CertiK Verification Report For SpendCoin

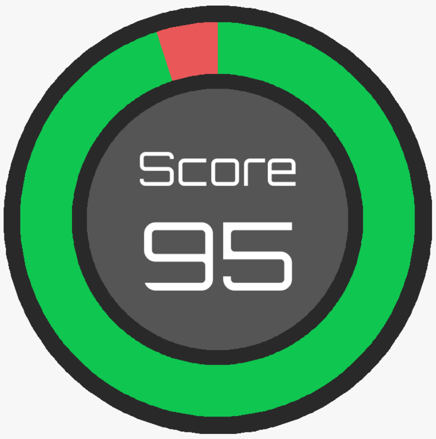
Request Date: 2019-03-29
Revision Date: 2019-03-31

Disclaimer

PASS

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

Mar 31, 2019



Summary

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

Type of Issues

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

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

Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0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

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

modifier `validLock` unused

Functions are protected by `onlyOwner`. So vulnerability is not a big issue. But modifier `validLock` is unnecessary at that point. Also the hardcoded `endtime` is not realistic.

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.

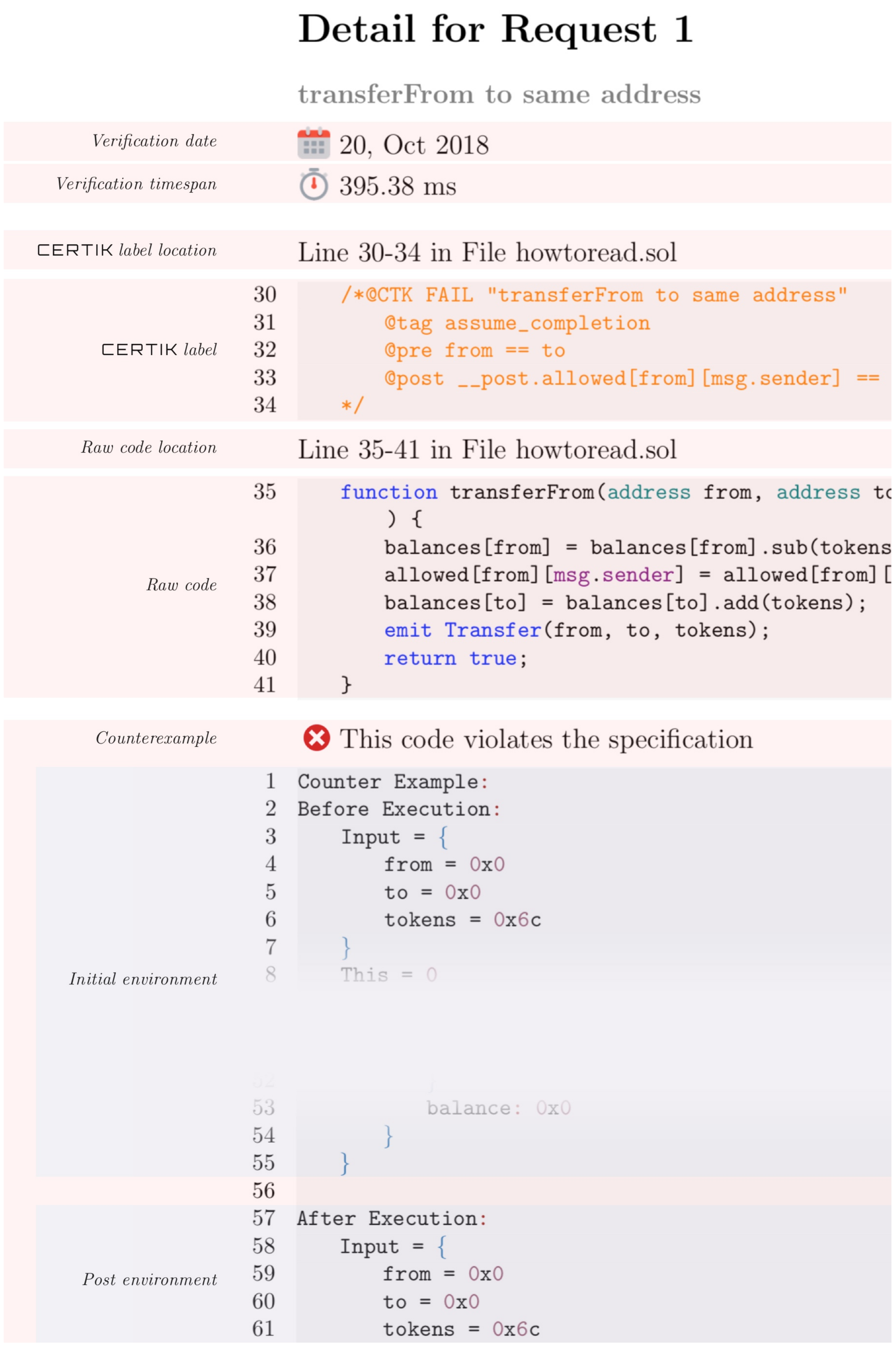
Source Code with CertiK Labels

File spendcoin.sol

```
pragma solidity ^0.4.18;


// ----------------------------------------------------------------------------

// Spendcoin Contract
//
// Symbol      : SPND
// Name        : Spendcoin
// Total supply: 2,000,000,000.000000000000000000
// Decimals    : 18
// Website     : https://spendcoin.org
// ----------------------------------------------------------------------------



// ----------------------------------------------------------------------------

// Safe maths

// ----------------------------------------------------------------------------

library SafeMath {
    /*@CTK SafeMath_add
      @post __reverted == __has_overflow
      @post __reverted == false -> c == a + b
      @post msg == msg__post
      @post (a + b < a) == __has_overflow
      @post __addr_map == __addr_map__post
     */
    function add(uint a, uint b) internal pure returns (uint c) {

        c = a + b;

        require(c >= a);

    }

    /*@CTK "SafeMath sub"
      @post (a < b) == __reverted
      @post !__reverted -> c == a - b
      @post !__reverted -> !__has_overflow
    */
    function sub(uint a, uint b) internal pure returns (uint c) {

        require(b <= a);

        c = a - b;

    }

    /*@CTK SafeMath_mul
      @post __reverted == __has_overflow
      @post __reverted == false -> c == a * b
      @post a == 0 -> c == 0
      @post msg == msg__post
      @post (a > 0 && (a * b / a != b)) == __reverted
      @post __addr_map == __addr_map__post
    */
    function mul(uint a, uint b) internal pure returns (uint c) {

        c = a * b;

        require(a == 0 || c / a == b);

    }

    /*@CTK "SafeMath div"
      @post b != 0 -> !__reverted
      @post !__reverted -> c == a / b
      @post !__reverted -> !__has_overflow
    */
    function div(uint a, uint b) internal 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);
```

```

}



// ----------------------------------------------------------------------------

// Owned contract

// ----------------------------------------------------------------------------

contract Owned {

    address public owner;

    /*@CTK Ownable
      @post __post.owner == msg.sender
     */
    function Owned() public {

        owner = msg.sender;

    }


    modifier onlyOwner {

        require(msg.sender == owner);

        _;

    }

}

contract Tokenlock is Owned {

    uint lockStartTime = 0; //time from when token will be locked
    uint lockEndTime = 0;   //time from when token will be locked
    uint8 isLocked = 0;     //flag indicates if token is locked

    event Freezed(uint starttime, uint endtime);
    event UnFreezed();

    modifier validLock {
        require(isLocked == 0 || (now < lockStartTime || now > lockEndTime));
        _;
    }

    /*@CTK freezeTime
      @tag assume_completion
      @post owner == msg.sender
      @post __post.isLocked == 1
      @post __post.lockStartTime == _startTime
      @post __post.lockEndTime == _endTime
     */
    function freezeTime(uint _startTime, uint _endTime) public onlyOwner {
        isLocked = 1;
        lockStartTime = _startTime;
        lockEndTime = _endTime;

        emit Freezed(lockStartTime, lockEndTime);
    }

    /*@CTK freeze
      @tag assume_completion
      @post owner == msg.sender
      @post __post.isLocked == 1
      @post __post.lockStartTime == 0
      @post __post.lockEndTime == 90000000000
     */
    function freeze() public onlyOwner {
        isLocked = 1;
        lockStartTime = 0;
        lockEndTime = 90000000000;

        emit Freezed(lockStartTime, lockEndTime);
    }

    /*@CTK unfreeze
      @tag assume_completion
      @post owner == msg.sender
      @post __post.isLocked == 0
      @post __post.lockStartTime == 0
      @post __post.lockEndTime == 0
     */
    function unfreeze() public onlyOwner {
        isLocked = 0;
        lockStartTime = 0;
        lockEndTime = 0;

        emit UnFreezed();
    }
}


// ----------------------------------------------------------------------------

// ERC20 Token, with the addition of symbol, name and decimals and an

// initial fixed supply

// ----------------------------------------------------------------------------

contract Spendcoin is ERC20Interface, Tokenlock {

    using SafeMath for uint;


    string public symbol;

    string public name;

    uint8 public decimals;

    uint public _totalSupply;


    mapping(address => uint) balances;

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



    // ------------------------------------------------------------------------

    // Constructor

    // ------------------------------------------------------------------------
    /*@CTK Spendcoin
      @post __post.symbol == "SPND"
      @post __post.name == "Spendcoin"
      @post __post.decimals == 18
      @post __post.balances[owner] == __post._totalSupply
     */
    function Spendcoin() public {

        symbol = "SPND";

        name = "Spendcoin";

        decimals = 18;

        _totalSupply = 2000000000 * 10**uint(decimals);

        balances[owner] = _totalSupply;

        emit Transfer(address(0), owner, _totalSupply);

    }



    // ------------------------------------------------------------------------

    // Total supply

    // ------------------------------------------------------------------------
    /*@CTK totalSupply
      @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 balanceOf
      @post balance == 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 transfer
      @tag assume_completion
      @pre msg.sender != to
      @post __post.balances[msg.sender] == balances[msg.sender] - tokens
      @post __post.balances[to] == balances[to] + tokens
     */
    function transfer(address to, uint tokens) public returns (bool success) {

        balances[msg.sender] = balances[msg.sender].sub(tokens);

        balances[to] = balances[to].add(tokens);

        emit 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 approve
      @post __post.allowed[msg.sender][spender] == tokens
     */
    function approve(address spender, uint tokens) public returns (bool success) {

        allowed[msg.sender][spender] = tokens;

        emit 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 transferFrom
      @tag assume_completion
      @pre from != to
      @post __post.balances[from] == balances[from] - tokens
      @post __post.balances[to] == balances[to] + tokens
      @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] = balances[from].sub(tokens);

        allowed[from][msg.sender] = allowed[from][msg.sender].sub(tokens);

        balances[to] = balances[to].add(tokens);

        emit 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 allowance
      @post remaining == allowed[tokenOwner][spender]
     */
    function allowance(address tokenOwner, address spender) public constant returns (
        uint remaining) {

        return allowed[tokenOwner][spender];

    }


    // ----------------------------------------------------------------------------

    // Do accept ETH

    // ----------------------------------------------------------------------------

    function () public payable {


    }


    // ----------------------------------------------------------------------------
    // Owner can withdraw ether if token received.
    // ----------------------------------------------------------------------------
    function withdraw() public onlyOwner returns (bool result) {
        address tokenaddress = this;
        // CTK: owner.send(this.balance)
        return owner.send(tokenaddress.balance);
    }

    // ----------------------------------------------------------------------------

    // 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);

    }

}
```

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
```

Static Analysis Request

TIMESTAMP_DEPENDENCY

Line 156 in File spendcoin.sol

```
156         require(isLocked == 0 || (now < lockStartTime || now > lockEndTime));
```

❗ ”now” can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 156 in File spendcoin.sol

```
156         require(isLocked == 0 || (now < lockStartTime || now > lockEndTime));
```

❗ ”now” can be influenced by minors to some degree

Formal Verification Request 1

SafeMath_add

📅 31, Mar 2019
⏱ 17.97 ms

Line 24-30 in File spendcoin.sol

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

Line 31-37 in File spendcoin.sol

```
function add(uint a, uint b) internal pure returns (uint c) {

    c = a + b;

    require(c >= a);

}
```

✓ The code meets the specification

Formal Verification Request 2

SafeMath sub

📅 31, Mar 2019
⏱ 14.26 ms

Line 39-43 in File spendcoin.sol

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

Line 44-50 in File spendcoin.sol

```
function sub(uint a, uint b) internal pure returns (uint c) {

    require(b <= a);

    c = a - b;

}
```

✓ The code meets the specification

Formal Verification Request 3

SafeMath_mul

📅 31, Mar 2019
⏱ 126.06 ms

Line 52-59 in File spendcoin.sol

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

Line 60-66 in File spendcoin.sol

```
    function mul(uint a, uint b) internal pure returns (uint c) {

        c = a * b;

        require(a == 0 || c / a == b);

    }
```

✓ The code meets the specification

Formal Verification Request 4

SafeMath div

📅 31, Mar 2019
⏱ 14.81 ms

Line 68-72 in File spendcoin.sol

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

Line 73-79 in File spendcoin.sol

```
    function div(uint a, uint b) internal pure returns (uint c) {

        require(b > 0);

        c = a / b;

    }
```

✓ The code meets the specification

Formal Verification Request 5

Ownable

31, Mar 2019
6.0 ms

Line 126-128 in File spendcoin.sol

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

Line 129-133 in File spendcoin.sol

```
function Owned() public {

    owner = msg.sender;

}
```

✓ The code meets the specification

Formal Verification Request 6

freezeTime

31, Mar 2019
26.1 ms

Line 160-166 in File spendcoin.sol

```
/*@CTK freezeTime
  @tag assume_completion
  @post owner == msg.sender
  @post __post.isLocked == 1
  @post __post.lockStartTime == _startTime
  @post __post.lockEndTime == _endTime
 */
```

Line 167-173 in File spendcoin.sol

```
function freezeTime(uint _startTime, uint _endTime) public onlyOwner {
    isLocked = 1;
    lockStartTime = _startTime;
    lockEndTime = _endTime;

    emit Freezed(lockStartTime, lockEndTime);
}
```

✓ The code meets the specification

Formal Verification Request 7

freeze

📅 31, Mar 2019
⏱ 23.95 ms

Line 175-181 in File spendcoin.sol

```
    /*@CTK freeze
      @tag assume_completion
      @post owner == msg.sender
      @post __post.isLocked == 1
      @post __post.lockStartTime == 0
      @post __post.lockEndTime == 90000000000
     */
```

Line 182-188 in File spendcoin.sol

```
    function freeze() public onlyOwner {
        isLocked = 1;
        lockStartTime = 0;
        lockEndTime = 90000000000;

        emit Freezed(lockStartTime, lockEndTime);
    }
```

✓ The code meets the specification

Formal Verification Request 8

unfreeze

📅 31, Mar 2019
⏱ 26.4 ms

Line 190-196 in File spendcoin.sol

```
    /*@CTK unfreeze
      @tag assume_completion
      @post owner == msg.sender
      @post __post.isLocked == 0
      @post __post.lockStartTime == 0
      @post __post.lockEndTime == 0
     */
```

Line 197-203 in File spendcoin.sol

```
    function unfreeze() public onlyOwner {
        isLocked = 0;
        lockStartTime = 0;
        lockEndTime = 0;

        emit UnFreezed();
    }
```

✓ The code meets the specification

Formal Verification Request 9

Spendcoin

31, Mar 2019
34.86 ms

Line 240-245 in File spendcoin.sol

```
/*@CTK Spendcoin
  @post __post.symbol == "SPND"
  @post __post.name == "Spendcoin"
  @post __post.decimals == 18
  @post __post.balances[owner] == __post._totalSupply
 */
```

Line 246-260 in File spendcoin.sol

```
function Spendcoin() public {

    symbol = "SPND";

    name = "Spendcoin";

    decimals = 18;

    _totalSupply = 2000000000 * 10**uint(decimals);

    balances[owner] = _totalSupply;

    emit Transfer(address(0), owner, _totalSupply);

}
```

✓ The code meets the specification

Formal Verification Request 10

totalSupply

31, Mar 2019
8.73 ms

Line 269-271 in File spendcoin.sol

```
/*@CTK totalSupply
  @post __return == _totalSupply - balances[address(0)]
 */
```

Line 272-276 in File spendcoin.sol

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

    return _totalSupply - balances[address(0)];

}
```

✓ The code meets the specification

Formal Verification Request 11

balanceOf

📅 31, Mar 2019
⏱ 7.22 ms

Line 285-287 in File spendcoin.sol

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

Line 288-292 in File spendcoin.sol

```
function balanceOf(address tokenOwner) public constant returns (uint balance) {

    return balances[tokenOwner];

}
```

✓ The code meets the specification

Formal Verification Request 12

transfer

📅 31, Mar 2019
⏱ 110.56 ms

Line 305-310 in File spendcoin.sol

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

Line 311-321 in File spendcoin.sol

```
function transfer(address to, uint tokens) public returns (bool success) {

    balances[msg.sender] = balances[msg.sender].sub(tokens);

    balances[to] = balances[to].add(tokens);

    emit Transfer(msg.sender, to, tokens);

    return true;

}
```

✓ The code meets the specification

Formal Verification Request 13

approve

📅 31, Mar 2019
⏱ 15.23 ms

Line 340-342 in File spendcoin.sol

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

Line 343-351 in File spendcoin.sol

```
function approve(address spender, uint tokens) public returns (bool success) {

    allowed[msg.sender][spender] = tokens;

    emit Approval(msg.sender, spender, tokens);

    return true;

}
```

✓ The code meets the specification

Formal Verification Request 14

transferFrom

📅 31, Mar 2019
⏱ 178.42 ms

Line 372-378 in File spendcoin.sol

```
/*@CTK transferFrom
  @tag assume_completion
  @pre from != to
  @post __post.balances[from] == balances[from] - tokens
  @post __post.balances[to] == balances[to] + tokens
  @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - tokens
 */
```

Line 379-391 in File spendcoin.sol

```
function transferFrom(address from, address to, uint tokens) public returns (bool
    success) {

    balances[from] = balances[from].sub(tokens);

    allowed[from][msg.sender] = allowed[from][msg.sender].sub(tokens);

    balances[to] = balances[to].add(tokens);

    emit Transfer(from, to, tokens);

```

```
        return true;

    }
```

✓ The code meets the specification

Formal Verification Request 15

allowance

31, Mar 2019
7.11 ms

Line 402-404 in File spendcoin.sol

```
    /*@CTK allowance
      @post remaining == allowed[tokenOwner][spender]
     */
```

Line 405-409 in File spendcoin.sol

```
    function allowance(address tokenOwner, address spender) public constant returns (
        uint remaining) {

        return allowed[tokenOwner][spender];

    }
```

✓ The code meets the specification