# CertiK Audit Report For Quras

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

px

# Contents

## Disclaimer

## About CertiK

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

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

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

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

## 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**

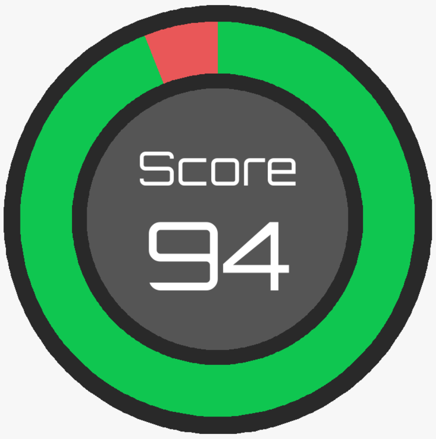
The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerabilies, but no concern found yet.

## Testing Summary

**PASS**

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

*Sep 06, 2019*



### 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. | 1 | SWC-102<br>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**

- `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

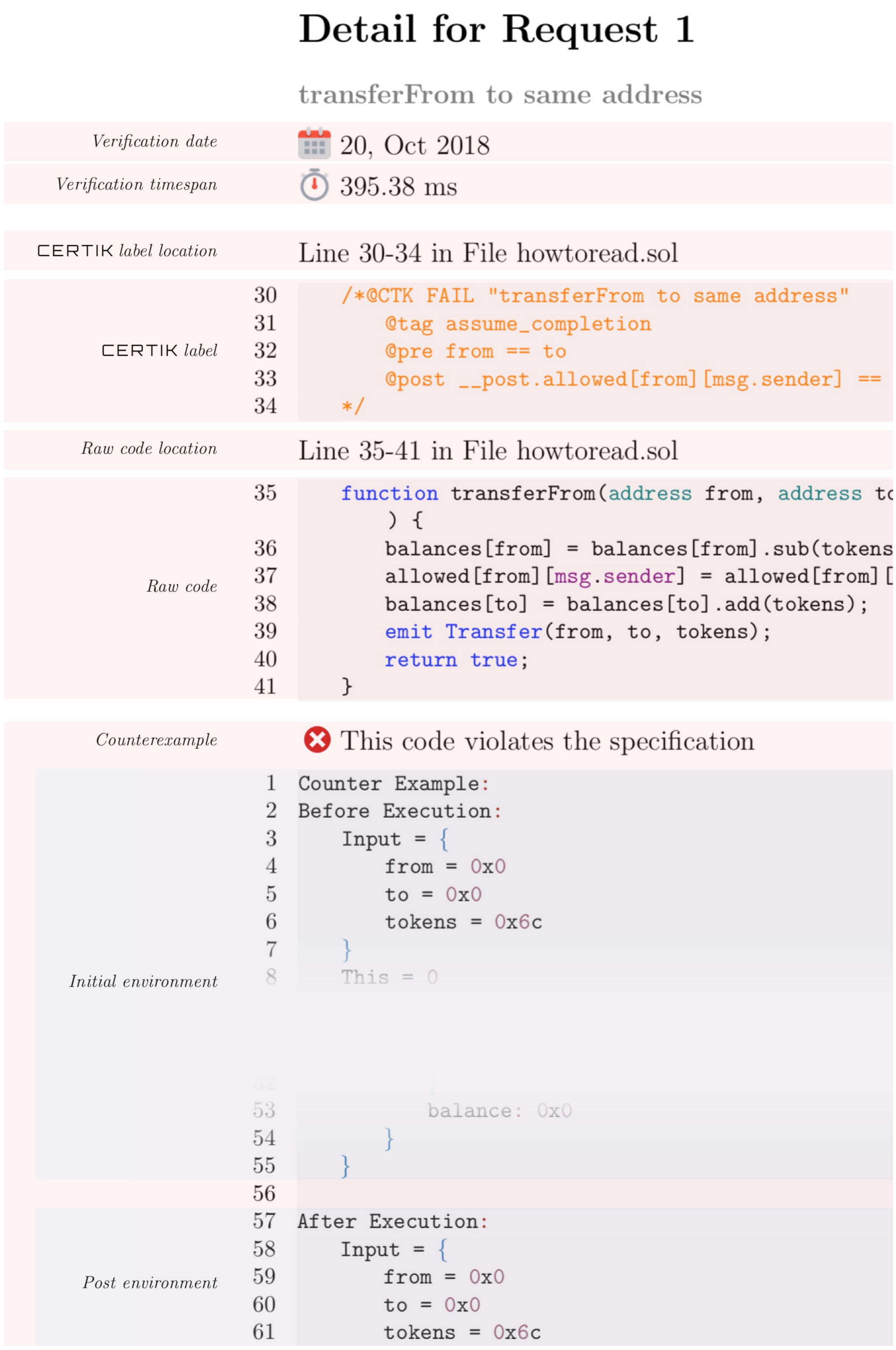
Line 1 in File XQC.sol

```
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, EventStructWro
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, EventStructWro
Data 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     }
56
```

*Post environment*

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

## Formal Verification Request 1

### SafeMath add

📅 06, Sep 2019
⏱ 13.65 ms

Line 22-30 in File XQC.sol

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

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
⏱ 10.53 ms

Line 35-43 in File XQC.sol

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

Line 44-47 in File XQC.sol

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

✓ The code meets the specification.

## Formal Verification Request 3

### SafeMath mul zero

📅 06, Sep 2019
⏱ 11.64 ms

Line 48-53 in File XQC.sol

```
/*@CTK "SafeMath mul zero"
  @tag spec
  @tag is_pure
  @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

📅 06, Sep 2019
⏱ 6.32 ms

Line 54-63 in File XQC.sol

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

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 5

### SafeMath div

📅 06, Sep 2019
⏱ 11.82 ms

Line 68-76 in File XQC.sol

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

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
      @tag assume_completion
      @pre msg.sender == owner
      @post __post.newOwner == _newOwner
     */
```

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
⏱ 13.67 ms

Line 137-142 in File XQC.sol

```
    /*@CTK acceptOwnership
      @tag assume_completion
      @pre msg.sender == newOwner
      @post __post.owner == newOwner
      @post __post.newOwner == address(0)
     */
```

Line 143-148 in File XQC.sol

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

✓ The code meets the specification.

## Formal Verification Request 8

**If method completes, integer overflow would not happen.**

06, Sep 2019
18.11 ms

Line 169 in File XQC.sol

```
    //@CTK NO_OVERFLOW
```

Line 176-183 in File XQC.sol

```
    function QurasTestCoin1() public {
        symbol = "XQC1";
        name = "Quras Test Coin1";
        decimals = 8;
        _totalSupply = 888888888800000000;
        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.**

06, Sep 2019
0.34 ms

Line 170 in File XQC.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 176-183 in File XQC.sol

```
    function QurasTestCoin1() public {
        symbol = "XQC1";
        name = "Quras Test Coin1";
        decimals = 8;
        _totalSupply = 8888888800000000;
        balances[0x6e0004D2639E9D45bCD3Af0498C379dBAb598E3E] = _totalSupply;
        Transfer(address(0), 0x6e0004D2639E9D45bCD3Af0498C379dBAb598E3E, _totalSupply);
    }
```

✓ The code meets the specification.

## Formal Verification Request 10

**Method will not encounter an assertion failure.**

📅 06, Sep 2019
⏱ 0.34 ms

Line 171 in File XQC.sol

```
    //@CTK NO_ASF
```

Line 176-183 in File XQC.sol

```
    function QurasTestCoin1() public {
        symbol = "XQC1";
        name = "Quras Test Coin1";
        decimals = 8;
        _totalSupply = 8888888800000000;
        balances[0x6e0004D2639E9D45bCD3Af0498C379dBAb598E3E] = _totalSupply;
        Transfer(address(0), 0x6e0004D2639E9D45bCD3Af0498C379dBAb598E3E, _totalSupply);
    }
```

✓ The code meets the specification.

## Formal Verification Request 11

**QurasTestCoin1**

📅 06, Sep 2019
⏱ 0.97 ms

Line 172-175 in File XQC.sol

```
    /*@CTK QurasTestCoin1
      @tag assume_completion
      @post __post._totalSupply == 8888888800000000
     */
```

Line 176-183 in File XQC.sol

```
    function QurasTestCoin1() public {
        symbol = "XQC1";
        name = "Quras Test Coin1";
        decimals = 8;
        _totalSupply = 8888888800000000;
        balances[0x6e0004D2639E9D45bCD3Af0498C379dBAb598E3E] = _totalSupply;
        Transfer(address(0), 0x6e0004D2639E9D45bCD3Af0498C379dBAb598E3E, _totalSupply);
    }
```

✓ 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

```
    //@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:
    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 = 0
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": {
          "contract_name": "QurasTestCoin1",
          "balance": 0,
          "contract": {
            "symbol": "",
            "name": "",
            "decimals": 0,
            "_totalSupply": 0,
            "balances": [
            {
              "key": 128,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 1
            }
          ],
          "allowed": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": 128,
                  "value": 0
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 1
                }
              ]
            }
          ],
          "owner": 0,
          "newOwner": 0
        }
      }
    }
  ]

After Execution:
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = true
        __has_returned = true
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = 255
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": {
          "contract_name": "QurasTestCoin1",
          "balance": 0,
      "contract": {
        "symbol": "",
        "name": "",
        "decimals": 0,
        "_totalSupply": 0,
        "balances": [
          {
            "key": 128,
            "value": 0
          },
          {
            "key": "ALL_OTHERS",
            "value": 1
          }
        ],
        "allowed": [
          {
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": 128,
                "value": 0
              },
              {
                "key": "ALL_OTHERS",
                "value": 1
              }
            ]
          }
        ],
        "owner": 0,
        "newOwner": 0
      }
    }
  }
]
```

## Formal Verification Request 13

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

📅 06, Sep 2019
⏱ 0.39 ms

Line 190 in File XQC.sol

```
190  //@CTK NO_BUF_OVERFLOW
```

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 14

**Method will not encounter an assertion failure.**

06, Sep 2019
0.3 ms

Line 191 in File XQC.sol

```
//@CTK NO_ASF
```

Line 197-199 in File XQC.sol

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

✓ The code meets the specification.

## Formal Verification Request 15

**totalSupply**

06, Sep 2019
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) {
    return _totalSupply - balances[address(0)];
}
```

✓ The code meets the specification.

## Formal Verification Request 16

**If method completes, integer overflow would not happen.**

06, Sep 2019
4.31 ms

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];
}
```

✓ The code meets the specification.

## Formal Verification Request 17

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

06, Sep 2019
0.27 ms

Line 206 in File XQC.sol

```
    //@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.

06, Sep 2019
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
0.28 ms

Line 208-210 in File XQC.sol

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

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 20

**If method completes, integer overflow would not happen.**

📅 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.**

📅 06, Sep 2019
⏱ 0.78 ms

Line 222 in File XQC.sol

```
//@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**

📅 06, Sep 2019
⏱ 69.13 ms

Line 223-230 in File XQC.sol

```
/*@CTK transfer
  @tag assume_completion
  @pre tokens <= balances[msg.sender]
  @post (msg.sender != to) -> (__post.balances[to] == balances[to] + tokens)
```

```
      @post (msg.sender != to) -> (__post.balances[msg.sender] == balances[msg.sender]
         - tokens)
      @post (msg.sender == to) -> (__post.balances[to] == balances[to])
      @post (msg.sender == to) -> (__post.balances[msg.sender] == balances[msg.sender
        ])
     */
```

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 23

**If method completes, integer overflow would not happen.**

📅 06, Sep 2019
⏱ 9.14 ms

Line 247 in File XQC.sol

```
   //@CTK NO_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;
   }
```

✅ The code meets the specification.

## Formal Verification Request 24

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

📅 06, Sep 2019
⏱ 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;
   }
```

✅ The code meets the specification.

## Formal Verification Request 25

### Method will not encounter an assertion failure.

📅 06, Sep 2019
⏱ 0.42 ms

Line 249 in File XQC.sol

```
    //@CTK NO_ASF
```

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

📅 06, Sep 2019
⏱ 1.6 ms

Line 250-252 in File XQC.sol

```
    /*@CTK "approve correctness"
      @post __post.allowed[msg.sender][spender] == tokens
     */
```

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.

📅 06, Sep 2019
⏱ 86.51 ms

Line 269 in File XQC.sol

```
    //@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.

📅 06, Sep 2019
⏱ 6.25 ms

Line 270 in File XQC.sol

```
    //@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

📅 06, 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 __post.allowed[from][msg.sender] == allowed[from][msg.sender] - tokens
     */
```

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 30

**If method completes, integer overflow would not happen.**

06, Sep 2019
5.61 ms

Line 292 in File XQC.sol

```
//@CTK NO_OVERFLOW
```

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 31

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

06, Sep 2019
0.39 ms

Line 293 in File XQC.sol

```
//@CTK NO_BUF_OVERFLOW
```

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 32

### Method will not encounter an assertion failure.

📅 06, Sep 2019
⏱ 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

📅 06, Sep 2019
⏱ 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

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

✓ The code meets the specification.

## Source Code with CertiK Labels

File XQC.sol

```
pragma solidity ^0.4.18;

// ----------------------------------------------------------------------------
// 'XQC1' token contract
//
// Deployed to : 0x6e0004D2639E9D45bCD3Af0498C379dBAb598E3E
// Symbol      : XQC1
// Name        : Quras Test Coin1
// Total supply: 888888888
// Decimals    : 8
//
// Enjoy.
//
// (c) by Moritz Neto with BokkyPooBah / Bok Consulting Pty Ltd Au 2017. The MIT
    Licence.
// ----------------------------------------------------------------------------


// ----------------------------------------------------------------------------
// Safe maths
// ----------------------------------------------------------------------------
contract SafeMath {
    /*@CTK "SafeMath add"
      @tag spec
      @tag is_pure
      @post (a + b < a || a + b < b) == __reverted
      @post !__reverted -> c == a + b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
    */
    function safeAdd(uint a, uint b) public pure returns (uint c) {
        c = a + b;
        require(c >= a);
    }
    /*@CTK "SafeMath sub"
      @tag spec
      @tag is_pure
      @post (b > a) == __reverted
      @post !__reverted -> c == a - b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
    */
    function safeSub(uint a, uint b) public pure returns (uint c) {
        require(b <= a);
        c = a - b;
    }
    /*@CTK "SafeMath mul zero"
      @tag spec
      @tag is_pure
      @pre (a == 0)
      @post c == 0
    */
    /*@CTK "SafeMath mul nonzero"
      @tag spec
      @tag is_pure
      @pre (a != 0)
      @post (a * b / a != b) == __reverted
      @post !__reverted -> c == a * b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
    */
    function safeMul(uint a, uint b) public pure returns (uint c) {
        c = a * b;
        require(a == 0 || c / a == b);
    }
    /*@CTK "SafeMath div"
      @tag spec
      @tag is_pure
      @post (b == 0) == __reverted
      @post !__reverted -> c == a / b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
    */
    function safeDiv(uint a, uint b) public pure returns (uint c) {
        require(b > 0);
        c = a / b;
    }
}


// ----------------------------------------------------------------------------
// ERC Token Standard #20 Interface
// https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
// ----------------------------------------------------------------------------
contract ERC20Interface {
    function totalSupply() public constant returns (uint);
    function balanceOf(address tokenOwner) public constant returns (uint balance);
    function allowance(address tokenOwner, address spender) public constant returns (
        uint remaining);
    function transfer(address to, uint tokens) public returns (bool success);
    function approve(address spender, uint tokens) public returns (bool success);
    function transferFrom(address from, address to, uint tokens) public returns (bool
        success);

    event Transfer(address indexed from, address indexed to, uint tokens);
    event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
}


// ----------------------------------------------------------------------------
// Contract function to receive approval and execute function in one call
//
// Borrowed from MiniMeToken
// ----------------------------------------------------------------------------
contract ApproveAndCallFallBack {
    function receiveApproval(address from, uint256 tokens, address token, bytes data)
        public;
}
```

```


// ----------------------------------------------------------------------------
// Owned contract
// ----------------------------------------------------------------------------
contract Owned {
    address public owner;
    address public newOwner;

    event OwnershipTransferred(address indexed _from, address indexed _to);

    function Owned() public {
        owner = msg.sender;
    }

    modifier onlyOwner {
        require(msg.sender == owner);
        _;
    }

    /*@CTK transferOwnership
      @tag assume_completion
      @pre msg.sender == owner
      @post __post.newOwner == _newOwner
     */
    function transferOwnership(address _newOwner) public onlyOwner {
        newOwner = _newOwner;
    }
    /*@CTK acceptOwnership
      @tag assume_completion
      @pre msg.sender == newOwner
      @post __post.owner == newOwner
      @post __post.newOwner == address(0)
     */
    function acceptOwnership() public {
        require(msg.sender == newOwner);
        OwnershipTransferred(owner, newOwner);
        owner = newOwner;
        newOwner = address(0);
    }
}


// ----------------------------------------------------------------------------
// ERC20 Token, with the addition of symbol, name and decimals and assisted
// token transfers
// ----------------------------------------------------------------------------
contract QurasTestCoin1 is ERC20Interface, Owned, SafeMath {
    string public symbol;
    string public name;
    uint8 public decimals;
    uint public _totalSupply;

    mapping(address => uint) balances;
    mapping(address => mapping(address => uint)) allowed;


    // ----------------------------------------------------------------------------
    // Constructor
    // ------------------------------------------------------------------------
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK QurasTestCoin1
      @tag assume_completion
      @post __post._totalSupply == 8888888800000000
     */
    function QurasTestCoin1() public {
        symbol = "XQC1";
        name = "Quras Test Coin1";
        decimals = 8;
        _totalSupply = 8888888800000000;
        balances[0x6e0004D2639E9D45bCD3Af0498C379dBAb598E3E] = _totalSupply;
        Transfer(address(0), 0x6e0004D2639E9D45bCD3Af0498C379dBAb598E3E, _totalSupply);
    }


    // ------------------------------------------------------------------------
    // Total supply
    // ------------------------------------------------------------------------
    //@CTK FAIL NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK totalSupply
        @tag assume_completion
        @pre _totalSupply >= balances[address(0)]
        @post (__return) == (_totalSupply - balances[address(0)])
      */
    function totalSupply() public constant returns (uint) {
        return _totalSupply - balances[address(0)];
    }


    // ------------------------------------------------------------------------
    // Get the token balance for account tokenOwner
    // ------------------------------------------------------------------------
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK balanceOf
      @post balance == __post.balances[tokenOwner]
      */
    function balanceOf(address tokenOwner) public constant returns (uint balance) {
        return balances[tokenOwner];
    }


    // ------------------------------------------------------------------------
    // Transfer the balance from token owner's account to to account
    // - Owner's account must have sufficient balance to transfer
    // - 0 value transfers are allowed
    // ------------------------------------------------------------------------
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@CTK transfer
       @tag assume_completion
      @pre tokens <= balances[msg.sender]
      @post (msg.sender != to) -> (__post.balances[to] == balances[to] + tokens)
      @post (msg.sender != to) -> (__post.balances[msg.sender] == balances[msg.sender]
          - tokens)
      @post (msg.sender == to) -> (__post.balances[to] == balances[to])
      @post (msg.sender == to) -> (__post.balances[msg.sender] == balances[msg.sender
        ])
     */
    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;
    }


    // ------------------------------------------------------------------------
    // Token owner can approve for spender to transferFrom(...) tokens
    // from the token owner's account
    //
    // https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
    // recommends that there are no checks for the approval double-spend attack
    // as this should be implemented in user interfaces
    // ------------------------------------------------------------------------
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "approve correctness"
      @post __post.allowed[msg.sender][spender] == tokens
     */
    function approve(address spender, uint tokens) public returns (bool success) {
        allowed[msg.sender][spender] = tokens;
        Approval(msg.sender, spender, tokens);
        return true;
    }


    // ------------------------------------------------------------------------
    // Transfer tokens from the from account to the to account
    //
    // The calling account must already have sufficient tokens approve(...)-d
    // for spending from the from account and
    // - From account must have sufficient balance to transfer
    // - Spender must have sufficient allowance to transfer
    // - 0 value transfers are allowed
    // ------------------------------------------------------------------------
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@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 __post.allowed[from][msg.sender] == allowed[from][msg.sender] - tokens
     */
    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;
    }


    // ------------------------------------------------------------------------
    // Returns the amount of tokens approved by the owner that can be
    // transferred to the spender's account
    // ------------------------------------------------------------------------
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "allowance correctness"
      @post remaining == allowed[tokenOwner][spender]
     */
    function allowance(address tokenOwner, address spender) public constant returns (
        uint remaining) {
        return allowed[tokenOwner][spender];
    }


    // ------------------------------------------------------------------------
    // Token owner can approve for spender to transferFrom(...) tokens
    // from the token owner's account. The spender contract function
    // receiveApproval(...) is then executed
    // ------------------------------------------------------------------------
    function approveAndCall(address spender, uint tokens, bytes data) public returns (
        bool success) {
        allowed[msg.sender][spender] = tokens;
        Approval(msg.sender, spender, tokens);
        ApproveAndCallFallBack(spender).receiveApproval(msg.sender, tokens, this, data)
            ;
        return true;
    }


    // ------------------------------------------------------------------------
    // Don't accept ETH
    // ------------------------------------------------------------------------
    function () public payable {
        revert();
    }


    // ------------------------------------------------------------------------
    // Owner can transfer out any accidentally sent ERC20 tokens
    // ------------------------------------------------------------------------
    function transferAnyERC20Token(address tokenAddress, uint tokens) public onlyOwner
         returns (bool success) {
        return ERC20Interface(tokenAddress).transfer(owner, tokens);
    }
}
```