CertiK Verification Report For Mycro

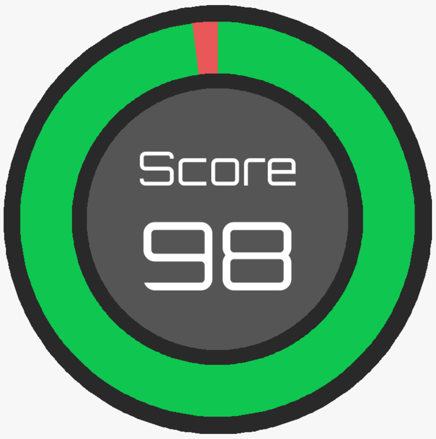
Request Date: 2019-04-16
Revision Date: 2019-04-18

Disclaimer

PASS

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

Apr 18, 2019



Summary

This audit report summarises the smart contract verification service requested by Mycro. 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.	2	SWC-116

Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	1	SWC-102 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

Seconds is the common unit of timestamp in Solidity

In new function `extendPrivateSaleDuration`, `extentionInDays.mul(1 days)` will be converted to seconds instead of milliseconds. Comments might need to be updated.

Naming of `mainSaleDurantionExtentionLimit`

Can be more explicit to be able to indicate that this is in `Days`. For example, One can rename it to `mainSaleDurationExtentionLimitInDay`. Also, there is typo in the variable name (`Durantion` should be `Duration`).

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 Token/ICOToken.sol

```
pragma solidity ^0.4.23;

import "zeppelin-solidity/contracts/token/ERC20/MintableToken.sol";
import "zeppelin-solidity/contracts/token/ERC20/PausableToken.sol";
import "zeppelin-solidity/contracts/token/ERC20/BurnableToken.sol";


/**
 * @title ICOToken
 * @dev Very simple ERC20 Token example.
 * `StandardToken` functions.
 */
contract ICOToken is MintableToken, PausableToken, BurnableToken {

    string public constant name = "Mycro Token";
    string public constant symbol = "MYO";
    uint8 public constant decimals = 18;


    /**
     * @dev Constructor that gives msg.sender all of existing tokens.
     */
    /*@CTK ICOToken
      @post !__reverted
     */
    constructor() public {
    }
}
```

File Crowdsale/BasicCrowdsale.sol

```
pragma solidity ^0.4.24;

import "zeppelin-solidity/contracts/crowdsale/emission/MintedCrowdsale.sol";
import "zeppelin-solidity/contracts/crowdsale/distribution/FinalizableCrowdsale.sol";


contract BasicCrowdsale is MintedCrowdsale, FinalizableCrowdsale {

    uint256 public cap = 100000000 * (10 ** 18); // Total number of MYO tokens that
        would be created
    uint256 public capForSale = 71000000 * (10 ** 18); // Total MYO tokens that could
        be sold during the ICO
    uint256 public bountyTokensCap = 5000000 * (10 ** 18); // Total number of MYO
        tokens that would be given as a reward
    uint256 public reservedForTeamTokens = 29000000 * (10 ** 18); // Tokens reserved
        for rewardpool, advisors and team that will be minted after Crowdsale
    uint256 public totalMintedBountyTokens; // Total number of MYO tokens given as a
        reward

    uint256 public privateSaleEndDate;
    mapping (address => bool) public minters;

    uint256 constant MIN_CONTRIBUTION_AMOUNT = 10 finney;
    uint256 constant MAX_CONTRIBUTION_AMOUNT = 250 ether;

    uint256 public constant PRIVATE_SALE_CAP = 26000000 * (10 ** 18);
    uint256 public constant PRIVATE_SALE_DURATION = 30 days; // to be calculated
        according to deployment day; the end date should be 30 May

    uint256 public constant MAIN_SALE_DURATION = 60 days;
    uint256 public mainSaleDurantionExtentionLimit = 60; //max days the duration of
        the ICO can be extended

    event LogFiatTokenMinted(address sender, address beficiary, uint256 amount);
    event LogFiatTokenMintedToMany(address sender, address[] beneficiaries, uint256[]
        amount);
    event LogBountyTokenMinted(address minter, address beneficiary, uint256 amount);
    event LogBountyTokenMintedToMany(address sender, address[] beneficiaries, uint256
        [] amount);
    event LogPrivateSaleExtended(uint256 extentionInDays);
    event LogMainSaleExtended(uint256 extentionInDays);
    event LogRateChanged(uint256 rate);
    event LogMinterAdded(address minterAdded);
    event LogMinterRemoved(address minterRemoved);

    constructor(uint256 _rate, address _wallet, address _token, uint256 _openingTime,
        uint256 _closingTime)
    Crowdsale(_rate, _wallet, ERC20(_token))
    TimedCrowdsale(_openingTime, _closingTime) public {
        minters[owner] = true;
        privateSaleEndDate = _openingTime.add(PRIVATE_SALE_DURATION);
    }

    // only addresses who are allowed to mint
    modifier onlyMinter (){
        require(minters[msg.sender]);
        _;
    }

    function buyTokens(address beneficiary) public payable {
        require(msg.value >= MIN_CONTRIBUTION_AMOUNT);
        require(msg.value <= MAX_CONTRIBUTION_AMOUNT);
        if(now <= privateSaleEndDate) {
            require(MintableToken(token).totalSupply() < PRIVATE_SALE_CAP);
        }
        uint amount = _getTokenAmount(msg.value);
        require(MintableToken(token).totalSupply().add(amount) <= capForSale);
        super.buyTokens(beneficiary);
    }

    /*@CTK addMinter
      @tag assume_completion
      @post owner == msg.sender
      @post _minter != address(0)
      @post __post.minters[_minter]
     */
    function addMinter(address _minter) public onlyOwner {
        require(_minter != address(0));
        minters[_minter] = true;
        emit LogMinterAdded(_minter);
    }

    /*@CTK removeMinter
      @tag assume_completion
      @post owner == msg.sender
      @post !__post.minters[_minter]
     */
    function removeMinter(address _minter) public onlyOwner {
        minters[_minter] = false;
        emit LogMinterRemoved(_minter);
    }

    /*@CTK createFiatToken
      @tag assume_completion
      @post block.timestamp <= closingTime
     */
    function createFiatToken(address beneficiary, uint256 amount) public onlyMinter()
        returns(bool){
        require(!hasClosed());
        mintFiatToken(beneficiary, amount);
        emit LogFiatTokenMinted(msg.sender, beneficiary, amount);
        return true;
    }

    function createFiatTokenToMany(address[] beneficiaries, uint256[] amount) public
        onlyMinter() returns(bool){
        multiBeneficiariesValidation(beneficiaries, amount);
        for(uint i = 0; i < beneficiaries.length; i++){
            mintFiatToken(beneficiaries[i], amount[i]);
        }
        emit LogFiatTokenMintedToMany(msg.sender, beneficiaries, amount);
        return true;
    }

    function mintFiatToken(address beneficiary, uint256 amount) internal {
        require(MintableToken(token).totalSupply().add(amount) <= capForSale);
        MintableToken(token).mint(beneficiary, amount);
    }

    /*@CTK createBountyToken
      @tag assume_completion
      @post block.timestamp <= closingTime
     */
    function createBountyToken(address beneficiary, uint256 amount) public onlyMinter
        () returns (bool) {
        require(!hasClosed());
        mintBountyToken(beneficiary, amount);
        emit LogBountyTokenMinted(msg.sender, beneficiary, amount);
        return true;
    }

    function createBountyTokenToMany(address[] beneficiaries, uint256[] amount) public
         onlyMinter() returns (bool) {
        multiBeneficiariesValidation(beneficiaries, amount);
        for(uint i = 0; i < beneficiaries.length; i++){
            mintBountyToken(beneficiaries[i], amount[i]);
        }

        emit LogBountyTokenMintedToMany(msg.sender, beneficiaries, amount);
        return true;
    }

    function mintBountyToken(address beneficiary, uint256 amount) internal {
        require(MintableToken(token).totalSupply().add(amount) <= capForSale);
        require(totalMintedBountyTokens.add(amount) <= bountyTokensCap);
        MintableToken(token).mint(beneficiary, amount);
        totalMintedBountyTokens = totalMintedBountyTokens.add(amount);
    }

    /*@CTK multiBeneficiariesValidation
      @tag assume_completion
      @post block.timestamp <= closingTime
      @post beneficiaries.length > 0
      @post amount.length > 0
      @post beneficiaries.length == amount.length
     */
    function multiBeneficiariesValidation(address[] beneficiaries, uint256[] amount) internal view {
        require(!hasClosed());
        require(beneficiaries.length > 0);
        require(amount.length > 0);
        require(beneficiaries.length == amount.length);
    }

    /**
        @param extentionInDays is a simple number of the days, e.c. 3 => 3 days
     */
    /*@CTK extendPrivateSaleDuration
      @tag assume_completion
      @post owner == msg.sender
      @post __post.privateSaleEndDate == privateSaleEndDate + extentionInDays * 86400
      @post __post.closingTime == closingTime + extentionInDays * 86400
     */
    function extendPrivateSaleDuration(uint256 extentionInDays) public onlyOwner returns (bool) {
        require(now <= privateSaleEndDate);
        extentionInDays = extentionInDays.mul(1 days); // convert the days in milliseconds
        privateSaleEndDate = privateSaleEndDate.add(extentionInDays);
        closingTime = closingTime.add(extentionInDays);
        emit LogPrivateSaleExtended(extentionInDays);
        return true;
    }

    /**
        @param extentionInDays is a simple number of the days, e.c. 3 => 3 days
     */
    /*@CTK extendMainSailDuration
      @tag assume_completion
      @post now > privateSaleEndDate
      @post block.timestamp <= closingTime
      @post mainSaleDurantionExtentionLimit - extentionInDays >= 0
      @post owner == msg.sender
      @post __post.mainSaleDurantionExtentionLimit == mainSaleDurantionExtentionLimit - extentionInDays
      @post __post.closingTime == closingTime + extentionInDays * 86400
     */
    function extendMainSailDuration(uint256 extentionInDays) public onlyOwner returns
```

```
        (bool) {
        require(now > privateSaleEndDate);
        require(!hasClosed());
        require(mainSaleDurantionExtentionLimit.sub(extentionInDays) >= 0);

        uint256 extention = extentionInDays.mul(1 days); // convert the days in
            milliseconds
        mainSaleDurantionExtentionLimit = mainSaleDurantionExtentionLimit.sub(
            extentionInDays); // substract days from the limit
        closingTime = closingTime.add(extention);

        emit LogMainSaleExtended(extentionInDays);
        return true;
    }

    /*@CTK changeRate
      @tag assume_completion
      @post owner == msg.sender
      @post block.timestamp <= closingTime
      @post _newRate != 0
      @post __post.rate == _newRate
     */
    function changeRate(uint _newRate) public onlyOwner returns (bool) {
        require(!hasClosed());
        require(_newRate != 0);
        rate = _newRate;
        emit LogRateChanged(_newRate);
        return true;
    }

    // after finalization will be minted manually reservedForTeamTokens amount
    function finalization() internal {
        MintableToken(token).transferOwnership(owner);
        super.finalization();
    }
}
```

File Crowdsale/WhitelistedBasicCrowdsale.sol

```
pragma solidity ^0.4.24;

import "./BasicCrowdsale.sol";
import "./MultipleWhitelistedCrowdsale.sol";


contract WhitelistedBasicCrowdsale is BasicCrowdsale, MultipleWhitelistedCrowdsale {


  /*@CTK WhitelistedBasicCrowdsale
    @tag assume_completion
    @post !__reverted
   */
    constructor(uint256 _rate, address _wallet, address _token, uint256 _openingTime,
        uint256 _closingTime)
    // BasicCrowdsale(_rate, _wallet, ERC20(_token), _openingTime, _closingTime)
    // MultipleWhitelistedCrowdsale()
    public {
    }
}
```

File Crowdsale/MultipleWhitelistedCrowdsale.sol

```
pragma solidity ^0.4.24;

import "zeppelin-solidity/contracts/crowdsale/Crowdsale.sol";
import "zeppelin-solidity/contracts/ownership/Ownable.sol";

/**
 * @title MultipleWhitelistedCrowdsale
 * @dev Crowdsale in which only whitelisted users can contribute.
 */
contract MultipleWhitelistedCrowdsale is Crowdsale, Ownable {

  mapping(address => bool) public whitelist;
  // keeps all addresses who can manage the whitelist
  mapping(address => bool) public whitelistManagers;

  /**
   * @dev Reverts if beneficiary is not whitelisted. Can be used when extending this
       contract.
   */
  modifier isWhitelisted(address _beneficiary) {
    require(whitelist[_beneficiary]);
    _;
  }

  /**
   * @dev Reverts if msg.sender is not whitelist manager
   */
  modifier onlyWhitelistManager(){
      require(whitelistManagers[msg.sender]);
      _;
  }

  /**
   * @dev Adds single address who can manage the whitelist.
   * @param _manager Address to be added to the whitelistManagers
   */
  /*@CTK addWhitelistManager
    @tag assume_completion
    @post owner == msg.sender
    @post _manager != address(0)
    @post __post.whitelistManagers[_manager]
   */
  function addWhitelistManager(address _manager) public onlyOwner {
      require(_manager != address(0));
      whitelistManagers[_manager] = true;
  }

  /**
  * @param _manager Address to remove from whitelistManagers
   */
  /*@CTK removeWhitelistManager
    @tag assume_completion
    @post owner == msg.sender
    @post !__post.whitelistManagers[_manager]
   */
  function removeWhitelistManager(address _manager) public onlyOwner {
      whitelistManagers[_manager] = false;
  }

  /**
   * @dev Adds single address to whitelist.
   * @param _beneficiary Address to be added to the whitelist
   */
  /*@CTK addToWhitelist
    @tag assume_completion
    @post whitelistManagers[msg.sender]
    @post __post.whitelist[_beneficiary]
   */
  function addToWhitelist(address _beneficiary) external onlyWhitelistManager() {
    whitelist[_beneficiary] = true;
  }

  /**
   * @dev Adds list of addresses to whitelist. Not overloaded due to limitations with
       truffle testing.
   * @param _beneficiaries Addresses to be added to the whitelist
   */
  function addManyToWhitelist(address[] _beneficiaries) external onlyWhitelistManager
      () {
    for (uint256 i = 0; i < _beneficiaries.length; i++) {
      whitelist[_beneficiaries[i]] = true;
    }
  }

  /**
   * @dev Removes single address from whitelist.
   * @param _beneficiary Address to be removed to the whitelist
   */
  /*@CTK removeFromWhitelist
    @tag assume_completion
    @post whitelistManagers[msg.sender]
    @post !__post.whitelist[_beneficiary]
   */
  function removeFromWhitelist(address _beneficiary) external onlyWhitelistManager() {
    whitelist[_beneficiary] = false;
  }

  /**
   * @dev Extend parent behavior requiring beneficiary to be in whitelist.
   * @param _beneficiary Token beneficiary
   * @param _weiAmount Amount of wei contributed
   */
  function _preValidatePurchase(address _beneficiary, uint256 _weiAmount) internal
      isWhitelisted(_beneficiary) {
    super._preValidatePurchase(_beneficiary, _weiAmount);
  }

}
```

File zeppelin-solidity/contracts/token/ERC20/StandardToken.sol

```
pragma solidity ^0.4.21;

import "./BasicToken.sol";
import "./ERC20.sol";

```

```

/**
 * @title Standard ERC20 token
 *
 * @dev Implementation of the basic standard token.
 * @dev https://github.com/ethereum/EIPs/issues/20
 * @dev Based on code by FirstBlood: https://github.com/Firstbloodio/token/blob/master
    /smart_contract/FirstBloodToken.sol
 */
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 __return == true
    @post __post.balances[_to] == balances[_to] + _value
    @post __post.balances[_from] == balances[_from] - _value
    @post __has_overflow == false
  */
  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.
   *
   * Beware that changing an allowance with this method brings the risk that someone
      may use both the old
   * and the new allowance by unfortunate transaction ordering. One possible solution
      to mitigate this
   * race condition is to first reduce the spender's allowance to 0 and set the
      desired value afterwards:
   * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
   * @param _spender The address which will spend the funds.
   * @param _value The amount of tokens to be spent.
   */
  /*@CTK approve_success
    @post _value == 0 -> __reverted == false
    @post allowed[msg.sender][_spender] == 0 -> __reverted == false
```

```
*/
/*@CTK approve
  @tag assume_completion
  @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 get_allowance
  @post __reverted == false
  @post __return == allowed[_owner][_spender]
  @post this == __post
*/
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)
 * From MonolithDAO Token.sol
 * @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)
 * From MonolithDAO Token.sol
 * @param _spender The address which will spend the funds.
 * @param _subtractedValue The amount of tokens to decrease the allowance by.
```

```
   */
  /*@CTK "decreaseApproval correctness case 1"
    @pre allowed[msg.sender][_spender] < _subtractedValue
    @post __post.allowed[msg.sender][_spender] == 0
    @post __return == true
    @post (!__has_assertion_failure)
  */
  /*@CTK "decreaseApproval correctness case 2"
    @pre allowed[msg.sender][_spender] >= _subtractedValue
    @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] -
        _subtractedValue
    @post __return == true
    @post (!__has_assertion_failure)
  */
  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;
  }

}
```

File zeppelin-solidity/contracts/token/ERC20/MintableToken.sol

```
pragma solidity ^0.4.21;

import "./StandardToken.sol";
import "../../ownership/Ownable.sol";


/**
 * @title Mintable token
 * @dev Simple ERC20 Token example, with mintable token creation
 * @dev Issue: * https://github.com/OpenZeppelin/openzeppelin-solidity/issues/120
 * Based on code by TokenMarketNet: https://github.com/TokenMarketNet/ico/blob/master/
     contracts/MintableToken.sol
 */
contract MintableToken is StandardToken, Ownable {
  event Mint(address indexed to, uint256 amount);
  event MintFinished();

  bool public mintingFinished = false;


  modifier canMint() {
    require(!mintingFinished);
    _;
  }

  /**
   * @dev Function to mint tokens
   * @param _to The address that will receive the minted tokens.
   * @param _amount The amount of tokens to mint.
   * @return A boolean that indicates if the operation was successful.
   */
  /*@CTK mint
    @tag assume_completion
    @post __post.totalSupply_ == totalSupply_ + _amount
    @post __post.balances[_to] == balances[_to] + _amount
   */
  function mint(address _to, uint256 _amount) onlyOwner canMint public returns (bool)
      {
    totalSupply_ = totalSupply_.add(_amount);
    balances[_to] = balances[_to].add(_amount);
    emit Mint(_to, _amount);
    emit Transfer(address(0), _to, _amount);
    return true;
  }

  /**
   * @dev Function to stop minting new tokens.
   * @return True if the operation was successful.
   */
  /*@CTK finishMinting
    @tag assume_completion
    @post owner == msg.sender
    @post !mintingFinished
    @post __post.mintingFinished
   */
  function finishMinting() onlyOwner canMint public returns (bool) {
    mintingFinished = true;
    emit MintFinished();
    return true;
  }
}
```

File zeppelin-solidity/contracts/token/ERC20/BasicToken.sol

```
pragma solidity ^0.4.21;


import "./ERC20Basic.sol";
import "../../math/SafeMath.sol";


/**
 * @title Basic token
 * @dev Basic version of StandardToken, with no allowances.
 */
contract BasicToken is ERC20Basic {
  using SafeMath for uint256;

  mapping(address => uint256) balances;

  uint256 totalSupply_;

  /**
  * @dev total number of tokens in existence
  */
  /*@CTK totalSupply
    @post __return == totalSupply_
   */
  function totalSupply() public view returns (uint256) {
    return totalSupply_;
  }

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

    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 __reverted == false
    @post __return == balances[_owner]
  */
  function balanceOf(address _owner) public view returns (uint256) {
    return balances[_owner];
  }

}
```

File zeppelin-solidity/contracts/token/ERC20/BurnableToken.sol

```
pragma solidity ^0.4.21;

import "./BasicToken.sol";

```

```

/**
 * @title Burnable Token
 * @dev Token that can be irreversibly burned (destroyed).
 */
contract BurnableToken is BasicToken {

  event Burn(address indexed burner, uint256 value);

  /**
   * @dev Burns a specific amount of tokens.
   * @param _value The amount of token to be burned.
   */
  /*@CTK _burn
    @tag assume_completion
    @post _value <= balances[msg.sender]
    @post __post.balances[msg.sender] == balances[msg.sender] - _value
    @post __post.totalSupply_ == totalSupply_ - _value
   */
  function burn(uint256 _value) public {
    _burn(msg.sender, _value);
  }

  /*@CTK _burn
    @tag assume_completion
    @post _value <= balances[_who]
    @post __post.balances[_who] == balances[_who] - _value
    @post __post.totalSupply_ == totalSupply_ - _value
   */
  function _burn(address _who, uint256 _value) internal {
    require(_value <= balances[_who]);
    // no need to require value <= totalSupply, since that would imply the
    // sender's balance is greater than the totalSupply, which *should* be an
        assertion failure

    balances[_who] = balances[_who].sub(_value);
    totalSupply_ = totalSupply_.sub(_value);
    emit Burn(_who, _value);
    emit Transfer(_who, address(0), _value);
  }
}
```

File zeppelin-solidity/contracts/lifecycle/Pausable.sol

```
pragma solidity ^0.4.21;


import "../ownership/Ownable.sol";


/**
 * @title Pausable
 * @dev Base contract which allows children to implement an emergency stop mechanism.
 */
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 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 owner == msg.sender
      @post __post.paused == false
  */
  function unpause() onlyOwner whenPaused public {
    paused = false;
    emit Unpause();
  }
}
```

File zeppelin-solidity/contracts/crowdsale/Crowdsale.sol

```
pragma solidity ^0.4.21;

import "../token/ERC20/ERC20.sol";
import "../math/SafeMath.sol";


/**
 * @title Crowdsale
 * @dev Crowdsale is a base contract for managing a token crowdsale,
 * allowing investors to purchase tokens with ether. This contract implements
 * such functionality in its most fundamental form and can be extended to provide
     additional
 * functionality and/or custom behavior.
 * The external interface represents the basic interface for purchasing tokens, and
     conform
 * the base architecture for crowdsales. They are *not* intended to be modified /
     overriden.
 * The internal interface conforms the extensible and modifiable surface of crowdsales
     . Override
 * the methods to add functionality. Consider using 'super' where appropiate to
     concatenate
 * behavior.
 */
contract Crowdsale {
  using SafeMath for uint256;

  // The token being sold
  ERC20 public token;

  // Address where funds are collected
  address public wallet;

  // How many token units a buyer gets per wei
  uint256 public rate;

  // Amount of wei raised
  uint256 public weiRaised;

  /**
   * Event for token purchase logging
   * @param purchaser who paid for the tokens
   * @param beneficiary who got the tokens
   * @param value weis paid for purchase
   * @param amount amount of tokens purchased
   */
  event TokenPurchase(address indexed purchaser, address indexed beneficiary, uint256
      value, uint256 amount);

  /**
   * @param _rate Number of token units a buyer gets per wei
   * @param _wallet Address where collected funds will be forwarded to
   * @param _token Address of the token being sold
   */
  /*@CTK Crowdsale
    @tag assume_completion
    @post _rate > 0
    @post _wallet != address(0)
    @post _token != address(0)
    @post __post.rate == _rate
    @post __post.wallet == _wallet
    @post __post.token == _token
   */
  function Crowdsale(uint256 _rate, address _wallet, ERC20 _token) public {
    require(_rate > 0);
    require(_wallet != address(0));
    require(_token != address(0));

    rate = _rate;
    wallet = _wallet;
    token = _token;
```

```
}

// -----------------------------------------
// Crowdsale external interface
// -----------------------------------------

/**
 * @dev fallback function ***DO NOT OVERRIDE***
 */
function () external payable {
  buyTokens(msg.sender);
}

/**
 * @dev low level token purchase ***DO NOT OVERRIDE***
 * @param _beneficiary Address performing the token purchase
 */
function buyTokens(address _beneficiary) public payable {

  uint256 weiAmount = msg.value;
  _preValidatePurchase(_beneficiary, weiAmount);

  // calculate token amount to be created
  uint256 tokens = _getTokenAmount(weiAmount);

  // update state
  weiRaised = weiRaised.add(weiAmount);

  _processPurchase(_beneficiary, tokens);
  emit TokenPurchase(
    msg.sender,
    _beneficiary,
    weiAmount,
    tokens
  );

  _updatePurchasingState(_beneficiary, weiAmount);

  _forwardFunds();
  _postValidatePurchase(_beneficiary, weiAmount);
}

// -----------------------------------------
// Internal interface (extensible)
// -----------------------------------------

/**
 * @dev Validation of an incoming purchase. Use require statements to revert state
     when conditions are not met. Use super to concatenate validations.
 * @param _beneficiary Address performing the token purchase
 * @param _weiAmount Value in wei involved in the purchase
 */
/*@CTK _preValidatePurchase
  @tag assume_completion
  @post _beneficiary != address(0)
  @post _weiAmount != 0
 */
function _preValidatePurchase(address _beneficiary, uint256 _weiAmount) internal {
  require(_beneficiary != address(0));
  require(_weiAmount != 0);
}

/**
 * @dev Validation of an executed purchase. Observe state and use revert statements
     to undo rollback when valid conditions are not met.
 * @param _beneficiary Address performing the token purchase
 * @param _weiAmount Value in wei involved in the purchase
 */
function _postValidatePurchase(address _beneficiary, uint256 _weiAmount) internal {
  // optional override
}

/**
 * @dev Source of tokens. Override this method to modify the way in which the
     crowdsale ultimately gets and sends its tokens.
 * @param _beneficiary Address performing the token purchase
 * @param _tokenAmount Number of tokens to be emitted
 */
function _deliverTokens(address _beneficiary, uint256 _tokenAmount) internal {
  token.transfer(_beneficiary, _tokenAmount);
}

/**
 * @dev Executed when a purchase has been validated and is ready to be executed. Not
      necessarily emits/sends tokens.
 * @param _beneficiary Address receiving the tokens
 * @param _tokenAmount Number of tokens to be purchased
 */
function _processPurchase(address _beneficiary, uint256 _tokenAmount) internal {
  _deliverTokens(_beneficiary, _tokenAmount);
}

/**
 * @dev Override for extensions that require an internal state to check for validity
      (current user contributions, etc.)
 * @param _beneficiary Address receiving the tokens
 * @param _weiAmount Value in wei involved in the purchase
 */
function _updatePurchasingState(address _beneficiary, uint256 _weiAmount) internal {
  // optional override
}

/**
 * @dev Override to extend the way in which ether is converted to tokens.
 * @param _weiAmount Value in wei to be converted into tokens
 * @return Number of tokens that can be purchased with the specified _weiAmount
 */
/*@CTK _getTokenAmount
  @post __return == _weiAmount * rate
 */
function _getTokenAmount(uint256 _weiAmount) internal view returns (uint256) {
  return _weiAmount.mul(rate);
}

/**
 * @dev Determines how ETH is stored/forwarded on purchases.
 */
function _forwardFunds() internal {
  wallet.transfer(msg.value);
}
}
```

File zeppelin-solidity/contracts/crowdsale/distribution/FinalizableCrowdsale.sol

```
pragma solidity ^0.4.21;

import "../../math/SafeMath.sol";
import "../../ownership/Ownable.sol";
import "../validation/TimedCrowdsale.sol";


/**
 * @title FinalizableCrowdsale
 * @dev Extension of Crowdsale where an owner can do extra work
 * after finishing.
 */
contract FinalizableCrowdsale is TimedCrowdsale, Ownable {
  using SafeMath for uint256;

  bool public isFinalized = false;

  event Finalized();

  /**
   * @dev Must be called after crowdsale ends, to do some extra finalization
   * work. Calls the contract's finalization function.
   */
  /*@CTK finalize
    @tag assume_completion
    @post !isFinalized
    @post block.timestamp > closingTime
    @post __post.isFinalized
   */
  function finalize() onlyOwner public {
    require(!isFinalized);
    require(hasClosed());

    finalization();
    emit Finalized();

    isFinalized = true;
  }

  /**
   * @dev Can be overridden to add finalization logic. The overriding function
   * should call super.finalization() to ensure the chain of finalization is
   * executed entirely.
   */
  function finalization() internal {
  }

}
```

File zeppelin-solidity/contracts/crowdsale/validation/TimedCrowdsale.sol

```
pragma solidity ^0.4.21;
```

```

import "../../math/SafeMath.sol";
import "../Crowdsale.sol";


/**
 * @title TimedCrowdsale
 * @dev Crowdsale accepting contributions only within a time frame.
 */
contract TimedCrowdsale is Crowdsale {
  using SafeMath for uint256;

  uint256 public openingTime;
  uint256 public closingTime;

  /**
   * @dev Reverts if not in crowdsale time range.
   */
  modifier onlyWhileOpen {
    // solium-disable-next-line security/no-block-members
    require(block.timestamp >= openingTime && block.timestamp <= closingTime);
    _;
  }

  /**
   * @dev Constructor, takes crowdsale opening and closing times.
   * @param _openingTime Crowdsale opening time
   * @param _closingTime Crowdsale closing time
   */
  /*@CTK TimedCrowdsale
    @tag assume_completion
    @post _openingTime >= block.timestamp
    @post _closingTime >= _openingTime
    @post __post.openingTime == _openingTime
    @post __post.closingTime == _closingTime
   */
  function TimedCrowdsale(uint256 _openingTime, uint256 _closingTime) public {
    // solium-disable-next-line security/no-block-members
    require(_openingTime >= block.timestamp);
    require(_closingTime >= _openingTime);

    openingTime = _openingTime;
    closingTime = _closingTime;
  }

  /**
   * @dev Checks whether the period in which the crowdsale is open has already elapsed
       .
   * @return Whether crowdsale period has elapsed
   */
  /*@CTK hasClosed
    @post __return == block.timestamp > closingTime
   */
  function hasClosed() public view returns (bool) {
    // solium-disable-next-line security/no-block-members
    return block.timestamp > closingTime;
  }

  /**
   * @dev Extend parent behavior requiring to be within contributing period
   * @param _beneficiary Token purchaser
   * @param _weiAmount Amount of wei contributed
   */
  /*@CTK _preValidatePurchase
    @tag assume_completion
    @post block.timestamp >= openingTime && block.timestamp <= closingTime
   */
  function _preValidatePurchase(address _beneficiary, uint256 _weiAmount) internal
      onlyWhileOpen {
    super._preValidatePurchase(_beneficiary, _weiAmount);
  }

}
```

File zeppelin-solidity/contracts/math/SafeMath.sol

```
pragma solidity ^0.4.21;


/**
 * @title SafeMath
 * @dev Math operations with safety checks that throw on error
 */
library SafeMath {

  /**
  * @dev Multiplies two numbers, throws on overflow.
  */
  /*@CTK SafeMath_mul
    @post __reverted == __has_assertion_failure
    @post __has_assertion_failure == __has_overflow
    @post __reverted == false -> c == a * b
    @post msg == msg__post
    @post (a > 0 && (a * b / a != b)) == __has_assertion_failure
    @post __addr_map == __addr_map__post
  */
  function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
    if (a == 0) {
      return 0;
    }
    c = a * b;
    assert(c / a == b);
    return c;
  }

  /**
  * @dev Integer division of two numbers, truncating the quotient.
  */
  /*@CTK SafeMath_div
    @post __reverted == __has_assertion_failure
    @post __has_overflow == true -> __has_assertion_failure == true
    @post __reverted == false -> __return == a / b
    @post msg == msg__post
    @post (b == 0) == __has_assertion_failure
    @post __addr_map == __addr_map__post
  */
  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 a / b;
  }

  /**
  * @dev Subtracts two numbers, throws on overflow (i.e. if subtrahend is greater than
      minuend).
  */
  /*@CTK SafeMath_sub
    @tag spec
    @post __reverted == __has_assertion_failure
    @post __has_overflow == true -> __has_assertion_failure == true
    @post __reverted == false -> __return == a - b
    @post msg == msg__post
    @post (a < b) == __has_assertion_failure
    @post __addr_map == __addr_map__post
  */
  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    assert(b <= a);
    return a - b;
  }

  /**
  * @dev Adds two numbers, throws on overflow.
  */
  /*@CTK SafeMath_add
    @tag spec
    @post __reverted == __has_assertion_failure
    @post __has_assertion_failure == __has_overflow
    @post __reverted == false -> c == a + b
    @post msg == msg__post
    @post ((a + b < a) || (a + b < b)) == __has_assertion_failure
    @post __addr_map == __addr_map__post
  */
  function add(uint256 a, uint256 b) internal pure returns (uint256 c) {
    c = a + b;
    assert(c >= a);
    return c;
  }
}
```

File zeppelin-solidity/contracts/ownership/Ownable.sol

```
pragma solidity ^0.4.21;


/**
 * @title Ownable
 * @dev The Ownable contract has an owner address, and provides basic authorization
     control
 * functions, this simplifies the implementation of "user permissions".
 */
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
   * account.
   */
  /*@CTK OwnableConstructor
    @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 msg.sender == owner
    @post __post.owner == newOwner
  */
  function transferOwnership(address newOwner) public onlyOwner {
    require(newOwner != address(0));
    emit OwnershipTransferred(owner, newOwner);
    owner = newOwner;
  }

}
```

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 tc
        ) {
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 ICOToken.sol

```
1  pragma solidity ^0.4.23;
```

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

INSECURE_COMPILER_VERSION

Line 1 in File BasicCrowdsale.sol

```
1  pragma solidity ^0.4.24;
```

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

TIMESTAMP_DEPENDENCY

Line 53 in File BasicCrowdsale.sol

```
53          if(now <= privateSaleEndDate) {
```

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 160 in File BasicCrowdsale.sol

```
160         require(now <= privateSaleEndDate);
```

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 181 in File BasicCrowdsale.sol

```
181         require(now > privateSaleEndDate);
```

"now" can be influenced by minors to some degree

INSECURE_COMPILER_VERSION

Line 1 in File WhitelistedBasicCrowdsale.sol

```
1  pragma solidity ^0.4.24;
```

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

INSECURE_COMPILER_VERSION

Line 1 in File MultipleWhitelistedCrowdsale.sol

```
1  pragma solidity ^0.4.24;
```

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

INSECURE_COMPILER_VERSION

Line 1 in File StandardToken.sol

```
pragma solidity ^0.4.21;
```

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

INSECURE_COMPILER_VERSION

Line 1 in File MintableToken.sol

```
pragma solidity ^0.4.21;
```

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

INSECURE_COMPILER_VERSION

Line 1 in File BasicToken.sol

```
pragma solidity ^0.4.21;
```

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

INSECURE_COMPILER_VERSION

Line 1 in File BurnableToken.sol

```
pragma solidity ^0.4.21;
```

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

INSECURE_COMPILER_VERSION

Line 1 in File Pausable.sol

```
pragma solidity ^0.4.21;
```

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

INSECURE_COMPILER_VERSION

Line 1 in File Crowdsale.sol

```
pragma solidity ^0.4.21;
```

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

INSECURE_COMPILER_VERSION

Line 1 in File FinalizableCrowdsale.sol

```
pragma solidity ^0.4.21;
```

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

INSECURE_COMPILER_VERSION

Line 1 in File TimedCrowdsale.sol

```
pragma solidity ^0.4.21;
```

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

TIMESTAMP_DEPENDENCY

Line 22 in File TimedCrowdsale.sol

```
    require(block.timestamp >= openingTime && block.timestamp <= closingTime);
```

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

TIMESTAMP_DEPENDENCY

Line 22 in File TimedCrowdsale.sol

```
    require(block.timestamp >= openingTime && block.timestamp <= closingTime);
```

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

TIMESTAMP_DEPENDENCY

Line 40 in File TimedCrowdsale.sol

```
    require(_openingTime >= block.timestamp);
```

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

TIMESTAMP_DEPENDENCY

Line 56 in File TimedCrowdsale.sol

```
    return block.timestamp > closingTime;
```

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

INSECURE_COMPILER_VERSION

Line 1 in File SafeMath.sol

```
pragma solidity ^0.4.21;
```

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

INSECURE_COMPILER_VERSION

Line 1 in File Ownable.sol

```
pragma solidity ^0.4.21;
```

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

Formal Verification Request 1

ICOToken

📅 18, Apr 2019
⏱ 3.42 ms

Line 23-25 in File ICOToken.sol

```
/*@CTK ICOToken
  @post !__reverted
 */
```

Line 26-27 in File ICOToken.sol

```
constructor() public {
}
```

✓ The code meets the specification

Formal Verification Request 2

addMinter

📅 18, Apr 2019
⏱ 31.48 ms

Line 61-66 in File BasicCrowdsale.sol

```
/*@CTK addMinter
  @tag assume_completion
  @post owner == msg.sender
  @post _minter != address(0)
  @post __post.minters[_minter]
 */
```

Line 67-71 in File BasicCrowdsale.sol

```
function addMinter(address _minter) public onlyOwner {
    require(_minter != address(0));
    minters[_minter] = true;
    emit LogMinterAdded(_minter);
}
```

✓ The code meets the specification

Formal Verification Request 3

removeMinter

📅 18, Apr 2019
⏱ 22.6 ms

Line 73-77 in File BasicCrowdsale.sol

```
    /*@CTK removeMinter
      @tag assume_completion
      @post owner == msg.sender
      @post !__post.minters[_minter]
     */
```

Line 78-81 in File BasicCrowdsale.sol

```
    function removeMinter(address _minter) public onlyOwner {
        minters[_minter] = false;
        emit LogMinterRemoved(_minter);
    }
```

✓ The code meets the specification

Formal Verification Request 4

createFiatToken

18, Apr 2019
285.9 ms

Line 83-86 in File BasicCrowdsale.sol

```
    /*@CTK createFiatToken
      @tag assume_completion
      @post block.timestamp <= closingTime
     */
```

Line 87-92 in File BasicCrowdsale.sol

```
    function createFiatToken(address beneficiary, uint256 amount) public onlyMinter()
        returns(bool){
        require(!hasClosed());
        mintFiatToken(beneficiary, amount);
        emit LogFiatTokenMinted(msg.sender, beneficiary, amount);
        return true;
    }
```

✓ The code meets the specification

Formal Verification Request 5

createBountyToken

18, Apr 2019
283.67 ms

Line 108-111 in File BasicCrowdsale.sol

```
    /*@CTK createBountyToken
      @tag assume_completion
      @post block.timestamp <= closingTime
     */
```

Line 112-117 in File BasicCrowdsale.sol

```
function createBountyToken(address beneficiary, uint256 amount) public onlyMinter
    () returns (bool) {
    require(!hasClosed());
    mintBountyToken(beneficiary, amount);
    emit LogBountyTokenMinted(msg.sender, beneficiary, amount);
    return true;
}
```

✓ The code meets the specification

Formal Verification Request 6

multiBeneficiariesValidation

18, Apr 2019
49.0 ms

Line 136-142 in File BasicCrowdsale.sol

```
/*@CTK multiBeneficiariesValidation
  @tag assume_completion
  @post block.timestamp <= closingTime
  @post beneficiaries.length > 0
  @post amount.length > 0
  @post beneficiaries.length == amount.length
 */
```

Line 143-148 in File BasicCrowdsale.sol

```
function multiBeneficiariesValidation(address[] beneficiaries, uint256[] amount)
    internal view {
    require(!hasClosed());
    require(beneficiaries.length > 0);
    require(amount.length > 0);
    require(beneficiaries.length == amount.length);
}
```

✓ The code meets the specification

Formal Verification Request 7

extendPrivateSaleDuration

18, Apr 2019
289.16 ms

Line 153-158 in File BasicCrowdsale.sol

```
/*@CTK extendPrivateSaleDuration
  @tag assume_completion
  @post owner == msg.sender
  @post __post.privateSaleEndDate == privateSaleEndDate + extentionInDays * 86400
  @post __post.closingTime == closingTime + extentionInDays * 86400
 */
```

Line 159-166 in File BasicCrowdsale.sol

```
function extendPrivateSaleDuration(uint256 extentionInDays) public onlyOwner
    returns (bool) {
    require(now <= privateSaleEndDate);
    extentionInDays = extentionInDays.mul(1 days); // convert the days in
        milliseconds
    privateSaleEndDate = privateSaleEndDate.add(extentionInDays);
    closingTime = closingTime.add(extentionInDays);
    emit LogPrivateSaleExtended(extentionInDays);
    return true;
}
```

✓ The code meets the specification

Formal Verification Request 8

extendMainSailDuration

18, Apr 2019
326.6 ms

Line 171-179 in File BasicCrowdsale.sol

```
/*@CTK extendMainSailDuration
  @tag assume_completion
  @post now > privateSaleEndDate
  @post block.timestamp <= closingTime
  @post mainSaleDurantionExtentionLimit - extentionInDays >= 0
  @post owner == msg.sender
  @post __post.mainSaleDurantionExtentionLimit == mainSaleDurantionExtentionLimit
      - extentionInDays
  @post __post.closingTime == closingTime + extentionInDays * 86400
 */
```

Line 180-191 in File BasicCrowdsale.sol

```
function extendMainSailDuration(uint256 extentionInDays) public onlyOwner returns
    (bool) {
    require(now > privateSaleEndDate);
    require(!hasClosed());
    require(mainSaleDurantionExtentionLimit.sub(extentionInDays) >= 0);

    uint256 extention = extentionInDays.mul(1 days); // convert the days in
        milliseconds
    mainSaleDurantionExtentionLimit = mainSaleDurantionExtentionLimit.sub(
        extentionInDays); // substract days from the limit
    closingTime = closingTime.add(extention);

    emit LogMainSaleExtended(extentionInDays);
    return true;
}
```

✓ The code meets the specification

Formal Verification Request 9

changeRate

📅 18, Apr 2019
⏱ 54.16 ms

Line 193-199 in File BasicCrowdsale.sol

```
    /*@CTK changeRate
      @tag assume_completion
      @post owner == msg.sender
      @post block.timestamp <= closingTime
      @post _newRate != 0
      @post __post.rate == _newRate
     */
```

Line 200-206 in File BasicCrowdsale.sol

```
    function changeRate(uint _newRate) public onlyOwner returns (bool) {
        require(!hasClosed());
        require(_newRate != 0);
        rate = _newRate;
        emit LogRateChanged(_newRate);
        return true;
    }
```

✓ The code meets the specification

Formal Verification Request 10

WhitelistedBasicCrowdsale

📅 18, Apr 2019
⏱ 4.67 ms

Line 10-13 in File WhitelistedBasicCrowdsale.sol

```
  /*@CTK WhitelistedBasicCrowdsale
    @tag assume_completion
    @post !__reverted
   */
```

Line 14-18 in File WhitelistedBasicCrowdsale.sol

```
    constructor(uint256 _rate, address _wallet, address _token, uint256 _openingTime,
        uint256 _closingTime)
    // BasicCrowdsale(_rate, _wallet, ERC20(_token), _openingTime, _closingTime)
    // MultipleWhitelistedCrowdsale()
    public {
    }
```

✓ The code meets the specification

Formal Verification Request 11

addWhitelistManager

18, Apr 2019
32.94 ms

Line 36-41 in File MultipleWhitelistedCrowdsale.sol

```
/*@CTK addWhitelistManager
  @tag assume_completion
  @post owner == msg.sender
  @post _manager != address(0)
  @post __post.whitelistManagers[_manager]
 */
```

Line 42-45 in File MultipleWhitelistedCrowdsale.sol

```
function addWhitelistManager(address _manager) public onlyOwner {
    require(_manager != address(0));
    whitelistManagers[_manager] = true;
}
```

✓ The code meets the specification

Formal Verification Request 12

removeWhitelistManager

18, Apr 2019
22.43 ms

Line 50-54 in File MultipleWhitelistedCrowdsale.sol

```
/*@CTK removeWhitelistManager
  @tag assume_completion
  @post owner == msg.sender
  @post !__post.whitelistManagers[_manager]
 */
```

Line 55-57 in File MultipleWhitelistedCrowdsale.sol

```
function removeWhitelistManager(address _manager) public onlyOwner {
    whitelistManagers[_manager] = false;
}
```

✓ The code meets the specification

Formal Verification Request 13

addToWhitelist

18, Apr 2019
23.48 ms

Line 63-67 in File MultipleWhitelistedCrowdsale.sol

```
/*@CTK addToWhitelist
  @tag assume_completion
  @post whitelistManagers[msg.sender]
  @post __post.whitelist[_beneficiary]
 */
```

Line 68-70 in File MultipleWhitelistedCrowdsale.sol

```
function addToWhitelist(address _beneficiary) external onlyWhitelistManager() {
  whitelist[_beneficiary] = true;
}
```

✓ The code meets the specification

Formal Verification Request 14

removeFromWhitelist

📅 18, Apr 2019
⏱ 23.56 ms

Line 86-90 in File MultipleWhitelistedCrowdsale.sol

```
/*@CTK removeFromWhitelist
  @tag assume_completion
  @post whitelistManagers[msg.sender]
  @post !__post.whitelist[_beneficiary]
 */
```

Line 91-93 in File MultipleWhitelistedCrowdsale.sol

```
function removeFromWhitelist(address _beneficiary) external onlyWhitelistManager() {
  whitelist[_beneficiary] = false;
}
```

✓ The code meets the specification

Formal Verification Request 15

transferFrom

📅 18, Apr 2019
⏱ 156.65 ms

Line 25-32 in File StandardToken.sol

```
/*@CTK transferFrom
  @tag assume_completion
  @pre _from != _to
  @post __return == true
  @post __post.balances[_to] == balances[_to] + _value
  @post __post.balances[_from] == balances[_from] - _value
  @post __has_overflow == false
*/
```

Line 33-43 in File StandardToken.sol

```
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;
}
```

✓ The code meets the specification

Formal Verification Request 16

approve_success

18, Apr 2019
9.8 ms

Line 55-58 in File StandardToken.sol

```
/*@CTK approve_success
  @post _value == 0 -> __reverted == false
  @post allowed[msg.sender][_spender] == 0 -> __reverted == false
*/
```

Line 63-67 in File StandardToken.sol

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

✓ The code meets the specification

Formal Verification Request 17

approve

18, Apr 2019
1.35 ms

Line 59-62 in File StandardToken.sol

```
/*@CTK approve
  @tag assume_completion
  @post __post.allowed[msg.sender][_spender] == _value
*/
```

Line 63-67 in File StandardToken.sol

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

✓ The code meets the specification

Formal Verification Request 18

get_allowance

18, Apr 2019
6.51 ms

Line 75-79 in File StandardToken.sol

```
/*@CTK get_allowance
  @post __reverted == false
  @post __return == allowed[_owner][_spender]
  @post this == __post
*/
```

Line 80-82 in File StandardToken.sol

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

✓ The code meets the specification

Formal Verification Request 19

increaseApproval

18, Apr 2019
20.06 ms

Line 94-97 in File StandardToken.sol

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

Line 98-102 in File StandardToken.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 20

decreaseApproval correctness case 1

📅 18, Apr 2019
⏱ 31.41 ms

Line 114-119 in File StandardToken.sol

```
/*@CTK "decreaseApproval correctness case 1"
  @pre allowed[msg.sender][_spender] < _subtractedValue
  @post __post.allowed[msg.sender][_spender] == 0
  @post __return == true
  @post (!__has_assertion_failure)
*/
```

Line 126-135 in File StandardToken.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 21

decreaseApproval correctness case 2

📅 18, Apr 2019
⏱ 2.24 ms

Line 120-125 in File StandardToken.sol

```
/*@CTK "decreaseApproval correctness case 2"
  @pre allowed[msg.sender][_spender] >= _subtractedValue
  @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] -
      _subtractedValue
  @post __return == true
  @post (!__has_assertion_failure)
*/
```

Line 126-135 in File StandardToken.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 22

mint

18, Apr 2019
166.11 ms

Line 31-35 in File MintableToken.sol

```
  /*@CTK mint
    @tag assume_completion
    @post __post.totalSupply_ == totalSupply_ + _amount
    @post __post.balances[_to] == balances[_to] + _amount
   */
```

Line 36-42 in File MintableToken.sol

```
  function mint(address _to, uint256 _amount) onlyOwner canMint public returns (bool)
      {
    totalSupply_ = totalSupply_.add(_amount);
    balances[_to] = balances[_to].add(_amount);
    emit Mint(_to, _amount);
    emit Transfer(address(0), _to, _amount);
    return true;
  }
```

✓ The code meets the specification

Formal Verification Request 23

finishMinting

18, Apr 2019
28.44 ms

Line 48-53 in File MintableToken.sol

```
  /*@CTK finishMinting
    @tag assume_completion
    @post owner == msg.sender
    @post !mintingFinished
    @post __post.mintingFinished
   */
```

Line 54-58 in File MintableToken.sol

```
function finishMinting() onlyOwner canMint public returns (bool) {
  mintingFinished = true;
  emit MintFinished();
  return true;
}
```

✓ The code meets the specification

Formal Verification Request 24

totalSupply

18, Apr 2019
6.6 ms

Line 22-24 in File BasicToken.sol

```
/*@CTK totalSupply
  @post __return == totalSupply_
 */
```

Line 25-27 in File BasicToken.sol

```
function totalSupply() public view returns (uint256) {
  return totalSupply_;
}
```

✓ The code meets the specification

Formal Verification Request 25

transfer_success

18, Apr 2019
55.06 ms

Line 34-40 in File BasicToken.sol

```
/*@CTK transfer_success
  @pre _to != address(0)
  @pre balances[msg.sender] >= _value
  @pre __reverted == false
  @post __reverted == false
  @post __return == true
*/
```

Line 53-61 in File BasicToken.sol

```
function transfer(address _to, uint256 _value) public returns (bool) {
  require(_to != address(0));
  require(_value <= balances[msg.sender]);

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

✓ The code meets the specification

Formal Verification Request 26

transfer_conditions

18, Apr 2019
118.28 ms

Line 41-46 in File BasicToken.sol

```
/*@CTK transfer_conditions
  @tag assume_completion
  @pre _to != msg.sender
  @post __post.balances[_to] == balances[_to] + _value
  @post __post.balances[msg.sender] == balances[msg.sender] - _value
*/
```

Line 53-61 in File BasicToken.sol

```
function transfer(address _to, uint256 _value) public returns (bool) {
  require(_to != address(0));
  require(_value <= balances[msg.sender]);

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

✓ The code meets the specification

Formal Verification Request 27

transfer_same_address

18, Apr 2019
7.91 ms

Line 47-52 in File BasicToken.sol

```
/*@CTK transfer_same_address
  @tag assume_completion
  @tag no_overflow
  @pre _to == msg.sender
  @post this == __post
*/
```

Line 53-61 in File BasicToken.sol

```
function transfer(address _to, uint256 _value) public returns (bool) {
  require(_to != address(0));
  require(_value <= balances[msg.sender]);

  balances[msg.sender] = balances[msg.sender].sub(_value);
```

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

✅ The code meets the specification

Formal Verification Request 28

balanceOf

📅 18, Apr 2019
⏱ 5.9 ms

Line 68-71 in File BasicToken.sol

```
  /*@CTK balanceOf
    @post __reverted == false
    @post __return == balances[_owner]
  */
```

Line 72-74 in File BasicToken.sol

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

✅ The code meets the specification

Formal Verification Request 29

_burn

📅 18, Apr 2019
⏱ 105.52 ms

Line 18-23 in File BurnableToken.sol

```
  /*@CTK _burn
    @tag assume_completion
    @post _value <= balances[msg.sender]
    @post __post.balances[msg.sender] == balances[msg.sender] - _value
    @post __post.totalSupply_ == totalSupply_ - _value
   */
```

Line 24-26 in File BurnableToken.sol

```
  function burn(uint256 _value) public {
    _burn(msg.sender, _value);
  }
```

✅ The code meets the specification

Formal Verification Request 30

_burn

📅 18, Apr 2019
⏱ 40.07 ms

Line 28-33 in File BurnableToken.sol

```
/*@CTK _burn
  @tag assume_completion
  @post _value <= balances[_who]
  @post __post.balances[_who] == balances[_who] - _value
  @post __post.totalSupply_ == totalSupply_ - _value
 */
```

Line 34-43 in File BurnableToken.sol

```
function _burn(address _who, uint256 _value) internal {
  require(_value <= balances[_who]);
  // no need to require value <= totalSupply, since that would imply the
  // sender's balance is greater than the totalSupply, which *should* be an
      assertion failure

  balances[_who] = balances[_who].sub(_value);
  totalSupply_ = totalSupply_.sub(_value);
  emit Burn(_who, _value);
  emit Transfer(_who, address(0), _value);
}
```

✓ The code meets the specification

Formal Verification Request 31

pause

📅 18, Apr 2019
⏱ 29.81 ms

Line 37-41 in File Pausable.sol

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

Line 42-45 in File Pausable.sol

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

✓ The code meets the specification

Formal Verification Request 32

unpause

📅 18, Apr 2019
⏱ 26.07 ms

Line 50-54 in File Pausable.sol

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

Line 55-58 in File Pausable.sol

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

✓ The code meets the specification

Formal Verification Request 33

Crowdsale

📅 18, Apr 2019
⏱ 38.96 ms

Line 48-56 in File Crowdsale.sol

```
/*@CTK Crowdsale
  @tag assume_completion
  @post _rate > 0
  @post _wallet != address(0)
  @post _token != address(0)
  @post __post.rate == _rate
  @post __post.wallet == _wallet
  @post __post.token == _token
 */
```

Line 57-65 in File Crowdsale.sol

```
function Crowdsale(uint256 _rate, address _wallet, ERC20 _token) public {
  require(_rate > 0);
  require(_wallet != address(0));
  require(_token != address(0));

  rate = _rate;
  wallet = _wallet;
  token = _token;
}
```

✓ The code meets the specification

Formal Verification Request 34

_preValidatePurchase

📅 18, Apr 2019
⏱ 19.37 ms

Line 116-120 in File Crowdsale.sol

```
/*@CTK _preValidatePurchase
  @tag assume_completion
  @post _beneficiary != address(0)
  @post _weiAmount != 0
 */
```

Line 121-124 in File Crowdsale.sol

```
function _preValidatePurchase(address _beneficiary, uint256 _weiAmount) internal {
  require(_beneficiary != address(0));
  require(_weiAmount != 0);
}
```

✓ The code meets the specification

Formal Verification Request 35

_getTokenAmount

📅 18, Apr 2019
⏱ 65.92 ms

Line 167-169 in File Crowdsale.sol

```
/*@CTK _getTokenAmount
  @post __return == _weiAmount * rate
 */
```

Line 170-172 in File Crowdsale.sol

```
function _getTokenAmount(uint256 _weiAmount) internal view returns (uint256) {
  return _weiAmount.mul(rate);
}
```

✓ The code meets the specification

Formal Verification Request 36

finalize

📅 18, Apr 2019
⏱ 78.23 ms

Line 24-29 in File FinalizableCrowdsale.sol

```
/*@CTK finalize
  @tag assume_completion
  @post !isFinalized
  @post block.timestamp > closingTime
  @post __post.isFinalized
 */
```

Line 30-38 in File FinalizableCrowdsale.sol

```
function finalize() onlyOwner public {
  require(!isFinalized);
  require(hasClosed());

  finalization();
  emit Finalized();

  isFinalized = true;
}
```

✓ The code meets the specification

Formal Verification Request 37

TimedCrowdsale

18, Apr 2019
33.73 ms

Line 31-37 in File TimedCrowdsale.sol

```
/*@CTK TimedCrowdsale
  @tag assume_completion
  @post _openingTime >= block.timestamp
  @post _closingTime >= _openingTime
  @post __post.openingTime == _openingTime
  @post __post.closingTime == _closingTime
 */
```

Line 38-45 in File TimedCrowdsale.sol

```
function TimedCrowdsale(uint256 _openingTime, uint256 _closingTime) public {
  // solium-disable-next-line security/no-block-members
  require(_openingTime >= block.timestamp);
  require(_closingTime >= _openingTime);

  openingTime = _openingTime;
  closingTime = _closingTime;
}
```

✓ The code meets the specification

Formal Verification Request 38

hasClosed

18, Apr 2019
7.07 ms

Line 51-53 in File TimedCrowdsale.sol

```
/*@CTK hasClosed
  @post __return == block.timestamp > closingTime
 */
```

Line 54-57 in File TimedCrowdsale.sol

```
function hasClosed() public view returns (bool) {
  // solium-disable-next-line security/no-block-members
  return block.timestamp > closingTime;
}
```

✓ The code meets the specification

Formal Verification Request 39

_preValidatePurchase

18, Apr 2019
60.02 ms

Line 64-67 in File TimedCrowdsale.sol

```
/*@CTK _preValidatePurchase
  @tag assume_completion
  @post block.timestamp >= openingTime && block.timestamp <= closingTime
 */
```

Line 68-70 in File TimedCrowdsale.sol

```
function _preValidatePurchase(address _beneficiary, uint256 _weiAmount) internal
    onlyWhileOpen {
  super._preValidatePurchase(_beneficiary, _weiAmount);
}
```

✓ The code meets the specification

Formal Verification Request 40

SafeMath_mul

18, Apr 2019
364.95 ms

Line 13-20 in File SafeMath.sol

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

Line 21-28 in File SafeMath.sol

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

✓ The code meets the specification

Formal Verification Request 41

SafeMath_div

18, Apr 2019
7.44 ms

Line 33-40 in File SafeMath.sol

```
/*@CTK SafeMath_div
  @post __reverted == __has_assertion_failure
  @post __has_overflow == true -> __has_assertion_failure == true
  @post __reverted == false -> __return == a / b
  @post msg == msg__post
  @post (b == 0) == __has_assertion_failure
  @post __addr_map == __addr_map__post
*/
```

Line 41-46 in File SafeMath.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 a / b;
}
```

✓ The code meets the specification

Formal Verification Request 42

SafeMath_sub

18, Apr 2019
14.27 ms

Line 51-59 in File SafeMath.sol

```
/*@CTK SafeMath_sub
  @tag spec
  @post __reverted == __has_assertion_failure
  @post __has_overflow == true -> __has_assertion_failure == true
  @post __reverted == false -> __return == a - b
  @post msg == msg__post
  @post (a < b) == __has_assertion_failure
  @post __addr_map == __addr_map__post
*/
```

Line 60-63 in File SafeMath.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 43

SafeMath_add

📅 18, Apr 2019
⏱ 19.29 ms

Line 68-76 in File SafeMath.sol

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

Line 77-81 in File SafeMath.sol

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

✓ The code meets the specification

Formal Verification Request 44

OwnableConstructor

📅 18, Apr 2019
⏱ 5.87 ms

Line 20-22 in File Ownable.sol

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

Line 23-25 in File Ownable.sol

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

✓ The code meets the specification

Formal Verification Request 45

transferOwnership

18, Apr 2019

26.52 ms

Line 39-43 in File Ownable.sol

```
/*@CTK transferOwnership
  @tag assume_completion
  @post msg.sender == owner
  @post __post.owner == newOwner
*/
```

Line 44-48 in File Ownable.sol

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

✓ The code meets the specification