# CERTIK AUDIT REPORT FOR QURAS



Request Date: 2019-09-02 Revision Date: 2019-09-10 Platform Name: Ethereum







px

# Contents

Disclaimer	1
About CertiK	2
Exective Summary	3
Vulnerability Classification	3
Testing Summary Audit Score	<b>4</b> 4 4
Manual Review Notes	6
Static Analysis Results	7
Formal Verification Results How to read	8
Source Code with CertiK Labels	25





## 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 Quras(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/





## **Exective Summary**

This report has been prepared as the product of the Smart Contract Audit request by Quras. This audit was conducted to discover issues and vulnerabilities in the source code of Quras's Smart Contracts. Utilizing CertiK's Formal Verification Platform, Static Analysis, and Manual Review, a comprehensive examination has been performed. The px auditing process pays special attention to the following considerations.

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessment of the codebase for 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**

For every issue 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 conditions, 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 vulnerabilies, but no concern found yet.

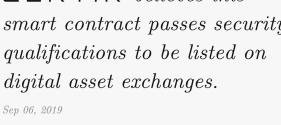




# Testing Summary



**CERTIK** believes this smart contract passes security qualifications to be listed on





# 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	An overflow/underflow happens when an arithmetic	0	SWC-101
and Underflow	operation reaches the maximum or minimum size of		
	a type.		
Function incor-	Function implementation does not meet the specifi-	0	
rectness	cation, leading to intentional or unintentional vul-		
	nerabilities.		
Buffer Overflow	An attacker is able to write to arbitrary storage lo-	0	SWC-124
	cations of a contract if array of out bound happens		
Reentrancy	A malicious contract can call back into the calling	0	SWC-107
	contract before the first invocation of the function is		
	finished.		
Transaction Or-	A race condition vulnerability occurs when code de-	0	SWC-114
der Dependence	pends on the order of the transactions submitted to		
	it.		
Timestamp De-	Timestamp can be influenced by minors to some de-	0	SWC-116
pendence	gree.		
Insecure Com-	Using an fixed outdated compiler version or float-	1	SWC-102
piler Version	ing pragma can be problematic, if there are publicly		SWC-103
	disclosed bugs and issues that affect the current com-		
	piler version used.		
Insecure Ran-	Block attributes are insecure to generate random	0	SWC-120
domness	numbers, as they can be influenced by minors to		
	some degree.		





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

## Vulnerability Details



No issue found.

Medium

No issue found.

### Low

• transfer(), transferFrom(): Token transferable to zero address.





## Manual Review Notes

### Review Details

Source Code SHA-256 Checksum

• XQC.sol fd55d785bebfab6410770570908be6083f02fe9c408bf42bdcae0a9f1b0bab1b

#### Summary

CertiK was chosen by Quras to audit the design and implementation of its xqc 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.

- transfer(), transferFrom(): Recommend adding checks to prevent transferring to zero address unless it is for token burning purpose.
- approve(), approveAndCall(): Recommend adding checks to make sure no approval larger than the current balance is allowed.





## Static Analysis Results

INSECURE\_COMPILER\_VERSION

Line 1 in File XQC.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 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, EventStructWron Data, NestedArrayFunctionCallDecoder 0.4.21: SignedArrayStorageCopy, ABIEncoderV2StorageArray 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, EventStructWron Data 0.4.24: SignedArrayStorageCopy, ABIEncoderV2StorageArrayWithMultiSlotElement, DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructionPointerInC tor\_0.4.x, IncorrectEventSignatureInLibraries\_0.4.x, ABIEncoderV2PackedStorage\_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: SignedArrayStorageCopy, ABI-EncoderV2StorageArrayWithMultiSlotElement, DynamicConstructorArgumentsClipped-ABIV2, UninitializedFunctionPointerInConstructor\_0.4.x, IncorrectEventSignatureInLibraries\_0.4.x, ABIEncoderV2PackedStorage\_0.4.x 0.4.26: SignedArrayStorageCopy, ABI-EncoderV2StorageArrayWithMultiSlotElement, DynamicConstructorArgumentsClipped-ABIV2





## Formal Verification Results

### How to read

## Detail for Request 1

transferFrom to same address

```
Verification date
                        20, Oct 2018
                        \bullet 395.38 ms
 Verification\ timespan
CERTIK label location
                        Line 30-34 in File howtoread.sol
                    30
                            /*@CTK FAIL "transferFrom to same address"
                    31
                                @tag assume_completion
     \Box \mathsf{ERTIK}\ label
                    32
                                @pre from == to
                    33
                                @post __post.allowed[from][msg.sender] ==
                    34
    Raw code location
                        Line 35-41 in File howtoread.sol
                    35
                            function transferFrom(address from, address to
                    36
                                balances[from] = balances[from].sub(tokens
                                allowed[from][msg.sender] = allowed[from][
                    37
          Raw\ code
                    38
                                balances[to] = balances[to].add(tokens);
                    39
                                emit Transfer(from, to, tokens);
                    40
                                return true;
                    41
                         This code violates the specification
     Counter example \\
                        Counter Example:
                     2
                        Before Execution:
                     3
                            Input = {
                                from = 0x0
                     4
                                to = 0x0
                     6
                                tokens = 0x6c
                     7
                            This = 0
  Initial environment
                                    balance: 0x0
                    54
                    55
                    56
                    57
                        After Execution:
                    58
                            Input = {
                                from = 0x0
                    59
    Post environment
                    60
                                to = 0x0
                    61
                                tokens = 0x6c
```





SafeMath add

```
mathred{m} 06, Sep 2019
```

(i) 13.65 ms

#### Line 22-30 in File XQC.sol

```
22
       /*@CTK "SafeMath add"
23
         @tag spec
24
         @tag is_pure
25
         @post (a + b < a || a + b < b) == __reverted</pre>
26
         @post !__reverted -> c == a + b
27
         @post !__reverted -> !__has_overflow
28
         @post !__reverted -> !__has_assertion_failure
         @post !(__has_buf_overflow)
29
30
```

#### Line 31-34 in File XQC.sol

```
function safeAdd(uint a, uint b) public pure returns (uint c) {
    c = a + b;
    require(c >= a);
}
```

The code meets the specification.

## Formal Verification Request 2

SafeMath sub

## 06, Sep 2019

 $\bullet$  10.53 ms

### Line 35-43 in File XQC.sol

```
35
       /*@CTK "SafeMath sub"
36
        @tag spec
37
        @tag is_pure
38
        @post (b > a) == __reverted
39
        @post ! reverted -> c == a - b
        @post !__reverted -> !__has_overflow
40
        @post !__reverted -> !__has_assertion_failure
41
42
        @post !(__has_buf_overflow)
43
```

#### Line 44-47 in File XQC.sol

```
function safeSub(uint a, uint b) public pure returns (uint c) {
    require(b <= a);
    c = a - b;
}</pre>
```





SafeMath mul zero

Line 48-53 in File XQC.sol

```
/*@CTK "SafeMath mul zero"

d9     @tag spec

0tag is_pure

f1     @pre (a == 0)

post c == 0

*/
```

Line 64-67 in File XQC.sol

```
function safeMul(uint a, uint b) public pure returns (uint c) {
    c = a * b;
    require(a == 0 || c / a == b);
}
```

The code meets the specification.

## Formal Verification Request 4

SafeMath mul nonzero

```
6.32 ms
```

Line 54-63 in File XQC.sol

```
/*@CTK "SafeMath mul nonzero"
54
55
         @tag spec
56
         @tag is_pure
57
         @pre (a != 0)
58
         @post (a * b / a != b) == __reverted
59
         @post !\_reverted \rightarrow c == a * b
60
         @post !__reverted -> !__has_overflow
61
         Opost ! reverted -> ! has assertion failure
62
         @post !(__has_buf_overflow)
63
```

Line 64-67 in File XQC.sol

```
function safeMul(uint a, uint b) public pure returns (uint c) {
    c = a * b;
    require(a == 0 || c / a == b);
}
```





SafeMath div

```
## 06, Sep 2019
```

(i) 11.82 ms

#### Line 68-76 in File XQC.sol

```
/*@CTK "SafeMath div"
68
69
         @tag spec
         @tag is_pure
70
71
         @post (b == 0) == __reverted
72
         @post !__reverted -> c == a / b
73
         @post !__reverted -> !__has_overflow
74
         @post !__reverted -> !__has_assertion_failure
         @post !(__has_buf_overflow)
75
76
```

#### Line 77-80 in File XQC.sol

```
function safeDiv(uint a, uint b) public pure returns (uint c) {
    require(b > 0);
    c = a / b;
}
```

The code meets the specification.

## Formal Verification Request 6

transferOwnership

```
☆ 06, Sep 2019
```

11.46 ms

### Line 129-133 in File XQC.sol

```
/*@CTK transferOwnership
130     @tag assume_completion
131     @pre msg.sender == owner
132     @post __post.newOwner == _newOwner
133     */
```

Line 134-136 in File XQC.sol

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

The code meets the specification.

## Formal Verification Request 7

acceptOwnership

```
## 06, Sep 2019
```

**1**3.67 ms





#### Line 137-142 in File XQC.sol

144 require(msg.sender == newOwner);
145 OwnershipTransferred(owner, newOwner);
146 owner = newOwner;
147 newOwner = address(0);
148 }

The code meets the specification.

## Formal Verification Request 8

If method completes, integer overflow would not happen.

```
6 06, Sep 201918.11 ms
```

Line 169 in File XQC.sol

```
169 //@CTK NO_OVERFLOW
```

Line 176-183 in File XQC.sol

```
function QurasTestCoin1() public {
    symbol = "XQC1";
    name = "Quras Test Coin1";
    decimals = 8;
    _totalSupply = 888888888000000000;
    balances[0x6e0004D2639E9D45bCD3Af0498C379dBAb598E3E] = _totalSupply;
    Transfer(address(0), 0x6e0004D2639E9D45bCD3Af0498C379dBAb598E3E, _totalSupply);
}
```

The code meets the specification.

## Formal Verification Request 9

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

```
6, Sep 2019
0.34 ms
```

Line 170 in File XQC.sol

```
170 //@CTK NO_BUF_OVERFLOW
```

Line 176-183 in File XQC.sol





```
176
        function QurasTestCoin1() public {
           symbol = "XQC1";
177
           name = "Quras Test Coin1";
178
179
           decimals = 8;
180
           _totalSupply = 8888888800000000;
           balances[0x6e0004D2639E9D45bCD3Af0498C379dBAb598E3E] = _totalSupply;
181
182
           Transfer(address(0), 0x6e0004D2639E9D45bCD3Af0498C379dBAb598E3E, _totalSupply);
183
        }
```

The code meets the specification.

## Formal Verification Request 10

Method will not encounter an assertion failure.

```
6 06, Sep 20190 0.34 ms
```

Line 171 in File XQC.sol

```
171 //@CTK NO_ASF
```

Line 176-183 in File XQC.sol

```
function QurasTestCoin1() public {
    symbol = "XQC1";
    name = "Quras Test Coin1";
    decimals = 8;
    _totalSupply = 88888888800000000;
    balances[0x6e0004D2639E9D45bCD3Af0498C379dBAb598E3E] = _totalSupply;
    Transfer(address(0), 0x6e0004D2639E9D45bCD3Af0498C379dBAb598E3E, _totalSupply);
}
```

The code meets the specification.

## Formal Verification Request 11

QurasTestCoin1

```
6, Sep 2019
0.97 ms
```

Line 172-175 in File XQC.sol

```
172     /*@CTK QurasTestCoin1
173     @tag assume_completion
174     @post __post._totalSupply == 88888888800000000
175     */
```

Line 176-183 in File XQC.sol

```
function QurasTestCoin1() public {
    symbol = "XQC1";
    name = "Quras Test Coin1";
    decimals = 8;
    _totalSupply = 88888888800000000;
    balances[0x6e0004D2639E9D45bCD3Af0498C379dBAb598E3E] = _totalSupply;
```





```
Transfer(address(0), 0x6e0004D2639E9D45bCD3Af0498C379dBAb598E3E, _totalSupply);
183 }
```

The code meets the specification.

## Formal Verification Request 12

If method completes, integer overflow would not happen.

```
## 06, Sep 2019
10.5 ms
```

Line 189 in File XQC.sol

```
189 //@CTK FAIL NO_OVERFLOW
```

Line 197-199 in File XQC.sol

```
function totalSupply() public constant returns (uint) {
return _totalSupply - balances[address(0)];
}
```

This code violates the specification.

```
Counter Example:
   Before Execution:
 2
 3
       This = 0
       Internal = {
 4
 5
           __has_assertion_failure = false
 6
           __has_buf_overflow = false
 7
           __has_overflow = false
           __has_returned = false
 8
           __reverted = false
 9
10
           msg = {
11
             "gas": 0,
             "sender": 0,
12
13
             "value": 0
14
       }
15
       Other = {
16
           __return = 0
17
18
           block = {
             "number": 0,
19
20
             "timestamp": 0
21
22
       }
23
       Address_Map = [
24
25
           "key": "ALL_OTHERS",
26
           "value": {
27
             "contract_name": "QurasTestCoin1",
28
             "balance": 0,
29
             "contract": {
30
               "symbol": "",
               "name": "",
31
32
               "decimals": 0,
33
               "_totalSupply": 0,
34
               "balances": [
```





```
35
                  "key": 128,
36
                  "value": 0
37
                },
38
                 {
39
                  "key": "ALL_OTHERS",
40
                  "value": 1
41
42
43
               ],
               "allowed": [
44
                {
45
46
                  "key": "ALL_OTHERS",
47
                  "value": [
48
                      "key": 128,
49
50
                      "value": 0
51
                    },
52
                      "key": "ALL_OTHERS",
53
                      "value": 1
54
55
                  ]
56
                }
57
58
               ],
               "owner": 0,
59
60
               "newOwner": 0
61
62
           }
         }
63
       ]
64
65
66
   After Execution:
       This = 0
67
68
       Internal = {
69
           __has_assertion_failure = false
           __has_buf_overflow = false
70
71
           __has_overflow = true
           __has_returned = true
72
           __reverted = false
73
74
           msg = {
             "gas": 0,
75
             "sender": 0,
76
77
             "value": 0
           }
78
       }
79
80
       Other = {
           __return = 255
81
82
           block = {
83
             "number": 0,
84
             "timestamp": 0
85
           }
       }
86
87
       Address_Map = [
88
         {
           "key": "ALL_OTHERS",
89
90
           "value": {
91
             "contract_name": "QurasTestCoin1",
92
             "balance": 0,
```





```
"contract": {
 93
 94
                "symbol": "",
                "name": "",
 95
                "decimals": 0,
 96
                "_totalSupply": 0,
 97
                "balances": [
 98
 99
100
                    "key": 128,
                    "value": 0
101
102
                  },
103
104
                    "key": "ALL_OTHERS",
105
                    "value": 1
106
107
                ],
108
                "allowed": [
109
                    "key": "ALL_OTHERS",
110
                    "value": [
111
112
                        "key": 128,
113
                        "value": 0
114
115
                      },
116
117
                        "key": "ALL_OTHERS",
118
                        "value": 1
119
120
                    ]
                  }
121
122
                ],
123
                "owner": 0,
                "newOwner": 0
124
125
126
            }
127
          }
128
```

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

```
606, Sep 2019000 0.39 ms
```

Line 190 in File XQC.sol

```
//@CTK NO_BUF_OVERFLOW
Line 197-199 in File XQC.sol

function totalSupply() public constant returns (uint) {
   return _totalSupply - balances[address(0)];
}
```





Method will not encounter an assertion failure.

```
6 06, Sep 2019○ 0.3 ms
```

Line 191 in File XQC.sol

```
191 //@CTK NO_ASF

Line 197-199 in File XQC.sol

197 function totalSupply() public constant returns (uint) {

198 return _totalSupply - balances[address(0)];

199 }
```

**⊘** The code meets the specification.

## Formal Verification Request 15

```
totalSupply
```

```
6 06, Sep 20190 0.3 ms
```

Line 192-196 in File XQC.sol

```
/*@CTK totalSupply

@tag assume_completion

@pre _totalSupply >= balances[address(0)]

@post (__return) == (_totalSupply - balances[address(0)])

*/

Line 197-199 in File XQC.sol

function totalSupply() public constant returns (uint) {
```

✓ The code meets the specification.

## Formal Verification Request 16

If method completes, integer overflow would not happen.

return \_totalSupply - balances[address(0)];

```
6 06, Sep 2019√ 4.31 ms
```

198

199

Line 205 in File XQC.sol

```
//@CTK NO_OVERFLOW
Line 211-213 in File XQC.sol

function balanceOf(address tokenOwner) public constant returns (uint balance) {
    return balances[tokenOwner];
}
```





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

```
## 06, Sep 2019

• 0.27 ms
```

Line 206 in File XQC.sol

```
206 //@CTK NO_BUF_OVERFLOW
```

Line 211-213 in File XQC.sol

```
function balanceOf(address tokenOwner) public constant returns (uint balance) {
return balances[tokenOwner];
}
```

The code meets the specification.

## Formal Verification Request 18

Method will not encounter an assertion failure.

```
6 06, Sep 20190 0.27 ms
```

Line 207 in File XQC.sol

```
//@CTK NO_ASF
Line 211-213 in File XQC.sol

function balanceOf(address tokenOwner) public constant returns (uint balance) {
   return balances[tokenOwner];
}
```

The code meets the specification.

## Formal Verification Request 19

balanceOf

```
## 06, Sep 2019
```

```
Line 208-210 in File XQC.sol
```

Line 211-213 in File XQC.sol

```
function balanceOf(address tokenOwner) public constant returns (uint balance) {
return balances[tokenOwner];
}
```





If method completes, integer overflow would not happen.

```
6 06, Sep 2019
71.83 ms
```

Line 221 in File XQC.sol

```
//@CTK NO_OVERFLOW
Line 231-236 in File XQC.sol

function transfer(address to, uint tokens) public returns (bool success) {
 balances[msg.sender] = safeSub(balances[msg.sender], tokens);
 balances[to] = safeAdd(balances[to], tokens);
 Transfer(msg.sender, to, tokens);
 return true;
}
```

The code meets the specification.

## Formal Verification Request 21

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

```
6 06, Sep 20190 0.78 ms
```

Line 222 in File XQC.sol

```
222 //@CTK NO_BUF_OVERFLOW
```

Line 231-236 in File XQC.sol

```
function transfer(address to, uint tokens) public returns (bool success) {
   balances[msg.sender] = safeSub(balances[msg.sender], tokens);
   balances[to] = safeAdd(balances[to], tokens);
   Transfer(msg.sender, to, tokens);
   return true;
}
```

The code meets the specification.

## Formal Verification Request 22

transfer

```
60, Sep 201969.13 ms
```

Line 223-230 in File XQC.sol





```
227
          @post (msg.sender != to) -> (_post.balances[msg.sender] == balances[msg.sender]
               - tokens)
          @post (msg.sender == to) -> (__post.balances[to] == balances[to])
228
229
          @post (msg.sender == to) -> (_post.balances[msg.sender] == balances[msg.sender
230
    Line 231-236 in File XQC.sol
231
        function transfer(address to, uint tokens) public returns (bool success) {
232
           balances[msg.sender] = safeSub(balances[msg.sender], tokens);
233
           balances[to] = safeAdd(balances[to], tokens);
234
           Transfer(msg.sender, to, tokens);
           return true;
235
        }
236
```

The code meets the specification.

## Formal Verification Request 23

If method completes, integer overflow would not happen.

```
6, Sep 2019
9.14 ms
```

Line 247 in File XQC.sol

```
Line 253-257 in File XQC.sol

function approve(address spender, uint tokens) public returns (bool success) {
 allowed[msg.sender] [spender] = tokens;
 Approval(msg.sender, spender, tokens);
 return true;
}
```

The code meets the specification.

## Formal Verification Request 24

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

```
6 06, Sep 20190 0.57 ms
```

Line 248 in File XQC.sol

```
//@CTK NO_BUF_OVERFLOW
Line 253-257 in File XQC.sol

function approve(address spender, uint tokens) public returns (bool success) {
   allowed[msg.sender][spender] = tokens;
   Approval(msg.sender, spender, tokens);
   return true;
}
```





Method will not encounter an assertion failure.

```
6 06, Sep 2019
0 0.42 ms
```

Line 249 in File XQC.sol

```
Line 253-257 in File XQC.sol

function approve(address spender, uint tokens) public returns (bool success) {
 allowed[msg.sender] [spender] = tokens;
 Approval(msg.sender, spender, tokens);
 return true;
}
```

The code meets the specification.

## Formal Verification Request 26

approve correctness

```
6 06, Sep 20191 1.6 ms
```

Line 250-252 in File XQC.sol

Line 253-257 in File XQC.sol

```
function approve(address spender, uint tokens) public returns (bool success) {
allowed[msg.sender] [spender] = tokens;
Approval(msg.sender, spender, tokens);
return true;
}
```

The code meets the specification.

## Formal Verification Request 27

If method completes, integer overflow would not happen.

```
6 06, Sep 20196 86.51 ms
```

Line 269 in File XQC.sol

```
269 //@CTK NO_OVERFLOW
```

Line 279-285 in File XQC.sol





```
function transferFrom(address from, address to, uint tokens) public returns (bool
    success) {
    balances[from] = safeSub(balances[from], tokens);
    allowed[from][msg.sender] = safeSub(allowed[from][msg.sender], tokens);
    balances[to] = safeAdd(balances[to], tokens);
    Transfer(from, to, tokens);
    return true;
}
```

The code meets the specification.

## Formal Verification Request 28

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

```
6.25 ms6.25 ms
```

Line 270 in File XQC.sol

```
270 //@CTK NO_BUF_OVERFLOW
```

Line 279-285 in File XQC.sol

```
function transferFrom(address from, address to, uint tokens) public returns (bool success) {
    balances[from] = safeSub(balances[from], tokens);
    allowed[from] [msg.sender] = safeSub(allowed[from] [msg.sender], tokens);
    balances[to] = safeAdd(balances[to], tokens);
    Transfer(from, to, tokens);
    return true;
}
```

The code meets the specification.

## Formal Verification Request 29

transferFrom correctness

```
6, Sep 2019
111.01 ms
```

Line 271-278 in File XQC.sol

```
/*@CTK "transferFrom correctness"

@tag assume_completion
@post tokens <= balances[from] && tokens <= allowed[from][msg.sender]
@post to != from -> __post.balances[from] == balances[from] - tokens
@post to != from -> __post.balances[to] == balances[to] + tokens
@post to == from -> __post.balances[from] == balances[from]
@post to == from -> __post.balances[from] == balances[from]

@post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - tokens

*/
```

Line 279-285 in File XQC.sol





```
279
        function transferFrom(address from, address to, uint tokens) public returns (bool
            success) {
           balances[from] = safeSub(balances[from], tokens);
280
           allowed[from][msg.sender] = safeSub(allowed[from][msg.sender], tokens);
281
282
           balances[to] = safeAdd(balances[to], tokens);
283
           Transfer(from, to, tokens);
284
           return true;
285
        }
```

The code meets the specification.

## Formal Verification Request 30

If method completes, integer overflow would not happen.

```
## 06, Sep 2019
(i) 5.61 ms
```

Line 292 in File XQC.sol

```
//@CTK NO_OVERFLOW
292
    Line 298-300 in File XQC.sol
298
        function allowance(address tokenOwner, address spender) public constant returns (
           uint remaining) {
299
           return allowed[tokenOwner][spender];
300
```

The code meets the specification.

## Formal Verification Request 31

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

```
## 06, Sep 2019
\bullet 0.39 ms
```

```
Line 293 in File XQC.sol
    //@CTK NO_BUF_OVERFLOW
293
    Line 298-300 in File XQC.sol
        function allowance(address tokenOwner, address spender) public constant returns (
298
           uint remaining) {
299
           return allowed[tokenOwner][spender];
300
```





Method will not encounter an assertion failure.

```
6 06, Sep 20190 0.42 ms
```

Line 294 in File XQC.sol

```
//@CTK NO_ASF
Line 298-300 in File XQC.sol

function allowance(address tokenOwner, address spender) public constant returns (
    uint remaining) {
    return allowed[tokenOwner][spender];
}
```

The code meets the specification.

## Formal Verification Request 33

allowance correctness

```
6 06, Sep 20190 0.49 ms
```

Line 295-297 in File XQC.sol

```
/*@CTK "allowance correctness"

@post remaining == allowed[tokenOwner][spender]

*/
```

Line 298-300 in File XQC.sol





## Source Code with CertiK Labels

File XQC.sol

```
1 pragma solidity ^0.4.18;
 2
 3 // -----
4 // 'XQC1' token contract
5 //
6 // Deployed to : 0x6e0004D2639E9D45bCD3Af0498C379dBAb598E3E
7 // Symbol : XQC1
8 // Name : Quras Test Coin1
9 // Total supply: 888888888
10 // Decimals : 8
11 //
12 // Enjoy.
13 //
14 // (c) by Moritz Neto with BokkyPooBah / Bok Consulting Pty Ltd Au 2017. The MIT
       Licence.
15
16
17
18 // -----
19 // Safe maths
20 // -----
21 contract SafeMath {
22
      /*@CTK "SafeMath add"
23
        @tag spec
24
        @tag is_pure
25
        26
        @post !__reverted -> c == a + b
27
        @post !__reverted -> !__has_overflow
        @post !__reverted -> !__has_assertion_failure
28
        @post !(__has_buf_overflow)
29
30
31
      function safeAdd(uint a, uint b) public pure returns (uint c) {
32
          c = a + b;
33
          require(c >= a);
34
      /*@CTK "SafeMath sub"
35
36
        Otag spec
37
        @tag is_pure
38
        @post (b > a) == __reverted
39
        @post !\_reverted \rightarrow c == a - b
40
        @post !__reverted -> !__has_overflow
41
        @post !__reverted -> !__has_assertion_failure
42
        @post !(__has_buf_overflow)
43
      function safeSub(uint a, uint b) public pure returns (uint c) {
44
45
          require(b <= a);</pre>
          c = a - b;
46
47
      }
48
      /*@CTK "SafeMath mul zero"
49
        @tag spec
50
        @tag is_pure
        @pre (a == 0)
51
52
        @post c == 0
```





```
/*@CTK "SafeMath mul nonzero"
54
55
         @tag spec
56
         @tag is_pure
         @pre (a != 0)
57
58
         @post (a * b / a != b) == __reverted
         @post !__reverted -> c == a * b
59
         @post !__reverted -> !__has_overflow
60
         @post !__reverted -> !__has_assertion_failure
61
        @post !(__has_buf_overflow)
62
63
       */
64
       function safeMul(uint a, uint b) public pure returns (uint c) {
65
          require(a == 0 || c / a == b);
66
       }
67
 68
       /*@CTK "SafeMath div"
69
        @tag spec
70
         @tag is_pure
         @post (b == 0) == __reverted
71
72
         @post !__reverted -> c == a / b
         @post !__reverted -> !__has_overflow
73
74
         @post !__reverted -> !__has_assertion_failure
         @post !(__has_buf_overflow)
75
76
77
       function safeDiv(uint a, uint b) public pure returns (uint c) {
78
          require(b > 0);
79
          c = a / b;
       }
80
81 }
82
83
84 // -----
85 // ERC Token Standard #20 Interface
86 // https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
87 // -----
88
   contract ERC20Interface {
89
       function totalSupply() public constant returns (uint);
90
       function balanceOf(address tokenOwner) public constant returns (uint balance);
       function allowance(address tokenOwner, address spender) public constant returns (
91
           uint remaining);
92
       function transfer(address to, uint tokens) public returns (bool success);
93
       function approve(address spender, uint tokens) public returns (bool success);
94
       function transferFrom(address from, address to, uint tokens) public returns (bool
           success);
95
       event Transfer(address indexed from, address indexed to, uint tokens);
96
       event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
97
98 }
99
100
101 // -----
102 // Contract function to receive approval and execute function in one call
103 //
104 // Borrowed from MiniMeToken
105 // -----
106 contract ApproveAndCallFallBack {
107
       function receiveApproval(address from, uint256 tokens, address token, bytes data)
108 }
```





```
109
110
111 // -----
112 // Owned contract
113 // -----
114
   contract Owned {
115
        address public owner;
116
        address public newOwner;
117
118
        event OwnershipTransferred(address indexed _from, address indexed _to);
119
120
        function Owned() public {
121
           owner = msg.sender;
122
123
124
        modifier onlyOwner {
125
           require(msg.sender == owner);
126
127
        }
128
129
        /*@CTK transferOwnership
130
         @tag assume_completion
131
         @pre msg.sender == owner
132
         @post __post.newOwner == _newOwner
133
134
        function transferOwnership(address _newOwner) public onlyOwner {
135
           newOwner = _newOwner;
136
137
        /*@CTK acceptOwnership
138
         @tag assume_completion
139
         @pre msg.sender == newOwner
140
         @post __post.owner == newOwner
         @post __post.newOwner == address(0)
141
142
         */
143
        function acceptOwnership() public {
           require(msg.sender == newOwner);
144
145
           OwnershipTransferred(owner, newOwner);
146
           owner = newOwner;
147
           newOwner = address(0);
148
        }
149 }
150
151
152
153 // ERC20 Token, with the addition of symbol, name and decimals and assisted
154 // token transfers
155 // --
156 contract QurasTestCoin1 is ERC20Interface, Owned, SafeMath {
157
        string public symbol;
158
        string public name;
159
        uint8 public decimals;
160
        uint public _totalSupply;
161
162
        mapping(address => uint) balances;
163
        mapping(address => mapping(address => uint)) allowed;
164
165
166
```





```
167
       // Constructor
168
169
       //@CTK NO_OVERFLOW
170
       //@CTK NO_BUF_OVERFLOW
171
       //@CTK NO_ASF
172
       /*@CTK QurasTestCoin1
173
         @tag assume_completion
174
         @post __post._totalSupply == 88888888800000000
175
176
        function QurasTestCoin1() public {
177
           symbol = "XQC1";
           name = "Quras Test Coin1";
178
179
           decimals = 8;
           _totalSupply = 8888888800000000;
180
181
           balances[0x6e0004D2639E9D45bCD3Af0498C379dBAb598E3E] = totalSupply;
182
           Transfer(address(0), 0x6e0004D2639E9D45bCD3Af0498C379dBAb598E3E, _totalSupply);
183
        }
184
185
       // -----
186
187
       // Total supply
        // -----
188
189
       //@CTK FAIL NO_OVERFLOW
190
       //@CTK NO_BUF_OVERFLOW
191
       //@CTK NO_ASF
192
        /*@CTK totalSupply
193
           @tag assume_completion
194
           @pre _totalSupply >= balances[address(0)]
195
           @post (__return) == (_totalSupply - balances[address(0)])
196
197
        function totalSupply() public constant returns (uint) {
198
           return _totalSupply - balances[address(0)];
        }
199
200
201
202
203
        // Get the token balance for account tokenOwner
204
        // -----
205
       //@CTK NO_OVERFLOW
206
       //@CTK NO_BUF_OVERFLOW
207
       //@CTK NO_ASF
208
       /*@CTK balanceOf
209
         @post balance == __post.balances[tokenOwner]
210
211
        function balanceOf(address tokenOwner) public constant returns (uint balance) {
212
           return balances[tokenOwner];
213
        }
214
215
216
        // Transfer the balance from token owner's account to to account
217
218
       // - Owner's account must have sufficient balance to transfer
219
       // - 0 value transfers are allowed
220
221
       //@CTK NO_OVERFLOW
222
       //@CTK NO_BUF_OVERFLOW
223
       /*@CTK transfer
224
      @tag assume_completion
```





```
225
          Opre tokens <= balances[msg.sender]</pre>
226
          @post (msg.sender != to) -> (__post.balances[to] == balances[to] + tokens)
227
          @post (msg.sender != to) -> (_post.balances[msg.sender] == balances[msg.sender]
               - tokens)
228
          @post (msg.sender == to) -> (__post.balances[to] == balances[to])
229
          @post (msg.sender == to) -> (__post.balances[msg.sender] == balances[msg.sender
230
        function transfer(address to, uint tokens) public returns (bool success) {
231
232
           balances[msg.sender] = safeSub(balances[msg.sender], tokens);
233
           balances[to] = safeAdd(balances[to], tokens);
234
           Transfer(msg.sender, to, tokens);
235
           return true;
236
        }
237
238
239
        // Token owner can approve for spender to transferFrom(...) tokens
240
241
        // from the token owner's account
242
        //
243
        // https://github.com/ethereum/EIPs/blob/master/EIPs/eip-20-token-standard.md
244
        // recommends that there are no checks for the approval double-spend attack
245
        // as this should be implemented in user interfaces
246
        //@CTK NO_OVERFLOW
247
248
        //@CTK NO_BUF_OVERFLOW
249
        //@CTK NO ASF
250
        /*@CTK "approve correctness"
251
         @post __post.allowed[msg.sender][spender] == tokens
252
253
        function approve(address spender, uint tokens) public returns (bool success) {
254
           allowed[msg.sender][spender] = tokens;
255
           Approval(msg.sender, spender, tokens);
256
           return true;
        }
257
258
259
260
261
        // Transfer tokens from the from account to the to account
262
263
        // The calling account must already have sufficient tokens approve(...)-d
264
        // for spending from the from account and
265
        // - From account must have sufficient balance to transfer
266
        // - Spender must have sufficient allowance to transfer
267
        // - 0 value transfers are allowed
268
        //@CTK NO_OVERFLOW
269
        //@CTK NO_BUF_OVERFLOW
270
271
        /*@CTK "transferFrom correctness"
272
         Otag assume completion
273
         @post tokens <= balances[from] && tokens <= allowed[from] [msg.sender]</pre>
         @post to != from -> __post.balances[from] == balances[from] - tokens
274
         @post to != from -> __post.balances[to] == balances[to] + tokens
275
276
         @post to == from -> __post.balances[from] == balances[from]
277
         @post __post.allowed[from] [msg.sender] == allowed[from] [msg.sender] - tokens
278
279
        function transferFrom(address from, address to, uint tokens) public returns (bool
            success) {
```





```
280
          balances[from] = safeSub(balances[from], tokens);
281
          allowed[from][msg.sender] = safeSub(allowed[from][msg.sender], tokens);
          balances[to] = safeAdd(balances[to], tokens);
282
283
          Transfer(from, to, tokens);
          return true;
284
285
       }
286
287
288
       // Returns the amount of tokens approved by the owner that can be
289
290
       // transferred to the spender's account
291
       // -----
292
       //@CTK NO_OVERFLOW
293
       //@CTK NO_BUF_OVERFLOW
       //@CTK NO ASF
294
295
       /*@CTK "allowance correctness"
296
         @post remaining == allowed[tokenOwner][spender]
297
298
       function allowance(address tokenOwner, address spender) public constant returns (
          uint remaining) {
299
          return allowed[tokenOwner][spender];
       }
300
301
302
303
       // -----
304
       // Token owner can approve for spender to transferFrom(...) tokens
305
       // from the token owner's account. The spender contract function
       // receiveApproval(...) is then executed
306
       // -----
307
       function approveAndCall(address spender, uint tokens, bytes data) public returns (
308
          bool success) {
309
          allowed[msg.sender][spender] = tokens;
310
          Approval(msg.sender, spender, tokens);
311
          ApproveAndCallFallBack(spender).receiveApproval(msg.sender, tokens, this, data)
312
          return true;
       }
313
314
315
316
       // Don't accept ETH
317
318
       // -----
319
       function () public payable {
320
          revert();
321
       }
322
323
324
       // Owner can transfer out any accidentally sent ERC20 tokens
325
326
       // -----
       function transferAnyERC20Token(address tokenAddress, uint tokens) public onlyOwner
327
           returns (bool success) {
328
          return ERC20Interface(tokenAddress).transfer(owner, tokens);
329
       }
330 }
```