 value);
}

contract BasicToken is ERC20Basic {
  using SafeMath for uint256;

  mapping(address => uint256) balances;

  uint256 totalSupply_;

  /**
  * @dev total number of tokens in existence
  */

  /*@CTK TotalSupply_return
    @tag spec
    @post __reverted == __has_assertion_failure
    @post __has_assertion_failure == __has_overflow
    @post __reverted == false -> __return == totalSupply_
    @post msg == msg__post
  */
```

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

/**
* @dev transfer token for a specified address
* @param _to The address to transfer to.
* @param _value The amount to be transferred.
*/


/*@CTK transfer_failure_addressEqualZero
  @pre _to == address(0)
  @pre balances[msg.sender] >= _value
  @post __reverted == true
*/
/*@CTK transfer_failure_balanceSmallerValue
  @pre _to != address(0)
  @pre balances[msg.sender] < _value
  @post __reverted == true
*/
/*@CTK transfer_conditions
  @tag assume_completion
  @pre _to != msg.sender
  @post __post.balances[_to] == balances[_to] + _value
  @post __post.balances[msg.sender] == balances[msg.sender] - _value
*/
/*@CTK transfer_same_address
  @tag assume_completion
  @tag no_overflow
  @pre _to == msg.sender
  @post this == __post
*/
function transfer(address _to, uint256 _value) public returns (bool) {
  require(_to != address(0));
  require(_value <= balances[msg.sender]);

  // SafeMath.sub will throw if there is not enough balance.
  balances[msg.sender] = balances[msg.sender].sub(_value);
  balances[_to] = balances[_to].add(_value);
  emit Transfer(msg.sender, _to, _value);
  return true;
}

/**
* @dev Gets the balance of the specified address.
* @param _owner The address to query the the balance of.
* @return An uint256 representing the amount owned by the passed address.
*/

/*@CTK balanceOf
  @post __reverted == false
  @post balance == balances[_owner]
*/
function balanceOf(address _owner) public view returns (uint256 balance) {
  return balances[_owner];
}

```

```
}

contract ERC20 is ERC20Basic {
  function allowance(address owner, address spender) public view returns (uint256);
  function transferFrom(address from, address to, uint256 value) public returns (bool)
      ;
  function approve(address spender, uint256 value) public returns (bool);
  event Approval(address indexed owner, address indexed spender, uint256 value);
}

contract StandardToken is ERC20, BasicToken {

  mapping (address => mapping (address => uint256)) internal allowed;
    event Burn(address indexed burner, uint256 value);


  /**
   * @dev Transfer tokens from one address to another
   * @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 transferFrom
    @tag assume_completion
    @pre _from != _to
    @post __return == true
    @post __post.balances[_to] == balances[_to] + _value
    @post __post.balances[_from] == balances[_from] - _value
    @post __has_overflow == false
  */
  /*@CTK FAIL "transferFrom_sameOwner"
    @pre _from == _to
    @post __post.allowed[_from][msg.sender] == allowed[_from][msg.sender]
  */
  function transferFrom(address _from, address _to, uint256 _value) public returns (
      bool) {
    require(_to != address(0));
    require(_value <= balances[_from]);
    require(_value <= allowed[_from][msg.sender]);

    balances[_from] = balances[_from].sub(_value);
    balances[_to] = balances[_to].add(_value);
    allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
    emit Transfer(_from, _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 approve_success
    @post _value == 0 -> __reverted == false
    @post allowed[msg.sender][_spender] == 0 -> __reverted == false
  */
  /*@CTK approve
    @tag assume_completion
    @post __post.allowed[msg.sender][_spender] == _value
  */
  function approve(address _spender, uint256 _value) public returns (bool) {
    allowed[msg.sender][_spender] = _value;
    emit Approval(msg.sender, _spender, _value);
    return true;
  }

  /**
   * @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 get_allowance
    @post __reverted == false
    @post __return == allowed[_owner][_spender]
    @post this == __post
  */
  function allowance(address _owner, address _spender) public view returns (uint256) {
    return allowed[_owner][_spender];
  }

  /**
   * @dev Increase the amount of tokens that an owner allowed to a spender.
   *
   * approve should be called when allowed[_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
   * @param _spender The address which will spend the funds.
   * @param _addedValue The amount of tokens to increase the allowance by.
   */

  /*@CTK increaseApproval
    @tag assume_completion
    @post __post.allowed[msg.sender][_spender] ==
          allowed[msg.sender][_spender] + _addedValue
    @post __post.allowed[msg.sender][_spender] ==
          allowed[msg.sender][_spender] + _addedValue -> __return == true
  */
  function increaseApproval(address _spender, uint _addedValue) public returns (bool)
      {
    allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_addedValue);
    emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
    return true;
  }

  /**
   * @dev Decrease the amount of tokens that an owner allowed to a spender.
   *
   * approve should be called when allowed[_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
   * @param _spender The address which will spend the funds.
   * @param _subtractedValue The amount of tokens to decrease the allowance by.
   */

  /*@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 decreaseApproval(address _spender, uint _subtractedValue) public returns (
      bool) {
    uint oldValue = allowed[msg.sender][_spender];
    if (_subtractedValue > oldValue) {
      allowed[msg.sender][_spender] = 0;
    } else {
      allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
    }
    emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
    return true;
  }


   /**
   * @dev Burns a specific amount of tokens.
   * @param _value The amount of token to be burned.
   */

  /*@CTK burn_success
    @tag assume_completion
    @post __post.balances[msg.sender] == balances[msg.sender] - _value
    @post __post.totalSupply_ == totalSupply_ - _value
  */
  /*@CTK burn_failure
    @pre _value > balances[msg.sender]
    @post __reverted == true
  */

  function burn(uint256 _value) public {
    _burn(msg.sender, _value);
  }


  /*@CTK _burn_failure
    @pre _value > balances[_who]
    @post __reverted == true
  */
  /*@CTK _burn_success
    @tag assume_completion
    @post __post.balances[_who] == balances[_who] - _value
    @post __post.totalSupply_ == totalSupply_ - _value
  */
  function _burn(address _who, uint256 _value) internal {
    require(_value <= balances[_who]);
    // no need to require value <= totalSupply, since that would imply the
    // sender's balance is greater than the totalSupply, which *should* be an
        assertion failure

    balances[_who] = balances[_who].sub(_value);
    totalSupply_ = totalSupply_.sub(_value);
    emit Burn(_who, _value);
    emit Transfer(_who, address(0), _value);
  }

}

contract BlockcloudToken is StandardToken {
  string public name  = "Blockcloud Token";
  string public symbol = "BLOC";
  uint8 public decimals = 18;


  // ten billion in initial supply
  uint256 public constant INITIAL_SUPPLY = 10000000000;


  /*@CTK BlockcloudToken_initial
    @pre __reverted == false
    @post __post.balances[msg.sender] == __post.totalSupply_
  */
  function BlockcloudToken() public {
    totalSupply_ = INITIAL_SUPPLY * (10 ** uint256(decimals));
    balances[msg.sender] = totalSupply_;
  }


}
```

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

```
Counter Example:
Before Execution:
    Input = {
        from = 0x0
        to = 0x0
        tokens = 0x6c
    }
    This = 0

            balance: 0x0
        }
    }
```

Post environment

```
After Execution:
    Input = {
        from = 0x0
        to = 0x0
        tokens = 0x6c
```

Static Analysis Request

Formal Verification Request 1

SafeMath_mul

📅 16, Apr 2019
⏱ 390.44 ms

Line 9-18 in File BlockcloudToken.sol

```
/*@CTK SafeMath_mul
  @tag spec
  @post __reverted == __has_assertion_failure
  @post __has_assertion_failure == __has_overflow
  @post __reverted == false -> __return == a * b
  @post a == 0 -> __return == 0
  @post msg == msg__post
  @post (a > 0 && (a * b / a != b)) == __has_assertion_failure
  @post __addr_map == __addr_map__post
*/
```

Line 19-26 in File BlockcloudToken.sol

```
function mul(uint256 a, uint256 b) internal pure returns (uint256) {
  if (a == 0) {
    return 0;
  }
  uint256 c = a * b;
  assert(c / a == b);
  return c;
}
```

✓ The code meets the specification

Formal Verification Request 2

SafeMath_div

📅 16, Apr 2019
⏱ 8.9 ms

Line 32-40 in File BlockcloudToken.sol

```
/*@CTK SafeMath_div
  @tag spec
  @post __reverted == __has_assertion_failure
  @post b == 0 -> __reverted == true // solidity throws on 0.
  @post __has_overflow == true -> __has_assertion_failure == true
  @post __reverted == false -> __return == a / b
  @post msg == msg__post
  @post __addr_map == __addr_map__post
*/
```

Line 41-46 in File BlockcloudToken.sol

```
function div(uint256 a, uint256 b) internal pure returns (uint256) {
  // assert(b > 0); // Solidity automatically throws when dividing by 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 3

SafeMath_sub

16, Apr 2019
15.5 ms

Line 53-61 in File BlockcloudToken.sol

```
/*@CTK SafeMath_sub
  @tag spec
  @post __reverted == __has_assertion_failure
  @post __has_overflow == true -> __has_assertion_failure == true
  @post __reverted == false -> __return == a - b
  @post msg == msg__post
  @post (a < b) == __has_assertion_failure
  @post __addr_map == __addr_map__post
*/
```

Line 62-65 in File BlockcloudToken.sol

```
function sub(uint256 a, uint256 b) internal pure returns (uint256) {
  assert(b <= a);
  return a - b;
}
```

✓ The code meets the specification

Formal Verification Request 4

SafeMath_add

16, Apr 2019
19.35 ms

Line 71-79 in File BlockcloudToken.sol

```
/*@CTK SafeMath_add
  @tag spec
  @post __reverted == __has_assertion_failure
  @post __has_assertion_failure == __has_overflow
  @post __reverted == false -> __return == a + b
  @post msg == msg__post
  @post (a + b < a) == __has_assertion_failure
  @post __addr_map == __addr_map__post
*/
```

Line 80-84 in File BlockcloudToken.sol

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

✓ The code meets the specification

Formal Verification Request 5

TotalSupply_return

16, Apr 2019
6.17 ms

Line 105-111 in File BlockcloudToken.sol

```
/*@CTK TotalSupply_return
  @tag spec
  @post __reverted == __has_assertion_failure
  @post __has_assertion_failure == __has_overflow
  @post __reverted == false -> __return == totalSupply_
  @post msg == msg__post
*/
```

Line 112-114 in File BlockcloudToken.sol

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

✓ The code meets the specification

Formal Verification Request 6

transfer_failure_addressEqualZero

16, Apr 2019
45.91 ms

Line 123-127 in File BlockcloudToken.sol

```
/*@CTK transfer_failure_addressEqualZero
  @pre _to == address(0)
  @pre balances[msg.sender] >= _value
  @post __reverted == true
*/
```

Line 145-154 in File BlockcloudToken.sol

```
function transfer(address _to, uint256 _value) public returns (bool) {
  require(_to != address(0));
  require(_value <= balances[msg.sender]);

  // SafeMath.sub will throw if there is not enough balance.
```

```
    balances[msg.sender] = balances[msg.sender].sub(_value);
    balances[_to] = balances[_to].add(_value);
    emit Transfer(msg.sender, _to, _value);
    return true;
  }
```

✓ The code meets the specification

Formal Verification Request 7

transfer_failure_balanceSmallerValue

16, Apr 2019
3.52 ms

Line 128-132 in File BlockcloudToken.sol

```
  /*@CTK transfer_failure_balanceSmallerValue
    @pre _to != address(0)
    @pre balances[msg.sender] < _value
    @post __reverted == true
  */
```

Line 145-154 in File BlockcloudToken.sol

```
  function transfer(address _to, uint256 _value) public returns (bool) {
    require(_to != address(0));
    require(_value <= balances[msg.sender]);

    // SafeMath.sub will throw if there is not enough balance.
    balances[msg.sender] = balances[msg.sender].sub(_value);
    balances[_to] = balances[_to].add(_value);
    emit Transfer(msg.sender, _to, _value);
    return true;
  }
```

✓ The code meets the specification

Formal Verification Request 8

transfer_conditions

16, Apr 2019
106.08 ms

Line 133-138 in File BlockcloudToken.sol

```
  /*@CTK transfer_conditions
    @tag assume_completion
    @pre _to != msg.sender
    @post __post.balances[_to] == balances[_to] + _value
    @post __post.balances[msg.sender] == balances[msg.sender] - _value
  */
```

Line 145-154 in File BlockcloudToken.sol

```
function transfer(address _to, uint256 _value) public returns (bool) {
  require(_to != address(0));
  require(_value <= balances[msg.sender]);

  // SafeMath.sub will throw if there is not enough balance.
  balances[msg.sender] = balances[msg.sender].sub(_value);
  balances[_to] = balances[_to].add(_value);
  emit Transfer(msg.sender, _to, _value);
  return true;
}
```

✓ The code meets the specification

Formal Verification Request 9

transfer_same_address

16, Apr 2019
11.32 ms

Line 139-144 in File BlockcloudToken.sol

```
/*@CTK transfer_same_address
  @tag assume_completion
  @tag no_overflow
  @pre _to == msg.sender
  @post this == __post
*/
```

Line 145-154 in File BlockcloudToken.sol

```
function transfer(address _to, uint256 _value) public returns (bool) {
  require(_to != address(0));
  require(_value <= balances[msg.sender]);

  // SafeMath.sub will throw if there is not enough balance.
  balances[msg.sender] = balances[msg.sender].sub(_value);
  balances[_to] = balances[_to].add(_value);
  emit Transfer(msg.sender, _to, _value);
  return true;
}
```

✓ The code meets the specification

Formal Verification Request 10

balanceOf

16, Apr 2019
7.26 ms

Line 162-165 in File BlockcloudToken.sol

```
/*@CTK balanceOf
  @post __reverted == false
  @post balance == balances[_owner]
*/
```

Line 166-168 in File BlockcloudToken.sol

```
function balanceOf(address _owner) public view returns (uint256 balance) {
  return balances[_owner];
}
```

✓ The code meets the specification

Formal Verification Request 11

transferFrom

16, Apr 2019
239.63 ms

Line 192-199 in File BlockcloudToken.sol

```
 /*@CTK transferFrom
  @tag assume_completion
  @pre _from != _to
  @post __return == true
  @post __post.balances[_to] == balances[_to] + _value
  @post __post.balances[_from] == balances[_from] - _value
  @post __has_overflow == false
*/
```

Line 204-214 in File BlockcloudToken.sol

```
function transferFrom(address _from, address _to, uint256 _value) public returns (
    bool) {
  require(_to != address(0));
  require(_value <= balances[_from]);
  require(_value <= allowed[_from][msg.sender]);

  balances[_from] = balances[_from].sub(_value);
  balances[_to] = balances[_to].add(_value);
  allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
  emit Transfer(_from, _to, _value);
  return true;
}
```

✓ The code meets the specification

Formal Verification Request 12

transferFrom_sameOwner

16, Apr 2019
68.16 ms

Line 200-203 in File BlockcloudToken.sol

```
/*@CTK FAIL "transferFrom_sameOwner"
  @pre _from == _to
  @post __post.allowed[_from][msg.sender] == allowed[_from][msg.sender]
*/
```

Line 204-214 in File BlockcloudToken.sol

```
function transferFrom(address _from, address _to, uint256 _value) public returns (
    bool) {
  require(_to != address(0));
  require(_value <= balances[_from]);
  require(_value <= allowed[_from][msg.sender]);

  balances[_from] = balances[_from].sub(_value);
  balances[_to] = balances[_to].add(_value);
  allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
  emit Transfer(_from, _to, _value);
  return true;
}
```

This code violates the specification

```
Counter Example:
Before Execution:
    Input = {
        _from = 64
        _to = 64
        _value = 127
    }
    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": "StandardToken",
          "balance": 0,
          "contract": {
            "allowed": [
              {
                "key": 64,
                "value": [
                {
                  "key": 4,
                  "value": 0
                },
                {
                  "key": 8,
                  "value": 16
                },
                {
                  "key": 0,
                  "value": 128
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 129
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 129
                }
              ]
            }
          ],
          "balances": [
            {
              "key": 4,
              "value": 1
            },
            {
              "key": 8,
              "value": 0
            },
            {
              "key": 0,
              "value": 0
            },
            {
              "key": 64,
              "value": 128
            },
            {
              "key": "ALL_OTHERS",
              "value": 129
            }
          ],
          "totalSupply_": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
    ]

After Execution:
    Input = {
        _from = 64
        _to = 64
        _value = 127
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = true
        __has_overflow = false
        __has_returned = true
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = true
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "StandardToken",
          "balance": 0,
          "contract": {
            "allowed": [
              {
                "key": 64,
                "value": [
                  {
                    "key": 4,
                    "value": 0
                  },
                  {
                    "key": 8,
                    "value": 16
                  },
                  {
                    "key": 0,
                    "value": 1
                  },
                  {
                    "key": "ALL_OTHERS",
                    "value": 129
                  }
                ]
              },
              {
                "key": "ALL_OTHERS",
                "value": [
                  {
                    "key": "ALL_OTHERS",
                    "value": 129
                  }
                ]
              }
            ],
            "balances": [
              {
                "key": 4,
                "value": 1
              },
              {
                "key": 8,
                "value": 0
              },
              {
                "key": 0,
                "value": 0
              },
              {
                "key": 64,
                "value": 128
              },
              {
                "key": "ALL_OTHERS",
                "value": 129
              }
            ],
            "totalSupply_": 0
          }
        }
      },
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]
```

Formal Verification Request 13

approve_success

16, Apr 2019
14.05 ms

Line 227-230 in File BlockcloudToken.sol

```
/*@CTK approve_success
  @post _value == 0 -> __reverted == false
  @post allowed[msg.sender][_spender] == 0 -> __reverted == false
*/
```

Line 235-239 in File BlockcloudToken.sol

```
function approve(address _spender, uint256 _value) public returns (bool) {
  allowed[msg.sender][_spender] = _value;
  emit Approval(msg.sender, _spender, _value);
  return true;
}
```

✓ The code meets the specification

Formal Verification Request 14

approve

16, Apr 2019
1.9 ms

Line 231-234 in File BlockcloudToken.sol

```
/*@CTK approve
  @tag assume_completion
  @post __post.allowed[msg.sender][_spender] == _value
*/
```

Line 235-239 in File BlockcloudToken.sol

```
function approve(address _spender, uint256 _value) public returns (bool) {
  allowed[msg.sender][_spender] = _value;
  emit Approval(msg.sender, _spender, _value);
  return true;
}
```

✓ The code meets the specification

Formal Verification Request 15

get_allowance

16, Apr 2019
7.48 ms

Line 248-252 in File BlockcloudToken.sol

```
/*@CTK get_allowance
  @post __reverted == false
  @post __return == allowed[_owner][_spender]
  @post this == __post
*/
```

Line 253-255 in File BlockcloudToken.sol

```
function allowance(address _owner, address _spender) public view returns (uint256) {
  return allowed[_owner][_spender];
}
```

✓ The code meets the specification

Formal Verification Request 16

increaseApproval

16, Apr 2019
28.8 ms

Line 268-274 in File BlockcloudToken.sol

```
/*@CTK increaseApproval
  @tag assume_completion
  @post __post.allowed[msg.sender][_spender] ==
        allowed[msg.sender][_spender] + _addedValue
  @post __post.allowed[msg.sender][_spender] ==
        allowed[msg.sender][_spender] + _addedValue -> __return == true
*/
```

Line 275-279 in File BlockcloudToken.sol

```
function increaseApproval(address _spender, uint _addedValue) public returns (bool)
    {
  allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_addedValue);
  emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
  return true;
}
```

✓ The code meets the specification

Formal Verification Request 17

decreaseApproval0

16, Apr 2019
78.06 ms

Line 292-296 in File BlockcloudToken.sol

```
/*@CTK decreaseApproval0
  @pre __return == true
  @pre allowed[msg.sender][_spender] <= _subtractedValue
  @post __post.allowed[msg.sender][_spender] == 0
*/
```

Line 303-312 in File BlockcloudToken.sol

```
function decreaseApproval(address _spender, uint _subtractedValue) public returns (
    bool) {
  uint oldValue = allowed[msg.sender][_spender];
  if (_subtractedValue > oldValue) {
    allowed[msg.sender][_spender] = 0;
  } else {
    allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
  }
  emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
  return true;
}
```

✓ The code meets the specification

Formal Verification Request 18

decreaseApproval

📅 16, Apr 2019
⏱ 18.73 ms

Line 297-302 in File BlockcloudToken.sol

```
/*@CTK decreaseApproval
  @pre __return == true
  @pre allowed[msg.sender][_spender] > _subtractedValue
  @post __post.allowed[msg.sender][_spender] ==
        allowed[msg.sender][_spender] - _subtractedValue
*/
```

Line 303-312 in File BlockcloudToken.sol

```
function decreaseApproval(address _spender, uint _subtractedValue) public returns (
    bool) {
  uint oldValue = allowed[msg.sender][_spender];
  if (_subtractedValue > oldValue) {
    allowed[msg.sender][_spender] = 0;
  } else {
    allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
  }
  emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
  return true;
}
```

✅ The code meets the specification

Formal Verification Request 19

burn_success

📅 16, Apr 2019
⏱ 119.13 ms

Line 320-324 in File BlockcloudToken.sol

```
/*@CTK burn_success
  @tag assume_completion
  @post __post.balances[msg.sender] == balances[msg.sender] - _value
  @post __post.totalSupply_ == totalSupply_ - _value
*/
```

Line 330-332 in File BlockcloudToken.sol

```
function burn(uint256 _value) public {
  _burn(msg.sender, _value);
}
```

✅ The code meets the specification

Formal Verification Request 20

burn_failure

📅 16, Apr 2019
⏱ 2.62 ms

Line 325-328 in File BlockcloudToken.sol

```
/*@CTK burn_failure
  @pre _value > balances[msg.sender]
  @post __reverted == true
*/
```

Line 330-332 in File BlockcloudToken.sol

```
function burn(uint256 _value) public {
  _burn(msg.sender, _value);
}
```

✓ The code meets the specification

Formal Verification Request 21

_burn_failure

📅 16, Apr 2019
⏱ 2.27 ms

Line 335-338 in File BlockcloudToken.sol

```
/*@CTK _burn_failure
  @pre _value > balances[_who]
  @post __reverted == true
*/
```

Line 344-353 in File BlockcloudToken.sol

```
function _burn(address _who, uint256 _value) internal {
  require(_value <= balances[_who]);
  // no need to require value <= totalSupply, since that would imply the
  // sender's balance is greater than the totalSupply, which *should* be an
      assertion failure

  balances[_who] = balances[_who].sub(_value);
  totalSupply_ = totalSupply_.sub(_value);
  emit Burn(_who, _value);
  emit Transfer(_who, address(0), _value);
}
```

✓ The code meets the specification

Formal Verification Request 22

_burn_success

16, Apr 2019
47.45 ms

Line 339-343 in File BlockcloudToken.sol

```
/*@CTK _burn_success
  @tag assume_completion
  @post __post.balances[_who] == balances[_who] - _value
  @post __post.totalSupply_ == totalSupply_ - _value
*/
```

Line 344-353 in File BlockcloudToken.sol

```
function _burn(address _who, uint256 _value) internal {
  require(_value <= balances[_who]);
  // no need to require value <= totalSupply, since that would imply the
  // sender's balance is greater than the totalSupply, which *should* be an
      assertion failure

  balances[_who] = balances[_who].sub(_value);
  totalSupply_ = totalSupply_.sub(_value);
  emit Burn(_who, _value);
  emit Transfer(_who, address(0), _value);
}
```

✓ The code meets the specification

Formal Verification Request 23

BlockcloudToken_initial

16, Apr 2019
27.64 ms

Line 367-370 in File BlockcloudToken.sol

```
/*@CTK BlockcloudToken_initial
  @pre __reverted == false
  @post __post.balances[msg.sender] == __post.totalSupply_
*/
```

Line 371-374 in File BlockcloudToken.sol

```
function BlockcloudToken() public {
  totalSupply_ = INITIAL_SUPPLY * (10 ** uint256(decimals));
  balances[msg.sender] = totalSupply_;
}
```

✓ The code meets the specification