CertiK Verification Report For Flowchain

Request Date: 2019-01-15
Revision Date: 2019-01-16

Disclaimer

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

WARNING

CERTIK *identified some potential security flaws in this contract and also provided corresponding solutions.*

Jan 16, 2019

Summary

This is the report for smart contract verification service requestd by Flowchain. 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.	N	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.	1	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Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0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

Fallback function in the VestingTokenSale.sol has an integer overflow issue that allows anyone to issue token to the vesting contract and becomes an accredited investor. Given that there is a separate function `setupAccreditedAddress` that is used to setup specific addresses as accredited investors, it is hard to believe that this is intentional.

Specifically, the fallback function of `VestingTokenSale` is:

```
function () payable {
    // check if we can offer the private sale
    require(isFunding == true \&\& amountRaised < fundingGoal);

    // the minimum deposit is 1 ETH
    uint256 amount = msg.value;
    require(amount >= 1 ether);

    require(accredited[msg.sender] - amount >= 0); // <<<<<<<<<<<< buggy line

    multiSigWallet.transfer(amount);
    balanceOf[msg.sender] += amount;
    accredited[msg.sender] -= amount;
    amountRaised += amount;
```

```
    FundTransfer(msg.sender, amount);

    // total releasable tokens
    uint256 value = amount.mul(tokensPerEther);

    // Mint tokens and keep it in the contract
    tokenReward.mintToken(addressOfVestingApp, value);
}
```

In the buggy line above, when `accredited[msg.sender]` is less than `amount`, not only will `accredited[msg.sender] - amount >= 0` evaluates to true, but the result of this statement is a huge number. As a result, real investors's coins can depreciate significantly. Use `require(accredited[msg.sender] >= amount)‘` instead to avoid this issue.

Medium

No issue found.

Low

A bunch of functions in the SafeMath library are redundant.

There are arithmetic functions for signed integer in the library but they are not used at all. Removing unnecessary function might help reduce maintenance cost.

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 VestingTokenSale.sol

```
pragma solidity 0.4.24;

/**
 * Copyright 2018, The Flowchain Foundation Limited
 *
 * The FlowchainCoin (FLC) Token Sale Contract
 *
 *  - Private Sale A
 *  - Monthly Vest
 */

/**
 * @title SafeMath
 * @dev Math operations with safety checks that revert on error
 */
library SafeMath {
    int256 constant private INT256_MIN = -2**255;

    /**
    * @dev Multiplies two unsigned integers, reverts on overflow.
    */
    /*@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) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b);

        return c;
    }

    /**
    * @dev Multiplies two signed integers, reverts on overflow.
    */
    function mul(int256 a, int256 b) internal pure returns (int256) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
            return 0;
        }

        require(!(a == -1 && b == INT256_MIN)); // This is the only case of overflow
            not detected by the check below
```

```
        int256 c = a * b;
        require(c / a == b);

        return c;
    }

    /**
    * @dev Integer division of two unsigned integers truncating the quotient, reverts on division by zero.
    */
    /*@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) {
        // Solidity only automatically asserts when dividing by 0
        require(b > 0);
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't hold

        return c;
    }

    /**
    * @dev Integer division of two signed integers truncating the quotient, reverts on division by zero.
    */
    function div(int256 a, int256 b) internal pure returns (int256) {
        require(b != 0); // Solidity only automatically asserts when dividing by 0
        require(!(b == -1 && a == INT256_MIN)); // This is the only case of overflow

        int256 c = a / b;

        return c;
    }

    /**
    * @dev Subtracts two unsigned integers, reverts on overflow (i.e. if subtrahend is greater than minuend).
    */
    /*@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) {
        require(b <= a);
        uint256 c = a - b;

        return c;
    }

    /**
    * @dev Subtracts two signed integers, reverts on overflow.
    */
    function sub(int256 a, int256 b) internal pure returns (int256) {
```

```
        int256 c = a - b;
        require((b >= 0 && c <= a) || (b < 0 && c > a));

        return c;
    }

    /**
    * @dev Adds two unsigned integers, reverts on overflow.
    */
    /*@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;
        require(c >= a);

        return c;
    }

    /**
    * @dev Adds two signed integers, reverts on overflow.
    */
    function add(int256 a, int256 b) internal pure returns (int256) {
        int256 c = a + b;
        require((b >= 0 && c >= a) || (b < 0 && c < a));

        return c;
    }

    /**
    * @dev Divides two unsigned integers and returns the remainder (unsigned integer
        modulo),
    * reverts when dividing by zero.
    */
    /*@CTK "SafeMath mod"
        @tag assume_completion
        @post b != 0
        @post __return == a % b
     */
    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b != 0);
        return a % b;
    }
}

interface Token {
    /// @dev Mint an amount of tokens and transfer to the backer
    /// @param to The address of the backer who will receive the tokens
    /// @param amount The amount of rewarded tokens
    /// @return The result of token transfer
    function mintToken(address to, uint amount) external returns (bool success);

    /// @param _owner The address from which the balance will be retrieved
    /// @return The balance
    function balanceOf(address _owner) public view returns (uint256 balance);

```

```
    /// @notice send `_value` token to `_to` from `msg.sender`
    /// @param _to The address of the recipient
    /// @param _value The amount of token to be transferred
    /// @return Whether the transfer was successful or not
    function transfer(address _to, uint256 _value) public returns (bool success);
}

contract MintableSale {
    // @notice Create a new mintable sale
    /// @param vestingAddress The vesting app
    /// @param rate The exchange rate
    /// @param fundingGoalInEthers The funding goal in ethers
    /// @param durationInMinutes The duration of the sale in minutes
    /// @return
    function createMintableSale(address vestingAddress, uint256 rate, uint256
        fundingGoalInEthers, uint durationInMinutes) public returns (bool success);
}

contract VestingTokenSale is MintableSale {
    using SafeMath for uint256;
    uint256 public fundingGoal;
    uint256 public tokensPerEther;
    uint public deadline;
    address public multiSigWallet;
    uint256 public amountRaised;
    Token public tokenReward;
    mapping(address => uint256) public balanceOf;
    bool fundingGoalReached = false;
    bool crowdsaleClosed = false;
    address public creator;
    address public addressOfTokenUsedAsReward;
    bool public isFunding = false;

    /* accredited investors */
    mapping (address => uint256) public accredited;

    event FundTransfer(address backer, uint amount);

    address public addressOfVestingApp;
    uint256 constant public VESTING_DURATION  =  31536000; // 1 Year in second
    uint256 constant public CLIFF_DURATION    =   2592000; // 1 months (30 days) in
        second

    /* Constrctor function */
    function VestingTokenSale(
        address _addressOfTokenUsedAsReward
    ) payable {
        creator = msg.sender;
        multiSigWallet = 0x9581973c54fce63d0f5c4c706020028af20ff723;

        // Token Contract
        addressOfTokenUsedAsReward = _addressOfTokenUsedAsReward;
        tokenReward = Token(addressOfTokenUsedAsReward);

        // Setup accredited investors
        setupAccreditedAddress(0xec7210E3db72651Ca21DA35309A20561a6F374dd, 1000);
    }

```

```
    // @dev Start a new mintable sale.
    // @param vestingAddress The vesting app
    // @param rate The exchange rate in ether, for example 1 ETH = 6400 FLC
    // @param fundingGoalInEthers
    // @param durationInMinutes
    /*@CTK createMintableSale
        @tag assume_completion
        @post msg.sender == creator
        @post rate <= 6400 && rate >= 1
        @post fundingGoalInEthers >= 1
        @post durationInMinutes >= 60 minutes
        @post __post.addressOfVestingApp == vestingAddrss
        @post __post.deadline == now + durationInMinutes * 1 minutes
        @post __post.fundingGoal == amountRaised + fundingGoalInEthers * 1 ether
        @post __post.tokensPerEther == rate
        @post __post.isFunding == true
     */
    function createMintableSale(address vestingAddrss, uint256 rate, uint256
        fundingGoalInEthers, uint durationInMinutes) public returns (bool success) {
        require(msg.sender == creator);
        require(isFunding == false);
        require(rate <= 6400 && rate >= 1);              // rate must be between 1 and
            6400
        require(fundingGoalInEthers >= 1);
        require(durationInMinutes >= 60 minutes);

        addressOfVestingApp = vestingAddrss;

        deadline = now + durationInMinutes * 1 minutes;
        fundingGoal = amountRaised + fundingGoalInEthers * 1 ether;
        tokensPerEther = rate;
        isFunding = true;
        return true;
    }

    modifier afterDeadline() { if (now > deadline) _; }
    modifier beforeDeadline() { if (now <= deadline) _; }

    /// @param _accredited The address of the accredited investor
    /// @param _amountInEthers The amount of remaining ethers allowed to invested
    /// @return Amount of remaining tokens allowed to spent
    /*@CTK setupAccreditedAddress
        @tag assume_completion
        @post msg.sender == creator
        @post __post.accredited[_accredited] == _amountInEthers * 1 ether
     */
    function setupAccreditedAddress(address _accredited, uint _amountInEthers) public
        returns (bool success) {
        require(msg.sender == creator);
        accredited[_accredited] = _amountInEthers * 1 ether;
        return true;
    }

    /// @dev This function returns the amount of remaining ethers allowed to invested
    /// @return The amount
    /*@CTK getAmountAccredited
      @post __return == accredited[_accredited]
     */
```

```
    function getAmountAccredited(address _accredited) view returns (uint256) {
        uint256 amount = accredited[_accredited];
        return amount;
    }

    /*@CTK closeSale
      @tag assume_completion
      @pre now <= deadline
      @post msg.sender == creator
      @post __post.isFunding == false
     */
    function closeSale() beforeDeadline {
        require(msg.sender == creator);
        isFunding = false;
    }

    // change creator address
    /*@CTK changeCreator
      @tag assume_completion
      @post __post.creator == _creator
     */
    function changeCreator(address _creator) external {
        require(msg.sender == creator);
        creator = _creator;
    }

    /// @dev This function returns the current exchange rate during the sale
    /// @return The address of token creator
    /*@CTK getRate
      @pre now <= deadline
      @post __return == tokensPerEther
     */
    function getRate() beforeDeadline view returns (uint) {
        return tokensPerEther;
    }

    /// @dev This function returns the amount raised in wei
    /// @return The address of token creator
    /*@CTK getAmountRaised
      @post __return == amountRaised
     */
    function getAmountRaised() view returns (uint) {
        return amountRaised;
    }

    /*CTK fallback
      @tag assume_completion
      @post isFunding == true && amountRaised < fundingGoal
      @post msg.value >= 1 ether
      @post accredited[msg.sender] >= msg.value
     */
    function () payable {
        // check if we can offer the private sale
        require(isFunding == true && amountRaised < fundingGoal);

        // the minimum deposit is 1 ETH
        uint256 amount = msg.value;
        require(amount >= 1 ether);
```

```

        /*****************************************************************/
        // Added by CTK. integer overflow when accredited[msg.sender] < 0
        // should be require(accredited[msg.sender] >= amount) instead.
        /*****************************************************************/
        require(accredited[msg.sender] - amount >= 0);

        multiSigWallet.transfer(amount);
        balanceOf[msg.sender] += amount;
        accredited[msg.sender] -= amount;
        amountRaised += amount;
        FundTransfer(msg.sender, amount);

        // total releasable tokens
        uint256 value = amount.mul(tokensPerEther);

        // Mint tokens and keep it in the contract
        tokenReward.mintToken(addressOfVestingApp, value);
    }
}
```

File VestingDapp.sol

```
pragma solidity 0.4.24;

/**
 * Copyright 2018, The Flowchain Foundation Limited
 *
 * The FlowchainCoin (FLC) token contract for vesting sale
 */

/**
 * @title SafeMath
 * @dev Math operations with safety checks that revert on error
 */
library SafeMath {
    int256 constant private INT256_MIN = -2**255;

    /**
    * @dev Multiplies two unsigned integers, reverts on overflow.
    */
    /*@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) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b);

        return c;
```

```
    }

    /**
    * @dev Multiplies two signed integers, reverts on overflow.
    */
    function mul(int256 a, int256 b) internal pure returns (int256) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
            return 0;
        }

        require(!(a == -1 && b == INT256_MIN)); // This is the only case of overflow
            not detected by the check below

        int256 c = a * b;
        require(c / a == b);

        return c;
    }

    /**
    * @dev Integer division of two unsigned integers truncating the quotient, reverts
        on division by zero.
    */
    /*@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) {
        // Solidity only automatically asserts when dividing by 0
        require(b > 0);
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't hold

        return c;
    }

    /**
    * @dev Integer division of two signed integers truncating the quotient, reverts on
         division by zero.
    */
    function div(int256 a, int256 b) internal pure returns (int256) {
        require(b != 0); // Solidity only automatically asserts when dividing by 0
        require(!(b == -1 && a == INT256_MIN)); // This is the only case of overflow

        int256 c = a / b;

        return c;
    }

    /**
    * @dev Subtracts two unsigned integers, reverts on overflow (i.e. if subtrahend is
         greater than minuend).
    */
    /*@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) {
        require(b <= a);
        uint256 c = a - b;

        return c;
    }

    /**
    * @dev Subtracts two signed integers, reverts on overflow.
    */
    function sub(int256 a, int256 b) internal pure returns (int256) {
        int256 c = a - b;
        require((b >= 0 && c <= a) || (b < 0 && c > a));

        return c;
    }

    /**
    * @dev Adds two unsigned integers, reverts on overflow.
    */
    /*@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;
        require(c >= a);

        return c;
    }

    /**
    * @dev Adds two signed integers, reverts on overflow.
    */
    function add(int256 a, int256 b) internal pure returns (int256) {
        int256 c = a + b;
        require((b >= 0 && c >= a) || (b < 0 && c < a));

        return c;
    }

    /**
    * @dev Divides two unsigned integers and returns the remainder (unsigned integer
        modulo),
    * reverts when dividing by zero.
    */
    /*@CTK "SafeMath mod"
        @tag assume_completion
        @post b != 0
        @post __return == a % b
     */
    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b != 0);
```

```
        return a % b;
    }
}

interface Token {
    /// @param _owner The address from which the balance will be retrieved
    /// @return The balance
    function balanceOf(address _owner) public view returns (uint256 balance);

    /// @notice send `_value` token to `_to` from `msg.sender`
    /// @param _to The address of the recipient
    /// @param _value The amount of token to be transferred
    /// @return Whether the transfer was successful or not
    function transfer(address _to, uint256 _value) public returns (bool success);
}

/**
 * @title TokenVesting
 * @dev A token holder contract that can release its token balance gradually like a
 * typical vesting scheme, with a cliff and vesting period. Optionally revocable by the
 * owner.
 */
contract Vesting {
    using SafeMath for uint256;

    Token public tokenReward;

    // beneficiary of tokens after they are released
    address private _beneficiary;

    uint256 private _cliff;
    uint256 private _start;
    uint256 private _duration;

    address public _addressOfTokenUsedAsReward;
    address public creator;

    mapping (address => uint256) private _released;

    /* Constrctor function */
    /*@CTK Vesting
        @post __post.creator == msg.sender
     */
    function Vesting() payable {
        creator = msg.sender;
    }

    /**
     * @dev Creates a vesting contract that vests its balance of FLC token to the
     * beneficiary, gradually in a linear fashion until start + duration. By then all
     * of the balance will have vested.
     * @param beneficiary address of the beneficiary to whom vested tokens are transferred
     * @param cliffDuration duration in seconds of the cliff in which tokens will begin to vest
     * @param start the time (as Unix time) at which point vesting starts
     * @param duration duration in seconds of the period in which the tokens will vest
```

```
     * @param addressOfTokenUsedAsReward where is the token contract
     */
    /*@CTK createVestingPeriod
       @tag assume_completion
       @post msg.sender == creator
       @post cliffDuration <= duration
       @post duration > 0
       @post start + duration > block.timestamp
       @post __post._beneficiary == beneficiary
       @post __post._duration == duration
       @post __post._cliff == start + cliffDuration
       @post __post._start == start
       @post __post._addressOfTokenUsedAsReward == addressOfTokenUsedAsReward
     */
    function createVestingPeriod(address beneficiary, uint256 start, uint256
        cliffDuration, uint256 duration, address addressOfTokenUsedAsReward) public {
        require(msg.sender == creator);
        require(cliffDuration <= duration);
        require(duration > 0);
        require(start.add(duration) > block.timestamp);

        _beneficiary = beneficiary;
        _duration = duration;
        _cliff = start.add(cliffDuration);
        _start = start;
        _addressOfTokenUsedAsReward = addressOfTokenUsedAsReward;
        tokenReward = Token(addressOfTokenUsedAsReward);
    }

    /**
     * @return the beneficiary of the tokens.
     */
    /*@CTK beneficiary
      @post __return == _beneficiary
     */
    function beneficiary() public view returns (address) {
        return _beneficiary;
    }

    /**
     * @return the cliff time of the token vesting.
     */
    /*@CTK cliff
      @post __return == _cliff
     */
    function cliff() public view returns (uint256) {
        return _cliff;
    }

    /**
     * @return the start time of the token vesting.
     */
    /*@CTK start
      @post __return == _start
     */
    function start() public view returns (uint256) {
        return _start;
    }
```

```
    /**
     * @return the duration of the token vesting.
     */
    /*@CTK duration
      @post __return == _duration
     */
    function duration() public view returns (uint256) {
        return _duration;
    }

    /**
     * @return the amount of the token released.
     */
    /*@CTK released
      @post __return == _released[token]
     */
    function released(address token) public view returns (uint256) {
        return _released[token];
    }

    /**
     * @notice Mints and transfers tokens to beneficiary.
     * @param token ERC20 token which is being vested
     */
    function release(address token) public {
        require(msg.sender == creator);

        uint256 unreleased = _releasableAmount(token);

        require(unreleased > 0);

        _released[token] = _released[token].add(unreleased);

        tokenReward.transfer(_beneficiary, unreleased);
    }

    /**
     * @dev Calculates the amount that has already vested but hasn't been released yet
        .
     * @param token ERC20 token which is being vested
     */
    function _releasableAmount(address token) private view returns (uint256) {
        return _vestedAmount(token).sub(_released[token]);
    }

    /**
     * @dev Calculates the amount that has already vested.
     * @param token ERC20 token which is being vested
     */
    function _vestedAmount(address token) private view returns (uint256) {
        uint256 currentBalance = tokenReward.balanceOf(address(this));
        uint256 totalBalance = currentBalance.add(_released[token]);

        if (block.timestamp < _cliff) {
            return 0;
        } else if (block.timestamp >= _start.add(_duration)) {
            return totalBalance;
```

```
        } else {
            return totalBalance.mul(block.timestamp.sub(_start)).div(_duration);
        }
    }
}
```

File FlowchainToken.sol

```
pragma solidity ^0.4.18;

/**
 * Copyright 2018, Flowchain.co
 *
 * The Flowchain tokens smart contract
 */

contract Mintable {
    function mintToken(address to, uint amount) external returns (bool success);
    function setupMintableAddress(address _mintable) public returns (bool success);
}

contract ApproveAndCallReceiver {
    function receiveApproval(address _from, uint256 _value, address _tokenContract,
        bytes _extraData);
}

contract Token {

    /// The total amount of tokens
    uint256 public totalSupply;

    /// @param _owner The address from which the balance will be retrieved
    /// @return The balance
    function balanceOf(address _owner) public view returns (uint256 balance);

    /// @notice send `_value` token to `_to` from `msg.sender`
    /// @param _to The address of the recipient
    /// @param _value The amount of token to be transferred
    /// @return Whether the transfer was successful or not
    function transfer(address _to, uint256 _value) public returns (bool success);

    /// @notice send `_value` token to `_to` from `_from` on the condition it is
        approved by `_from`
    /// @param _from The address of the sender
    /// @param _to The address of the recipient
    /// @param _value The amount of token to be transferred
    /// @return Whether the transfer was successful or not
    function transferFrom(address _from, address _to, uint256 _value) public returns (
        bool success);

    /// @notice `msg.sender` approves `_addr` to spend `_value` tokens
    /// @param _spender The address of the account able to transfer the tokens
    /// @param _value The amount of wei to be approved for transfer
    /// @return Whether the approval was successful or not
    function approve(address _spender, uint256 _value) public returns (bool success);

    /// @param _owner The address of the account owning tokens
    /// @param _spender The address of the account able to transfer the tokens
    /// @return Amount of remaining tokens allowed to spent
```

```
    function allowance(address _owner, address _spender) public view returns (uint256 remaining);

    event Transfer(address indexed _from, address indexed _to, uint256 _value);
    event Approval(address indexed _owner, address indexed _spender, uint256 _value);
}

contract StandardToken is Token {

    uint256 constant private MAX_UINT256 = 2**256 - 1;
    mapping (address => uint256) public balances;
    mapping (address => mapping (address => uint256)) public allowed;

    /*@CTK transfer
        @tag assume_completion
        @pre msg.sender != _to
        @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 success) {
        require(balances[msg.sender] >= _value);
        // Not overflow
        require(balances[_to] + _value >= balances[_to]);
        balances[msg.sender] -= _value;
        balances[_to] += _value;
        Transfer(msg.sender, _to, _value);
        return true;
    }

    /*@CTK transferFrom
        @tag assume_completion
        @pre allowed[_from][msg.sender] < MAX_UINT256
        @pre _from != _to
        @post __post.balances[_from] == balances[_from] - _value
        @post __post.balances[_to] == balances[_to] + _value
        @post __post.allowed[_from][msg.sender] == allowed[_from][msg.sender] - _value
    */
    function transferFrom(address _from, address _to, uint256 _value) public returns (bool success) {
        uint256 allowance = allowed[_from][msg.sender];
        require(balances[_from] >= _value && allowance >= _value);
        // Not overflow
        require(balances[_to] + _value >= balances[_to]);
        balances[_to] += _value;
        balances[_from] -= _value;
        if (allowance < MAX_UINT256) {
            allowed[_from][msg.sender] -= _value;
        }

        Transfer(_from, _to, _value);
        return true;
    }


    /*@CTK balanceOf
        @post balance == balances[_owner]
    */
    function balanceOf(address _owner) public view returns (uint256 balance) {
```

```
        return balances[_owner];
    }

    /*@CTK approve
        @post __post.allowed[msg.sender][_spender] == _value
     */
    function approve(address _spender, uint256 _value) public returns (bool success) {
        allowed[msg.sender][_spender] = _value;
        Approval(msg.sender, _spender, _value);
        return true;
    }

    /*@CTK allowance
        @post remaining == allowed[_owner][_spender]
    */
    function allowance(address _owner, address _spender) public view returns (uint256
        remaining) {
      return allowed[_owner][_spender];
    }
}


//name this contract whatever you'd like
contract FlowchainToken is StandardToken, Mintable {

    /* Public variables of the token */
    string public name = "FlowchainCoin";
    string public symbol = "FLC";
    uint8 public decimals = 18;
    string public version = "1.0";
    address public mintableAddress;
    address public multiSigWallet;
    address public creator;

    function() payable { revert(); }

    /*CTK FlowchainToken
      @post __post.creator == msg.sender
      @post __post.balances[multiSigWallet] == totalSupply
     */
    function FlowchainToken() public {
        // 1 billion tokens + 18 decimals
        totalSupply = 10**27;
        creator = msg.sender;
        mintableAddress = 0x9581973c54fce63d0f5c4c706020028af20ff723;
        multiSigWallet = 0x9581973c54fce63d0f5c4c706020028af20ff723;
        // Give the multisig wallet all initial tokens
        balances[multiSigWallet] = totalSupply;
        Transfer(0x0, multiSigWallet, totalSupply);
    }

    /*@CTK setupMintableAddress
      @tag assume_completion
      @post msg.sender == creator
      @post __post.mintableAddress == _mintable
     */
    function setupMintableAddress(address _mintable) public returns (bool success) {
        require(msg.sender == creator);
```

```
        mintableAddress = _mintable;
        return true;
    }

    /// @dev Mint an amount of tokens and transfer to the backer
    /// @param to The address of the backer who will receive the tokens
    /// @param amount The amount of rewarded tokens
    /// @return The result of token transfer
    /*@CTK mintToken
      @tag assume_completion
      @pre multiSigWallet != to
      @post msg.sender == mintableAddress
      @post balances[multiSigWallet] >= amount
      @post __post.balances[multiSigWallet] == balances[multiSigWallet] - amount
     */
    function mintToken(address to, uint256 amount) external returns (bool success) {
        require(msg.sender == mintableAddress);
        require(balances[multiSigWallet] >= amount);
        balances[multiSigWallet] -= amount;
        balances[to] += amount;
        Transfer(multiSigWallet, to, amount);
        return true;
    }

    /// @dev This function makes it easy to get the creator of the tokens
    /// @return The address of token creator
    /*@CTK getCreator
      @post __return == creator
     */
    function getCreator() constant returns (address) {
        return creator;
    }

    /// @dev This function makes it easy to get the mintableAddress
    /// @return The address of token creator
    /*@CTK getMintableAddress
      @post __return == mintableAddress
     */
    function getMintableAddress() constant returns (address) {
        return mintableAddress;
    }

    /* Approves and then calls the receiving contract */
    /*@CTK approveAndCall
      @post __post.allowed[msg.sender][_spender] == _value
     */
    function approveAndCall(address _spender, uint256 _value, bytes _extraData)
        external returns (bool success) {
        allowed[msg.sender][_spender] = _value;
        Approval(msg.sender, _spender, _value);

        //call the receiveApproval function on the contract you want to be notified.
            This crafts the function signature manually so one doesn't have to include
            a contract in here just for this.

        //ApproveAndCallReceiver(_spender).receiveApproval(msg.sender, _value, this,
            _extraData);

```

```
        return true;
    }
}
```

File Migrations.sol

```
pragma solidity ^0.4.18;

contract Migrations {
  address public owner;
  uint public last_completed_migration;

  modifier restricted() {
    if (msg.sender == owner) {
      _;
    }
  }

  /*@CTK migrations
    @post __post.owner == msg.sender
   */
  function Migrations() {
    owner = msg.sender;
  }

  /*@CTK setCompleted
    @pre msg.sender == owner
    @post __post.last_completed_migration == completed
   */
  function setCompleted(uint completed) restricted {
    last_completed_migration = completed;
  }

  function upgrade(address new_address) restricted {
    Migrations upgraded = Migrations(new_address);
    upgraded.setCompleted(last_completed_migration);
  }
}
```

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

INSECURE_COMPILER_VERSION

Line 1 in File VestingTokenSale.sol

```
1 pragma solidity 0.4.24;
```

Version to compile has the following bug: 0.4.24: ExpExponentCleanup, EventStructWrongData

TIMESTAMP_DEPENDENCY

Line 248 in File VestingTokenSale.sol

```
248         deadline = now + durationInMinutes * 1 minutes;
```

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 255 in File VestingTokenSale.sol

```
255     modifier afterDeadline() { if (now > deadline) _; }
```

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 256 in File VestingTokenSale.sol

```
256     modifier beforeDeadline() { if (now <= deadline) _; }
```

"now" can be influenced by minors to some degree

INSECURE_COMPILER_VERSION

Line 1 in File VestingDapp.sol

```
1 pragma solidity 0.4.24;
```

Version to compile has the following bug: 0.4.24: ExpExponentCleanup, EventStructWrongData

TIMESTAMP_DEPENDENCY

Line 221 in File VestingDapp.sol

```
221         require(start.add(duration) > block.timestamp);
```

"block.timestamp" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 313 in File VestingDapp.sol

```
313         if (block.timestamp < _cliff) {
```

"block.timestamp" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 315 in File VestingDapp.sol

```
315        } else if (block.timestamp >= _start.add(_duration)) {
```

"block.timestamp" can be influenced by minors to some degree

INSECURE_COMPILER_VERSION

Line 1 in File FlowchainToken.sol

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

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

INSECURE_COMPILER_VERSION

Line 1 in File Migrations.sol

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

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

Formal Verification Request 1

SafeMath mul

16, Jan 2019
380.94 ms

Line 22-27 in File VestingTokenSale.sol

```
22  /*@CTK "SafeMath mul"
23      @post (a > 0) && (((a * b) / a) != b) -> __reverted
24      @post __reverted -> (a > 0) && (((a * b) / a) != b)
25      @post !__reverted -> __return == a * b
26      @post !__reverted == !__has_overflow
27  */
```

Line 28-40 in File VestingTokenSale.sol

```
28  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
29      // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
30      // benefit is lost if 'b' is also tested.
31      // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
32      if (a == 0) {
33          return 0;
34      }
35
36      uint256 c = a * b;
37      require(c / a == b);
38
39      return c;
40  }
```

The code meets the specification

Formal Verification Request 2

SafeMath div

16, Jan 2019
16.2 ms

Line 64-68 in File VestingTokenSale.sol

```
64  /*@CTK "SafeMath div"
65      @post b != 0 -> !__reverted
66      @post !__reverted -> __return == a / b
67      @post !__reverted -> !__has_overflow
68  */
```

Line 69-76 in File VestingTokenSale.sol

```
69  function div(uint256 a, uint256 b) internal pure returns (uint256) {
70      // Solidity only automatically asserts when dividing by 0
71      require(b > 0);
72      uint256 c = a / b;
73      // assert(a == b * c + a % b); // There is no case in which this doesn't hold
74
```

```
        return c;
    }
```

The code meets the specification

Formal Verification Request 3

SafeMath sub

16, Jan 2019
14.8 ms

Line 93-97 in File VestingTokenSale.sol

```
93    /*@CTK "SafeMath sub"
94        @post (a < b) == __reverted
95        @post !__reverted -> __return == a - b
96        @post !__reverted -> !__has_overflow
97    */
```

Line 98-103 in File VestingTokenSale.sol

```
98    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
99        require(b <= a);
100       uint256 c = a - b;
101
102       return c;
103   }
```

The code meets the specification

Formal Verification Request 4

SafeMath add

16, Jan 2019
17.16 ms

Line 118-122 in File VestingTokenSale.sol

```
118   /*@CTK "SafeMath add"
119       @post (a + b < a || a + b < b) == __reverted
120       @post !__reverted -> __return == a + b
121       @post !__reverted -> !__has_overflow
122   */
```

Line 123-128 in File VestingTokenSale.sol

```
123   function add(uint256 a, uint256 b) internal pure returns (uint256) {
124       uint256 c = a + b;
125       require(c >= a);
126
127       return c;
128   }
```

The code meets the specification

Formal Verification Request 5

SafeMath mod

16, Jan 2019
14.4 ms

Line 144-148 in File VestingTokenSale.sol

```
144 /*@CTK "SafeMath mod"
145     @tag assume_completion
146     @post b != 0
147     @post __return == a % b
148  */
```

Line 149-152 in File VestingTokenSale.sol

```
149 function mod(uint256 a, uint256 b) internal pure returns (uint256) {
150     require(b != 0);
151     return a % b;
152 }
```

The code meets the specification

Formal Verification Request 6

createMintableSale

16, Jan 2019
131.69 ms

Line 227-238 in File VestingTokenSale.sol

```
227 /*@CTK createMintableSale
228     @tag assume_completion
229     @post msg.sender == creator
230     @post rate <= 6400 && rate >= 1
231     @post fundingGoalInEthers >= 1
232     @post durationInMinutes >= 60 minutes
233     @post __post.addressOfVestingApp == vestingAddrss
234     @post __post.deadline == now + durationInMinutes * 1 minutes
235     @post __post.fundingGoal == amountRaised + fundingGoalInEthers * 1 ether
236     @post __post.tokensPerEther == rate
237     @post __post.isFunding == true
238  */
```

Line 239-253 in File VestingTokenSale.sol

```
239 function createMintableSale(address vestingAddrss, uint256 rate, uint256
        fundingGoalInEthers, uint durationInMinutes) public returns (bool success) {
240     require(msg.sender == creator);
241     require(isFunding == false);
242     require(rate <= 6400 && rate >= 1);             // rate must be between 1 and
            6400
243     require(fundingGoalInEthers >= 1);
244     require(durationInMinutes >= 60 minutes);
245
```

```
        addressOfVestingApp = vestingAddrss;

        deadline = now + durationInMinutes * 1 minutes;
        fundingGoal = amountRaised + fundingGoalInEthers * 1 ether;
        tokensPerEther = rate;
        isFunding = true;
        return true;
    }
```

The code meets the specification

Formal Verification Request 7

setupAccreditedAddress

16, Jan 2019
23.98 ms

Line 261-265 in File VestingTokenSale.sol

```
    /*@CTK setupAccreditedAddress
        @tag assume_completion
        @post msg.sender == creator
        @post __post.accredited[_accredited] == _amountInEthers * 1 ether
     */
```

Line 266-270 in File VestingTokenSale.sol

```
    function setupAccreditedAddress(address _accredited, uint _amountInEthers) public
        returns (bool success) {
        require(msg.sender == creator);
        accredited[_accredited] = _amountInEthers * 1 ether;
        return true;
    }
```

The code meets the specification

Formal Verification Request 8

getAmountAccredited

16, Jan 2019
6.03 ms

Line 274-276 in File VestingTokenSale.sol

```
    /*@CTK getAmountAccredited
      @post __return == accredited[_accredited]
     */
```

Line 277-280 in File VestingTokenSale.sol

```
    function getAmountAccredited(address _accredited) view returns (uint256) {
        uint256 amount = accredited[_accredited];
        return amount;
    }
```

The code meets the specification

Formal Verification Request 9

closeSale

16, Jan 2019
28.98 ms

Line 282-287 in File VestingTokenSale.sol

```
282 /*@CTK closeSale
283   @tag assume_completion
284   @pre now <= deadline
285   @post msg.sender == creator
286   @post __post.isFunding == false
287  */
```

Line 288-291 in File VestingTokenSale.sol

```
288 function closeSale() beforeDeadline {
289     require(msg.sender == creator);
290     isFunding = false;
291 }
```

The code meets the specification

Formal Verification Request 10

changeCreator

16, Jan 2019
19.76 ms

Line 294-297 in File VestingTokenSale.sol

```
294 /*@CTK changeCreator
295   @tag assume_completion
296   @post __post.creator == _creator
297  */
```

Line 298-301 in File VestingTokenSale.sol

```
298 function changeCreator(address _creator) external {
299     require(msg.sender == creator);
300     creator = _creator;
301 }
```

The code meets the specification

Formal Verification Request 11

getRate

16, Jan 2019
9.63 ms

Line 305-308 in File VestingTokenSale.sol

```
305    /*@CTK getRate
306      @pre now <= deadline
307      @post __return == tokensPerEther
308     */
```

Line 309-311 in File VestingTokenSale.sol

```
309    function getRate() beforeDeadline view returns (uint) {
310        return tokensPerEther;
311    }
```

The code meets the specification

Formal Verification Request 12

getAmountRaised

16, Jan 2019
6.49 ms

Line 315-317 in File VestingTokenSale.sol

```
315    /*@CTK getAmountRaised
316      @post __return == amountRaised
317     */
```

Line 318-320 in File VestingTokenSale.sol

```
318    function getAmountRaised() view returns (uint) {
319        return amountRaised;
320    }
```

The code meets the specification

Formal Verification Request 13

SafeMath mul

16, Jan 2019
380.94 ms

Line 19-24 in File VestingDapp.sol

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

```
22        @post !__reverted -> __return == a * b
23        @post !__reverted == !__has_overflow
24    */
```

Line 25-37 in File VestingDapp.sol

```
25    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
26        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
27        // benefit is lost if 'b' is also tested.
28        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
29        if (a == 0) {
30            return 0;
31        }
32
33        uint256 c = a * b;
34        require(c / a == b);
35
36        return c;
37    }
```

The code meets the specification

Formal Verification Request 14

SafeMath div

16, Jan 2019
16.2 ms

Line 61-65 in File VestingDapp.sol

```
61    /*@CTK "SafeMath div"
62        @post b != 0 -> !__reverted
63        @post !__reverted -> __return == a / b
64        @post !__reverted -> !__has_overflow
65    */
```

Line 66-73 in File VestingDapp.sol

```
66    function div(uint256 a, uint256 b) internal pure returns (uint256) {
67        // Solidity only automatically asserts when dividing by 0
68        require(b > 0);
69        uint256 c = a / b;
70        // assert(a == b * c + a % b); // There is no case in which this doesn't hold
71
72        return c;
73    }
```

The code meets the specification

Formal Verification Request 15

SafeMath sub

16, Jan 2019
14.8 ms

Line 90-94 in File VestingDapp.sol

```
90  /*@CTK "SafeMath sub"
91      @post (a < b) == __reverted
92      @post !__reverted -> __return == a - b
93      @post !__reverted -> !__has_overflow
94  */
```

Line 95-100 in File VestingDapp.sol

```
95  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
96      require(b <= a);
97      uint256 c = a - b;
98
99      return c;
100 }
```

The code meets the specification

Formal Verification Request 16

SafeMath add

16, Jan 2019
17.16 ms

Line 115-119 in File VestingDapp.sol

```
115 /*@CTK "SafeMath add"
116     @post (a + b < a || a + b < b) == __reverted
117     @post !__reverted -> __return == a + b
118     @post !__reverted -> !__has_overflow
119 */
```

Line 120-125 in File VestingDapp.sol

```
120 function add(uint256 a, uint256 b) internal pure returns (uint256) {
121     uint256 c = a + b;
122     require(c >= a);
123
124     return c;
125 }
```

The code meets the specification

Formal Verification Request 17

SafeMath mod

16, Jan 2019
14.4 ms

Line 141-145 in File VestingDapp.sol

```
141     /*@CTK "SafeMath mod"
142         @tag assume_completion
143         @post b != 0
144         @post __return == a % b
145      */
```

Line 146-149 in File VestingDapp.sol

```
146     function mod(uint256 a, uint256 b) internal pure returns (uint256) {
147         require(b != 0);
148         return a % b;
149     }
```

✅ The code meets the specification

Formal Verification Request 18

Vesting

📅 16, Jan 2019
⏱ 7.76 ms

Line 188-190 in File VestingDapp.sol

```
188     /*@CTK Vesting
189         @post __post.creator == msg.sender
190      */
```

Line 191-193 in File VestingDapp.sol

```
191     function Vesting() payable {
192         creator = msg.sender;
193     }
```

✅ The code meets the specification

Formal Verification Request 19

createVestingPeriod

📅 16, Jan 2019
⏱ 170.47 ms

Line 205-216 in File VestingDapp.sol

```
205     /*@CTK createVestingPeriod
206         @tag assume_completion
207         @post msg.sender == creator
208         @post cliffDuration <= duration
209         @post duration > 0
210         @post start + duration > block.timestamp
211         @post __post._beneficiary == beneficiary
212         @post __post._duration == duration
213         @post __post._cliff == start + cliffDuration
214         @post __post._start == start
215         @post __post._addressOfTokenUsedAsReward == addressOfTokenUsedAsReward
216      */
```

Line 217-229 in File VestingDapp.sol

```
217    function createVestingPeriod(address beneficiary, uint256 start, uint256
           cliffDuration, uint256 duration, address addressOfTokenUsedAsReward) public {
218        require(msg.sender == creator);
219        require(cliffDuration <= duration);
220        require(duration > 0);
221        require(start.add(duration) > block.timestamp);
222
223        _beneficiary = beneficiary;
224        _duration = duration;
225        _cliff = start.add(cliffDuration);
226        _start = start;
227        _addressOfTokenUsedAsReward = addressOfTokenUsedAsReward;
228        tokenReward = Token(addressOfTokenUsedAsReward);
229    }
```

✅ The code meets the specification

Formal Verification Request 20

beneficiary

📅 16, Jan 2019
⏱ 6.83 ms

Line 234-236 in File VestingDapp.sol

```
234    /*@CTK beneficiary
235      @post __return == _beneficiary
236     */
```

Line 237-239 in File VestingDapp.sol

```
237    function beneficiary() public view returns (address) {
238        return _beneficiary;
239    }
```

✅ The code meets the specification

Formal Verification Request 21

cliff

📅 16, Jan 2019
⏱ 6.16 ms

Line 244-246 in File VestingDapp.sol

```
244    /*@CTK cliff
245      @post __return == _cliff
246     */
```

Line 247-249 in File VestingDapp.sol

```
    function cliff() public view returns (uint256) {
        return _cliff;
    }
```

The code meets the specification

Formal Verification Request 22

start

16, Jan 2019
5.81 ms

Line 254-256 in File VestingDapp.sol

```
    /*@CTK start
      @post __return == _start
     */
```

Line 257-259 in File VestingDapp.sol

```
    function start() public view returns (uint256) {
        return _start;
    }
```

The code meets the specification

Formal Verification Request 23

duration

16, Jan 2019
5.59 ms

Line 264-266 in File VestingDapp.sol

```
    /*@CTK duration
      @post __return == _duration
     */
```

Line 267-269 in File VestingDapp.sol

```
    function duration() public view returns (uint256) {
        return _duration;
    }
```

The code meets the specification

Formal Verification Request 24

released

16, Jan 2019
5.98 ms

Line 274-276 in File VestingDapp.sol

```
274    /*@CTK released
275      @post __return == _released[token]
276     */
```

Line 277-279 in File VestingDapp.sol

```
277    function released(address token) public view returns (uint256) {
278        return _released[token];
279    }
```

✅ The code meets the specification

Formal Verification Request 25

transfer

📅 16, Jan 2019
⏱ 122.67 ms

Line 61-66 in File FlowchainToken.sol

```
61    /*@CTK transfer
62        @tag assume_completion
63        @pre msg.sender != _to
64        @post __post.balances[msg.sender] == balances[msg.sender] - _value
65        @post __post.balances[_to] == balances[_to] + _value
66     */
```

Line 67-75 in File FlowchainToken.sol

```
67    function transfer(address _to, uint256 _value) public returns (bool success) {
68        require(balances[msg.sender] >= _value);
69        // Not overflow
70        require(balances[_to] + _value >= balances[_to]);
71        balances[msg.sender] -= _value;
72        balances[_to] += _value;
73        Transfer(msg.sender, _to, _value);
74        return true;
75    }
```

✅ The code meets the specification

Formal Verification Request 26

transferFrom

📅 16, Jan 2019
⏱ 291.27 ms

Line 77-84 in File FlowchainToken.sol

```
77  /*@CTK transferFrom
78      @tag assume_completion
79      @pre allowed[_from][msg.sender] < MAX_UINT256
80      @pre _from != _to
81      @post __post.balances[_from] == balances[_from] - _value
82      @post __post.balances[_to] == balances[_to] + _value
83      @post __post.allowed[_from][msg.sender] == allowed[_from][msg.sender] - _value
84  */
```

Line 85-98 in File FlowchainToken.sol

```
85  function transferFrom(address _from, address _to, uint256 _value) public returns (
        bool success) {
86      uint256 allowance = allowed[_from][msg.sender];
87      require(balances[_from] >= _value && allowance >= _value);
88      // Not overflow
89      require(balances[_to] + _value >= balances[_to]);
90      balances[_to] += _value;
91      balances[_from] -= _value;
92      if (allowance < MAX_UINT256) {
93          allowed[_from][msg.sender] -= _value;
94      }
95
96      Transfer(_from, _to, _value);
97      return true;
98  }
```

The code meets the specification

Formal Verification Request 27

balanceOf

16, Jan 2019
5.83 ms

Line 101-103 in File FlowchainToken.sol

```
101 /*@CTK balanceOf
102     @post balance == balances[_owner]
103 */
```

Line 104-106 in File FlowchainToken.sol

```
104 function balanceOf(address _owner) public view returns (uint256 balance) {
105     return balances[_owner];
106 }
```

The code meets the specification

Formal Verification Request 28

approve

16, Jan 2019
11.1 ms

Line 108-110 in File FlowchainToken.sol

```
108     /*@CTK approve
109        @post __post.allowed[msg.sender][_spender] == _value
110      */
```

Line 111-115 in File FlowchainToken.sol

```
111     function approve(address _spender, uint256 _value) public returns (bool success) {
112        allowed[msg.sender][_spender] = _value;
113        Approval(msg.sender, _spender, _value);
114        return true;
115     }
```

The code meets the specification

Formal Verification Request 29

allowance

16, Jan 2019
6.22 ms

Line 117-119 in File FlowchainToken.sol

```
117     /*@CTK allowance
118        @post remaining == allowed[_owner][_spender]
119     */
```

Line 120-122 in File FlowchainToken.sol

```
120     function allowance(address _owner, address _spender) public view returns (uint256
           remaining) {
121       return allowed[_owner][_spender];
122     }
```

The code meets the specification

Formal Verification Request 30

setupMintableAddress

16, Jan 2019
20.85 ms

Line 155-159 in File FlowchainToken.sol

```
155     /*@CTK setupMintableAddress
156       @tag assume_completion
157       @post msg.sender == creator
158       @post __post.mintableAddress == _mintable
159      */
```

Line 160-164 in File FlowchainToken.sol

```
    function setupMintableAddress(address _mintable) public returns (bool success) {
        require(msg.sender == creator);
        mintableAddress = _mintable;
        return true;
    }
```

The code meets the specification

Formal Verification Request 31

mintToken

16, Jan 2019
86.77 ms

Line 170-176 in File FlowchainToken.sol

```
    /*@CTK mintToken
      @tag assume_completion
      @pre multiSigWallet != to
      @post msg.sender == mintableAddress
      @post balances[multiSigWallet] >= amount
      @post __post.balances[multiSigWallet] == balances[multiSigWallet] - amount
     */
```

Line 177-184 in File FlowchainToken.sol

```
    function mintToken(address to, uint256 amount) external returns (bool success) {
        require(msg.sender == mintableAddress);
        require(balances[multiSigWallet] >= amount);
        balances[multiSigWallet] -= amount;
        balances[to] += amount;
        Transfer(multiSigWallet, to, amount);
        return true;
    }
```

The code meets the specification

Formal Verification Request 32

getCreator

16, Jan 2019
6.05 ms

Line 188-190 in File FlowchainToken.sol

```
    /*@CTK getCreator
      @post __return == creator
     */
```

Line 191-193 in File FlowchainToken.sol

```
    function getCreator() constant returns (address) {
        return creator;
    }
```

The code meets the specification

Formal Verification Request 33

getMintableAddress

16, Jan 2019
6.0 ms

Line 197-199 in File FlowchainToken.sol

```
197 /*@CTK getMintableAddress
198   @post __return == mintableAddress
199  */
```

Line 200-202 in File FlowchainToken.sol

```
200 function getMintableAddress() constant returns (address) {
201     return mintableAddress;
202 }
```

The code meets the specification

Formal Verification Request 34

approveAndCall

16, Jan 2019
13.74 ms

Line 205-207 in File FlowchainToken.sol

```
205 /*@CTK approveAndCall
206   @post __post.allowed[msg.sender][_spender] == _value
207  */
```

Line 208-217 in File FlowchainToken.sol

```
208 function approveAndCall(address _spender, uint256 _value, bytes _extraData)
        external returns (bool success) {
209     allowed[msg.sender][_spender] = _value;
210     Approval(msg.sender, _spender, _value);
211
212     //call the receiveApproval function on the contract you want to be notified.
            This crafts the function signature manually so one doesn't have to include
            a contract in here just for this.
213
214     //ApproveAndCallReceiver(_spender).receiveApproval(msg.sender, _value, this,
            _extraData);
215
216     return true;
217 }
```

The code meets the specification

Formal Verification Request 35

migrations

16, Jan 2019
6.09 ms

Line 13-15 in File Migrations.sol

```
13 /*@CTK migrations
14   @post __post.owner == msg.sender
15  */
```

Line 16-18 in File Migrations.sol

```
16 function Migrations() {
17   owner = msg.sender;
18 }
```

The code meets the specification

Formal Verification Request 36

setCompleted

16, Jan 2019
10.97 ms

Line 20-23 in File Migrations.sol

```
20 /*@CTK setCompleted
21   @pre msg.sender == owner
22   @post __post.last_completed_migration == completed
23  */
```

Line 24-26 in File Migrations.sol

```
24 function setCompleted(uint completed) restricted {
25   last_completed_migration = completed;
26 }
```

The code meets the specification