

# CERTiK VERIFICATION REPORT FOR MYCRO



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

## 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 Mycro(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.

# 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% covered 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	Com-	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	Ran-	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	for	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115
Delegatecall to Untrusted Callee	to	Calling into untrusted contracts is very dangerous, the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Variable	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	Default	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

```

1  pragma solidity ^0.4.23;
2
3  import "zeppelin-solidity/contracts/token/ERC20/MintableToken.sol";
4  import "zeppelin-solidity/contracts/token/ERC20/PausableToken.sol";
5  import "zeppelin-solidity/contracts/token/ERC20/BurnableToken.sol";
6
7
8  /**
9   * @title ICOToken
10  * @dev Very simple ERC20 Token example.
11  * 'StandardToken' functions.
12  */
13  contract ICOToken is MintableToken, PausableToken, BurnableToken {
14
15      string public constant name = "Mycro Token";
16      string public constant symbol = "MYO";
17      uint8 public constant decimals = 18;
18
19
20      /**
21       * @dev Constructor that gives msg.sender all of existing tokens.
22       */
23      /*@CTK ICOToken
24       @post !__reverted
25       */
26      constructor() public {
27      }
28  }
```

File Crowdsale/BasicCrowdsale.sol

```

1  pragma solidity ^0.4.24;
2
3  import "zeppelin-solidity/contracts/crowdsale/emission/MintedCrowdsale.sol";
4  import "zeppelin-solidity/contracts/crowdsale/distribution/FinalizableCrowdsale.sol";
5
6
7  contract BasicCrowdsale is MintedCrowdsale, FinalizableCrowdsale {
8
9      uint256 public cap = 100000000 * (10 ** 18); // Total number of MYO tokens that
10         would be created
11      uint256 public capForSale = 71000000 * (10 ** 18); // Total MYO tokens that could
12         be sold during the ICO
13      uint256 public bountyTokensCap = 5000000 * (10 ** 18); // Total number of MYO
14         tokens that would be given as a reward
15      uint256 public reservedForTeamTokens = 29000000 * (10 ** 18); // Tokens reserved
16         for rewardpool, advisors and team that will be minted after Crowdsale
17      uint256 public totalMintedBountyTokens; // Total number of MYO tokens given as a
18         reward
19
20      uint256 public privateSaleEndDate;
21      mapping (address => bool) public minters;
22
23      uint256 constant MIN_CONTRIBUTION_AMOUNT = 10 finney;
24      uint256 constant MAX_CONTRIBUTION_AMOUNT = 250 ether;
25  }
```

```

20
21 uint256 public constant PRIVATE_SALE_CAP = 26000000 * (10 ** 18);
22 uint256 public constant PRIVATE_SALE_DURATION = 30 days; // to be calculated
    according to deployment day; the end date should be 30 May
23
24 uint256 public constant MAIN_SALE_DURATION = 60 days;
25 uint256 public mainSaleDurationExtensionLimit = 60; //max days the duration of
    the ICO can be extended
26
27 event LogFiatTokenMinted(address sender, address beneficiary, uint256 amount);
28 event LogFiatTokenMintedToMany(address sender, address[] beneficiaries, uint256[]
    amount);
29 event LogBountyTokenMinted(address minter, address beneficiary, uint256 amount);
30 event LogBountyTokenMintedToMany(address sender, address[] beneficiaries, uint256
    [] amount);
31 event LogPrivateSaleExtended(uint256 extensionInDays);
32 event LogMainSaleExtended(uint256 extensionInDays);
33 event LogRateChanged(uint256 rate);
34 event LogMinterAdded(address minterAdded);
35 event LogMinterRemoved(address minterRemoved);
36
37 constructor(uint256 _rate, address _wallet, address _token, uint256 _openingTime,
    uint256 _closingTime)
38 Crowdsale(_rate, _wallet, ERC20(_token))
39 TimedCrowdsale(_openingTime, _closingTime) public {
40     minters[owner] = true;
41     privateSaleEndDate = _openingTime.add(PRIVATE_SALE_DURATION);
42 }
43
44 // only addresses who are allowed to mint
45 modifier onlyMinter (){
46     require(minters[msg.sender]);
47     _;
48 }
49
50 function buyTokens(address beneficiary) public payable {
51     require(msg.value >= MIN_CONTRIBUTION_AMOUNT);
52     require(msg.value <= MAX_CONTRIBUTION_AMOUNT);
53     if(now <= privateSaleEndDate) {
54         require(MintableToken(token).totalSupply() < PRIVATE_SALE_CAP);
55     }
56     uint amount = _getTokenAmount(msg.value);
57     require(MintableToken(token).totalSupply().add(amount) <= capForSale);
58     super.buyTokens(beneficiary);
59 }
60
61 /*@CTK addMinter
62   @tag assume_completion
63   @post owner == msg.sender
64   @post _minter != address(0)
65   @post __post.minters[_minter]
66 */
67 function addMinter(address _minter) public onlyOwner {
68     require(_minter != address(0));
69     minters[_minter] = true;
70     emit LogMinterAdded(_minter);
71 }
72

```

```

73  /*@CTK removeMinter
74      @tag assume_completion
75      @post owner == msg.sender
76      @post !__post.minters[_minter]
77  */
78  function removeMinter(address _minter) public onlyOwner {
79      minters[_minter] = false;
80      emit LogMinterRemoved(_minter);
81  }
82
83  /*@CTK createFiatToken
84      @tag assume_completion
85      @post block.timestamp <= closingTime
86  */
87  function createFiatToken(address beneficiary, uint256 amount) public onlyMinter()
88      returns(bool){
89      require(!hasClosed());
90      mintFiatToken(beneficiary, amount);
91      emit LogFiatTokenMinted(msg.sender, beneficiary, amount);
92      return true;
93  }
94
95  function createFiatTokenToMany(address[] beneficiaries, uint256[] amount) public
96      onlyMinter() returns(bool){
97      multiBeneficiariesValidation(beneficiaries, amount);
98      for(uint i = 0; i < beneficiaries.length; i++){
99          mintFiatToken(beneficiaries[i], amount[i]);
100      }
101      emit LogFiatTokenMintedToMany(msg.sender, beneficiaries, amount);
102      return true;
103  }
104
105  function mintFiatToken(address beneficiary, uint256 amount) internal {
106      require(MintableToken(token).totalSupply().add(amount) <= capForSale);
107      MintableToken(token).mint(beneficiary, amount);
108  }
109
110  /*@CTK createBountyToken
111      @tag assume_completion
112      @post block.timestamp <= closingTime
113  */
114  function createBountyToken(address beneficiary, uint256 amount) public onlyMinter
115      () returns (bool) {
116      require(!hasClosed());
117      mintBountyToken(beneficiary, amount);
118      emit LogBountyTokenMinted(msg.sender, beneficiary, amount);
119      return true;
120  }
121
122  function createBountyTokenToMany(address[] beneficiaries, uint256[] amount) public
123      onlyMinter() returns (bool) {
124      multiBeneficiariesValidation(beneficiaries, amount);
125      for(uint i = 0; i < beneficiaries.length; i++){
126          mintBountyToken(beneficiaries[i], amount[i]);
127      }
128
129      emit LogBountyTokenMintedToMany(msg.sender, beneficiaries, amount);
130      return true;

```



```

127     }
128
129     function mintBountyToken(address beneficiary, uint256 amount) internal {
130         require(MintableToken(token).totalSupply().add(amount) <= capForSale);
131         require(totalMintedBountyTokens.add(amount) <= bountyTokensCap);
132         MintableToken(token).mint(beneficiary, amount);
133         totalMintedBountyTokens = totalMintedBountyTokens.add(amount);
134     }
135
136     /*@CTK multiBeneficiariesValidation
137     @tag assume_completion
138     @post block.timestamp <= closingTime
139     @post beneficiaries.length > 0
140     @post amount.length > 0
141     @post beneficiaries.length == amount.length
142     */
143     function multiBeneficiariesValidation(address[] beneficiaries, uint256[] amount)
144         internal view {
145         require(!hasClosed());
146         require(beneficiaries.length > 0);
147         require(amount.length > 0);
148         require(beneficiaries.length == amount.length);
149     }
150
151     /**
152     @param extentionInDays is a simple number of the days, e.c. 3 => 3 days
153     */
154     /*@CTK extendPrivateSaleDuration
155     @tag assume_completion
156     @post owner == msg.sender
157     @post __post.privateSaleEndDate == privateSaleEndDate + extentionInDays * 86400
158     @post __post.closingTime == closingTime + extentionInDays * 86400
159     */
160     function extendPrivateSaleDuration(uint256 extentionInDays) public onlyOwner
161         returns (bool) {
162         require(now <= privateSaleEndDate);
163         extentionInDays = extentionInDays.mul(1 days); // convert the days in
164             milliseconds
165         privateSaleEndDate = privateSaleEndDate.add(extentionInDays);
166         closingTime = closingTime.add(extentionInDays);
167         emit LogPrivateSaleExtended(extentionInDays);
168         return true;
169     }
170
171     /**
172     @param extentionInDays is a simple number of the days, e.c. 3 => 3 days
173     */
174     /*@CTK extendMainSailDuration
175     @tag assume_completion
176     @post now > privateSaleEndDate
177     @post block.timestamp <= closingTime
178     @post mainSaleDurantionExtentionLimit - extentionInDays >= 0
179     @post owner == msg.sender
180     @post __post.mainSaleDurantionExtentionLimit == mainSaleDurantionExtentionLimit
181         - extentionInDays
182     @post __post.closingTime == closingTime + extentionInDays * 86400
183     */
184     function extendMainSailDuration(uint256 extentionInDays) public onlyOwner returns

```

```

181     (bool) {
182         require(now > privateSaleEndDate);
183         require(!hasClosed());
184         require(mainSaleDurationExtensionLimit.sub(extensionInDays) >= 0);
185
186         uint256 extension = extensionInDays.mul(1 days); // convert the days in
187         milliseconds
188         mainSaleDurationExtensionLimit = mainSaleDurationExtensionLimit.sub(
189             extensionInDays); // subtract days from the limit
190         closingTime = closingTime.add(extension);
191
192         emit LogMainSaleExtended(extensionInDays);
193         return true;
194     }
195
196     /*@CTK changeRate
197     @tag assume_completion
198     @post owner == msg.sender
199     @post block.timestamp <= closingTime
200     @post _newRate != 0
201     @post __post.rate == _newRate
202     */
203     function changeRate(uint _newRate) public onlyOwner returns (bool) {
204         require(!hasClosed());
205         require(_newRate != 0);
206         rate = _newRate;
207         emit LogRateChanged(_newRate);
208         return true;
209     }
210
211     // after finalization will be minted manually reservedForTeamTokens amount
212     function finalization() internal {
213         MintableToken(token).transferOwnership(owner);
214         super.finalization();
215     }
216 }

```

File Crowdsale/WhitelistedBasicCrowdsale.sol

```

1  pragma solidity ^0.4.24;
2
3  import "./BasicCrowdsale.sol";
4  import "./MultipleWhitelistedCrowdsale.sol";
5
6
7  contract WhitelistedBasicCrowdsale is BasicCrowdsale, MultipleWhitelistedCrowdsale {
8
9
10     /*@CTK WhitelistedBasicCrowdsale
11     @tag assume_completion
12     @post !__reverted
13     */
14     constructor(uint256 _rate, address _wallet, address _token, uint256 _openingTime,
15         uint256 _closingTime)
16         // BasicCrowdsale(_rate, _wallet, ERC20(_token), _openingTime, _closingTime)
17         // MultipleWhitelistedCrowdsale()
18     public {
19
20     }
21 }

```

File Crowdsale/MultipleWhitelistedCrowdsale.sol

```

1 pragma solidity ^0.4.24;
2
3 import "zeppelin-solidity/contracts/crowdsale/Crowdsale.sol";
4 import "zeppelin-solidity/contracts/ownership/Ownable.sol";
5
6 /**
7  * @title MultipleWhitelistedCrowdsale
8  * @dev Crowdsale in which only whitelisted users can contribute.
9  */
10 contract MultipleWhitelistedCrowdsale is Crowdsale, Ownable {
11
12     mapping(address => bool) public whitelist;
13     // keeps all addresses who can manage the whitelist
14     mapping(address => bool) public whitelistManagers;
15
16     /**
17      * @dev Reverts if beneficiary is not whitelisted. Can be used when extending this
18      contract.
19      */
20     modifier isWhitelisted(address _beneficiary) {
21         require(whitelist[_beneficiary]);
22     }
23
24     /**
25      * @dev Reverts if msg.sender is not whitelist manager
26      */
27     modifier onlyWhitelistManager(){
28         require(whitelistManagers[msg.sender]);
29         _;
30     }
31
32     /**
33      * @dev Adds single address who can manage the whitelist.
34      * @param _manager Address to be added to the whitelistManagers
35      */
36     /*@CTK addWhitelistManager
37      @tag assume_completion
38      @post owner == msg.sender
39      @post _manager != address(0)
40      @post __post.whitelistManagers[_manager]
41      */
42     function addWhitelistManager(address _manager) public onlyOwner {
43         require(_manager != address(0));
44         whitelistManagers[_manager] = true;
45     }
46
47     /**
48      * @param _manager Address to remove from whitelistManagers
49      */
50     /*@CTK removeWhitelistManager
51      @tag assume_completion
52      @post owner == msg.sender
53      @post !__post.whitelistManagers[_manager]
54      */
55     function removeWhitelistManager(address _manager) public onlyOwner {
56         whitelistManagers[_manager] = false;

```

```

57 }
58
59 /**
60  * @dev Adds single address to whitelist.
61  * @param _beneficiary Address to be added to the whitelist
62  */
63 /*@CTK addToWhitelist
64  @tag assume_completion
65  @post whitelistManagers[msg.sender]
66  @post __post.whitelist[_beneficiary]
67  */
68 function addToWhitelist(address _beneficiary) external onlyWhitelistManager() {
69     whitelist[_beneficiary] = true;
70 }
71
72 /**
73  * @dev Adds list of addresses to whitelist. Not overloaded due to limitations with
74    truffle testing.
75  * @param _beneficiaries Addresses to be added to the whitelist
76  */
77 function addManyToWhitelist(address[] _beneficiaries) external onlyWhitelistManager
78    () {
79     for (uint256 i = 0; i < _beneficiaries.length; i++) {
80         whitelist[_beneficiaries[i]] = true;
81     }
82 }
83
84 /**
85  * @dev Removes single address from whitelist.
86  * @param _beneficiary Address to be removed to the whitelist
87  */
88 /*@CTK removeFromWhitelist
89  @tag assume_completion
90  @post whitelistManagers[msg.sender]
91  @post !__post.whitelist[_beneficiary]
92  */
93 function removeFromWhitelist(address _beneficiary) external onlyWhitelistManager() {
94     whitelist[_beneficiary] = false;
95 }
96
97 /**
98  * @dev Extend parent behavior requiring beneficiary to be in whitelist.
99  * @param _beneficiary Token beneficiary
100  * @param _weiAmount Amount of wei contributed
101  */
102 function _preValidatePurchase(address _beneficiary, uint256 _weiAmount) internal
103     isWhitelisted(_beneficiary) {
104     super._preValidatePurchase(_beneficiary, _weiAmount);
105 }

```

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

```

1 pragma solidity ^0.4.21;
2
3 import "./BasicToken.sol";
4 import "./ERC20.sol";
5

```

```

6
7 /**
8  * @title Standard ERC20 token
9  *
10 * @dev Implementation of the basic standard token.
11 * @dev https://github.com/ethereum/EIPs/issues/20
12 * @dev Based on code by FirstBlood: https://github.com/Firstbloodio/token/blob/master
    /smart_contract/FirstBloodToken.sol
13 */
14 contract StandardToken is ERC20, BasicToken {
15
16     mapping (address => mapping (address => uint256)) internal allowed;
17
18
19     /**
20     * @dev Transfer tokens from one address to another
21     * @param _from address The address which you want to send tokens from
22     * @param _to address The address which you want to transfer to
23     * @param _value uint256 the amount of tokens to be transferred
24     */
25     /*@CTK transferFrom
26     @tag assume_completion
27     @pre _from != _to
28     @post __return == true
29     @post __post.balances[_to] == balances[_to] + _value
30     @post __post.balances[_from] == balances[_from] - _value
31     @post __has_overflow == false
32     */
33     function transferFrom(address _from, address _to, uint256 _value) public returns (
        bool) {
34         require(_to != address(0));
35         require(_value <= balances[_from]);
36         require(_value <= allowed[_from][msg.sender]);
37
38         balances[_from] = balances[_from].sub(_value);
39         balances[_to] = balances[_to].add(_value);
40         allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
41         emit Transfer(_from, _to, _value);
42         return true;
43     }
44
45     /**
46     * @dev Approve the passed address to spend the specified amount of tokens on behalf
        of msg.sender.
47     *
48     * Beware that changing an allowance with this method brings the risk that someone
        may use both the old
49     * and the new allowance by unfortunate transaction ordering. One possible solution
        to mitigate this
50     * race condition is to first reduce the spender's allowance to 0 and set the
        desired value afterwards:
51     * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
52     * @param _spender The address which will spend the funds.
53     * @param _value The amount of tokens to be spent.
54     */
55     /*@CTK approve_success
56     @post _value == 0 -> __reverted == false
57     @post allowed[msg.sender][_spender] == 0 -> __reverted == false

```

```

58  */
59  /*@CTK approve
60      @tag assume_completion
61      @post __post.allowed[msg.sender][_spender] == _value
62  */
63  function approve(address _spender, uint256 _value) public returns (bool) {
64      allowed[msg.sender][_spender] = _value;
65      emit Approval(msg.sender, _spender, _value);
66      return true;
67  }
68
69  /**
70   * @dev Function to check the amount of tokens that an owner allowed to a spender.
71   * @param _owner address The address which owns the funds.
72   * @param _spender address The address which will spend the funds.
73   * @return A uint256 specifying the amount of tokens still available for the spender
74   */
75  /*@CTK get_allowance
76      @post __reverted == false
77      @post __return == allowed[_owner][_spender]
78      @post this == __post
79  */
80  function allowance(address _owner, address _spender) public view returns (uint256) {
81      return allowed[_owner][_spender];
82  }
83
84  /**
85   * @dev Increase the amount of tokens that an owner allowed to a spender.
86   *
87   * approve should be called when allowed[_spender] == 0. To increment
88   * allowed value is better to use this function to avoid 2 calls (and wait until
89   * the first transaction is mined)
90   * From MonolithDAO Token.sol
91   * @param _spender The address which will spend the funds.
92   * @param _addedValue The amount of tokens to increase the allowance by.
93   */
94  /*@CTK increaseApproval
95      @tag assume_completion
96      @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] +
          _addedValue
97  */
98  function increaseApproval(address _spender, uint _addedValue) public returns (bool)
99  {
100      allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_addedValue);
101      emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
102      return true;
103  }
104
105  /**
106   * @dev Decrease the amount of tokens that an owner allowed to a spender.
107   *
108   * approve should be called when allowed[_spender] == 0. To decrement
109   * allowed value is better to use this function to avoid 2 calls (and wait until
110   * the first transaction is mined)
111   * From MonolithDAO Token.sol
112   * @param _spender The address which will spend the funds.
113   * @param _subtractedValue The amount of tokens to decrease the allowance by.

```

```

113  */
114  /*@CTK "decreaseApproval correctness case 1"
115    @pre allowed[msg.sender][_spender] < _subtractedValue
116    @post __post.allowed[msg.sender][_spender] == 0
117    @post __return == true
118    @post (!__has_assertion_failure)
119  */
120  /*@CTK "decreaseApproval correctness case 2"
121    @pre allowed[msg.sender][_spender] >= _subtractedValue
122    @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] -
        _subtractedValue
123    @post __return == true
124    @post (!__has_assertion_failure)
125  */
126  function decreaseApproval(address _spender, uint _subtractedValue) public returns (
        bool) {
127    uint oldValue = allowed[msg.sender][_spender];
128    if (_subtractedValue > oldValue) {
129      allowed[msg.sender][_spender] = 0;
130    } else {
131      allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
132    }
133    emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
134    return true;
135  }
136
137 }

```

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

```

1  pragma solidity ^0.4.21;
2
3  import "./StandardToken.sol";
4  import "../ownership/Ownable.sol";
5
6
7  /**
8   * @title Mintable token
9   * @dev Simple ERC20 Token example, with mintable token creation
10  * @dev Issue: * https://github.com/OpenZeppelin/openzeppelin-solidity/issues/120
11  * Based on code by TokenMarketNet: https://github.com/TokenMarketNet/ico/blob/master/contracts/MintableToken.sol
12  */
13  contract MintableToken is StandardToken, Ownable {
14    event Mint(address indexed to, uint256 amount);
15    event MintFinished();
16
17    bool public mintingFinished = false;
18
19
20    modifier canMint() {
21      require(!mintingFinished);
22      _;
23    }
24
25    /**
26     * @dev Function to mint tokens
27     * @param _to The address that will receive the minted tokens.
28     * @param _amount The amount of tokens to mint.

```

```

29  * @return A boolean that indicates if the operation was successful.
30  */
31  /*@CTK mint
32  @tag assume_completion
33  @post __post.totalSupply_ == totalSupply_ + _amount
34  @post __post.balances[_to] == balances[_to] + _amount
35  */
36  function mint(address _to, uint256 _amount) onlyOwner canMint public returns (bool)
37  {
38      totalSupply_ = totalSupply_.add(_amount);
39      balances[_to] = balances[_to].add(_amount);
40      emit Mint(_to, _amount);
41      emit Transfer(address(0), _to, _amount);
42      return true;
43  }
44  /**
45  * @dev Function to stop minting new tokens.
46  * @return True if the operation was successful.
47  */
48  /*@CTK finishMinting
49  @tag assume_completion
50  @post owner == msg.sender
51  @post !mintingFinished
52  @post __post.mintingFinished
53  */
54  function finishMinting() onlyOwner canMint public returns (bool) {
55      mintingFinished = true;
56      emit MintFinished();
57      return true;
58  }
59  }

```

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

```

1  pragma solidity ^0.4.21;
2
3
4  import "./ERC20Basic.sol";
5  import "../math/SafeMath.sol";
6
7
8  /**
9  * @title Basic token
10 * @dev Basic version of StandardToken, with no allowances.
11 */
12 contract BasicToken is ERC20Basic {
13     using SafeMath for uint256;
14
15     mapping(address => uint256) balances;
16
17     uint256 totalSupply_;
18
19     /**
20     * @dev total number of tokens in existence
21     */
22     /*@CTK totalSupply
23     @post __return == totalSupply_
24     */

```



```

25 function totalSupply() public view returns (uint256) {
26     return totalSupply_;
27 }
28
29 /**
30  * @dev transfer token for a specified address
31  * @param _to The address to transfer to.
32  * @param _value The amount to be transferred.
33  */
34 /*@CTK transfer_success
35     @pre _to != address(0)
36     @pre balances[msg.sender] >= _value
37     @pre __reverted == false
38     @post __reverted == false
39     @post __return == true
40 */
41 /*@CTK transfer_conditions
42     @tag assume_completion
43     @pre _to != msg.sender
44     @post __post.balances[_to] == balances[_to] + _value
45     @post __post.balances[msg.sender] == balances[msg.sender] - _value
46 */
47 /*@CTK transfer_same_address
48     @tag assume_completion
49     @tag no_overflow
50     @pre _to == msg.sender
51     @post this == __post
52 */
53 function transfer(address _to, uint256 _value) public returns (bool) {
54     require(_to != address(0));
55     require(_value <= balances[msg.sender]);
56
57     balances[msg.sender] = balances[msg.sender].sub(_value);
58     balances[_to] = balances[_to].add(_value);
59     emit Transfer(msg.sender, _to, _value);
60     return true;
61 }
62
63 /**
64  * @dev Gets the balance of the specified address.
65  * @param _owner The address to query the the balance of.
66  * @return An uint256 representing the amount owned by the passed address.
67  */
68 /*@CTK balanceOf
69     @post __reverted == false
70     @post __return == balances[_owner]
71 */
72 function balanceOf(address _owner) public view returns (uint256) {
73     return balances[_owner];
74 }
75
76 }

```

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

```

1 pragma solidity ^0.4.21;
2
3 import "./BasicToken.sol";
4

```

```

5
6 /**
7  * @title Burnable Token
8  * @dev Token that can be irreversibly burned (destroyed).
9  */
10 contract BurnableToken is BasicToken {
11
12     event Burn(address indexed burner, uint256 value);
13
14     /**
15      * @dev Burns a specific amount of tokens.
16      * @param _value The amount of token to be burned.
17      */
18     /*@CTK _burn
19      @tag assume_completion
20      @post _value <= balances[msg.sender]
21      @post __post.balances[msg.sender] == balances[msg.sender] - _value
22      @post __post.totalSupply_ == totalSupply_ - _value
23      */
24     function burn(uint256 _value) public {
25         _burn(msg.sender, _value);
26     }
27
28     /*@CTK _burn
29      @tag assume_completion
30      @post _value <= balances[_who]
31      @post __post.balances[_who] == balances[_who] - _value
32      @post __post.totalSupply_ == totalSupply_ - _value
33      */
34     function _burn(address _who, uint256 _value) internal {
35         require(_value <= balances[_who]);
36         // no need to require value <= totalSupply, since that would imply the
37         // sender's balance is greater than the totalSupply, which *should* be an
38         // assertion failure
39
40         balances[_who] = balances[_who].sub(_value);
41         totalSupply_ = totalSupply_.sub(_value);
42         emit Burn(_who, _value);
43         emit Transfer(_who, address(0), _value);
44     }
45 }

```

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

```

1 pragma solidity ^0.4.21;
2
3
4 import "../ownership/Ownable.sol";
5
6
7 /**
8  * @title Pausable
9  * @dev Base contract which allows children to implement an emergency stop mechanism.
10  */
11 contract Pausable is Ownable {
12     event Pause();
13     event Unpause();
14
15     bool public paused = false;

```

```

16
17
18  /**
19   * @dev Modifier to make a function callable only when the contract is not paused.
20   */
21  modifier whenNotPaused() {
22      require(!paused);
23      _;
24  }
25
26  /**
27   * @dev Modifier to make a function callable only when the contract is paused.
28   */
29  modifier whenPaused() {
30      require(paused);
31      _;
32  }
33
34  /**
35   * @dev called by the owner to pause, triggers stopped state
36   */
37  /*@CTK pause
38   @tag assume_completion
39   @post owner == msg.sender
40   @post __post.paused == true
41  */
42  function pause() onlyOwner whenNotPaused public {
43      paused = true;
44      emit Pause();
45  }
46
47  /**
48   * @dev called by the owner to unpause, returns to normal state
49   */
50  /*@CTK unpause
51   @tag assume_completion
52   @post owner == msg.sender
53   @post __post.paused == false
54  */
55  function unpause() onlyOwner whenPaused public {
56      paused = false;
57      emit Unpause();
58  }
59  }

```

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

```

1  pragma solidity ^0.4.21;
2
3  import "../token/ERC20/ERC20.sol";
4  import "../math/SafeMath.sol";
5
6
7  /**
8   * @title Crowdsale
9   * @dev Crowdsale is a base contract for managing a token crowdsale,
10  * allowing investors to purchase tokens with ether. This contract implements
11  * such functionality in its most fundamental form and can be extended to provide
    additional

```

```

12 * functionality and/or custom behavior.
13 * The external interface represents the basic interface for purchasing tokens, and
    conform
14 * the base architecture for crowdsales. They are *not* intended to be modified /
    overridden.
15 * The internal interface conforms the extensible and modifiable surface of crowdsales
    . Override
16 * the methods to add functionality. Consider using 'super' where appropriate to
    concatenate
17 * behavior.
18 */
19 contract Crowdsale {
20     using SafeMath for uint256;
21
22     // The token being sold
23     ERC20 public token;
24
25     // Address where funds are collected
26     address public wallet;
27
28     // How many token units a buyer gets per wei
29     uint256 public rate;
30
31     // Amount of wei raised
32     uint256 public weiRaised;
33
34     /**
35      * Event for token purchase logging
36      * @param purchaser who paid for the tokens
37      * @param beneficiary who got the tokens
38      * @param value weis paid for purchase
39      * @param amount amount of tokens purchased
40      */
41     event TokenPurchase(address indexed purchaser, address indexed beneficiary, uint256
        value, uint256 amount);
42
43     /**
44      * @param _rate Number of token units a buyer gets per wei
45      * @param _wallet Address where collected funds will be forwarded to
46      * @param _token Address of the token being sold
47      */
48     /*@CTK Crowdsale
49      @tag assume_completion
50      @post _rate > 0
51      @post _wallet != address(0)
52      @post _token != address(0)
53      @post __post.rate == _rate
54      @post __post.wallet == _wallet
55      @post __post.token == _token
56      */
57     function Crowdsale(uint256 _rate, address _wallet, ERC20 _token) public {
58         require(_rate > 0);
59         require(_wallet != address(0));
60         require(_token != address(0));
61
62         rate = _rate;
63         wallet = _wallet;
64         token = _token;

```

```

65 }
66
67 // -----
68 // Crowdsale external interface
69 // -----
70
71 /**
72  * @dev fallback function ***DO NOT OVERRIDE***
73  */
74 function () external payable {
75     buyTokens(msg.sender);
76 }
77
78 /**
79  * @dev low level token purchase ***DO NOT OVERRIDE***
80  * @param _beneficiary Address performing the token purchase
81  */
82 function buyTokens(address _beneficiary) public payable {
83
84     uint256 weiAmount = msg.value;
85     _preValidatePurchase(_beneficiary, weiAmount);
86
87     // calculate token amount to be created
88     uint256 tokens = _getTokenAmount(weiAmount);
89
90     // update state
91     weiRaised = weiRaised.add(weiAmount);
92
93     _processPurchase(_beneficiary, tokens);
94     emit TokenPurchase(
95         msg.sender,
96         _beneficiary,
97         weiAmount,
98         tokens
99     );
100
101     _updatePurchasingState(_beneficiary, weiAmount);
102
103     _forwardFunds();
104     _postValidatePurchase(_beneficiary, weiAmount);
105 }
106
107 // -----
108 // Internal interface (extensible)
109 // -----
110
111 /**
112  * @dev Validation of an incoming purchase. Use require statements to revert state
113  *       when conditions are not met. Use super to concatenate validations.
114  * @param _beneficiary Address performing the token purchase
115  * @param _weiAmount Value in wei involved in the purchase
116  */
117 /*@CTK _preValidatePurchase
118   @tag assume_completion
119   @post _beneficiary != address(0)
120   @post _weiAmount != 0
121  */
122 function _preValidatePurchase(address _beneficiary, uint256 _weiAmount) internal {

```

```

122     require(_beneficiary != address(0));
123     require(_weiAmount != 0);
124 }
125
126 /**
127  * @dev Validation of an executed purchase. Observe state and use revert statements
128  *       to undo rollback when valid conditions are not met.
129  * @param _beneficiary Address performing the token purchase
130  * @param _weiAmount Value in wei involved in the purchase
131  */
132 function _postValidatePurchase(address _beneficiary, uint256 _weiAmount) internal {
133     // optional override
134 }
135
136 /**
137  * @dev Source of tokens. Override this method to modify the way in which the
138  *       crowdsale ultimately gets and sends its tokens.
139  * @param _beneficiary Address performing the token purchase
140  * @param _tokenAmount Number of tokens to be emitted
141  */
142 function _deliverTokens(address _beneficiary, uint256 _tokenAmount) internal {
143     token.transfer(_beneficiary, _tokenAmount);
144 }
145
146 /**
147  * @dev Executed when a purchase has been validated and is ready to be executed. Not
148  *       necessarily emits/sends tokens.
149  * @param _beneficiary Address receiving the tokens
150  * @param _tokenAmount Number of tokens to be purchased
151  */
152 function _processPurchase(address _beneficiary, uint256 _tokenAmount) internal {
153     _deliverTokens(_beneficiary, _tokenAmount);
154 }
155
156 /**
157  * @dev Override for extensions that require an internal state to check for validity
158  *       (current user contributions, etc.)
159  * @param _beneficiary Address receiving the tokens
160  * @param _weiAmount Value in wei involved in the purchase
161  */
162 function _updatePurchasingState(address _beneficiary, uint256 _weiAmount) internal {
163     // optional override
164 }
165
166 /**
167  * @dev Override to extend the way in which ether is converted to tokens.
168  * @param _weiAmount Value in wei to be converted into tokens
169  * @return Number of tokens that can be purchased with the specified _weiAmount
170  */
171 /*@CTK _getTokenAmount
172    @post __return == _weiAmount * rate
173 */
174 function _getTokenAmount(uint256 _weiAmount) internal view returns (uint256) {
175     return _weiAmount.mul(rate);
176 }
177
178 /**
179  * @dev Determines how ETH is stored/forwarded on purchases.

```

```

176  */
177  function _forwardFunds() internal {
178      wallet.transfer(msg.value);
179  }
180 }

```

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

```

1  pragma solidity ^0.4.21;
2
3  import "../math/SafeMath.sol";
4  import "../ownership/Ownable.sol";
5  import "../validation/TimedCrowdsale.sol";
6
7
8  /**
9   * @title FinalizableCrowdsale
10   * @dev Extension of Crowdsale where an owner can do extra work
11   * after finishing.
12   */
13  contract FinalizableCrowdsale is TimedCrowdsale, Ownable {
14      using SafeMath for uint256;
15
16      bool public isFinalized = false;
17
18      event Finalized();
19
20      /**
21       * @dev Must be called after crowdsale ends, to do some extra finalization
22       * work. Calls the contract's finalization function.
23       */
24      /*@CTK finalize
25       @tag assume_completion
26       @post !isFinalized
27       @post block.timestamp > closingTime
28       @post __post.isFinalized
29       */
30      function finalize() onlyOwner public {
31          require(!isFinalized);
32          require(hasClosed());
33
34          finalization();
35          emit Finalized();
36
37          isFinalized = true;
38      }
39
40      /**
41       * @dev Can be overridden to add finalization logic. The overriding function
42       * should call super.finalization() to ensure the chain of finalization is
43       * executed entirely.
44       */
45      function finalization() internal {
46      }
47
48  }

```

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

```

1  pragma solidity ^0.4.21;

```

```

2
3 import "../../math/SafeMath.sol";
4 import "../Crowdsale.sol";
5
6
7 /**
8  * @title TimedCrowdsale
9  * @dev Crowdsale accepting contributions only within a time frame.
10 */
11 contract TimedCrowdsale is Crowdsale {
12     using SafeMath for uint256;
13
14     uint256 public openingTime;
15     uint256 public closingTime;
16
17     /**
18      * @dev Reverts if not in crowdsale time range.
19      */
20     modifier onlyWhileOpen {
21         // solium-disable-next-line security/no-block-members
22         require(block.timestamp >= openingTime && block.timestamp <= closingTime);
23         _;
24     }
25
26     /**
27      * @dev Constructor, takes crowdsale opening and closing times.
28      * @param _openingTime Crowdsale opening time
29      * @param _closingTime Crowdsale closing time
30      */
31     /*@CTK TimedCrowdsale
32      @tag assume_completion
33      @post _openingTime >= block.timestamp
34      @post _closingTime >= _openingTime
35      @post __post.openingTime == _openingTime
36      @post __post.closingTime == _closingTime
37      */
38     function TimedCrowdsale(uint256 _openingTime, uint256 _closingTime) public {
39         // solium-disable-next-line security/no-block-members
40         require(_openingTime >= block.timestamp);
41         require(_closingTime >= _openingTime);
42
43         openingTime = _openingTime;
44         closingTime = _closingTime;
45     }
46
47     /**
48      * @dev Checks whether the period in which the crowdsale is open has already elapsed
49      *
50      * @return Whether crowdsale period has elapsed
51      */
52     /*@CTK hasClosed
53      @post __return == block.timestamp > closingTime
54      */
55     function hasClosed() public view returns (bool) {
56         // solium-disable-next-line security/no-block-members
57         return block.timestamp > closingTime;
58     }

```



```

59  /**
60   * @dev Extend parent behavior requiring to be within contributing period
61   * @param _beneficiary Token purchaser
62   * @param _weiAmount Amount of wei contributed
63   */
64  /*@CTK _preValidatePurchase
65   @tag assume_completion
66   @post block.timestamp >= openingTime && block.timestamp <= closingTime
67  */
68  function _preValidatePurchase(address _beneficiary, uint256 _weiAmount) internal
        onlyWhileOpen {
69      super._preValidatePurchase(_beneficiary, _weiAmount);
70  }
71
72  }

```

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

```

1  pragma solidity ^0.4.21;
2
3
4  /**
5   * @title SafeMath
6   * @dev Math operations with safety checks that throw on error
7   */
8  library SafeMath {
9
10     /**
11     * @dev Multiplies two numbers, throws on overflow.
12     */
13     /*@CTK SafeMath_mul
14      @post __reverted == __has_assertion_failure
15      @post __has_assertion_failure == __has_overflow
16      @post __reverted == false -> c == a * b
17      @post msg == msg__post
18      @post (a > 0 && (a * b / a != b)) == __has_assertion_failure
19      @post __addr_map == __addr_map__post
20     */
21     function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
22         if (a == 0) {
23             return 0;
24         }
25         c = a * b;
26         assert(c / a == b);
27         return c;
28     }
29
30     /**
31     * @dev Integer division of two numbers, truncating the quotient.
32     */
33     /*@CTK SafeMath_div
34      @post __reverted == __has_assertion_failure
35      @post __has_overflow == true -> __has_assertion_failure == true
36      @post __reverted == false -> __return == a / b
37      @post msg == msg__post
38      @post (b == 0) == __has_assertion_failure
39      @post __addr_map == __addr_map__post
40     */
41     function div(uint256 a, uint256 b) internal pure returns (uint256) {

```

```

42 // assert(b > 0); // Solidity automatically throws when dividing by 0
43 // uint256 c = a / b;
44 // assert(a == b * c + a % b); // There is no case in which this doesn't hold
45 return a / b;
46 }
47
48 /**
49 * @dev Subtracts two numbers, throws on overflow (i.e. if subtrahend is greater than
    minuend).
50 */
51 /*@CTK SafeMath_sub
52 @tag spec
53 @post __reverted == __has_assertion_failure
54 @post __has_overflow == true -> __has_assertion_failure == true
55 @post __reverted == false -> __return == a - b
56 @post msg == msg__post
57 @post (a < b) == __has_assertion_failure
58 @post __addr_map == __addr_map__post
59 */
60 function sub(uint256 a, uint256 b) internal pure returns (uint256) {
61     assert(b <= a);
62     return a - b;
63 }
64
65 /**
66 * @dev Adds two numbers, throws on overflow.
67 */
68 /*@CTK SafeMath_add
69 @tag spec
70 @post __reverted == __has_assertion_failure
71 @post __has_assertion_failure == __has_overflow
72 @post __reverted == false -> c == a + b
73 @post msg == msg__post
74 @post ((a + b < a) || (a + b < b)) == __has_assertion_failure
75 @post __addr_map == __addr_map__post
76 */
77 function add(uint256 a, uint256 b) internal pure returns (uint256 c) {
78     c = a + b;
79     assert(c >= a);
80     return c;
81 }
82 }

```

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

```

1 pragma solidity ^0.4.21;
2
3
4 /**
5 * @title Ownable
6 * @dev The Ownable contract has an owner address, and provides basic authorization
    control
7 * functions, this simplifies the implementation of "user permissions".
8 */
9 contract Ownable {
10     address public owner;
11
12
13     event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

```

```

14
15
16  /**
17   * @dev The Ownable constructor sets the original 'owner' of the contract to the
18     sender
19   * account.
20   */
21  /*@CTK OwnableConstructor
22   @post __post.owner == msg.sender
23  */
24  function Ownable() public {
25      owner = msg.sender;
26  }
27
28  /**
29   * @dev Throws if called by any account other than the owner.
30   */
31  modifier onlyOwner() {
32      require(msg.sender == owner);
33      _;
34  }
35
36  /**
37   * @dev Allows the current owner to transfer control of the contract to a newOwner.
38   * @param newOwner The address to transfer ownership to.
39   */
40  /*@CTK transferOwnership
41   @tag assume_completion
42   @post msg.sender == owner
43   @post __post.owner == newOwner
44  */
45  function transferOwnership(address newOwner) public onlyOwner {
46      require(newOwner != address(0));
47      emit OwnershipTransferred(owner, newOwner);
48      owner = newOwner;
49  }
50  }

```

## 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	
Post environment	52	}	
	53	balance: 0x0	
	54	}	
	55	}	
	56		
	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

```
1 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

```
1 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

```
1 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

```
1 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

```
1 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

```
1 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

```
1 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

```
1 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

```
22 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

```
22 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

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

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

## TIMESTAMP\_DEPENDENCY

Line 56 in File TimedCrowdsale.sol

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

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

## INSECURE\_COMPILER\_VERSION

Line 1 in File SafeMath.sol

```
1 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


```
1 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

```
23  /*@CTK ICOToken
24      @post !__reverted
25  */
```

Line 26-27 in File ICOToken.sol


```
26  constructor() public {
27  }
```

 The code meets the specification

## Formal Verification Request 2

### addMinter

 18, Apr 2019

 31.48 ms

Line 61-66 in File BasicCrowdsale.sol

```
61  /*@CTK addMinter
62      @tag assume_completion
63      @post owner == msg.sender
64      @post _minter != address(0)
65      @post __post.minters[_minter]
66  */
```

Line 67-71 in File BasicCrowdsale.sol


```
67  function addMinter(address _minter) public onlyOwner {
68      require(_minter != address(0));
69      minters[_minter] = true;
70      emit LogMinterAdded(_minter);
71  }
```

 The code meets the specification

## Formal Verification Request 3

### removeMinter

 18, Apr 2019

 22.6 ms

Line 73-77 in File BasicCrowdsale.sol



```
73  /*@CTK removeMinter
74      @tag assume_completion
75      @post owner == msg.sender
76      @post !__post.minters[_minter]
77  */
```

Line 78-81 in File BasicCrowdsale.sol

```
78  function removeMinter(address _minter) public onlyOwner {
79      minters[_minter] = false;
80      emit LogMinterRemoved(_minter);
81  }
```

✓ The code meets the specification

## Formal Verification Request 4

createFiatToken

📅 18, Apr 2019

🕒 285.9 ms

Line 83-86 in File BasicCrowdsale.sol

```
83  /*@CTK createFiatToken
84      @tag assume_completion
85      @post block.timestamp <= closingTime
86  */
```

Line 87-92 in File BasicCrowdsale.sol

```
87  function createFiatToken(address beneficiary, uint256 amount) public onlyMinter()
88      returns(bool){
89      require(!hasClosed());
90      mintFiatToken(beneficiary, amount);
91      emit LogFiatTokenMinted(msg.sender, beneficiary, amount);
92      return true;
93  }
```

✓ The code meets the specification

## Formal Verification Request 5

createBountyToken

📅 18, Apr 2019

🕒 283.67 ms

Line 108-111 in File BasicCrowdsale.sol

```
108 /*@CTK createBountyToken
109     @tag assume_completion
110     @post block.timestamp <= closingTime
111 */
```

Line 112-117 in File BasicCrowdsale.sol

```
112     function createBountyToken(address beneficiary, uint256 amount) public onlyMinter
113         () returns (bool) {
114             require(!hasClosed());
115             mintBountyToken(beneficiary, amount);
116             emit LogBountyTokenMinted(msg.sender, beneficiary, amount);
117             return true;
118         }
```

✓ The code meets the specification

## Formal Verification Request 6

### multiBeneficiariesValidation

📅 18, Apr 2019

🕒 49.0 ms

Line 136-142 in File BasicCrowdsale.sol

```
136     /*@CTK multiBeneficiariesValidation
137         @tag assume_completion
138         @post block.timestamp <= closingTime
139         @post beneficiaries.length > 0
140         @post amount.length > 0
141         @post beneficiaries.length == amount.length
142     */
```

Line 143-148 in File BasicCrowdsale.sol

```
143     function multiBeneficiariesValidation(address[] beneficiaries, uint256[] amount)
144         internal view {
145         require(!hasClosed());
146         require(beneficiaries.length > 0);
147         require(amount.length > 0);
148         require(beneficiaries.length == amount.length);
149     }
```

✓ The code meets the specification

## Formal Verification Request 7

### extendPrivateSaleDuration

📅 18, Apr 2019

🕒 289.16 ms

Line 153-158 in File BasicCrowdsale.sol

```
153     /*@CTK extendPrivateSaleDuration
154         @tag assume_completion
155         @post owner == msg.sender
156         @post __post.privateSaleEndDate == privateSaleEndDate + extentionInDays * 86400
157         @post __post.closingTime == closingTime + extentionInDays * 86400
158     */
```

Line 159-166 in File BasicCrowdsale.sol

```
159     function extendPrivateSaleDuration(uint256 extentionInDays) public onlyOwner
160         returns (bool) {
161             require(now <= privateSaleEndDate);
162             extentionInDays = extentionInDays.mul(1 days); // convert the days in
163                 milliseconds
164             privateSaleEndDate = privateSaleEndDate.add(extentionInDays);
165             closingTime = closingTime.add(extentionInDays);
166             emit LogPrivateSaleExtended(extentionInDays);
167             return true;
168         }
```

✓ The code meets the specification

## Formal Verification Request 8

extendMainSailDuration

📅 18, Apr 2019

🕒 326.6 ms

Line 171-179 in File BasicCrowdsale.sol

```
171     /*@CTK extendMainSailDuration
172         @tag assume_completion
173         @post now > privateSaleEndDate
174         @post block.timestamp <= closingTime
175         @post mainSaleDurantionExtentionLimit - extentionInDays >= 0
176         @post owner == msg.sender
177         @post __post.mainSaleDurantionExtentionLimit == mainSaleDurantionExtentionLimit
178             - extentionInDays
179         @post __post.closingTime == closingTime + extentionInDays * 86400
180     */
```

Line 180-191 in File BasicCrowdsale.sol

```
180     function extendMainSailDuration(uint256 extentionInDays) public onlyOwner returns
181         (bool) {
182         require(now > privateSaleEndDate);
183         require(!hasClosed());
184         require(mainSaleDurantionExtentionLimit.sub(extentionInDays) >= 0);
185
186         uint256 extention = extentionInDays.mul(1 days); // convert the days in
187             milliseconds
188         mainSaleDurantionExtentionLimit = mainSaleDurantionExtentionLimit.sub(
189             extentionInDays); // subtract days from the limit
190         closingTime = closingTime.add(extention);
191
192         emit LogMainSaleExtended(extentionInDays);
193         return true;
194     }
```

✓ The code meets the specification

## Formal Verification Request 9

changeRate

📅 18, Apr 2019

🕒 54.16 ms

Line 193-199 in File BasicCrowdsale.sol

```
193  /*@CTK changeRate
194     @tag assume_completion
195     @post owner == msg.sender
196     @post block.timestamp <= closingTime
197     @post _newRate != 0
198     @post __post.rate == _newRate
199  */
```

Line 200-206 in File BasicCrowdsale.sol

```
200  function changeRate(uint _newRate) public onlyOwner returns (bool) {
201      require(!hasClosed());
202      require(_newRate != 0);
203      rate = _newRate;
204      emit LogRateChanged(_newRate);
205      return true;
206  }
```

✅ The code meets the specification

## Formal Verification Request 10

WhitelistedBasicCrowdsale

📅 18, Apr 2019

🕒 4.67 ms

Line 10-13 in File WhitelistedBasicCrowdsale.sol

```
10  /*@CTK WhitelistedBasicCrowdsale
11     @tag assume_completion
12     @post !__reverted
13  */
```

Line 14-18 in File WhitelistedBasicCrowdsale.sol


```
14  constructor(uint256 _rate, address _wallet, address _token, uint256 _openingTime,
15             uint256 _closingTime)
16  // BasicCrowdsale(_rate, _wallet, ERC20(_token), _openingTime, _closingTime)
17  // MultipleWhitelistedCrowdsale()
18  public {
19  }
```

✅ The code meets the specification

## Formal Verification Request 11

### addWhitelistManager

 18, Apr 2019

 32.94 ms

Line 36-41 in File MultipleWhitelistedCrowdsale.sol

```
36  /*@CTK addWhitelistManager
37  @tag assume_completion
38  @post owner == msg.sender
39  @post _manager != address(0)
40  @post __post.whitelistManagers[_manager]
41  */
```

Line 42-45 in File MultipleWhitelistedCrowdsale.sol


```
42  function addWhitelistManager(address _manager) public onlyOwner {
43      require(_manager != address(0));
44      whitelistManagers[_manager] = true;
45  }
```

 The code meets the specification

## Formal Verification Request 12

### removeWhitelistManager

 18, Apr 2019

 22.43 ms

Line 50-54 in File MultipleWhitelistedCrowdsale.sol

```
50  /*@CTK removeWhitelistManager
51  @tag assume_completion
52  @post owner == msg.sender
53  @post !__post.whitelistManagers[_manager]
54  */
```

Line 55-57 in File MultipleWhitelistedCrowdsale.sol


```
55  function removeWhitelistManager(address _manager) public onlyOwner {
56      whitelistManagers[_manager] = false;
57  }
```

 The code meets the specification

## Formal Verification Request 13

### addToWhitelist

 18, Apr 2019

 23.48 ms

Line 63-67 in File MultipleWhitelistedCrowdsale.sol

```

63  /*@CTK addToWhitelist
64      @tag assume_completion
65      @post whitelistManagers[msg.sender]
66      @post __post.whitelist[_beneficiary]
67  */

```

Line 68-70 in File MultipleWhitelistedCrowdsale.sol

```

68  function addToWhitelist(address _beneficiary) external onlyWhitelistManager() {
69      whitelist[_beneficiary] = true;
70  }

```

✓ The code meets the specification

## Formal Verification Request 14

removeFromWhitelist

📅 18, Apr 2019

🕒 23.56 ms

Line 86-90 in File MultipleWhitelistedCrowdsale.sol

```

86  /*@CTK removeFromWhitelist
87      @tag assume_completion
88      @post whitelistManagers[msg.sender]
89      @post !__post.whitelist[_beneficiary]
90  */

```

Line 91-93 in File MultipleWhitelistedCrowdsale.sol

```

91  function removeFromWhitelist(address _beneficiary) external onlyWhitelistManager() {
92      whitelist[_beneficiary] = false;
93  }

```

✓ The code meets the specification

## Formal Verification Request 15

transferFrom

📅 18, Apr 2019

🕒 156.65 ms

Line 25-32 in File StandardToken.sol

```

25  /*@CTK transferFrom
26      @tag assume_completion
27      @pre _from != _to
28      @post __return == true
29      @post __post.balances[_to] == balances[_to] + _value
30      @post __post.balances[_from] == balances[_from] - _value
31      @post __has_overflow == false
32  */

```

Line 33-43 in File StandardToken.sol

```

33  function transferFrom(address _from, address _to, uint256 _value) public returns (
      bool) {
34      require(_to != address(0));
35      require(_value <= balances[_from]);
36      require(_value <= allowed[_from][msg.sender]);
37
38      balances[_from] = balances[_from].sub(_value);
39      balances[_to] = balances[_to].add(_value);
40      allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
41      emit Transfer(_from, _to, _value);
42      return true;
43  }

```

✓ The code meets the specification

## Formal Verification Request 16

approve\_success

📅 18, Apr 2019

🕒 9.8 ms

Line 55-58 in File StandardToken.sol

```

55  /*@CTK approve_success
56      @post _value == 0 -> __reverted == false
57      @post allowed[msg.sender][_spender] == 0 -> __reverted == false
58  */

```

Line 63-67 in File StandardToken.sol

```

63  function approve(address _spender, uint256 _value) public returns (bool) {
64      allowed[msg.sender][_spender] = _value;
65      emit Approval(msg.sender, _spender, _value);
66      return true;
67  }

```

✓ The code meets the specification

## Formal Verification Request 17

approve

📅 18, Apr 2019

🕒 1.35 ms

Line 59-62 in File StandardToken.sol

```

59  /*@CTK approve
60      @tag assume_completion
61      @post __post.allowed[msg.sender][_spender] == _value
62  */

```

Line 63-67 in File StandardToken.sol

```
63 function approve(address _spender, uint256 _value) public returns (bool) {
64     allowed[msg.sender][_spender] = _value;
65     emit Approval(msg.sender, _spender, _value);
66     return true;
67 }
```

✓ The code meets the specification

## Formal Verification Request 18

get\_allowance

📅 18, Apr 2019

🕒 6.51 ms

Line 75-79 in File StandardToken.sol

```
75 /*@CTK get_allowance
76     @post __reverted == false
77     @post __return == allowed[_owner][_spender]
78     @post this == __post
79 */
```

Line 80-82 in File StandardToken.sol

```
80 function allowance(address _owner, address _spender) public view returns (uint256) {
81     return allowed[_owner][_spender];
82 }
```

✓ The code meets the specification

## Formal Verification Request 19

increaseApproval

📅 18, Apr 2019

🕒 20.06 ms

Line 94-97 in File StandardToken.sol

```
94 /*@CTK increaseApproval
95     @tag assume_completion
96     @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] +
97         _addedValue
98 */
```

Line 98-102 in File StandardToken.sol

```
98 function increaseApproval(address _spender, uint _addedValue) public returns (bool)
99 {
100     allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_addedValue);
101     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
102     return true;
103 }
```


✓ 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

```
114  /*@CTK "decreaseApproval correctness case 1"
115    @pre allowed[msg.sender][_spender] < _subtractedValue
116    @post __post.allowed[msg.sender][_spender] == 0
117    @post __return == true
118    @post (!__has_assertion_failure)
119  */
```

Line 126-135 in File StandardToken.sol


```
126  function decreaseApproval(address _spender, uint _subtractedValue) public returns (
127    bool) {
127    uint oldValue = allowed[msg.sender][_spender];
128    if (_subtractedValue > oldValue) {
129      allowed[msg.sender][_spender] = 0;
130    } else {
131      allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
132    }
133    emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
134    return true;
135  }
```

 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

```
120  /*@CTK "decreaseApproval correctness case 2"
121    @pre allowed[msg.sender][_spender] >= _subtractedValue
122    @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] -
123          _subtractedValue
123    @post __return == true
124    @post (!__has_assertion_failure)
125  */
```

Line 126-135 in File StandardToken.sol

```
126  function decreaseApproval(address _spender, uint _subtractedValue) public returns (
127    bool) {
127    uint oldValue = allowed[msg.sender][_spender];
128    if (_subtractedValue > oldValue) {
129      allowed[msg.sender][_spender] = 0;
130    } else {
```

```

131     allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
132 }
133 emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
134 return true;
135 }

```

✓ The code meets the specification

## Formal Verification Request 22

mint

📅 18, Apr 2019

🕒 166.11 ms

Line 31-35 in File MintableToken.sol

```

31  /*@CTK mint
32     @tag assume_completion
33     @post __post.totalSupply_ == totalSupply_ + _amount
34     @post __post.balances[_to] == balances[_to] + _amount
35  */

```

Line 36-42 in File MintableToken.sol

```

36  function mint(address _to, uint256 _amount) onlyOwner canMint public returns (bool)
37  {
38      totalSupply_ = totalSupply_.add(_amount);
39      balances[_to] = balances[_to].add(_amount);
40      emit Mint(_to, _amount);
41      emit Transfer(address(0), _to, _amount);
42      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

```

48  /*@CTK finishMinting
49     @tag assume_completion
50     @post owner == msg.sender
51     @post !mintingFinished
52     @post __post.mintingFinished
53  */

```

Line 54-58 in File MintableToken.sol

```
54 function finishMinting() onlyOwner canMint public returns (bool) {  
55     mintingFinished = true;  
56     emit MintFinished();  
57     return true;  
58 }
```

✓ The code meets the specification

## Formal Verification Request 24

totalSupply

📅 18, Apr 2019

🕒 6.6 ms

Line 22-24 in File BasicToken.sol

```
22 /*@CTK totalSupply  
23     @post __return == totalSupply_  
24 */
```

Line 25-27 in File BasicToken.sol

```
25 function totalSupply() public view returns (uint256) {  
26     return totalSupply_;  
27 }
```

✓ The code meets the specification

## Formal Verification Request 25

transfer\_success

📅 18, Apr 2019

🕒 55.06 ms

Line 34-40 in File BasicToken.sol

```
34 /*@CTK transfer_success  
35     @pre _to != address(0)  
36     @pre balances[msg.sender] >= _value  
37     @pre __reverted == false  
38     @post __reverted == false  
39     @post __return == true  
40 */
```

Line 53-61 in File BasicToken.sol

```
53 function transfer(address _to, uint256 _value) public returns (bool) {  
54     require(_to != address(0));  
55     require(_value <= balances[msg.sender]);  
56  
57     balances[msg.sender] = balances[msg.sender].sub(_value);  
58     balances[_to] = balances[_to].add(_value);  
59     emit Transfer(msg.sender, _to, _value);  
60     return true;  
61 }
```

✓ The code meets the specification

## Formal Verification Request 26

transfer\_conditions

📅 18, Apr 2019

🕒 118.28 ms

Line 41-46 in File BasicToken.sol

```
41  /*@CTK transfer_conditions
42     @tag assume_completion
43     @pre _to != msg.sender
44     @post __post.balances[_to] == balances[_to] + _value
45     @post __post.balances[msg.sender] == balances[msg.sender] - _value
46  */
```

Line 53-61 in File BasicToken.sol

```
53  function transfer(address _to, uint256 _value) public returns (bool) {
54      require(_to != address(0));
55      require(_value <= balances[msg.sender]);
56
57      balances[msg.sender] = balances[msg.sender].sub(_value);
58      balances[_to] = balances[_to].add(_value);
59      emit Transfer(msg.sender, _to, _value);
60      return true;
61  }
```

✓ 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

```
47  /*@CTK transfer_same_address
48     @tag assume_completion
49     @tag no_overflow
50     @pre _to == msg.sender
51     @post this == __post
52  */
```

Line 53-61 in File BasicToken.sol

```
53  function transfer(address _to, uint256 _value) public returns (bool) {
54      require(_to != address(0));
55      require(_value <= balances[msg.sender]);
56
57      balances[msg.sender] = balances[msg.sender].sub(_value);
```


```
58     balances[_to] = balances[_to].add(_value);
59     emit Transfer(msg.sender, _to, _value);
60     return true;
61 }
```

✓ The code meets the specification

## Formal Verification Request 28

balanceOf

 18, Apr 2019

 5.9 ms

Line 68-71 in File BasicToken.sol

```
68     /*@CTK balanceOf
69     @post __reverted == false
70     @post __return == balances[_owner]
71     */
```

Line 72-74 in File BasicToken.sol


```
72     function balanceOf(address _owner) public view returns (uint256) {
73         return balances[_owner];
74     }
```

✓ The code meets the specification

## Formal Verification Request 29

\_burn

 18, Apr 2019

 105.52 ms

Line 18-23 in File BurnableToken.sol

```
18     /*@CTK _burn
19     @tag assume_completion
20     @post _value <= balances[msg.sender]
21     @post __post.balances[msg.sender] == balances[msg.sender] - _value
22     @post __post.totalSupply_ == totalSupply_ - _value
23     */
```

Line 24-26 in File BurnableToken.sol

```
24     function burn(uint256 _value) public {
25         _burn(msg.sender, _value);
26     }
```

✓ The code meets the specification

## Formal Verification Request 30

\_burn

📅 18, Apr 2019

🕒 40.07 ms

Line 28-33 in File BurnableToken.sol

```
28  /*@CTK _burn
29      @tag assume_completion
30      @post _value <= balances[_who]
31      @post __post.balances[_who] == balances[_who] - _value
32      @post __post.totalSupply_ == totalSupply_ - _value
33  */
```

Line 34-43 in File BurnableToken.sol

```
34  function _burn(address _who, uint256 _value) internal {
35      require(_value <= balances[_who]);
36      // no need to require value <= totalSupply, since that would imply the
37      // sender's balance is greater than the totalSupply, which *should* be an
38      // assertion failure
39
40      balances[_who] = balances[_who].sub(_value);
41      totalSupply_ = totalSupply_.sub(_value);
42      emit Burn(_who, _value);
43      emit Transfer(_who, address(0), _value);
44  }
```

✅ The code meets the specification

## Formal Verification Request 31

pause

📅 18, Apr 2019

🕒 29.81 ms

Line 37-41 in File Pausable.sol

```
37  /*@CTK pause
38      @tag assume_completion
39      @post owner == msg.sender
40      @post __post.paused == true
41  */
```

Line 42-45 in File Pausable.sol


```
42  function pause() onlyOwner whenNotPaused public {
43      paused = true;
44      emit Pause();
45  }
```

✅ The code meets the specification

## Formal Verification Request 32

unpause

 18, Apr 2019

 26.07 ms

Line 50-54 in File Pausable.sol

```
50  /*@CTK unpause
51      @tag assume_completion
52      @post owner == msg.sender
53      @post __post.paused == false
54  */
```

Line 55-58 in File Pausable.sol


```
55  function unpause() onlyOwner whenPaused public {
56      paused = false;
57      emit Unpause();
58  }
```

 The code meets the specification

## Formal Verification Request 33

Crowdsale

 18, Apr 2019

 38.96 ms

Line 48-56 in File Crowdsale.sol

```
48  /*@CTK Crowdsale
49      @tag assume_completion
50      @post _rate > 0
51      @post _wallet != address(0)
52      @post _token != address(0)
53      @post __post.rate == _rate
54      @post __post.wallet == _wallet
55      @post __post.token == _token
56  */
```

Line 57-65 in File Crowdsale.sol


```
57  function Crowdsale(uint256 _rate, address _wallet, ERC20 _token) public {
58      require(_rate > 0);
59      require(_wallet != address(0));
60      require(_token != address(0));
61
62      rate = _rate;
63      wallet = _wallet;
64      token = _token;
65  }
```

 The code meets the specification

## Formal Verification Request 34

`_preValidatePurchase`

 18, Apr 2019

 19.37 ms

Line 116-120 in File Crowdsale.sol

```
116  /*@CTK _preValidatePurchase
117     @tag assume_completion
118     @post _beneficiary != address(0)
119     @post _weiAmount != 0
120  */
```

Line 121-124 in File Crowdsale.sol


```
121  function _preValidatePurchase(address _beneficiary, uint256 _weiAmount) internal {
122      require(_beneficiary != address(0));
123      require(_weiAmount != 0);
124  }
```

 The code meets the specification

## Formal Verification Request 35

`_getTokenAmount`

 18, Apr 2019

 65.92 ms

Line 167-169 in File Crowdsale.sol

```
167  /*@CTK _getTokenAmount
168     @post __return == _weiAmount * rate
169  */
```

Line 170-172 in File Crowdsale.sol


```
170  function _getTokenAmount(uint256 _weiAmount) internal view returns (uint256) {
171      return _weiAmount.mul(rate);
172  }
```

 The code meets the specification

## Formal Verification Request 36

`finalize`

 18, Apr 2019

 78.23 ms

Line 24-29 in File FinalizableCrowdsale.sol



```
24  /*@CTK finalize
25     @tag assume_completion
26     @post !isFinalized
27     @post block.timestamp > closingTime
28     @post __post.isFinalized
29  */
```

Line 30-38 in File FinalizableCrowdsale.sol

```
30  function finalize() onlyOwner public {
31      require(!isFinalized);
32      require(hasClosed());
33
34      finalization();
35      emit Finalized();
36
37      isFinalized = true;
38  }
```

✓ The code meets the specification

## Formal Verification Request 37

TimedCrowdsale

📅 18, Apr 2019

🕒 33.73 ms

Line 31-37 in File TimedCrowdsale.sol

```
31  /*@CTK TimedCrowdsale
32     @tag assume_completion
33     @post _openingTime >= block.timestamp
34     @post _closingTime >= _openingTime
35     @post __post.openingTime == _openingTime
36     @post __post.closingTime == _closingTime
37  */
```

Line 38-45 in File TimedCrowdsale.sol

```
38  function TimedCrowdsale(uint256 _openingTime, uint256 _closingTime) public {
39      // solium-disable-next-line security/no-block-members
40      require(_openingTime >= block.timestamp);
41      require(_closingTime >= _openingTime);
42
43      openingTime = _openingTime;
44      closingTime = _closingTime;
45  }
```

✓ The code meets the specification

## Formal Verification Request 38

hasClosed

📅 18, Apr 2019

🕒 7.07 ms

Line 51-53 in File TimedCrowdsale.sol

```
51  /*@CTK hasClosed
52    @post __return == block.timestamp > closingTime
53  */
```

Line 54-57 in File TimedCrowdsale.sol


```
54  function hasClosed() public view returns (bool) {
55    // solium-disable-next-line security/no-block-members
56    return block.timestamp > closingTime;
57  }
```

✓ The code meets the specification

## Formal Verification Request 39

`_preValidatePurchase`

 18, Apr 2019

 60.02 ms

Line 64-67 in File TimedCrowdsale.sol

```
64  /*@CTK _preValidatePurchase
65    @tag assume_completion
66    @post block.timestamp >= openingTime && block.timestamp <= closingTime
67  */
```

Line 68-70 in File TimedCrowdsale.sol


```
68  function _preValidatePurchase(address _beneficiary, uint256 _weiAmount) internal
    onlyWhileOpen {
69    super._preValidatePurchase(_beneficiary, _weiAmount);
70  }
```

✓ The code meets the specification

## Formal Verification Request 40

`SafeMath.mul`

 18, Apr 2019

 364.95 ms

Line 13-20 in File SafeMath.sol

```
13  /*@CTK SafeMath_mul
14    @post __reverted == __has_assertion_failure
15    @post __has_assertion_failure == __has_overflow
16    @post __reverted == false -> c == a * b
17    @post msg == msg__post
18    @post (a > 0 && (a * b / a != b)) == __has_assertion_failure
19    @post __addr_map == __addr_map__post
20  */
```

Line 21-28 in File SafeMath.sol

```
21 function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
22     if (a == 0) {
23         return 0;
24     }
25     c = a * b;
26     assert(c / a == b);
27     return c;
28 }
```

✓ The code meets the specification

## Formal Verification Request 41

SafeMath\_div

📅 18, Apr 2019

🕒 7.44 ms

Line 33-40 in File SafeMath.sol

```
33 /*@CTK SafeMath_div
34     @post __reverted == __has_assertion_failure
35     @post __has_overflow == true -> __has_assertion_failure == true
36     @post __reverted == false -> __return == a / b
37     @post msg == msg__post
38     @post (b == 0) == __has_assertion_failure
39     @post __addr_map == __addr_map__post
40 */
```

Line 41-46 in File SafeMath.sol

```
41 function div(uint256 a, uint256 b) internal pure returns (uint256) {
42     // assert(b > 0); // Solidity automatically throws when dividing by 0
43     // uint256 c = a / b;
44     // assert(a == b * c + a % b); // There is no case in which this doesn't hold
45     return a / b;
46 }
```

✓ The code meets the specification

## Formal Verification Request 42

SafeMath\_sub

📅 18, Apr 2019

🕒 14.27 ms

Line 51-59 in File SafeMath.sol

```
51 /*@CTK SafeMath_sub
52     @tag spec
53     @post __reverted == __has_assertion_failure
54     @post __has_overflow == true -> __has_assertion_failure == true
```

```

55     @post __reverted == false -> __return == a - b
56     @post msg == msg__post
57     @post (a < b) == __has_assertion_failure
58     @post __addr_map == __addr_map__post
59 */

```

Line 60-63 in File SafeMath.sol

```

60     function sub(uint256 a, uint256 b) internal pure returns (uint256) {
61         assert(b <= a);
62         return a - b;
63     }

```

✓ The code meets the specification

## Formal Verification Request 43

SafeMath.add

📅 18, Apr 2019

🕒 19.29 ms

Line 68-76 in File SafeMath.sol

```

68     /*@CTK SafeMath_add
69     @tag spec
70     @post __reverted == __has_assertion_failure
71     @post __has_assertion_failure == __has_overflow
72     @post __reverted == false -> c == a + b
73     @post msg == msg__post
74     @post ((a + b < a) || (a + b < b)) == __has_assertion_failure
75     @post __addr_map == __addr_map__post
76 */

```

Line 77-81 in File SafeMath.sol

```

77     function add(uint256 a, uint256 b) internal pure returns (uint256 c) {
78         c = a + b;
79         assert(c >= a);
80         return c;
81     }

```

✓ The code meets the specification

## Formal Verification Request 44

OwnableConstructor

📅 18, Apr 2019

🕒 5.87 ms

Line 20-22 in File Ownable.sol

```

20     /*@CTK OwnableConstructor
21     @post __post.owner == msg.sender
22 */

```

Line 23-25 in File Ownable.sol

```
23  function Ownable() public {  
24      owner = msg.sender;  
25  }
```

✓ The code meets the specification

## Formal Verification Request 45

transferOwnership

📅 18, Apr 2019

🕒 26.52 ms

Line 39-43 in File Ownable.sol

```
39  /*@CTK transferOwnership  
40      @tag assume_completion  
41      @post msg.sender == owner  
42      @post __post.owner == newOwner  
43  */
```

Line 44-48 in File Ownable.sol

```
44  function transferOwnership(address newOwner) public onlyOwner {  
45      require(newOwner != address(0));  
46      emit OwnershipTransferred(owner, newOwner);  
47      owner = newOwner;  
48  }
```

✓ The code meets the specification