CERTIK

Audit Report

Produced by CertiK

for 

10TH Feb, 2020

CertiK Audit Report For QuizTok

Request Date: 2019-02-07
Revision Date: 2020-02-10
Platform Name: Ethereum

Contents

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Verification Services Agreement between CertiK and QuizTok(the "Company"), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, the "Agreement"). This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

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 QuizTok to discover issues and vulnerabilities in the source code of their qtcon.sol smart contracts. 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 practices 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 three buckets based on overall risk levels:

Critical

Code implementation does not match specification, which could result in the loss of funds for contract owner or users.

Medium

Code implementation does not match the specification under certain conditions, which could affect the security standard by loss of access control.

Low

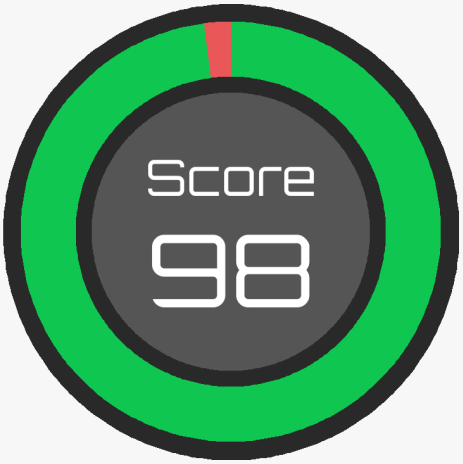
Code implementation does not follow best practices, or uses suboptimal design patterns, which could lead to security vulnerabilities further down the line.

Testing Summary

PASS

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

Feb 10, 2020



Type of Issues

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

Title	Description	Issues	SWC ID
Integer Overflow/ Underflow	An overflow/underflow occurs when an arithmetic operation reaches the maximum or minimum size of a type.	0	SWC-101
Function Incorrectness	Function implementation does not meet specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker can 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 miners to some degree.	0	SWC-116
Insecure Compiler Version	Using a 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.	1	SWC-102 SWC-103
Insecure Randomness	Using block attributes to generate random numbers is unreliable, as they can be influenced by miners 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

Title	Description	Issues	SWC ID
Delegatecall to Untrusted Callee	Calling untrusted contracts is very dangerous, so 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, meaning a malicious user can 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.	1	SWC-110
Deprecated Solidity Features	Several functions and operators in Solidity are deprecated and should not be used.	0	SWC-111
Unused Variables	Unused variables reduce code quality	0	SWC-131

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

Issue 1:

- Recommend renaming the token contract from `SimpleToken` to `QuizToken`

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File qtcon.sol

```
1 pragma solidity ^0.4.18;
```

Version to compile has the following bug: 0.4.18: SignedArrayStorageCopy, ABIEncoderV2StorageArrayWithMultiSlotElement , DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder ABIEncoderV2StorageArrayWithMultiSlotElement 0.4.19: SignedArrayStorageCopy, ABIEncoderV2StorageArrayWithMultiSlotElement , DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.20: SignedArrayStorageCopy, ABIEncoderV2StorageArrayWithMultiSlotElement , DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.21: SignedArrayStorageCopy, ABIEncoderV2StorageArrayWithMultiSlotElement , DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.22: SignedArrayStorageCopy, ABIEncoderV2StorageArrayWithMultiSlotElement , DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, OneOfTwoConstructorsSkipped 0.4.23: SignedArrayStorageCopy, ABIEncoderV2StorageArrayWithMultiSlotElement , DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.24: SignedArrayStorageCopy, ABIEncoderV2StorageArrayWithMultiSlotElement , DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: SignedArrayStorageCopy, ABIEncoderV2StorageArrayWithMultiSlotElement , DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: SignedArrayStorageCopy, ABIEncoderV2StorageArrayWithMultiSlotElement , DynamicConstructorArgumentsClippedABIV2

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

Post environment

```
56
57 After Execution:
58     Input = {
59         from = 0x0
60         to = 0x0
61         tokens = 0x6c
```

Formal Verification Request 1

SafeMath mul

10, Feb 2020
279.37 ms

Line 13-18 in File qtcon.sol

```
13  /*@CTK "SafeMath mul"
14    @post (a > 0) && (((a * b) / a) != b) -> __reverted
15    @post __reverted -> (a > 0) && (((a * b) / a) != b)
16    @post !__reverted -> __return == a * b
17    @post !__reverted == !__has_overflow
18  */
```

Line 19-26 in File qtcon.sol

```
19  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
20    if (a == 0) {
21      return 0;
22    }
23    uint256 c = a * b;
24    assert(c / a == b);
25    return c;
26  }
```

The code meets the specification.

Formal Verification Request 2

SafeMath div

10, Feb 2020
6.53 ms

Line 28-32 in File qtcon.sol

```
28  /*@CTK "SafeMath div"
29    @post b != 0 -> !__reverted
30    @post !__reverted -> __return == a / b
31    @post !__reverted -> !__has_overflow
32  */
```

Line 33-38 in File qtcon.sol

```
33  function div(uint256 a, uint256 b) internal pure returns (uint256) {
34    // assert(b > 0); // Solidity automatically throws when dividing by 0
35    uint256 c = a / b;
36    // assert(a == b * c + a % b); // There is no case in which this doesn't hold
37    return c;
38  }
```

The code meets the specification.

Formal Verification Request 3

SafeMath sub

10, Feb 2020
11.58 ms

Line 40-44 in File qtcon.sol

```
40  /*@CTK "SafeMath sub"
41    @post (a < b) == __reverted
42    @post !__reverted -> __return == a - b
43    @post !__reverted -> !__has_overflow
44  */
```

Line 45-48 in File qtcon.sol

```
45  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
46    assert(b <= a);
47    return a - b;
48  }
```

The code meets the specification.

Formal Verification Request 4

SafeMath add

10, Feb 2020
14.07 ms

Line 50-54 in File qtcon.sol

```
50  /*@CTK "SafeMath add"
51    @post (a + b < a || a + b < b) == __reverted
52    @post !__reverted -> __return == a + b
53    @post !__reverted -> !__has_overflow
54  */
```

Line 55-59 in File qtcon.sol

```
55  function add(uint256 a, uint256 b) internal pure returns (uint256) {
56    uint256 c = a + b;
57    assert(c >= a);
58    return c;
59  }
```

The code meets the specification.

Formal Verification Request 5

If method completes, integer overflow would not happen.

10, Feb 2020
4.12 ms

Line 69 in File qtcon.sol

```
69  //@CTK NO_OVERFLOW
```

Line 75-77 in File qtcon.sol

```
75  function totalSupply() public view returns (uint256) {
76    return totalSupply_;
77  }
```

✅ The code meets the specification.

Formal Verification Request 6

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

📅 10, Feb 2020
⏱ 0.35 ms

Line 70 in File qtcon.sol

```
70  //@CTK NO_BUF_OVERFLOW
```

Line 75-77 in File qtcon.sol

```
75  function totalSupply() public view returns (uint256) {
76    return totalSupply_;
77  }
```

✅ The code meets the specification.

Formal Verification Request 7

Method will not encounter an assertion failure.

📅 10, Feb 2020
⏱ 0.35 ms

Line 71 in File qtcon.sol

```
71  //@CTK NO_ASF
```

Line 75-77 in File qtcon.sol

```
75  function totalSupply() public view returns (uint256) {
76    return totalSupply_;
77  }
```

✅ The code meets the specification.

Formal Verification Request 8

totalSupply

📅 10, Feb 2020
⏱ 0.49 ms

Line 72-74 in File qtcon.sol

```
/*@CTK totalSupply
  @post __return == totalSupply_
*/
```

Line 75-77 in File qtcon.sol

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

The code meets the specification.

Formal Verification Request 9

transfer

10, Feb 2020
141.88 ms

Line 80-88 in File qtcon.sol

```
/*@CTK transfer
  @tag assume_completion
  @pre _to != msg.sender
  @pre _value <= balances[msg.sender]
  @post _to != address(0)
  @post __post.balances[msg.sender] == balances[msg.sender] - _value
  @post __post.balances[_to] == balances[_to] + _value
  @post __return == true
*/
```

Line 89-98 in File qtcon.sol

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


  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

10, Feb 2020
4.36 ms

Line 100-102 in File qtcon.sol

```
/*@CTK balanceOf
  @post balance == __post.balances[_owner]
*/
```

Line 103-105 in File qtcon.sol

```
103  function balanceOf(address _owner) public view returns (uint256 balance) {
104    return balances[_owner];
105  }
```

The code meets the specification.

Formal Verification Request 11

transferFrom

10, Feb 2020
254.19 ms

Line 120-132 in File qtcon.sol

```
120  /*@CTK transferFrom
121    @tag assume_completion
122    @pre _to != _from
123    @pre _value <= balances[_from]
124    @pre _value <= allowed[_from][msg.sender]
125    @post _to != address(0)
126    @post _value <= balances[_from]
127    @post _value <= allowed[_from][msg.sender]
128    @post __post.balances[_from] == balances[_from] - _value
129    @post __post.allowed[_from][msg.sender] == allowed[_from][msg.sender] - _value
130    @post __post.balances[_to] == balances[_to] + _value
131    @post __return == true
132  */
```

Line 139-149 in File qtcon.sol

```
139  function transferFrom(address _from, address _to, uint256 _value) public returns (bool) {
140    require(_to != address(0));
141    require(_value <= balances[_from]);
142    require(_value <= allowed[_from][msg.sender]);
143
144    balances[_from] = balances[_from].sub(_value);
145    balances[_to] = balances[_to].add(_value);
146    allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
147    emit Transfer(_from, _to, _value);
148    return true;
149  }
```

The code meets the specification.

Formal Verification Request 12

transferForm_fail

10, Feb 2020
11.21 ms

Line 133-138 in File qtcon.sol

```
/*@CTK "transferForm_fail"
  @tag assume_completion
  @post (_to == address(0)) -> __reverted
  @post (_value > balances[_from]) -> __reverted
  @post (_value > allowed[_from][msg.sender]) -> __reverted
*/
```

Line 139-149 in File qtcon.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 13

If method completes, integer overflow would not happen.

📅 10, Feb 2020
⏱ 7.15 ms

Line 151 in File qtcon.sol

```
//@CTK NO_OVERFLOW
```

Line 157-161 in File qtcon.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

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

📅 10, Feb 2020
⏱ 0.31 ms

Line 152 in File qtcon.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 157-161 in File qtcon.sol

```
157  function approve(address _spender, uint256 _value) public returns (bool) {
158    allowed[msg.sender][_spender] = _value;
159    emit Approval(msg.sender, _spender, _value);
160    return true;
161  }
```

The code meets the specification.

Formal Verification Request 15

Method will not encounter an assertion failure.

10, Feb 2020
0.3 ms

Line 153 in File qtcon.sol

```
153  //@CTK NO_ASF
```

Line 157-161 in File qtcon.sol

```
157  function approve(address _spender, uint256 _value) public returns (bool) {
158    allowed[msg.sender][_spender] = _value;
159    emit Approval(msg.sender, _spender, _value);
160    return true;
161  }
```

The code meets the specification.

Formal Verification Request 16

approve

10, Feb 2020
1.06 ms

Line 154-156 in File qtcon.sol

```
154  /*@CTK approve
155    @post __post.allowed[msg.sender][_spender] == _value
156  */
```

Line 157-161 in File qtcon.sol

```
157  function approve(address _spender, uint256 _value) public returns (bool) {
158    allowed[msg.sender][_spender] = _value;
159    emit Approval(msg.sender, _spender, _value);
160    return true;
161  }
```

The code meets the specification.

Formal Verification Request 17

If method completes, integer overflow would not happen.

📅 10, Feb 2020
⏱ 4.35 ms

Line 163 in File qtcon.sol

```
163  //@CTK NO_OVERFLOW
```

Line 169-171 in File qtcon.sol

```
169  function allowance(address _owner, address _spender) public view returns (uint256) {
170    return allowed[_owner][_spender];
171  }
```

✅ The code meets the specification.

Formal Verification Request 18

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

📅 10, Feb 2020
⏱ 0.36 ms

Line 164 in File qtcon.sol

```
164  //@CTK NO_BUF_OVERFLOW
```

Line 169-171 in File qtcon.sol

```
169  function allowance(address _owner, address _spender) public view returns (uint256) {
170    return allowed[_owner][_spender];
171  }
```

✅ The code meets the specification.

Formal Verification Request 19

Method will not encounter an assertion failure.

📅 10, Feb 2020
⏱ 0.3 ms

Line 165 in File qtcon.sol

```
165  //@CTK NO_ASF
```

Line 169-171 in File qtcon.sol

```
169  function allowance(address _owner, address _spender) public view returns (uint256) {
170    return allowed[_owner][_spender];
171  }
```

✅ The code meets the specification.

Formal Verification Request 20

allowance

10, Feb 2020
0.3 ms

Line 166-168 in File qtcon.sol

```
166  /*@CTK allowance
167    @post __return == allowed[_owner][_spender]
168  */
```

Line 169-171 in File qtcon.sol

```
169  function allowance(address _owner, address _spender) public view returns (uint256) {
170    return allowed[_owner][_spender];
171  }
```

The code meets the specification.

Formal Verification Request 21

increaseApproval

10, Feb 2020
36.99 ms

Line 174-177 in File qtcon.sol

```
174  /*@CTK increaseApproval
175    @tag assume_completion
176    @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] +
           _addedValue
177   */
```

Line 178-182 in File qtcon.sol

```
178  function increaseApproval(address _spender, uint _addedValue) public returns (bool) {
179    allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_addedValue);
180    emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
181    return true;
182  }
```

The code meets the specification.

Formal Verification Request 22

If method completes, integer overflow would not happen.

10, Feb 2020
36.07 ms

Line 184 in File qtcon.sol

```
184  //@CTK NO_OVERFLOW
```

Line 197-206 in File qtcon.sol

```
197  function decreaseApproval(address _spender, uint _subtractedValue) public returns (bool) {
198    uint oldValue = allowed[msg.sender][_spender];
199    if (_subtractedValue > oldValue) {
200      allowed[msg.sender][_spender] = 0;
201    } else {
202      allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
203    }
204    emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
205    return true;
206  }
```

The code meets the specification.

Formal Verification Request 23

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

10, Feb 2020
0.64 ms

Line 185 in File qtcon.sol

```
185  //@CTK NO_BUF_OVERFLOW
```

Line 197-206 in File qtcon.sol

```
197  function decreaseApproval(address _spender, uint _subtractedValue) public returns (bool) {
198    uint oldValue = allowed[msg.sender][_spender];
199    if (_subtractedValue > oldValue) {
200      allowed[msg.sender][_spender] = 0;
201    } else {
202      allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
203    }
204    emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
205    return true;
206  }
```

The code meets the specification.

Formal Verification Request 24

Method will not encounter an assertion failure.

10, Feb 2020
1.21 ms

Line 186 in File qtcon.sol

```
186  //@CTK NO_ASF
```

Line 197-206 in File qtcon.sol

```
197  function decreaseApproval(address _spender, uint _subtractedValue) public returns (bool) {
198    uint oldValue = allowed[msg.sender][_spender];
199    if (_subtractedValue > oldValue) {
200      allowed[msg.sender][_spender] = 0;
201    } else {
```

```
202       allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
203     }
204     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
205     return true;
206   }
```

The code meets the specification.

Formal Verification Request 25

decreaseApproval subval > oldVal

10, Feb 2020
1.83 ms

Line 187-191 in File qtcon.sol

```
187   /*@CTK "decreaseApproval subval > oldVal"
188     @tag assume_completion
189     @pre _subtractedValue > allowed[msg.sender][_spender]
190     @post __post.allowed[msg.sender][_spender] == 0
191    */
```

Line 197-206 in File qtcon.sol

```
197   function decreaseApproval(address _spender, uint _subtractedValue) public returns (bool) {
198     uint oldValue = allowed[msg.sender][_spender];
199     if (_subtractedValue > oldValue) {
200       allowed[msg.sender][_spender] = 0;
201     } else {
202       allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
203     }
204     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
205     return true;
206   }
```

The code meets the specification.

Formal Verification Request 26

decreaseApproval subval <= oldVal

10, Feb 2020
2.25 ms

Line 192-196 in File qtcon.sol

```
192   /*@CTK "decreaseApproval subval <= oldVal"
193     @tag assume_completion
194     @pre _subtractedValue <= allowed[msg.sender][_spender]
195     @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] -
          _subtractedValue
196    */
```

Line 197-206 in File qtcon.sol

```
197  function decreaseApproval(address _spender, uint _subtractedValue) public returns (bool) {
198    uint oldValue = allowed[msg.sender][_spender];
199    if (_subtractedValue > oldValue) {
200      allowed[msg.sender][_spender] = 0;
201    } else {
202      allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
203    }
204    emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
205    return true;
206  }
```

The code meets the specification.

Formal Verification Request 27

constructor

10, Feb 2020
8.21 ms

Line 217-221 in File qtcon.sol

```
217  /*@CTK "constructor"
218    @tag assume_completion
219    @post __post.totalSupply_ == INITIAL_SUPPLY
220    @post __post.balances[msg.sender] == INITIAL_SUPPLY
221  */
```

Line 222-226 in File qtcon.sol

```
222  constructor() public {
223    totalSupply_ = INITIAL_SUPPLY;
224    balances[msg.sender] = INITIAL_SUPPLY;
225    emit Transfer(0x0, msg.sender, INITIAL_SUPPLY);
226  }
```

The code meets the specification.

Source Code with CertiK Labels

File qtcon.sol

```
1  pragma solidity ^0.4.18;
2
3
4  contract ERC20Basic {
5    function totalSupply() public view returns (uint256);
6    function balanceOf(address who) public view returns (uint256);
7    function transfer(address to, uint256 value) public returns (bool);
8    event Transfer(address indexed from, address indexed to, uint256 value);
9  }
10
11 library SafeMath {
12
13   /*@CTK "SafeMath mul"
14     @post (a > 0) && (((a * b) / a) != b) -> __reverted
15     @post __reverted -> (a > 0) && (((a * b) / a) != b)
16     @post !__reverted -> __return == a * b
17     @post !__reverted == !__has_overflow
18   */
19   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
20     if (a == 0) {
21       return 0;
22     }
23     uint256 c = a * b;
24     assert(c / a == b);
25     return c;
26   }
27
28   /*@CTK "SafeMath div"
29     @post b != 0 -> !__reverted
30     @post !__reverted -> __return == a / b
31     @post !__reverted -> !__has_overflow
32   */
33   function div(uint256 a, uint256 b) internal pure returns (uint256) {
34     // assert(b > 0); // Solidity automatically throws when dividing by 0
35     uint256 c = a / b;
36     // assert(a == b * c + a % b); // There is no case in which this doesn't hold
37     return c;
38   }
39
40   /*@CTK "SafeMath sub"
41     @post (a < b) == __reverted
42     @post !__reverted -> __return == a - b
43     @post !__reverted -> !__has_overflow
44   */
45   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
46     assert(b <= a);
47     return a - b;
48   }
49
50   /*@CTK "SafeMath add"
51     @post (a + b < a || a + b < b) == __reverted
52     @post !__reverted -> __return == a + b
53     @post !__reverted -> !__has_overflow
54   */
```

```
55    function add(uint256 a, uint256 b) internal pure returns (uint256) {
56      uint256 c = a + b;
57      assert(c >= a);
58      return c;
59    }
60  }
61
62  contract BasicToken is ERC20Basic {
63    using SafeMath for uint256;
64
65    mapping(address => uint256) balances;
66
67    uint256 totalSupply_;
68
69    //@CTK NO_OVERFLOW
70    //@CTK NO_BUF_OVERFLOW
71    //@CTK NO_ASF
72    /*@CTK totalSupply
73      @post __return == totalSupply_
74    */
75    function totalSupply() public view returns (uint256) {
76      return totalSupply_;
77    }
78
79
80    /*@CTK transfer
81      @tag assume_completion
82      @pre _to != msg.sender
83      @pre _value <= balances[msg.sender]
84      @post _to != address(0)
85      @post __post.balances[msg.sender] == balances[msg.sender] - _value
86      @post __post.balances[_to] == balances[_to] + _value
87      @post __return == true
88    */
89    function transfer(address _to, uint256 _value) public returns (bool) {
90      require(_to != address(0));
91      require(_value <= balances[msg.sender]);
92
93
94      balances[msg.sender] = balances[msg.sender].sub(_value);
95      balances[_to] = balances[_to].add(_value);
96      emit Transfer(msg.sender, _to, _value);
97      return true;
98    }
99
100   /*@CTK balanceOf
101     @post balance == __post.balances[_owner]
102   */
103   function balanceOf(address _owner) public view returns (uint256 balance) {
104     return balances[_owner];
105   }
106 }
107
108 contract ERC20 is ERC20Basic {
109   function allowance(address owner, address spender) public view returns (uint256);
110   function transferFrom(address from, address to, uint256 value) public returns (bool);
111   function approve(address spender, uint256 value) public returns (bool);
112   event Approval(address indexed owner, address indexed spender, uint256 value);
```

```
113 }
114
115
116 contract StandardToken is ERC20, BasicToken {
117
118   mapping (address => mapping (address => uint256)) internal allowed;
119
120   /*@CTK transferFrom
121     @tag assume_completion
122     @pre _to != _from
123     @pre _value <= balances[_from]
124     @pre _value <= allowed[_from][msg.sender]
125     @post _to != address(0)
126     @post _value <= balances[_from]
127     @post _value <= allowed[_from][msg.sender]
128     @post __post.balances[_from] == balances[_from] - _value
129     @post __post.allowed[_from][msg.sender] == allowed[_from][msg.sender] - _value
130     @post __post.balances[_to] == balances[_to] + _value
131     @post __return == true
132   */
133   /*@CTK "transferForm_fail"
134     @tag assume_completion
135     @post (_to == address(0)) -> __reverted
136     @post (_value > balances[_from]) -> __reverted
137     @post (_value > allowed[_from][msg.sender]) -> __reverted
138   */
139   function transferFrom(address _from, address _to, uint256 _value) public returns (bool) {
140     require(_to != address(0));
141     require(_value <= balances[_from]);
142     require(_value <= allowed[_from][msg.sender]);
143
144     balances[_from] = balances[_from].sub(_value);
145     balances[_to] = balances[_to].add(_value);
146     allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
147     emit Transfer(_from, _to, _value);
148     return true;
149   }
150
151   //@CTK NO_OVERFLOW
152   //@CTK NO_BUF_OVERFLOW
153   //@CTK NO_ASF
154   /*@CTK approve
155     @post __post.allowed[msg.sender][_spender] == _value
156   */
157   function approve(address _spender, uint256 _value) public returns (bool) {
158     allowed[msg.sender][_spender] = _value;
159     emit Approval(msg.sender, _spender, _value);
160     return true;
161   }
162
163   //@CTK NO_OVERFLOW
164   //@CTK NO_BUF_OVERFLOW
165   //@CTK NO_ASF
166   /*@CTK allowance
167     @post __return == allowed[_owner][_spender]
168   */
169   function allowance(address _owner, address _spender) public view returns (uint256) {
170     return allowed[_owner][_spender];
```

```
171   }
172
173
174   /*@CTK increaseApproval
175     @tag assume_completion
176     @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] +
          _addedValue
177    */
178   function increaseApproval(address _spender, uint _addedValue) public returns (bool) {
179     allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_addedValue);
180     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
181     return true;
182   }
183
184   //@CTK NO_OVERFLOW
185   //@CTK NO_BUF_OVERFLOW
186   //@CTK NO_ASF
187   /*@CTK "decreaseApproval subval > oldVal"
188     @tag assume_completion
189     @pre _subtractedValue > allowed[msg.sender][_spender]
190     @post __post.allowed[msg.sender][_spender] == 0
191    */
192   /*@CTK "decreaseApproval subval <= oldVal"
193     @tag assume_completion
194     @pre _subtractedValue <= allowed[msg.sender][_spender]
195     @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] -
          _subtractedValue
196    */
197   function decreaseApproval(address _spender, uint _subtractedValue) public returns (bool) {
198     uint oldValue = allowed[msg.sender][_spender];
199     if (_subtractedValue > oldValue) {
200       allowed[msg.sender][_spender] = 0;
201     } else {
202       allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
203     }
204     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
205     return true;
206   }
207 }
208
209 contract SimpleToken is StandardToken {
210
211   string public constant name = "Qtcon";
212   string public constant symbol = "QTCON";
213   uint8 public constant decimals = 18;
214
215   uint256 public constant INITIAL_SUPPLY = 60000000000 * (10 ** uint256(decimals));
216
217   /*@CTK "constructor"
218     @tag assume_completion
219     @post __post.totalSupply_ == INITIAL_SUPPLY
220     @post __post.balances[msg.sender] == INITIAL_SUPPLY
221   */
222   constructor() public {
223     totalSupply_ = INITIAL_SUPPLY;
224     balances[msg.sender] = INITIAL_SUPPLY;
225     emit Transfer(0x0, msg.sender, INITIAL_SUPPLY);
226   }
```

```
227 }
```