CertiK Verification Report For X-Block

Request Date: 2019-01-20
Revision Date: 2019-01-23

Disclaimer

PASS

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

Jan 23, 2019

Summary

This is the report for smart contract verification service requestd by X-Block. The goal of the audition is to guarantee that verified smart contracts are robust enough to avoid potentially unexpected loopholes.
The result of this report is only a reflection of the source code that was determined in this scope, and of the source code at the audit time.

Type of Issues

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

Title	Description	Issues	SWC ID
Integer Overflow and Underflow	An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum size of a type.	0	SWC-101
Function incorrectness	Function implementation does not meet the specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker is able to write to arbitrary storage locations of a contract if array of out bound happens	0	SWC-124
Reentrancy	A malicious contract can call back into the calling contract before the first invocation of the function is finished.	0	SWC-107
Transaction Order Dependence	A race condition vulnerability occurs when code depends on the order of the transactions submitted to it.	0	SWC-114
Timestamp Dependence	Timestamp can be influenced by minors to some degree.	0	SWC-116

Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0	SWC-102 SWC-103
Insecure Randomness	Block attributes are insecure to generate random numbers, as they can be influenced by minors to some degree.	0	SWC-120
"tx.origin" for authorization	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115
Delegatecall to Untrusted Callee	Calling into untrusted contracts is very dangerous, the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	Functions are public by default. A malicious user is able to make unauthorized or unintended state changes if a developer forgot to set the visibility.	0	SWC-100
Uninitialized variables	Uninitialized local storage variables can point to other unexpected storage variables in the contract.	0	SWC-109
Assertion Failure	The assert() function is meant to assert invariants. Properly functioning code should never reach a failing assert statement.	0	SWC-110
Deprecated Solidity Features	Several functions and operators in Solidity are deprecated and should not be used as best practice.	0	SWC-111
Unused variables	Unused variables reduce code quality	0	

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

Deprecated Syntax

Use `constructor` keyword to replace `Ownable` and `XBlockToken` as the function name of the constructors.

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

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

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

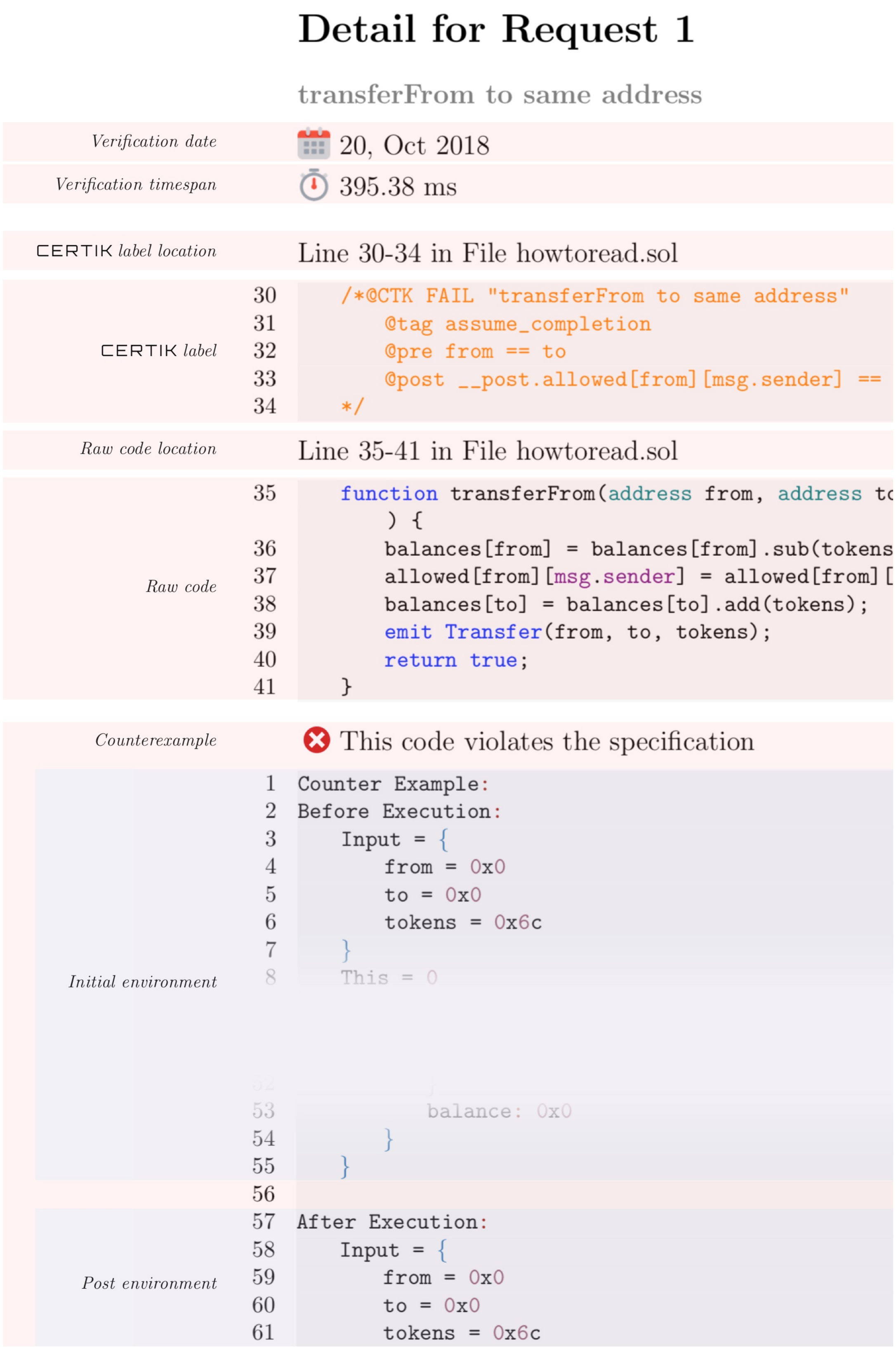
Source Code with CertiK Labels

File xblock.sol

```
pragma solidity ^0.4.13;

library SafeMath {
  /*@CTK "SafeMath mul"
      @post (a > 0) && (((a * b) / a) != b) -> __reverted
      @post __reverted -> (a > 0) && (((a * b) / a) != b)
      @post !__reverted -> __return == a * b
      @post !__reverted == !__has_overflow
  */
  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
    if (a == 0) {
      return 0;
    }
    uint256 c = a * b;
    assert(c / a == b);
    return c;
  }

  /*@CTK "SafeMath div"
      @post b != 0 -> !__reverted
      @post !__reverted -> __return == a / b
      @post !__reverted -> !__has_overflow
  */
  function div(uint256 a, uint256 b) internal pure returns (uint256) {
    // assert(b > 0); // Solidity automatically throws when dividing by 0
    uint256 c = a / b;
    // assert(a == b * c + a % b); // There is no case in which this doesn't hold
    return c;
  }

  /*@CTK "SafeMath sub"
      @post (a < b) == __reverted
      @post !__reverted -> __return == a - b
      @post !__reverted -> !__has_overflow
  */
  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    assert(b <= a);
    return a - b;
  }

  /*@CTK "SafeMath add"
      @post (a + b < a || a + b < b) == __reverted
      @post !__reverted -> __return == a + b
      @post !__reverted -> !__has_overflow
  */
  function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    assert(c >= a);
    return c;
  }
}

contract Ownable {
  address public owner;

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

  /**
   * @dev The Ownable constructor sets the original ‘owner‘ of the contract to the
       sender
   */
  /*@CTK Ownable
    @post __post.owner == msg.sender
   */
  function Ownable() public {
    owner = msg.sender;
  }


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


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

}

contract Pausable is Ownable {
  event Pause();
  event Unpause();

  bool public paused = false;

  /**
   * @dev Modifier to make a function callable only when the contract is not paused.
   */
  modifier whenNotPaused() {
    require(!paused);
    _;
  }

  /**
   * @dev Modifier to make a function callable only when the contract is paused.
   */
  modifier whenPaused() {
    require(paused);
    _;
  }

  /**
   * @dev called by the owner to pause, triggers stopped state
   */
  /*@CTK pause
    @tag assume_completion
    @post paused == false
    @post owner == msg.sender
    @post __post.paused == true
   */
  function pause() onlyOwner whenNotPaused public {
    paused = true;
    emit  Pause();
  }

  /**
   * @dev called by the owner to unpause, returns to normal state
   */
  /*@CTK unpause
    @tag assume_completion
    @post paused == true
    @post owner == msg.sender
    @post __post.paused == false
   */
  function unpause() onlyOwner whenPaused public {
    paused = false;
    emit Unpause();
  }
}

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

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

contract BasicToken is ERC20Basic {
  using SafeMath for uint256;
  mapping(address => uint256) balances;

  /**
   * @dev transfer token for a specified address
   * @param _to The address to transfer to.
   * @param _value The amount to be transferred.
   */
  /*@CTK transfer
    @tag assume_completion
    @pre _to != msg.sender
    @post _to != address(0)
    @post _value <= balances[msg.sender]
    @post __post.balances[msg.sender] == balances[msg.sender] - _value
    @post __post.balances[_to] == balances[_to] + _value
   */
  function transfer(address _to, uint256 _value) public returns (bool) {
    require(_to != address(0));
    require(_value <= balances[msg.sender]);

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

  /**
   * @dev Gets the balance of the specified address.
   * @param _owner The address to query the the balance of.
   * @return An uint256 representing the amount owned by the passed address.
   */
  /*@CTK balanceOf
    @post balance == balances[_owner]
   */
  function balanceOf(address _owner) public view returns (uint256 balance) {
    return balances[_owner];
  }

}

contract StandardToken is ERC20, BasicToken {
  mapping (address => mapping (address => uint256)) internal allowed;

  /**
   * @dev Transfer tokens from one address to another
   * @param _from address The address which you want to send tokens from
   * @param _to address The address which you want to transfer to
   * @param _value uint256 the amount of tokens to be transferred
   */
  /*@CTK transferFrom
    @tag assume_completion
    @pre _from != _to
    @post _to != address(0)
    @post _value <= balances[_from]
    @post _value <= allowed[_from][msg.sender]
    @post __post.balances[_from] == balances[_from] - _value
    @post __post.balances[_to] == balances[_to] + _value
   */
  function transferFrom(address _from, address _to, uint256 _value) public returns (
      bool) {
    require(_to != address(0));
    require(_value <= balances[_from]);
    require(_value <= allowed[_from][msg.sender]);

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

  /**
   * @dev Approve the passed address to spend the specified amount of tokens on behalf
       of msg.sender.
   * @param _spender The address which will spend the funds.
   * @param _value The amount of tokens to be spent.
   */
  /*@CTK approve
    @post __post.allowed[msg.sender][_spender] == _value
   */
  function approve(address _spender, uint256 _value) public returns (bool) {
    allowed[msg.sender][_spender] = _value;
    emit  Approval(msg.sender, _spender, _value);
    return true;
  }

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

  /**
   * @dev Increase the amount of tokens that an owner allowed to a spender.
   * approve should be called when allowed[_spender] == 0. To increment
   * allowed value is better to use this function to avoid 2 calls (and wait until
   * the first transaction is mined)
   * @param _spender The address which will spend the funds.
   * @param _addedValue The amount of tokens to increase the allowance by.
   */
  /*@CTK increaseApproval
    @tag assume_completion
    @post __post.allowed[msg.sender][_spender] ==
          allowed[msg.sender][_spender] + _addedValue
   */
  function increaseApproval(address _spender, uint _addedValue) public returns (bool)
      {
    allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_addedValue);
    emit  Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
    return true;
  }

  /**
   * @dev Decrease the amount of tokens that an owner allowed to a spender.
   * approve should be called when allowed[_spender] == 0. To decrement
   * allowed value is better to use this function to avoid 2 calls (and wait until
   * the first transaction is mined)
   * @param _spender The address which will spend the funds.
   * @param _subtractedValue The amount of tokens to decrease the allowance by.
   */
  /*@CTK decreaseApproval_1
    @pre _subtractedValue > allowed[msg.sender][_spender]
    @post __post.allowed[msg.sender][_spender] == 0
   */
  /*@CTK decreaseApproval_2
    @pre _subtractedValue <= allowed[msg.sender][_spender]
    @post __post.allowed[msg.sender][_spender] ==
          allowed[msg.sender][_spender] - _subtractedValue
   */
  function decreaseApproval(address _spender, uint _subtractedValue) public returns (
      bool) {
    uint oldValue = allowed[msg.sender][_spender];
    if (_subtractedValue > oldValue) {
      allowed[msg.sender][_spender] = 0;
    } else {
      allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
    }
  emit  Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
    return true;
  }

}

contract PausableToken is StandardToken, Pausable {

  function transfer(address _to, uint256 _value) public whenNotPaused returns (bool) {
    return super.transfer(_to, _value);
  }

  function transferFrom(address _from, address _to, uint256 _value) public
      whenNotPaused returns (bool) {
    return super.transferFrom(_from, _to, _value);
  }

  function approve(address _spender, uint256 _value) public whenNotPaused returns (
      bool) {
    return super.approve(_spender, _value);
  }

  function increaseApproval(address _spender, uint _addedValue) public whenNotPaused
      returns (bool success) {
    return super.increaseApproval(_spender, _addedValue);
  }

  function decreaseApproval(address _spender, uint _subtractedValue) public
      whenNotPaused returns (bool success) {
    return super.decreaseApproval(_spender, _subtractedValue);
  }
}

/**
 * @dev Initialize contract basic information
 */
```

```
contract XBlockToken is PausableToken {
    string public name = "XBlock";
    string public symbol = "IX";
    uint public decimals = 18;
    uint public INITIAL_SUPPLY = 5000000000000000000000000000;

    /*@CTK XBlockToken
      @post __post.totalSupply == __post.balances[msg.sender]
     */
    function XBlockToken() public {
        totalSupply = INITIAL_SUPPLY;
        balances[msg.sender] = INITIAL_SUPPLY;
    }
}
```

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 t
        ) {
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
```

Static Analysis Request

INSECURE_COMPILER_VERSION

Line 1 in File xblock.sol

```
pragma solidity ^0.4.13;
```

Only these compiler versions are safe to compile your code: 0.4.25

Formal Verification Request 1

SafeMath mul

📅 23, Jan 2019
⏱ 485.14 ms

Line 4-9 in File xblock.sol

```
/*@CTK "SafeMath mul"
    @post (a > 0) && (((a * b) / a) != b) -> __reverted
    @post __reverted -> (a > 0) && (((a * b) / a) != b)
    @post !__reverted -> __return == a * b
    @post !__reverted == !__has_overflow
*/
```

Line 10-17 in File xblock.sol

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

✅ The code meets the specification

Formal Verification Request 2

SafeMath div

📅 23, Jan 2019
⏱ 7.62 ms

Line 19-23 in File xblock.sol

```
/*@CTK "SafeMath div"
    @post b != 0 -> !__reverted
    @post !__reverted -> __return == a / b
    @post !__reverted -> !__has_overflow
*/
```

Line 24-29 in File xblock.sol

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

✅ The code meets the specification

Formal Verification Request 3

SafeMath sub

📅 23, Jan 2019
⏱ 14.01 ms

Line 31-35 in File xblock.sol

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

Line 36-39 in File xblock.sol

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

✓ The code meets the specification

Formal Verification Request 4

SafeMath add

📅 23, Jan 2019
⏱ 16.73 ms

Line 41-45 in File xblock.sol

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

Line 46-50 in File xblock.sol

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

✓ The code meets the specification

Formal Verification Request 5

Ownable

📅 23, Jan 2019
⏱ 5.44 ms

Line 61-63 in File xblock.sol

```
/*@CTK Ownable
  @post __post.owner == msg.sender
 */
```

Line 64-66 in File xblock.sol

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

✓ The code meets the specification

Formal Verification Request 6

transferOwnership

23, Jan 2019
24.34 ms

Line 82-86 in File xblock.sol

```
/*@CTK transferOwnership
  @tag assume_completion
  @post newOwner != address(0)
  @post __post.owner == newOwner
 */
```

Line 87-91 in File xblock.sol

```
function transferOwnership(address newOwner) public onlyOwner {
  require(newOwner != address(0));
  emit OwnershipTransferred(owner, newOwner);
  owner = newOwner;
}
```

✓ The code meets the specification

Formal Verification Request 7

pause

23, Jan 2019
27.1 ms

Line 120-125 in File xblock.sol

```
/*@CTK pause
  @tag assume_completion
  @post paused == false
  @post owner == msg.sender
  @post __post.paused == true
 */
```

Line 126-129 in File xblock.sol

```
function pause() onlyOwner whenNotPaused public {
  paused = true;
  emit  Pause();
}
```

✓ The code meets the specification

Formal Verification Request 8

unpause

📅 23, Jan 2019
⏱ 24.23 ms

Line 134-139 in File xblock.sol

```
/*@CTK unpause
  @tag assume_completion
  @post paused == true
  @post owner == msg.sender
  @post __post.paused == false
 */
```

Line 140-143 in File xblock.sol

```
function unpause() onlyOwner whenPaused public {
  paused = false;
  emit Unpause();
}
```

✓ The code meets the specification

Formal Verification Request 9

transfer

📅 23, Jan 2019
⏱ 188.7 ms

Line 169-176 in File xblock.sol

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

Line 177-186 in File xblock.sol

```
177  function transfer(address _to, uint256 _value) public returns (bool) {
178    require(_to != address(0));
179    require(_value <= balances[msg.sender]);
180
181    // SafeMath.sub will throw if there is not enough balance.
182    balances[msg.sender] = balances[msg.sender].sub(_value);
183    balances[_to] = balances[_to].add(_value);
184    emit Transfer(msg.sender, _to, _value);
185    return true;
186  }
```

✓ The code meets the specification

Formal Verification Request 10

balanceOf

23, Jan 2019
5.79 ms

Line 193-195 in File xblock.sol

```
193  /*@CTK balanceOf
194    @post balance == balances[_owner]
195   */
```

Line 196-198 in File xblock.sol

```
196  function balanceOf(address _owner) public view returns (uint256 balance) {
197    return balances[_owner];
198  }
```

✓ The code meets the specification

Formal Verification Request 11

transferFrom

23, Jan 2019
317.13 ms

Line 211-219 in File xblock.sol

```
211  /*@CTK transferFrom
212    @tag assume_completion
213    @pre _from != _to
214    @post _to != address(0)
215    @post _value <= balances[_from]
216    @post _value <= allowed[_from][msg.sender]
217    @post __post.balances[_from] == balances[_from] - _value
218    @post __post.balances[_to] == balances[_to] + _value
219   */
```

Line 220-230 in File xblock.sol

```
220  function transferFrom(address _from, address _to, uint256 _value) public returns (
         bool) {
221    require(_to != address(0));
222    require(_value <= balances[_from]);
223    require(_value <= allowed[_from][msg.sender]);
224
225    balances[_from] = balances[_from].sub(_value);
226    balances[_to] = balances[_to].add(_value);
227    allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
228    emit Transfer(_from, _to, _value);
229    return true;
230  }
```

✓ The code meets the specification

Formal Verification Request 12

approve

📅 23, Jan 2019
⏱ 10.38 ms

Line 237-239 in File xblock.sol

```
237  /*@CTK approve
238    @post __post.allowed[msg.sender][_spender] == _value
239   */
```

Line 240-244 in File xblock.sol

```
240  function approve(address _spender, uint256 _value) public returns (bool) {
241    allowed[msg.sender][_spender] = _value;
242    emit  Approval(msg.sender, _spender, _value);
243    return true;
244  }
```

✓ The code meets the specification

Formal Verification Request 13

allowance

📅 23, Jan 2019
⏱ 5.94 ms

Line 252-254 in File xblock.sol

```
252  /*@CTK allowance
253    @post __return == allowed[_owner][_spender]
254   */
```

Line 255-257 in File xblock.sol

```
255  function allowance(address _owner, address _spender) public view returns (uint256) {
256    return allowed[_owner][_spender];
257  }
```

✓ The code meets the specification

Formal Verification Request 14

increaseApproval

23, Jan 2019
35.94 ms

Line 267-271 in File xblock.sol

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

Line 272-276 in File xblock.sol

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

✓ The code meets the specification

Formal Verification Request 15

decreaseApproval_1

23, Jan 2019
47.43 ms

Line 286-289 in File xblock.sol

```
/*@CTK decreaseApproval_1
  @pre _subtractedValue > allowed[msg.sender][_spender]
  @post __post.allowed[msg.sender][_spender] == 0
 */
```

Line 295-304 in File xblock.sol

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

✓ The code meets the specification

Formal Verification Request 16

decreaseApproval_2

23, Jan 2019
2.88 ms

Line 290-294 in File xblock.sol

```
/*@CTK decreaseApproval_2
  @pre _subtractedValue <= allowed[msg.sender][_spender]
  @post __post.allowed[msg.sender][_spender] ==
        allowed[msg.sender][_spender] - _subtractedValue
 */
```

Line 295-304 in File xblock.sol

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

✓ The code meets the specification

Formal Verification Request 17

XBlockToken

23, Jan 2019
13.43 ms

Line 340-342 in File xblock.sol

```
  /*@CTK XBlockToken
    @post __post.totalSupply == __post.balances[msg.sender]
   */
```

Line 343-346 in File xblock.sol

```
  function XBlockToken() public {
      totalSupply = INITIAL_SUPPLY;
      balances[msg.sender] = INITIAL_SUPPLY;
  }
```

✓ The code meets the specification